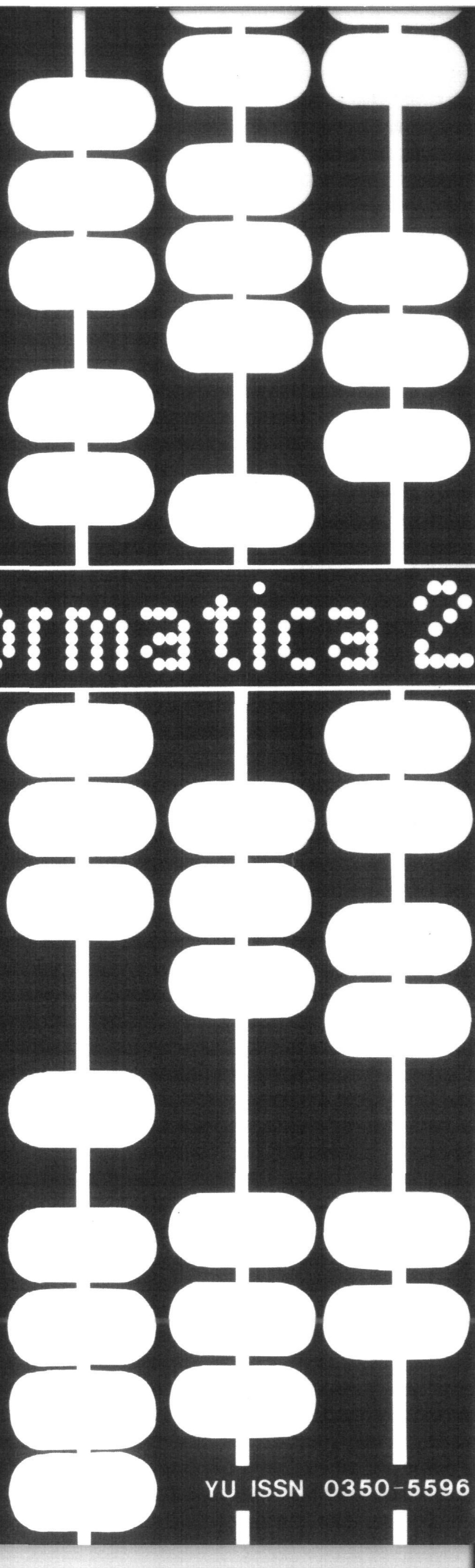
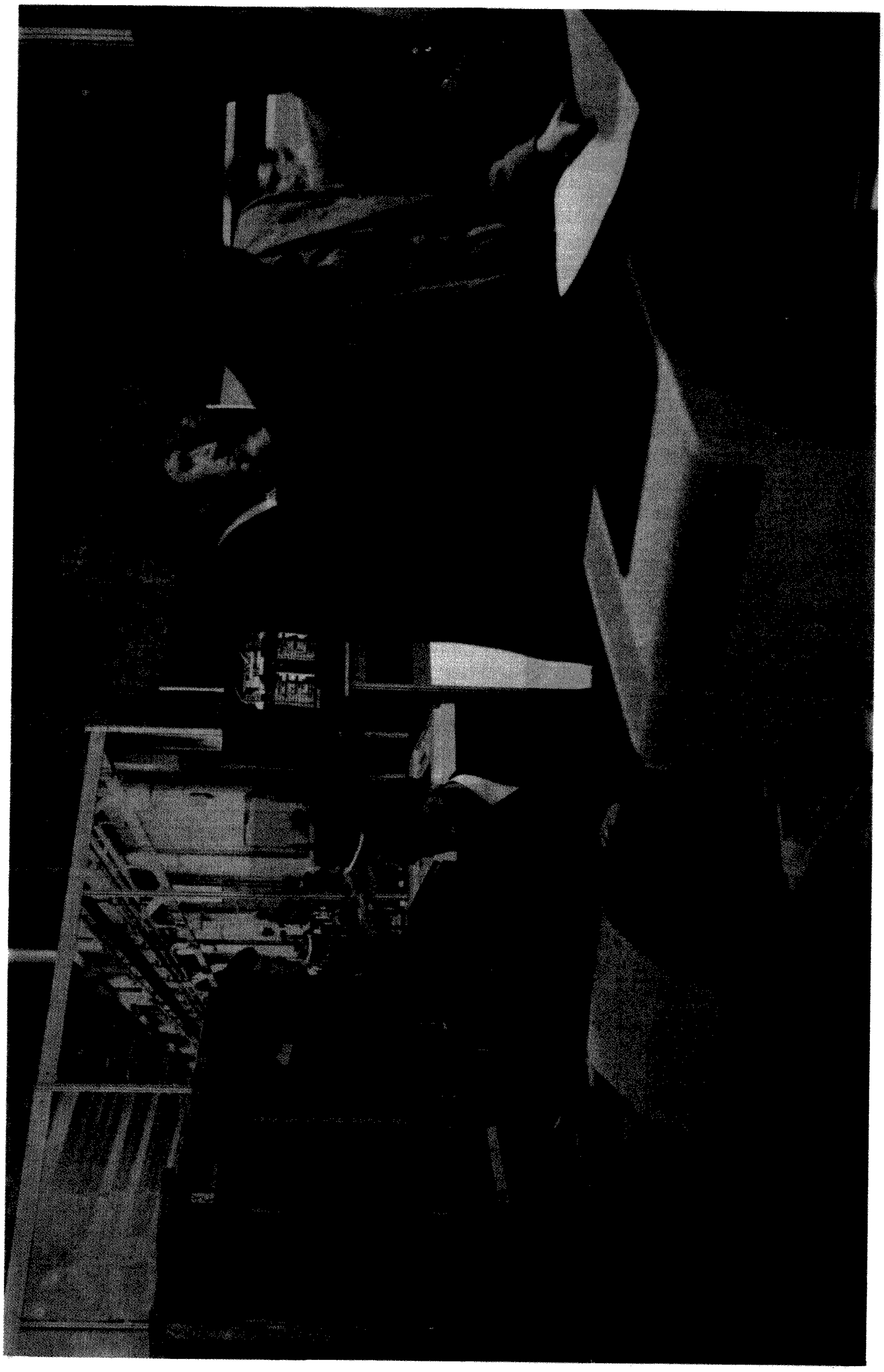


82 informatica 2

YU ISSN 0350-5596



®

računalniški sistemi delta

informatica

Published by INFORMATIKA, Slovene Society for Informatics, 61000 Ljubljana, Parmova 41, Yugoslavia

JOURNAL OF COMPUTING AND INFORMATICS

YU ISSN 0350–5596

VOLUME 6, 1982 – No. 2

Headquarters: Informatica, Parmova 41, 61000 Ljubljana, Phone: (061) 312-988, Telex: 31366 Delta

Annual subscription rate for abroad is US $ 22 for companies, and US $ 7,5 for individuals.

Opinions expressed in the contributions are not necessarily shared by the Editorial Board.

Printed by: Tiskarna KRESIJA, Ljubljana

DESIGN: Rasto Kirn

CONTENTS

informatica

ČASOPIS ZA TEHNOLOGIJO RAČUNALNIŠTVA IN PROBLEME INFORMATIKE
ČASOPIS ZA RAČUNARSKU TEHNOLOGIJU I PROBLEME INFORMATIKE
SPISANIE ZA TEHNOLOGIJA NA SMETANJETO I PROBLEMI OD OBLASTA NA INFORMATIKATA

YU ISSN 0350–5596

LETNIK 6, 1982 – št. 2

Časopis izdaja Slovensko društvo INFORMATIKA, 61000 Ljubljana, Parmova 41, Jugoslavija

Uredništvo in uprava: Informatica, Parmova 41, 61000 Ljubljana, telefon (061) 312-988, teleks: 31366 YU DELTA

Letna naročnina za delovne organizacije je 500,00 din, za redne člane 200,00 din, za študente 100,00/50,00 din, posamezne številke 100,00 din
Žiro račun št.: 50101-678-51841
Stališče uredništva se lahko razlikuje od mnenja avtorjev.

Pri financiranju revije sodeluje tudi Raziskovalna skupnost Slovenije.

Na podlagi mnenja Republiškega sekretariata za prosveto in kulturo št. 4210-44/79 z dne 1.2.1979, je časopis oproščen temeljnega davka od prometa proizvodov.

Tisk: Tiskarna KRESIJA, Ljubljana

Grafična oprema: Rasto Kirn

V S E B I N A

INFORMATICA 2/1982

STEPS TOWARDS NATURAL COMPUTATION

PIERRE MARIE LAWOREL

CNRS, INSERM UNIT 94, LYON, FRANCE

UDK: 159.953:681.3

This article deals with problems of natural computation, i.e. processes in human brain indicating some avenues of research in neuropsychology and neurolinguistics where inspiration may be found for future advances in computational science. The article discovers problems of human software and brain wetware, cooperative processes in the brain, and computer simulation of activity in the brain. The new approach presented in this article enables neuropsychologists to test cognitive models of brain processing that are thought to match anatomical and psysiological discoveries. Computer simulation thus appears to be an efficient method to verify the relevance and the robustness of theoretical models in the neurosciences.

(The Editor)

Koraki v smeri naravnega računanja. Ta članek obravnava problematiko naravnega računanja, tj. procesov v človekovih možganih s posebnim ozirom na raziskave v nevropsihologiji in nevrolingvistiki, kjer je moč pričakovati ideje za prihodnji razvoj računalniških znanosti. Članek odkriva probleme človekove programske opreme in možganske "vlažnosti", kooperativnih procesov v možganih ter računalniške simulacije možganskih aktivnosti. Nov pristop, ki je v članku prikazan, omogoča nevropsihologu preizkušanje kognitivnih modelov možganskega procesiranja, ki se ujemajo z anatomskimi in fiziološkimi odkritji. Računalniška simulacija se tako pokaže kot učinkovito sredstvo pri preverjanju tehtnosti in obstojnosti teoretičnih modelov v nevroznanosti.

In 1961, Marvin Minsky published a paper entitled "Steps towards A.I. (Artificial Intelligence)". His influence and that of his students or colleagues (Schank, Winograd, Newell, Colby, Bobrow, Woods) were so great that the crowd of computer scientists quickly realized that a new science was born. Cybernetics, the mother discipline concerned with machine processes as well as brain processes and other natural systems dynamics, was considered to be too old and too general. Cognitive psychology, brain theory, social systems dynamics, and AI gradually developed as diversified fields. But processing models remained a central issue. In computer programs for the comprehension of language, for instance, debates went on about which top-down or bottom-up algorithms were more efficient, or what types of data were to be processed (acoustic data, syntactic structures, or semantic-pragmatic representations). Yet, one sweeping statement frequently made by AI specialists kept disturbing people who worked in neuropsychology and in linguistics. Almost everyone in the computer sciences claimed that psychological reality as well as neurophysiological reality could be forgotten when one conceived a system that would handle language, or control a robot, or solve heuristic problems. The only questions to be asked were : Does it do the trick? Is it both economical and efficient? What is the percentage of error? Very soon, however, honest scientists had to admit that while AI systems were at the origin of very useful methodological progress in program designing and in computational theory, they could not successfully replace man for verbal communication. The only systems that seemed to yield satisfactory results were extremely sophisticated and costly, and could only handle relatively small samples of natural language (e.g. for English : Thorne, Bratley and Dewar (1968), Woods (1970), Rieger (1975), Riesbeck (1975), Winograd (1972), Schank (1973, 1975), Simmons (1973), Small (1980)). Besides, these systems did not solve many morpho-syntactic ambiguities, and anaphora or ellipsis caused misinterpretations.

For all these reasons, foremost AI experts started meeting cognitive scientists, linguists, and neurologists in the 1970's, in order to renew the partly unsuccessful approaches which they had favored before. After several individual experiments, like HARPY (Newell, 1978), HWIM (Woods et al., 1976), HEARSAY I and II (Erman et al., 1980), PARSIFAL (Marcus, 1978) and PSI-KLONE (Bobrow and Webber (1980), where some hypotheses about natural language comprehension were tested, two conferences were organized by Arbib, Caplan and Marshall in 1980-1981 at the University of Massachusetts (Arbib et al., 1982).

The purpose was explicitely to formulate some neural models of language processing and encourage participants to put them to the test with computer simulation and renewed testing in clinical neuropsychology.

Following the same epistemological track, I shall now try to indicate some avenues of research in neuropsychology and neurolinguistics where inspiration may be found for future advances in computational science. I suggest to call this new style of informatics N.C. (Abbreviated for Natural Computation).

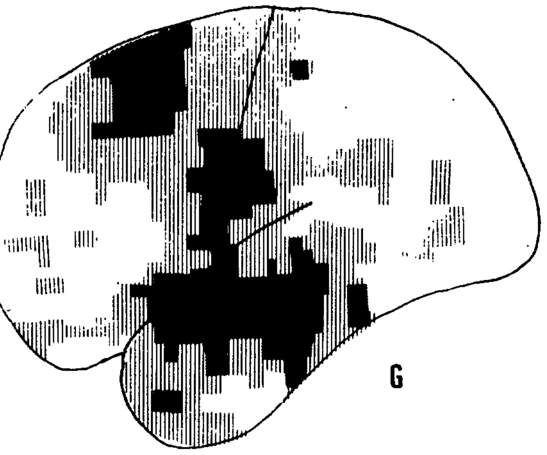
HUMAN SOFTWARE AND BRAIN WETWARE.

Thanks to many successful discoveries of the last twenty years, brain sciences have much to offer to theoreticians of informatics. The biophysical study of the central nervous system of animals and the now powerfully instrumented observations of lesioned human brains have enabled neurophysiologists and neuropsychologists to formulate models of how the 14 billion neuron brain processes

- representations of the physical world,
- representations of real or imaginary social, cultural, and artistic worlds,
- schemas of elementary motor acts,
- goals, plans and strategies for complex behavior,
- interactive control between representations, schemas, and overall scheming,
- second or third order representations of representations, called voluntary or involuntary language and sign communication (See Lavorel, 1983).

It must first be admitted that the brain does not work like a classical computer with stored data and sets of coded rules to be applied in a given order corresponding to algorithms or sets of programs. The nature of neural activity is totally unrelated to the nature of intellectual objects being processed. Neurons or neuron modules just keep being active or inhibited (Szentâgothai and Arbib, 1974). Basically, the biochemical and electro-magnetic phenomena of neural tissue are the same in the most stupid amphibians and in the most famous Nobel prize winners (Eccles, 1977). What is different in animal and in man is first the number of neural processors and their ability to memorize and forget. Secondly, the number of connections, their architecture and their plasticity determine self-organization in the system, and a gradual diversification or a redundancy of its functional subsystems. Thirdly, the order of complexity of the simultaneous propagations of rhythmic signals in the cortex, which is a consequence of the two properties already distinguished, ensures a style of computation which can transfer and transform information in a functionally efficient manner, in spite of conflicting stimulations, of minor breakdowns, or of possibly erroneous knowledge. Thus, the human brain can be hyperactive in order to adapt itself instantaneously to intricate aspects of novel stimulation, or it can react more economically and more automatically to recurring situations that it has learned to identify and categorize.

Some neurophysiologists and some electrical engineers have tried to devise networks that would possibly approach the computational properties of mammal neural modules (e.g. Reiss (1962), Mc Gregor and Lewis (1977), Anderson, Ritz, Silverstein and Jones (1977), de Callatay (1981), Wood (1982)). However, the fact that they do research for electronic computers limits their attempts essentially to the study of multiplexing, of timing control, of noise filtering, and of cellular automata with fuzzy inputs. What they show has in fact already been predicted by mathematicians like Winograd and Cowan (1961), Nilsson (1965), Amari and Arbib (1977), Arbib and Manes (1975) or Dalenoort (1981). It has also been observed at a microscopic level by anatomists and physiologists. Mountcastle (1976) for example showed how skin sensations are distributed in the monkey's brain in an



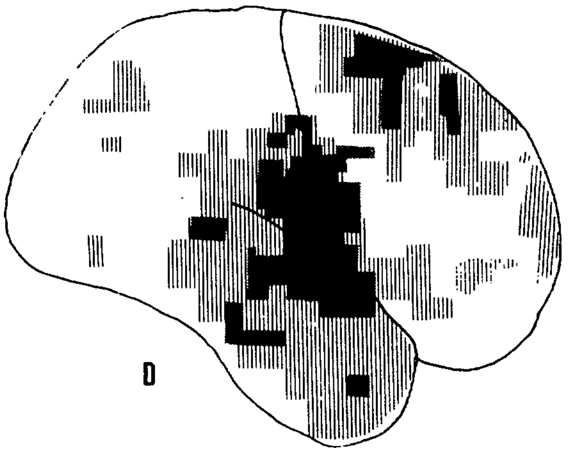


Figure 1. Cumulated PET Scans during a repetition task (Left and Right hemispheres, in Lavorel, 1983). G = left; D = right.

overlapping manner across the post-rolandic somato-sensory area. He and others explained that parallel representations ensure rich, accurate, and safe computation when thousands of neurons can answer simultaneously to, say, a prickling stimulation at the tip of one finger. Thus, first generation networks of artificial modules have only verified mathematical or electrophysiological models..

In fact, a reductionist attitude, although it teaches us something about some physical properties of some layers of neural tissue, does not increase our global comprehension of the processing style of the brain when, for instance, an animal sees a prey and catches it, or a man sees cherries and says "I'd like a pound of these cherries, please". So, what are the main principles that subserve mental representations and schemas? To-day, by covering much multidisciplinary theoretical ground I will try and define cooperative computation which is one of these principles. I claim that this aspect of Natural Computation can be implemented on modern computers and that it could enable scientists to improve the efficiency of their systems for language comprehension, picture recognition and automatic control of robot movements.

COOPERATIVE PROCESSES

The first striking fact that PET Scan pictures of the brain at work (Lassen et al., 1978) have shown neurologists is that when a subject is either perceiving something (light, sound or body sensation), or doing something (talking, moving a hand, etc...), injected glucids can be seen to be metabolized at the same time in several areas of the brain. The following picture is a cumulated representation of such an activity for several subjects who repeat numbers as they are hearing them.

Here, high activity can be observed in the temporal lobes, in lower and upper parts of the frontal lobes, along Rolando's fissure, and relatively intense activity is also to be found in several areas of the anterior and posterior lobes. Although the localization of high activity varies from one task to another, or from one type of stimulus to another, mental computation always seems to take place in a cooperative manner between several perceptual or gnosic areas of the posterior lobes and several motor or planning areas of the frontal lobes. Pathology had indicated that if only one area is severely lesioned, processing is disturbed at least for some time. That led to the belief of gnosic centers, praxic centers, language centers, etc... The global observation of normal activity indicates that the emphasis should be put on cooperation (or connexions, for neuroanatomists) rather than on localization. Performance may of course be disturbed and will have to be reorganized if part of one processing subsystem is destroyed. It may even not be reorganized if the system has aged and evolved into an over-specialized, over-automatized entity. That is particularly true for secondary perceptual and motor areas which already reach a relatively stable state a few weeks after birth.

From the standpoint of the cognitive functions that have to be inferred whenever psychologists study intelligent activities, like form recognition, or the grasping of an object on a table, or language comprehension, it is of course comforting to find that many cortical and subcortical areas have to collaborate in every form of behavior. To claim that the understanding of a spoken message must involve many levels of analysis and that these parallel searches may cancel or confirm one another (Lesser et al., 1981), or to claim that differences in processing speeds between one comprehension process and another may occasion hasty, partial, and erroneous interpretations (Frazier and Rayner, 1980), all that is entirely in keeping with what conjectures can be made about more or less successful idiosyncracies of calculation going on between the cooperating subsystems of the brain. But, instead of reasoning in abstracto as linguists have been accused of doing, it would be better to try and stipulate how cooperative interaction really takes place. There are four levels of issue that need to be considered by brain theory scientists :

1. What is the nature of information propagation between co-active areas? Is there something like bottom-up or top-down transfer between primary, secondary and other representational areas? Or between different modalities of cognition (Visual, auditory, somato-sensory)? Are control modalities distributed or centralized in certain areas (Frontal, limbic, sub-cortical)?
2. What are the basic functions for computation between the six layers of the cortex, between adjoining columns or modules, and between neighboring neurons?
3. Is there wild competition or well-timed harmonious regulation between the two hemispheres? Or between these two hemispheres and other sub-cortical more primitive subsystems?
4. How does a co-operative system evolve over time?

I will not address the fourth issue in this paper. Elements for an understanding of the problem have been suggested in Lavorel (1982) and in various papers about the research carried on by Spinelli, Bartow, or Bozinovski (1982).

1. What is the nature of information propagation between co-active areas?

Since no ultimate all-inclusive model can be devised at the present stage of physiological knowledge, it might be best to cite a few significant experiments that will suggest the style of information propagation in the cortex.

The recording of individual neurons or of groups of neurons in animals, as well as the anatomical study of cortico-cortical pathways with chemical or radio-active tracers, and with artificial lesions (Jones and Powell, 1970) has indicated that afferent signals travel from primary to associative areas (secondary, tertiary, etc...). It was therefore believed until recently that processing was hierarchically organized and that the associative areas dealt with more abstract and multimodal patterns while primary areas only identified features. It was also believed that secondary associative sensory information was normally sent to motor areas (Everts and Fromm, 1977). At least three experiments have recently shown that things are a little more complex. For instance, ascending activation waves are regulated (i.e. inhibited or preactivated) by descending priming. In many cases, perceptual neurons (e.g. of the primary visual cortex) may fire when motor responses to their previous sensory inputs are recorded (Fischer and Boch, 1981). For example, the following histograms from Bridgeman's recent paper (1982) show the peak activity of three neurons of layer IV of the primary visual cortex of monkeys (Macaca fascicularis) while these trained (awake) animals performed a voluntary movement in answer to a visual stimulus (light). Two neurons out of three keep firing while the motor response is triggered off. The histograms on the right show mean activity for a whole group of visual neurons (Multiplexing hypothesis) when there is and when there is not a motor response (= incorrect behavior) :

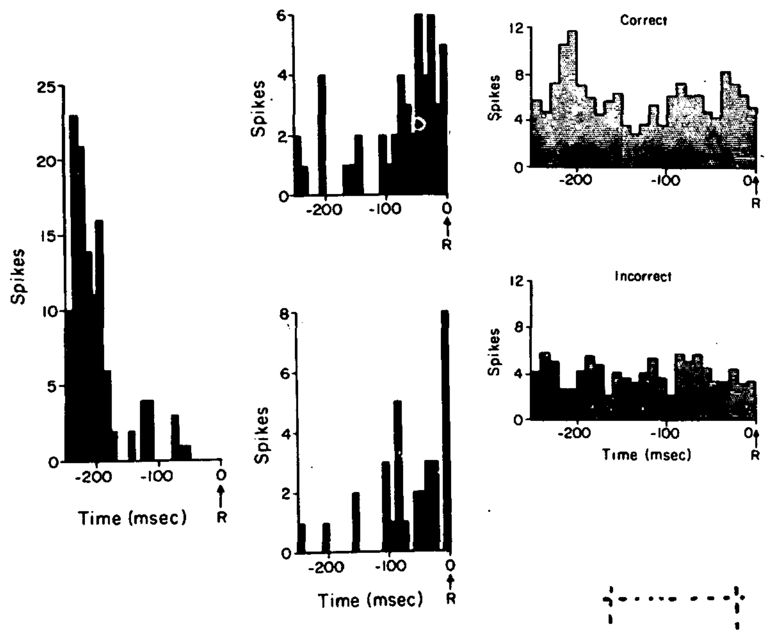
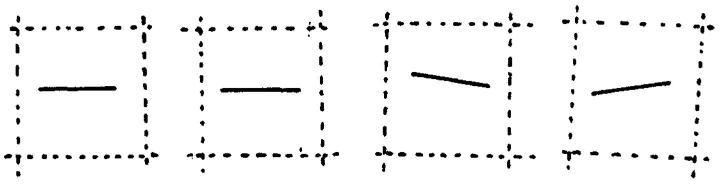


Figure 2. Two activity peaks in the visual cortex of monkeys. One corresponds to the motor response (R), one to visual stimulation that preceeded it (-120 milli-seconds).Two visual neurons out of three show renewed low activity (spikes = 6 per sec. and 8 per sec.) during the motor response.

Figure 4. Is this line horizontal? (Square frame). Discrimination is difficult. Visual feature perception is not independent from associated information.



The next figure represents peaks recorded in the secondary associative visual cortex of anesthetized paralyzed cats. Through learning, comparable responses can be evoked when an individual neuron is first primed by simultaneous visual and auditory stimuli and then excited only by the visual stimuli previously associated with auditory information.

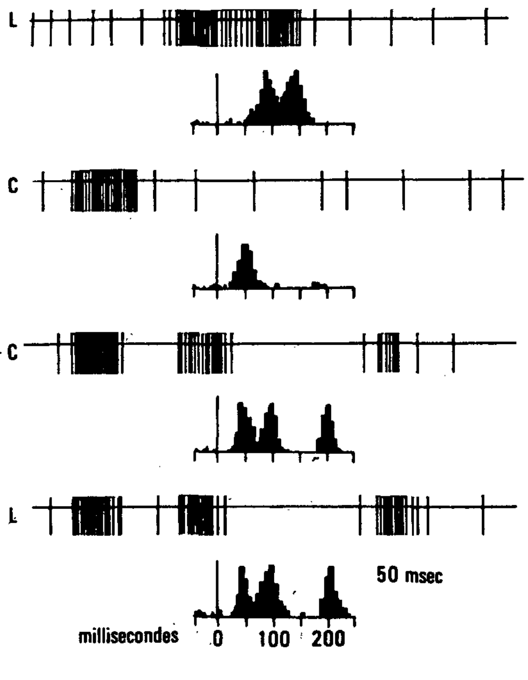


Figure 3. Modifiable discharge of a visual cortex neuron (Reproduced from Morrel, 1972). L stands for "response to a light beam stimulus". C stands for "response to a click". The fourth series of peaks (L) matches the third (L + C). The four series correspond to successive sequences of light, click, light and click, light stimulations. 40 repetitions of L + C are sufficient to determine multimodal learning.

The third experiment consists in showing horizontal lines on a circular screen to a subject (animal or human) waiting in the dark. The task consists in detecting the lines which are not perfectly horizontal. Physiological studies have shown that such a job is essentially depending on the activity of specific primary and secondary visual cortex neurons firing in a selective manner for lines oriented one way or another. Now, if the same lines are shown inside of a square frame that may be slightly tilted to the left or to the right, subjects very quickly make errors, thus indicating that the identification of oriented lines is far from being only a primary feature detection task.

As a first conclusion to these three experiments, one might say that visual perception, which has been picked up as an example of a bottom-up function where modules only have to answer to individual visual features, is in fact a very interactive process. The propagation of information to and from *multi-modal secondary perceptual areas,* to and from motor control areas appears to be just as important as the input stimuli from the outside world. Seeing is not just using one's visual system.

If we want to predict all the sorts of feedback and feed-forward regulation between subsystems in the brain, we only have to look at anatomy. Connections between hemispheres, cortex and subcortex, lobes, local modules are inumerable. Each area of the layered brain incorporates different neurons partaking in representational or in motor functions. Associative cortex itself generally includes layers of pyramidal motoneurons as well as layers of other neurons indirectly connected to perceptual pathways (4 to 7 relays).

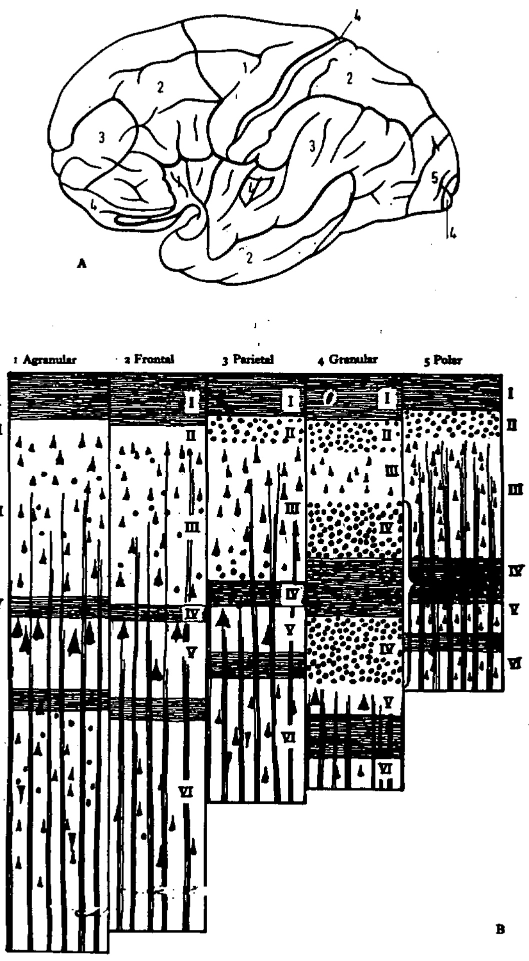


Figure 5. Von Economo's model of the distribution of neuron types in five major types of cerebral cortex (Williams and Warwick, 1975). As seen on this picture, when cytoarchitectonic studies of the cortex are consulted, they reveal that the so-called specific centers of perception or of motor control are not fitted exclusively with motoneurons or with perceptual neurons. For example, in addition to the giant pyramidal neurons of "frontal" and "parietal" cortex (layers III and V), the above "granular" and "polar" visual cortex (occipital lobe) also includes a number of neurons which are known to mediate motor processes (in layer V more exactly). The cortex of Brodmann's frontal areas 6-8 (premotor cortex) includes a good number of neurons answering to visual stimuli (in layer IV). As for the lower part of the parietal cortex, it includes layers of neurons participating in almost every representational or motor activity.

Consequently, a reasonable computational hypothesis about cooperation between cortical areas might be that activation is constantly propagated bottom-up and top-down without any other preestablished control principles than "motor neurons project to motor neurons, perceptual neurons project to perceptual neurons. As seen in Figure 3 above, specificity for one modality of information for individual areas would not even be an absolute principle. According to such hypothesis, what would tune down or preorder trains of signals would not be a preestablished central schedule, but only the varying effects of preactivation, of propagation timing, of post-firing refractory states, and of local inhibition by pre-synaptic and post-synaptic regulatory neurons. Varying delaying effects might also be attributed to regulatory neurons of layers I and II (See below). If such a theory was valid, control could even be completely distributed. Yet, pathology has shown that overall planning and scheming ensured by the limbic system in collaboration with fronto-orbital and pre-frontal areas also play a role : Self stimulation and conscious control are constantly introducing mediation, or priority, or global inhibition constraints on the information traffic (Pribram, 1967; Merzenich and Kaas, 1980). Therefore, control is partly distributed and partly regulated by coordinating units.

2. What are the basic functions for computation between the six layers of the cortex, between adjoining columns or modules, and between neighboring neurons?

Ever since Pitts and Mc Culloch started modelling neuron networks in the fifties, many contradictory theories have blossomed up and faded away. Among the most robust principles kept by all the theoreticians in physiological processes, two must be distinguished because they are relevant to the issues raised here.

First, the principle of cooperative excitation and of reinforcement or inhibition of the activity of motor cell layers by the other layers of the same module. An early piece of reliable evidence came from studies on the frog tectum by Lettvin et al. (1959), by Ingle (1976) and by Ewert (1976). When a frog sitting on a fixed disk is confronted with flies placed on a circular disk revolving around the first one, it will snap at them after having visually recognized them. Evoked potentials can be recorded in cells of various layers of the tectum. These physical measurements have been analyzed by mathematicians (Didday, 1970), and by theoreticians who built up a plausible theory of visually controlled motor performance (Arbib, 1981). In short, visual information gradually builds up an excitation cycle in several layers of neurons that all interact directly or indirectly with "foodness layers" where motoneurons are going to fire eventually and send information to the efferent motor system (For a short report and figures, see Lavorel and Arbib, 1981). What is interesting is that, while different layers compete to build up activation, cells of the superior layers (I, II) may or may not inhibit the process on the basis of complementary sensory input (perhaps visual or auditory context effects) and of motor feedback and feed-forward that has to do with present and past motor activity. In summary, everything looks like a cooperative decision function based on a very tricky timing of :

- one sensory input,
- multimodal knowledge mediation, and
- activation decay when a delay is introduced between the initial input and the final motor output.

Hardcore cyberneticians would argue that this looks like a sort of Boolean AND gate function mediated by a delay that is itself a function of activation, inhibition and decay functions. But that would be trying to think in terms of irrelevant concepts about a new type of calculus that is not really based on gate management but on very complex interactive excitation and inhibition among associated cells in an adaptive network. Furthermore, our frog example does not incorporate linear multiplexing effects which are obviously fundamental in mammal layered brain modules, where we do not deal with half a dozen neurons in six layers but with up to ten thousand of them competing for one decision. But results on mammal studies would perhaps make the computational style of local modules less obvious because of many physiological details that would hide the principal procedure. For the same didactic reason, the frog model does not go very far in integrating post-firing refractory states, peripheral inhibition around excited cells, and competition between neighboring modules or columns that receive inputs from contralateral sensory channels (e.g. alternating visual columns in the visualcortex correspond to ipsilateral or contralateral visual stimuli). In fact, phenomena are so intricate that paper and pencil modelling can not represent reality. Only computer simulation can handle many the variables and keep track of timing constraints.

The second basic function for computation between local cells is spreading activation dynamics which has been considered by psycholinguists to be a plausible explanation for associative behavior observed in semantic tests (Quillian, 1968; Collins and Loftus (1975)). Better computational arguments in favor of spreading activation have been offered by neuroscientists who work on wave dynamics and resonance phenomena in the axon and dendrite mesh of the cortex (Khodorov (1974); Scott (1977); and Scott (1978)). De Callatay (1981) recently conceived a brain simulation model that incorporates cascading top-down and bottom-up wave dynamics in an additive memory connected with a robot and pattern recognition devices. This bidirectional network, where excitation is propagated after input stimulation,is capable of recognizing previously memorized patterns and of associating motor schemas to sensory input. Activity is distributed to the different processing units in a periodical manner and it spreads along connections as long as it is not inhibited by concurrent propagation (Refractory or inhibiting effects). The bursting (i.e. firing) of cells determines vital intelligent operations, like the decision functions described above. But, simultaneously, the spreading of activation throughout the network ensures learning. Acquired knowledge can thus play a more and more important role as the system evolves and becomes more or less sensitive to individual patterns of excitation (For a better understanding, see Minsky, 1980).

Not only learning and physical growth, but also pathology and ageing must be considered when one deals with biological machines. Zaimov, for instance, studied the effects of biodegradation and, based on studies of Wedensky (1901), Pavlov (1949, 1951) and Uchtomsky (1966), found that in the case of lesions normal processing by a neural network may go through successive paranormal phases of extreme activity, of undifferentiated behavior, or of complete inhibition corresponding to necrotization.

In short, apart from cooperative decision functions, and spreading activation effects, many other computational aspects of neural modules are being observed, characterized and modelled. Most of the theoreticians conjectures are still to be verified. But it is already possible to assemble some principles and rules into prototheories of brain-like layered machines.

3. Is there wild competition or well-timed harmonious regulation between the two hemispheres? Or between these two hemispheres and other sub-cortical systems?

Most of the arguments in this field of research come from the observation of pathology. Nature performs experiments on the human central nervous system when small or large lesions appear for various traumatic or etiological reasons. The present state of knowledge about hemispheric specialization was reviewed recently by Springer and Deutsch (1981), and by Gazzaniga (1982). Rather than attributing particular faculties to either hemisphere, neuroscientists now consider that everything works as if two different entities, each one having its own processing style, were functioning simultaneously and certainly exchanging results at every step of calculation. Of course, each stimulus remains the same for the two hemispheres if the sensory captors are in a good functional state (Lavorel, 1983 b). What differs is the role of each hemisphere in ultimate representations of real or imaginary worlds.

What is more interesting is to consider control and timing effects when parallel procedures come to trigger off behavior.

A very spectacular and convincing experiment consists in presenting dicampic visual stimuli to normal or lesioned subjects (Lavorel, forthcoming). For a few milliseconds (50 to 200), they are shown a picture on one side (i.e., in one hemifield) and a word on the other side. For example, in one hemifield they see the picture of a snake and, in the center of the other hemifield, they see the word SERGENT. If the stimulus is presented after many other pictures (preferably faces or moving scenes), subjects often read SERPENT. If the stimulus is presented after many written words (preferably abstracts or compound words), they see the trap and read SERGENT in many cases. If, instead of presenting the picture of a snake, together with SERGENT, we present another word like SNAKE or VIPER, very few subjects read SERPENT by mistake. The results of this experiment in artificially controlled dyslexia are founded on competition between right and left hemisphere processes. When the double brain-system is habituated to a global Gestalt approach to pictures, it neglects analytical routines that identify individual letters and transcode them into a phonological structure. When the system is habituated to a linguistic task, like pure graphic-phonic translation, it is less influenced by a pictorial context. That the global Gestalt approach corresponds to right hemisphere dominance and that the analytical graphic-phonic translation corresponds to a left hemisphere dominance can be concluded from the fact that the side of the stimuli influences performance according to whether subjects are right-handed or left-handed. Another convincing element is the fact that patients with unilateral lesions tend to have either a unilateral extinction or a marked inferiority of one procedural style.

Other pieces of evidence, not about competition, but rather about harmonious cooperation between hemispheres have been found in neurolinguistics, the study of the neuropsychological substrates of language. The analysis of slips of the tongue or of lapsus linguae in normals, and the observation of semantic paraphasia in aphasics (Lavorel, 1980), indicate that repetition, naming or invoking rest upon converging left and right hemisphere cues (Lavorel, 1981). It is well known that lesions in the *dominant hemisphere* are detrimental to language. But it is also obvious that lesions in the minor hemisphere disturb word-processing transitorily or in a discrete manner. Furthermore, split-brain and disconnection studies have proved the minor hemisphere's verbal competence to be real though different and much weaker than the left hemisphere's glib abilities.

A word must be said about sub-cortical

external regulation. Pathology has demonstrated the essential functions of the limbic system, for example. However, much remains to be observed before a good understanding of goal calculation, of reward and punishment modalities, of attention, of automatization of behavior, etc, can be reached. Meanwhile,it is possible to consider that most of the already complex cooperative computation in the cortex is either prepared, or cross-checked, or completed by sub-cortical computation. But we will leave such issues aside for the moment.

After having accumulated a few pieces of evidence about what is meant by cooperative computation in the central nervous system, I will now conclude this study of one aspect of N.C. with a brief illustration of computer models that try to emulate the human brain.

COMPUTER SIMULATION OF ACTIVITY IN THE BRAIN.

Computer simulation of Natural Computation does not only provide a way of understanding complex interaction and fragile timing control in neurological systems. It is also an experimental approach to natural computation as long as the models implemented are true to life and not oversimplified. Initially observed phenomena can be reproduced, often leading to the discovery of unsuspected relationships and blatant insufficiencies in the models. Later, bold predictions about the weaknesses and the strength of neurobiological systems will be possible. Ageing, handicaps and possible recoveries will be understood from the inside of the black box. More strikingly, even heuristic processes of language and of problem solving that had remained out of the reach of machines will be grasped by mimicking the marvellous human brain.

One such system called HOPE has been developed at the University of Massachusetts by Helen Gigley, with the collaboration of neuropsychologists (Duffy and Lavorel). HOPE has been used for the simulation of some aspects of language pathology (Aphasia). Instead of trying to simulate human performance with a sequential model, the system permits simultaneous processing that is time synchronous. Several modules deal simultaneously with different aspects of a given task. Each module is a threshold mechanism with an associated activation. Excitation propagates at each step of calculation, either according to a fixed-time spreading schedule, or after reaching or surpassing threshold and firing (indicating a decision has been made about one facet of representations or schemas). Other control principles of HOPE are directly inspired by neurophysiology rather than by traditional algorithmic concerns. They include :

- Memory decay : any active information decays in time unless it is reinforced;
- Preorganized distribution of inputs into parallel modules that may collaborate or compete according to goals and to the nature of inputs;
- Post-firing effects : the firing of one module confirming one decision inhibits all the other processes that were competing on the same task;

The firing of one module also determines a very short-term refractory state in the same module.

All these characteristics will not be discussed here. They are presented at length in Gigley's PhD dissertation (1982) and in Gigley (1982). The system can be defined both as a distributive cooperative computation tool and as a set of deterministic devices achieving undeterministic calculus. For those two reasons HOPE is the first computer system with control modalities based on N.C.

The results of processing implemented from 1980 to 1982 have surpassed all expectations. Not only does HOPE perform typically human functions but it can also be induced to commit errors exactly like normal or pathological human brains. To illustrate these statements, here is an account of how HOPE can be used to reproduce our naming behavior and also our difficulties when we cannot for instance recall somebody's name, or remember how to say "good bye" in Japanese (Although we do know the word and feel that it is on the tip of our tongues). The processing of visual inputs is distributed in the following manner :

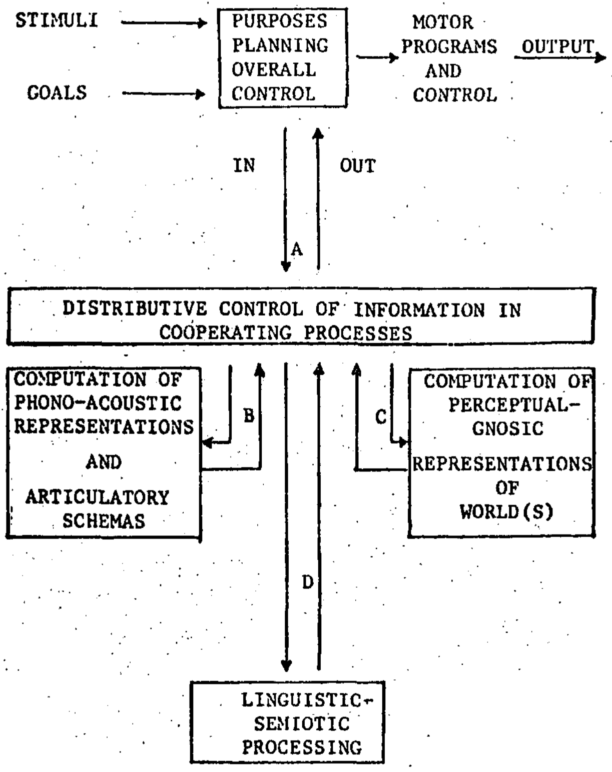

STIMULI
GOALS
PURPOSES PLANNING OVERALL CONTROL
MOTOR PROGRAMS AND CONTROL
OUTPUT
IN
OUT
A
DISTRIBUTIVE CONTROL OF INFORMATION IN COOPERATING PROCESSES
COMPUTATION OF PHONO-ACOUSTIC REPRESENTATIONS AND ARTICULATORY SCHEMAS
B
C
COMPUTATION OF PERCEPTUAL-GNOSIC REPRESENTATIONS OF WORLD(S)
D
LINGUISTIC-SEMIOTIC PROCESSING

Figure 6. Distributive control of representation modules used for naming tasks (reproduced from Lavorel and Gigley, 1982). It must be added that each one of the processing modules (Phono-acoustic; Perceptual-gnosic; Linguistic-semiotic) may itself be subdivided into two complementary ones (One for each hemisphere).

One instance of cooperative naming performance can be represented very schematically by the following flowchart :

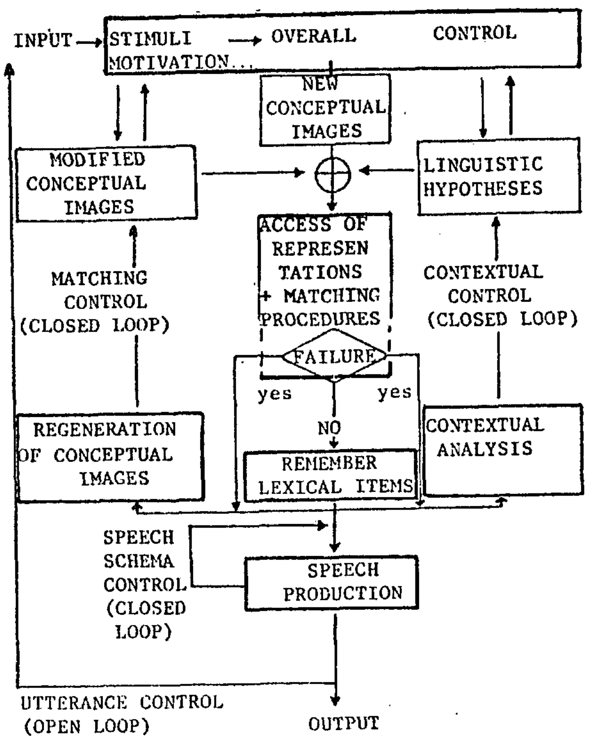

INPUT
STIMULI
OVERALL
CONTROL
MOTIVATION...
NEW CONCEPTUAL IMAGES
MODIFIED CONCEPTUAL IMAGES
LINGUISTIC HYPOTHESES
ACCESS OF REPRESEN TATIONS + MATCHING PROCEDURES
MATCHING CONTROL (CLOSED LOOP)
CONTEXTUAL CONTROL (CLOSED LOOP)
FAILURE
yes
yes
NO
REGENERATION OF CONCEPTUAL IMAGES
REMEMBER LEXICAL ITEMS
CONTEXTUAL ANALYSIS
SPEECH SCHEMA CONTROL (CLOSED LOOP)
SPEECH PRODUCTION
UTTERANCE CONTROL (OPEN LOOP)
OUTPUT

Figure 7. Naming procedure : one instance among many (Lavorel and Gigley, 1982). Note that distributive access to representations is associated with matching procedures (i.e., a sort of dynamic lexical component). For the sake of simplicity, only three interacting modes are focussed upon here (In the center, context free component; on the right, context sensitive component; on the left, transformational component). Each one of the three procedural modes competes with others or corrects them.

It must be remembered that since control is basically a matter of interactive activation, inhibition, and decay, knowledge consulted or produced at all levels will only play a part in world representations and in motor acts if it reaches a threshold. And it will do so only if the level of confidence associated with individual processes is high, or if several processes concur in building up confidence.

Abnormal performance, like word amnesia or misnaming, will take place if information that was to be processed in A, B, C, or D (Figure 7) is not transferred, if it is transferred too early or too late, and if it is distorted, fuzzied, or incomplete. For example, it has been observed that in left hemi-sphere lesions, proper names cannot be associated with portraits of well-known personalities either because patients find it difficult to recognize the individual features of the face presented to them, or because they cannot retrieve the phono-articulatory pattern to be associated with the given visual stimulus. Such a natural situation can be simulated by lowering activation either in the given artificial vision module or in the given artificial speech generation module. Thus a decision threshold can never be reached. Other solutions are to lengthen processing time (i.e. activation build-up) and to accelerate information decay, so that for instance visual identification is fuzzied or erased before it can be matched with one and only one candidate among phonological patterns.

The present state of N.C. models like HOPE can only be a far cry from actual brain work. However, all our recent experiments in cooperative computation tend to encourage us to pursue this new approach to the processing of natural language. Other principles of N.C. based brain studies will soon be tested and applied in systems development and programming. Another advantage of this new approach is that it enables neuropsychologists to test cognitive models of brain processing that are thought to match anatomical and physiological discoveries. Computer simulation thus appears to be an efficient method to verify the relevance and the robustness of theoretical models in the neurosciences.

BIBLIOGRAPHY

AMARI S. and ARBIB M.A. (1977) Competition and cooperation in neural nets. Systems Neuroscience. In J. Metzler (Ed.), Academic Press, 119-165.

ANDERSON J., SILVERSTEIN J., RITZ S. and JONES R. (1977) Distinctive features, categorical perception, and probability learning: some application of a neural model. Psychological Review, 84, 5, 413-451.

ARBIB M.A. (1981) Perceptual structures and distributed motor control. In V.B. Brooks (Ed.), Motor control, vol. 3, Handbook of Physiology. Bethesda MD: American Physiological Society.

ARBIB M.A., CAPLAN D. and MARSHALL J. (1982) Neural models of language processes. New-York: Academic Press.

ARBIB M.A. and MANES E.G. (1975) A category-theoretic approach to systems in a fuzzy world. Synthese, 30, 381-606.

BRIDGEMAN B. (1982) Multiplexing in single cells of the alert monkey's visual cortex during brightness discrimination. Neuropsychologia, 20, 1, 33-42.

BOBROW R.J. and WEBBER B.L. (1980) PSI-KLONE, Parsing and semantic interpretation in the BBN Natural Language Understanding System. Proceedings of the Third Bi-annual Conference of the Canadian Soc. for computational Studies of Intelligence. Victoria B.C., May, 1980, 131-142.

BOZINOVSKI S. (1982) A self-learning system using secondary reinforcement. In R. TRAPPL (Ed.), Sixth European Meeting Cybernetics and Systems. The Hague: Mouton, 87.

CALLATAY A.M. de (1981) Parallel list processing in a brain like hardware. IBM Educational Center Technical Report IEC 0021. Brussels.

COLLINS A.M. and LOFTUS E.A. (1975) A spreading activation theory of semantic processing. Psychological Review, 82, 6, 407-428.

DALENOORT G.J. (1981) Modularity of behavior of networks. In R. TRAPPL (Ed.), Sixth European Meeting on Cybernetics and Systems. The Hague: Mouton, 1982, 67.

DIDDAY R.L. (1976) A model of visuomotor mechanisms in the frog optic tectum. Mathematical Bioscience, 30, 169-180.

ECCLES J.C. (1977) The understanding of the brain. New-York : Mc Graw-Hill.

ERMAN L. and LESSER V.R. (1980) The HEARSAY II system : a tutorial. In W.A. LEA (Ed.), Trends in speech recognition. Englewood Cliffs, N.J. : Prentice-Hall, 361-381.

EVART E.V. and FROMM C. (1977) Sensory responses in motor cortex neurons during precise motor control. Neuroscience Letters, 5, 267-272.

EWERT J.P. (1976) The visual system of the toad. In K.T. Fite (Ed.), The amphibian visual system : a multidisciplinary approach. New-York : Academic Press.

FRAZIER L. and RAYNER K. (1980) Making and correcting errors during sentence comprehension : eye-movements in the analysis of structurally ambiguous sentences. Univ. of Massachusetts, Amherst. Working paper.

GAZZANIGA M.S. (1982) Right hemisphere language following brain bisection : a twenty year perspective. Cornell Medical College. Working paper.

GIGLEY A.M. (1982) Artificial Intelligence meets Brain Theory. An integrated approach to simulation modelling of natural language processing. In R. TRAPPL (Ed.), Sixth European Meeting on Cybernetics and Systems. The Hague: Mouton, 1982, 190.

INGLE D. (1976) Spatial vision in anurans. In K.T. Fite (Ed.), The amphibian visual system: a multidisplinary approach. New-York : Academic Press.

JONES E.G. and POWELL T.P.S. (1970) An anatomical study of converging sensory pathways within the cerebral cortex of the monkey. Brain, 93, 793-820.

KHODOROV R.I. (1974) The problem of excitability. New-York : Plenum Press.

LASSEN N.A., INGVAR D.H. and SKINHOJ E. (1978) Brain function and blood flow. Scientific American, 239, 4, 50-59.

LAVOREL P.M. (1980) Sept formes de jargon. Grammatica, 1, 61-100. Numéro spécial: Etudes Neurolinguistiques. Presses Université de Toulouse.

LAVOREL P.M. (1981) Quand la langue fourche. In J. Oudot and A. Morgon (Eds.), L'erreur. Colloque Patch de Lyon. Presses Universite de de Lyon, 1982.

LAVOREL P.M. (1982) Production strategies: a systems approach to Wernicke's aphasia. In M.A. Arbib, D. Caplan and J.Marshall (Eds.), Neural models of language processes. New-York: Academic Press, 135-164.

LAVOREL P.M. (1982 b) The sensible brain. In O. Selfridge, E. Rissman and M.A. Arbib (Eds.), Adaptive control of ill-defined systems. New-York: Plenum Press (forthcoming).

LAVOREL P.M. (1983) The distributed processing of knowledge and belief in the human brain. New prospects in Natural and Artificial Intelligence, NATO Symposium. Elithorn, Schank and Banerji (Eds.), Cambridge U.P. (forthcoming).

LAVOREL P.M. (1983 b) Le cerveau se représente-t-il des arbres syntaxiques? In J.P. Desclés (Ed.), Les arbres en linguistique. Colloque AFCET-ATALA Paris: Hermann (forthcoming).

LAVOREL P.M. (forthcoming) An artifical simulation of deep dyslexia. International Neuropsychology Society European Conference. Deauville, France, June 1982.

LAVOREL P.M. and ARBIB M.A. (1981) Towards a theory of language performance. Theoretical Linguistics, 8, 1. Berlin: Walter de Gruyter.

LAVOREL P.M. and GIGLEY H.M. (1982) How we name or misname objects. The simulation of cooperative computation in the human brain. In TRAPPL (Ed.), Sixth European Meeting on Cybernetics and Systems. The Hague: Mouton.

LESSER V.R. et al. (1981) A high-level simulation test-bed for cooperative distributed problem-solving. Computer and Information Science, University of Massachusetts. Technical Report, 81-16.

LETTVIN J.Y., MATURANA H., MC CULLOCH N.S. and PITTS W.H. (1959) What the frog's eye tells the frog's brain. Proceedings of the Institute of Radio Engineers, 47, 1940-1951.

MAC GREGOR R.J. and LEWIS E.R. (1977) Neural modelling. New-York : Plenum Press.

MARCUS M. (1978) A theory of syntactic recognition for Natural Language. Ph D. Dissertation, Cambridge : MIT.

MARCUS M. (1982) Consequences of functional deficits in a parsing model : implications for Broca's aphasia. In M.A. Arbib, D. Caplan and J. Marshall (Eds.), Neural models of language processes. New-York : Academic Press.

MERZENICH M.M. and KAAS J.H. (1980) Principles of organization of sensory-perceptual systems in mammals. Progress in Psychobiology and Physiological Psychology, 9, 1-42. New-York: Academic Press.

MINSKY M. (1961) Steps towards Artifical Intelligence. Proceedings of the IRE, 49, 8-29.

MINSKY M. (1980) K-Lines : a theory of memory. Cognitive Science, 4, 117-133.

MORREL F. (1972) Integrative properties of parastriate neurons. In Karczmar and Eccles (Eds.), Brain and human behavior. Berlin : Springer, 259-288.

MOUNTCASTLE V.B. (1976) The world around us : neural command functions for selective attention. Neurosciences Research Program. Bull., 14, suppl. 1, 1-47.

NEWELL A. (1978) HARPY, production systems, and human cognition. CMU Technical Report : Carnegie-Mellon University.

NILSSON N.J. (1965) Learning machines. New-York : Mc Graw-Hill.

PAVLOV I.P. (1949) Lectures on the functioning of the two hemispheres of the brain (cited in K. Zaimov) Moscow Academy of Sciences. Soviet Union Publishing House.

PAVLOV I.P. (1951) Twenty year experience of objective research in the higher nervous activity of animals (cited in K. Zaimov) Moscow : Medgiz.

PRIBRAM K.H. (1967) The limbic systems, efferent control and neural inhibition and behavior. In Adey and Tokizane (Eds.), Progress in Brain Research, vol. 27, Amsterdam : Elsevier, 318-336.

REISS R. (1962) A theory and simulation of rhythmic behavior due to reciprocal inhibition in small nerve nets. Proceedings AFIPS, Spring Conf., 171-194.

RIEGER C. (1975) Conceptual memory and inference. Conceptual Information Processing. In R.C. Schank (Ed.), San Francisco: North-Holland, 157-288.

RIESBECK C. (1975) Conceptual analysis. Conceptual Information Processing. In R.C. Schank (Ed.), San Francisco: North-Holland, 83-156.

QUILLIAN M.R. (1968) Semantic Memory. In M. Minsky (Ed.), Semantic Information Processing. Cambridge: MIT Press, 227-270.

SCHANK R.C. (1973) Identification of conceptionalizations underlying natural language. In R.C. Schank and K.M. Colby (Eds.), Computer models of thought and language. San Francisco: Freeman and C°, 187-248.

SCHANK R.C. (1975) Using knowledge to understand. Theoretical issues in natural language processing. Cambridge, Mass., 131-135.

SCOTT A.C. (1977) Neurophysics. New-York: Wiley Interscience.

SCOTT A.C. (1978) Brain theory from a hierarchical perspective. Brain Theory Newletters, 3, 3-4, 66-69.

SIMMONS R.F. (1973) Semantic networks; Their computation and use for understanding English sentences. In R.C. Schank and K.M. Colby (Eds.), Computer models of thought and language, New-York: Freeman and C°, 63-113.

SMALL S. (1980) Word expert parsing: a theory of distributed word-based natural language understanding. Ph D. Dissertation, University of Maryland, College Park;

SPRINGER S.P. and DEUTSCH G. (1981) Left brain, right brain. San Francisco: Freeman and C°.

SZENTAGOTHAI J. and ARBIB M.A. (1974) Conceptual models of neural organization. Cambridge, Mass.: MIT Press, 1975 (also NRP Bulletin, 1974).

THORNE J., BRATLEY P. and DEWAR H. (1968) The syntactic analysis of English by machine. In Mitchie (Ed.), Machine intelligence, Edinburgh Univ. Press, 281-309.

UCHTOMSKY A.A. (1966) The dominant (cited in K. Zaimov). Moscow: Nauka Publishing House.

WEDENSKY N.E. (1901) On the reciprocal relations between the psychomotor centres. In K.M. Bikov (Ed.), Physiology of the nervous system, vol. 3. Moscow: Medgiz, 1952, 181-189.

WILLIAMS P.L. and WARWICK R. (1975) Functional neuroanatomy of man. Philadelphia: Saunders C°.

WINOGRAD J. and COWAN J.D. (1961) Reliable computation in the presence of noise. Cambridge: MIT Press, 1963.

WINOGRAD J. (1972) Understanding natural language. New-York: Academic Press.

WOOD C.C. (1982) Implications of simulated lesion experiments for the interpretation of lesions in real nervous systems. In M.A. Arbib, D. Caplan and J. Marshall (Eds.), Neural models of language processes. New-York: Academic Press.

WOODS W.A. (1970) Transition network grammars for natural language analysis. Communications of the ACM, 13, 10.

WOODS W.A. et al. (1976) HWIM, Speech understanding systems. BBN Report : 34-38.

ZAIMOV K. (1981) Associative processes in semantic jargon and in schizophrenic language. In J.W. Brown (Ed.), Jargonaphasia. New-York : Academic Press, 151-167.

INFORMATICA 2/1982

INFORMATIZACIJA KOT SVETOVNI IZZIV

ANTON P. ŽELEZNIKAR

UDK: 681.3:007

DO ISKRA-DELTA, LJUBLJANA

Informatization as the world challenge. Informatization is a notion which can be deduced in the full sense from information, informatics, intelligence, technological, and social progress in the last two or three decades. Informatization includes computer usage, robotics, new education, intelligent environment, new management information, information home; it transforms the human behaviour into informatic one and enables the existence of subcultures and supercultures.

An information is a process of interpretation, representation, evaluation, execution of a message (linguistic, sensual, data, signal) in the central nervous system or also a reaction of a living cell to the changes of its environment. A technological information is convergent (single meaning) in the contrary to the human information which is normally divergent (pragmatic).

Informatics is a working, research, technological sphere using computer as a concentration of algorithmic intelligence (deterministic knowledge). Informatics is amplifying the human information environment, it develops its ideology, style of life and of work; here, the silent transition into informatization as an organizing strategy of life is taking place.

In the transition phase of informatization the intelligence as a specific information process in brains and machines is getting more and more important creating the so called intelligent environment. This kind of environment evoluates the use of human brain substentially, raising the intensity of intellectual work. Informatization connects systems into planetary net, information into sceleton world information, production parts into distributed production, so, the integration of resources is getting a complete form which could not be achieved in the wave of industrialism. In this way, informatization is becoming a challenge for developed and undeveloped countries giving them an encouraging vision of the future.

Informatizacija je pojem, ki ga je moč smiselno izpeljati iz pojmov informacije, informatike, inteligence ter iz tehnološkega in družbenega napredka v zadnjih dveh ali treh desetletjih. Informatizacija vključuje uporabo računalnikov, robotiko, novo prosveto, inteligentno okolje, novo upravljalsko informacijo, informatizirani dom; informatizacija transformira človekovo obnašanje v informatično in omogoči enakopraven obstoj sub in super civilizacij.

Informacija je proces interpretacije, predstavitve, vrednotenja in izvršitve sporočila (jezikovnega, čutnega, podatkovnega, signalnega) v centralnem živčnem sistemu in je tudi reakcija žive celice na spremembe njenega okolja. Tehnološka informacija je konvergentna (enoumno pomenska) v primerjavi s človeško informacijo, ki je normalno divergentna (pragmatična).

Informatika je delovno, raziskovalno in tehnološko področje, ki uporablja računalnik kot nakopičeno aloritmično inteligenco (determinirano znanje). Informatika ojačuje človekovo informacijsko okolje, razvija pa tudi svojo ideologijo, stil življenja in dela; tu se začenja tihi prehod v informatizacijo kot organizacijsko strategijo življenja.

V prehodnem obdobju informatizacije postaja inteligenca kot posebna oblika informacijskega procesa v možganih in strojih vse bolj pomembna, saj oblikuje tklm. inteligentno (pametno) okolje. To okolje bistveno evoluira (razvija) uporabo človekovih možganov, ko pospešuje intenzivnost umskega dela. Informatizacija poveže sisteme v planetarno mrežo, informacijo v skeletno svetovno informacijo, proizvodne dele v porazdeljeno proizvodnjo; tako dobi integracija virov (človeških, materialnih) tisto popolno obliko, ki je ni bilo moč doseči v dobi industrializma. Informatizacija postane izziv za razvite in nerazvite, saj jim daje vzpodbudno sliko prihodnosti.

Uvod

Informatizacija je pojem, ki se pojavlja med drugim v delih J.-J. Servan-Schreiberja (1) in A. Tofflerja (2) in ga je moč smiselno izpeljati iz pojmov informacije, informatike in inteligence. Informatizacija je novi val (tehnološki, družbeni), ki se razvija po industrializmu in ta val se začenja v ZDA v razdobju 1955 do 1965, nekaj kasneje zajame Japonsko z vso svojo močjo, v lažji obliki pa tudi zapadno Evropo in druge predele zemeljske oble. Informatizacija obsega uporabo računalnikov na vseh ravninah, robotizacijo proizvodnje s pomočjo inteligentnih informatičnih pripomočkov, novo prosveto, oblikuje pa tudi pametno (inteligentno) okolje, upravljalsko informacijo v družbeni in zasebni infosferi, informacijski dom (Toffler imenuje ta pojav elektronski dom), človeka preoblikuje v informacijsko intenzivno bitje, ki iz skeletne informacije ustvarja (ustvarjalno oblikuje) svojo polno informacijo, omogoča pojav, razvijanje in ohranjanje lokalnih in globalnih kultur (subkulture in superkulture), ki so s planetarnim informacijskim sistemom povezane v mrežo. Informatizacija spremeni način življenja iz industrijskega v informatičnega.

Toffler predvideva, da bi zaradi informatizacije in robotizacije proizvodnje v prihodnje kar 75 % delavcev opravljalo svoje delo iz svojega elektronskega doma, ki je seveda ustrezno informatično opremljen in povezan npr. s tovarno, pisarno oziroma z delovno funkcijo. To pa pomeni, da tovarna kot industrijska katedrala ne predstavlja več dnevnega romanja delavcev na delovno mesto, delavci pa so z delovnim procesom povezani iz svojih domov, ko krmilijo, nadzorujejo in usmerjajo proizvodne tokove skladno s svojo delovno funkcijo. To je morda vizija prihodnosti, ki pa se uresničuje že danes z uporabo lokalnih informacijskih mrež in še ne v polnem obsegu.

Informacija, informatika, informatizacija

Kratko: informacija je proces interpretacije, predstavitve, vrednotenja, izvrševanja sporočila (jezikovnega, čutnega, podatkovnega, signalnega) v centralnem živčnem sistemu. Človek sprejema in oddaja sporočila oziroma ukrepa. Če je informacija reakcija na spremembo ali stanje okolja, potem jo je moč smiselno posplošiti tudi na procese v živi celici (prilagajanje) in na tehnološke procese (avtomatsko krmiljenje z višjo ali nižjo stopnjo t.i. umetne inteligence).

Sporočilo kot dražljaj sproži npr. interpretivni proces v možganih z vsemi pragmatičnimi možnostmi, zato je informacija kot mednevronski proces (nevron je živčna celica) oziroma proces v nevronski mreži različna pri enakem sporočilu (ponovljenem, v različnih možganih, pri različnih stanjih spomina, pri različnem kemizmu, električni prevodnosti oz. biološkemu stanju nevronske substance).

Tehnološka informacija je predvsem enopomenska (npr. strojno prevajanje jezikov), da bi bili procesi predvidljivi ter v tehnoloških mejah. Tehnološka informacija je proces ugotavljanja "pomena" (semantike) brez pragmatičnih ekskurzij, je torej konvergentna za razliko od divergentne, razvijajoče informacije v možganih.

Informatika je delovno, življensko, tehnološko področje, uporablja računalnik, ki ima nakopičeno algoritmično inteligenco (determinirano znanje) v obliki programov za najrazličnejše uporabe. Informatika kot tehnološki pojav ojačuje človekovo informacijsko okolje z znanjem (programi) in sredstvi (računalniki, informacijski sistemi, telekomunikacije). Današnja informatika se tako lahko utemelji na računalniku z njegovo nakopičeno inteligenco in v obliki informacijskih sistemov seže na vsa bistvena področja dela in bivanja. Informatika razvije svojo tehnologijo in metodologijo ter tudi svojo ideologijo, stil življenja in dela. Tu pa se pravzaprav že začenja tihi prehod v informatizacijo kot organizacijsko strategijo družbe.

Informatika je kot raziskovalno in delovno področje utemeljena z znanostjo o informaciji, raziskavami centralnega živčnega sistema, umetno inteligenco, z računalniškimi znanostmi (raziskovalno in izobraževalno področje), z znanostjo in uporabo informacijskih sistemov, z uporabo računalnikov, z uvajanjem inteligence v robotiko pa tudi z njeno teorijo in prakso, s sistemi za iskanje podatkov in naposled tudi s sredstvi in metodami masovnega in drugega obveščanja.

Na prehodu v informatizacijo dobi inteligenca kot specifičen informacijski proces v možganih in strojih svoj poseben, poudarjen pomen. Prehod iz informatike v informatizacijo je bistveno pogojen z inteligenco, in sicer s povečano strojno pametjo ter z oblikovanjem t.i. inteligentnega (vse bolj pametnega) okolja. Stroj, ki ojačuje znanje človeka in razpolaganje z znanjem, evoluira tudi človekove možgane, sili ga v konstruktivno mišljenje, v delo, ki zahteva obvladanje (odkrivanje in reševanje) problemov ter ima svojo praktično, obstojno utemeljeno upodobitev oziroma realno projekcijo. Tako se človek z informatizacijo realizira na drugačen, času in prostoru ustreznejši način, ko izstopa iz dobe industrializma novemu upanju naproti.

Informatizacija se tako glede na informatiko razširi, v svoje okrilje vključi in poveže različne infrastrukture, kot so avtomatizacija (regulacijski sistemi), informatika (porazdeljena obdelava podatkov), telekomunikacije (sateliti, mreže), inteligenca (v sistemu, okolju), robotika (robotizacija proizvodnje in storitvenih dejavnosti), raziskovanje (informacijski procesi, nove tehnologije) in izobraževanje. Informatizacija pomeni tako med drugim tudi realizacijo teleinformatike in njenih posledic, kot sta porazdeljena proizvodnja (delo iz informatiziranega doma) in porazdeljena administracija (informatizirani urad). Informatizacija omogoči dostop do željenih podatkov, ki so zbrani v posebnih bazah podatkov za javno uporabo – tudi posameznikom iz njihovih domov – omogoči kakršnokoli od doma ali od drugje naročeno obdelavo podatkov za različne namene: podatki in njihove obdelave postanejo tako porazdeljeni po informacijski mreži, ki zajame in poveže celoten planet pa tudi izvenplanetarne podatkovne postojanke. Tako niso porazdeljeni samo podatki, porazdeljena je tudi obdelovalna moč in v informacijskem pomenu postane svetovna mreža podobna velikemu možganskemu informacijskemu sistemu.

Pametno okolje

Dvom o tem, da je informatizacija na pohodu, nima več realnih temeljev. Japonska hiti v robotizacijo in se dviga na višje ravnine vsakodnevnega delavčevega izobraževanja iz strahu pred prihodnostjo, v kateri bo moč zadovoljivo reševati probleme njenega in svetovnega obstoja le s pomočjo informatizacije. Z njeno močjo bodo Japonci racionalno uravnavali svojo potrošnjo (surovinsko, energetsko, prehrambeno) in dovolj hitro odpirali možnosti za nove poti proizvodnje, ki bodo postale nujnost (biološka, akvakulturna, vesoljska, elektronska industrija).

Izgleda, da bo informatizacija povzročila intenzivnejše izkoriščanje in uporabo človekovih možganov, s tem da bo zavestni del drugače napolnjen oziroma informacijsko bolj intenziven, saj se bo naposled tudi človek prilagajal informacijskemu okolju.

Značilnost informatizacije kot družbenega procesa je tedaj tudi v intenzivaciji človekovega intelektualnega dela, v prestrukturiranju človeka kot včerajšnjega ročnega, fizičnega, pisarniškega in upravljalskega delavca v današnjega in novega delavca, ki bistveno drugače uresničuje svoje delovne naloge v nastajajočem informacijskem okolju, ki je bolj pametno. V čem se ta višja oblika pameti odraža?

Značilnost informatizacije je možnost in nujnost stalnega delavčevega izobraževanja (v šoli in na delovnem mestu) zaradi hitrih sprememb informacijske (in druge) tehnologije. Prosveta dobi v informatizaciji novo obliko: intenzivno je podprta z informatičnimi pripomočki za učenje, dopolnilno in delovno izobraževanje, za reševanje in odpiranje problemov, za doseganje vsakršnih podatkov (strokovnih, upravljalskih, statističnih, poslovnih) iz specializiranih podatkovnih baz ter v prebavljivi, interaktivni, dialogni, menujski obliki. Uporaba te informatične podpore ne zahteva od človeka posebnega znanja: sistemi vodijo človeka prek uporabnih možnosti, prikazujejo mu možne alternative in pripomočke za iskanje podatkov, učenje, dopolnjevanje znanja. Seveda se ta informatična infrastruktura vzdržuje s človekom: dopolnjuje se ne samo kot uporabniški marveč tudi kot delovni sistem v proizvodnji. Vedno več ljudi je udeleženih v proizvodnji informatičnih sredstev, tehnologije, programov, uporabniških in delovnih sistemov. Človek tako povečuje ne samo svojo tehnološko pamet, povečuje tudi svojo konstruktivno informacijo in iz informacijskih skeletov oblikuje sebi ustrezno, novo informacijo, s sebi primernejšo vsebino. Takšen način informatizacije bivanja povečuje človekovo neodvisnost, njegovo varnost, saj mu daje možnosti predvidevanja dogodkov z uporabo dovolj kvalitetnih podatkov.

Ker je informatizacija izrazito integrativen pojav, ki poveže sisteme v svetovni sistem, informacijo v skeletno svetovno informacijo, dele proizvodnje v združeno proizvodnjo, dobi integracija v informatizacijski dobi tisto polno obliko, ki je nikoli ni mogla doseči v dobi industrializma. Vlogo industrializacijskih integratorjev in elit prevzamejo enostavno informatizacijske možnosti, informatizacijska moč in povezanost in tako se začenjajo zmanjševati razlike v položajih človeka, v stopnjah njegove obveščenosti, povečuje se človekova delovna primernost, njegova motivacija in zadovoljstvo. Človek zaživi svojim hotenjem in svoji motivaciji primerno, izboljša svoje zdravje in počutje, svojo vključenost v svetovno informacijo, postane dejansko pametnejši (bolj razvit, bolje obveščen), preudarnejši, sposobnejši za polno življenje.

Vprašanje inteligence (pameti) postane naposled nakako obrobno: inteligenca je vselej relativna, specializirana, prilagojena prostoru in času. Inteligenca je le posebna oblika trenutno aktualne informacije, ta pa se pretaka (v obliki procesov) v živi in tehnološki informacijski mreži. Inteligenca postane tako le dovolj bistveni del informacije in njena statistična prisotnost zagotavlja smotrno delovanje sistema: inteligenca postane bolj regularnost kot izjemnost, čeprav se povečuje njena raznolikost in količina (kopičenje znanja).

Sklep

Informatizacija postaja izziv za razvite in razvijajoče, saj kot možnost in nekje tudi kot dejansko stanje daje vzpodbudno vizijo prihodnosti: ukinja klasičen industrializem in njegova tipična nasprotja, spremeni nastajanje in strukturo upravljalske informacije in z njo povezanih institucij (demokratičen stroj, upravljalske elite), utrdi družino in manjše delovne skupnosti v neposrednosti preko dela iz informatiziranih domov, razreši s spremenjenim načinom bivanja in dela energetske, transportne, prehrambene in druge ključne človekove probleme. Informatizacija se pojavi kot počasi naraščajoči val, ki prekrije planet z novo življensko vsebino.

Slovstvo

(1) J.-J. Servan-Schreiber: Svetovni izziv. Globus, Zagreb (1981).

(2) A. Toffler: The Third Wave. Bantam Book, New York (1981).

INFORMATICA 2/1982

COMPUTERGRAPHICS AND CAD ACTIVITIES IN AUSTRIA

JOHANN WEISS

INSTITUT FOR DATENVERARBEITUNG TU WIEN, AUSTRIA
SYSGRAPH GESMBH, WIEN, AUSTRIA

UDK: 519.688

Abstract:

Computergraphics (CG) and CAD have had a quick development in Austria during the last 5 years. Of course, Austria has no companies producing graphic equipment in large quantities nor do we have big companies dealing with graphic software. But there is big need for such software and hardware and some progress has been made in the development. This paper is giving a survey about the Austrian activities research and application in this field.

1. Computergraphic: Research and Development

Basic research has been made on Computergraphics at the Technical University of Vienna (TU Wien). There are several institutes involved in CG-research.

In 1977 a Computergraphic Research Group was founded at the "Institut für Datenverarbeitung" (Dataprocessing) (Prof.Dr.Eier, J.Weiß). First a three years program was started including hardware and software development. Several papers of this work have been published, including "circle clipping", decentralization of graphic systems, standardization, development of a colour raster display, development of a graphic workstation etc. This project was followed by some others, mostly oriented to CG-application and CG-standardization. e. g.: A program for input of city maps together with statistical information is now being developed.

Beside these above mentioned activities the "Institut für Datenverarbeitung" offers also an educational program for computer graphic. For interested students exists the possibility of a lecture including practicum (3 less. per week/ one semester). This lecture shows an interesting development in the number of students: In 1978 there were 18 students, in 1982 there are now 175 (!) students. Further there exists the possibility of practicas and diploma works.

Another Institute working (mostly) on graphic languages is the "Institut für praktische Informatik" (practical informatics, Prof. Barth). Prof. Barth and his crew designed a graphic extension to Pascal. This Pascal/Graph was presented 1981 at Eurographics and was elected as 3rd paper. This institute, too, offers possibilities working with grahpics for the students.

The work of the "Institut für Informationsverarbeitung" of the "Österreichische Akademie der Wissenschaften" (ÖAW) - Austrian Academy of Sciences - (Dr.Firneis, DI Herzner) concentrates to several University application projects. Since 1979, the Institute is also working in the fields of graphic standardization.

Some examples of this graphic application are:
Drawings of cinematic functions,
contours of complex differential equation systems,
3-dimensional representation of parameterized surfaces using arbitrary sets of curves.

Furthermore the institute is supporting many applications with graphical software in the area of University of Vienna and Academy of Sciences. In the future it is planned to adapt existing software and to base new application on the Graphical Kernel System (GKS).

There is not only the university area, but also companies developing computergrahpic software in Austria.
The bigger companies as Siemens, CDC, IBM etc. provide their packages and develop user-specific adaptions. There are companies developing graphic software. Fa. Technodat provides a Calcomp based package named "Dipp", which was extended by segment- and colour-functions, and can handle a big variety of devices.

Another company, "Sysgraph", is consulting and developing in computergraphic applications. Several user-specific programs have been installed. A standard graphic package and a business graphic package are announced for this year.

The "Forschungszentrum Seibersdorf" also has a crew (Dr.Schoitsch) working on process-control and computer-graphic. They wrote graphic software whichs runs on Digital computers.

2. CAD - Development

2.1) TU Wien

In the department of mechanical engineers the "Institut für Maschinenelemente" (Prof. Kazda, Doz.Reinauer) has built up a CAD-Center. They use PROREN and have already made several pilote studies. Therefore the PROREN-system was extended several times. The applications are not restricted to drafting purposes. Beside these there are modelling functions, dimensioning, selecting of machine parts supported by computer. Application which have been done are crank shaft drives, gear boxes, clutches etc. With CAD-kernel package DINAS a roller bearing catalogue system was installed. A surface design program originally made by the Institut für Datenverarbeitung was extended to a 3D surface modelling system including hidden line removal.

Since 1979 a CALMA system has veen installed at TU Wien (Prozeßrechnerzentrum). This system is mostly used with CALMA software for applications, now there is also development done on it.

2.2) Industry

In the industrial field CAD-development is done at various places. A Viennese company dealing with prefabricated buildings (FTB) has designed their own CAD - and drafting system.
VOEST developed a CAM-system which is also marked outside the company. Support for technical computation (Finite elements etc.) is given by Technodat.

3. Applications:

3.1) Computer Animation

Since 1980 a company exists in Vienna which only deals with computer animation. A $2^{1}/2$ dimensional animation system was designed which is now used in production.
$2^{1}/2$ dimension means, that it is a system based on surfaces which can be moved in 3-d without restrictions. You can even have perspective views from this system. Generally the ANIMA-System is organized like an ordinary cell-production animation. There are of course no limitations in numbers of cells, which you can define and move simultaneously. The system takes care about what is visible or not. There are, more possibilities like on Walt Disney's multiplane camera, but with less costs.
The artist (animator) has an excellent user aid and a great variety of effects on controls.

Just to define effects and controls:E.g.: moving a character (figure) from left to right is an effect. The definition of the velocity - how fast the character has to be moved - is a control.
The pictures are made out of lines, areas with or without borders, areas with partially existing borders, curves, texts (freely defineable) etc.
There is a variable text defineable which can also be controlled by the effects.
The system is capable to make "in-betweens" between defined keyframes. The whole system consists of 3 subsystems
1. Input of the graphic
2. Definition of effects and controls
3. Film production.

Output is done on colour raster displays ranging in resolution from 48o x 64o to 1o24 x 1o24. Direct filming from screen wouldn't give good resolution and stable colour. It is done via a video-camera. Even in use the system is expanded with new functions. There are many possibilities using computergraphic for animation. This system was mainly developed at the Institut für Datenverarbeitung, TU Wien.

3.2) Computercarthographics:

The Municipal Council of Vienna, Department for Dataprocessing, (DI Koloseus, DI Wilmersdorf) developed a rather big system, which is capable of storing the whole city map of Vienna in a very detailed manner. It is possible to connect geometric-oriented data bases to the graphical objects.In the last years nearly the whole city was digitized. This department also takes care of statistical information needed by the municipal authority. It is produced in form of business graphic charts.

Another Institution is the "Amt für Vermessungswesen" (Dr.Zimmermann). It is digitizing the territory of Austria. Its aim is to get the land-register informations into this database. Out of this information the official maps are produced.

3.3) Business graphics:

Besides the foreign business-graphic systems in use, an interesting development is going on in business and representation graphics. In the university area the Institute for Statistics (Prof.Viertl, DI Guttmann) is developing graphical output methods for time series and other statistical data. It is a similar approach like business graphic. A new system is announced by the company "Sysgraph". Due to this activities, ACGA (see chapter 4) offers one day courses in this field starting in autumn 82.

3.4) ground glass joint design program:

As an example of an industrial order a ground glass joint design program was developed. An example is shown in the figures.

3.5 Process control for power supply

In the power supply companies in Austria there is a movement to use more full graphic output instead of semigraphic displays in process control. An installation in Linz, OKA, made 1979, gives the managing person all consuming and producing data in a graphic representation. The user decides on the type of diagram. The system then gives the picture, which is updated every minute or 5 minutes. The controlsystem not only shows the data which are already available. There is an extrapolation done for all important data.

4. Standardization of Computer Graphic Systems

The Austrian Standard Institute "Österreichisches Normungsinstitut") has an active group which is working on computer graphics. Due to the reason that Austria is not big enough for its own graphic standard, this group supports international activities. e.g. the graphical kernel system (GKS) which is now ISO draft standard is supported within ISO, but also discussed and published in Austria, GKS was originally a German draft. There was some Austrian-German cooperation on GKS. The Austrian relation to GKS exists since this time. The future plans for standards are not detailed. But it looks like, if the work will continue on a graphical metafile and a device driver interface.

5. Organisations supporting CG and CAD

5.1) The Austrian Computer Society (OCG)

The OCG (Österreichische Computer Gesellschaft) is an organisation supporting all activities in the area of computing (President: Dr.N.Rozsenich). The OCG is based upon personal and institutional members. OCG is supporting several special interest groups (working groups). There is also one for Computergraphics and CAD. This working group (Arbeitskreis) is organizing lectures, tutorials and conferences. The last greater one was "Graphische Datenverarbeitung 79" (Computergraphics 79) which was organized together with TU Wien (Institut für Datenverarbeitung) 1979.
OCG is also a liaison member of Eurographics.

5.2) Austrian Computer Graphic Association (ACGA)

The Austrian Computer Graphic Association was founded in Vienna in December 198o. This association's purpose is to foster the research and the practical use of this fast proceeding computer science in Austria. The association is a member of the World Computer Graphics Association.
The ACGA coordinates its activity with the Austrian Computer Association. The major

evidence of the ACGA's activity is the publishing of a bymonthly publication, the CAD, and the organization of seminars and conferences.

6. Conclusion

Computergraphic and CAD are fast growing branches in Austria. This paper could not cover all the projects going on now. The activities which have been mentioned concern subjects which are completely or at least to a great extent designed and programmed in Austria.

Of course, there is a lot of software - especially in CAD - coming from abroad. It is my opinion there should be a balanced relation between development and buying. This depends very much on the particular application.

E.g.: It needs a lot of good ideas and reason to start a new (3-D) modelling system for CAD. It is here similar situation like for programming languages some years ago. But the daily use computer graphics offers many possibilities for "home brewn" Systems which are not covered by already existing systems on the market. This seems to me the area for national activities. Last but not least, doing so, a very important knowledge is built up step by step. To get the information exchanged among these people is the task of ACGA and OCG. If they are working well - like they did - it will be fine to see you some day in Austria at a graphic conference, tutorial or seminar.

The author's address: DI Johann Weiss
Castellezgasse 4/13
A - 1020 Wien, Austria

7. References

(1) H. Guttmann, J. Weiss: Clipping the View of decentralization in computer graphics, Interactive Techniques in Computer Aided Design, Bologna 1978 Conference proceedings, p 235-p 240.

(2) N. Nagler, J. Weiss: Dezentrale Benutzerführung bei graphischen Systemen, OCG Schriftenreihe Band 4, p 45-p 59.

(3) J. Weiss: Device Driver Interface for Decentral Device Drivers, Eurographics 79, 25.10.-27.10. Bologna, p 252-p 263.

(4) J. Weiss: Decentral Graphic Systems (Decentralization of Workstations) Computer Graphics 80, Conferenceproceedings p 471-p 485.

(5) H. Hillebrand, J. Weiss: Considerations of Extensions of Inputfunctions based on a Graphical Kernel System, Eurographics 80 Conference Proceedings p 21-p31.

(6) G. Reinauer: Der Aufbau von anwendergerechten CAD-Systemen, OCG-Schriftenreihe, 1981, ISBN 3-7029-0169-8 (230 pages)

(7) G. Reinauer: Darstellung von Gebilden unter Verwendung räumlicher Elemente, CAD Heft 15 p 10-p 16.

(8) G. Reinauer: Variantenkonstr. von Maschinenelementen Der Kurbeltrieb, CAD Heft 9 p 2-p 9.

(9) E.Huttar, G. Reinauer: Anwendererfahrungen mit der B-Spline Approximation, CAD-Heft 13 p 2- p 9.

(10) G. Reinauer, K. Sträßler: Konstruktion, Auslegung und Auswahl von Lamellenkupplungen mittels CAD CAD Heft 13, p 16-p 22.

(11) Graphische Datenverarbeitung 79, book OCG Schriftenreihe ISBN 3-85403-004-5, 1979 (344 pages)

8. Pictures

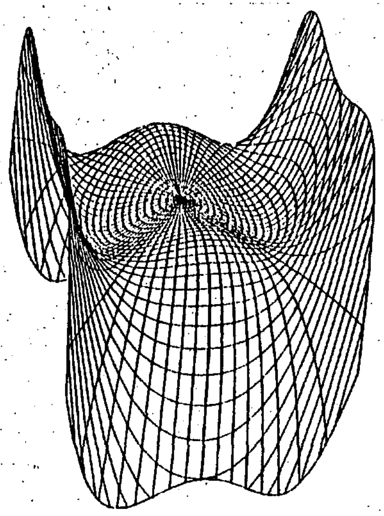


figure 1: Surface representation using general parameter lines (OAW)

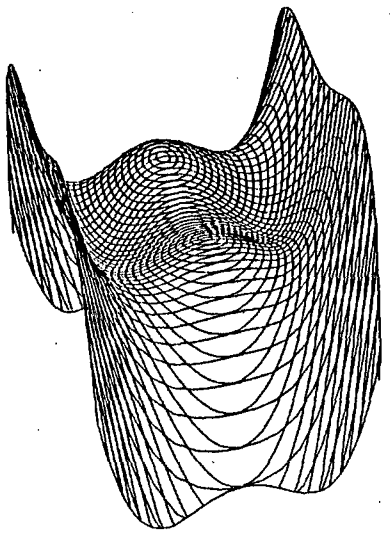


figure 2; Surface representation using general parameter lines (ÖAW)

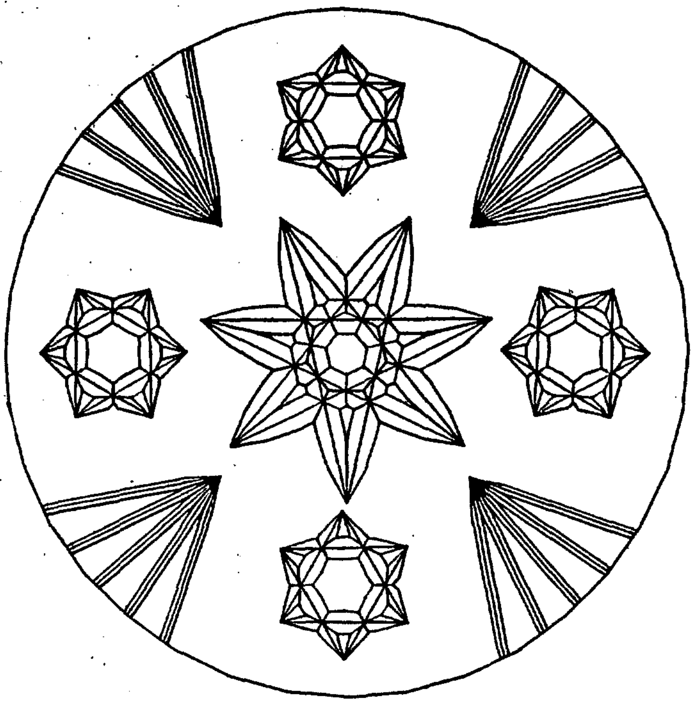


figure 3: Example of ground glass design layout (TU Wien)

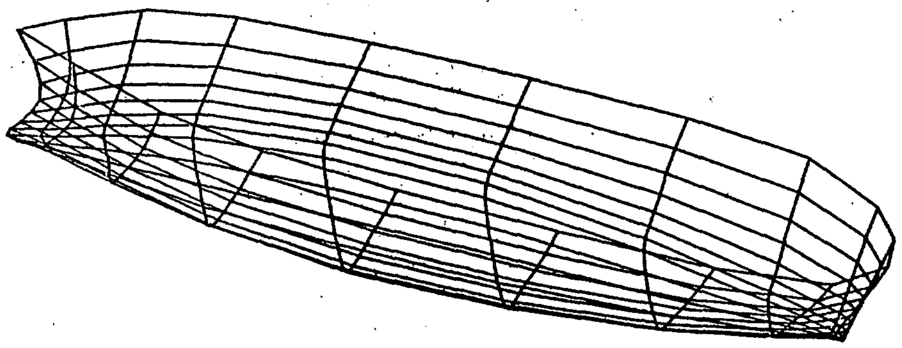


figure 4; Surface modelling for technical applications (TU Wien)

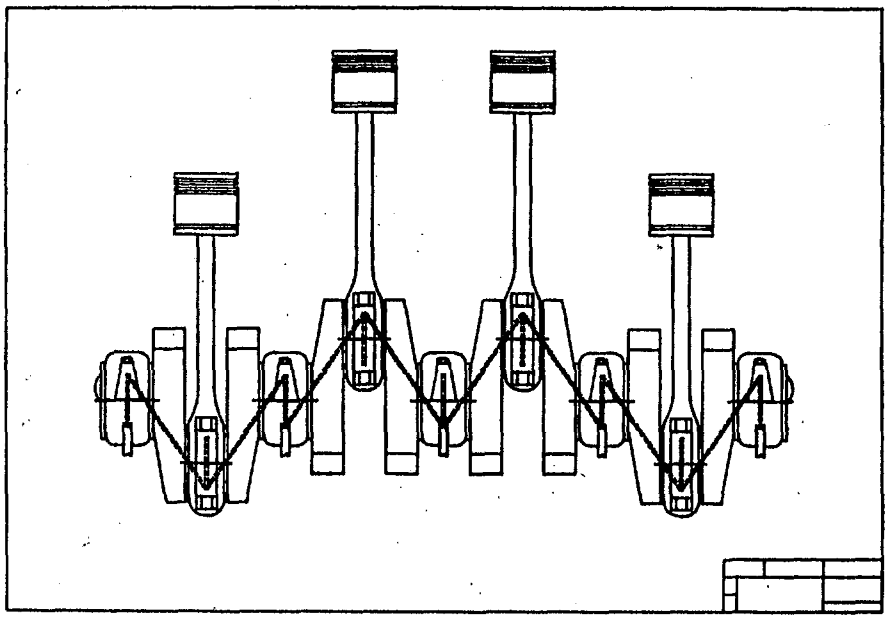


figure 5: Mechanical Engineering: crank shaft drive (TU Wien)

INFORMATICA 2/1982

AN EFFICIENT IMPLEMENTATION OF ADVICE LANGUAGE 2

IVAN BRATKO[1,2], IGOR KONONENKO[1], IGOR MOZETIČ[2]

[1] FACULTY OF ELECTRICAL ENGINEERING, E. KARDELJ UNIVERSITY, TRŽAŠKA 25, LJUBLJANA
[2] JOŽEF STEFAN INSTITUTE, JAMOVA 39 LJUBLJANA

UDK: 519.682.7

Advice Languages were developed to facilitate the implementation of knowledge-based solutions to problems that are combinatorial in nature. This paper reports on an efficient implementation of AL2 (in Fortran), and experiments with its application to chess endgames. In particular, a difficult ending with king and knight vs. king and rook was implemented in AL2 following the structure of an older advice program which was previously written in AL1. The new implementation of this ending, in AL2, is about two orders of magnitude faster than the previous AL1 program.

UČINKOVITA IMPLEMENTACIJA JEZIKA NASVETOV AL2. Jeziki nasvetov olajšujejo reševanje kombinatoričnih problemov s pomočjo aktivne baze znanja. V članku je opisana učinkovita implementacija AL2 in eksperimenti z njegovo uporabo v šahovskih končnicah. Na podlagi prejšnjega programa napisanega v AL1, je bila v AL2 implementirana težka končnica kralja in skakača proti kralju in trdnjavi. Implementacija v AL2 je od prvotne, programirane v AL1, hitrejša za dva reda velikosti.

1. INTRODUCTION

One useful way of looking at problems is: given an initial problem situation, S, find a sequence of actions which transforms S into a final situation, F, such that F satisfies a given goal condition G. The actions can be chosen from a predefined repertoire of permissible operations on problem situations. This general concept is reflected in various formalisms for representing problems, widely used in artificial intelligence, such as state-space, AND/OR graphs, or game-trees (e.g. Nilsson 1980). Examples of practical problems which naturally fit these formalisms are: combinatorial optimization, computer-aided design, automatic programming, failure detection and diagnosis, game-playing, etc.

When solving problems within this paradigm, the domain-specific knowledge has to be used for practical reasons in order to reduce the combinatorial complexity of the search process. One way of expressing knowledge for streamlining the problem-solving process is to specify subgoals which are relevant to the given goal. This can be done in the framework of production rules in the form popularly used in artificial intelligence (e.g . Waterman and Hayes-Roth 1978). A series of rule-based systems, called Advice Languages, has been based on this idea (AL1: Bratko and Michie 1980a; 1980b; AL2: Mozetic 1980; AL3: Bratko 1982a; 1982b). In this paper we describe an implementation, in Fortran, of AL2. AL1 and AL3 were implemented (elswhere) in AI languages POP-2 and Prolog respectively.

In experiments with the Advice Languages, the game of chess has been mainly used as an experimental domain. The adversary nature of games introduces complications, but the idea of breaking the whole problem into subgoals is still viable and rather straightforward. Consider, for example, the ending with king and rook vs. king (KRK for short). A useful piece of advice for this ending would be: in order to mate, first squeeze the opponent's king against the edge, and then try to mate. The whole problem of mating has been reduced into the (easier) subgoals: force the opponent's king to the edge, and, when on the edge, carry out the final phase for mate.

This is, basically, the form in which the knowledge about problems is communicated to the problem-solver via Advice Language 2.

Two nontrivial endgames have been implemented in AL1 and AL3: king and knight vs. king and rook (KNKR) in AL1 (Bratko and Michie 1980b), and king and pawn vs. king and pawn in AL3 (Bratko 1982a; 1982b). As shown by Kopec and Niblett (1980), the difficulty of perfect play in KNKR is beyond the capabilities of chess masters. The performance of an advice program in AL1 for KNKR was, if not perfect, at least comparable to that of human masters.

One trouble with AL1 was its slowness, due to an inefficient implementation in POP-2. To play a move in a nontrivial KNKR position, it took the system typically a few minutes of CPU on a DEC KI-10 processor. We translated the AL1 program for KNKR into AL2 and used it as a benchmark for testing our present implementation of AL2. The present implementation typically uses between a second and a few seconds to play a move in a difficult KNKR position on a DEC KL-10 processor. The pace of play of a few seconds per move is attained by AL2 when searching about 1000 positions in the game-tree. Under regular chess tournament conditions, the player is allowed about three minutes per move. So under tournament conditions, the program would be allowed to search correspondingly larger portions of the game-tree. With, say, 50 times more time per move, the system would be able search some 50 thousand of positions per move actually played across the board. This figure is comparable to the speed of some (not fastest !) tournament chess programs.

The rest of the paper describes the details of AL2 and its implementation, and some experimental results.

2. AL2 DESIGN

The overall structure of the AL2 system is depicted in Fig. 1. The main modules of the system are :

(1) the knowledge base
Knowledge is represented in advice-tables and pieces-of-advice. Each advice-table is specialised as to deal with certain problem-subdomain. According to the current problem-situation it chooses an apropriate advice-list. Advice-list is an ordered list of pieces-of-advice. A piece-of-advice suggests what goal should be achieved next while preserving some other condition. If this goal can be achieved in a given problem-situation then we say that the piece-of-advice is satisfiable.

(2) the search module
The search module takes an advice-list from the knowledge base and tries to satisfy a piece-of-advice in the list, sequentially attempting the pieces one after another, until it succeeds. To satisfy a piece-of-advice is to find a specific subtree of the game-tree called forcing-tree. Forcing-tree may be interpreted as a strategy that guarantees us the achievement of the goals prescribed in the first satisfiable piece-of-advice from the advice-list.

(3) the interactive interface
The interface facilitates the communication between the user-player and the system. It accepts a forcing-tree from search module and interprets it as our strategy for playing our moves. Besides this it can show useful statistics and enable trace during the search for a forcing-tree.

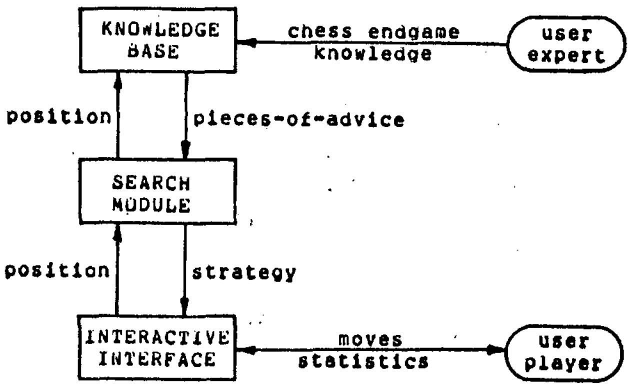


Fig. 1: The basic structure of the AL2 system.

It should be noted that the basic structure of AL2 as described above is almost identical to that of the AL1 system (implemented in POP-2 at the University of Illinois, outlined also in Bratko and Michie 1980).

In the following paragraphs we describe the modules of AL2 in more details.

2.1. The knowledge base

Advice-table is a set of rules of the form

if precondition *then* advice-list

If more then one precondition is satisfied then simply the first rule is chosen.

A piece-of-advice is a five-tuple

(HG, BG, UMC, TMC, MAXD)

where HG and BG are predicates on positions called holding-goal and better-goal respectively, UMC and TMC are predicates on moves, called move-constraints for us and for them respectively and MAXD is a depth-bound of a forcing-tree. By "us" we mean the side to move in a given position, either white or black, and by "them" the opponent of "us".

Move-constraints can besides a mere selection of a subset of legal moves prescribe an ordering on the moves that are selected.

We say that a piece-of-advice is satisfiable in a given position if and only if :

(1) the "us" side can force the achievement of the better-goal in less then MAXD plies, while
(2) during the play toward the better-goal, the holding-goal is never violated, and
(3) the "us" side always chooses its moves only from the set of moves that satisfy UMC, and
(4) the "them" side's choice of moves is limited only to the moves that satisfy TMC.

Goals may also be expressed by the satisfiability of some other piece-of-advice, as follows :

Better-goal (or holding-goal) is achieved in a position if the other piece-of-advice is satisfiable in the same position.

Take, as an example, the king and pawn vs. king ending. A piece-of-advice for the stronger side that suggests simply "push the pawn to the queening square" can be defined as :

(pawn_safe, pawn_queened, pawn_moves,
king_macro_move_to_queening_square, 9_plies)

If we take for example the position in Fig. 2, it is obvious that this piece-of-advice fails, because the black king can capture the white pawn.

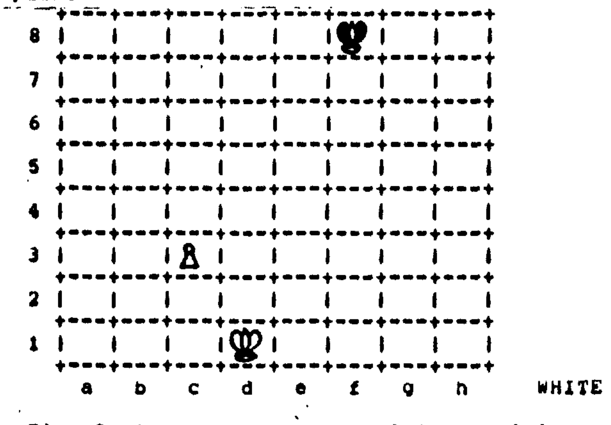


Fig. 2: An example of nontrivial position for white to move and win. After the natural move K d2 black can draw by K e7. The only correct move is K c2 !.

2.2. The search module

Control structure chooses which piece-of-advice from advice-list to apply next. Then it tries to satisfy it by simple depth-first search. If and only if a piece-of-advice is satisfiable in a given position there exists a forcing-tree and the search module finds it.

<u>Forcing-tree</u> is a subtree of the game-tree rooted in a given position, such that :

(1) every node, except the root-node, satisfies HG,
(2) every nonterminal node satisfies not BG,
(3) every terminal node satisfies BG,
(4) there is exactly one move from every nonterminal us-to-move node; that move must satisfy UMC,
(5) there are all legal moves from every them-to-move nonterminal node that satisfy TMC,
(6) the lenght of the longest path must not exceed the depth-bound MAXD; the root-node is supposed to be at depth 0.

We say that in a position, from which there is no legal "us" move that satisfies UMC, the better-goal can not be achieved. But if there is no legal "them" move that satisfies TMC, the position is a terminal node of a forcing-tree.

In Fig. 3 there is an example of forcing-tree generated for the position in Fig. 2. Here, a piece-of-advice for white that suggests how to get the king in front of the pawn is satisfied.

```
WK c2   BK e7   WK b3   BK d6   WK b4   BK c6  /
                                        BK d5  /
                        BK e6   WK c4   BK d6  /
                        BK d7   WK b4   BK d6  /
                                        BK c6  /
        BK f7   WK d3   BK e7   WK c4   BK d6  /
                        BK e6   WK c4   BK d6  /
        BK e8   WK b3   BK e7   WK c4   BK d6  /
                        BK d7   WK b4   BK d6  /
                                        BK c6  /
```

Fig. 3: The forcing-tree for white for the position in Fig. 2. On any black's move white has an answer that brings his king in front of the pawn, so that black cannot eventually prevent the queening.

```
 ADVICE  NODES    1   2   3   4   5   6   7   8
-------------------------------------------------
   1       19     1   5   1   5   1   4   1   1
   2       58     3   9   6  15  10  15   0   0
 NODES =   77         (CPU time = 0.18 sec)
```

Fig. 4: Statistic typed during game-tree search for white in the position from Fig. 2. The first piece-of-advice "push_pawn" fails in 8 plies, but the second "bring_king_in_front_of_pawn" succeeds.

2.3. The interactive interface

The interface can interpret some dozen commands from the user, to start various functions of the system. The search module triggers the searching for a forcing-tree on the user-player request. This one is then interpreted as our strategy and executed by interactive interface in across-the-board play.

The user-expert may specify the system to play in the "strong-mode" or in the "weak-mode". In the first case the strategy is executed up to a terminal node in the forcing-tree. On the other hand if the "weak-mode" is chosen it means that there is no point in following the current strategy up to its end, and a new forcing-tree is generated immediately on the next move.

3. IMPLEMENTATION

A Fortran implementation of the AL2 system was developed on a PDP-11/34 at "Jozef Stefan" Institute, Ljubljana. Later it was installed on a DEC-10 at the University of Edinburgh and at the E. Kardelj University at Ljubljana. The development of the AL2 programming package which contains about 3000 Fortran commands required about a half a year work of one of us.

Fortran is not very convenient language for such a kind of programming task, but at the moment when the work was initiated no other high-level programming language was available. On the other hand the Fortran implementation has turned out to be much more efficient then the known implementations in others, more powerful languages (POP-2, Prolog). For example, the current Fortan implementation on DEC-10 is about 100 times faster and even on PDP-11/34 about 10 times faster then POP-2 implementation of the AL1 system. The reason is only partly due to the 4 times faster processor in the first case. The efficiency is measured in the number of positions generated per second during the game-tree search. The AL2 system examines between 500 and 1000 positions per second, depending on the complexity of predicates in pieces-of-advice.

4. EXPERIMENTS AND RESULTS

Two major experiments conducted with AL2 were concerned with the implementation of two endgames: King and pawn vs. king (Mozetic 1980), and king and knight vs. king and rook (KNKR). Here we present the second experiment, the more complex test of AL2: a KNKR advice program. Our AL2 program for KNKR is, in fact, a translation of an advice program for KNKR in AL1 (Bratko and Michie 1980b).

The following is a summary of basic ideas underlying the KNKR advice program. The program plays for the weaker side, i.e. the knight's side. If a KNKR position is theoretically won for the rook's side, the win is based on two motifs: first, mating threats, and second, capture of the knight. The latter is possible either after a short, forced combination (e.g. based on pinning the knight), or after a long-term strategic plan which consists of separating the knight and its friendly king, and subsequently surrounding the knight. If neither of the two basic motifs can be exploited then the position is theoretically a draw. The advice program for the knight's side is based on negating the attacker's goals. The corresponding broad advice is as follows:

1. Avoid mate: keep the king as far from the edge as possible; keep the king as far from the corner as possible.

2. Preserve the knight: keep the knight as near the friendly king as possible and far from the enemy king.

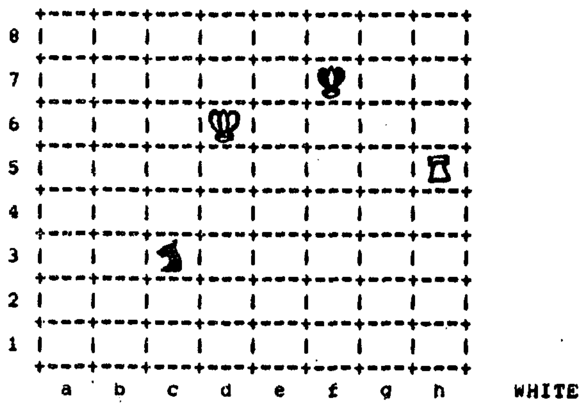
The details of this broad strategy are incorporated into an advice table comprised of 5 columns (which corresponds to 5 production rules) and 16 pieces-of-advice.

As our KNKR advice program is a rather straightforward translation of the original program in AL1 we here present only examples of its behaviour, specially with respect to its efficiency.

In the following examples we give the CPU time the system spent for playing the move, and for some positions statistics on the number of positions searched (for separate pieces-of-advice tried before the move was found, showing also these numbers over the depth of positions in the game-tree; depths correspond to columns labeled 1-6).

Example 1:

This is a game between the advice program and an international chess master. The starting position of the game is in the following diagram. This position is won for white, as known from databases of complete KNKR positions. White can force the capture of the knight against best defence in 24 moves.



1. K e5 , N a4 ! (1.88 sec)

(The best defence, rather hard to find.)
Statistics on search follows.)

ADVICE	NODES	1	2	3	4	5	6
1	14	14	0	0	0	0	0
15	172	14	25	30	31	21	51
14	172	14	25	30	31	21	51
13	423	7	54	47	86	30	199
NODES =	781						

2. R h7+, K e8 (2.54 sec, 1078 nodes)

3. K d6 , N b6 (2.46 sec, 1074 nodes)

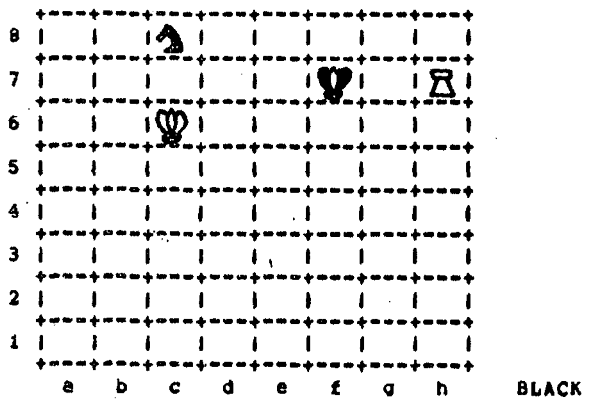
4. R h8+!, K f7 (0.98 sec, 389 nodes)

5. R h4 , N c8+ (1.98 sec, 880 nodes)

6. K d7 , N b6+ (0.54 sec, 218 nodes)

7. K c6 , N c8 (1.44 sec, 594 nodes)

8. R h7+, ...



8. ... , K e6? (5.78 sec)

ADVICE	NODES	1	2	3	4	5	6
1	6	6	0	0	0	0	0
15	828	6	65	18	100	71	568
14	828	6	65	18	100	71	568
13	896	6	105	33	113	71	568
12	93	1	18	7	67	0	0
NODES =	2651						

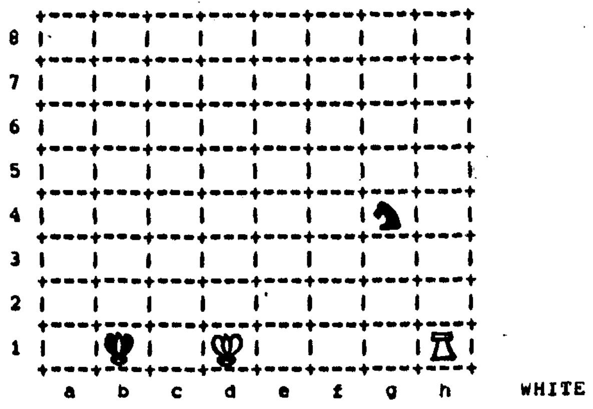
Up to this point both the program and the master played optimally. This is the first mistake in this game which considerably shortens the resistance of the weaker side. Correct was K f6!.

9. R h6+, K e5 (0.90 sec, 464 nodes)

Play was stopped here as the white's win is clear by: 10. K d7, N a7 11. R a6, N b5 12. R a5 etc.

Example 2:

In the following game against another chess master the program again defended a lost position. This time, after a master's mistake the program saved the game.



1. R h4!, N f6?! (2.32 sec)

ADVICE	NODES	1	2	3	4	5	6
1	39	9	6	24	0	0	0
15	980	1	17	13	186	91	672
NODES =	1019						

The move N f6 was good, but not optimal. Best was N e5.

2. K d2 , K b2 (1.00 sec, 417 nodes)

3. R d4?

After this mistake the position is theoretically drawn. Correct was K d3.

3. .... , K b3 (1.06 sec, 434 nodes)

4. K d3 , N e8 (0.74 sec, 287 nodes)

5. R d7 , K b4 (0.88 sec, 388 nodes)

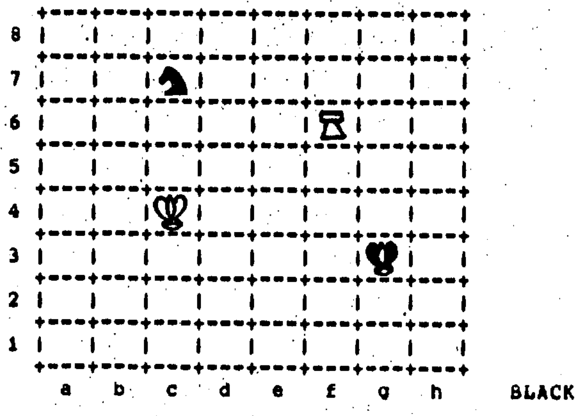
6. K d4 , K b5 (1.02 sec, 448 nodes)

7. R e7 , N d6 (1.66 sec, 717 nodes)

8. K d5 , N c8 (0.64 sec, 241 nodes)

At this point draw was agreed. Position is obviously drawn.

Example 3:

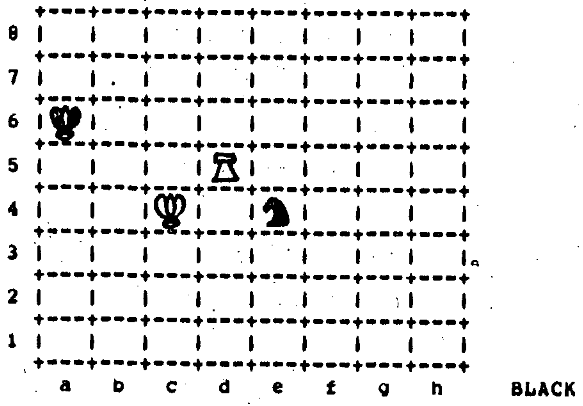


Program plays: K h4 (3.60 sec)

ADVICE	NODES	1	2	3	4	5	6
1	11	11	0	0	0	0	0
15	196	11	20	15	38	16	96
14	349	11	42	28	84	21	163
13	956	5	62	36	201	48	604
NODES =	1512						

The move played by the program K h4 is the only, study-like drawing move. The natural move K g4 would lose after R c6, N d8, R e6.

Example 4:

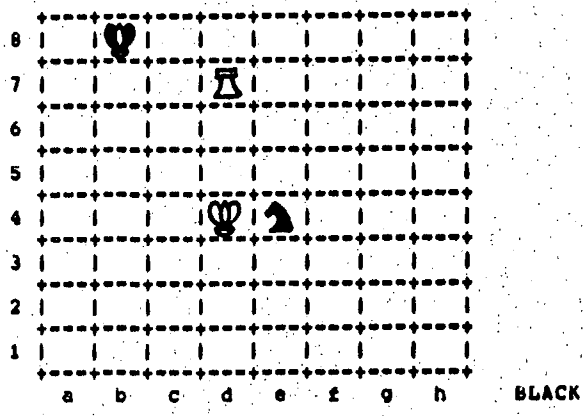


Program plays: K b7 (2.06 sec)

ADVICE	NODES	1	2	3	4	5	6
1	13	11	2	0	0	0	0
15	706	4	37	29	142	49	445
NODES =	719						

The move K b7 is again surprizing, but only correct. The intuitive move K b6 loses after R d4 forcing the knight far from its own king.

Example 5:

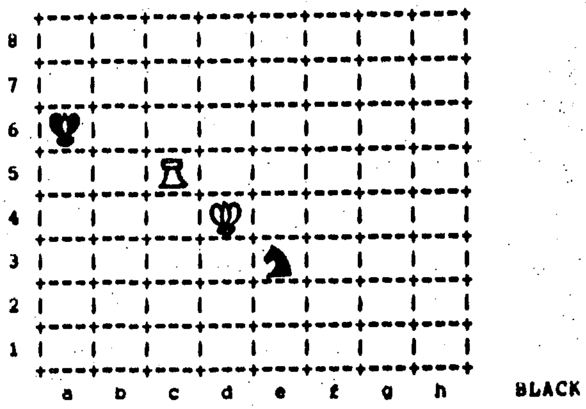


The program plays the only drawing move:

K c8 (2.24 sec)

ADVICE	NODES	1	2	3	4	5	6
1	10	10	0	0	0	0	0
15	897	5	32	18	152	58	632
NODES =	907						

Example 6:



The program finds the pretty and only correct:

K b6 (0.86 sec)

ADVICE	NODES	1	2	3	4	5	6
1	13	11	2	0	0	0	0
15	271	3	19	27	34	26	162
NODES =	284						

5. CONCLUSION

The implementation of AL2, presented in this paper, is by about two orders of magnitude faster than the previous implementation of AL1. AL2 is also currently the only implementation of Advice Languages whose legal move generator is complete. Others are limited to certain subgames only.

AL2 as a language is very similar to AL1, it only introduces small but useful improvements. The major improvement over AL1 thus lies in AL2's efficiency. On the other hand, AL3 is, compared to AL1 and AL2, as a nonprocedural programming language, much more flexible and general. However, its present inefficiency makes it a rather impractical tool for solving realistic problems. One attractive possibility is to combine the AL3's linguistic power and AL2's efficiency. Thus the high level reasoning would be done in the domain of AL3. The result of this would be 'plans', or pieces-of-advice, which would be passed to AL2 for concrete checking by search.

REFERENCES

1. Bratko, I. (1982a) Knowledge-based problem-solving in AL3. Machine Intelligence 10 (eds. D.Michie, J.H.Pao) Ellis Horwood and Wiley.

2. Bratko, I. (1982b) Advice and planning in chess endgames. NATO Symposium on Artificial and Human Intelligence, Lyon, Oct. 1981. To appear in Artificial and Human Thinking (eds. A.Elithorn, R.Banerji, in preparation).

3. Bratko, I., Michie, D. (1980a) A representation for pattern-knowledge in chess end-games. Advances in Computer Chess 2 (ed. M.R.B.Clarke) Edinburgh University Press.

4. Bratko, I., Michie, D. (1980b) An advice program for a complex chess programming task. The Computer Journal, Vol. 23, pp. 353-359.

5. Kopec, D., Niblett, T. (1980) How difficult is the play in the KNKR ending? Advances in Computer Chess 2 (ed. M.R.B.Clarke) Edinburgh University Press.

6. Nilsson, N.J. (1980) Principles of Artificial Intelligence, Tioga Publishing Co.

7. Mozetic, I. (1980) User's manual for the AL2 system, An AL2 strategy for king and pawn vs. king. Research Memorandum MIP-R-130, Machine Intelligence Research Unit, University of Edinburgh.

8. Waterman, D.A., Hayes-Roth, F. (1978) Pattern-directed Inference Systems. New York: Academic Press.

INFORMATICA 2/1982

TEHNOLOGIJA ELEKTRONSKIH RAČUNARSKIH SISTEMA

MARIJAN M. MILETIĆ

DO ISKRA – DELTA

UDK: 681.3.02

ELECTRONIC COMPUTERS TECHNOLOGY

This article briefly describes seven levels of computer technologies from manufacturers point of view. These are: semiconductors, modules, peripherals, systems hardware integration, basic software, applications software and services.

Manufacturing process in DELTA is presented.

Članak opisuje tehnološke nivoje računara opšte namene sa aspekta proizvođača.

To su: poluprovodnici, moduli, periferne jedinice, integracije sistema, osnovna programska oprema, aplikacioni programi i servisi.

Prikazan je proizvodni proces u RO DELTA.

1. UVOD

U ovom članku daćemo kratak prikaz osnovnih tehnologija elektronskih računarskih sistema opšte namene sa aspekta proizvođača računara.

Nastojačemo da koliko je moguče izostavimo numeričke podatke i različite dijagrame. Na slici 1. je dat prikaz poteka proizvodnje računara u RO DELTA.

Pojava jeftinih mikroprocesora daje snažan impuls rešavanju aplikaciono usmerenih računarskih sistema. Proizvođači integrisanih kola su preuzeli i deo pripreme osnovne programske opreme te značajno olakšali široko uvođenje ovih najkompleksnijih poluprovodničkih komponenti.

Veliki broj programskih timova i samostalnih kuća lansira programske pakete opšte namene i osvaja mikroračunarima sve veči deo kolača namenjenog sistemima opšte namene.

To je rezultiralo u konačnom "Priznanju" mikroračunara kao nove klase računara od strane IBM-a, DEC-a i ostalih.

Značajno je još uočiti sve tesniju integraciju računara i telekomunikacija, kako u digitalizaciji komunikacija tako i u kompleksnim mrežama računara sa raznovrsnim bazama podataka.

I pored ova dva značajna trenda u kompjuterskoj tehnici, principi proizvodnje računarskih sistema se nisu bitnije menjali skoro dvadeset godina i ne očekuje se drastičnija promena ni u sledečih deset.

2. TEHNOLOŠKI NIVOJI

Sa aspekta proizvođača elektronskih računara moguče je izlučiti pojedine faze pri kompletnoj proizvodnji sistema.

To su sledeči specifični tehnološki nivoi:

1. poluprovodnici,
2. moduli,
3. periferne jedinice,
4. integracija mašinske opreme,
5. osnovna programska oprema,
6. aplikativna programska oprema,
7. servisi.

Ovih sedam vertikalnih nivoa čine zaokruženu celinu u proizvodnom ciklusu, mada se pojedini proizvođači specijalizuju samo za pojedine horizontalne nivoje.

Ove nivoje treba posmatrati dinamički sa migracijom na niže. Tipični primeri su VLSI kola koja preuzimaju ranije funkcije više modula, implementacija operativnog sistema i prevodioca u ROM-u, koriščenje daljinske diagnostike u servisiranju itd.

Karakteristična je velika primena računara u samoj proizvodnji računara.

3. POLUPROVODNIČKA TEHNOLOGIJA

Performanse poluprovodničkih elemenata u največoj meri određuju i performanse kompletnih elektronskih računara.

Danas svi veči proizvođači računara poseduju sopstvene kapacitete za razvoj i proizvodnju integrisanih kola. Interesantno je, da su mogučnosti nekih od njih (IBM, FUJITSU) bolje od mogučnosti primarnih proizvođača komponenti (TI, INTEL).

Osnovna svrha sopstvene poluprovodničke tehnologije proizvođača računara nije se bitno promenila od

IBM-ovog poteza 60-ih godina pri uvođenju Serije 360.

Na silikonskoj pločici lakše je sakriti specifična organizaciona rešenja kompleksnih arhitektura čime se postiže veća vremenska distanca od "kopirajućih" konkurenata.

Dominante poluprovodničke tehnologije visokog nivoa integracije su nizovi logičkih vrata u bipolarnoj tehnici za logičke funkcije i NMOS za memorijska kola.

Trendovi su ka korišćenju galijum - arsenida za veće brzine i submikronskih rezolucija za veće gustine.

Kao jednu od visoko zahtevnih tehnologija, karakteriše je brz razvoj i velika kapitalna ulaganja.

Izražena je vodeća uloga Amerike i Japana sa grčevitim nastojanjima zapadne Evrope da održi korak pomoću infuzija državnog kapitala.

4. TEHNOLOGIJA MODULA

Modul baziran na štampanim kolima je osnovni oblik integracije različitih elektronskih i električnih elementa sa odgovarajućim priključcima za međusobna povezivanja.

Kao i kod poluprovodnika, uočljiv je trend ka primeni većih štampanih ploča sa velikom gustinom komponenti radi smanjenja manje pouzdanih, eksternih kontakata i veza.

Dvoslojna štampa sa metaliziranim rupama je minimalni zahtev, a sve više se koristi četvoroslojna sa velikim, unutrašnjim površinama srednja dva sloja namenjena napajanju.

Skupa sa poluprovodničkom i magnetskom tehnologijom, tehnološki postupci u proizvodnji štampanih pločica su veoma "prljavi" zbog korišćenja jako otrovnih kemikalija.

Radi smanjenja proizvodnih troškova teži se ka što višem stepenu automatizacije u projektovanju, testiranju i ulaganju komponenata.

Deo ljudskog rada na modulima je veći nego kod poluprovodnika što se odražava u sporijem padu cena.

5. TEHNOLOGIJA PERIFERNIH JEDINICA

Ovde možemo diferencirati dve dodatne tehnologije: magnetnu i finu mehaniku.

Magnetna tehnologija je neophodna za izradu fiksnih disk jedinica koju su glavna eksterna memorija računara.

Nedefinisana je sudbina memorija sa mehurovima, dok feritne memorije sve više gube tržište.

Kod magnetnih traka se očekuje znatno veće gustine pakovanja informacija jer su mehanički limiti brzina već dostignuti. Time će se magnetna traka i dalje održati kao najjeftiniji medij za arhiviranje i razmenu podataka.

Tehnologija fine mehanike je sve značajnija u eri minijaturizacije, a odlučujuća kod štampača.

Ekranski terminali su najsličniji proizvodima široke potrošnje (televizorima) sa klasičnim zahtevima po plastici i vakumskoj tehnici katodnih cevi.

Karakteristična je sve šira primena mikroprocesora u kontrolnoj elektronici perifernih jedinica.

Apetiti korisnika su u stalnom porastu, posebno za diskovima, te je proizvodnja perifernih jedinica značajan izvor prihoda proizvođača računara.

Pored klasičnih kompjuterskih perifernih jedinica, ne treba zaboraviti široki spektar komunikacionih jedinica i industrijske periferije, no ovi su najčešće na nivoju modula.

6. INTEGRACIJA MAŠINSKE OPREME

Tehnologija ovog nivoa sastoji se u nizu procedura za električno, elektronsko i programsko testiranje računarskog sistema.

Integracija mašinske opreme je prva faza u kojoj se pojavljuju fizičke konture budućeg računarskog sistema. Ova faza je neophodna zbog kompleksnosti sastavnih podsklopova mašinske opreme: centralnog procesora, glavne memorije, kontrolnih i perifernih jedinica, te niza različitih kombinacija istih u finalnom sistemu.

Svaki od ovih podsklopova moguće je velikoserijski proizvoditi u nekoliko standardnih verzija.

Ovde su najizraženiji problemi mehaničkih rešenja podsklopova, napajanja, hlađenja i povezivanja.

Da bi se obezbedila pouzdanost celog sistema vrši se integracija najosnovnije konfiguracije za svakog pojedinačnog korisnika sa obimnim testiranjima radi otklanjanja eventualnih međusobnih smetnji u skupnom radu.

Kako ova faza stvara dodatne troškove u proizvodnji, kod manjih računarskih sistema rizikuje se "integracija pri otpremi", te se ceo sistem prvi put "oživljava" u instalaciji kod korisnika.

Kod mikroračunarskih sistema ostavljena je mogućnost samom korisniku da izvrši integraciju najrazličnije mašinske opreme.

7. OSNOVNA PROGRAMSKA OPREMA

Optimalan rad mašinske opreme kontrolisan je operativnim sistemom.

Proizvođači računara često nude više operativnih sistema koji su prilagođeni različitim režimima korišćenja računara.

Operativni sistem doživljava česte promene pre svega zbog novih komponenata mašinske opreme, no uvek se nastoji da se održi kompatibilnost na nivoju korisničkih programa u mašinskom jeziku.

U osnovnu programsku opremu još spadaju prevodioci različitih računarskih jezika, programi za rad sa bazama podataka, programi za računarske komunikacije i niz pomoćnih, često korišćenih programa.

Dosta se očekuje od novih jezika za strukturalno programiranje (ADA), te relacionih baza podataka.

Svi proizvođači računara se trude da ponude što širi spektar osnovne programske opreme radi veće unifikacije obrade podataka na jednoj familiji računara.

8. APLIKACIONA PROGRAMSKA OPREMA

Aplikacionu programsku opremu većinom razvija korisnik sam, dok proizvođači najčešće nude programske pakete prethodno razvijene kod nekog korisnika ili za interne potrebe.

Sve većom primenom računara u najrazličitijim oblastima, prevaziđena je podela aplikacija samo na poslovne i tehničke, pojavom značajnih programskih segmenata u proizvodnji, transportu, obrazovanju, obradi tekstova, automatizaciji kancelarijskog poslovanja i komunikacijama u širem smislu.

Kod aplikacionih programa je veoma izražen nizak stepen unifikacije i standardizacije sa več poslovično niskom produktivnošču programera, što rezultira u porastu cena programske opreme i anuliranju pada cena mašinske opreme.

Upravo na području aplikacione programske opreme neophodan je tehnološki proboj, da bi se omogučila najmasovnija primena mikroprocesora.

9. SERVISI

Računarski sistemi zbog svoje kompleksnosti zahtevaju dosta servisnih usluga od strane proizvođača.

To su pre svega održavanje mašinske opreme, programi školovanja, te pomoč kod uvođenja sistema. Ovde možemo svrstati i prodajne aktivnosti koje traže jedan viši nivo znanja komercijalista.

Održavanje mašinske opreme se pospešuje ugradivanjem dijagnostičkih funkcija još u fazi projektovanja sistema, daljinskom dijagnostikom kvarova te poboljšanjima operativnog sistema za rad sa delimično ispravnom mašinskom opremom.

Servisna organizacija obično uključuje i obimnu logistiku rezervnih delova, programskih produkata i potrošnog materijala.

Školovanje treba da zadovolji interne potrebe po permanentnom obrazovanju, te mnogostruke zahteve korisnika.

Pomoč kod uvođenja sistema ide paralelno sa prodajnim aktivnostima i ogleda se planiranju računarskih resursa, prostornim, klimatskim i energetskim zahtevima računara i instalaciji računara.

Korisniku se najčešče nude kompleksne inženjering usluge koje obuhvataju i značajan deo programski pomoči.

10. ZAKLJUČAK

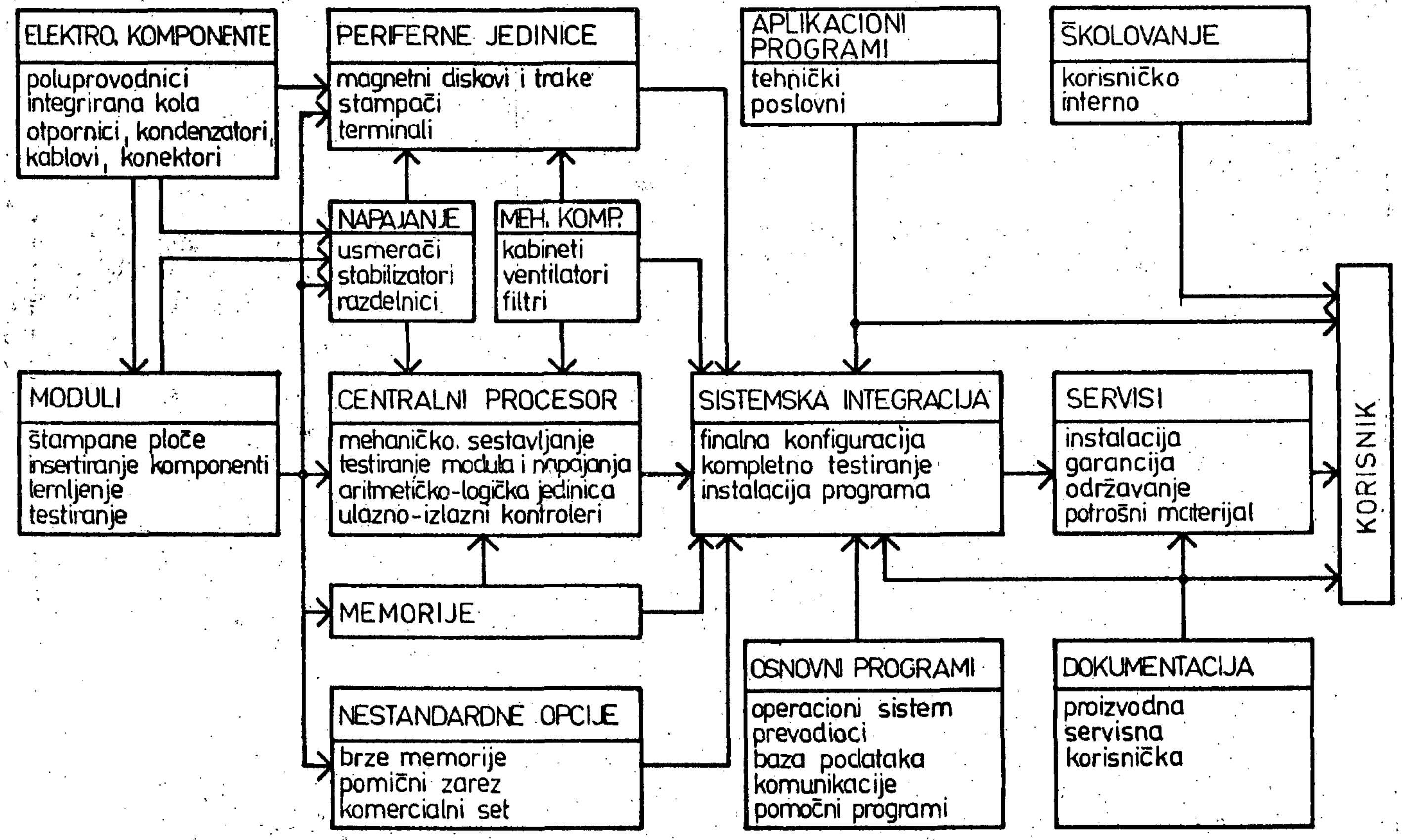
Dinamička industrija elektronskih računara ima jedno od glavnih mesta u budučem, informacionom društvu.

Petu generaciju elektronskih računara 90-ih godina baziranu na principima veštačke inteligencije biče moguče realizovati tehnološkim napredkom na svih sedam nivoa u cilju permanentnog poboljšanja odnosa cene i performansi sistema.

Ovaj sažet prikaz tehnologija elektronskih računara dat je sa željom da omoguči potencijalnim, jugoslovanskim "proizvođačima" računara da sagledaju svoje odgovarajuče mesto u jednoj specifičnoj industriji, koja kod nas preživljava bezbroj porođajnih muka.

Literatura:

C.G. Bell et al. COMPUTER ENGINEERING, Digital Press 1978



SHEMA PROIZVODNJE ELEKTRONSKIH RAČUNARSKIH SISTEMA U RO ISKRA-DELTA

INFORMATICA 2/1982

MEETING THE CHALLENGE FOR INFORMATION SYSTEMS IN THE 80'S

A. MILTON JENKINS

INDIANA UNIVERSITY

U DK: 681.3:007

Izziv informacijskih sistemov v osemdesetih letih. Težave in tveganje v povezavi z upravljavskim informacijskim sistemom (MIS) so znane že iz sedemdesetih let. Današnja organizacija MIS predvideva (več ali manj standardno) pet članov. Projekt vodi navadno poseben pomočnik direktorja (tudi podpredsednik podjetja), kateremu poročajo (odgovarjajo) štirje izkušeni vodje (slika 1). Administrator baze podatkov odgovarja za stanje podatkovnih virov, sistemski upravnik pa za ustrezno materialno, programsko in komunikacijsko opremo organizacije. Operacijski upravnik odgovarja za obdelavo vseh aplikativnih paketov in upravnik razvoja aplikativnih sistemov za pokrivanje potreb vseh uporabniških organizacij. Članek razčlenjuje vrsto bistvenih nalog sodobnega MIS.

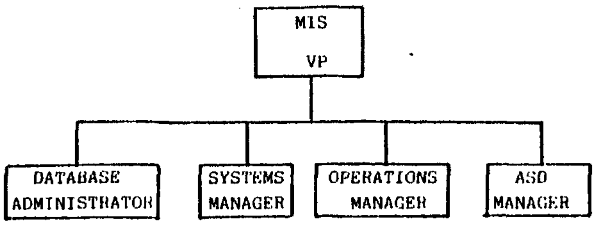
(Urednik)

Introduction

The hazards and difficulties associated with management positions in the MIS department of any organization are well known. During the early and mid sixties computer systems management was the most difficult position. A major contributor to this difficulty was the rapid technical advances in both the hardware and systems software. During the late sixties and through the mid seventies the Data Processing (DP) Manager's position was the "hot seat" in MIS. Several surveys reported "life-expectancies" of less than one year for DP managers during this time. The application system explosion is usually credited with being the largest contributor to the problems DP Managers faced during this time. The most difficult managerial position in MIS durig the late seventies was that of the Database Administrator. The rapid expanding supply of corporate data from transactional and operational application systems, coupled with an increasing awareness that data and information could and should be viewed as organizational resources, contributed toward making the Database Administrator's position difficult during this time. The rapid technological advances in database management systems and the proliferation of new software products by both hardware vendors and newly evolving software houses farther contributed to the problems of managing the organization's data resources.

From 1960 to 1980 the typical MIS organization chart changed considerably, reflecting the changing role of MIS in the organization, the changing management needs within the MIS organization, and the influences of computer and systems technology. Today the top levels of most MIS organization charts appear as illustrated in figure 1. The senior MIS manager today is usually at the vice president level and has four senior managers reporting directly to him/her. The Database Administrator is responsible for managing the organizationb' data resources and increasingly plays a major role in providing consulting and training services to the user community. The Systems Manager is responsible for providing appropriate hardware, software and communications facilities for the organization.

The Operations Manager (formally called the DP Manager) is responsible for processing of all applications systems in the organization. The ASD Manager is responsible for meeting the needs of all user organizations for application systems.



Typical Early 1980' MIS Organization Chart

FIGURE 1

The current managerial "hot seat" in MIS is the Applications Systems Development (ASD) Manager. The biggest task facing this manager today is meeting the challenge to deliver effective and efficient information systems to any user group in the organization that "requests" an information system. That challenge is the topic of this paper.

The Major Challenges

The major challenges to the ASD Manager arise from three sources: 1) technological advances in computer hardware, software, and methodologies, 2) increasing organizational awareness of the need to manage the information resource, and 3) increasing management awareness of the value of information in increasing their own and their subordinates' productivity. This paper will focus on those issues that are most common to all ASD Managers, recognizing that the sequence and level of concern over the issues may vary significantly across organizations. These issues will be presented in the form of questions - questions that the ASD manager will have to find answers for in the near future.

How do I manage the growing backlog of new application system development projects?

A frequent first reaction to this question is to ask why such a backlog exists and why is it growing. These are natural questions given that hardware costs in 1980 are 1% of what they were in 1960 and that programmer productivity is five times what it was in 1960. The explanation of this apparent paradox lies in the fact that user expectations are growing faster than development capacity, even with increasing development budgets. Another factor explaining the lag in development capability is that a high proportion of most development budgets are expended on maintenance of or enhancements to existing systems. Reports of maintenance costs at eighty percent or more of the total development budget are common.

Let's look at the feasibility of some typical solutions considered by ASD Managers. The first solution set is based on expanding the MIS resource base.

Since the largest capacity constraint is the programmer/analyst, why not just hire more programmers. Intuitively this response is appealing. But even if the development budget could be expanded to meet increase in the development staff, are these skilled people readily available? The answer is almost always no. While the number of programmers has grown considerably the demand for these people has grown even faster, as illustrated in figure 2.

Date	Programmer Jobs Available	Programmers Available	Shortfall
1975	320,000	308,000	12,000 (4%)
1985	640,000	476,000	164,000 (26%)

Source: U.S. Department of Labor Statistics

Programmer Availability

FIGURE 2

Turning from the resources available to the ASD Manager, let's examine the demand side of the question. Assuming that the requested application systems have been subjected to an appropriate value (costbenefit) analysis, the demand is real. The best that can be gained from prioritizing the backlog is a higher return on investment. Given the difficulty in quantifying payoffs (especially for decision support systems (DSS) which are rapidly becoming the most common type of new application system) there is little to be gained in structuring the backlog.

Can the ASD Manager learn to live with the backlog? Probably not, because of the next question to be answered by the ASD Manager.

How do I respond to upper managements' increasing demand for solutions?

One solution that is attracting considerable attention today is user-developed systems. The title of James Martin's latest book, Application Development Without Programmers, reflects this orientation. Further, the tremendous number of personal computers in use and their continued decreasing cost indicates that the typical organization user can afford this solution. Many advocates of user-developed systems argue that non-procedural languages and user-friendly software packages allow the user to implement this solution. Given that this solution is economically and technically feasible raises the next question.

How do I handle users seeking their own solutions?

One answer, suggested by advocates of user-developed systems is: You don't. Leave the users alone and let them seek their own solutions. The problem most ASD Managers have with this solution is: How do I manage application systems development if any user can build his/her system? The answer, of course, is he/she can't. The bigger question is, does the process need to be managed? The weight of evidence at this time seems to clearly indicate an affirmative answer. The major reason for managing the process is that the risks of not managing are too high. Gordon Davis recently articulated several of these risks: the high probability of programming errors leading to a loss of information integrity, a resurgence of all the problems associated with duplication of data files and databases, and large inefficiencies associated with duplication of effort. The next question is closely linked to this issue.

How do I effectively manage the organizations' information resources?

One necessary response is, by not advocating managerial responsibility or totally delegating thar responsibility to the users. The most important issue to understand here is that the ASD Manager manages in cooperation with the other MIS managers. These managers must work in concert to manage the organizations' information resources. When the MIS function is managed, new technologies are regularly introduced in the organization. This raises another question for the ASD Manager.

How do I utilize technological advances wisely?

Given the scarcity and high cost of people, the increased availability of inexpensive computational capacity, and the availability of fourth generation software, what tradeoffs are available to increase the productivity of the systems/information analyst? In other words, how do I best use technology to get systems developed?

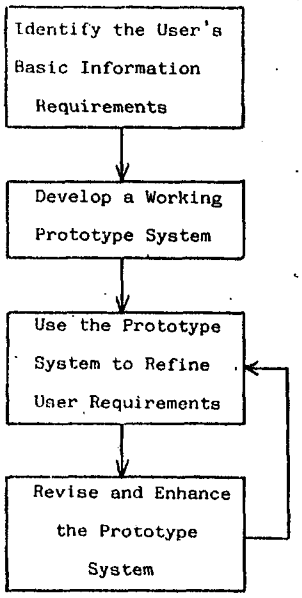
The answer to this, and the other questions asked by the ASD Manager, lies in the examination of the process used to build systems. Ninety percent of all systems developed in 1980 were built using the same methodology that was used in 1960. That methodology is the systems' development life cycle. Over the last twenty years the major costs (in time and dollars) have shifted from the physical design phase of the life cycle to the logical design phase, and particularly to the information requirements definition step in that phase. But the methodology has remained essentially unchanged. A new system development methodology, prototyping, goes a long way in helping the ASD Manager meet the challenge for information systems in the eighties.

Meeting The Challenges

Prototyping is presented here as the most appropriate way for the ASD Manager to meet the challenge for information systems in the '80's. Prototyping is not a panacea, it is simply a sound methodology that provides answers to many ot the present needs in the systems development area. The following sections define and describe the methodology, the tools required for its successful application and then describes how it meets the challenges facing the ASD Manager.

What is Prototyping?

Prototyping is a development methodology; A methodology which has been used in the design fields (engineering, architecture, etc.) for centuries; A methodology both priholophically and operationally different from the traditional life cycle methodolog. Prototyping has not been used in the application systems development area in the past for one simple reason - the tools required to enable prototyping of application systems were not available until the late '70's. Prototyping is not piloting, although one can pilot a system that was developed by prototyping. The prototype process is typically carried out by two individuals -- a competent user and a technically skilled systems analyst. The prototype process is described in figure 3.



Identify the User's Basic Information Requirements
Develop a Working Prototype System
Use the Prototype System to Refine User Requirements
Revise and Enhance the Prototype System

The Prototype Process

FIGURE 3

Prototyping is a four step process. The first step is to identify and capture a nucleus or skeleton that embodies the essential features of the user's requirements. The systems analyst works with the user during this process which usually takes less than four hours. Both the data abstracting and the process simulating approaches are successful in this step. The second step is completed by the systems analyst using a unique set of resources that have become widely available in the last five years. The most important characteristic of this initial prototype is that it must be implemented in a very short time -- over night, if possible. The initial prototype is purposefully incomplete. It is a simulation in the sense that it represents the essential elements desired by the user in a simplified form. Design and implementation of the system is accomplished not by completely determining the user's information requirements but by developing a system which delivers as output the key or critical information requirements.

Steps one and two of the prototype process return nothing to the user. Step three, however, is delivery, installation, training, operation and use of the prototype system. This "hands-on" experience with a real system provides a basis for mutual understanding of the system by both user and analyst. The user exercises the system during this step and records the problems and incongruities as well as his/her discoveries as to what they wish the system could do. The user controls the time spent in this step and then contacts the analyst to begin step four. This final step involves only the analyst. The analyst's job is to enhance and modify the prototype system to meet the desires and needs of the user. These modifications must be made rapidly -- within a few days. Steps 3 and 4 are repeated as many times as the user feels necessary and at intervals determined by the user.

This prototype process is based on a simple principle -- that it is easier for the users to describe what they don't like about an existing system, than it is for them to describe what they would like in an imaginary system.

The Tools Required for Prototyping

Prototyping takes advantage of the technological advances made in the last decade. The basic tools that are essential for prototyping are: 1) Interactive systems (either through time sharing or mini/micro computers). Interactive facilities extend the apparent power of information processing resources by reducing delays and by extending control over the resources to the user. 2) Database management systems (with as full a range of enhancements as possible). A natural language based query language is essential. 3) Generalized input and output software is required if not available with the database management system. This software includes report generators, report writers and nonprocedural, natural languages. 4) A model base (which contains all of the essential features of a database). Many other tools are desirable when prototyping but the purpose of this paper is not to discuss the tools in detail, rather it is to describe how to meet and beat the challenges for information systems in the 80's.

How Prototyping Meets the Challenges for Information Systems

The prototyping solution addresses the basic reason for most of the challenges ASD Managers face today. It strikes at the heart of the problems -- an inefficient, time consuming and labor intensive development methodology. For the last twenty years the systems development life cycle has remained essentially unchanged and has been the only methodology employed to develop application systems. Prototyping provides a viable and attractive alternative methodology. All future application systems will not be developed using prototyping. But based on the reported evidence and my personal experiences with prototyping over the last four years, I am convinced that prototyping will be the dominant development methodology by the mid-eighties. Let's examine some of the reasons why.

Prototyping provides a real response to upper management's demands for solutions. Prototyping provides a real solution because: 1) It reduces the development time required to deliver a functional and useful system to the user. While the reductions in development time vary considerably, times between 10% and 20% of that required by the traditional methodology are very common. 2) The proportion of systems analyst time to user time in the development process shifts significantly placing a greater load on the users to do what they can do best -- define their own information and systems requirements. By reducing the time required to develop application systems and by lessening the systems analyst's involvement, prototyping shows great promise for reducing the applications development backlog.

Prototyping helps handle users seeking their own solutions. Prototyping provides an alternative to the users. The users can get the system they want without having to become a computer non-programmer. Prototyping focuses the user's attention and energy on that aspect of systems development for which he/she is best qualified. The results, universally reported, are much higher user satisfaction with the application system.

Prototyping helps the ASD Manager to manage the organization's information resources. Prototyping trades off machine inefficiencies for people efficiencies. With cheap and abundant computational resources and expensive and scarce human resources, this is the appropriate management tradeoff.

Conclusion

There are large number of compelling reasons to use prototyping as a development methodology. It far surpasses any other available alternative in meeting the challenges facing ASD Managers today. But a word of caution is appropriate in closing. There appear to be two potential problems with prototyping: the reduction of controls on both the process and product and the difficulty in easily integrating prototyped systems with other, traditionally developed systems. The ASD Manager should carefully assess these potential problems lest he/she simply trade one set of challenges for another.

INFORMATICA 2/1982

STANDARDI I POLITIKA STANDARDIZACIJE U OBLASTI INFORMATIKE

II. deo

S. BRAJOVIĆ–BRATANOVIĆ
B. DŽONOVA–JERMAN–BLAŽIČ

SAVEZNI ZAVOD ZA STANDARDIZACIJU, BEOGRAD
INSTITUT JOŽEF STEFAN', JAMOVA 39, LJUBLJANA

UDK: 389.6:681.3

Razmatrana je problematika i ciljevi standardizacije za područje informatike i računarske tehnike. Razradjena je metodologija rada na standardima u oblasti informatike kao i srednjeročni plan rada Grupe za standardizaciju Saveznog zavoda za standardizaciju u oblasti informatike.

STANDARDS AND STANDARDISATION POLICY IN THE FIELD OF INFORMATICS AND COMPUTER SCIENCE. The paper discuss the problems encountered in the development of standards in the field of informatics and computer science as well as the standardisation policies. The areas and procedures of standardisation are treated too. The working plans of the Yugoslav Committee for standardisation in the field of informatics for the next five years are presented.

4. METODOLOGIJA RADA NA STANDARDIMA

U metodologiji rada na standardima, koja predstavlja sadašnju praksu rada u SZS.

4.1. Faza planiranja

Planiranje je prva aktivnost u radu na standardima. Zadatak planiranja je da obezbedi da standardi koji će biti predmet rada u narednom periodu odražavaju stvarne potrebe društva u odredjenoj oblasti. U oblasti informatike plan se donosi usaglašavanjem mišljenja stručne javnosti, radnih organizacija i organa uprave.

Pod terminom stručna javnost podrazumevaju se:
- naučni instituti i fakulteti u čiju delatnost spada oblast informatike,
- udruženja korisnika računara u republikama i pokrajinama,
- saveti za informatiku republika i pokrajina, odnosno upravni organi za područje informatike i Društveni sistemi informisanja.

Pod terminom radne organizacije podrazumevaju se
- predstavnici velikih centara za informatiku, elektronskih računskih centara i sl.,
- predstavnici proizvodjača računara odnosno zastupnici u SFRJ svojim specijalizovanim tehničkim službama.

Pod terminom organi uprave podrazumevaju se predstavnici organa savezne uprave koji imaju ili će imati Centre za automatsku obradu podataka ili im je delatnost vezana za oblast informatike.

Konkretan plan rada na standardima, kako petogodišnji tako i godišnji predlože odgovarajuća služba, odnosno stručnjak iz SZS na osnovu usvojene koncepcije i sagledanih potreba u odredjenoj oblasti standardizacije. Za oblast informatike, iz napred izloženih razloga, najpogodnije da se rad na standardima organizuje u okviru kompleksnih programa za odredjene oblasti. U pojedinim oblastima radili bi specijalizovani radni timovi sastavljeni od stručnjaka potrebnih profila, odnosno odredjenom radnom timu bi poverili rad na celom kompleksnom programu u dotičnoj oblasti.

U narednom petogodišnjem planu u informatici su načelno sagledane sledeće oblasti standardizacije :

Grupa I. Informatika i obrada podataka

1. Opšta grupa
2. Skupovi znakova i kodovi i šifarski sistemi
   2.1 Skupovi znakova
   2.2 Kodiranje
   2.3 Šifarski sistemi
3. Magnetni nosioci podataka
   3.1 Magnetne trake
   3.2 Kasete
   3.3 Diskete
   3.4 Diskovi
4. Prenos podataka
   4.1 Opšti deo
   4.2 Osnovne kontrolne procedure sistema za prenos podataka
   4.3 Kontrolne procedure visokog nivoa sistema za prenos podataka
   4.4 Veze terminalne opreme sa komunikacionim sistemom
5. Projektovanje sistema, programiranje i dokumentacija projekta
   5.1 Projektovanje
   5.2 Metodologija programiranja
   5.3 Izbor sistema za AOP
   5.4 Unos podataka
6. Ostali periferici
   6.1 Papirne kartice
   6.2 Papirna traka
   6.3 Uredjaji za optičko čitanje znakova
   6.4 Terminali
7. Programski jezici
8. Računarski sistemi i oprema
   8.1 Opšti standardi
   8.2 Testiranje
   8.3 Sigurnost
9. Numeričko upravljanje mašinama·

Mehanizam prihvatanja plana u principu se odvija prema procedurama uobičajenim za druge oblasti standardizacije, odnosno verifikuje se u toku javne diskusije, a potom prihvata u odgovarajućoj komisiji kao definitivan plan. Na osnovu plana pravi se detaljan program rada.

U fazi planiranja rada na standardima SZS učestvuje kao:
- inicijator aktivnosti
- formalni davalac predloga plana
- organizator aktivnosti kod verifikacije i konačnog usvajanja plana
- osnivač radnog tela za rad na standardu (komisije, podkomisije, radne grupe).

Na osnovu prihvaćenih godišnjih planova SZS organizuje rad na predlogu standarda. Predlog standarda radi se u samom SZS ili se za rad na predlogu zadužuje radna grupa ili stručni tim izvan SZS.

4.2. Faza pripreme

Faza pripreme počinje u trenutku kad SZS od svog saradnika, radne grupe ili stručnog tima dobije pripremljen radni materijal - predlog za donošenje standarda. Predlog se najpre metodološki kontroliše, a potom se vrši i provera njegove tehničke sadržine.
Pošto se sve stručne primedbe otklone vrši se unifikacija predloga standarda i njegova pravna i terminološka kontrola. Nakon ispravke dobijenih primedbi završava se faza pripreme i predlog standarda postoje prednacrt standarda.

4.3. Faza prednacrta

U fazi prednacrta suradjuje Komisija za standarde, koja nastavlja da radi i u fazi nacrta standarda. Najveći problem u fazi prednacrta je obezbedjenje komisije koja poseduje potrebnu stručnost i koja istovremeno reprezentuje sve zainteresovane nivoe i strukture korisnika. Formiranje strukture komisije u smislu odredjivanja stručnog profila članova komisije, koji bi bili u stanju da sa stručnog aspekta aktivno saradjuju u radu, kao i sagledavanja interesa korisnika trebalo bi *vršiti za svaku oblast standardizacije posebno.* Tako, na primer, kod standardizacije protokola za prenos podataka u sastav komisije bi trebala da bude uključena JPTT, koja sa druge strane nema nikakvog interesa za rad u druge oblasti informatike. Kod standardizacije programskih jezika, koji nisu razvijeni u našoj zemlji, trebalo bi obavezno uključiti i zastupnike proizvodjača računara i sl. Kod formiranja komisije za standarde u SZS je i do sada bila praksa da se nastoji da budu uključeni svi subjekti za koje bi se moglo pretpostaviti da su zainteresovani. Specifičnost situacije u oblasti informatike je ta da za sada još ne postoji *razradjeni mehanizmi organizacije,* a sa druge strane broj i disperzija interesovanja subjekata je izuzetno velika, odnosno korisnici potiču iz privrede, bankarstva, javne uprave, specijalizovanih institucija, naučnih institucija itd. Kod sastava komisije je potrebno posvetiti ozbiljnu pažnju i pristupiti tom poslu izuzetno aktivno i profesionalno.

Na osnovu primedbi usaglašenih komisije vrši se ispravka prednacrta standarda i tako se dobija nacrt standarda, koji je spreman za javnu diskusiju.

4.4. Faza nacrta

Faza nacrta počinje anotacijom, odnosno formalnim stavljanjem standarda na javnu diskusiju koja može trajati do 3 meseca.

Nakon završetka javne diskusije komisija pregleda prispele primedbe i usaglašava ih a potom vrši ispravku nacrta standarda. Ukoliko su ispravke suštinske priroda, ova faza se može ponoviti. Inače, posle unošenja ispravki završava komisija sa radom, a nacrt postoje konačni predlog standarda.

4.5. Faza uskladjivanja

Konačni predlog standarda prolazi u SZS poslednju stručnu kontrolu zatim pravnu i terminološku kontrolu a potom unifikaciju.

Pošto se sve primedbe otklone dobija se konačni tekst standarda.

4.6. Faza objavljivanja

Konačni tekst standarda prolazi zakonski propisanu proceduru objavljivanja, jugoslovenskog standarda.

4.7. Faza tehničke obrade

Jugoslovenski standard se prevodi na jezike naroda i narodnosti, vrši se njegova tehnička obrada i standard izdaje u njegovoj konačnoj formi.

5. SREDNJEROČNI PLAN RADA GRUPE ZA STANDARDIZACIJU U OBLASTI INFORMATIKE U SZS

Detaljnim uvidom u medjunarodnu i inostranu nacionalnu standardizaciju došlo se do osnovnog spiska standarda koji u narednih 5 godina treba da budu predmet jugoslovenske standardizacije. *Spisak treba shvatiti kao minimalni skup standarda,* potreban da se dostigne prihvatljiv nivo standardizacije u oblasti informatike u našoj zemlji.

Grupa 1: Opšta grupa

1.1. Rečnici pojmova

1. Osnovni pojmovi (postoji standard)
2. Aritmetičke i logičke operacije
3. Tehnologija opreme
4. Organizacija i iskazivanje podataka
5. Priprema i rukovanje podacima
6. Programiranje
7. Periferna oprema
8. Medijumi za skladištenje i čuvanje podataka
9. Pouzdanost, raspoloživost i održavanje
10. Prenos podataka (postoji, potrebna revizija)

Grupa 2: Skupovi znakova, kodovi i šifarski sistemi

2.1. Skupovi znakova
2.2. Kodovi

1. 7-bitni jugoslovenski kod za razmenu podataka za latinično srpskohrvatsko i slovenačko pismo
2. 7-bitni jugoslovenski kod za razmenu podataka za ćirilično srpskohrvatsko pismo
3. 7-bitni jugoslovenski kod za razmenu podataka za ćirilično makedonsko pismo
4. 7-bitni jugoslovenski kod za razmenu podataka za albansko pismo
5. 7-bitni jugoslovenski kod za razmenu podataka. Osnovna verzija
6. Pravila za proširenje 7-bitnog na 8-bitni kod
7. Primena 7-bitnog i 8-bitnog koda za razmenu podataka pri razmeni podataka na 9-kanalnim magnetnim trakama širine 12,7 mm
8. Primena 7-bitnog i 8-bitnog koda za razmenu podataka pri razmeni podataka sa kasetom sa magnetnom trakom širine 3,81 mm

2.3. Šifarski sistem

9. Šifarnik gradova i opština Jugoslavije
10. Šifarnik zemalja
11. Šifarnik valuta
12. Šifarnik OOUR
13. Šifarnik artikala
14. Šifarnik republika i pokrajina
15. Obeležavanje datuma i vremena

Grupa 3: Magnetni nosioci podataka

3.1. Magnetne trake

1. Neispisane magnetne trake za razmenu podataka širine 12,7 mm (0.5 in) sa 8 i 32 reda po mm (200 i 800 bpi) NRZI i 63 reda po mm (1600 bpi) fazno kodirane - fizičke i magnetne osobine trake
2. 9-kanalna magnetna traka za razmenu podataka širine 12,7 mm (0.5 in) sa gustinom upisa 32 reda po mm (1600 bpi)
3. 9-kanalna magnetna traka za razmenu podataka širine 12,7 mm (0.5 in) sa gustinom upisa 63 reda po mm (1600 bpi) fazno kodirana
4. Labelisanje i struktura datoteka kod magnetnih traka za razmenu podataka

3.2. Kasete sa magnetnom trakom

5. Kasete sa magnetnom trakom za razmenu podataka širine 3,81 mm (0.15 in) sa gustinom upisa 63 reda po mm (1600 bpi) fazno kodirane
6. Labelisanje i struktura datoteka kaseta sa magnetnom trakom za razmenu podataka

3.3. Diskete

7. Izmenjive diskete za razmenu podataka, fizičke i magnetne osobine
8. Izmenjive diskete za razmenu podataka, format staze
9. Labelisanje i struktura datoteka disketa za razmenu podataka

3.4. Diskovi

10. Izmenjivi magnetni diskovi sa 6 ploča, fizičke i magnetne osobine
11. Izmenjivi magnetni diskovi sa 6 ploča, format staze
12. Izmenjivi magnetni diskovi sa 11 ploča, fizičke i magnetne osobine
13. Izmenjivi magnetni diskovi sa 12 ploča (1oo Mbytes)
14. Izmenjivi magnetni diskovi sa 12 ploča (200 Mbytes)

Grupa 4: Prenos podataka

4.1. Opšti deo

1. Struktura znakova za start-stop i sinhroni prenos
2. Upotreba provere longitudinalnog pariteta za otklanjanje grešaka kod prenosa podataka
3. Sinhrona signalizacija kod prenosa podataka
4. Sinhrona signalizacija velike brzine izmedju terminalne (DTE) i komunikacione (DCE) opreme kod prenosa podataka
5. Odredjivanje performansi sistema za prenos podataka
6. Kvalitet prenosa i granične vrednosti kvaliteta sistema za prenos podataka
7. Funkcionalni zahtevi za medjuveze nižeg nivoa kod dupleks veze od tačke do tačke

4.2. Osnovne kontrolne procedure sistema za prenos podataka

8. Osnovne kontrolne procedure sistema za prenos podataka
9. Osnovne kontrolne procedure, prenos podataka nezavisan od koda
10. Osnovne kontrolne procedure, procedure za prekid i raskidanje veze
11. Osnovne kontrolne procedure, procedure za oporavak
12. Osnovne kontrolne procedure, procedure za odabiranje stanica
13. Osnovne kontrolne procedure, konverzaciona razmena poruka
14. Osnovne kontrolne procedure, ostale procedure

4.3. Kontrolne procedure visokog nivoa sistema za prenos podataka

15. Kontrolne procedure visokog nivoa, osnovne procedure
16. Kontrolne procedure visokog nivoa, dodatne procedure
17. Kontrolne procedure visokog nivoa, struktura okvira
18. Kontrolne procedure visokog nivoa, uravnotežena veza stanica istih nivoa
19. Kontrolne procedure visokog nivoa, neuravnotežena veza sa primarnom i sekundarnom stanicom
20. Kontrolne procedure visokog nivoa, komunikacioni transportni protokol

4.4. Veze terminalne opreme sa komunikacionim sistemom

21. Specifikacija konektora za DTE-DCE vezu, sa 15 nožica
22. Specifikacija konektora za DTE-DCE vezu, sa 25 nožica
23. Specifikacija konektora za DTE-DCE vezu, sa 37 nožica
24. Medjuveze DTE-DCE kod start-stop prenosa u javnim mrežama za prenos podataka
25. Medjuveze DTE-DCE kod sinhronog prenosa u javnim mrežama za prenos podataka
26. Medjuveze DTE-DCE kod paketnog prenosa u javnim mrežama za prenos podataka

Grupa 5: Projektovanje sistema, programiranje (dokumentacija projekta)

5.1. Projektovanje sistema

1. Metodologija projektovanja informacionih sistema
2. Projektna dokumentacija

5.2. Metodologija programiranja

3. Simboli za dijagrame sistema obrade informacija (postoji, potr. reviz.)
4. Metodologija struktuiranog programiranja
5. Programska dokumentacija

5.3. Izbor sistema za AOP

6. Metodologija izbora sistema za automatsku obradu podataka

5.4. Unos podataka i obrada

7. Dokumentacija za sistem unosa podataka
8. Dokumentacija obrade

Grupa 6: Ostali periferni uredjaji

6.1. Papirne kartice

1. Nebušene papirne kartice, mere, uslovi kvaliteta (postoji, potr. rev.)
2. 80 kolonske bušene papirne kartice - specifikacija ubušenja
3. Primena jugoslovenskih latiničnih kodova za razmenu podataka na bušenoj papirnoj kartici
4. Primena jugoslovenskih ćiriličnih kodova za razmenu podataka na bušenoj papirnoj kartici

6.2. Papirna traka

5. Nebušena papirna traka, osobine
6. Bušena papirna traka, dimenzije i lokacije ubušenja
7. Razmena podataka na bušenoj papirnoj traci
8. Primena jugoslovenskih latiničnih kodova za razmenu podataka na bušenoj papirnoj traci
9. Primena jugoslovenskih ćiriličnih kodova u razmeni podataka na bušenoj papirnoj traci

6.3. Uredjaji za optičko prepoznavanje znakova

10. Skup znakova za ručno pisanje za uredjenje za optičko prepoznavanje znakova
11. Hartija za uredjaje za optičko prepoznavanje znakova, specifikacije, optičke osobine, testiranje

12. Skup štampanih znakova za mašinsko pisanje za uredjaje za optičko prepoznavanje znakova
13. Dimenzije znakova za uredjaje za optičko prepoznavanje znakova
14. Pozicija linija na dokumentima za uredjaje za optičko prepoznavanje znakova

6.4. Terminali

15. Alfanumerička tastatura za latinično srpskohrvatsko i slovenačko pismo za 44,46 i 48 tipki
16. Alfanumerička tastatura za ćirilično srpskohrvatsko i makedonsko pismo za 44,47 i 48 tipki
17. Alfanumerička tastatura za albansko pismo za 44, 47 i 48 tipki
18. Alfanumerička tastatura za madjarsko pismo za 44, 47 i 48 tipki
19. Tastature za numerički unos podataka
20. Skup znakova za ručno pisanje za uredjaje za prepoznavanje znakova pisanih magnetnim mastilom

Grupa 7: Programski jezici

1. Programski jezik FORTRAN
2. Programski jezik COBOL
3. Programski jezik PL/1
4. Programski jezik BASIC
5. Reprezentacija izvornog programa kod razmene programa na jezicima FORTRAN, COBOL, PL/1, BASIC

Grupa 8: Računarski sistemi i oprema

8.1. Opšti standardi

1. Ocena performansi sistema za AOP

8.2. Sigurnost opreme

2. Opšti zahtevi za smeštaj opreme za automatsku obradu podataka
3. Zahtevi za sigurnost opreme za AOP, mehanički in fizički zahtevi
4. Zahtevi za sigurnost opreme za AOP, električne instalacije
5. Zahtevi za sigurnost opreme za AOP, protivpožarne i drugo inst.
6. Merenje radio-interferencije opreme za AOP i dozvoljene vrednosti

8.3. Testiranje sistema

7. Hardversko testiranje računara, testiranje centralnog procesora, centralne memorije i perifernih procesora
8. Hardversko testiranje računara, testiranje periferika
9. Hardvarsko testiranje računara, testiranje analognih ulaza i izlaza
10. Testiranje operativnog sistema
11. Testiranje programa prevodilaca

9. Numeričko upravljanje mašinama

1. Numeričko upravljanje, simboli
2. Procesor za numeričko upravljanje, logička struktura izlaza.

6. ZAKLJUČAK

U materijalu je učinjen napor da se sagleda problematika standardizacije u oblasti informatike i računarske tehnike. Uočeni su takodje, problemi koji bi mogli iskrsnuti u toku realizacije srednjoročnog plana grupe za standardizaciju u oblasti informatike.

U utečenoj metodologiji rada SZS uočene su faze gde bi SZS trebao da zadrži iniciativu, koordinativnu i kontrolnu ulogu, dok bi ostalu aktivnost trebalo da se prepusti odgovarajućim subjektima izvan SZS.

7. REFERENCE

U pripremi materijala korišćeni su sledeći dokumenti:

1. Politika standardizacije u Jugoslaviji, Beograd 1979, izdavač SZS, odgovorni urednik Milan Krajnović.
2. Srednjoročni plan rada na standardima u oblasti informatike, jun 1981, SZS, autor S. Brajević-Bratanović.
3. P. Wolley, Standards in Computing, S. Tech. College, 1978, London.

INFORMATICA 2/1982

THE DESIGN AND DEVELOPMENT OF A MICROCOMPUTER BASED CAD/CAM SYSTEM FOR 2 1/2 MILLING OPERATIONS

S.K. KHURMI, C.B. Besant and H.A. Pak

MECHANICAL ENGINEERING DEPARTMENT,
IMPERIAL COLLEGE OF SCIENCE & TECHNOLOGY,
LONDON.

UDK: 681.3:621.914

The ever diminishing size and cost of microelectronic devices has brought about two technological achievements. Firstly, in the integration of two technologies, namely, Computer Aided Design and Computer Aided Manufacture into unified CAD/CAM systems, whereby a design is developed and the manufacturing process controlled from start to finish with a single system. Secondly, in the feasibility of low cost CAD/CAM systems on an economic scale which can be related to small sized manufacturing industries.

The system concept described is in an effort to obtain the optimum benefit from recent developments in the use of microprocessors in the field of CAD/CAM.

The current system is capable of 2D and 2 1/2 D design and machining of milling components comprising of straight lines and arcs, which conforms to the need of a large section of the manufacturing industry.

HARDWARE CONFIGURATION

The CAD/CAM Workstation

The CAD/CAM workstation's hardware is based around the Motorola 6809 8-bit microprocessor and at the time of purchase (1980) it represented one of the most versatile microcomputer systems providing a wide range of hardware and software support.

The overall hardware configuration of the integrated CAD Workstation, Fig. 1, consists of:

1. A Southwest Technical Products (SWTP) 6809 8.bit 2 MHz microprocessor based single user system with 64 K bytes of user RAM, of which 8 K bytes is utilised by FLEX9 Operation System. For programs requiring large data storage the RAM can be expanded up to 768K on a single user system. The 6809 microprocessor, although an 8-bit device, has a 16-bit architecture which makes it one of the most versatile and fastest 8-bit microprocessors in the market (1).

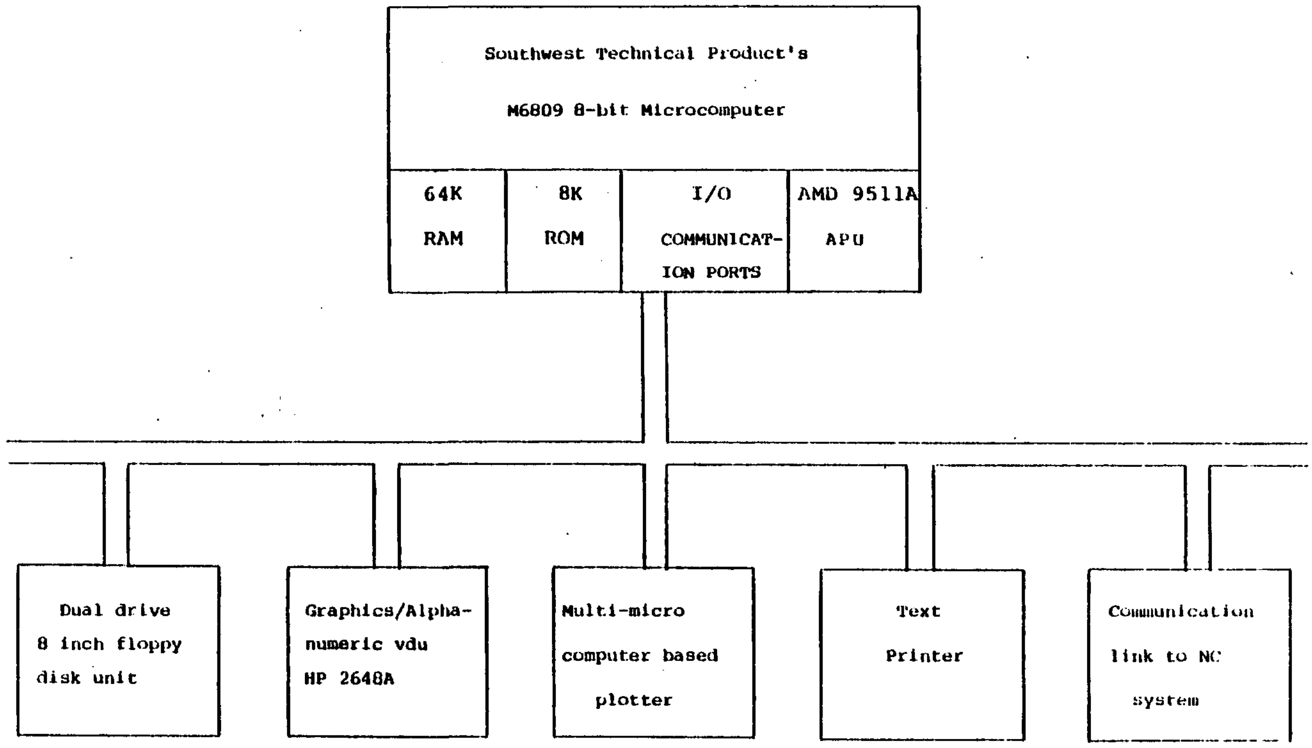

Southwest Technical Product's
M6809 8-bit Microcomputer
64K RAM
8K ROM
I/O COMMUNICATION PORTS
AMD 9511A APU
Dual drive 8 inch floppy disk unit
Graphics/Alphanumeric vdu HP 2648A
Multi-micro computer based plotter
Text Printer
Communication link to NC system

Fig.1 The basic hardware of the CAD/CAM workstation.

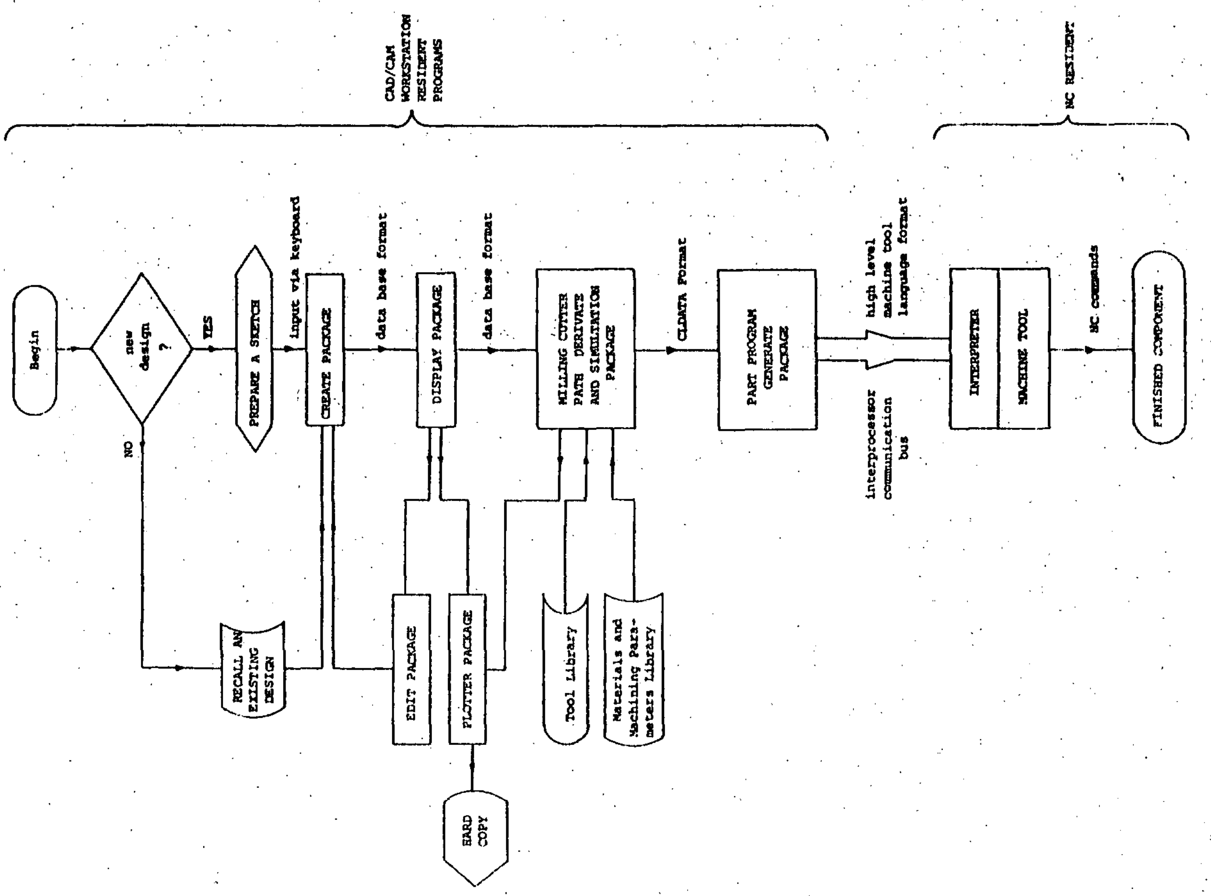

CAD/CAM WORKSTATION RESIDENT PROGRAMS
NC RESIDENT
Begin
new design ?
YES
NO
PREPARE A SKETCH
input via keyboard
CREATE PACKAGE
data base format
DISPLAY PACKAGE
data base format
MILLING CUTTER PATH DERIVATE AND SIMULATION PACKAGE
CLDATA format
PART PROGRAM GENERATE PACKAGE
high level machine tool language format
interprocessor communication bus
INTERPRETER
MACHINE TOOL
NC commands
FINISHED COMPONENT
RECALL AN EXISTING DESIGN
EDIT PACKAGE
PLOTTER PACKAGE
Tool Library
Materials and Machining Parameters Library
HARD COPY

Fig.2 Macro Interpackage Communication Diagram

2. A dual 8 inch double density, double sided flexible disk unit, providing up to 2 M bytes of usable (formatted) online storage.
3. An "intelligent" combined alphanumeric/graphics raster scan vdu - Hewlett Packard 2648A refresh terminal. It contains its own microcomputer which commands the execution of display control functions on a screen resolution of 720 x 360 pixels, and operates at 9600 baud.
4. A flat-bed AO multi-microcomputer based plotter.
5. And a text printer.

Besides the standard I/O ports the SWTP system also provides the user with additional peripheral communication ports (i.e. serial or parallel) so that several peripherals as well as computers can be linked to the workstation. The ability to communicate with other computers is a very important feature in that it provides the capability of developing larger and more complex programs by distributing the tasks amongst several heterogeneous computers arranged in a hierarchical configuration. Another advantage is that the CAD/CAM workstation is capable of simultaneously communicating with several NC machine tools, similar to a DNC system configuration.

The Multi-Microcomputer CNC System

The CAM hardware can be sub-divided into two discrete elements, namely, the control system and the mechanical components associated with the machine tool.

The control system is one which has been developed at Imperial College for the control of multi axes machines such as machine tools, plotters and robots (2). The basic architecture of the control system is a hierarchical one of a master/slaves configuration. By employing such an architecture system modularity, flexibility and expandability are maintained. The concept of the general control system is that it can be customised to the specifications and peculiarities of existing machine tools.

WORKSTATION's SOFTWARE ARCHITECTURE

Software Hierarchy

Successful design and development of CAD/CAM software packages on a microcomputer requires careful evaluation of the architecture, capabilities and limitations of the system (3). The three most obvious limitations that are inherent to most microcomputer systems, especially 8-bit microcomputers, are firstly, they possess relatively slow processing speeds compared with minicomputer based CAD/CAM systems, secondly, they lack software backup, and thirdly, they have severe faststore restrictions. Despite these implications there are several ways one can minimise these restrictions for the development of a CAD/CAM system. Some of the most common ways are:

. development of a modular software
. development of a modular and intelligent CNC system
. utilization of a "powerful" operating system
. use of intelligent peripherals
. allow communication with other computers.

As a result of these recommendations the software packages described were formulated not only to be modular but be executed in such a manner as to optimise the flow of information from the design stage to manufacture. Since the process of product design and manufacture is a systematic one following a standard design sequence, whereby first the design is formulated, analysed and then the cutter path movements generated followed by the machining process, the sequence of the CAD/CAM software must adhere to it. However, it does not imply that all the packages should be resident in memory at the same time during the various stages of the product design. Thus, in order to minimise on board storage the software packages can be sub-divided into three main classifications:

1. Creation and visual display programs.
2. Cutter path simulation programs.
3. Part program generation programs.

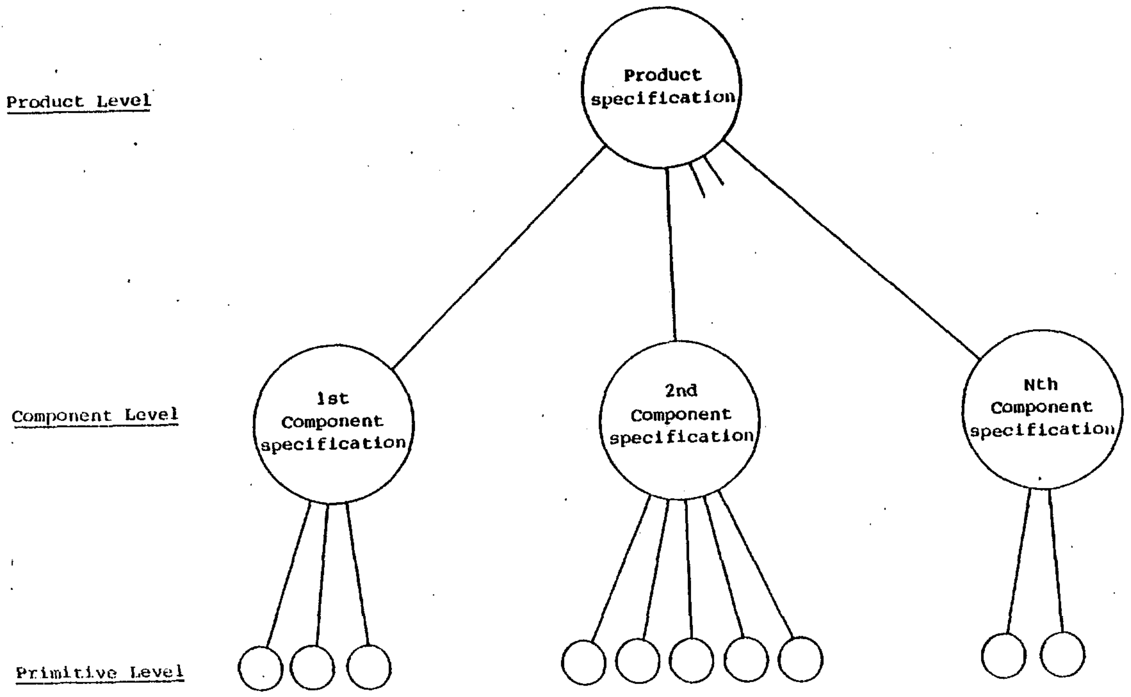

Product Level
Product specification
Component Level
1st Component specification
2nd Component specification
Nth Component specification
Primitive Level

Fig.3 A three level hierarchical data base structure

By sub-dividing the software packages into the above-mentioned classifications one can develop quite large scale CAD/CAM programs on microsystems by executing them sequentially with intermediate formats transferred from one package to another, Fig. 2.

The CREATE Package

The CREATE package is responsible for the storage of data and the general management of the data base (DB). CREATE is written in 6809 assembly language mainly for speed and ease of memory management and data decoding. The CREATE package accepts command from the user and calls display and edit routines to interactively visualise and modify the elements of a component. The component can be related to any machining operation e.g. turning, milling, punching, drilling, etc.

The structure of the data base is illustrated in Fig.3 and represents a simple way of storing data (4). Hierarchical configurations represent an ideal way of representing information which itself is not centralised.

The hierarchical structure falls/three main categories (5), namely Product Specification, Component Specification and Primitives. The Product Specification contains information pertinent to the managerial aspects associated with the product such as the materials, quantities required,contract number, customer's name and delivery dates. The product is then sub-divided into several components, which comprise of several geometrical Primitives. These Primitives are a collection of geometrical attributes which can be called and appended to from the geometry of the component. For milling operations some of the commonly used primitives are cylinder, thread, line, fillet, box, pyramid, etc.

Thus by either using the standard primitives (or by defining new primitives) complex milling geometries can be quickly and easily entered into the computer.

The EDIT Package

This package, also written in assembly language, allows components to be recalled and modified interactively. Similar to a text editor, primitives can be analysed graphically as well as numerically. By sequentially stepping through the primitives which define a component, geometrical errors can be quickly and easily detected. Once the error has been detected, two very powerful commands, namely Delete and Insert, allow entire primitives to be deleted and replaced by new ones.

The DISPLAY Package

DISPLAY is an interactive, modular graphics package, written in PASCAL, and is responsible for the comprehensive display of the technological primitives and components as they are created or modified by the above-mentioned packages.

The output is either in the form of a single view of the part or a four view first angle orthographic projection whereby the vdu screen is sub-divided into four viewports as illustrated in Fig. 4.

General display features such as orthographic views, isometric views, clipping, windowing, rotation, scaling and dimensioning form the basis of the DISPLAY package.

The PLOTTER Package

The PLOTTER Package communicates directly with the multi-microcomputer based flatbed plotter designed and built at Imperial College via a standard RS 232 serial communication bus. The PLOTTER package contains an ASCII character generation set in software and is responsible for the production of graphical hard copies of the component in first angle orthographic projection and/or rotated 3D views. Due to the limited resolution of the vdu screen high resolution plotter hard copies can be used for the verification of the component's geometry as well as cutter paths.

The Milling Cutter Path Derivate and Simulate MCPDRS Package

Once the geometry of the component to be pocket-milled has been entered, displayed, analysed and stored, using the above-mentioned packages, the cutter path's loci can be generated. The automatic generation of the loci and the machining commands, such as feed rates, spindle speeds and tool changes, are the responsibility of the MCPDRS package. A macro flowchart of the MCPDRS package is illustrated in Fig. 5.

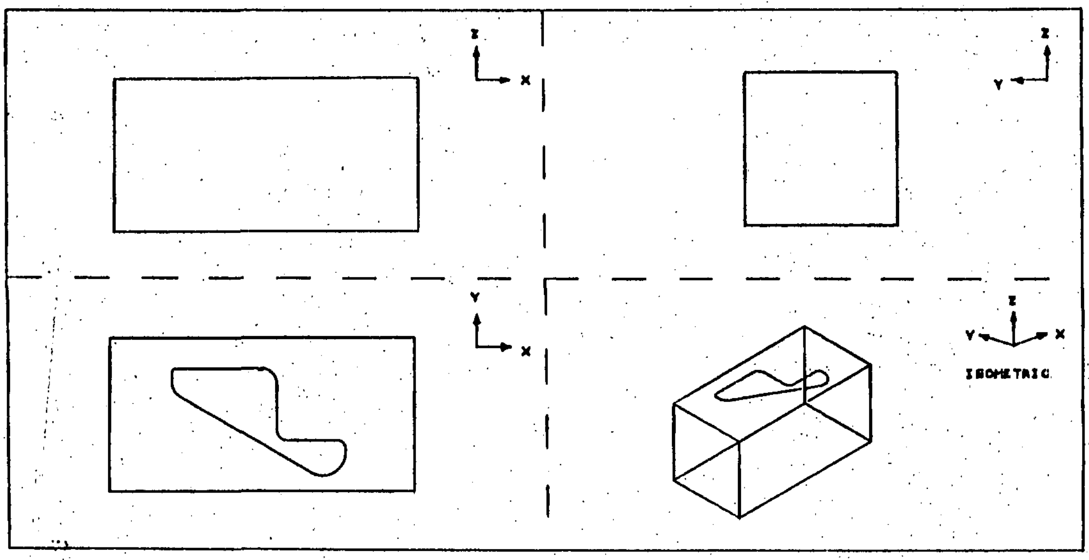


Fig.4 A typical view of a component displayed in four views.

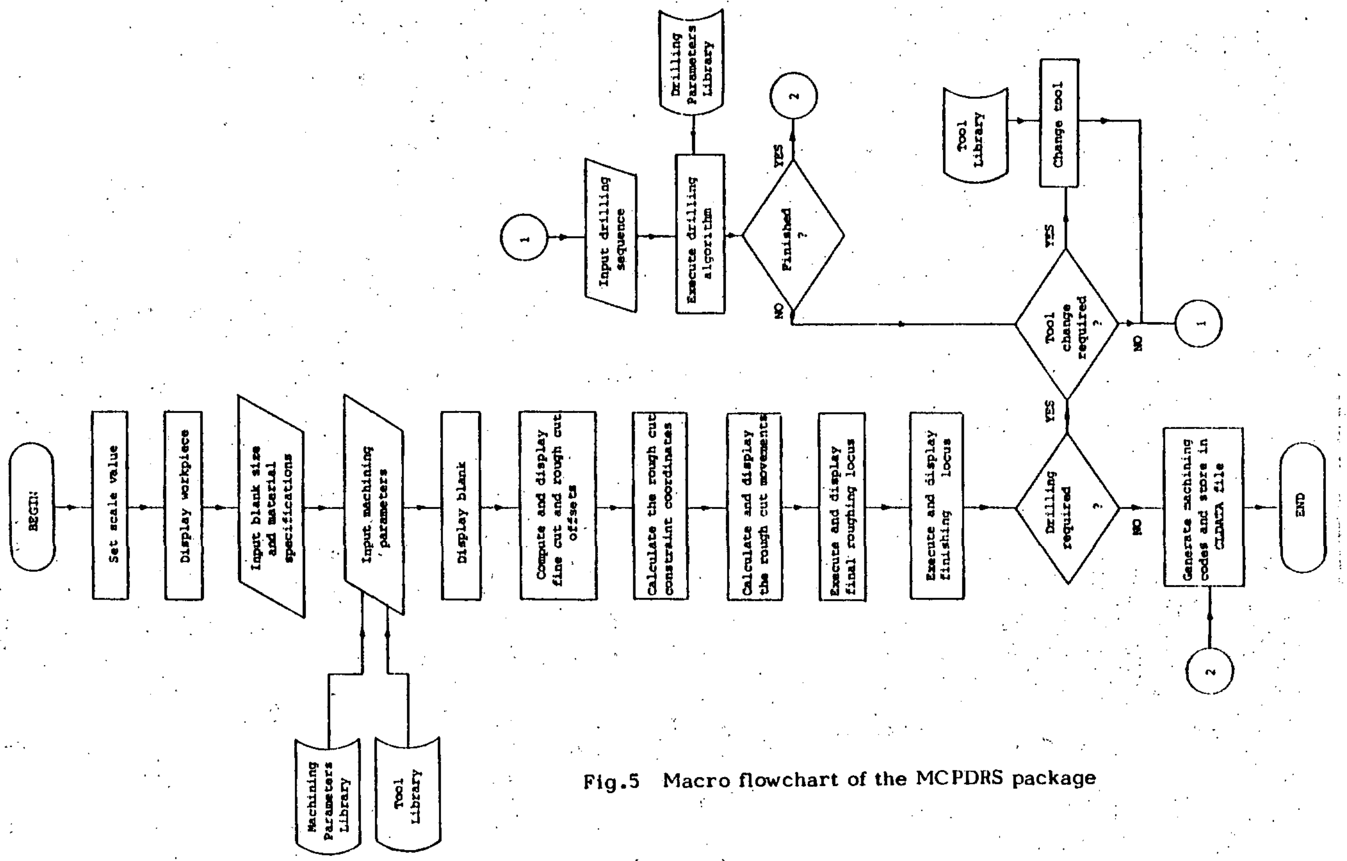

BEGIN
Set scale value
Display workpiece
Input blank size and material specifications
Input machining parameters
Machining Parameters Library
Tool Library
Display blank
Compute and display fine cut and rough cut offsets
Calculate the rough cut constraint coordinates
Calculate and display the rough cut movements
Execute and display final roughing locus
Execute and display finishing locus
Drilling required ?
YES
NO
Generate machining codes and store in CLDATA file
END
2
1
Input drilling sequence
Execute drilling algorithm
Drilling Parameters Library
Finished ?
YES
NO
2
Tool change required ?
YES
NO
1
Tool Library
Change tool

Fig.5 Macro flowchart of the MCPDRS package

The geometry of the workpiece is decoded from the compact data base format and displayed in any desired view. Conventionally, the desired views are the plan, elevation, side elevation, and isometric views. Thus, by displaying the above-mentioned four views a quick and accurate graphical visualisation of geometrical errors or illegal cutter path movements can be detected and corrected.

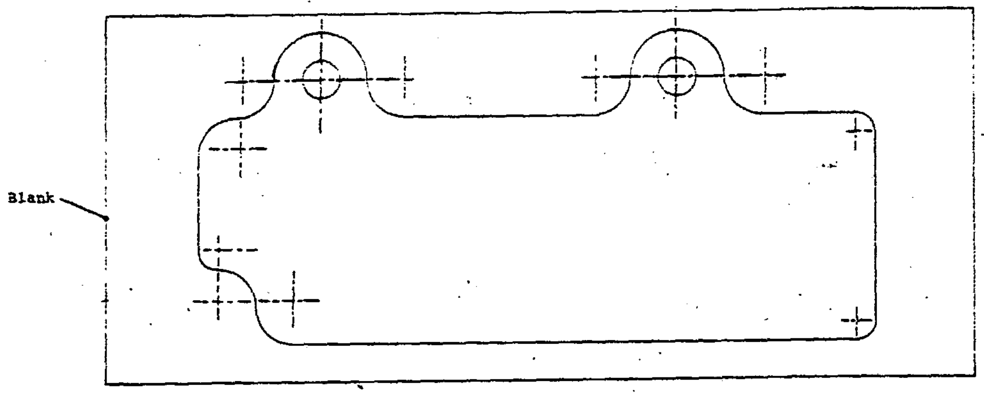
The blank size and geometrical parameters are input while tools and the machining parameters are selected from tooling and machining data libraries. The libraries contain the commonly used tools (cutters), materials and their corresponding machining speeds and feeds. From this information the blank, roughcut and finecut offsets are displayed, Fig. 6.

The spindle speeds and the feed rates are computed using the following formulae:

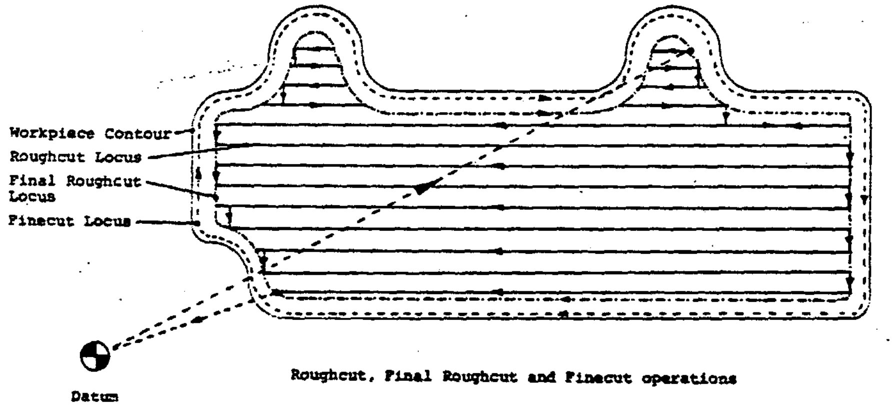
$$\text{Spindle speed} = \frac{\text{cutting speed of the milling cutter}}{\text{(cutter's diameter)}}$$

Feed rate = Number of teeth in the cutter
X recommended feed per tooth
X r/min of the cutter

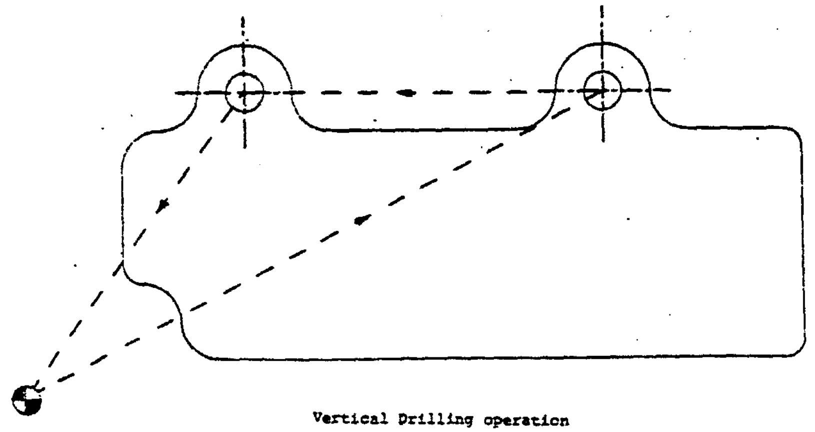
Having computed the speeds and feeds a visual simulation of the roughing process is displayed (either on the vdu or on the plotter), Figs. 7,8, followed by the final roughcut at a reduced feed rate. After a tool change for a finecut tool, usually an end mill, the finishing cuts are displayed at a slower feed rate



AN EXAMPLE OF POCKET MILLING



Roughcut, Final Roughcut and Finecut operations



Vertical Drilling operation

FIG. 7

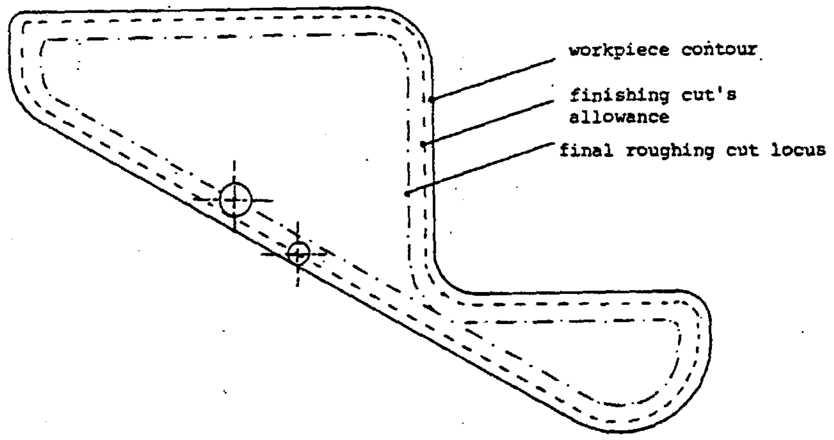


Fig.6a Visualisation of the workpiece's geometry and contouring roughcut and finecut loci

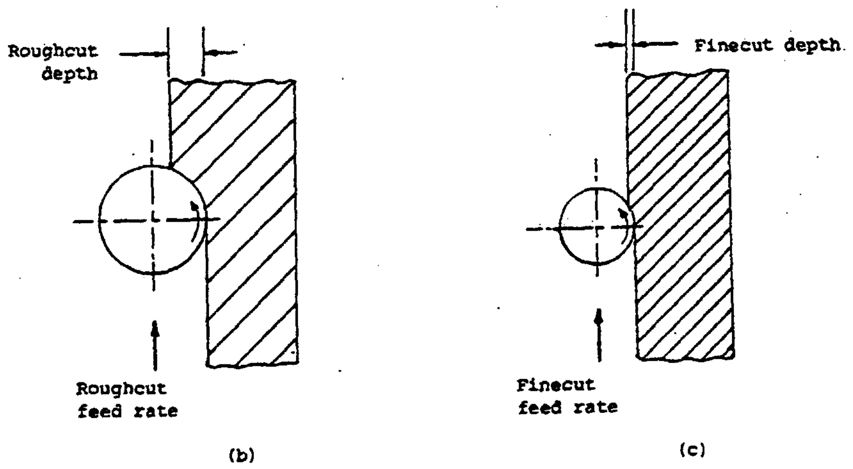


Fig.6 Removal of material during a (b) roughcut, (c) finecut process

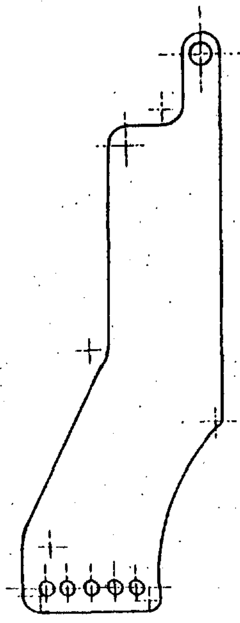


Fig.8a Geometry of the workpiece.

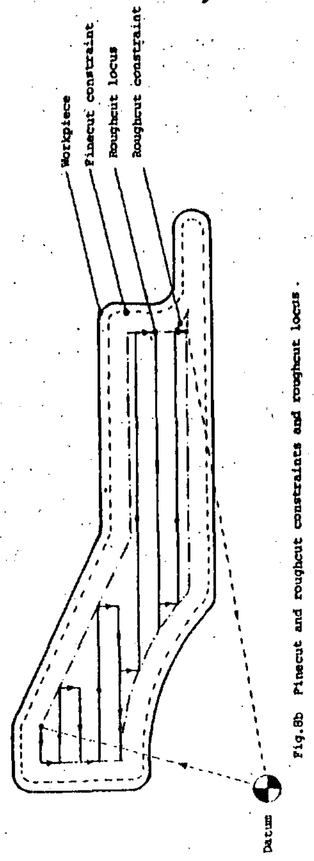


Fig.8b Finecut and roughcut constraints and roughcut locus.

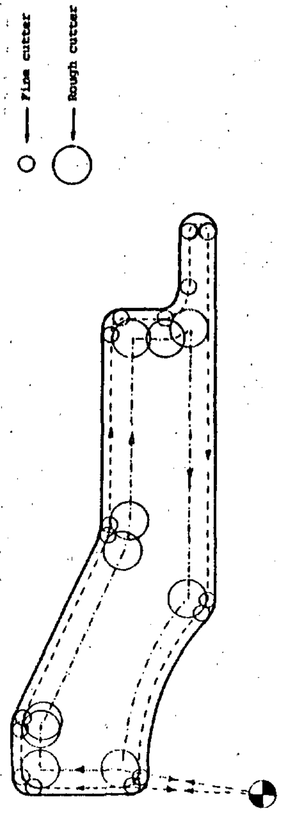


Fig.8c Final roughcut and finecut loci.

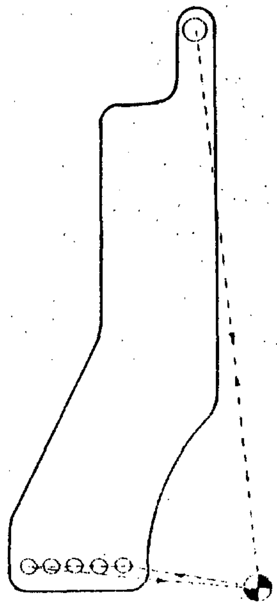


Fig.8d Drilling operation locus.

and spindle speed so as to achieve a good surface finish.

The roughing algorithm has been formulated bearing in mind the limitations and capabilities of the present microcomputer based workstation. Thus the roughing cutter path locus consists of horizontal and vertical (X,Y) movements only. For the majority of cases where the machining time is not a critical factor this algorithm is an acceptable one. However, the modularity of the package enables such algorithms to be optimised or changed without grossly affecting the rest of the package. Furthermore, fast algorithms do exist (6) for the optimisation of the cutter paths for machining arbitrarily shaped pockets, and they can be easily incorporated on a commercial system.

The final operation is drilling. If there are any drilling operations to be performed then they too can be executed by performing tool change(s) and using the milling head as a drilling machine.

The tool changes, spindle speeds, feed rates and cutter paths are stored in an integer binary format in a cutter location data (CLDATA) file for subsequent access by the part program GENERATE package.

The Part Program GENERATE (PPGP) Package

This package is responsible for the conversion of the CLDATA from a binary to an ASCII format and transmitting it, via a standard RS232 serial communication bus to the CNC system(s).

COMMUNICATION BETWEEN THE WORKSTATION AND THE SYSTEM

One of the advantages of the above system is that is eliminates the NC papertape which is conventionally used in transferring NC commands from the CAD workstation to the NC machine tool. The papertape have several disadvantages over the direct communication link, and are as follows:

. relatively slow to generate and to read,
. takes time to be transported from the workstation to the NC machine tool,
. expensive in terms of paper and storage costs,
. easily damaged,
. and are difficuld to identify.

Thus, by generating an ASCII CLDATA file and transmitting it down a standard communication bus (e.g. RS 232) all of the above-mentioned disadvantages are eliminated. Furthermore, the bus can also be utilised for the two way communication. The implications associated with this are almost limitless.

RELATIVE MERITS OF THE CURRENT SYSTEM

Several benefits of the current system can be postulated and summarised as:

1. A reduction in the total system costs due to the introduction of micro-electronic technology in the processing, control and mechanical engineering areas.
2. Low computer software costs due to the modular structure of the packages. The modules' architecture is such that modification and interfacing with other modules is a relatively straightforward task.
3. Interactive and visual (simulation) features lead to the rapid detection and correction of errors.
4. Customised software packages can be quickly and cheaply assembled from existing modules.
5. A general purpose hierarchical DB which is common to all the software packages.
6. A direct communication link between the CAD workstation and the CAM system greatly reduces CAD to CAM data transfer times by eliminating the generation of NC papertapes. This greatly improves design to production times with lower overheads and skilled labour costs.

LIMITATIONS OF THE CURRENT SYSTEM

As briefly mentioned earlier, microcomputer based systems possess certain restrictions in the form of online memory, processing speed, fast backup storage, or existing software backup. This is especially true for most 8-bit microsystems and the above described CAD/CAM packages were formulated with those implications in mind. However, by developing most of the software packages in a high level language and maintaining modularity then the task of transferring them from an 8-bit to a 16-bit microcomputer is a relatively simple one.

The transfer to a 16-bit microcomputer with a link to other computers will be welcomed since it encourages more complex 2D, 2 1/2 D and even 3D milling operations to be performed.

CONCLUSIONS

The design and development of an 8-bit microcomputer based CAD/CAM system for the design and direct manufacture of simple milling components has been described. Consequently, the system has been formulated for 2D and 2 1/2 D pocket milling operations but software modularity enables expansion of the packages to handle more complex 2D, 2 1/2 D and 3D milling operations. Furthermore, other machining operations such as turning, drilling, shaping and grinding can easily be introduced since all that is rewuired is the replacement of the cutter path derivate and the part program generate packages.

The immediate future of this CAD/CAM system is to transfer it onto a 16-bit microsystem and further develop it into a commercially viable low cost system mainly for use in small to medium sized industries.

REFERENCES

1. Gooze, M. "How a 16-bti microprocessor makes it in an 8-bit world". Electronics, September 27, 1979.

2. Dalzell, D.T. "Intelligent machine tools and the microprocessor". Ph.D. Thesis, Imperial College, University of London, 1981.

3. Khurmi, S.K. "Computer Aided Design and Manufacture and the Microprocessor". Ph. D. Thesis, Imperial College, University of London, 1982.

4. Booth, G.M. "Hierarchical configurations for distributed processing". Compcon Digest, September 1977.

5. Pak, H.A. "Microprocessors in Computer Aided Design and in Computer Aided Manufacture in Mechanical Engineering". Ph. D. Thesis, Imperial College, University of London, 1981.

6. Persson, H. "NC machining of arbitrarily shaped pockets". Computer Aided Design, Vol. 10, No. 3.

CENIK OGLASOV

Ovitek - notranja stran (za letnik 1982)

2 stran	28.000 din
3 stran	21.000 din

Vmesne strani (za letnik 1982)

1/1 stran	13.000 din
1/2 strani	9.000 din

Vmesne strani za posamezno številko

1/1 stran	5.000 din
1/2 strani	3.300 din

Oglasi o potrebah po kadrih (za posamezno številko)

2.000 din

Razen oglasov v klasični obliki so zaželjene tudi krajše poslovne, strokovne in propagandne informacije in članki. Cene objave tovrstnega materiala se bodo določale sporazumno.

ADVERTIZING RATES

Cover page (for all issues of 1982)

2nd page	1300 $
3rd page	1000 $

Inside pages (for all issues of 1982)

1/1 page	790 $
1/2 page	520 $

Inside pages (individual issues)

1/1 page	260 $
1/2 page	200 $

Rates for classified advertizing:

each ad	66 $

In addition to advertisment, we welcome short business or product news, notes and articles. The related charges are negotiable.

INFORMATICA 2/1982

COMPILER PERFORMANCE MEASUREMENT AND ANALYSIS

JOZO J. DUJMOVIĆ

DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF BELGRADE, YUGOSLAVIA

UDK: 681.3.068

An empirical analysis of various practical aspects of compiler performance is presented. The basic goal of the analysis is to show the range of possible performance levels corresponding to various language processors. The paper presents two simple formulas for computing (1) the memory requirements and (2) the compilation time of a source program written in a high-level language. Various examples verifying these formulas are also presented. A part of the paper is devoted to a statistical analysis of the memory consumption per source instruction. Finally, various compilers are compared from the standpoint of resource consumption during the execution of compiled programs. The comparison is based on an extensive case study performed using the Amdahl 470-V6 computer.

INTRODUCTION

Modern computer systems regularly offer a variety of options for processing a high-level language program. A user should be interested in selecting the best option, i.e. the option which outperforms all other alternatives. Generally, there are three basic steps in selecting the best computer implementation of an algorithm:

(1) select a proper language,
(2) select one of available compilers for the selected language, and
(3) select appropriate compilation parameters (switches) of the selected compiler.

The selection procedure is to be based on performance considerations since the basic user's goal should be to process his programs using minimal computer resources. Of course, we assume that the user has to pay for the use of computer resources.

In some cases the selection of an appropriate programming language may be a strategic decision which is based on external mandatory requirements, different from the optimization of performance and cost. However, the selection of an efficient compiler and compilation parameters for a given language represents always an important task. It seems that general computer users frequently neglect various compiler-related performance issues and that the default versions of compilers are the most frequently used. Unfortunately, many algorithms are quite sensitive to compilation parameters. Consequently, a skilled user can often substantially improve the performance level of his programs without changing the initial source program. The level of possible improvements is exemplified in subsequent sections.

AN EXPERIMENT WITH THE STRAIGHT SELECTION SORT ALGORITHM

Let us analyze various equivalent implementations of an algorithm using different programming languages. The algorithm to be studied is an internal straight selection sort (SSS) [WIR76] used for sorting an integer vector $A_1, A_2, \dots, A_N$. A simple (but inefficient) version of the SSS algorithm is presented in Fig. 1. In addition, five equivalent implementations of the SSS algorithm using COBOL, Pascal, FORTRAN, BASIC, and MACRO-11 assembly language are also shown in Fig. 1.

All these programs sort the array A following exactly the same algorithm. Consequently, one could expect similar memory consumptions and similar processing times for all equivalent versions of this simple algorithm. In this case, however, the expectations and the reality are two different things. The memory requirement for each (translated) source instruction is shown in Fig. 1. Using PDP 11/34 the minimum space for the assembly language version of the SSS algorithm was 32 bytes, while the default version of OMSI Pascal used eight times more, i.e. 260 bytes. Optimal versions of FORTRAN and Pascal also dramatically differ: Pascal needs two times greater memory space. Similar situation was found using the Honeywell System 6. The COBOL version of the SSS algorithm used three times more space than the same algorithm written in FORTRAN.

Run times for sorting N random numbers using the PDP 11/34 are shown in Fig. 2. The differences are again substantial. The default version of FORTRAN uses an internal counting of source line numbers. For simple source statements this can generate large overheads, and in our case the line-number overhead is greater than 100%. The default version of OMSI Pascal implies a number of tests including the array bounds check and the stack overflow check, as well as slow floating-point calls (compiler automatically generates floating-point machine instructions and, for machines having no floating-point hardware, traps are used for jumping to the floating-point arithmetic subroutines). The resulting overhead for the default version of OMSI Pascal was greater than 50%. The lowest execution speed can be expected when using an interpreter. According to Fig. 2 the BASIC sort was 200 times slower than the assembly language sort.

Similar results were obtained using the Honeywell System 6. The results shown in Fig. 3 correspond to the vector A = -1, -2,..., -10, -1,-2,..., -10,... . The assembly language sort was almost two times faster than the FORTRAN sort. However, the COBOL sort was shown to be 30 times slower than the FORTRAN sort for equivalent two--byte binary components of the vector A. The use of three-digit decimal integers for vector A makes the corresponding COBOL sort 7 times slower than the binary version of the same COBOL sort.

A comparison of space and time requirements for PDP 11/34 is summarized in Fig. 4. Obviously, the obtained results are insufficient for deriving conclusions about general ranking of languages and their compilers. However, these results prove the importance of compiler performance measurement and analysis. They also illustrate the range of possible improvements, and show how dramatic can be the difference between any pair of different programming languages.

```
(1)  i := 1          A STRAIGHT SELECTION
                     SORT ALGORITHM
(2)  j := i
(3)  j := j + 1
(4)  if A_j < A_i then exchange A_j and A_i
(5)  if j<N then (3)
(6)  i = i + 1
(7)  if i<N then (2)
```

C FORTRAN IV	PDP 11/34, RT-11 F. IV V02.04 Opt.	Default	Honeywell 6 FORTRAN 3.0
DO 5 I = 1 , N-1	20	(26)	24
DO 5 J = I+1 , N	28	(34)	22
IF(A(J).GE.A(I)) GOTO 5	4	(10)	20
T = A(I)	6	(12)	10
A(I) = A(J)	2	(6)	12
A(J) = T	2	(6)	10
5 CONTINUE	24	(30)	14
TOTAL =	86	(124)	112 B

* HONEYWELL 6/43 COBOL A 2.0	
I-LOOP.	
PERFORM J-LOOP VARYING I	
FROM 1 BY 1 UNTIL I = N.	62
J-LOOP.	
COMPUTE K = I + 1.	30
PERFORM SWAP VARYING J	
FROM K BY 1 UNTIL J>N.	62
SWAP.	
IF A(J) < A(I)	54
MOVE A(I) TO T	30
MOVE A(J) TO A(I)	50
MOVE T TO A(J).	30
TOTAL =	318 B

10 REM MICROSOFT BASIC IN ROM V.1 REV 3.2	
20 FOR I = 1 TO N-1	16
30 FOR J = I+1 TO N	16
40 IF A(J)<A(I) THEN T=A(I): A(I)=A(J): A(J)=T	44
50 NEXT J,I	10
TOTAL =	86 B

{OMSI Pascal V1.2G}	PDP 11/34, RT-11 Opt.	Default
FOR I := 1 TO N-1 DO	30	(30)
FOR J := I+1 TO N DO	34	(34)
IF A(J)<A(I) THEN	26	(58)
BEGIN		
T := A [I]	12	(28)
A [I] := A [J]	20	(52)
A [J] := T	8	(24)
END	34	(34)
TOTAL =	164	(260) B

; PDP 11/34, MACRO-11 assembly language			
	MOV	#A, R0	4
	MOV	N, R1	4
	DEC	R1	2
SRT1:	MOV	R0,R2	2
	TST	(R2)+	2
	MOV	R1,R3	2
SRT2:	CMP	(R0),(R2)+	2
	BLE	SRT3	2
	MOV	(R0),R4	2
	MOV	-(R2),(R0)	2
	MOV	R4,(R2)+	2
SRT3:	SOB	R3,SRT2	2
	TST	(R0)+	2
	SOB	R1,SRT1	2
		TOTAL =	32 B

Figure 1. A simple version of the straight selection sort and its implementations in COBOL, Pascal, FORTRAN, BASIC, and MACRO-11 assembly language

MEMORY CONSUMPTION FORMULA

The first step in a study of compiler performance should be an analysis of memory requirements of a compiled program. In this section we extend the results presented in [DUJ73] for FORTRAN and ALGOL with similar measurements for a COBOL compiler.

Memory consumption of a compiled program depends on many factors. The most important factor is the number of executable source instructions. Let M denote the memory needed for a compiled program, excluding system subroutines and data used by the program. The executable source instructions directly generate the machine code, but some related code may precede and/or follow the program yielding increased memory requirements. Consequently, a simple deterministic expression of the memory consumption as a function of the number of executable source language instructions N can be formulated as the following memory consumption formula:

$$M = m_o + m_1 N \quad , \quad m_o, m_1 = \text{const.}$$

The parameter m_o represents a fixed part of the memory requirements, i.e. the average memory occupied independently of the number of instructions N. The main components of m_o are:

(a) Introductory code which may be used for various data initializations using system subroutine calls, and for program structuring (e.g. for controlling sections and paragraphs in COBOL).
(b) Internal variables (loop indices, subroutine linkage parameters, and auxiliary local variables).
(c) character strings included in output statements,

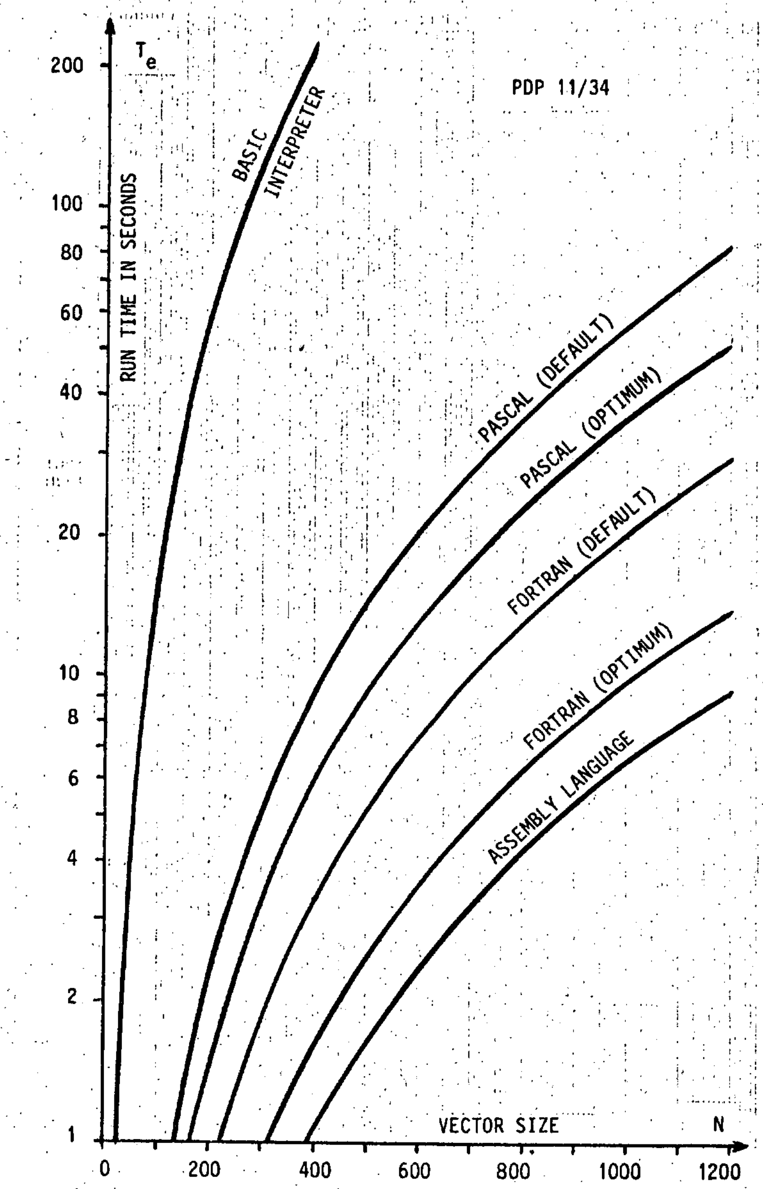


Figure 2. Run times of the straight selection sort using the PDP 11/34 and random numbers

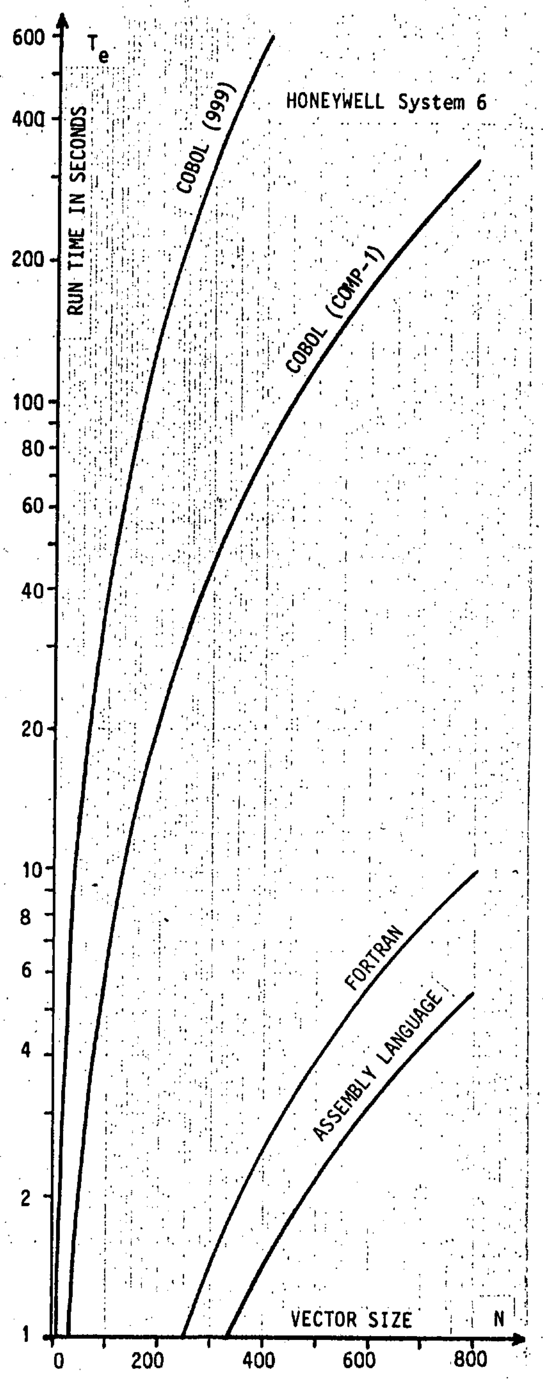


Figure 3. Run times of the straight selection sort using Honeywell System 6

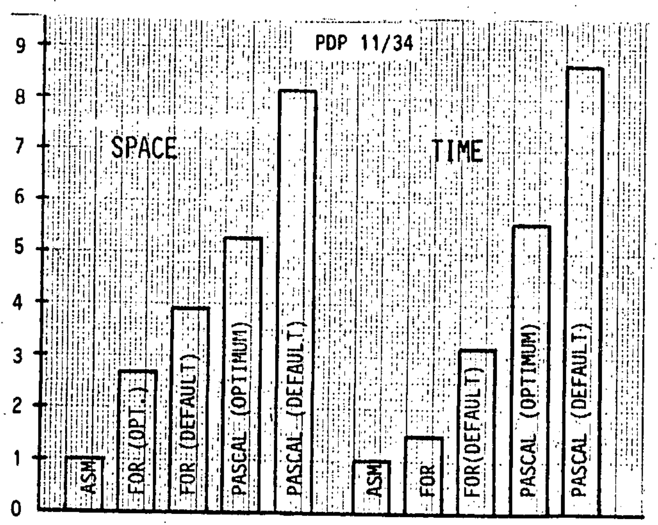


Figure 4. A comparison of space and time requirements for the straight selection sort using assembly language, FORTRAN, and Pascal. (PDP 11/34, RT-11, random numbers)

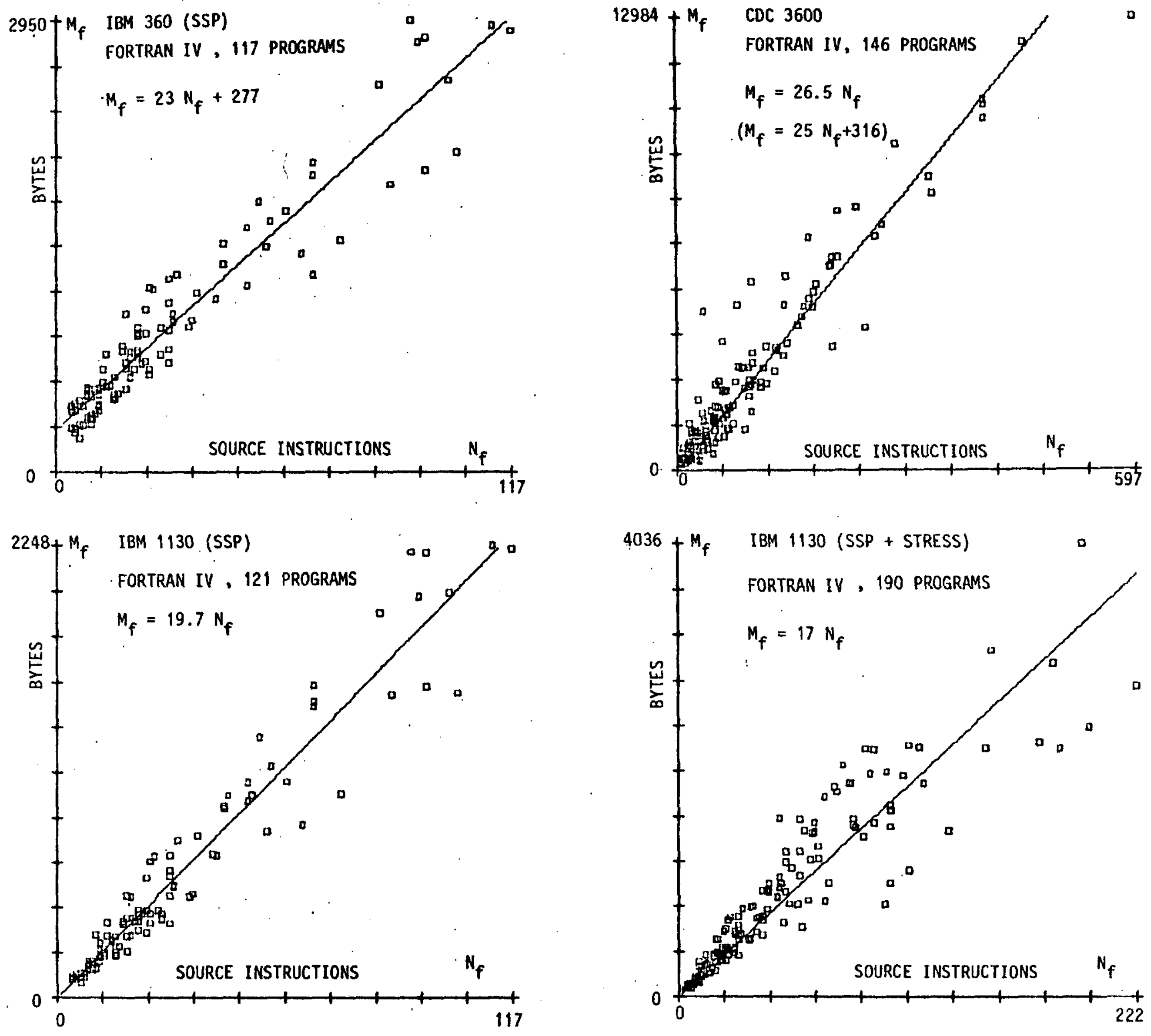


Figure 5. Four examples of the memory consumption formula

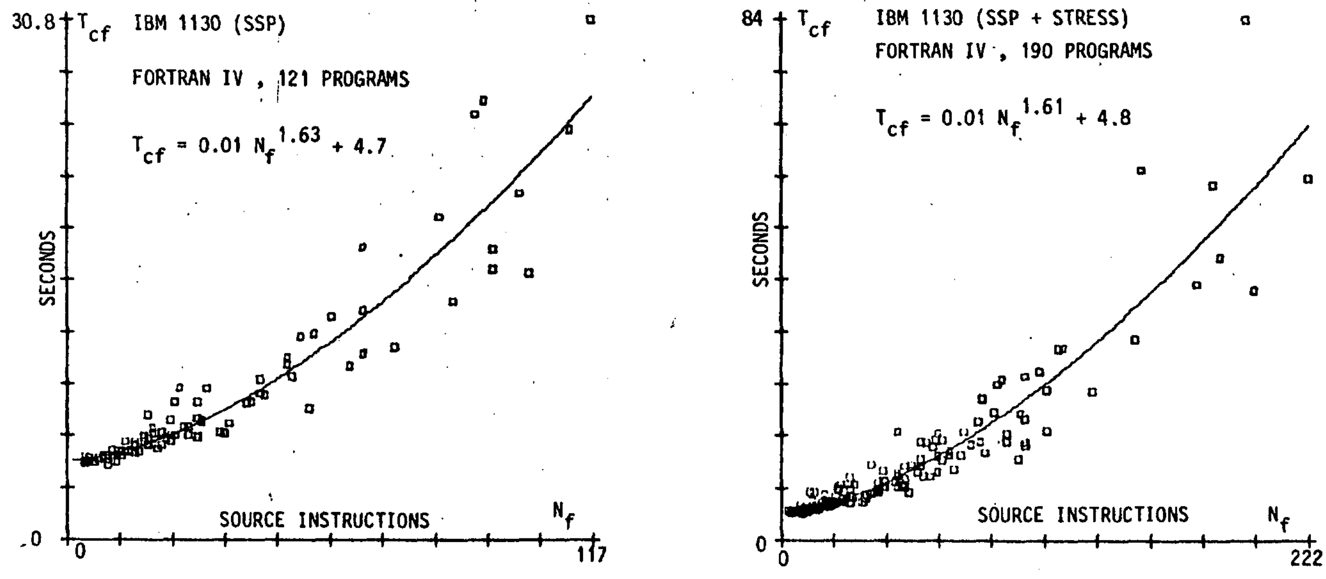


Figure 6. Examples of the compilation time formula

and various numeric constants.
(d) Terminating code at the end of program.

The parameter m_1 represents the average memory occupied by a translated source instruction (or the memory occupied by an average source instruction). Since the memory consumption formula is usually generated using a regression analysis, it follows that m_1 can also be affected by all related memory requirements which are approximately proportional to N. For example, the number of constants, internal variables and section labels may be proportional to N, increasing the computed value of m_1. However, under normal circumstances, these influences are not substantial and they can be analyzed within the context of the code generated by the executable source instructions.

There are two approaches to the measurement and analysis of m_o and m_1. The first approach is based on an analysis of individual programs, and the second approach is based on an analysis of individual compiled instructions.

The first approach assumes the compilation of a set of programs and the measurement of the N,M pair for each compiled program using global statistical data generated by compiler. Since N represents the number of individual executable instructions the computatation of N should take into account the complexity of source instructions, separating multiple and nested instructions. After applying a linear regression to the set of points in the N,M plane the parameters m_o and m_1 can be easily determined.

The second approach consists of analyzing the assembly language equivalents of individual source instructions. The memory space used by a compiled instruction depends on the context in which the instruction appears. Consequently, the memory space per source instruction ($\tilde{m}_1$) is a random variable, and using a sufficiently large sample it is possible to determine the distribution of frequencies of various values of $\tilde{m}_1$. The parameter m_1 can then be defined as the mean value of this distribution. Results of this approach are presented in subsequent sections.

The first approach (i.e. the program-oriented analysis of memory requirements) is exemplified in Fig. 5. The first example shows the memory consumption data for the IBM System /360 Scientific Subroutine Package according to data published in [IBM70] . The resulting parameters are m_1=23 B and m_o=277 B. The memory consumption for the CDC 3600 FORTRAN is slightly larger: m_1=26.5 B (assuming m_o=0) or m_1=25 B and m_o=316 B. Finally, the 16-bit IBM 1130 used for an average FORTRAN statement from 17 to 19.7 bytes.

COMPILATION TIME FORMULA

The compilation process includes two basic steps: (1) fetch and initialization of a selected compiler, and (2) the translation of the source program. The first step needs (approximately) a constant time and represents the fixed part of the compilation time. The time necessary for the second step is a function of the number of executable source instructions (N), and consequently it represents a variable part of the compilation time. Generally, the translation of the source program can be a nonlinear process, i.e. the variable part of the compilation time can be a nonlinear function of the number of source instructions. Let $T_c(N_1)$ and $T_c(N_2)$ denote the compilation times of two related programs containing respectively N_1 and N_2 executable source instructions. Suppose that these programs can be merged yielding the compound compilation time $T_c(N_1+N_2)$. The nonlinearity of the compilation process can be expressed by the following inequality:

$$T_c(N_1+N_2) \geqslant T_c(N_1) + T_c(N_2) \ .$$

This inequality can be explained by the extra time needed for searching various tables generated by the compiler (the tables which correspond to the compound program are greater than the equivalent tables of the constituent programs). Therefore, the variable part of the compilation time can be expressed using the simple power function t_1N^a where $a \geqslant 1$. If the compilation process is linear, i.e. a=1, then t_1 denotes the average compilation time per executable source instruction. In the case of a nonlinear compilation process we have $a>1$ and t_1 denotes the average compilation time for the first executable instruction in the program, while each additional instruction has an increased compilation time.

The above analysis shows that the average compilation time can be computed using the following compilation time formula:

$$T_c = t_o + t_1 N^a \ , \quad a \geqslant 1 \ .$$

In commercial literature the performance of a compiler is frequently expressed using a unique indicator called "the number of translated source statements per minute". This seems to be an incorrect indicator since it is meaningful only if t_o=0 and a=1. However, the measurements of compiler performance show that regularly $t_o > 0$ and very frequently $a>1$.

Two examples of the compilation of FORTRAN programs are presented in Fig. 6. These are typical examples of the nonlinear compilation process. The compilation process of ALGOL programs reported in [DUJ73] also showed the nonlinear characteristic. The compilation time formula for the IBM 1130 ALGOL is

$$T_{ca} = 18.7 + 0.11 \ N_a^{1.3} \quad \text{seconds.}$$

This formula is to be compared with the compilation time formula of the IBM 1130 FORTRAN compiler:

$$T_{cf} = 4.8 + 0.01 \ N_f^{1.61} \quad \text{seconds.}$$

For example, if $N_a=N_f$=100 then the corresponding average compilation times are $T_{ca}(100)$=62.5 seconds and $T_{cf}(100)$= =21.4 seconds.

The examples shown in Fig. 7 illustrate a linear compilation process. The Burroughs 1714 COBOL compiler was analyzed using two independent and different program samples. The first sample consists of 32 small programs ($N_c \leqslant 42$) used as programming exercises. The second sample consists of 41 medium commercial programs ($N_c \leqslant 498$) developed and used by professional programmers. Figure 6 shows separately the results for small programs, the results for medium programs, and the results for the merged sample containing both small and medium programs. It is important to emphasize the consistency of results both for the memory consumption ($10.2 \leqslant m_1 \leqslant 10.7$) and for the compilation time ($1 \leqslant t_1 \leqslant 1.5$). The parameter m_1 has an excellent value which can be explained by an extremely efficient organization of machine instruction coding (see [WIL72]). However, the range of the parameter t_o ($172 \leqslant t_o \leqslant 278$ sec) represents a surprisingly poor result.

AN ANALYSIS OF THE AVERAGE NUMBER OF BYTES PER SOURCE INSTRUCTION

In cases where a compiler can generate an assembly language listing of the translated source program it is possible to analyze various details of the compilation process. In particular, it is easy to count both the number of bytes and the number of machine instructions per source instruction. Such an analysis yields the corresponding frequency distributions which enable a detailed analysis of the parameter m_1. Examples of these analises are presented in Figures 8,9,10, and 11. Figures 8 and 9 illustrate a comparison of FORTRAN and Pascal compilers. The analyzed Pascal compiler generates more machine instructions and needs more memory space for a source instruction than the FORTRAN compiler. However, the machine instructions generated by the Pascal compiler are shorter than those generated by the FORTRAN compiler.

A comparison of the Honeywell System 6 FORTRAN and COBOL compilers (for equivalent sample programs) is shown in Fig. 10. The COBOL compiler needs more memory space and generates more machine instructions than the FORTRAN compiler, but the machine instructions generated by COBOL are shorter than those generated by FORTRAN. The obtained parameter $m_1 \cong 22$ bytes for the Honeywell System 6 COBOL A is substantially greater than the minimum value $m_1 \cong 10$ bytes which corresponds to the

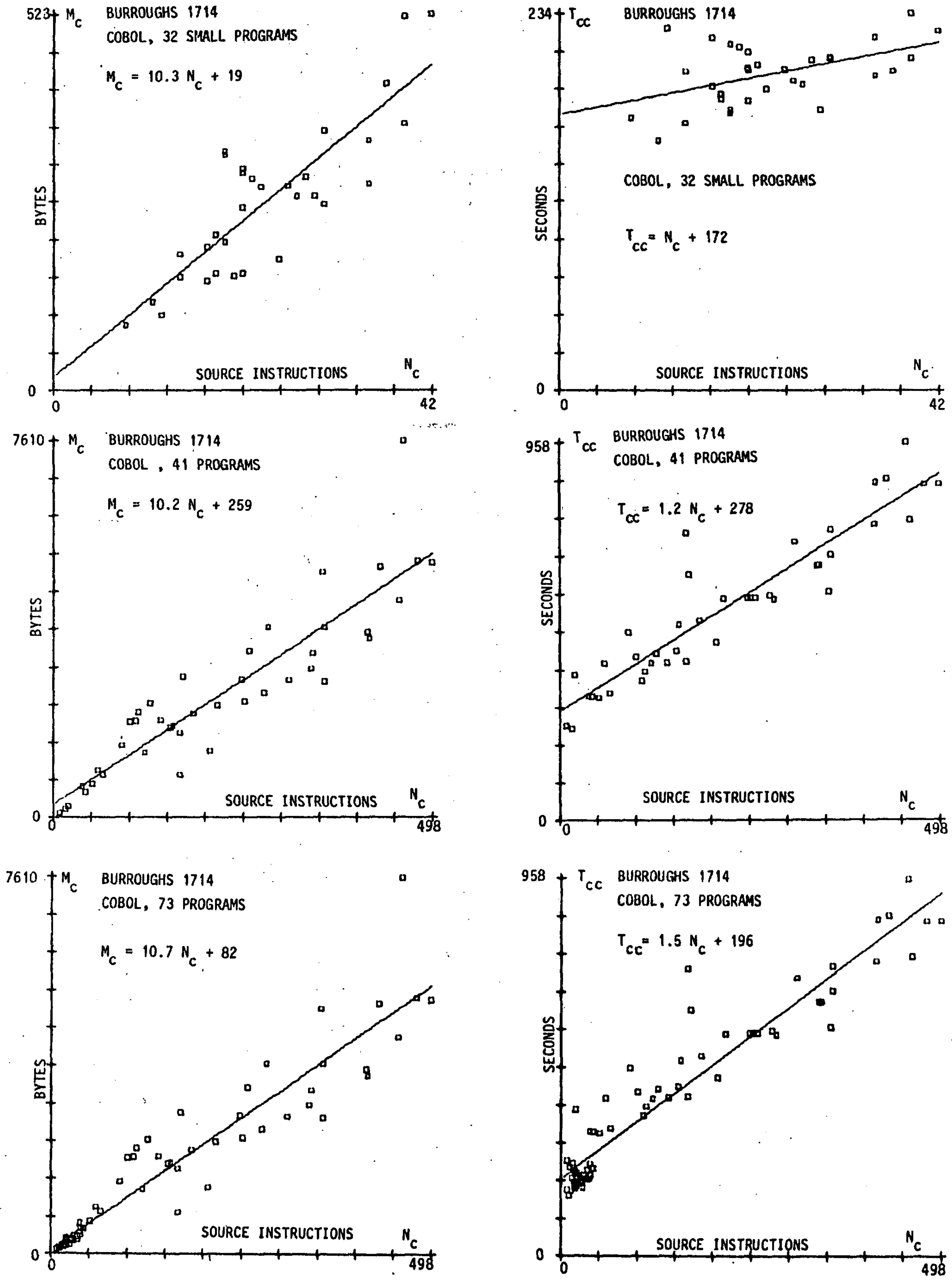


Figure 7. Memory consumption and compilation time for two samples of COBOL programs

```
NUMBER OF SOURCE INSTRUCTIONS =   149
AVERAGE # OF BYTES PER SRC. IN.=   21.89
STANDARD DEVIATION =    15.18

        0         10        20        30
    ----+----+----+----+----+----+----+
  0   2 I
  0   4 I
  2   6 I**
  6   8 I******
 19  10 I*******************
 26  12 I**************************
 14  14 I**************
  7  16 I*******
  7  18 I*******
 17  20 I*****************
  7  22 I*******
  0  24 I
  4  26 I****
  6  28 I******
  3  30 I***                1  56 I*      0  82 I
  4  32 I****               2  58 I**     1  84 I*
  4  34 I****               0  60 I       0  86 I
  6  36 I******             1  62 I*      0  88 I
  4  38 I****               1  64 I*      0  90 I
  0  40 I                   1  66 I*      0  92 I
  1  42 I*                  0  68 I       0  94 I
  2  44 I**                 0  70 I       0  96 I
  0  46 I                   0  72 I       0  98 I
  2  48 I**                 0  74 I       0 100 I
  0  50 I                   0  76 I       0 102 I
  0  52 I                   0  78 I       1 104 I*
  0  54 I BYTES             0  80 I
```

```
AVERAGE # OF MACHINE IN. PER SRC. IN.=    5.77
STANDARD DEVIATION=    4.56
AVERAGE # OF BYTES PER MACHINE IN.=    3.79

        0         10        20        30        40
    ----+----+----+----+----+----+----+----+----+
  0   1 I
 31   2 I*******************************
 31   3 I*******************************
 17   4 I*****************
 22   5 I**********************
  3   6 I***
  6   7 I******
  9   8 I*********
  6   9 I******
  5  10 I*****
  6  11 I******
  0  12 I
  4  13 I****
  1  14 I*
  1  15 I*
  0  16 I
  2  17 I**
  1  18 I*
  0  19 I
  2  20 I**
  0  21 I
  0  22 I
  0  23 I
  0  24 I
  0  25 I
  2  26 I** MACHINE INSTRUCTIONS
```

Figure 8. An analysis of the parameter m_1 for PDP 11/34 and FORTRAN IV V02.04

```
NUMBER OF SOURCE INSTRUCTIONS =   515
AVERAGE # OF BYTES PER SRC. IN.=   23.65
STANDARD DEVIATION =    20.51

        0
    ----+----+----+----+----+----+----+----+----+
  5   2 I***
 75   4 I*****************************************
 27   6 I**************
 20   8 I***********
 43  10 I***********************
 37  12 I********************
 25  14 I*************                      1  72 I*
 31  16 I*****************                  1  74 I*
  8  18 I****                               3  76 I**
 42  20 I**********************             1  78 I*
  4  22 I**                                 2  80 I*
  5  24 I***                                1  82 I*
  8  26 I****                               1  84 I*
 17  28 I*********                          2  86 I*
  4  30 I**                                 0  88 I
 23  32 I************                       1  90 I*
 15  34 I********                           0  92 I
  8  36 I****                               0  94 I
  2  38 I*                                  0  96 I
 15  40 I********                           0  98 I
  4  42 I**                                 0 100 I
  5  44 I***                       FOR      0 102 I
 58  46 I******************************** 0 104 I
  2  48 I*                                  1 106 I*
  4  50 I**                                 0 108 I
  2  52 I*                                  0 110 I
  2  54 I*                                  0 112 I
  3  56 I**                                 0 114 I
  0  58 I                                   0 116 I
  1  60 I*                                  0 118 I
  0  62 I                                   2 120 I*
  1  64 I*                                  0 122 I
  0  66 I                                   0 124 I
  0  68 I BYTES                             2 126 I*
  0  70 I                                   1 128 I*
```

```
AVERAGE # OF MACHINE IN. PER SRC. IN.=    6.88
STANDARD DEVIATION=    6.43
AVERAGE # OF BYTES PER MACHINE IN.=    3.44

        0
    ----+----+----+----+----+----+----+----+----+
104   1 I*****************************************
 38   2 I***************
 53   3 I********************
 61   4 I***********************
 36   5 I**************
 17   6 I*******
 14   7 I*****
 20   8 I********
 10   9 I****
 35  10 I*************
 19  11 I*******
  6  12 I**
 52  13 I******************** FOR
 15  14 I******
  6  15 I**
  5  16 I**
  0  17 I
  3  18 I*                     0  36 I
  1  19 I                      0  37 I
  1  20 I                      0  38 I
  0  21 I                      0  39 I
  2  22 I*                     4  40 I**
  0  23 I                      0  41 I
  3  24 I*                     1  42 I
  5  25 I**
  0  26 I
  3  27 I*
  1  28 I
  0  29 I
  0  30 I
  0  31 I
  0  32 I
  0  33 I
  0  34 I MACHINE INSTRUCTIONS
  0  35 I
```

Figure 9. An analysis of the parameter m_1 for PDP 11/34 and OMSI Pascal V1.2 G

```
NUMBER OF SOURCE INSTRUCTIONS =   137
AVERAGE # OF BYTES PER SRC. IN.=  17.72
STANDARD DEVIATION =   22.31


        0
    ----+----+----+----+----+----+----+----+----+
  1    2 I*
 43    4 I****************************************
 10    6 I**********
 12    8 I************
 11   10 I***********
 13   12 I*************       0  66 I     0  96 I
  7   14 I*******             0  68 I     0  98 I
  3   16 I***                 0  70 I     0 100 I
  2   18 I**       2  42 I**  0  72 I     0 102 I
  3   20 I***      0  44 I    0  74 I     0 104 I
  5   22 I*****    0  46 I    0  76 I     0 106 I
  1   24 I*        2  48 I**  0  78 I     1 108 I*
  1   26 I*        2  50 I**  0  80 I     0 110 I
  0   28 I         2  52 I**  0  82 I     0 112 I
  0   30 I         0  54 I    0  84 I     0 114 I
  5   32 I*****    2  56 I**  4  86 I**** 0 116 I
  1   34 I*        0  58 I    0  88 I     0 118 I
  2   36 I**       0  60 I    0  90 I     0 120 I
  0   38 I         0  62 I    1  92 I*    0 122 I
  0   40 I         0  64 I    0  94 I     1 124 I*

     BYTES
```

```
AVERAGE # OF MACHINE IN. PER SRC. IN.=    3.54
STANDARD DEVIATION=    4.22
AVERAGE # OF BYTES PER MACHINE IN.=    5.01


           0
       ----+----+----+----+----+----+----+----+----+
  44     1 I****************************************
  33     2 I*********************************
  18     3 I******************
  15     4 I***************
   8     5 I********
  10     6 I********** -
   0     7 I
   0     8 I
   0     9 I
   0    10 I
   0    11 I
   0    12 I
   2    13 I**
   0    14 I              0  21 I
   0    15 I              0  22 I
   0    16 I              0  23 I
   2    17 I**            0  24 I
   2    18 I**            0  25 I
   2    19 I**            0  26 I
   0    20 I              1  27 I*

      MACHINE INSTRUCTIONS
```

Honeywell System 6 , FORTRAN 3.0

```
NUMBER OF SOURCE INSTRUCTIONS =   146
AVERAGE # OF BYTES PER SRC. IN.=   21.99
STANDARD DEVIATION =   16.67


        0         10        20        30
    ----+----+----+----+----+----+----+
  0    2 I
  0    4 I
  8    6 I********
  1    8 I*
 18   10 I******************
 21   12 I*********************
 26   14 I**************************
 10   16 I**********
  8   18 I********
  7   20 I*******      1  48 I*
  5   22 I*****        4  50 I****    2  76 I**
  3   24 I***          0  52 I        0  78 I
  5   26 I*****        1  54 I*       0  80 I
  4   28 I****         0  56 I        0  82 I
  4   30 I****         0  58 I        0  84 I
  0   32 I             1  60 I*       0  86 I
  1   34 I*            2  62 I**      0  88 I
  1   36 I*            1  64 I*       0  90 I
  3   38 I***          0  66 I        0  92 I
  4   40 I****         0  68 I        0  94 I
  2   42 I**           2  70 I**      0  96 I
  0   44 I             0  72 I        0  98 I
  0   46 I             0  74 I        1 100 I*

   BYTES
```

```
AVERAGE # OF MACHINE IN. PER SRC. IN.=    6.73
STANDARD DEVIATION=    5.04
AVERAGE # OF BYTES PER MACHINE IN.=    3.27


          0         10        20        30        40
      ----+----+----+----+----+----+----+----+----+
   0    1 I
   9    2 I*********
  31    3 I*******************************
  17    4 I*****************
  30    5 I******************************
  12    6 I************
  10    7 I**********
   3    8 I***
   7    9 I*******
   6   10 I******
   2   11 I**
   0   12 I
   0   13 I
   0   14 I
   4   15 I****             0  24 I
   3   16 I***              0  25 I
   5   17 I*****            0  26 I
   2   18 I**               0  27 I
   1   19 I*                0  28 I
   1   20 I*                0  29 I
   1   21 I*                0  30 I
   1   22 I*                0  31 I
   0   23 I                 1  32 I*

     MACHINE INSTRUCTIONS
```

Honeywell System 6 , COBOL A

Figure 10. A comparative analysis of the parameter m_1 for Honeywell System 6 FORTRAN 3.0 and COBOL A

Burroughs 1714 COBOL. However, the consumption of memory per source instruction can vary depending on the complexity of program. As opposed to rather complex programs used for generating the histograms presented in Fig. 10, Figure 11 shows the histogram for a typical simple commercial program. Now $m_1 \cong 15$ bytes and the histogram enables the identification of lines which correspond to various individual source instructions.

The parameters m_0 and m_1 do not depend exclusively on compiler performance. They can be substantially affected by the processor architecture including the available machine instruction set, addressing modes, and processor registers. However, if m_0 and m_1 are determined using the same hardware but different compilers, then the resulting performance levels can be considered a direct consequence of the properties of the analyzed compilers.

A COMPARATIVE ANALYSIS OF FORTRAN, PASCAL AND PL/1 COMPILERS BASED ON THE RUN TIME RESOURCE CONSUMPTION OF A BENCHMARK PROGRAM

The comparative analyses of compiler performance that appear in the literature are usually restricted to the comparison of various compilers for a given language (e.g. see [WOR76] for PL/1 and [WIC73] for ALGOL). However, the computer users are generally interested in

```
NUMBER OF SOURCE INSTRUCTIONS =    354
AVERAGE # OF BYTES PER SRC. IN.=    14.90
STANDARD DEVIATION =       7.23

          0                               ABSOLUTE FREQUENCY
      ----+----+----+----+----+----+----+----+----+
 0     2 I
52     4 I************************** GO TO
 1     6 I
 0     8 I                                    MOVE , ADD
87    10 I*******************************************
 0    12 I
55    14 I*************************** IF
 8    16 I****
32    18 I****************
68    20 I********************************** PERFORM
16    22 I********
10    24 I*****
 0    26 I
12    28 I******
 3    30 I*
 0    32 I
10    34 I*****

     BYTES PER SOURCE INSTRUCTION


NUMBER OF SOURCE INSTRUCTIONS =    354
AVERAGE # OF MACHINE IN. PER SRC. IN.=    4.23
STANDARD DEVIATION=     2.10
AVERAGE # OF BYTES PER MACHINE IN.=     3.52

          0                               ABSOLUTE FREQUENCY
      ----+----+----+----+----+----+----+----+----+
52     1 I************************** GO TO
 1     2 I                                    MOVE , ADD
87     3 I*******************************************
60     4 I****************************** IF
76     5 I**************************************
33     6 I****************                    PERFORM
20     7 I**********
11     8 I*****
 4     9 I**
10    10 I*****

  MACHINE INSTRUCTIONS PER SOURCE INSTRUCTION
```

Figure 11. An example of the m_1-analysis for a simple commercial program (Honeywell System 6, COBOL I)

TABLE 1. A SURVEY OF FORTRAN, Pascal AND PL/1 COMPILERS USED ON AMDAHL 470-V6 (August 1980)

COMPILER	LANGUAGE	PRECISION		LEVEL OF OPTIMIZATION			
		Single	Double	0	1	2	3
FORTG	FORTRAN	+	+	+	-	-	-
FORTH	FORTRAN	+	+	+	+	+	-
FORTX	FORTRAN	+	+	+	+	+	-
FORTQ	FORTRAN	+	+	+	+	+	+
WATFIV	FORTRAN	+	+	+	-	-	-
WATPAS*	Pascal	-	+	+	-	-	-
P-8000	Pascal	-	+	+	-	-	-
PASCAL VS*	Pascal	+	+	-	+	-	-
PLC	PL/1	-	+	+	-	-	-
PL1F	PL/1	+	+	+	-	-	-
PL1X	PL/1	+	+	+	+	-	-

* August 1981

TABLE 2. A COMPARISON OF RESOURCE CONSUMPTION FOR COMPILERS USED ON AMDAHL 470-V6 (August 1980)

COMPILER	OPTIMIZATION LEVEL	TOTAL RELATIVE CONSUMPTION OF RESOURCES
FORTQ	3	1.00
FORTQ	2	1.03
FORTH	2	1.3
FORTX	2	1.3
FORTX	1	1.4
FORTQ	1	1.4
FORTH	1	1.6
FORTG	0	2.0
FORTH	0	2.1
FORTQ	0	2.1
FORTX	0	2.2
PL1X (no test)	1	2.2
PL1X (no test)	0	2.3
PL1F	0	2.8
PASCAL 8000	0	3.1
PASCAL VS*	1	3.1
PLC	0	8.8
PL1X (all tests)	0	17.1
WATFIV	0	18.4
WATPAS*(all tests)	0	162.8

* August 1981

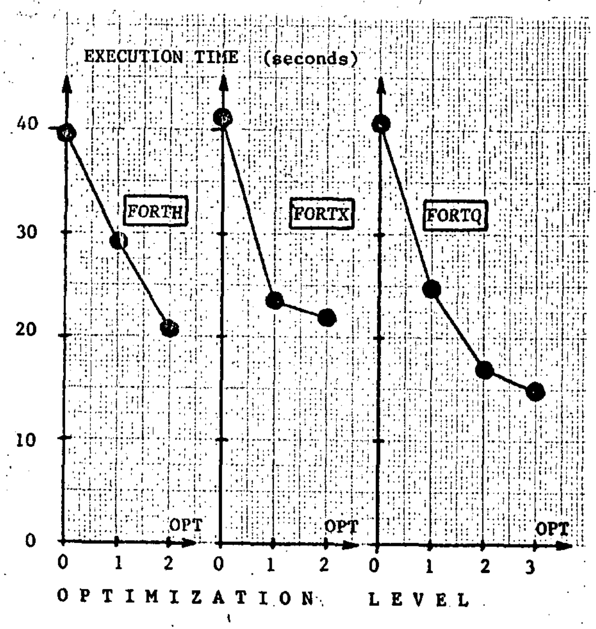
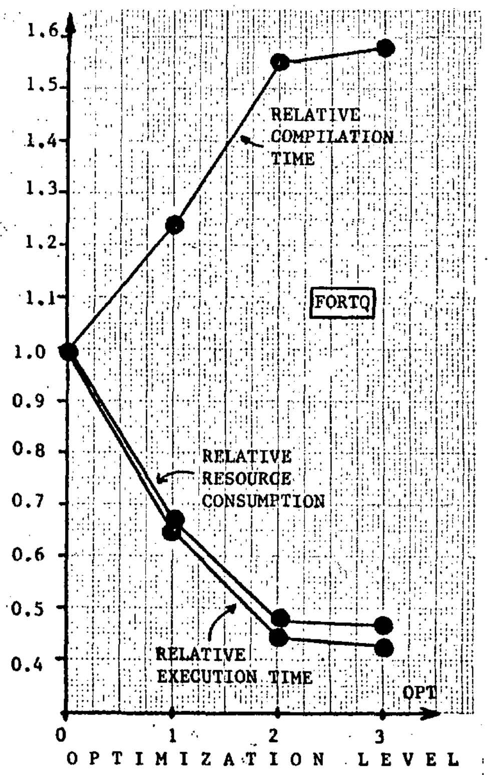


Figure 12. Execution time as the function of the level of optimization



FFigure 13. The results of the optimization process

reducing the cost of computing and this implies a comparison of different languages available in a computing center. In this section we present a comparative analysis of eleven compilers for FORTRAN, Pascal, and PL/1 running on the Amdahl 470-V6. The analysis is aimed at showing how the proper selection of compiler reduces both the computer workload and the cost of computing for the user.

A standard matrix-inversion-based benchmark program (see [DUJ80]) was used for comparing various FORTRAN, PL/1 and Pascal compilers. The FORTRAN, PL/1 and Pascal versions of the program were equivalent, used the same hardware (Amdahl 470-V6) and the same operating system (MVS), and generated equivalent results. The only difference was in the code generated by the compilers.

The analyzed compilers are described in Table 1. Optimization level 0 denotes no optimization. Since three of the available compilers default to double precision, only the results for double precision are presented. These resultis for total resources consumption (according to the accounting routine) are shown in Table 2. From this Table the following conclusions can be derived:

(1) It is rather easy to spend 100% more computer resources than necessary for solving a problem. Conversely, by careful selection of the compiler it is frequently possible to halve the workload and the user's cost of computing.
(2) For numerical problems, FORTRAN compilers are still substantially more efficient than PL/1 and Pascal compilers (of course, this is not proved generally, but it holds on Amdahl 470-V6, IBM 30XX and similar machines).
(3) It is very important to know how to use the optimization features of the compilers because this is the key to reducing workload and costs.
(4) The development compilers like PLC, WATFIV and WATPAS must be used with extreme care since they can consume substantial amounts of computer resources (during execution of program).
(5) "Tests" in the PLIX compiler are dangerous consumers of computer resources (this also frequently holds for various run-time checking implemented in other compilers).

The reduction of computing cost using the optimization procedures can be substantial, as shown in Figures 12 and 13. (The optimization procedures are described in [SCA80] and [RUS72].) According to Fig. 13 the optimization procedures increase the compilation time. However, the compilation time is regularly much shorter than the execution time. Consequently, the reduction of resource consumption is almost as efficient as the reduction of the execution time shown in Fig. 13.

ACKNOWLEDGMENT

The author is grateful to Mr. Laslo Kraus for his contributions to this paper.

REFERENCES

[DUJ73] Dujmovic J.J. and Klem N. A COMPARISON OF FORTRAN AND ALGOL COMPILERS. (In Serbo-Croatian) Informatica 1973.

[DUJ80] Dujmovic J.J. WORKLOAD CHARACTERIZATION, BENCHMARKING, AND THE CONCEPT OF TOTAL RESOURCES CONSUMPTION. Proceedings of the Second International Conference on Computer Capacity Management. San Francisco 1980, pp151-163.

[IBM70] IBM:System /360 Scientific Subroutine Package. Version III, Fifth Edition, 1970. Form No. GH20-0205-4.

[SCA80] Scarborough, R.G. and H.G. Kolsky. IMPROVED OPTIMIZATION OF FORTRAN OBJECT PROGRAMS. IBM J. Res. Develop. Vol. 24, No.6, Nov. 1980.

[WIC73] Wichmann, B.A. ALGOL 60 COMPILATION AND ASSESMENT. Academic Press, London 1973.

[RUS72] Rustin, R. (Ed.). DESIGN AND OPTIMIZATION OF COMPILERS. Prentice-Hall, 1972.

[WIL72] Wilner, W.T. BURROUGHS B1700 MEMORY UTILIZATION. Proc. AFIPS FJCC Vol. 41 , 1972.

[WIR76] Wirth, N. ALGORITHMS+DATA STRUCTURES = PROGRAMS. Prentice-Hall 1976.

[WOR76] Wortman, D.B., Khaiat, P.J. and D.M. Lasker. SIX PL/1 COMPILERS. Software- Practice and Experience Vol. 6, 411-422 (1976).

INFORMATICA 2/1982

TRANSPARENTNO MULTIPROCESIRANJE –(MIKRO) RAČUNALNIŠKI SISTEMI JUTRIŠNJEGA DNE

DUŠAN PEČEK, BORUT KASTELIC, RUDOLF MURN

UDK: 681.3.012

INSTITUT JOŽEF STEFAN, LJUBLJANA

Namen članka je seznaniti bralca z novim pristopom v arhitekturi mikroprocesorjev. S pristopom, ki pomeni revolucijo in bo imel v bodočnosti daljnosežne posledice za snovanje katerega koli (mikro) računalniškega sistema.

TRANSPARENT MULTIPROCESSING - (MICRO) COMPUTER SYSTEMS OF TOMORROW. The purpose of the article is to acquaint the reader with a new approach in microprocessor architecture. With the approach that means a revolution and that will have far-reaching consequences for projecting any (micro) computer system in future.

1. UVOD

Beseda bo tekla o novi 32 bitni procesorski družini, ki so jo razvili in izdelali v tvrdki INTEL. Le-ta je s tem postala nesporni nosilec razvoja bodočih mikroračunalniških družin. Novo družino predstavljajo tri integrirana vezja z oznakami: iAPX43201, iAPX43202, iAPX43203. Vezji iAPX 43201 in iAPX43202 nosita skupno ime General Data Processor (GDP), vezje iAPX 43203 pa se imenuje Interface Processor (IP). Povezava vseh treh vezij v celoto pa se imenuje Sistem 432. Namen članka ni opisovati zgradbe, funkcij in medsebojnega delovanja teh treh elementov, pač pa prikazati možnosti, ki so dane na osnovi nove arhitekture.

Pri vsaki novosti se vedno vprašamo zakaj? Odgovor je na dlani:

> Nova arhitektura zato in tako, da bo bistveno zmanjšala ceno programske opreme.

Arhitektura 432 je programsko usmerjena. Za ilustracijo pa tudi morda za povečanje zanimanja, navedimo nekaj najbolj splošnih lastnosti:

- Omogočena je gradnja različnih sistemov z različnimi lastnostmi, ne da bi bilo potrebno spreminjati programsko opremo. Programi pisani za 432 delujejo na sistemih z enim, dvema ali več procesorji. Vsak sistem 432 lahko izboljšamo tako, da mu dodamo nov GDP, ne da bi bilo potrebno, razen mehanskega posega, še karkoli spreminjati. Procesno moč sistema VAX-PDP je možno enostavno doseči s paralelnim delovanjem tridesetih GDP-jev.

- V sistemu 432 so vse sistemske funkcije standardizirane in ker so realizirane v materialni opremi, je izvrševanje le-teh izredno hitro. Tipične sistemske funkcije so npr.: dodeljevanje spomina, razvrščanje posameznih opravil (multiprogramiranje), dodeljevanje prostih procesorjev (multiptocesiranje).

- V sistemu 432 je nemogoče zagrešiti naslednje programske napake, ki smo jih lahko delali do sedaj.

- program ne more nikdar poseči v spominsko področje, ki mu ni dodeljeno:
- ne more čitati iz spominskih lokacij, ki so mu dodeljene samo za vpisovanje in obratno:
- nad poljubno podatkovno strukturo je možno narediti samo tiste operacije, ki so ji predpisane (npr.: nemogoče je izvrševati podatek kot instrukcijo).

Če na kratko strnemo primerjavo med dosedanjimi uR sistemi in sistemom 432, lahko rečemo, da je potrebno funkcije, ki so v sistemih 432 realizirane v materialni opreme enkrat za vedno, v dosedanjih sistemih realizirati s programsko opremo na nivoju operacijskih sistemov in prevajalnikov. Razlika v zanesljivosti delovanja in hitrosti izvajanja funkcij je milo rečeno, ogromna.

Članek je razdeljen v dva dela. V prvem delu se bomo srečali z novo terminologijo in razlago nekaterih bistvenih pojmov. Drugi del pa govori o rešitvi problema sočasnega izvajanja programov kot uvod v transparentno multiprocesiranje.

2. POJEM OBJEKTA

V sistemu 432 sta definirana dva tipa podatkovnih struktur:

- podatkovna struktura primitiv
- podatkovna struktura objekt

a) Sistem 432 pozna osem vrst primitivov:

- CHARACTER (8 bitov)
- SHORT INTEGER (16 bitov)
- SHORT ORDINAL (16 bitov)
- INTEGER (32 bitov)
- ORDINAL (32 bitov)
- SHORT REAL (32 bitov)
- REAL (64 bitov)
- TEMPORARY REAL (80 bitov)

Primitivi imajo tri bistvene značilnosti:

- *so podatkovne strukture, ki na urejen način vsebujejo neko informacijo;*
- *nad množico primitivov je definirana množica operatorjev*
- *primitive naslavljamo kot celoto.*

b) Pojem objekt bomo pojasnili s pomočjo poenostavljenega prikaza razvoja mikroprocesorjev. Prvi procesorji so vsebovali zelo enostavno materialno opremo za izvajanje operacij (npr.: MOVE BYTE, ADD INTEGER, ipd.) in niso bili sposobni izvajati operacij aritmetike v plavajoči vejici. To je bilo potrebno realizirati s programsko opremo. Razvoj tehnologije je pripomogel k materialni realizaciji kompleksnejših operatorjev kot npr.: ADD, MULTIPLY, DEVIDE, itd. kar je bistveno olajšalo implementacijo aritmetike v plavajoči vejici.

Pri razvoju sistema 432 pa je bil storjen še en bistven korak naprej. Kot smo že omenili, so v 432 z materialno opremo realizirane tudi sistemske funkcije (operacije).

- Implementacija aritmetike v plavajoči vejici z materialno opremo pomeni, da mora materialna oprema operirati s podatkovnimi strukturami, ki predstavljajo števila zapisana v obliki plavajoče vejice.

- Implementacija sistemskih funkcij v materialni opremi pomeni, da mora materialna oprema operirati s podatkovnimi strukturami, ki so operandi naslednjih operacij (funkcij):

- *dodeljevanje spomina*
- *dodeljevanje procesov*
- *medprocesna komunikacija*

Te podatkovne strukture (operande) imenujemo objekte (objekt). Slika 2 prikazuje sistemske opreacije in pripadajoče objekte.

OPERACIJA	*OBJEKT*
dodeljevanje procesov	*objek procesa* *objekt procesorja* *objekt dodeljevalnih vrat*
dinamično dodeljevanje spomina	*objekt izvora spominskega prostora*
medprocesna komunikacija	*objekt komunikaciskih vrat*

Slika 2.

Imena objektov predstavljajo novo terminologijo, ki jo bo nadaljnje spoznavanje in delo s sistemi 432 udomačilo in dopolnilo.

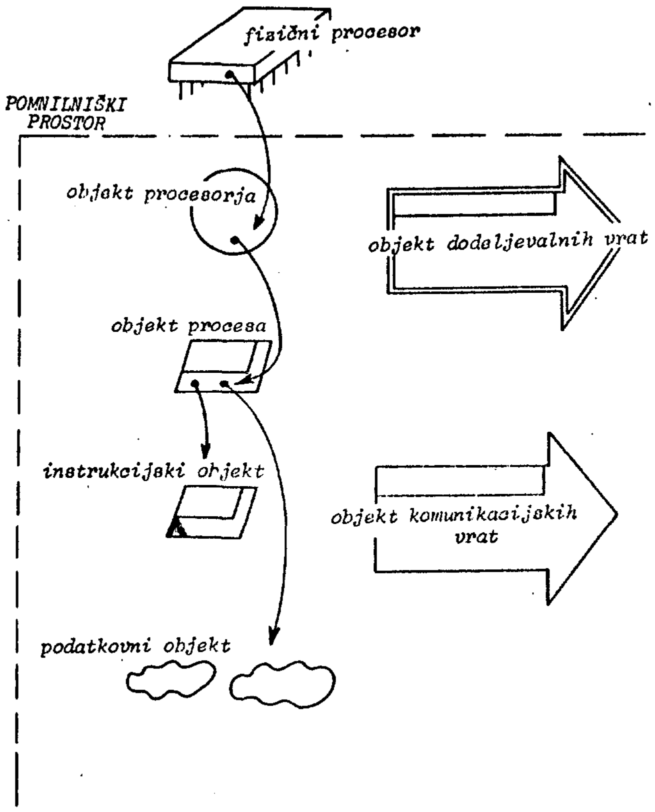
Strnimo lastnosti objekta:

- Objekt je podatkovna struktura, s katero lahko manipuliramo na točno določen način. Tako s pomočjo vgrajene materialne opreme, kot s pomočjo programske opreme. Za določeno vrsto objekta je možen samo eden od treh načinov manipuliranja:

- *z materialno opremo*
- *z materialno in/ali programsko opremo*
- *s programsko opremo*

- *Objekt naslavljamo kot celoto*
- *Objekt ima značko, ki določa njegov tip.*

V nadaljevanju članka bomo objekte grafično predstavljali s pomočjo simbolov. Slika 3. zaradi enostavnosti vsebuje samo tiste objekte, ki jih potrebujemo za razumevanje sočasnega izvajanja programov in transparentnega multiprocesiranja.



Slika 3.

Množica puščic na sliki 3. je posebnost sistema 432. Le-ta pozna samo objektno orientirano adresiranje. To je metoda, ki omogoča objektu, da naslovi drug objekt samo v primeru, če vsebuje kazalec nanj. Med samim izvajanjem programov se stanje kazalcev dinamično spreminja. Slika 3. predstavlja obenem tudi strukturo kakšnega programa pisanega za sistem 432.

3. MEDPROCESNA KOMUNIKACIJA

Na kratko bomo bralca seznanili kako so v sistemih 432 rešeni problemi medprocesne komunikacije pri paralelnem izvajanju procesov.

Kaj je to proces?
Proces je enota programske opreme, ki je lahko dodeljena procesorju za izvrševanje.

Kaj je to program?
Program je enota programske opreme, ki izvršuje neko nalogo. (Program je največkrat zbirka procesov).

Medprocesna komunikacija zahteva vpeljavo novega objekta - objekt komunikacijskih vrat.

Komunikacijska vrata predstavljajo vmesnik med dvema asinhronima sočasnima procesoma, preko katerega procesa izmenjujeta sporočila.

Kaj je to sporočilo?
Sporočilo je lahko katerikoli objekt.

Sporočila in operatorji

Sistem 432 vsebuje dva osnovna operatorja, ki lahko delujeta nad sporočili: SEND in RECEIVE.

Operator SEND ima dva operanda: sporočilo in naslov komunikacijskih vrat. Pred izvršitvijo operatorja (instrukcije) SEND vsebuje proces sporočilo ki naj bo odposlano. Po izvršitvi se sporočilo nahaja v objektu komunikacijskih vrat.

Operator RECEIVE ima en operand: naslov komunikacijskih vrat. Po izvršitvi operatorja RECEIVE se sporočilo prenese v proces. Če sporočila ni v trenutku, ko je proces pričel z izvrševanjem operatorja RECEIVE, proces čaka v komunikacijskih vratih, procesor pa preide na izvrševanje kakšnega drugega procesa. V sistemu 432 obstajajo tudi izvedenke osnovnih operatorjev kot npr.: CONDITIONAL SEND, CONDITIONAL RECEIVE ipd. Pri izvrševanju operatorjev SEND in RECEIVE seveda ne gre za dejansko kopiranje (prenašanje) sporočil, pač pa se dinamično prenašajo samo kazalci na sporočila: dinamično se spreminja množica kazalcev med objekti.

4. TRANSPARENTNO MULTIPROCESIRANJE

Arhitektura sistema 432 je zasnovana tako, da podpira paralelno izvajanje programov na sistemih z enim, dvema ali več procesorji. Izvajanje istih programov na različnih sistemih z različnim številom procesorjev ne zahteva spremembe v programski opremi. Ta način imenujemo transparentno multiprocesiranje. Sistem 432 je zasnovan tako, da naloga vodenja multiprocesorskih sistemov razpade v tri dele:

- kreiranje razvrščevalnega pravila: določevanje kriterija v kakšnem vrstnem redu si bodo procesi delili procesorje. Obstajajo različni načini: FIFO, ROUND-ROBIN, PRIORITY, DEADLINE, WEIGHTED ROUND-ROBIN, ipd.

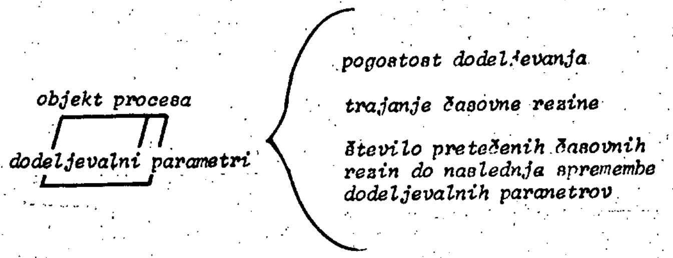
- razvrščanje glede na razvrščevalno pravilo: razvrščanje čakajočih procesov, ki so pripravljeni za izvajanje glede na razvrščevalno pravilo.

- dodeljevanje: dodeljevanje razvrščenih procesov procesorjem v izvrševanje.

In kdo opravlja nalogo vodenja multiprocesorskih sistemov? Morda nadzorni procesor? Ne. Sposobnost vodenja je dana vsem procesorjem v sistemu. Kateri od njih pa v nekem trenutku opravlja funkcijo vodenja, je odvisno od stanja sistema (razjasnitev te problematike bo sledila iz zgleda).

Kreiranje razvrščevalnega pravila

Funkcija kreiranja razvrščevalnega pravila je prepuščena programski opremi - programerju. Zelo pomembno je, pač glede na različne aplikacije, da je možno definirati različne razvrščevalne kriterije. Rezultat kreacije razvrščevalnega pravila je posredovan materialni opremi za razvrščanje in dodeljevanje preko razvrščevalnih parametrov. Slika 5. prikazuje zgled razvrščevalnih parametrov.
Ko je razvrščevalno pravilo določeno, je proces pripravljen za razvrščanje, ki ga bo opravila materialna oprema. Pričetek operacije razvrščanje je sprožen z operacijo SEND v dodeljevalna vrata. Dodeljevalna vrata so objekt, ki je osnova za realizacijo multiprocesorskih sistemov. So prostor, mesto ali del spomina, kjer se "srečujejo" procesi in procesorji.


objekt procesa
dodeljevalni parametri
pogostost dodeljevanja
trajanje časovne rezine
število pretečenih časovnih rezin do naslednje spremembe dodeljevalnih parametrov

Slika 5.

Dodeljevanje

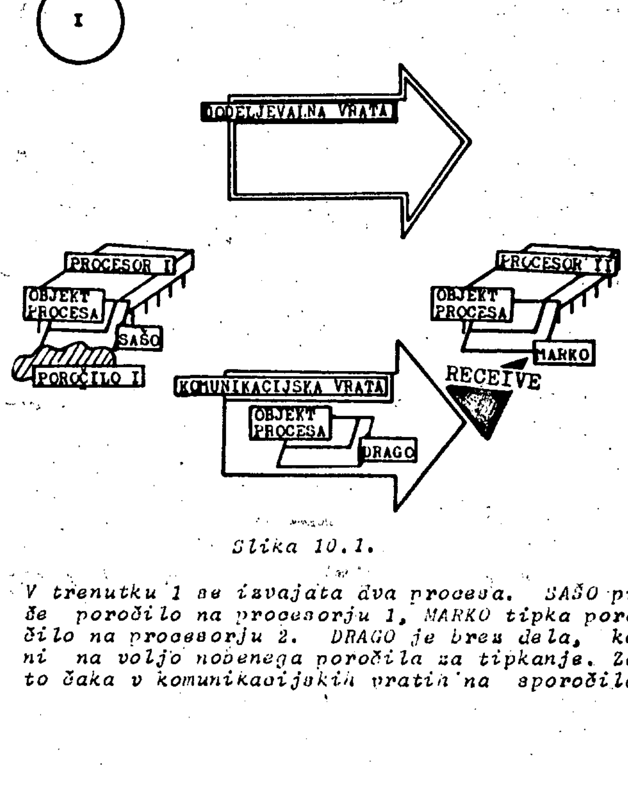
Kot smo omenili so dodeljevalna vrata mesto, kjer se "srečujejo" procesi in procesorji. Procesi čakajo v vrsti (glede na razvrščevalno pravilo) procesor pa "gre" k razvrščevalnim vratom, ko je pripravljen izvršiti nov proces. Če so dodeljevalna vrata prazna, procesor čaka, da se pojavi kak proces.

Zgled za transparentno multiprocesiranje

Na zgledu si bomo ogledali dinamičen potek multiprocesiranja. Za primer bomo vzeli program z imenom REPORTER, ki je sestavljen iz treh procesov. Proces SAŠO, ki piše poročila o prometnih nezgodah in procesa DRAGO in MARKO, ki tipkata poročila in jih dajeta procesu SAŠO v pregled. Na voljo nam bo naslednja struktura:

- trije procesi: SAŠO, DRAGO, MARKO
- dva procesorja: 1 in 2
- ena dodeljevalna vrata
- ena komunikacijska vrata

Trenutek 1 (slika 10.1.)

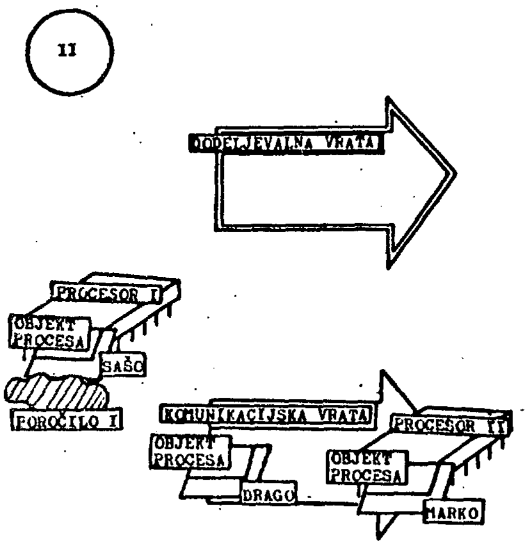


Slika 10.1.

V trenutku 1 se izvajata dva procesa. SAŠO piše poročilo na procesorju 1, MARKO tipka poročilo na procesorju 2. DRAGO je brez dela, ker ni na voljo nobenega poročila za tipkanje. Zato čaka v komunikacijskih vratih na sporočilo.

Marko je s tipkanjem končal in pričel izvajati inštrukcijo RECEIVE, ki naj mu dodeli sporočilo iz komunikacijskih vrat.

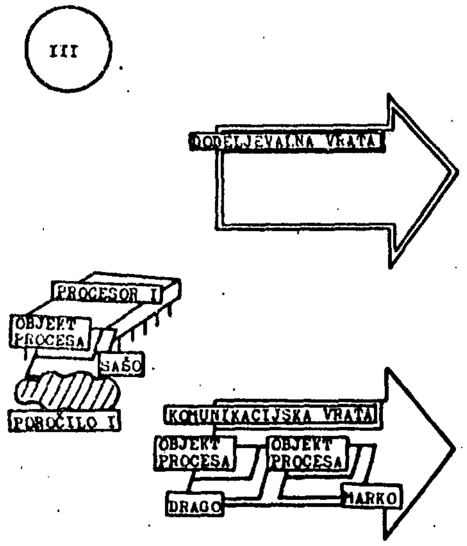
Trenutek 2 (slika 10.2.)



Slika 10.2.

MARKO-va inštrukcija RECEIVE je bila samo delno izvršena, ker v komunikacijskih vratih ni sporočila. Zato mora v čakalno vrsto za DRAGO-m v komunikacijskih vratih. V tem trenutku je seveda procesor 2 končal proces MARKO (MARKO čaka in se ne izvršuje). Procesor 2 se napoti k dodeljevalnim vratom po nov proces.

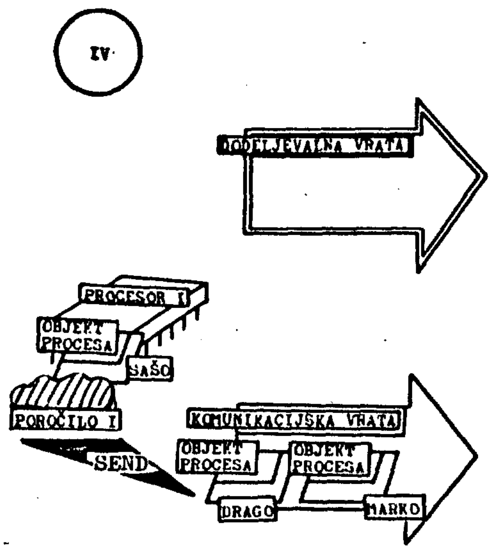
Trenutek 3 (slika 10.3.)



Slika 10.3.

Procesor 2 je pregledal dodeljevalna vrata in ugotovil, da so prazna. Zato vstopi vanje in čaka na nov proces. Proces 1 še vedno izvaja proces SAŠO.

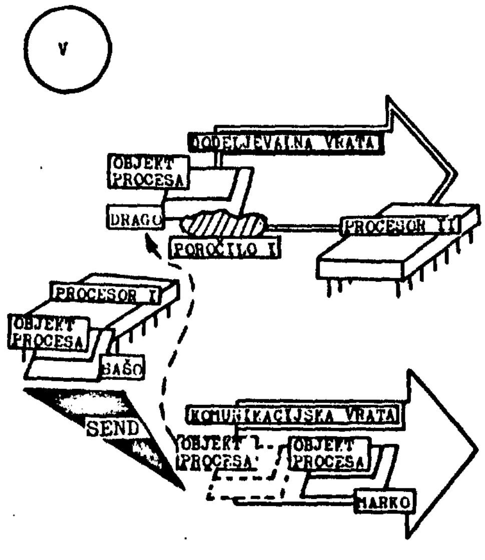
Trenutek 4 (slika 10.4.)



Slika 10.4.

SAŠO je končal pisanje poročila 1 in ga z inštrukcijo SEND pošlje v komunikacijska vrata, kjer čakata MARKO in DRAGO, da bi ga natipkala. Procesor 2 še vedno čaka.

Trenutek 5 (slika 10.5.)

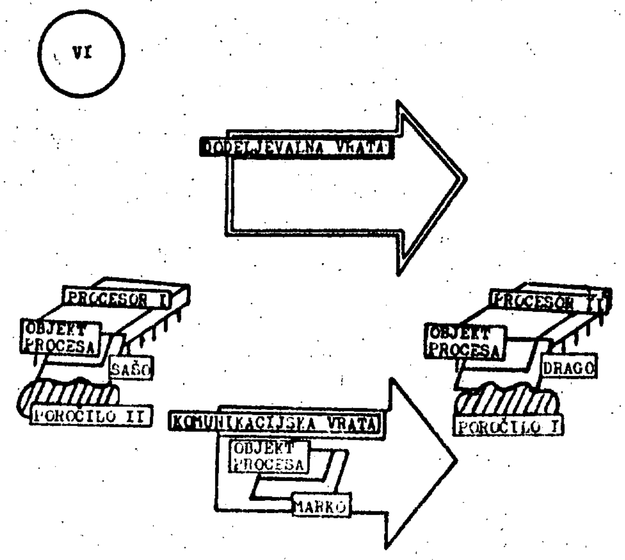


Slika 10.5.

Inštrukcija SEND procesa SAŠO je izvršena samo delno. Sporočilo 1 je dodeljeno procesu DRAGO, ki je sedaj pripravljen za izvajanje. Procesor 1 ga sedaj dodeli dodeljevalnim vratom, kjer pa najde procesor 2, ki čaka. Procesor 1 "ukaže" procesorju 2 naj prične z izvajanjem procesa DRAGO, sam pa se vrne na izvajanje procesa SAŠO.

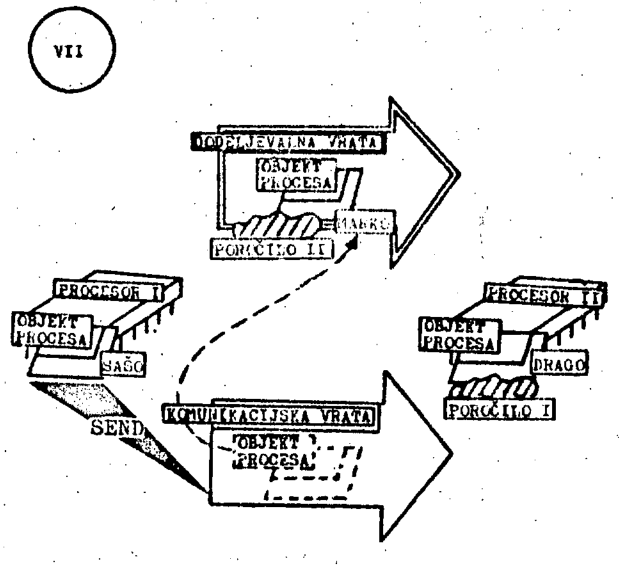
Trenutek 6 (slika 10.6.)

Sedaj delujeta oba procesorja in izvajata procesa SAŠO in DRAGO. Proces MARKO še vedno čaka na sporočilo (poročilo za tipkanje).



Slika 10.6.

Trenutek 7 (slika 10.7.)

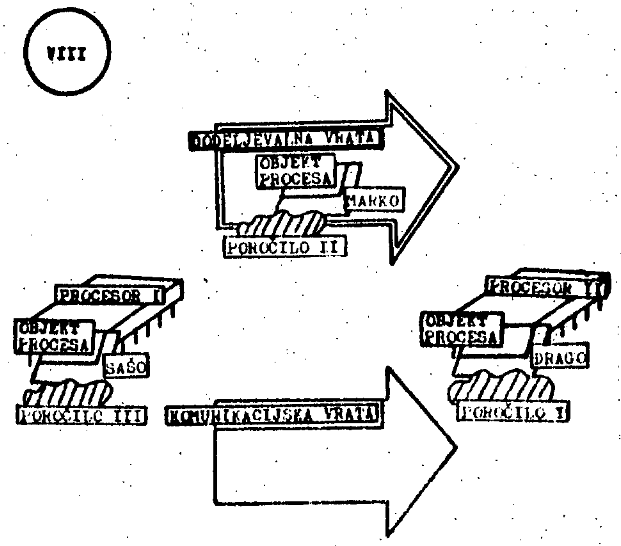


Slika 10.7.

Smo v trenutku, ko proces SAŠO izvaja instrukcijo SEND (sporočilo 2). V komunikacijskih vratih odkrije proces MARKO, ki čaka na katerokoli sporočilo. Procesor 1 mu sporočilo dodeli in prenese proces MARKO s sporočilom vred v dodeljevalna vrata. MARKO je sedaj pripravljen za izvajanje. Vse opisane akcije procesorja 1 so rezultat izvajanje instrukcije SEND.

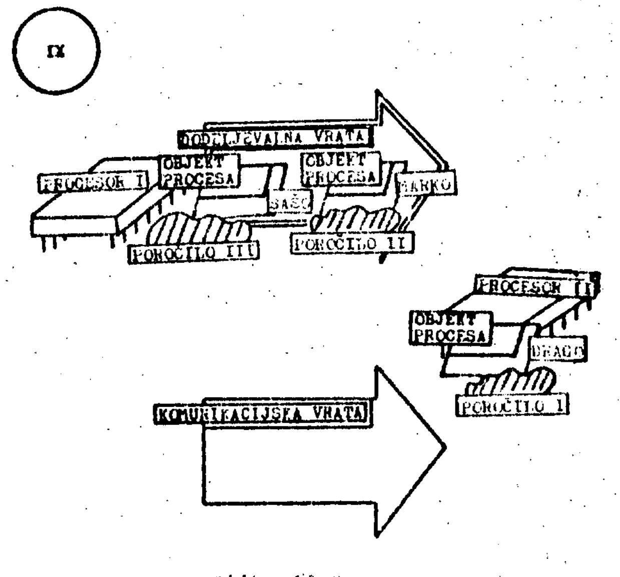
Trenutek 8 (slika 10.8.)

Proces MARKO čaka v dodeljevalnih vratih. Procesor 1 izvršuje proces SAŠO, procesor 2 izvršuje proces DRAGO. Med trenutkoma 6 in 8 je bistvena razlika. V trenutku 6 MARKO še ni pripravljen za izvajanje in zato čaka v komunikacijskih vratih. (Kadarkoli je proces pripravljen za izvajanje, čaka v dodeljevalnih vratih: kadar ni pripravljen, čaka v komunikacijskih vratih na sporočilo procesa s katerim je funkcionalno povezan).



Slika 10.8.

Trenutek 9 (slika 10.9.)

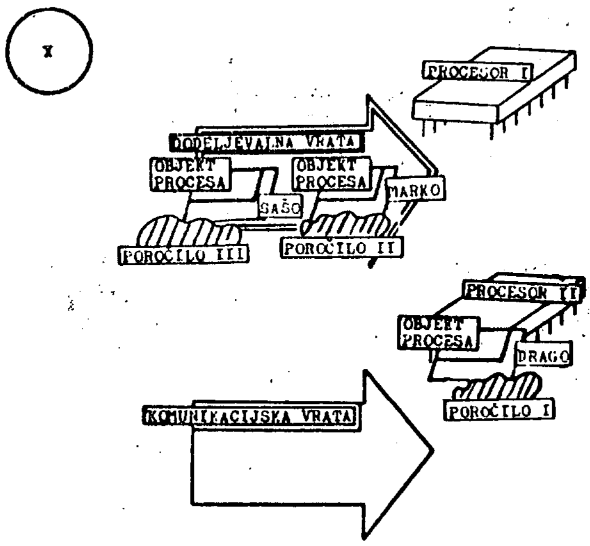


Slika 10.9.

Proces SAŠO je v celoti izkoristil časovno rezino, ki mu je bila dodeljena (glej pojem razvrščevalni parametri). Procesor ugotovi, da mu je čas potekel, konča tekočo instrukcijo in prenese proces SAŠO v dodeljevalna vrata, ker pa ima MARKO višjo prioriteto, je SAŠO na vrsti za njim.

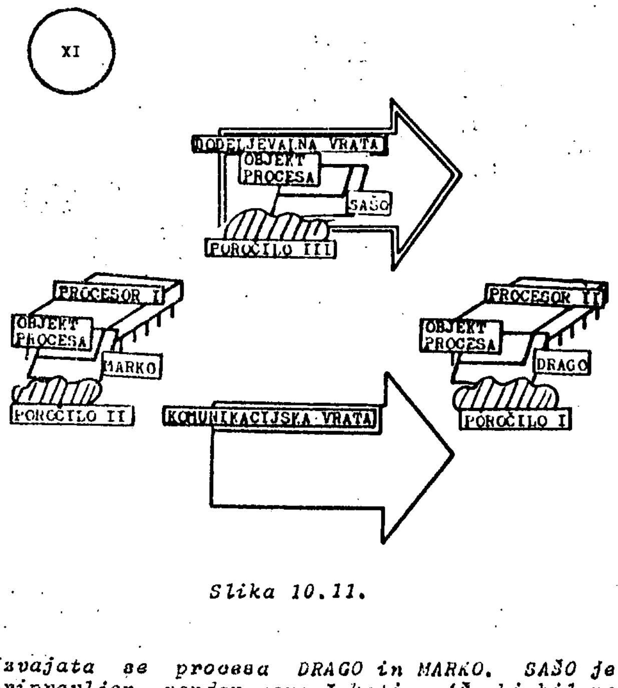
Trenutek 10 (slika 10.10.)

Proces DRAGO se izvršuje na procesorju 2, procesa SAŠO in MARKO čakata na izvrševanje. Procesor 1 je prost, zato vzame iz dodeljevalnih vrat proces MARKO in ga prične izvajati.



Slika 10.10.

Trenutek 11 (slika 10.11.)



Slika 10.11.

Izvajata se procesa DRAGO in MARKO. SAŠO je pripravljen, vendar mora čakati. (Če bi bil na voljo še en procesor, bi se tudi SAŠO pričel izvrševati).

Iz prikazanega zgleda lahko izluščimo pomemben razlog kdaj in zakaj procesor preide iz izvrševanja enega procesa na drug proces:

- kadar se časovna rezina izteče - procesor, ki izvaja proces, izvrši instrukcijo RECEIVE nad praznimi komunikacijskimi vrati

Ob koncu navedimo še odgovore na nekatera osnovna vprašanja, ki bo se morda porodila pozornemu bralcu.

Kako procesor ve, iz katerih dodeljevalnih vrat bo odvzemal procese v obdelavo?

Ena od informacij v objektu procesorja je tudi kazalec na ustrezna dodeljevalna vrata.

Kako procesor ve, v katera dodeljevalna vrata mora vstaviti proces, ki je pripravljen za razvrščanje in čaka v komunikacijskih vratih?

Ena od informacij v objektu procesa je tudi kazalec na ustrezna dodeljevalna vrata.

Koliko dodeljevalnih vrat lahko vsebuje nek sistem?

V principu jih je lahko več. Vendar so samo ena centralna dodeljevalna vrata, v katera vstopajo vsi procesi, ki so pripravljeni za izvajanje in v katera posegajo procesorji, ki iščejo novo opravilo. Pravimo v principu, dejansko stanje pa je naslednje: za vsak tip procesorja obstajajo samo ena dodeljevalna vrata. Arhitektura 432 je namreč takšna, da dopušča tudi sočasno delovanje več različnih procesorjev. Različni procesorji imajo različne nabore instrukcij in zato različne tipe procesov. Nadvse važno je, da izvršujemo nek tip procesa na pripadajočem procesorju. Zato morajo obstajati najmanj ena dodeljevalna vrata za vsak tip procesorja.

V eni od naslednjih številk bodo objavljen opis možnosti sistema 432 dopolnili še z opisom načina gradnje kompleksnih programskih paketov ter prikazali način priključevanja I/O enot.

Ob koncu navedimo še nekaj zanimivosti o sistemu 432. Kot smo že poudarili je le-ta tak, da v popolnosti podpira multiprocesiranje v realnem času. Nekateri osnovni algoritmi so povzetek jeder operacijskih sistemov RMX 80, RMX 88 in še posebaj RMX 86. Za prelitje programske opreme v materialno je bilo potrebno rešiti vrsto tehnoloških problemov. Eden od njih je uvedba štiri vrednostne logike. Celoten razvoj družine 432 (tri integrirana vezja) je zahteval za naše razmere nepredstavljivo velike napore. Samo za I/O procesor (iAPX 43203), ki ga sestavlja 100.00 transistorjev, je bilo potrebno napisati 30 M byt-ov programov za simulacijo in diagnostiko, računalniški čas za izvajanje teh programov pa je znašal 1,3 leta. In kot zadnja zanimivost: ko so izdelali in pretestirali prvi primer I/O procesorja, je ta deloval brez napake.

3. ZAKLJUČEK

V članku smo prikazali samo nekatere detalje in ugodnosti, ki nam jih nudi nova arhitektura. Ni potrebno biti jasnovidec, da ugotoviš, da se bo snovanje novih (mikro) računalniških sistemov popolnoma podredilo novi arhitekturi. Če pogledamo samo svet aplikacij mikro računalnikov, lahko vidimo, da bo možno graditi aplikacije in sisteme, ki so bili do danes izključno domena velikih firm, ali pa so bili procesi, ki bi jih lahko avtomatizirali za poljubnega naročnika, predragi. Operacijski sistemi za delo v realnem času so preliti v materialno opremo. To bo bistveno znižalo ceno sistemov "po naročilu", obenem pa bo pripomoglo k veliko večji zanesljivosti delovanja materialne, še posebaj pa programske opreme. Na tisoče firm po vsem svetu bo gradilo katerekoli računalniške aplikacije za ceno, ki bo povsem sprejemljiva.

Ali lahko sledimo temu razvoju?

To ni odvisno samo od pripravljenosti vodilnih firm v svetu, pač pa predvsem od nas samih.

INFORMATICA 2/1982

PILOT/F – PREPROCESOR ZA RAČUNALNIŠKO PODPRTO PROGRAMIRANO UČENJE

VITOMIR dr. SMOLEJ

UDK: 681.39:371

VISOKA ŠOLA ZA ORGANIZACIJO DELA, KRANJ
UNIVERZA V MARIBORU

Članek opisuje dialekt jezika za programiranje učnih sekvenc za računalniško podprto programirano učenje. Na enostavnem primeru je potem prikazano delovanje preprocesorja. Članek zaključuje ocena opravljenega dela in napotki za nadaljnje delo.

PILOT/F - A PREPROCESSOR FOR A CAI LANGUAGE - A dialect of PILOT is described and its implementation shown. Practical hints for future enhancements and possible additional elements is given at the end.

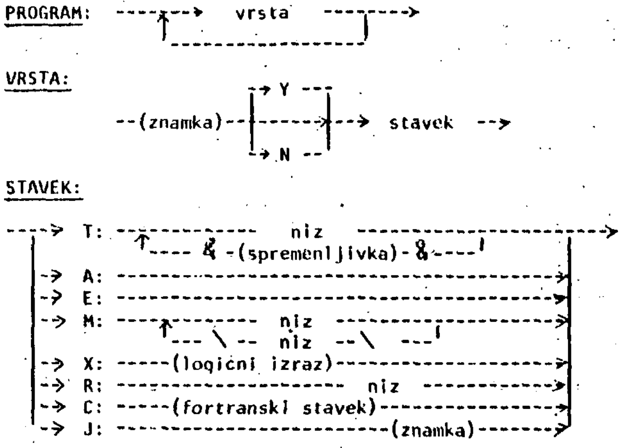
UVOD

Uvajanje računalnika v proces izobraževanja predstavlja eno od možnosti, ki s prodorom tehnologije postaja vse realnejša. Osnovni predpogoj - to je računalnik z ustrezno opremo - je tudi v naših razmerah precej bliže uresničitvi, kot je bilo to pred nekaj leti. Kljub temu pa nekateri elementi celotnega sklopa manjkajo. Predvsem je - in bo še vnaprej -, problem ustrezen kader, ki naj bi učno snov oblikoval na način, ki ustreza novemu mediju.

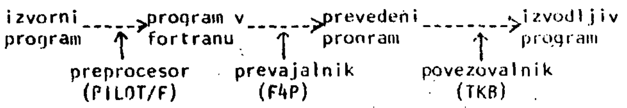
Razkorak med pedagoško usposobljenostjo in pa tehnološkim znanjem, ki je potrebno za delo z računalnikom, skuša predloženi preprocesor prebroditi; z njegovo pomočjo namreč lahko pedagoško usposobljen poznavalec učne snovi izdela učne enote in sekvence, ne da bi moral posebej poznati vse trike in podrobnosti računalnika.

PILOT/F

PILOT (akronim za Programmed Inquiry, Learning and Teaching) predstavlja enega od jezikov, ki so namenjeni računalniško podprtemu pouku. Izbrani dialekt jezika, ki sem ga imenoval PILOT/F (zaradi uporabe fortrana), opisujejo naslednji sintaksni grafi:



Terminalni simboli, ki so v oklepajih, predstavljajo sicer za PILOT/F nize znakov, ki ga ne zanimajo, vendar morajo odgovarjati pravilom fortrana, ker nam PILOT/F, kot vsak preprocesor, izvorni program ne prevaja neposredno v strojno kodo, pač pa v ciljni jezik, ki je v tem primeru fortran. Celotno zaporedje od izvornega programa v PILOT/F do programa, ki ga izvajamo, je tako podano z naslednjo sekvenco operacij:



Čeprav, kot rečeno, preprocesor pravilnosti simbolov v oklepaju ne preverja, morajo odgovarjati pravilom fortrana, v kolikor nočemo imeti težav s prevajalnikom, ki preprocesorju sledi.

Pri povezovanju moramo dodati prevedenemu izvornemu programu (poleg drugega) tudi kodo subrutine MATCH, ki skrbi za primerjanje nizov (gl. dalje).

Ostali simboli v sintaksnih grafih imajo naslednji pomen:

simbol	pomen
Y	izvedi ukaz, če FLAG = .true. (FLAG je rezervirana logična spremenljivka). Primer: Y T: v redu, lahko nadaljujem?
N	izvedi ukaz, če FLAG = .false. Primer: N T: flag ni .true.
A	čitaj stavek s terminala in priredi vsebino nizu ANS (rezerviran niz - ANS kot ANSwer)
M	Primerjaj ANS z nizom, ki sledi, in ustrezno priredi FLAG. Primer: M: v redu\ V REDU\VREDU\ DA\JA\§ V kolikor je ANS enak (znak po znak, vključno ƀ) enemu od nizov za dvopičjem (prvi brez vodečih ƀ ali katerikoli med dvema \ ), je FLAG .true., sicer pa .false. Za najpogostejše odgovore - kot recimo da - je na razpolago meta znak § (ki odgovarja tipki return oz. CR/LF)
X	priredi spremenljivki FLAG logično vrednost izraza, ki sledi. Primer: X: SPOL.EQ.'Z'

T   Izpiši ostanek vrstice in spremenljivke. Primer:

     T: v redu, IME, gremo dalje

R   komentar. Primer:

```
     R:-------------------------------
     R: u č n a   e n o t a   1.23/A
```

J   skoči na znamko, ki je navedena v stavku. Primer:

     N J: 999

C   stavek za dvopičjem preprocesor nespremenljen prenese v ciljni jezik. Primer:

     C: FLAG = FLAG.AND.NAPAK.LT.3

PRIMER

Spodnji programv PILOT/F naj služi za primer. Namen učne enote je preverjanje poznavanja pojma glagol.

```
        R:----------------------------------
        R:  t e s t n i    p r o g r a m
        R:
        C:   T = 0
        T: Poišči glagol v naslednjem stavku
        T: "Pek Peter peče preste po pet para"
1       C:   T = T   1
X:      T.GT.4
        R: dovolili bomo do štiri poizkuse
Y       J:9
        A:
        M:peče\ PEČE\ peči\PEČI
Y       T: pravilno!
Y       J:9
        R:.
        R: ali je bil odgovor samostalnik?
        M:pek PEK\preste\PRESTE\para\PARA
Y       T: ta beseda je samostalnik
Y       J:1
        R:
        R: ali je bil odgovor po?
        M:po  PO
Y       T: ta beseda je predlog
Y       J:1
        A: peter\PETER
        R: če je odgovoril s Petrom, je verjetno
        R: neresen; naj bo tudi odgovor neresen:
Y       T: Petri ti znajo to zameriti
Y       J:1
        R:
        R: samo še pet ostane
        M:pet\PET
Y       T:pet je stevilo (5)
Y       J:1
        R:
        R: verjetno je bila tipkarska napaka
        T: napačno! poizkusiva znova
        T: ( pazi, ko tipkaš !)
        J:1
        R:--------------------------------------
        R:  k o n e c   e n o t e
9       T: glagol je bil "peče"
        T: hvala lepa
        E:
```

Čeprav je program okoren, lahko setsvaljalec učne enote svoje misli zelo hitro izrazi. Primer: v kolikor učencu ni uspelo pravilno prepisati niti ene od besed v stavku, je verjetno začetnik. Zato bi se spodobilo, da delamo z njim v rokavicah, bolj, kot pa smo sprva hoteli. Predvsem je pametno, da mu vprašanje znova postavimo, da mu ne bo treba iskati po ekranu. Zato vrstico, kjer prvič izpišemo tekst, takole spremenimo:

```
99      T: Poišči glagol v naslednjem stavku:
```

(vpeljemo novo labelo 99)

Za tekstom " (pazi, ko tipkaš !) " dodamo še dve vrsti, tako da se na tem mestu končna verzija programa v PILOT/F glasi:

```
        T: napačno! poizkusiva znova
        T: ( pazi, ko tipkaš !)
        T: da ponoviva vprasanje
        J:99
```

Tako pripravljen program v PILOT/F z naslednjim zaporedjem ukazov (na sistemu PDP 11/34 z prevajalnikom F4P in RSX operacijskim sistemom) spravimo v obliko, ki je zrela za izvajanje:

```
>run pilot
 plt > glag=glag
 plt > ^Z
>f4p glag=glag
>tkb glag=glag,match
```

Sedaj si lahko privoščimo naslednji preizkus našega poznavanje pojma glagol.(s crkami elite je vpisan tekst, ki ga izpisuje računalnik):

```
>run glag
 Poišči glagol v naslednjem stavku
 "Pek Peter peče preste po pet para"
 PEK
 ta beseda je samostalnik
 peti
 napačno! poizkusiva znova
 ( pazi, ko tipkaš! )
 da ponoviva vprašanje
 Poišči glagol v naslednjem stavku
 "Pek Peter peče preste po pet para"
 peči
 pravilno!
 glagol je bil "peče"
 hvala lepa
>
```

ZAKLJUČEK

Za konec dve kriticni misli:

Pripravljalec snovi bi mobiti imel lažje delo, ko bi pri rokah imel konstrukte kot so CASE, REPEAT in podobno. Razlog, da tega ni, je najprej PILOT/F sam; dodaten razlog je fortran kot ciljni jezik; ko bi uporabili PASCAL, bi zlahka razširili zalogo ukazov (vprašanje je le, ce bi preprocesor bil tako kratek, kot je sedaj - obsega vsega skupaj okoli 300 vrstic fortranske kode, od tega predstavlja subrutina MATCH eno tretjino).

Drug razlog se skriva v naravi samega učenja;grafi z enim samim vhodom in izhodom predstavljajo le del struktur, s katerimi skušamo slediti procesu učenja in ga posnemati.

Druga kritična misel se tice koncepta preprocesorja: v primeru sistema za več uporabnikov bi bil verjetno tolmač bolj primeren, kar se izrabe sistema tiče.

ZAHVALA

Omenjeno delo je bilo opravljeno v času, ko je bil avtor gost Max-Planck Instituta, Institut für Werkstoffwissenschaften, Stuttgart, Zvezna republika Nemcija.

INFORMATICA 2/1982

SIMULATOR MIKRORAČUNALNIKA 8051

TOMAŽ ERJAVEC

MAJARONOVA 5, 61000 LJUBLJANA

UDK: 519.685.8:681.3

Sestavek opisuje programski simulator za družino zaključenih mikroračunalnikov MCS-51. Najprej sta predstavljeni arhitektura družine MCS-51 in organizacija pomnilnikov. Nato se bralec seznani s skupino programov,ki sestavljajo simulator in z opisom enega instrukcijskega cikla.

SIMULATOR OF THE 8051 MICROCOMPUTER. The article describes a software simulator for the MCS-51 family of single chip microcomputers. First the architecture of the MCS-51 and the organisation of the memories is presented. The reader is then informed with the set of programs that form the simulator and with the description of one instruction cycle.

1. UVOD

Zaključeni mikroračunalniki so zaradi svojih lastnosti primerni za procesno vodenje. V enem samem ohišju (navadno s 40 priključki) združujejo mikroračunalniško strukturo,ki je samozadostna za delovanje. S prihajajočimi zaključenimi mikroračunalniki postaja ta struktura vse bogatejša.

Primerna metoda uvajanja v tehnologijo zaključenih mikroračunalnikov je simulacija, saj lahko na simulacijskem računalniku programsko ponazorimo njihovo delovanje in opazujemo obnašanje. Pri taki simulaciji je seveda težko doseči izvajanje funkcij v realnem času, pač pa je mogoče s primernim vpogledom v notranjost simulatorja nazorno zasledovati dogajanja ob izvajanju instrukcij.

Intel MCS-51 je med najsodobnejšimi družinami zaključenih mikroračunalnikov. Opisani simulacijski programski paket izvaja večino funkcij družine MCS-51, modularna zgradba pa omogoča enostavno dograditev ostalih oziroma izklapljanje že vgrajenih funkcij in s tem prirejanje obsega in hitrosti simulacije uporabniku.

V nadaljnem besedilu se spoznamo z arhitekturo družine MCS-51, modularno zgradbo programskega paketa in opisom poteka simulacije.

2. OPIS DRUŽINE MCS-51

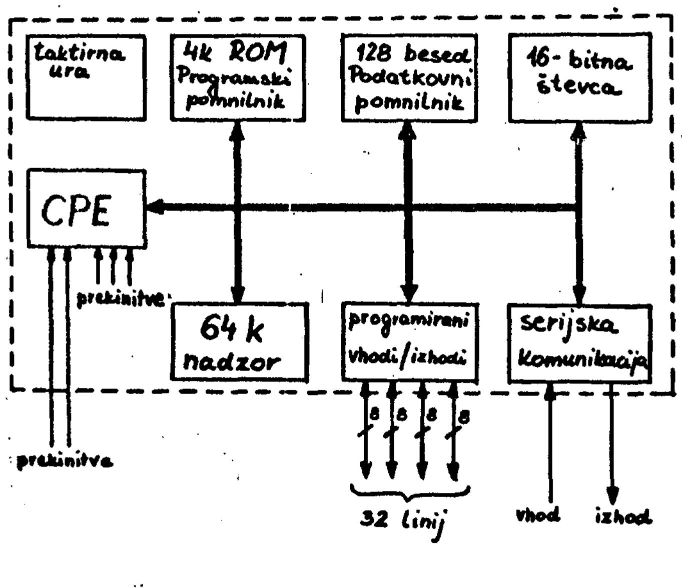
Družina MCS-51 obsega tri zaključene mikroračunalnike, ki se med seboj razlikujejo le po obsegu in vrsti programskega pomnilnika. 8031 je najpreprostejši in nima vgrajenega programskega pomnilnika. 8051 vsebuje 4k ROM, 8751 pa 4k EPROM pomnilnik. Pri vseh treh je naslovljivi prostor programskega pomnilnika omejen s 64k lokacijami. V vseh ostalih lastnostih se člani družine MCS-51 ne razlikujejo, zato se od tu dalje nanje obračamo kar z imenom 8051.

2.1. Arhitektura 8051

Zelo visoka stopnja integracije s 60.000 transistorji na isti silicijevi ploščici je omogočila naslednjo arhitekturo:
- centralna procesna enota
- 4k x 8 programski pomnilnik
- 128 x 8 podatkovni pomnilnik
- 32 vhodno/izhodnih linij
- dva 16-bitna časovnika/števca dogodkov
- 5-izvorna vgnezdena prekinitvena struktura z dvema prednostnima nivojema
- serijska vhodno/izhodna linija
- taktirna ura .

Blokovna shema je podana na sliki 1.

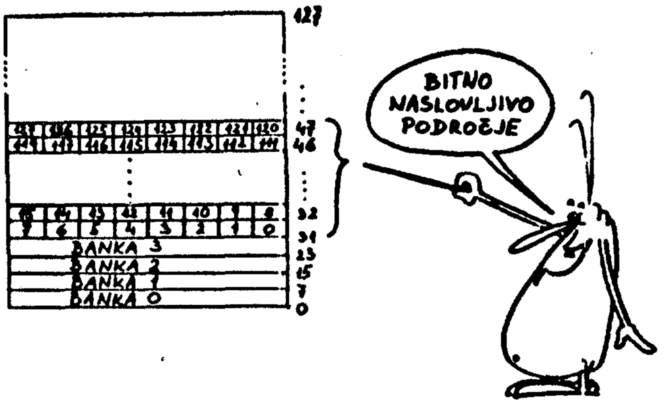
Centralna procesna enota lahko obdeluje podatke iz obeh 64k obsegov programskega in podatkovnega pomnilnika, pri čemer se del obsegov nanaša na vgrajena dela pomnilnikov. Družina je opredeljena kot 8-bitna, izvaja pa nekatere funkcije tudi na 16 bitih, 4 bitih,odlikuje jo še poseben bit-procesor za obdelavo Boolovih spremenljivk. Aritmetični del procesorja izvaja vrsto aritmetičnih in logičnih operacij, posebnost pa sta celoštevilsko množenje in deljenje, ki sicer pri zaključenih mikroračunalnikih nista bila doslej pogosti operaciji.



Slika 1.: Arhitektura 8051

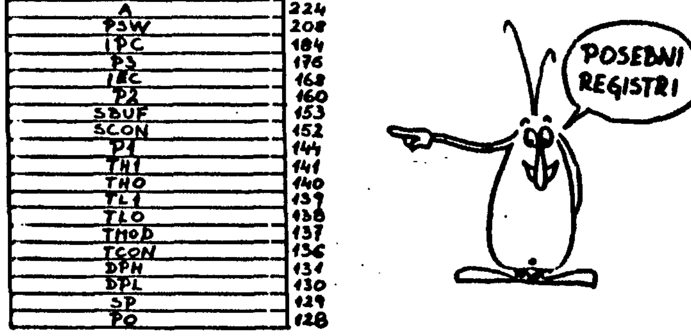
2.2. Organizacija pomnilnikov

Programski števnik (16-bitni) dopušča klice in vejitve po vsem 64k prostoru programskega pomnilnika, nikakor pa ne more pokazati na lokacije podatkovnega pomnilnika, kamor se ne da prenesti izvajanja programa. Ko vrednost v programskem števniku preseže 4095, se naslednja instrukcijska koda prenese preko vhodno/izhodnih vrat iz zunanjega pomnilnika avtomatsko. Določene lokacije programskega pomnilnika so rezervirane za inicializacijo in prekinitvene programe.



Slika 2.: Vgrajeni podatkovni pomnilnik

Vgrajeni podatkovni pomnilnik obsega 128 lokacij. Prvih 32 je zaobseženih v 4 bankah podatkov s po 8 registri. Izbira banke se izvaja s postavljanjem zaznamkov v statusni besedi, izbira registra pa je zajeta že v kodi instrukcije. Na lokacijah od 32 do 47 se nahaja 128 bitov, ki jih je moč naslavljati neposredno in služijo kot pomnilni prostor za bit-procesor. Po 8 bitov skupaj je seveda moč naslavljati tudi besedno kot katerokoli drugo besedo v pomnilniku. Od lokacije 47 naprej so nenamenske lokacije za branjenje poljubnih podatkov.



Slika 3.: Razporeditev posebnih registrov

Poleg tega je mogoče podatke hraniti tudi v posebnih registrih, ki obsegajo 20 lokacij. Tu so akumulator in pomožni akumulator,ki sodeluje pri množenju in deljenju, sicer pa je navaden register, statusna beseda vsebuje štiri aritmetične zaznamke, zaznamka za izbiro banke podatkov in prosti uporabnikov zaznamek. V skupino kazalcev sodijo kazalec sklada in 16-bitni podatkovni kazalec. Poleg štirih 8-bitnih vrat in in dveh 16-bitnih števcev so tu še registri, ki omogočajo programiranje in nadzor delovanja števcev, serijske komunikacije in prekinitvene strukture.

3. ZASNOVA PROGRAMSKEGA PAKETA MCS-51

Programski paket MCS-51 omogoča izvajanje, preizkušanje in nadziranje dogajanja ob izvajanju programov, ki so pisani v strojni kodi 8051, na simulacijskem računalniku.Napisan je v zbirnem jeziku 8080.

Naloge, ki jih opravlja, lahko značajno razdelimo v dve skupini. V statični del sodijo:
- simulacija programskega pomnilnika
- tabeliranje operacijskih kod in naslovov njihovih subrutin
- tabeliranje kod in naslovov izpisnih subrutin.

V dinamični del, katerega spremenljivke se spreminjajo v skladu z izvajanimi instrukcijami, pa sodijo:
- simulacija podatkovnega pomnilnika
- prepoznavanje operacijskih kod
- klicanje instrukcijskih subrutin
- izvajanje funkcij posameznih instrukcij
- postavljanje zaznamkov
- vlaganje naslova naslednje operacijske kode v programski števnik
- izpisovanje vsebine registrov .

Blokovna shema programskega paketa MCS-51 je prikazana na sliki 4.

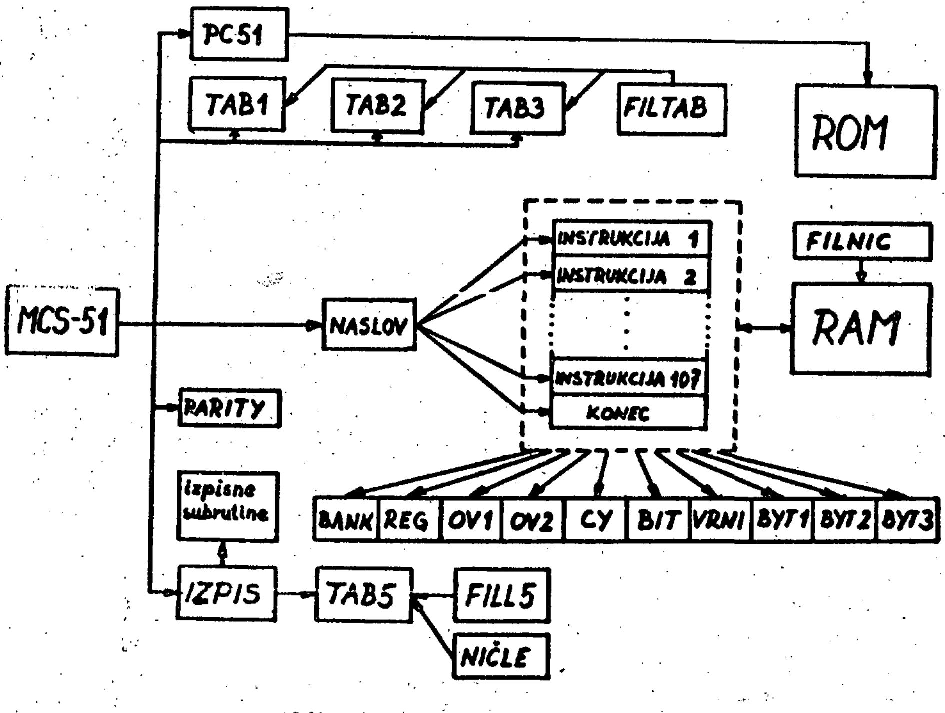
Funkcijski opis blokov

Simulirani programski (ROM) in podatkovni (RAM) pomnilnik sta organizirana prav tako, kot je bilo opisano v poglavju 2.2. Pred izvajanjem simulacije v za programski pomnilnik rezervirano področje pomnilnika simulacijskega računalnika vpišemo program v strojni kodi 8051.

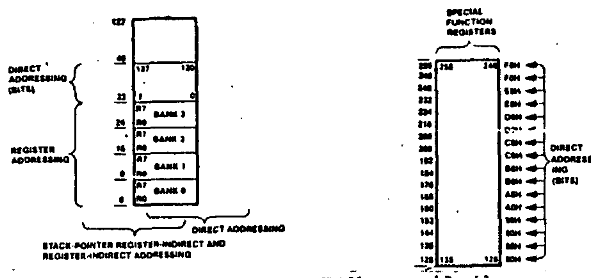
PC51 je simulirani programski števnik in v njem se nahaja naslov v programskem pomnilniku, na katerem je koda instrukcije, ki se bo naslednja izvedla.

TAB1,TAB2,TAB3 so tabele, v katerih se nahajajo podatki v obliki:

koda stran lokacija .



Slika 4.: Shema programskega paketa MCS-51



Slika 5: Od vgrajenega RAM pomnilnika program IZPIS doseže le posebne registre.

Operacijski kodi instrukcije sledita stran in lokacija, na kateri se začenja subrutina, ki izvede operacijo instrukcije.Operacijske kode 8051 je moč po njihovih karakterističnih lastnostih razdeliti v štiri skupine:
- aaak0001
- kkkklrrr
- kkkkklli
- kkkkkkkk

kjer pomeni "a" naslov, "r" register v določeni banki podatkov, "i" register v banki podatkov pri indirektnem naslavljanju in "k" samo operacijsko kodo. Kodi prvega tipa sta samo dve in ju prepoznava glavni program neposredno, ostale pa so razdeljene v tri tabele zaradi krajšega iskanja in razpoznavanja.

FILTAB pred začetkom izvajanja simulacije vpiše v pomnilnik simulacijskega računalnika tabele TAB1,TAB2 in TAB3.

FILL5 pred začetkom simulacije zapiše v pomnilnik simulacijskega računalnika tabelo TAB5, ki služi prepoznavanju imen posebnih registrov, katerih vsebine želimo izpisati v podprogramu IZPIS. Tabelo vpisuje v obliki:
znak znak znak znak stran lokacija.
Če ima ime registra manj kot 4 znake, je namesto njih vpisana 0. Stran in lokacija se nanašata na naslov subrutine, ki izpiše vsebino željenega registra.

NIČLE pred začetkom izvajanja simulacije zapiše v pomnilnik simulacijskega računalnika ničle na področje tabele TAB5.

FILNIČ pred začetkom izvajanja simulacije na področje simuliranega podatkovnega pomnilnika zapiše ničle.

Dinamični del pa tvorijo:

MCS-51 je glavni program in nadzira delovanje vseh ostalih. Pri zagonu nastavi programski števnik PC51 na začetek programskega pomnilnika ter vpraša po načinu izpisovanja vrednosti registrov med izvajanjem. Nato se izvajanje ponavlja v zanki instrukcijskega cikla, dokler ni razpoznana koda 360 (dodatno vstavljena koda za prenehanje simuliranja). Podrobnejši opis delovanja je v poglavju 4.

NASLOV : Ko MCS-51 razpozna operacijsko kodo v eni od tabel, vloži naslov instrukcijske subrutine v NASLOV, ki jo pokliče takoj za tem, ko je naslov vrnitve v glavni program založil na sklad.

IZPIS si zapomni način izpisovanja, ki je bil vstavljen ob zagonu glavnega programa. Pozna tri načine delovanja.
Način 0 : Želimo izpisovanje poljubnih posebnih registrov. Najprej se izpiše vrednost programskega števnika PC51 in koda izvedene instrukcije. Nato IZPIS čaka, da vtipkamo ime registra. Prebrani podatek primerja z vrednostmi v tabeli TAB5 in po uspešni primerjavi pokliče izpis vsebine željenega registra, sicer javi napako operaterja. Po opravljenem izpisu vsebine registra čaka na novo ime registra. Izpisovalno sekvenco prekinemo z 'return'
Način 1 : Želimo izpisovanje vsebin osmih pomembnejših registrov (A, B, PSW, SP, DPH, DPL, PO, P1) po vsaki izvedeni instrukciji. Hkrati se izpisuje tudi stanje programskega števnika in koda izvedene instrukcije. Na vsakih 20 vrstic številčnih izpisov (na vsakih 20 instrukcij) se ponovi vrstica z imeni registrov, ki jim pripada številčni stolpec pod njimi.
Način 2 : Ne želimo izpisa na terminal. To pospeši simulacijo, a prepreči vpogled v dogajanje v mikroračunalniku.

IZPISNE SUBRUTINE imajo nalogo izpisati vsebino registra, ki ga prepozna IZPIS.

INSTRUKCIJSKE SUBRUTINE : obsegajo 107 subrutin, ki opravljajo funkcije 107 operacijskih kod. Instrukcije se delijo v štiri funkcijske skupine : aritmetične, logične, za prenos podatkov in za prenos nadzora. Dodana jim je še funkcija KONEC.

KONEC v pravem 8051 ne obstaja, tu pa je uporabljen za zaključitev simulacije. Ko glavni program pokliče KONEC, ta zahteva način končnega izpisa vsebin registrov in po njem zapiše, da je izvajanje simulacije končano.

KONEC se odziva na kodo 360 (oktalno).

Instrukcijske subrutine pri svojem delovanju lahko pokličejo 10 podprogramov, ki jih potrebujejo za določitev izvornih in ponornih operandov, ki sodelujejo v instrukciji, za računanje zaznamkov v statusni besedi in za ustrezno inkrementiranje programskega števnika PC51.

BANK kličejo instrukcijske subrutine, katerih operandi so registri v eni izmed štirih bank podatkov. BANK prebere vsebino tretjega in četrtega bita v statusni besedi, ki določujeta, v kateri od bank podatkov se nahaja operand. Nato nastavi kazalec na eno od štirih bank podatkov.

REG dopolnjuje BANK, saj iz vrednosti prostih bitov v kodi instrukcije razbere, kateri register v pokazani banki podatkov je operand instrukcije.

OV1 kličejo instrukcijske subrutine za seštevanje. Njegova funkcija je ugotoviti, če je pri seštevanju dveh operandov prišlo do prenosa iz šestega in sedmega bita rezultata. Nato postavi tretji bit statusne besede (zaznamek 'preliv') na 1, če je vsota po modulu 2 prenosov iz šestega in sedmega bita rezultata enaka 1, sicer postane 'preliv ' enak 0.

OV2 opravlja podobno funkcijo kot OV1, le da ga kličejo instrukcijske subrutine za odštevanje s sposojanjem.

CY kličejo vse aritmetične instrukcijske subrutine. Njegova funkcija je ugotoviti, če je pri izvršeni aritmetični operaciji prišlo do prenosa iz tretjega in sedmega bita. V skladu z ugotovljenim postavi zaznamek 'pomožni prenos'oziroma 'prenos'na 6. oziroma 7. mestu statusne besede na 1, če je prišlo do prenosa, sicer na 0.

BIT služi za določanje naslovov operandov pri delovanju instrukcij bit-procesorja.Na - slavljanje operandov bit-procesorja je direktno, torej je naslovljen bit neposredno brez imenovanja besede, v kateri se nahaja. Z uporabo rotacij, prištevanja konstant in maskiranja besed omogoča BIT simulacijo takšnega naslavljanja bitnih operandov.

VRNI opravlja inverzno funkcijo BIT-a. Bitno spremenljivko, ki smo jo dobili kot rezultat bitnih operacij, vrne na ponorno mesto v podatkovnem pomnilniku. Bit zarotira na pravo mesto v besedi, nato maskira besedo v pomnilniku in zbriše bitno lokacijo, na katero pride ponorni operand. Besedo z zarotiranim bitom nato prišteje besedi v pomnilniku. (V prištevani besedi so vsi biti razen ponornega enaki 0).

BYT1, BYT2, BYT3 služijo nastavljanju programskega števnika PC51 na naslov naslednje operacijske kode. Kličejo jih instrukcijske subrutine v skladu s številom besed, ki jih obsegajo njihove operacijske kode (ena, dve ali tri).

PARITY izračunava pariteto podatka v akumulatorju. Kliče ga glavni program MCS-51 po vsaki opravljeni instrukciji in pred izpisom rezultatov le-te. Njegova funkcija je seštevanje po modulu 2 vseh bitov v akumulatorju.Če je vsota 1, postane zaznamek 'pariteta'v statusni besedi enak 1, sicer 0.

4. OPIS INSTRUKCIJSKEGA CIKLA

PC51 kaže na lokacijo v programskem pomnilniku, na kateri se nahaja koda instrukcije. MCS-51 prebere kodo, ugotovi njeno karakteristično lastnost ter izbere eno od treh tabel, iz katere bo črpal podatke za primerjavo. Ko je koda prepoznana, se naslov njene instrukcijske subrutine vloži v NASLOV, ki pokliče subrutino. Ta v skladu s številom besed svoje operacijske kode pokliče BYT1, BYT2 ali BYT3,ki inkrementira programski števnik PC51, da kaže na lokacijo naslednje operacijske kode. Če operandi izvajane operacijske kode niso takojšnji ali direktno naslovljeni, instrukcijska subrutina pokliče BANK in REG in z njuno pomočjo določa registre iz bank podatkov ali pa indirektno naslovljene operande. Ce je instrukcija aritmetična, lahko pokliče OV1, OV2 in CY ter z njihovo pomočjo izračuna zaznamke 'preliv'in 'prenos'. Ce sodi instrukcija v bit-procesor, izračuna naslov operandov s klicem podprograma BIT, ko pa izračuna rezultat, ga vrne kot ponorni operand s klicem VRNI na njegovo mesto v podatkovnem pomnilniku. Ob zaključku instrukcijske subrutine se izvajanje vrne v MCS-51. Ob tem času so spremembe, ki jih je povzročila instrukcija že vpisane v podatkovnem pomnilniku. Glavni program pokliče PARITY, ki izračuna pariteto podatka, ki se nahaja v akumulatorju in vnese ustrezni zaznamek v statusno besedo. Zadnji klic v glavnem programu je klic izpisnega podprograma IZPIS, ki v skladu s predhodno vloženim podatkom o načinu izpisovanja vsebin registrov izpiše željene vrednosti. S tem je cikel ene instrukcije končan, v PC51 pa že čaka naslov naslednje operacijske kode.

5. SKLEP

Simulacijski programski paket je zanimiv kot učni pripomoček. Z njim lahko podrobno sledimo dogajanju v mikroračunalniku. V kombinaciji s prevajalnikom za zbirni jezik 8051 lahko predstavlja učinkovit učni pripomoček za učenje uporabe zbirnega jezika, ker lahko zasledujemo neposredni učinek izvajanja instrukcij.

Pri simulaciji posebnih nalog,kjer ne uporabljamo celotnega nabora ukazov, lahko poljuben obseg instrukcij povsem izklopimo, s tem skrajšamo čas dekodiranja in pospešimo delovanje simulatorja.

Čeprav vse funkcije 8051 še niso vgrajene v simulator, je bilo dograjevanje predvideno in ga modularna zgradba omogoča. To velja predvsem v primeru dograjevanja v emulator, ki ga sestavlja simulacijski programski paket in digitalne vhodno/izhodne enote, priključene na simulacijski računalnik. Na tak način je mogoče prikazati celotno delovanje zaključenega mikroračunalnika vključno s priključnimi sponkami, seveda ne v realnem času.

LITERATURA

1. Intel Corporation, MCS-51 Family of Single Chip Microcomputers - User's Manual, Intel Corporation, Santa Clara California, 1981
2. Adam Osborne, An Introduction to Microcomputers, volume 1, Osborne & Associates, Inc., Berkeley, California, 1978
3. Intel Software Subroutines,fotokopije, last IJS/El

STRUKTURIRANI ZBIRNIK ZA MIKROPROCESOR 68000

ROK SOSIČ, ALOJZ HODOBIVNIK
RUČIGAJEVA 14, 64000 KRANJ
SUHA 1/a, 64000 KRANJ

UDK: 681.3.068

Članek opisuje zbirnik za mikroprocesor 68000 (M68000) in predprocesor, ki omogoča strukturirano pisanje zbirniških programov. Zbirnik je bil napisan kot pripomoček za razvoj računalnika na osnovi M 68000 v ISKRI TOZD Računalniki Kranj.

Structured assembler for microprocessor 68000: This describes assembler for microprocessor 68000 and preprocessor for structured syntax. Both of them were developed in ISKRA TOZD Računalniki Kranj.

A. PREČNI MAKRO ZBIRNIK ZA M68000

1. UVOD

Ko so se pojavili zmogljivi 16-bitni mikroprocesorji, se je ISKRA odločila za razvoj računalnika na osnovi M68000. Kot osnova za razvoj novega sistema je bil napisan križni zbirnik, kasneje pa je bil dodan še predprocesor za strukturirani zbirnik. Oba programa tečeta na računalniku ISKRADATA ID-19, napisana pa sta v jeziku FORTRAN.

2. KRATEK OPIS ZBIRNIKA ZA M68000

Sam mikroprocesor je bil opisan v (1), zato tega ne bomo ponavljali. Zbirnik za M68000 je podoben ostalim zbirnikom za mikroprocesorje. Ima le precej več načinov adresiranja - kar 14. Čeprav je osnovnih instrukcijskih mnemonikov le 56, pa lahko z izbiro načina adresiranja dobimo več kot 1000 različnih instrukcij.

Ker je M68000 precej zmogljiv, ima temu primerno zapleteno zgradbo instrukcij. Kot primer naj povem, da zaseda dekodiranje instrukcije pri zbirniku za M68000 okrog 260 besed, pri zbirniku za M68000 pa kar 3500 besed.
Oglejmo si zgradbo :
Dolžina osnovne instrukcije je 16 bitov, posamezni biti pa pomenijo:
15 - 8 - koda instrukcije
7 - 6 - dolžina operanda:
00 - zlog (8 bitov)
01 - beseda (16 bitov)
10 - dolga beseda (32 bitov)
5 - 3 - način adresiranja
2 - 0 - številka registra

Poleg tega ima zbirnik še polno posebnih oblik instrukcij, tako da je zgornja instrukcija skoraj izjema in ne pravilo.

2.1 NAČINI ADRESIRANJA:

Oznake:
EA - adresa operanda An - naslovni register
Dn - podatkovni reg.
Xn - indeksni register (An ali Dn) SR-statusni reg.
PC - programski števec d8 - 8 - bitni odmik
d16 - 16 - bitni odmik N - 1 za zlog, 2 za besedo, 4 za dolgo besedo
( ) - vsebina = - nadomesti

1. Direktno adresiranje podatkovnega registra:
EA = Dn
način : 000

2. Direktno adresiranje naslovnega registra:
EA = An
način : 001

3. Indirektno adresiranje:
EA = (An)
način: 010

4. Indirektno adresiranje s povečanjem:
EA = (An) , An = An + N
način: 011
sintaksa: (An)+

5. Indirektno adresiranje z zmanjšanjem:
An = An - N, EA = (An)
način : 100
sintaksa: - (An)

6. Indirektno adresiranje z odmikom:
EA = (An) + d16
način : 101
sintaksa: d(An)
Instrukciji je dodana še 16-bitna beseda.

7. Indirektno adresiranje z odmikom in indeksnim registrom:
EA = (An) + (Xn) + d8
način : 110
sintaksa : d(An,Xn)
Instrukciji je dodana še 16-bitna beseda s sledečo zgradbo:
bit 15
0 - indeksni register je Dn
1 - indeksni register je An
biti 14-12
številka indeksnega registra
bit 11:
0 - indeks je samo spodnjih 16 bitov v registru
1 - indeks je 32 biten
biti 7 - 0:
odmik

8. Absolutno kratko adresiranje:
EA = naslednja beseda
način: 111
številka registra: 000
sintaksa : nnn
Instrukciji je dodana 16 bitna beseda, v kateri je naslov operanda.

9. Absolutno dolgo adresiranje:
EA 1 naslednji dve besedi
način: 111
številka registra: 001
sintaksa:nnnnnnn
Instrukciji sta dodani dve besedi, v katerih je naslov operanda.

10. Relativno adresiranje z odmikom:
EA = (PC) + d16
način : 111
številka registra : 010
sintaksa : nnn - relativno na PC
Instrukciji je dodana 16 bitna beseda, v kateri je odmik.

11. Relativno adresiranje z odmikom in indeksnim registrom:
EA = (PC) + (Xn) + d8
način: 111
register: 011
sintaksa: nnn(Xn) - relativno na PC
Instrukciji je dodana 16-bitna beseda z naslednjo zgradbo:
bit 15 :
0 - indeksni register Dn
1 - indeksni register je An
biti 14-12:
številka indeksnega registra
bit 11:
0 - indeks je samo spodnjih 16 bitov v registru
1 - indeks je cel register
biti 7 - 0 :
odmik

12. Takojšnji način adresiranja9
EA = v naslednjih besedah
način : 111
številka registra: 100
sintaksa : nnn
Instrukciji je dodan operand. Ta je lahko :
- spodnjih 8 bitov naslednje besede
- cela naslednja beseda (16 bitov)
- dve naslednji besedi (32 bitov)

13. Takojšni hitri način :
Ta način imata le instrukciji ADD in SUB. Operand je vsebovan že v kodi instrukcije. Odštevamo ali prištevamo lahko največ 3 bitno število.

14. Adresiranje statusnega registra:
EA = SR
način : 111
številka registra: 100
sintaksa : SR

2.2 DOLOČANJE DOLŽINE OPERANDA

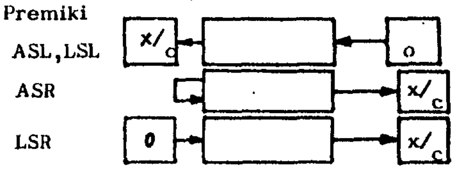
V splošnem so lahko operandi dolgi 8, 16 ali 32 bitov. Poleg tega imamo še možnost operirati z BCD kodo (4 biti) in z biti. Dolžino operanda določimo tako, da z mnemonikom instrukcije napišemo .B , .W ali .L.

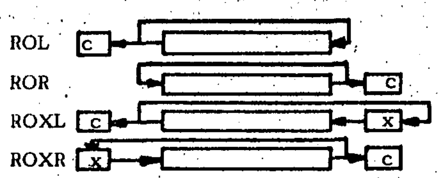
Primer:
ADD.B DO,D1 (prišteje spodnjih 8 bitov iz D0 v D1)
ADD.W D0,D1 (prišteje spodnjih 16 bitov iz D0 v D1)
ADD.L D0,D1 (prišteje celoten D0 v D1)
Operacije nad BCD kodo in biti pa so določene že s samo instrukcijo.

2.3 Vrste instrukcij

Pri dvooperandskih instrukcijah je le en naslov lahko v spominu, izjema je MOVE, kjer sta lahko oba.

1. Premikanje podatkov
MOVE - prenos iz ene lokacije v drugo
MOVEM - prenos vsebine več registrov naenkrat
MOVEP - prenos po bytih
EXG - zamenjaj vsebino dveh registrov
LINK - poveži
UNLK - razveži
(zadnji dve instrukciji uporabljamo za prenašanje parametrov in vzdrževanje list)
2. Celoštevilčna aritmetike
ADD - seštej
CLR - briši
CMP - primerjaj
DIVS - deli s predznakom
DIVU - deli brez predznaka
EXT - razširi operand na 32 bitov
MULS - množi s predznakom
MULU - množi brez predznaka
NEG - negiraj
SUB - odštej
TAS - testiraj in postavi
TST - testiraj
3. Logične operacije
AND - logični in
EOR - ekskluzivni ali
OR - logični ali
NOT - eniški komplement
4. Premiki





5. Operacije z biti
   BCLR - briši bit
   BCHG - spremeni bit
   BSET - postavi bit
   BTST - testiraj bit

6. Operacije z BCD kodo:
   ABCD - seštej
   NBCD - negiraj
   SBCD - odštej

7. Kontrola programa:
   BCC - pogojni skoki (CC - glej CON CODE v opisu strukturiranega zbirnika)
   DBcc - tvorba kratkih zank
   Scc - pogojno postavi
   BRA - relativen skok
   JSR - absoluten skok v podprogramu
   RTR - vrnitev iz podprograma in obnovitev statusnega registra
   RTS - vrnitev iz podprograma
   NOP - prazna instrukcija

8. Priviligirane instrukcije
   RESET - začetek
   RTE - vrnitev iz prekinitve
   STOP - stoj
   in instrukcije za popravljanje statusnega registra

9. Pasti
   TRAP - začetek pasti
   TRAPV - testiranje prekoračitve
   CHK - testiranje mej operanda

2.4 Pogojno zbiranje

Pogojno zbiranje začnemo z enim izmed naslednjih pogojev: IFEQ, IFNE, IFGT, IFGE, IFL, IFLE. Pogoj zaključimo z ENDC.
Vsi pogoji so testirani na 0.
Primer:

```
IFEQ A
ADD DO, D1
ENDC
```

Gnezdenje pogojnega zbiranja je neomejeno.

2.5 Makro zbiranje:

Makro definiramo z ukazom MACRO in končamo z ukazom ENDM.
Kličemo ga po imenu in za njim navedemo listo parametrov. Zamenjavo parametra označimo z znakom Ð (@) in za njim številko parametra (0 - 99). Lokalno labelo zahtevamo z znakom Ð in za njim črko, ki označuje eno izmed lokalnih label (A - Z). Globina makro klicev je omejena na 25 (zaradi sklada lokalnih label). Parametri so ločeni z vejicami, če pa parameter vsebuje vejico, ga lahko damo v oglati oklepaj .
Primeri definicije:

```
ZAM    MACRO
        BEQ      ÐA
        ADD      Ð1
        ADD      Ð2
ÐA      NOP
        ENDM
```

klic:

```
ZAM<#5, D1>,<D1,D2>
```

razširitev:

```
BEQ  .00001
ADD #5,D1
ADD  D1,D2
.00001  NOP
```

3. OPIS PROGRAMA - ZBRINIK ZA M68000

Program je napisan po priročniku /3/, tako da je kompatibilen z originalnim zbirnikom. Delno spremenjeno je le pogojno in makro zbiranje.
Pri pogojnem zbiranju so dodani še 4 pogoji - poleg IFEQ in IFNE - še IFGT, IFLT, IFGE in IFLE.
Pri makro zbiranju pa je število možnih parametrov razširjeno z 9 na 99 in število lokalnih oznak z 1 na 26, globina gnezdenja pa povečana na 25.

Zanimiva je tudi organizacija programa. Za razliko od večine zbirnikov, kjer je vsa teža na drugem prehodu, večino dela opravi prvi prehod. Tak pristop je bilo potrebno uporabiti zaradi nefleksibilnosti dela z datotekami na računalniku ID-19. Tako prvi prehod že generira listing in vanj vpisuje oznake za drugi prehod. Ker ta pregleduje le tiste vrstice, kjer je oznaka, je tudi zelo hiter - približno 50 krat hitrejši od prvega prehoda, katerega hitrost znaša 500 - 2000 vrstic na minuto.
Program generira na koncu listinga še listo napak z diagnozo in referenčno tabelo.

B. PREDPROCESOR ZA STRUKTURIRANI ZBIRNIK

1. UVOD

Da bi izboljšali preglednost in hitrost pisanja učinkovitih programov v zbirniku, je bil kot implementacija zbirnika za M68000 izdelan križni predprocesor za strukturirani zbirnik, ki se imenuje SASS (Structured ASSembler).
SASS ima za vhod program v strukturiranem zbirniku v editorski datoteki, za izhod pa dobimo navadni zbirnik.
SASS je trenutno še ločen od zbirnika, vendar sta strogo kompatibilna.

2.0 PREGLED KONTROLNIH STAVKOV STRUKTURIRANEGA ZBIRNIKA

Za predstavitev bomo uporabili BNF-zapis.

```
<IF STAVEK>: : = IF [.<size code>] <boolean exp>
                 THEN[.<extent>]<stavek>[ELSE [.<extent>]
                 <stavek>]
<WHILE-STAVEK>: : = WHILE[.<size code>]<boolean exp>
                    DO [.<extent>]<stavek>
<REPEAT-STAVEK>: : = REPEAT[.<extent>]<stavek>
                     UNTIL[.<size code>]<boolean exp>
<FOR - STAVEK>: : = FOR[.<size code>](<A reg>|<D reg>)=
                  <operand>TO|DOWNTO<operand>
           [BY <abs exp> DO[.<extent>]<stavek>
```

op.: z BY definiramo korak zanke, če ga izpustimo je 1

```
<CALL - STAVEK>: : = CALL<ime>
<SESTAVLJEN STAVEK>: := BEGIN{<stavek>}END
<DEFINICIJA PROCEDURE>: :=PROCEDURE<ime><se stav-
                                   ljen stavek>
```

2.1 Opis izrazov, ki pojasnujejo definicije

```
<STAVEK>: : =<zbir. direktiva>
             <makro klic>
             <68000 - instrukcija>}glej opis v zbirniku
             <if- stavek><while -
              stavek><repeat - stavek>
             <for-stavek><call-stavek><sestavljen stavek>
             <definicija-procedure>

<IME>: :  alfanum{alfanum}
<A reg>: := A0|...|A7
<D reg>: := D0|...|D7
<size code>: : = B|W|L
<extent>: : = S|W
<BOOLEAN EXP>: : =<cmp exp>|<con code>
<CMP EXP>: : =<operand><can code><operand>
<CON CODE>: :=<LQ|NE|PL|MI|GT|LT|GE|HI|LS|CS| CC|
               VS|VC)>
<OPERAND>: : = operand združuje vse adresne načine
               in mora ustrezati instrukciji, kateri
               pripada:
               operand iz<cmp exp>pripada instrukciji
               CMP
               operand iz<for-stavka>pripada CMP in
               MOVE
<ABS EXP>: : = vsi načini, ki se lahko pojavijo na me-
               stu xxx v zbirniških stavkih:
                     SUB xxx,Rn
                     ADD xxx,Rn
               to je lahko dolg sestavljen aritmetični
               izraz
```

Program zaključuje END stavek. Programiramo tako, da napišemo mešanico strukturiranih kontrolnih stavkov in zbirniške stavke. Strogo rezervirane besede, ki se ne smejo pojaviti nitt v navadnem zbirniku M68000 so:
IF,ELS,REPEAT,UNTIL,WHILE,FOR,BEGIN,END,CALL, PROCEDURE

3.0 OPIS KOMENTARJEV

Za uporabnika, ki bo naletel na program v strukturiranem zbirniku, bodo verjetno zanimive pojavne oblike komentarjev:

1. Komentar, ki se prične v 1.koloni vrstice se razteza preko celotne vrstice 72 znakov (ekvivalentno komentarju v FORTRAN-u : C.....) Ta komentar se prepiše tudi v prevod.
2. Zbirniški stavek ima v vrstici sledečo strukturo:
   [<naslov>]<operator><operand-><komentar>]
   V kolikor se pojavi naslov stavka, se mora pričeti v prvi koloni, kar velja tudi za kontrolne stavke. Komentar je celotno polje od operanda do znaka; ali konca vrstice (72 kolone). Komentar ni napovedan z nobenim posebnim znakom in se prepiše v prevod.
3. Komentar, ki se pojavi med meta besedami in deli kontrolnih stavkov, mora biti napovedan z! in se razteza do konca vrstice. Ta komentar je viden samo v listingu.

```
PROCEDURE! definiramo podprogram
BRISI     ! to je ime podprograma
* slediti mora sestavljen stavek
 BEGIN
CLR D1 brišemo register D1; CLR D2
* v nadaljevanju se pojavi drug stavek
* sledi zaključek podprograma
 END
```

4.0 LASTNOSTI PREDPROCESORJA SASS

Oglejmo si nekaj lastnosti in omejitev, ki so značilni za strukturirani zbirnik M68000 in procesor SASS:

- prevajanje strukturiranega zbirnika se izvede ob enem samem prehodu
- pravilnost zbirniških stavkov testira zbirnik, medtem ko SASS kontrolira samo sintakso in semantiko kontrolnih stavkov
- več stavkov je lahko zapisano v eno vrstico, če so med seboj ločeni z znakom; .Sicer so zbirniški stavki omejeni na eno vrstico in zaključeni s koncem vrstice. Elementi kontrolnega stavka se lahko raztezajo preko več vrstic ob pogoju, da ne delimo besed ali izrazov.
- procedura je lahko definirana kjerkoli v programu na osnovnem nivoju t.j. ne sme biti vgnezdena. Vse spremenljivke so globalne, parametrov nimamo. Programer lahko sprogramira lokalnost, če uporablja kompleksne instrukcije za delo z mikroproprocesorjevim skladom.
- naslovljivi so vsi stavki na nevgnezdenem nivoju ter stavki v REPEAT-bloku in sestavljenem stavku.
- globina gnezdenja je omejena na 50. Omejitev izhaja iz globine skladov, s katerimi dela program in morajo biti v FORTRAN-u vnaprej določene. FOR-stavkov lahko vgnezdimo največ 16, ker nimamo več indeknsih registrov pri mikroprocesorju.
- skočne naslove generiramo umetno.
- vsi stavki, ki so izpeljani iz določenega kontrolnega stavka imajo v polju 72-80 kolone indeks vrstice kontrolnega stavka, ki mu pripadajo. To omogoča hitro primerjavo med izvornim in prevedenim programom.
- če program odkrije manjšo napako, izpiše v listing sporočilo o vrsti in lokaciji napake. Če pa je napaka groba, se prevajanje prekine in vsa obdelana vsebina se shrani.Nato se brez obdelave prepiše še ostanek izvornega programa.Napake, ki so v zvezi z operacijskim sistemom (delo z datotekami) povzročijo izpis sporočila na uporabnikov terminal in prekinejo izvajanje.
- vgrajen je sistemski interapt, ki ob pritisku na tipko prekine prevajanje in shrani vso prevedeno vsebino, ki jo kasneje uporabnik lahko pregleda.
- maksimalna dolžina izvornega programa je 13000 vrstic.
- dolžina obeh delovnih datotek, za katere uporabnik na začetku poda ime in enoto je odvisna od dolžine izvorne datoteke in se na koncu reducira na dejansko dolžino. V prvi datoteki (80 kolonski) je prevod v drugi (132) pa je listing strukturiranega zbirnika.
- prevod je vhod dvopasovnega zbirnika, ki nam generira zbirniški listing in binarni program na poljubni enoti, ki jo specificiramo (disk, disketa, trak,...).Binarni program je lahko preveden absolutno ali relativno.

Če navedeni makro zbirnik za M68000 obdelamo s SASS ne doživi nobenih sprememb, zato kompleks strukturi-

ranega zbirnika omogoča pisanje v navadnem ali strukturiranem zbirniku.
Predstavljeni strukturirani zbirnik omogoča učinkovito programiranje v lepem stilu.

Povzetek:

Strukturirani zbirnik se je že izkazal kot zelo uporabno orodje pri razvoju 16-bitnega sistema. Omogoča strukturiran pristop k razvoju programov v zbirnem jeziku. Tako napisani programi so preglednejši in precej lažje je njihovo vzrževanje.
Strukturirani zbirnik je prvi korak v razvoju kompleksnejše programske opreme za novi 16-bitni sistem, ki bo temeljil na domačem znanju.

Literatura:

1. Janez Uratnik - 16 bitni mikroprocesor
   Motorola MC68000,INFORMATICA
   1, 1980
2. 16-bit microprocessor , User's manual,
   Motorola 1979
3. MC68000 systems cross macro assembler,
   Reference manual, Motorola 1979
4. Resident structured assembler , Reference manual
   Motorola 1979

```
      TTL PRIMER-PROGRAMA-V-STRUKTURIRANEM-ZBIRNIKU
      ORG  $1000
        PROCEDURE GREATER
        BEGIN
         LEA D(A1,D1),A0    LOAD NEW ADDRESS
         MOVE.L (A0),D3   LOAD NEW VALUE
        END
        BEGIN
SIZE    DC $20
DATA    DS.L SIZE
        END
        PROCEDURE INIT
        BEGIN
         LEA DATA,A1 LOAD BASE ADDRESS INTO A1
         CLR.L D0;  CLR.L D4; CLR.L D5   CLEAR REGISTERS
         MOVE.L A1,A0;  MOVE.L (A0),D3  LOAD ADDRESS AND VALUE
        END
        CALL INIT      INITIALIZATION
*EXECUTION 1
*PROGRAM WHICH FINDS THE GREATEST VALUE IN ARRAY 'DATA' LEAVING
*THAT VALUE IN REGISTER D3 AND THE ADDRESS OF ITS FIRST OCCURENCE IN
*REGISTER A0
        FOR.L D1=#4 TO #4*(SIZE-1) BY #4 DO !D1 IS USED AS AN INDEX
        IF.L 0(A1,D1) <GT> D3
         THEN CALL GREATER
*EXECUTION 2
*TEST THE GREATEST VALUE
*NUMBER OF 0'S IS IN D4
*NUMBER OF 1'S IS IN D5
        REPEAT
         ADD.B #1,D0   D0 IS USED AS A POINTER
         BTST.L D3,D0
         IF <EQ> THEN ADD.B #1,D4
                 ELSE ADD.B #1,D5
        UNTIL.B #32 <EQ> D0
*                   D3 - THE GREATEST VALUE
*                   D4 - NUMBER OF 0'S THE GREATEST VALUE
*                   D5 - NUMBER OF 1'S THE GREATEST VALUE
*                   D5 - NUMBER OF 1'S THE GREATEST VALUE
*                   A0 - ADDRESS OF THE GREATEST VALUE
        END
```

UPORABNI PROGRAMI

Avtomatična začetna naložitev uporabniškega programa v sistemu CP/M

```
**********************************
*  Informatica UP 6              *
*  CP/M Autoload Feature         *
*  maj 1982                      *
*  Anton P. Železnikar           *
*  sistem CP/M, operacijski s.   *
**********************************
```

1. Področje uporabe

Opisani primer je modifikacija operacijskega sistema, točneje modifikacija njegovega modula CCP (konzolnega ukaznega procesorja sistema CP/M). Ta modifikacija povzroči, da se ob začetnem zagonu sistema začne brez dodatnega posega uporabnika takoj izvajati določen uporabniški program. Takšen zagon sistema onemogoči uporabniku doseganje CP/M operacijskega sistema in izvaja se lahko le določen program za določene namene. Ta lastnost sistema je tako zlasti priporočljiva pri sistemih na ključ, tako da se uporabniku ni potrebno seznanjati z ukazi operacijskega sistema.

Podoben primer bi se lahko pojavil tudi tedaj, ko imamo računalnik doma in se otroci med drugim večkrat poigrajo z ukazom ERA *.*, ki zbriše celotno disketo.

2. Kratek opis modifikacije

CP/M sistem ima pravzaprav že vgrajeno lastnost avtomatičnega začetnega nalaganja, le da ta lastnost ni objavljena v dokumentaciji proizvajalca Digital Research. Ta informacija se praviloma daje samo OEM proizvajalcem in distributerjem programske opreme. Pokažimo sedaj, kako je mogoče ta mehanizem realizirati na vsakem CP/M sistemu.

Vzemimo program za izračunavanje bioritma z imenom BIO2, ki je bil razvit v jeziku CBASIC2 in se izvaja po prevodu v okviru modula za izvajanje BASIC programov z imenom CRUN2 (glej primer UP 3, Informatica 5(1981), št.3, str. 79). Program BIO2 naj se začne izvajati avtomatično ob začetnem zagonu pod kontrolo programa CRUN2.

Ker imamo celoten operacijski sistem (CP/M = INIT + BDOS + BIOS) shranjen v zbirki DOS64P, pokličemo to zbirko v hitri pomnilnik s prehodnim programom DDT in jo modificiramo. Najprej imamo tole:

```
B>DDT DOS64.COM        --- modificiramo DOS64.COM
DDT VERS 2.2
NEXT  PC
2300 0100              --- število zlogov zbirke
-D980                  --- izpis z začetkom 980H
```

Modificirali bomo zbirko DOS64P.COM, ki se nahaja po naložitvi z ukazom DDT na lokacijah hitrega pomnilnika (PC, NEXT-1), kot kaže zgornji primer. Z uporabo direktive 'D' in naslova začetka CCP v BDOS, dobimo tole, nekoliko okrašeno sliko:

```
     ┌───── Skočni vstop v CCP  (normalni vstop)
     │        ┌───── Skočni vstop v CCP z obhodom avt.zač.naložitve
     │        │        ┌───── 128 zlogov, dovoljenih za ukaze
     │        │        │  ┌───── Število zlogov v imenih zbirk
     │        │        │  │  ┌───── Začetek imen ukaznih zbirk
0980 C3 5C E7 C3 58 E7 7F 00 20 20 20 20 20 20 20 20 .\..X...
0990 20 20 20 20 20 20 20 20 43 4F 50 59 52 49 47 48         COPYRIGH
09A0 54 20 28 43 29 20 31 39 37 39 2C 20 44 49 47 49 T (C) 1979, DIGI
09B0 54 41 4C 20 52 45 53 45 41 52 43 48 20 20 00 00 TAL RESEARCH  ..
09C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
09D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
09E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
09F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

Iz te slike je razvidno, kaj moramo v zbirki DOS64P spremeniti (modificirati). Za to opravilo uporabimo direktivo 'S' ukaza DDT. Imamo tole:

```
-S987         --- modifikacija začne pri 987H
0987 00 0A    --- v ukazni vrstici je 10 zlogov
0988 20 43    --- ASCII znak "C"
0989 20 52    --- ASCII znak "R"
098A 20 55    --- ASCII znak "U"
098B 20 4E    --- ASCII znak "N"
098C 20 32    --- ASCII znak "2"
098D 20       --- ASCII znak "SPACE"
098E 20 42    --- ASCII znak "B"
098F 20 49    --- ASCII znak "I"
0990 20 4F    --- ASCII znak "O"
0991 20 32    --- ASCII znak "2"
0992 20 .     --- konec modifikacije
```

S ponovno uporabo direktive 'D' se pokaže tole:

10 znakov v ukazni vrstici

Začetek ukazne vrstice

```
-D980
0980 C3 5C E7 C3 58 E7 7F 0A 43 52 55 4E 32 20 42 49 .\..X...CRUN2 BI
0990 4F 32 20 20 20 20 20 20 43 4F 50 59 52 49 47 48 O2      COPYRIGH
09A0 54 20 28 43 29 20 31 39 37 39 2C 20 44 49 47 49 T (C) 1979, DIGI
09B0 54 41 4C 20 52 45 53 45 41 52 43 48 20 20 00 00 TAL RESEARCH  ..
09C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
09D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
09E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
09F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
```

Modifikacija je s tem opravljena in to kar se sedaj nahaja v hitrem pomnilniku, moramo rešiti z vgrajenim ukazom SAVE na disketo, npr. takole:

K opisanemu primeru dodajmo še tele bistvene opombe: pri navajanju zaporedje zbirk, ki bodo po vrsti izvršene, nikakor nismo omejeni samo na imena posameznih zbirk, temveč lahko uporabimo tudi SUBMIT zbirke (vključno z ukaznimi zaporedji). Pri tem smemo v celoti prekriti licenčno sporočilo podjetja Digital Research vse do lokacije 9FFH. Tako imamo dovolj prostora za vstavitev vrste imen, torej za izvedbo zelo zapletene funkcije.

Končajmo sedaj naš primer! Najprej izskočimo iz ukaza (stanja) DDT z uporabo 'CTL c' ali z G0. Takoj nato uporabimo ukaz SAVE, in sicer:

```
B>SAVE 34 BIORIT.COM
```

S tem ukazom smo rešili modificirano zbirko, sedaj pa moramo to zbirko še zapisati na sistemski stezi z uporabo prehodnega ukaza SYSGEN. Ob začetnem zagonu se tako najprej naloži CRUN2, nato še BIO2, nakar se začne BIO2 izvajati pod kontrolo programa CRUN2. To pa je prav tisto, kar smo želeli. Otroci tako nimajo več dostopa do vgrajenih in drugih prehodnih ukazov sistema CP/M.

3. Preizkus lastnosti avtomatične naložitve in izvajanja programa BIO2

S prehodnim ukazom SYSGEN smo na sistemski stezi zapisali modificirani sistem CP/M, ki naloži najprej modul CRUN2, nato modul BIO2 ter začne izvajati modul BIO2 pod kontrolo modula CRUN2. To pa je natanko lastnost, ki smo jo zahtevali in ob tem sistem CP/M za uporabnika ostane v celoti zakrit. Lista 1 pokaže, da se to zgodi tudi v praktičnem primeru. S tem je korektnost naših predpostavk praktično dokazana, uporabnik CP/M sistema pa si lahko v celoti zgradi modifikacijo za svoj poseben primer.

```
CRUN VER 2.07P
                   POCAKAJ, DA SE IZRACUNAJO TABELE !
VSTAVI IME
? PETRICA
VSTAVI DATUM ROJSTVA: DAN, MES, LETO:
? 27,6,1981
VSTAVI ZACETNI MESEC IN LETO BIOKOLEDARJA:
? 6,1982
VSTAVI STEVILO MESECEV BIOKOLEDARJA:
? 1

  BIO                                                   JUN
 INDEKS                                                1982
       --....................0....................++
 60259   F                   :           E       I      1 TOR
 62460   F                   :               E  I       2 SRE
 64650     F                 :                IE        3 CET
 66662        F              :              I   E       4 PET
 68307             F         :           I       E      5 SOB
 69387                  F    :        I         E       6 NED
 69709                       F     I           E        7 PON
 69109                       : I  F          E          8 TOR
 67465                     I :         F E              9 SRE
 64714                 I     :        E     F          10 CET
 60868              I        :   E             F       11 PET
 56014           I           E                   F     12 SOB
 50322        I          E   :                   F     13 NED
 44038      I       E        :                  F      14 PON
```

Lista 1. Ta rezultatna lista kaže, kako se pri mrzlem ali toplem zagonu javi sistem. Ker imammo kot prvi modul CRUN2, se ta modul javi s svojo verzijo 2.07P, nakar preide kontrola že na modul BIO2. Po običajni vstavitvi vhodnih podatkov se začne izpisovati bioritemski diagram, kot kaže zgornja lista.

4. Opombe k danemu primeru

Da bi bila slika popolnejša, navedimo še, katere zbirke morajo biti v našem primeru zapisane na disketi. Te zbirke so:

CRUN2.COM, CRUN204P.COM, CRUN237.COM in BIO2.INT

Na sistemskih stezah te diskete se nahaja modificirani CP/M sistem, ki smo ga poimenovali z imenom BIORIT.COM (glej SAVE ukaz). Ta sistem smo shranili v posebno zbirko zaradi možnosti dodatne modifikacije kdaj pozneje pa tudi zaradi možnosti opazovanja zgradbe te zbirke.

Program, ki pomakne samega sebe tik pod obstoječi operacijski sistem CP/M

```
*********************************
* Informatica UP 7              *
* CP/M Relocating User Program  *
* junij 1982                    *
* Anton P. Železnikar           *
* sistem CP/M, Delta 323/M      *
*********************************
```

1. Področje uporabe

Uporabnik želi večkrat imeti program, ki bi se naložil sam pod operacijski sistem v pomnilniku v odvisnosti od danega obsega pomnilnika, na katerega je CP/M sistem prilagojen. Takšen uporabniški program deluje navadno še nad kakšno drugo zbirko in ima tako določene "sistemske" lastnosti. Primer takega uporabniškega programa je npr. inverzni zbirnik, ki smo ga napisali sami in ga želimo uporabljati nad ukaznimi zbirkami tipa COM.

Pojasnimo še, zakaj želimo, da se ta uporabniški program samodejno premesti pod sam operacijski sistem CP/M. Z znanimi lastnostmi CP/M sistema bomo lahko brez posebnih težav dosegli naložitev COM zbirke na naslov 100H in navzgor. Ker bodo te COM zbirke, ki jih želimo npr. prevajati nazaj v zbirni jezik, različnih obsegov, želimo imeti na razpolago čim večji del tkim. TPA (Transient Program Area) območja. Zaradi te lastnosti želimo imeti naš uporabniški program pod samim operacijskim sistemom. Namen naše naloge je tako pojasnjen.

2. Kratek opis začetnega (premestitvenega) dela programa

Opisali bomo samo začetni del programa, saj bo nadaljevanje odvisno od konkretnega namena tega programa. Ta program bo premestil (relociral) samega sebe v odvisnosti od instaliranega CP/M sistema. Kot vemo, lahko CP/M sistem verzije 2.2 instaliramo v obsegih po 1k-zložnih stopnjah v intervalu (20k, ... , 64k). Premestitev bo tedaj odvisna od dejanske vrednosti obsega iz gornjega intervala.

Začetni del našega programa je prikazan na listi 1. V tej listi se pojavljata med drugimi spremenljivki BDOSAD in WBOOTE, ki sta odvisni od trenutno instaliranega CP/M sistema oziroma od njegovega obsega. WBOOTE določa začetek podsistema BIOS v CP/M, BDOSAD pa začetek podsistema BDOS v CP/M. S tema spremenljivkama se tako uravna premestitev uporabniškega programa v odvisnosti od obsega CP/M sistema.

Naslednji dve spremenljivki PRBEG in PRLEN se nanašata na začetek in dolžino uporabniškega programa. V naši shemi smo predvideli prostor med naslovoma 100H in vključno 1FFH za premestitveni podprogram, ki bo del celotnega programa, ki ga bomo naložili v obliki zbirke iz diskete. Ta premestitveni del je tudi prikazan v listi 1.

Modifikacijska aditivna tabela začenja takoj za koncem uporabniškega programa in je določena s spremenljivko MODTAB. V tej tabeli so pravzaprav shranjeni vsi naslovi uporabniškega programa, ki jih je potrebno modificirati v odvisnosti od obsega CP/M sistema. V vsakem zlogu te tabele je shranjena le razlika do predhodnega naslova, ki smo ga modificirali, torej aditivni pomik. Vrednost 0FFH je določena kot markacija za konec tabele ter je izražena s spremenljivko ENDMK.

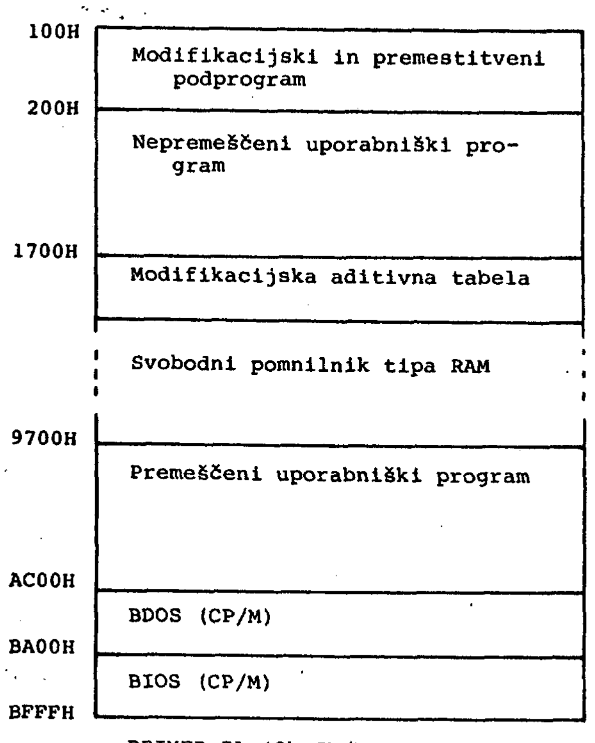
Ostale spremenljivke liste 1 (v začetnem segmentu) so poljubne in v našem primeru smo le pokazali, kako lahko predvidimo določen prostor v uporabniškemu programu za kopiranje delov iz BDOS in BIOS. To sta spremenljivki BDLEN in BIOLEN. Na lokacijo POS smo v našem primeru shranili za potrebe uporabniškega programa naslov na naslovu BDOSAD (vrstica 27 liste 1). Prvotna lokacija s tem podatkom bo namreč dobila naslov začetka premeščenega uporabniškega programa (vrstica 47).

V našem primeru se bo nerelocirani uporabniški program začel pri lokaciji 200H in se bo razprostiral do lokacije 16FFH. Na lokaciji 1700H se bo začela modifikacijska tabela, ki jo bomo ustrezno napisali k uporabniškemu programu.

Program za avtorelokacijo v listi 1 pojasnjuje samega sebe, tako da nadaljni komentar ni več potreben.

3. Izvajanje programa

Podprogram z liste 1, uporabniški program in modifikacijska tabela so shranjeni kot zbirka na disketi. Ko pokličemo to zbirko v izvajanje, se najprej uporabniški program modificira in relocira pod BDOS, nakar se začne še izvajati. S tem je naša naloga opravljena.



100H
Modifikacijski in premestitveni podprogram
200H
Nepremeščeni uporabniški program
1700H
Modifikacijska aditivna tabela
Svobodni pomnilnik tipa RAM
9700H
Premeščeni uporabniški program
AC00H
BDOS (CP/M)
BA00H
BIOS (CP/M)
BFFFH

PRIMER ZA 48k CP/M

```
ADDR  CODE        STMT SOURCE STATEMENT

                  0001 ;***************************************
'>0000            0002 RELOAD:; ZACETEK DEJANSKEGA UPORABNISKEGA
                  0003 ;          PROGRAMA:
>0200             0004 PRBEG  EQU 200H
                  0005 ;        NASLOV NASLOVA VSTOPA V BDOS:
>0006             0006 BDOSAD EQU 6
                  0007 ;        NASLOV NASLOVA VSTOPA V BIOS:
>0001             0008 WBOOTE EQU 1
                  0009 ;        DOLZINA KOPIRANJA ZLOGOV IZ BDOS
>0012             0010 BDLEN  EQU 12H
                  0011 ;        DOLZINA DEJANSKEGA UPORABNISKEGA
                  0012 ;          PROGRAMA:
                  0013 ;        KOPIJA DELA SKOCNE TABELE IZ
                  0014 ;          BIOS-A:
>0027             0015 BIOLEN EQU 27H
>1500             0016 PRLEN  EQU 1500H
                  0017 ;        ZACETEK MODIFIKACIJSKE TABELE:
>1700             0018 MODTAB EQU PRBEG+PRLEN
                  0019 ;        ZNAK KONCA V MODIFIKACIJSKI TAB.
>00FF             0020 ENDMK  EQU 0FFH
                  0021 ;        LOKACIJA ZA SHRANITEV (BDOSAD):
>0239             0022 POS    EQU PRBEG+BDLEN+BIOLEN
                  0023 ;***************************************
>0100             0024        ORG 100H
'0100  31FF01     0025        LD SP,PRBEG-1 ;NASTAVITEV SP
'0103  2A0600     0026        LD HL,(BDOSAD);*POMIK VSEBINE
'0106  223902     0027        LD (POS),HL   ;* BDOS/BDOS+11 NA
'0109  110002     0028        LD DE,PRBEG   ;* PRBEG/PRBEG+11
'010C  011200     0029        LD BC,BDLEN   ;*
'010F  EDB0       0030        LDIR          ;*
                  0031 ;***************************************
'0111  2A0100     0032        LD HL,(WBOOTE);ZACETEK SKOCNE TA-
'0114  23         0033        INC HL        ; BELE PLUS TRI
'0115  23         0034        INC HL
'0116  23         0035        INC HL
'0117  012700     0036        LD BC,BIOLEN  ;DOLZINA TABELE
'011A  EDB0       0037        LDIR          ;PRENOS SKOCNE TAB.
                  0038 ;                    V UPORABNISKI PR.
                  0039 ;***************************************
'011C  2A0600     0040        LD HL,(BDOSAD)
'011F  110015     0041        LD DE,PRLEN   ;DOLOCITEV ZACETKA
'0122  AF         0042        XOR A         ; UPORABNISKEGA PRO-
'0123  6F         0043        LD L,A        ; GRAMA GLEDE NA ZA-
'0124  E5         0044        PUSH HL       ; CETEK BDOS
'0125  ED52       0045        SBC HL,DE
                  0046 ;***************************************
'0127  220600     0047        LD (BDOSAD),HL;NOVI ZACETEK BDOS
                  0048 ;***************************************
'012A  210002     0049        LD HL,PRBEG   ;NASLOVNA MODIFIKA-
'012D  110017     0050        LD DE,PRLEN+PRBEG; CIJA UPORAB-
'0130  0600       0051        LD B,0        ; NISKEGA PROGRAMA
'0132  1A         0052 NEXTTB:LD A,(DE)
'0133  FEFF       0053        CP 0FFH       ;KONEC MOD. TAB.?
'0135  280E       0054        JR Z,EXIT-$
'0137  4F         0055        LD C,A
'0138  09         0056        ADD HL,BC     ;OBLIKOVANJE KAZALCA
'0139  FE80       0057        CP 80H        ; ZA POPRAVO
'013B  2805       0058        JR Z,OVERJ-$
```

```
ADDR  CODE        STMT SOURCE STATEMENT

'013D  3A0700     0059        LD A,(BDOSAD+1)
'0140  86         0060        ADD A,(HL)    ;OBLIKOVANJE POPRAVKA
'0141  77         0061        LD (HL),A     ;IZVRSITEV POPRAVKA
'0142  13         0062 OVERJ: INC DE
'0143  18ED       0063        JR NEXTTB-$
                  0064 ;***************************************
'0145  D1         0065 EXIT:  POP DE
'0146  1B         0066        DEC DE
'0147  21FF16     0067        LD HL,PRLEN+PRBEG-1;KOPIRANJE UPO-
'014A  010015     0068        LD BC,PRLEN   ; RAB. PROGR. NA NO-
'014D  EDB8       0069        LDDR          ; VO LOKACIJO
                  0070 ;***************************************
                  0071 ;SKOK ZA IZVAJANJE RELOCIRANEGA UPORABNI-
                  0072 ; SKEGA PROGRAMA S PRESKOKOM ZACETNE TA-
'014F  2A0600     0073        LD HL,(BDOSAD); BELE V PROGRAMU
'0152  2E3B       0074        LD L,BDLEN+BIOLEN+2
'0154  E9         0075        JP (HL)
                  0076 ;***************************************
                  0077        END

ERRORS=0000
```

Lista 1. Ta lista je tisti del celotnega uporabniškega programa (je torej podprogram), ki opravi modifikacijo dejanskega uporabniškega programa ter ga relocira pod sam vrh razpoložljivega pomnilniškega prostora v danem CP/M sistemu. Takšen tip uporabniškega programa je največkrat sistemski program, kot je npr. zbirnik, urejevalnik, inverzni zbirnik ali kakšen drugi storitveni program.
Iz programske liste je razvidno, da se pri tem vselej upošteva trenutno instalirani CP/M sistem oziroma zanj razpoložljivi pomnilni prostor. S tem je dosežena največja izkoriščenost pomnilnega prostora, saj je na razpolago največji mogoči prostor za objektni program (program, ki bo npr. prevajan, obdelan). Prav to pa si v takih primerih želimo.
Posamezni segmenti podprograma na tej listi so omejeni z zvezdnimi vrsticami. V prvem segmentu imamo definicijo spremenljivk, tako da lahko te definicije v vsakem posebnem primeru enostavno popravimo. Vobče bomo imeli različne dolžine dejanskih uporabniških programov (PRLEN), tako bo poravljena avtomatično tudi lokacija, za začetek modifikacijske tabele.
Drugi segment je inicializacijski in premestitveni (kopira se del BDOS). Tretji segment je kopirni (kopira se del BIOSa). V četrtem segmentu se izračuna začetek relociranega programa, peti segment preimenuje naslov na naslovu BDOSAD. Šesti segment opravi modifikacijo uporabniškega programa, sedmi segment pa ta modificirani del relocira. V osmem segmentu imamo skok v izvajanje relociranega programa.

NOVICE IN ZANIMIVOSTI

****
SB-80 diskovni operacijski sistem
****

SB-80 je diskovni operacijski sistem za procesorja 8080 in Z80, ki uporablja upogljive diske, vinčestrske diske ali pa tudi kombinacijo obeh diskovnih tipov. Ta sistem je v celoti združljiv s sistemom CP/M-80 in ga je moč dobiti le na OEM osnovi. Sistem je v prodaji pri podjetju Lifeboat Associates, 1651 Third Ave., New York, NY 10028, U.S.A.

SB-80 dovoljuje uporabo 16-krat večjih zbirk (glede na CP/M), vsebuje integralno lastnost tiskanja v ozadju ter ima prožnejšo množico ukazov. Ima tudi običajne storitvene programe, kot so mehanizem za prenos zbirk, urejevalnik teksta, zbirnik in popravljalnik.

SB-80 je grajen za procesorje 8080/8085/Z80 z najmanj 20k-zložnim strnenjim bralnim/pisalnim pomnilnikom, z začetkom pri naslovu 0. Sistem lahko uporabi vrsto krmilnikov za upogljive in vinčestrske diske. Dokumentacija sistema SB-80 dopušča lastno generiranje BIOSa za uporabnikovo periferijo.

Program sistema SB-80 je sestavljen iz treh modulov: iz diskovnega operacijskega sistema (DOS), zbirčnega upravljavnika in BIOSa. Prva dva modula se nahajata na sistemskih stezah diskete, enako velja tudi za BIOS, ki je odvisen od uporabnikovih krmilnikov oziroma periferije. Interpret ukazne vrstice (CLI), ki izvršuje posamezne ukaze, nalaga uporabniške programe ter sproža njihovo izvajanje, pa je shranjen na disketi kot zbirka.

Operacijski sistem zasede 10k-zlogov pomnilnika in se lahko poveča zaradi potreb v okviru BIOSa. Naslov začetka operacijskega sistema (z izjemo prve strani) lahko določi uporabnik.

Operacijski sistem dovoljuje oblikovanje, brisanje, preimenovanje, branje in zapisovanje zaporednih zbirk z zapisi spremenljive dolžine in zbirk z naključnim dostopom, ki imajo zapise stalnih dolžin. Sistem podpira uporabo 16 diskovnih enot in zbirčni prostor se dodeljuje dinamično. Največji mogoči prostor za zbirko v sistemu SB-80 znaša 128M zlogov in ta prostor je tudi največ, kar je dovoljeno za posamezno diskovno enoto. Ta prostor se dodeljuje dinamično med posameznimi zbirkami.

SB-80 ima rezidentno lastnost tiskanja iz ozadja; s tem se izločijo slabosti, ki se pojavijo pri odvijalcih (despooler), ki niso integrirani direktno v operacijski sistem. Tiskanje iz ozadja temelji na vrsti največ šestih zbirk, ki čakajo na tiskanje. Uporabnik lahko prekine, ukine ali uvede funkcijo tiskanja iz ozadja, dodaja zbirke v vrsto in jih iz nje odvzema v vsakem trenutku. Operacijski sistem ima tudi možnost paketne obdelave programov.

A.P.Železnikar

JANEZ BLEIWEIS IN KOERTING

V sestavku Bojana Štiha: Bleiweis 1982, Z nedavnega simpozija o "očetu slovenskega naroda" (Dnevnik 3.4.1982, Sobotna priloga) lahko preberemo tole zanimivo misel:

XVI.
Na vprašanje, ali bi Janez Bleiweis kupil Koerting svojega in našega časa, je treba odgovoriti ostro zanikujoče. Veselil bi se kot iskren rodoljub kranjski, slovenski in slovanski poraza germanske in nemške moči v industriji in kapitalu, obenem pa postavljal temelje taki industriji, ki bi bila kos razmeram na evropskem in svetovnem tržišču. Z Bleiweisovim odklonilnim stališčem do prevzema Koertinga v šibke slovenske roke, pa je tudi že izoblikovana misel o progresivni vlogi doktorja Janeza Bleiweisa na polju slovenskega narodnega gospodarstva.

A.P.Železnikar

CP/M-86 v integriranem vezju

Ameriško podjetje Intel je najavilo izdelavo operacijskega sistema CP/M-86 (za 16-bitni procesor I8086 in njegove naslednike) v enem samem integriranem vezju (tipa ROM). Ta sistem deluje tudi na 8-bitnem procesorje I8088, ki ga uporablja osebni računalnik podjetja IBM (Personal Computer). Takšna izvedba operacijskega sistema omogoča brezdiskovno obratovanje daljinskih postaj v porazdeljenih lokalnih mrežah, kjer se večje diskovne enote nahajajo na centralnih mestih. Vezje je ROM obsega 16k zlogov ter vsebuje še časovnike in drugo logiko; oznaka tega vezja je 8086-E3, vezje pa bodo začeli prodajati še to poletje (1982).

Podjetje Intel je tudi objavilo, da je podpisalo sporazum za OEM distribucijo s podjetjem Digital Research, ki bo oskrbovalo Intelove plošče (tiskana vezja) z naročnikovimi verzijami sistemov CP/M in MP/M.

A. P. Železnikar

****
Novice uporabniških skupin SIG/M in CPMUG
****

SIG/M skupina ja najavila dvanajst nadaljnih zvezkov CP/M programske opreme, tako da ima sedaj skupaj 55 zvezkov (disket). SIG/M skupina je izdala tudi 12-stranski katalog svoje programske opreme, ki navaja programsko knjižnico in listo distribucijskih pogojev. Cena tega kataloga je $ 2,00 za zračno pošto (Evropa), naslov pošiljatelja pa je: SIG/M, Box 97, Iselin, NJ 08830.

Tudi CPMUG skupina je najavila in že realizirala izdajo 15 novih programskih zvezkov, tako da jih ima sedaj že preko 80. Pri tem velja omeniti, da so zvezki 55 do 75 ponatisi starih SIG/M zvezkov 1 do 20. Na upogljivih diskih obeh uporabniških skupin je tako več deset milijonov zlogov uporabniških in sistemskih programov. Cena diskete je $ 12,00 (za Evropo) pri CPMUG. Uporabnik lahko dobi na ta način ceneno sistemsko in aplikativno programsko opremo.

A. P. Železnikar

****
68000 produkti podjetja Cromemco
****

Podjetje Cromemco iz Mountain Viewa, Ca je začelo dobavljati produkte, temelječe na procesorski tehnologiji 68000. Ti produkti so bili najavljeni že v septembru/oktobru leta 1981 v časopisu Microsystems. Novi sistem je sestavljen iz treh S-100 plošč: DPU (Dual Processor Unit) plošče z dvema procesorjema (68000 in Z80) (cena $ 995), MCU pomnilniškega krmilnika (cena $ 495) in 256MSU pomnilnika s 256k zlogi (cena $ 1995). Dobavljiva je tudi plošča s 512k zlogi.

DPU lahko izvršuje izmenoma ukaze procesorjev Z80 in 68000 in med tem, ko na tržišču še ni izdatne programske opreme za 68000, se uporablja programska oprema za procesor Z80. Vsaka MCU enota se lahko uporabi za krmiljenje osmih 256MSU plošč. Uporabljena pomnilniška vezja so dinamična in plošča ima mehanizem za razpoznavanje napak in njihovo poravljanje. Napake se tudi beležijo in so za uporabnika razvidne. Tako lahko uporabnik sistema sam ugotovi, katero integrirano pomnilno vezje povzroča napake in to vezje sam zamenja.

A. P. Železnikar

****
PODATKI O PREVAJALNIKU PASCAL/Z
****

Pascalski prevajalnik z oznako Pascal/Z deluje na CP/M sistemih s procesorjem Z80, imeti pa mora 56k zlogov "čistega" pomnilnika za svojo uporabo. Podatki o napakah prevajalnika se objavljajo dvomesečno, prav tako pa se preizkušajo poslani pascalski programi, ki pridejo v poštev za prodajo in distribucijo. Tisti, ki že imate programe, napisane v Pascal/Z, jih lahko pošljete na naslov: Charlie Foster, Director, Z-User Group, 7962 Center Pkwy, Sacramento, Ca 95823.

A. P. Železnikar

****
CP/M katalog
****

V februarju letos je izšla že druga knjiga tkim. kataloga programske opreme za javno uporabo. Ta katalog je izdal New York Computer Club Inc., P.O.Box 106, New York, NY 10008. Natančen naslov publikacije je: The Catalog of Public Domain Software for CP/M. Ta katalog vsebuje pregled objavljenih disket iz prve knjige (49 disket) ter zaporedne številke CP/M disket 50 do 53 in SIG/M disket od 1 do 42. Posamezni programi iz teh disket so podprti s primeri in v tej knjigi najdemo vrsto zanimivih in uporabnih informacij. Seveda je ta druga knjiga namenjena "pravim" uporabnikom CP/M programske opreme.

A. P. Železnikar

****
Pohitritev CP/M sistema
****

Cache/Q proizvajalca Queue Computer Corporation povečuje hitrost izvajanja CP/M sistema za faktor 35. Cache/Q zahteva CP/M 2.2 sistem s 64k zlogi pomnilnika ali tudi manj. Pri tem je moč uporabiti tudi mehanizem pomnilniških bank. Prenosi podatkov na in iz disket se opravljajo preko vmesnikov, tako da se znižuje aktivnost disket pri dani aplikaciji. Cache/Q je za uporabnika in za sistemske programe transparenten.

Ta novi produkt ima svoj instalacijski program, ki je interaktiven, nadalje rekonfiguracijski (modifikacijski) program, s pomočjo katerega uporabnik lahko določi, katere zbirke ali razred zbirk bodo dostopljene ali pisane skozi vmesnike. Paket vsebuje tudi prikazovalni program, ki pokaže operacijsko statistiko za Cache/Q.

A.P.Železnikar

****
Pascalski prevajalnik za procesor 6809
****

Pascalski prevajalnik podjetja Omegasoft deluje na operacijskih sistemih MDOS, XDOS, Flex, DOS69 ali OS-9. Enoprehodni prevajalnik prevaja hitro pascalske programe v optimizirani zbirni jezik procesorja 6809. Ta prevajalnik dopušča dolga cela števila, tako da je moč komercialne podatke obravnavati hitro brez izgube hitrosti pri dvojni natančnosti realnih ali BCD števil. Prevajalnik ima v celoti dinamično dodeljevanje pomnilnika spremenljivkam z uporabo procedur NEW, DISPOSE, MARK in RELEASE. Prevajalnik podpira zbirke z naključnim dostopom. Druge lastnosti prevajalnika so še simbolično odpravljanje napak, izvajalna knjižnica z izvirnim kodom in storitveni program za pomoč in oblikovanje verižne zbirke, ki povezuje uporabniški program z knjižnico izvajalnega časa.
Prevajalnik za Pascal je napisan v strojnem kodu procesorja 6809 in potrebuje sistem z 48k zlogi. Razen prevajalnika sta na voljo tudi premestitveni zbirnik in povezovalnik. K prevajalniku je dodan jezikovni priročnik in konfiguracijski priročnik, ki določa oziroma opisuje spremenljivost sistema. Cena prevajalnika znaša $ 425, zbirnik in prevajalnik pa imata ceno $ 75. Naslov proizvajalca je: Omegasoft Industrial Products Group, P.O.Box 70265, Sunnyvale, Ca 94086.

A. P. Železnikar

****
Priročnik za operacijski sistem UNIX
****

Uporabniški priročnik za sistem UNIX sta napisala Jean Yates in Rebecca Thomas ter predstavlja uvod v zgradbo, programsko opremo in lastnosti tega znanega operacijskega sistema. Knjiga je pisana za začetnika in opisuje začetne korake na sistemu, kot so vstop v lupino, obdelava zbirk, pošiljanje pošte in oblikovanje imenikov. Opisani so tudi storitveni programi sistema UNIX in njihove lastnosti. Dana je informacija o programski opremi za sistem UNIX in za sorodne proizvode, naslovi univerz, uporabniških skupin in časopisov, ki se ukvarjajo s problematiko tega operacijskega sistema.

Cena te knjige (broširane) je $ 15,99, izdajatelj pa je Osborne/McGraw-Hill, 630 Bancroft Way, Berkeley, Ca 94710.

A. P. Železnikar

SREČANJA

8-13 avgust, Montreal , Canada

10th IMACS World Congress on System Simulation and Scientific Computation

Informacije: IMACS Secretariat, Departement of Computer Science, Rutgers University, New Brunswick, NJ 08903

19 -21 avgust, St.Gall, Švica

7th Symposium on Operation Research

Organizator: Society for Mathematics, Economics and Operation Research
Informacije: H.Loeffel and P.Stahly, Institut fur Unternehmensforschung, Hochschule St.Gallen, Bodanstrasse 6, CH-9000 St.Gallen, Switzerland

24-27 avgust, Bordeaux, Francija

First Working Conference on Advances in Production Management Systems - APMS 82

Organizator: IFIP WG 5.7
Informacije: IFIP Secretariat, 3, rue du Marche, CH-1204 Geneva, Switzerland

29 avgust- 3 september, Ghent, Belgium

Mathematical and Computer Aided Modelling in Medical Engineering and Biocybernetics

Organizator: IFIP WG 7.1
Informacije: IFIP Secretariat, 3, rue du Marche, CH-1204 Geneva, Switzerland

30 avgust-3 september, Toulouse, Francija

5th Symposium on Computational Statistics

Organizator: IASC
Informacije: COMPSTAT 82, c/o Lab. de Statistique et Probabilites, E.R.A -C.N.R.S 591
118 route de Narbonne, 31062 Toulouse Cedex, France

31 avgust - 3 september, Berlin, ZRNemčija

International Congress for Data Processing and Information Technology

Organizator:AMK Berlin and ACM European region
Informacije:AMK Berlin, Congress and Convention Division Dep. K 1, Messedamm 22, D-1000 Berlin, W.Germany

1-3 september, Berlin, ZR Nemčija

Secon International Symposium on Distributed Data Bases

Organizator: IFIP TC 2
Informacije:IFIP Secretariat, 3 rue du Marche, CH-1204 Geneva, Switzerland

6-12 september, Haifa, Israel

EUROMICRO 82

Organizator: EUROMICRO

7-10 september, London, Velika Britanija

Sixth International Conference on Computer Communication ICCC 82

Organizator: ICCC
Informacije: ICCC 82, P.O.B. 23, Northwood Hills HA6 ITT Middlesex, U.K.

7-10 september, Kiel, ZRNemčija

Working Conference on Data Protection and Health Informatics Systems

Organizator: IMIA
Informacije:Mr.J.Roukens, Commission of the European Communities, DG III/B/1, A25-3/8, Wetstraat 200 B-1049 Brussels,Belgium

8-10 september,Manchester, Velika Britanija

Eurographics 82

Organizator:Eurographics association, IFIP W.G. 5.2 BCS, CAD/CAM Association, University of Manchester, ACM European Chapiter
Informacije: Eurographics 82, c/o 170A Park Road, Peterborough, England PE1 2UF

12-18 september, Harrogate, Velika Britanija

Working Conference on Nursing Informatics

Organizator:IMIA
Informacije: Mr.J.Roukens, Commission of the European Communities, DG III/B/1, A25-3/8, Wetstraat 200 B-1049 Brussels, Belgium

20-24 september, Paris, Francija

Working Conference on Education in Health Informatics

Organizator: IMIA
Informacije: Mr.J.Roukens, Commission of the European Communities, DG III/B/1, A25-3/8, Wetstraat 200 B-1049 Brussels, Belguim

27-29 september, Baden-Baden, ZR Nemčija

General Conference on Analysis, Design and Evaluation of Man-Machine Systems

Organizator:IFAC/IFIP/IFORS/IEA
Informacije:Prof.T.Vamos, Computer and Automation Institute Hungarian Academy of Sciences, B.O.X 63,H 1502 Budapest 112, Hungary

28-30 september, Paris, Francija

2nd International Conference on the Impact of CAD in Small and Medium Sized Industries

Organizator:IFIP TC 5
Informacije: IFIP Secretariat, 3 rue du Marche, CH-1204 Geneva, Switzerland

5-8 oktober, Madrid, Španija

3rd IFAC/IFIP Symposium on Software for Computer Control SOCOCO 82

Organizator:IFIP/IFAC
Informacije:Prof.T.Vamos, Computer and Automation Institute, Hungarian Academy of Sciences, B.O.Box 63 H-1502 Budapest 112, Hungary

19-22 oktober, Munich, ZRNemčija

6th International Conference on Pattern Recognition - ICPR 82

Organizator: IAPR
Informacije:IFIP Secretariat, rue du Marche,CH-1204 Geneva, Switzerland

26-28 oktober, Berlin, ZRNemčija

International Symposium on Medical Imaging and Image Interpretation

Organizator:IEEE-CS
Informacije: ISMIII 82, 1109 Spring Street, Suite 201 Silver Spring, MD 20901; USA

3-5 november, Versailles, Francija

Real Time Data 82,. Second International Conference on Real Time Data Handling and Process Control

Organizator:INRIA
Informacije: T.Bricheteau, INRIA, Services des Relationes Exterieures, Domaine de Voluceau- BP. 105, 78 153 Le Chesnay Cedex, France

21-22 november, Rive del Sole, Italija

Working Conference on System Design for and with Users

Organizator: IFIP TC 9, WG 9.1
Informacije: IFIP Secretariat, rue du Marche, CH-1204 Geneve, Switzerland

13-15 december, Bangalore, India

Second Conference on Foundations of Software Technology and Theoretical Computer Science

Organizator: National Centre for Software Development and Computing Techniques, Tata Institute of Fundamental Research
Informacije:M.Joseph, NCSDCT, TATA Institute of Fundamental Research, Colaba , Borbay 400 005, India

13-15 december, Cairo, Egypt

First International ESIT/ASIS Conference

Organizator:Egyptian Society for Information Technology and American Society for Information Science
Informacije: Skip McAfee, ASIS, 1010 Sixteenth St., N.W., Washington D.C 20036, USA

15-20 december, Oslo, Norveška

General Conference on Comparative Review of Information Systems Methodologies

Organizator: IFIP WG 8.1/NCS
Informacije: IFIP Secretariat, rue du Marche, CH-1204 Geneve, Switzerland

1983

17-21 januar, Berlin, ZR Nemčija

Communications in Distributed Systems - Applications and Operations

Organizator:Gesellschaft fur Informatik and Nachrichtentechnische Gesellschaft
Informacije:Otto Spaniol, Fachbereich Informatik, Rechnerbetriebssysteme Universitat Frankfurt, 6000 Frankfurt, W.Germany

23-25 februar, Vienna, Austria

First International Conference on Governmental and Municipal Data Processing

Organizator:IFIP/ADV
Informacije:Conference Secretariate, ADV-Arbeitsgemeinschaft fur Datenverarbeitung, Trattnerhof 2, A-1010 Vienna Austria

14-18 marec, Tel Aviv, Izrael

International Conference on Mini and Micro Computer Applications and Operations

Organizator:National Centre for Scientific and Technological Information
Informacije: Secretariat, Mini and Micro Computer Applications and Operations, Documentations and Libraries, P.O.Box 3054, Tel Aviv, 61030, Izrael

NOVE KNIGE IN ČASOPISI

Založniška hiša North-Holland Publishing Company je v letu 1982 začela z izdajanjem časopisa " COMPUTERS AND STANDARDS". Objavljeni prispevki bodo obravnavali teme s področja razvoja računalniških standardov, njihovo implementacijo in uporabo.
Uporaba računalniških standardov zajema naslednja področja: avtomatizacija pisarniškega poslovanja, proektiranje s pomočjo računalnika (CAD/CAM), robotika, programski jeziki, podatkovne baze, informacijski sistemi, računalniška grafika, vhodno/izhodni vmesniki, podatkovni slovarji, dokumentacija, operacijski sistemi in dr. Objavljeni bodo tudi prispevki s področja aplikacij računalniških standardov v tehnologiji, eknomiji ter trgovini. Namen časopisa je da informira mednarodno EDP javnost o poteku razvoja računalniških standardov na mednarodnem in nacionalnem nivoju ter da zagotovi forum za mednarodno javno debato o problematiki računalniških standardov. Naročniki časopisa so: računalniški strokovnjaki(individualno), institucije za standardizacijo ter agencije, ki sodelujejo pri razvoju standardov ali pa so vključeni na kakšen drug način v procesu izdelave in uporabe računalniških standardov.
Glavni in odgovorni urednik je John L. Berg, Standard Oil Company,Indiana, USA. V uredniškem odboru so strokovnjaki iz ZDA, Kanade, Francije,Japonske, SSSR, ZRNemčije, Italije in Belgije. Letna naročnina je 80 $.

NAVODILO ZA PRIPRAVO ČLANKA

Avtorje prosimo, da pošljejo uredništvu naslov in kratek povzetek članka ter navedejo približen obseg članka (število strani A 4 formata). Uredništvo bo nato poslalo avtorjem ustrezno število formularjev z navodilom.

Članek tipkajte na priložene dvokolonske formularje. Če potrebujete dodatne formularje, lahko uporabite bel papir istih dimenzij. Pri tem pa se morate držati predpisanega formata, vendar pa ga ne vrišite na papir.

Bodite natančni pri tipkanju in temeljiti pri korigiranju. Vaš članek bo s foto postopkom pomanjšan in pripravljen za tisk brez kakršnihkoli dodatnik korektur.

Uporabljajte kvaliteten pisalni stroj. Če le tekst dopušča uporabljajte enojni presledek. Črni trak je obvezen.

Članek tipkajte v prostor obrobljen z modrimi črtami. Tipkajte do črt - ne preko njih. Odstavek ločite z dvojnim presledkom in brez zamikanja prve vrstice novega odstavka.

Prva stran članka :
a) v sredino zgornjega okvira na prvi strani napišite naslov članka z velikimi črkami;
b) v sredino pod naslov članka napišite imena avtorjev, ime podjetja, mesto, državo;
c) na označenem mestu čez oba stolpca napišite povzetek članka v jeziku, v katerem je napisan članek. Povzetek naj ne bo daljši od 10 vrst.
d) če članek ni v angleščini, ampak v katerem od jugoslovanskih jezikov izpustite 2 cm in napišite povzetek tudi v angleščini. Pred povzetkom napišite angleški naslov članka z velikimi črkami. Povzetek naj ne bo daljši od 10 vrst. Če je članek v tujem jeziku napišite povzetek tudi v enem od jugoslovanskih jezikov;
e) izpustite 2 cm in pričnite v levo kolono pisati članek.

Druga in naslednje strani članka:
Kot je označeno na formularju začnite tipkati tekst druge in naslednjih strani v zgornjem levem kotu,

Naslovi poglavij:
naslove ločuje od ostalega teksta dvojni presledek.

Če nekaterih znakov ne morete vpisati s strojem jih čitljivo vpišite s črnim črnilom ali svinčnikom. Ne uporabljajte modrega črnila, ker se z njim napisani znaki ne bodo preslikali.

Ilustracije morajo biti ostre, jasne in črno bele. Če jih vključite v tekst, se morajo skladati s predpisanim formatom. Lahko pa jih vstavite tudi na konec članka, vendar morajo v tem primeru ostati v mejah skupnega dvokolonskega formata. Vse ilustracije morate (nalepiti) vstaviti sami na ustrezno mesto.

Napake pri tipkanju se lahko popravljajo s korekcijsko folijo ali belim tušem. Napačne besede, stavke ali odstavke pa lahko ponovno natipkate na neprozoren papir in ga pazljivo nalepite na mesto napake.

V zgornjem desnem kotu izven modro označenega roba oštevilčite strani članka s svinčnikom, tako da jih je mogoče zbrisati.

Časopis INFORMATICA
Uredništvo, Parmova 41, 61000 Ljubljana

Naročam se na časopis INFORMATICA. Predplačilo bom izvršil po prejemu vaše položnice.

Cenik: letna naročnina za delovne organizacije 500,00 din, za posameznika 200,00/100,00/50,00 din

Časopis mi pošiljajte na naslov ☐ stanovanja ☐ delovne organizacije.

Priimek.......................................

Ime.......................................

Naslov stanovanja

Ulica.......................................

Poštna številka________ Kraj.......................................

Naslov delovne organizacije

Delovna organizacija.......................................

....

Ulica.......................................

Poštna številka________ Kraj.......................................

Datum..................... Podpis:

....

INSTRUCTIONS FOR PREPARATION OF A MANUSCRIPT

Authors are invited to send in the address and short summary of their articles and indicate the approximate size of their contributions (in terms of A 4 paper). Subsequently they will receive the outor's kits.

Type your manuscript on the enclosed two-column-format manuscript paper. If you require additional manuscript paper you can use similar-size white paper and keep the proposed format but in that case please do not draw the format limits on the paper.

Be accurate in your typing and through in your proof reading. This manuscript will be photographically reduced for reproduction without any proof reading or corrections before printing.

INFORMATICA, Journal Headquarters
Parmova 41, 61000 Ljubljana, Yugoslavia

Please enter my subscription to INFORMATICA and send me the bill.

Annual subscription price: companies US $ 22, individuals US $ 7,5.

Send journal to my ☐ home address ☐ company's address.

Surname...

Name..

Home address

Street..

Postal code __________ City.........................

Company address

Company...

....

Street..

Postal code __________ City.........................

Date.......................... Signature

....

Use a good typewriter. If the text allows it, use single spacing. Use a black ribbon only.

Keep your copy within the blue margin lines on the paper, typing to the lines, but not beyond them. Double space between paragraphs.

First page manuscript:
a) Give title of the paper in the upper box on the first page. Use block letters.
b) Under the title give author's names, company name, city and state - all centered.
c) As it is marked, begin the abstract of the paper. Type over both the columns. The abstract should be written in the language of the paper and should not excesed 10 lines.
d) If the paper is not in English, drop 2 cm after having written the abstract in the language of the paper and write the abstract in English as well. In front of the abstract put the English title of the paper. Use block letters for the title. The lenght of the abstract should not be greater than 10 lines.
e) Drop 2 cm and begin the text of the paper in the left column.

Second and succeeding pages of the manuscript:
As it is marked on the paper, begin the text of the second and succeeding pages in the left upper corner.

Format of the subject headings:
Headings are separated from text by double spacing.

If some characters are not available on your typwriter write them legibly in black ink or with a pencil. Do not use blue ink, because it shows poorly.

Illustrations must be black and white, sharp and clear. If you incorporate your illustrations into the text keep the proposed format. Illustration can also be placed at the end of all text material provided, however, that they are kept within the margin lines of the full size two-column format. All illustrations must be placed into appropriate positons in the text by the author.

Typing errors may be corrected by using white correction paint or by retyping the word, sentence or paragraph on a piece of apaque, white paper and pasting it nearly over errors

Use pencil to number each page on the upper-right-hand corner of the manuscript, outside the blue margin lines so that the numbers may be erased.

DELTA-KOPA 2000
video terminal

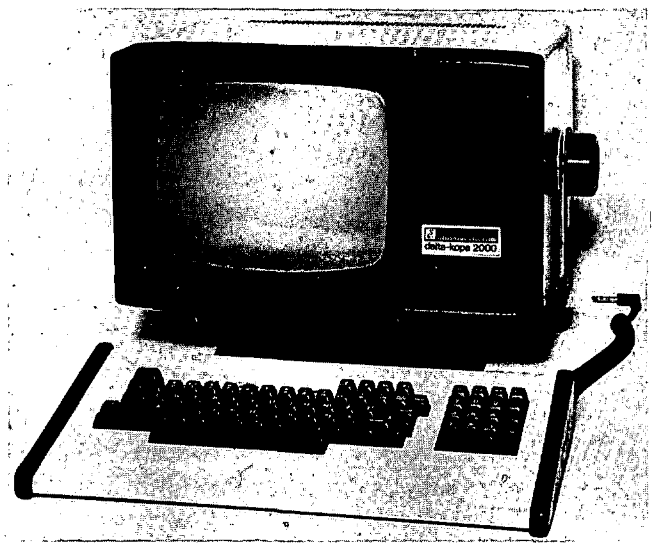
računalniški sistemi delta

SPLOŠNI OPIS

Video terminal DELTA KOPA 2000 je računalniška vhodno/izhodna enota. Terminal je zasnovan na mikroprocesorski tehnologiji in ga lahko izpopolnimo in usposobimo za opravljanje zahtevnejših nalog. Je enostaven, vendar z lastnostmi, ki olajšajo delo in izboljšajo komunikacijski odnos računalnik — človek.
Terminal DELTA KOPA 2000 je v celoti plod lastnega razvoja DO DELTA.

LASTNOSTI

— do 132 znakov v vrstici
— dvojna velikost znakov; dvojna višina in dvojna širina
— delitev zaslona
— možnost dopolnitve v samostojen sistem
— izbira svetlega ali temnega ozadja na zaslonu
— *matrika za izpis znakov s 7 × 9 točkami*
— utripanje, podčrtavanje in dvojna jakost znakov, kombinacija vseh atributov na enem znaku, brez izgube položaja na ekranu
— stalni in nastavljivi tabulatorji
— ustavljanje izpisa pri polnem zaslonu
— ločena tastatura
— standardna skupina številčnih tipk
— posebni grafični znaki
— *izjemno enostavno vzdrževanje*
— vgrajeni lastni diagnostični programi
— poseben izhod za video signal
— poseben izhod za tiskalnik
— prilagoditev nagiba monitorja
— namizna, viseča in stenska montaža
— univerzalni močnostni del
— različne prenosne hitrosti
— kompatibilni način delovanja s KOPO 700, KOPO 1000, z VT 100, VT 52
— LED indikatorji za kontrolo delovanja
— brezkontaktna nastavitev lastnosti
— izbrane lastnosti terminala se ohranijo tudi ob izklopu
— dupleksni prenos prek asihrone komunikacijske linije



TEHNIČNE SPECIFIKACIJE

Dimenzije:	
Monitor/ brez podstavka	dolžina 46 cm širina 43 cm višina 28 cm
s podstavkom	dolžina 52 cm širina 43 cm višina 36 cm
Tastatura	dolžina 46 cm širina 24 cm višina 6 cm
Teža:	15,6 kp
Pogoji delovanja:	temperatura od 10—40° C relativna vlaga 10—90 %
Napajanje:	180—256 V/47—63 Hz/100 VA
Zaslon:	
Katodna cev	diagonala meri 31 cm, fosfor GR
Format	24 vrstic po 80 znakov ali 24 vrstic po 132 znakov (po izbiri)
Znaki	matrika s 7 × 9 točkami
Aktivna površina zaslona	205 mm × 115 mm
Znakovni niz	96 ASCII znakov
Tastatura:	
Tipke	65 tipk je izdelanih in razporejenih podobno kot pri pisalnem stroju
Pomožna tastatura	18 numeričnih tipk s piko, vejico, minusom, tipko ENTER in štirimi programsko-funkc. tipkami zvočna potrditev vtipkanega znaka in mejni signal za napako
Povezave:	
Tip	EIA (RS-232-C)
Hitrosti:	polni dupleks 50, 75, 110 (dva stop bita), 134, 150, 200, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 9600, 19200
Format znakov	asinhronski
Dolžina znakov	7 ali 8 bitov; izbira na tastaturi. (Če izberemo 8 bitov za znak, osmi bit ne nosi informacije.)
Kode	USASCII, JUS A.FO.101
Parnost	soda, liha, če je ni, izbira na tastaturi
Sinhronizacija	izbiramo jo s tastaturo, tako da generiramo kontrolno kodo XON/XOFF s tiskalnikom CTS ali XON/XOFF

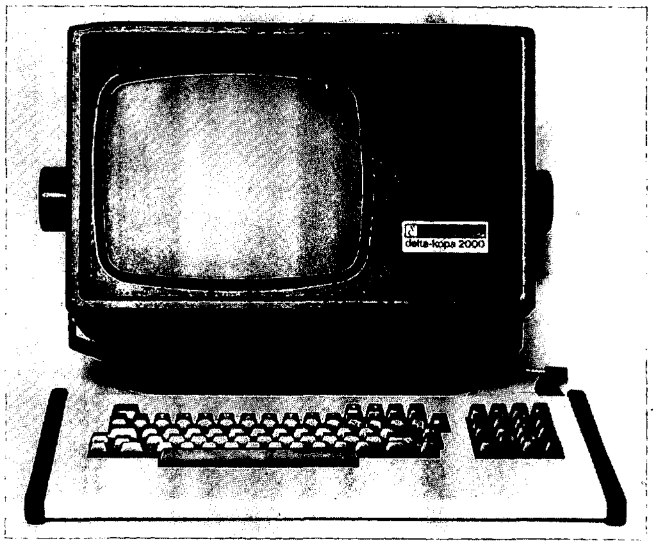
Iskra **Delta**, Ljubljana, Yugoslavia

TASTATURA

Podobna je tastaturi pisalnega stroja in je ločena od ohišja monitorja. Z njim jo povezuje 1,50 m dolg kabel, ki dovoljuje postavitev monitorja iin tastature v različne položaje. S tem dosežemo zorni in delovni kot. Na tastaturi so posebne funkcijske tipke za prenos kontrolnih znakov, ki krmilijo delovanje terminala. Skupina številčnih in funkcijskih tipk, oblikovana podobno kot pri kalkulatorjih, služi za vnašanje numeričnih podatkov in uporabo programskih operacij na terminalu. Na tastaturi je 7 LED indikatorjev, ki dajejo operaterju informacijo o delovanju terminala in služijo za odkrivanje napak.

ZASLON

Ena od prednosti video terminala DELTA KOPA 2000 je, da lahko prikazuje poročila v dveh formatih: 80 in 132 znakov v vrstici. 132 znakov v vrstici omogoča zapis poročil, ki so standardno generirana v formatu za tiskalnik in direkten prenos iz zaslona na tiskalnik brez preoblikovanja. V načinu delovanja drseče obračanje (SMOOTH-SCROLL) lahko operater kontrolira podatke pri visokih hitrostih prenosa. S tipko NO-SCROLL pa lahko izpis kjerkoli ustavi in ga s pritiskom ponovno sproži. Zaslon lahko logično razdelimo v dva dela tako, da se del 24-vrstičnega zaslona odvija ločeno. Podatke lahko vpisujemo na enem in izpisujemo na drugem delu zaslona, kar je ugodno za programiranje in operaterja.



S pritiskom na posebno SET-UP tipko operater nadomesti vsebino zaslona s standardno sliko (SET-up A). S pomočjo te slike lahko izbere število znakov v vrstici, določi svetlost zaslona in nastavi tabulatorje. S pritiskom na tipko 5 operater zamenja prikaz SET-UP A s prikazom SET-UP B. Ta prikaz omogoča izbiro prenosnih hitrosti in drugih lastnosti terminala (npr. svetlo ozadje na zaslonu, oblika zaslonskega kazalca, mejni signal).

ZNAKI

Matrika za izpis znakov obsega 7 × 9 točk in se razprostira na prostoru 10 × 10 točk, kar omogoča spuščanje nižje ležečih znakov za dve točki. Operater lahko izbere svetle znake na temni podlagi ali temne znake na svetli podlagi, in sicer za vsak znak posebej ali za cel zaslon. Ta lastnost poudarja določene dele teksta, temni znaki na svetlem ozadju pa dajejo videz tiskanega teksta na papirju. Uporabniku je na voljo dvojna višina in dvojna širina znakov, s čemer dosežemo preglednost teksta in čitanje na večjo razdaljo. Osnovni nabor znakov vsebuje poleg črk, številk in ločil, še 39 grafičnih znakov za prikaz grafičnih informacij na zaslonu.

SPLOŠNI PODATKI

Video terminal DELTA KOPA 2000 ima dve mehanski stikali, eno za vklop terminala in drugo za preklop močnostnega dela. Vse druge funkcije terminala, kot so prenosna hitrost, tabulatorji, pariteta, itd., so shranjene v posebnem pomnilniku in jih spreminjamo preko tastature. Nastavljive lastnosti terminala se ohranijo, tudi če terminal izključimo in ga ponovno vključimo. Odstranitev mehanskih stikal olajša uporabo testnih diagnostičnih programov in omogoča enostavno prilagajanje terminala. Vgrajeni testni diagnostični programi poenostavijo vzdrževanje in zmanjšajo čas osamitve in popravila napak. Ohišje je spojeno s čepi, kar omogoča hiter dostop in lahko vzdrževanje.

Univerzalni močnostni del je prilagojen za napajanje terminala in vseh dodatkov in omogoča njihovo vgrajevanje na terenu.

Posebni izhod za video signal lahko krmili pomožni video monitor in tako omogoča posredovanje podatkov večji skupini ljudi. Terminal DELTA KOPA 2000 deluje z dupleksno asinhrono komunikacijsko linijo in ima standardni vmesnik EI A 232 in 20 mA vmesnik.
Novost terminala DELTA KOPA 2000 je, da poleg glavnega vhoda vsebuje posebna serijska vrata za tiskalnik.