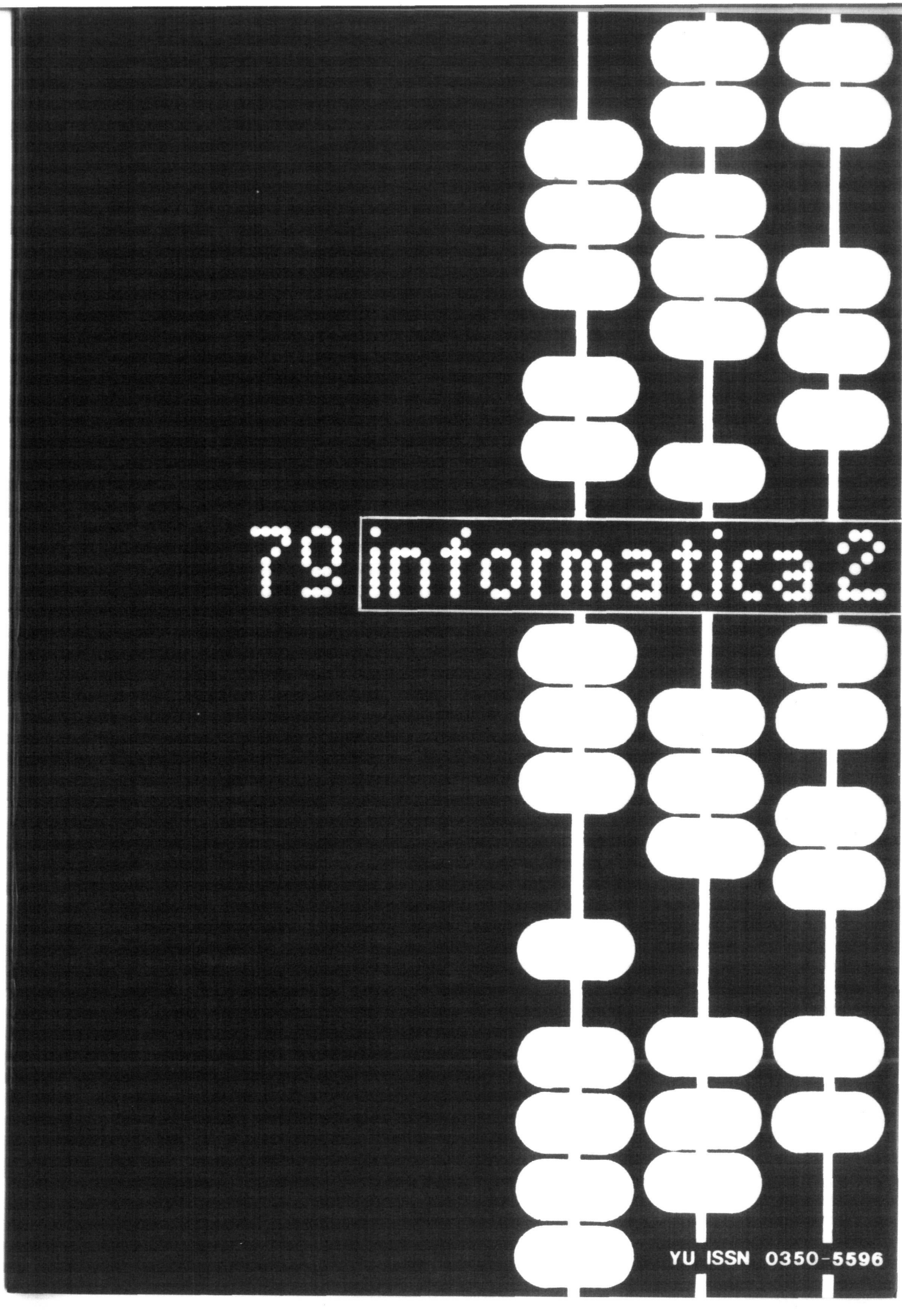
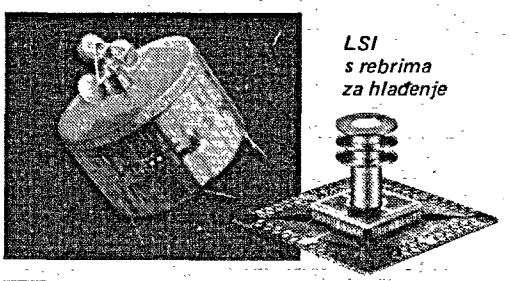

79 informatica 2
YU ISSN 0350-5596

FACOM kompjutere proizvodi Fujitsu, tvrtka koja najveću pažnju posvećuje sistemima.



LSI s rebrima za hlađenje

Prije svega kompjuter je sistem, tj. sredstvo za obradu podataka koji u sebi sadrži hardware, software i aplikacionu tehnologiju. Naravno razne tvrtke bave se prodajom kompjutera. Ipak, malo je tvrtki koje mogu ponuditi potpuni izbor sredstava za automatsku obradu podataka — konstruirani tako, da osim optimalnih performanci, imaju mogućnost ugradnje u veće sisteme.

FUJITSU je jedna od tvrtki koja to može ponuditi. Kao vodeći proizvođač kompjuterskih sistema u Japanu, FUJITSU proizvodi široki asortiman proizvoda od minikompjutera s jednim LSI čipom do u svijetu najmoćnijih LSI sistema, kao i široki izbor periferne i terminalne opreme.

FACOM kompjuteri obavljaju važne aktivnosti u poslovnim i državno-administrativnim organizacijama u mnogim zemljama širom svijeta. U Japanu, drugom po redu najvećem tržištu kompjutera u svijetu, instalirano je najviše FACOM sistema u usporedbi s drugim modelima ostalih proizvođača. Ovi moćni, pouzdani FACOM kompjuteri sposobni su za obavljanje svih mogućih poslova. Oni upravljaju satelitima u svemiru, daju prikaz atmosferskih prilika real-time grafikonima u boji, obavljaju bankovno poslovanje pomoću on-line sistema za više od 7.000 filijala i ekspozitura i još mnogo, mnogo toga.

FACOM kompjuteri su potpuno integrirani sistemi gdje se kombinacijom visoko-kvalitetne tehnologije, moćnog softwarea i već provjerenih aplikacionih programa postiže efikasnost i pouzdanost kojima nema premca.

Za dalje informacije obratite se na:

zpr **Zavod za primjenu elektroničkih računala i ekonomski inženjering**

41000 ZAGREB Savska c. 56 Telefon: 518-706, 510-760 Telex: 21689 YU ZPR FJ

FUJITSU

Fujitsu Limited•Tokyo, Japan

MIKRORAČUNALNIK IKE-1 S PROCESORJEM I 8086

A.P. ŽELEZNIKAR
D. NOVAK

UDK: 681.3 – 181.4

INSTITUT JOŽEF STEFAN, LJUBLJANA

Članek prinaša opis strukture, programiranja ter podroben načrt določene računalniške konfiguracije s 16-bitnim procesorjem nove generacije Intel 8086. Zgradba procesorja 8086 ima vrsto novosti, pa tudi ukazna zaloga procesorja vsebuje nekatere ukaze, ki so namenjeni zlasti multiprocesorskim sistemom. Opisana konfiguracija vsebuje krmilnik za prekinitve, časovnik, dva USARTa in paralelni izhodni port. Monitor konfiguracije ima le najosnovnejše direktive, ki pa omogočajo postopno razširitev monitorja. Zanimivi so tudi primeri uporabe posebnih ukazov, kot je prikazano v članku.

IKE-1 Microcomputer using 8086 Processor. This article deals with architechture, programming and with design of a computer configuration using new generation 16-bit processor Intel 8086. Processor architecture has some novelties and instruction set includes instructions dedicated to the design of multiprocessing systems. The described computer configuration is composed of interrupt controller, timer, two USARTs, and paralell output. Monitor of this configuration is very basic with only few directives enabling for their monitor development. Examples of particular instructions usage are presented.

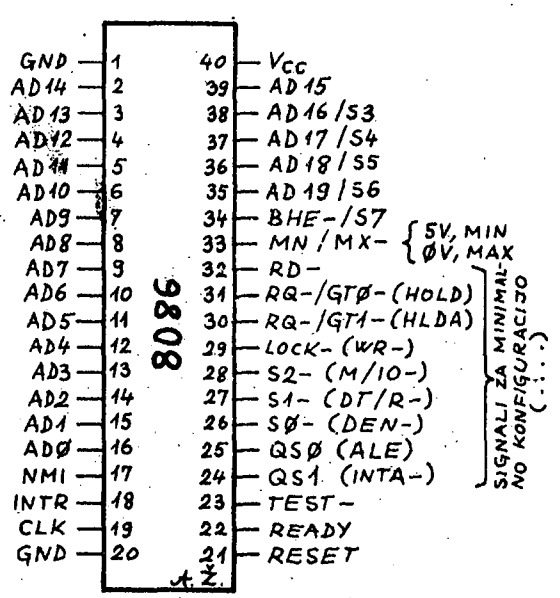
1. UVOD

Članek opisuje mikroračunalniško konfiguracijo s 16-bitnim mikro procesorjem INTEL 8086. Ta procesor je član t.i. novega mikroprocesorskega vala (prehodne generacije), ki jo sestavljata še Z-8000 in MC68000 ter drugi prihajajoči 16 bitni procesorji.

Bržkone bi bila ocena procesorja 8086, kot prvega iz svoje generacije, površna, če bi preprosto povedali, da je 8086 le nekakšno nadaljevanje njegovih 8-bitnih predhodnikov 8080A in 8085. Res je, da lahko procesor 8086 uporabi periferne člane procesorske družine 8080A, kot so npr. PIO 8255, USART 8251, PIT 8253, DMAC 8257, FDC 8271, DEU 8294 itn. na delu svojega podatkovnega vodila, res pa je tudi, da ima procesor 8086 še svoje specifične člane, kot je taktni generator in vmesnik 8284, krmilnik vodila 8288 ter registrski vmesniki 8286/8287 za demultipleksiranje vodila itn. Računalniške programe 8080A tudi ne moremo direktno izvajati na procesorju 8086; potreben je predhodni prevod koda 8080A v kod procesorja 8086.

Procesor 8086 ima svojo novo zgradbo in konfiguracijo ter uresničuje vrsto novih programskih konceptov, zlasti tistih, ki so bistveni za sistemsko programiranje in sistemsko varnost, zanesljivost in razširljivost (multiprocesorsko). Njegovi registri procesirajo 16- in 8- bitne podatke in procesor naslavlja milijon zlogov pomnilnika in 64K vhodno/izhodnih vrat.

Računalniška konfiguracija s procesorjem 8086, ki je opisana v tem članku, uporablja 2K 8- bitnih zlogov pomnilnika RAM, 4K zlogov ROMa 2616 (PROMa ali EPROMa), krmilnik prekinitev 8259, dva USATa 8251, programirljivi intervalni časovnik 8253, krmilni izhodni vmesnik 8212, vhodni/izhodni taktni generator 14411 ter V/I linije za 20 mA (navadni ASCII teleprinter), dvakrat RS-232 (video terminal), za 40 do 60 mA



Slika 1. Shema signalov na podnožju procesorskga vezja 8086

(poštni teleprinter) ter dvakratni dupleksni TTL kanal (za kasetne naprave). Vodilo je ojačeno in demultipleksirano z registri tipa 8282 in 8286 ter s krmilnikom 8288, taktnikom 8284 in ojačevalniki 8T97. Računalnik je ožičen na plošči dvojnega evropskega formata s standardnimi priključnicami.

Članek opisuje zgradbo procesorja 8086, njegovo programiranje, računalniško konfiguracijo s procesorjem 8086, shemo računalniškega monitorja ter preizkus dane konfiguracije.

2. Zgradba in delovanje procesorja I 8086

16-bitni mikro prcesor 8086 naslovi direktno 1M zlogov pomnilnika, ima 14 besednih (16-bitnih) registrov za simetrične operacije, 24 naslovnih načinov, zmore operacije nad biti, zlogi, besedami in bloki, uporablja 8- in 16-bitno aritmetiko z množenjem in deljenjem za binarna in decimalna števila, deluje s taktom do 5 MHz (ima tudi različico do 4 MHz, ki je cenejša) ter razpolaga z vodilom, ki med drugim omogoča tudi realizacijo sistema z večkratnim vodilom (multibus).

I 8086 sodi v novo generacijo visokosposobnih mikro procesorjev (HMOS tehnologija) ter lahko naslavlja pomnilnike v obliki zaporedja 8-bitnih zlogov ali 16-bitnih besed. Procesorsko vezje napaja ena sama napetost 5V in je vsebovano v standardnemu 40 nožičnemu paketu s podnožjem na sliki 1 (za maksimalno in mimimalno sistemsko konfiguracijo).

Povezava procesorja s pomnilnikom in periferijo je realizirana prek 20-bitnega časovno multipleksiranega naslovnega in podatkovnega vodila, ki omogoča manjše število nožic na integriranem vezju. To lokalno vodilo se lahko ojači ter se uporablja v sistemu v multipleksirani ali demultipleksirani obliki.

Procesorsko komplekstnost (minimalni ali maksimalni sistem) nastavimo s posebnim bitom (MN/MX-), ko generira procesor svoje lastno krmiljenje vodila. Pri kompleksnem sistemu uporabimo krmilnik vodila 8288.

Družina 8086 ima tele ožje člane:

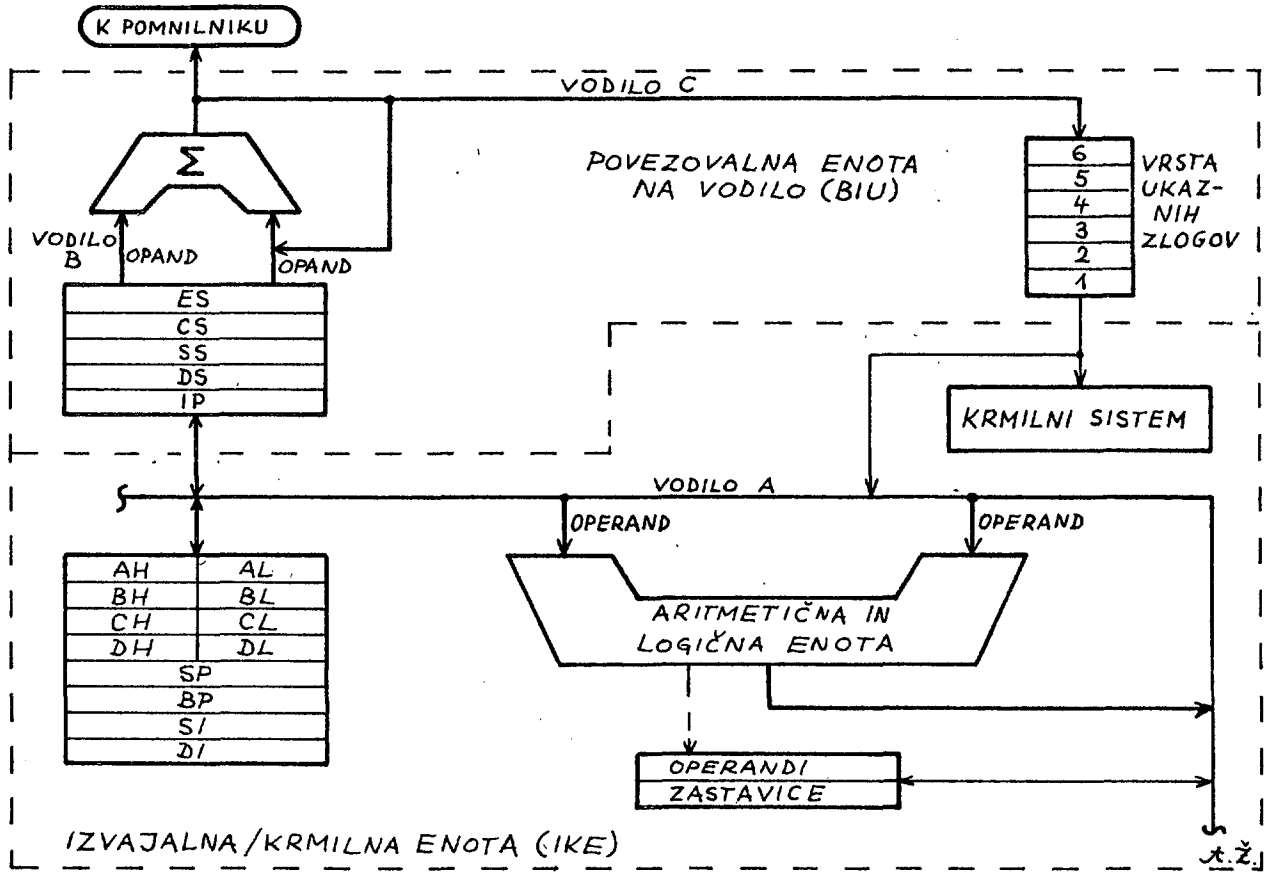
8086 : centralna procesorska enota
8284 : taktnik in ojačevalnik
8282 : 8-bitni vmesnik
8283 : 8-bitni invertirajoči vmesnik
8286 : 8-bitni dvosmerni vmesnik s tremi stanji
8287 : 8-bitni invertirajoči dvosmerni vmesnik s tremi stanji
8288 : krmilnik vodila
8259 : krmilnik prekinitev

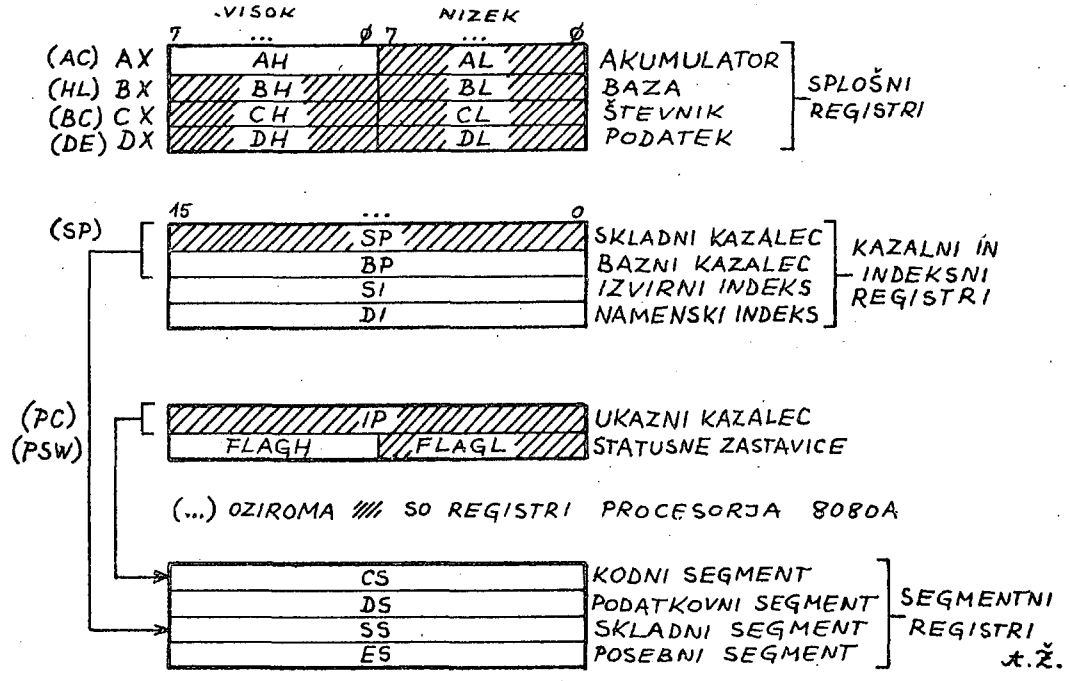
Struktura vodila procesorja 8086 omogoča uporabo perifernih integriranih vezij 8-bitnih procesorskih družin 8080A in 8085. Programe za 8080A je mogoče iz zbirnega nivoja prevesti v strojni jezik procesorja 8086, saj je množica registrov procesorja 8080A podmnožica registrske množice procesorja 8086. Pri tem pa moramo predelati prostorsko in časovno odvisne rutine, saj je potrebna sprememba njihovih shem. Tudi vektorske prekinitve se obravnavajo v sistemih s procesorjem 8086 drugače kot v sistemih s procesorjem 8080A, tako da moramo predelati tudi prekinitveno programsko opremo.

Funkcije procesorja 8086 lahko razdelimo v

- izvajalne/krmilne (IKE) in
- povezovalne (na vodilo PVE)

Slika 2. Funkcijski diagram procesorja 8086 z registri





Slika 3. Registri procesorja 8086

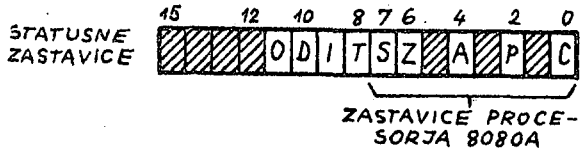
Posebnost sistema na shemi 2 je, da jemlje ukazne zloge iz pomnilnika pred njihovim izvajanjem v IKE: ti zlogi se nalagajo v vrsto (FIFO), kot kaže slika 2. IKE pobira te zloge iz vrste takoj, ko je izvedla prejšnji ukaz. Na ta način se poveča hitrost delovanja procesorja.

Procesor 8086 ima 3 skupine štirih 16-bitnih registrov ter 9 zastavic za prikazovanje statusa. 16-bitni ukazni kazalec ni direktno dostopen ter ga je moč manipulirati le z ukazi krmilnega prenosa. Registri na sliki 3 so splošni (HL skupina), kazalni/indeksni in segmentni. Splošni registri imajo visoki (H) in nizki (L) del, npr. AH in AL, vendar označuje AX 16-bitno celoto (stik AL in AH). Ostali, nesplošni registri so nedeljivi (16-bitni).

Statusne zastavice procesorja 8086 pa so tele:

AF : pomožni prenos
DF : smer
OF : prestop
SF : predznak
ZF : ničla
CF : prenos
IF : omogočitev prekinitve
PF : parnost
TF : past

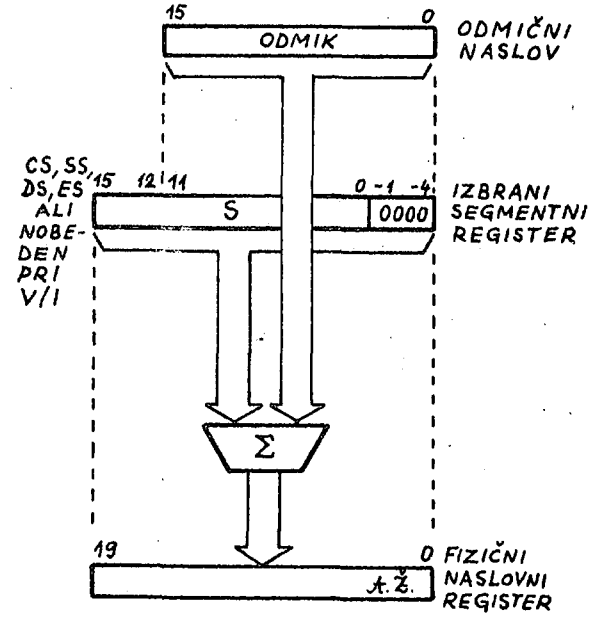
Razporeditev teh zastavic v registru je tale:



Segmentni registri CS (kodni), DS (podatkovni), SS (skladni) in ES (posebni) se uporabljajo pri izračunu pomnilniških lokacij. Npr. vsebina registra CS določa trenutni kodni segment. Lokacije ukazov so relativne glede na vsebino CS in na vsebino IP (ukazni kazalec) je odmik (pomik) od vsebine CS. Slika 4. prikazuje generiranje naslovov s pomočjo segmentnih registrov.

Pomnilniški naslovi so logično razporejeni v segmente s 64K zlogi: te segmente dodeljujemo kodu, podatkom in skladu. Meje teh segmentov se ujemajo s 16-zložnimi intervali (glej sliko 4). Segmenti se lahko prekrivajo v okviru intervalov omejitev, torej vsak segment začenja pri naslovu, ki je deljiv s 16 (najnižji 4 biti segmentnega naslova so 0). Štirje značilni segmenti se imenujejo:

- trenutni kodni segment
- trenutni podatkovni segment
- trenutni skladni segment
- trenutni posebni segment



Slika 4. Generiranje naslovov s segmentnimi registri

Kot že povedano, se segmenti lahko tudi prekrivajo. Višjih 16-bitov naslova trenutnega segmenta je shranjenih v ustreznem 16-bitnem segmentnem registru in teh 16-bitov imenujemo segmentni naslov.

Vsebina DS registra določa trenutni podatkovni segment. Vse podatkovne navedbe se smatrajo kot relativne glede na vsebino v DS; izjema so le tiste podatkovne navedbe, ki se nanašajo na BP (bazni kazalec), SP (skladni kazalec), ali DI (namenski indeksni register) v niznem ulazu. Podatkovne navedbe so lahko relativne tudi glede na ostale tri segmentne registre, če je pred ukazom t.i. enozložni prednostni prefiks.

Vsebino SS registra določa trenutni skladni segment. Vse podatkovne navedbe, ki se nanašajo na SP ali BP, so relativne k SS. To se nanaša na operaciji PUSH in POP, na CALL, prekinitve in vrnitve (iz subrutin). Podatkovne navedbe, ki se nanašajo na BP (toda ne na SP), so lahko relativne tudi k ostalim trem segmentnim registrom z uporabo posebnega enozložnega baznega prefiksa.

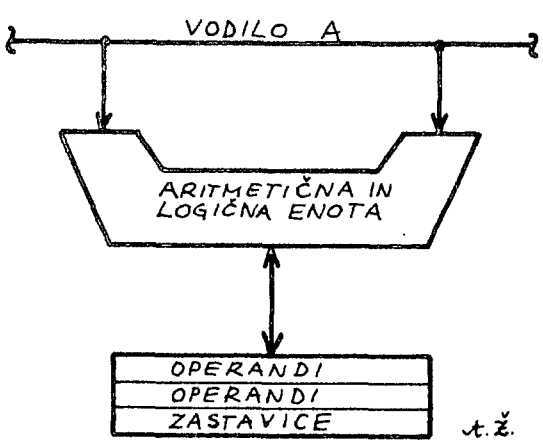
Vsebino ES registra določa trenutni posebni segment. Ta segment je navadno dodatni podatkovni segment. Podatkovne navedbe v niznih ukazih, ki uporabljano DI (namenski indeksni register), so relativne glede na ES.

Programi, ki ne nalagajo podatkov oziroma ne manipulirajo s segmentnimi registri, so dinamično premestljivi. Takšen program lahko prekinemo, ga pomaknemo v pomnilniku na novo lokacijo, ter nadaljujemo z njegovim izvajanjem pri novih vrednostih segmentnih registrov.

Aritmetična/logična enota (ALE) na sliki 5 vsebuje naslednje registre:
- dva 16-bitna operandna registra, ki sta uporabniško nedostopna.
- register zastavic z 9 statusnimi zastavicami, kot je bilo opisano.

ALE izvaja aritmetične, logične in rotirne operacije.

Procesor 8086 ima 20- linjsko naslovno vodilo za lociranje zloga ali besede (2 zaporedna zloga) v pomnilniku ali na V/I in ima tako dostop do 2 na potenco 20 zlogov (1M zlogov ali natančno 1048576 zlogov). Vsaka lokacija hrani 8 bitov. Beseda, ki jo sestavlja 16 bitov, zasede dva zaporedna zloga, katerih začetek je lahko liha ali soda lokacija. Procesor ima dva signala, in sicer BHE- in A0, s katerima selektivno izbere liho lokacijo, sodo lokacijo ali obe lokaciji. Pri naslovnih in podatkovnih operandih se nižji zlog besede shrani na nižji naslovni lokaciji in višji zlog na naslednji višji lokaciji. Pri optimalnih pogojih naj bi bil nižji zlog shranjen na sodem (pravem) naslovu.



Slika 5. Aritmetična/logična enota

Procesor 8086 ima 64K naslovljivih V/I vrat (portov). V/I se naslavlja kot en sam pomnilniški segment brez uporabe segmentnih registrov. V/I fizični naslovi imajo sicer 20 bitov, toda najvišji 4 biti so vselej enaki vrednosti nič. Porti imajo 8 ali 16 bitov in 16-bitni podatki imajo lahko sode in lihe naslove; hitrejša operacija prek portov se doseže, če je nižji zlog na sodi lokaciji. Sodo naslovljeni zlog se pojavi na vodilu D7-D0, liho naslovljeni pa na vodih D15-D8. Vsak register v 8-bitenm perifernem vezju (perifernem krmilniku) mora biti naslovljiv prek sodih naslovov, tj. prek vodov D7-D0. S signalom M/IO se preklaplja vodilo (pomnilnik ; V/I). Direktni V/I ukazi lahko naslavljajo enega ali dva zloga v območju prvih 256 V/I lokacij.

Procesor 8086 uporablja časovno multipleksirano naslovno in podatkovno vodilo. Ta metoda omogoča uporabo standardnega 40-nožičnega integriranega vezja. Demultipleksiranje se opravi z uporabo vmesnih registrov.

3. Osnove programiranja procesorja 8086

Šestnajstbitni procesorji imajo poleg izboljšane zgradbe, ki že sama prinaša nekaj specifičnosti v programiranje, še razširjen nabor ukazov. Ti so sedaj bolj kompleksni in poleg tega še operirajo z dvakrat daljšimi operandi. Novost , ki je bila opisana že v prejšnjem poglavju, je naslavljanje pomnilnika. To sedaj poleg običajne strukture vključuje še segmentne registre. Vpeljava segmentov omogoča multiprogramiranje , saj so programi dinamično premakljivi. Izvajanje takega programa je mogoče prekiniti in ga ponovno startati v drugem segmentu. Zanimiva novost je tudi predpona LOCK, ki omogoča enostavno implementacijo sinhronizacijskih mehanizmov (semaforjev) v večprocesorskem sistemu.

3.1. Načini naslavljanja

Ukazi procesorja 8086 imajo lahko do dva operanda. V večini ukazov z dvema operandoma je en operand register ali pomnilniška lokacija, drugi pa register ali konstanta. Operande v pomnilniku je mogoče naslavljati direktno ali indirektno. Pri direktnem vsebuje ukaz 16-bitni odmik, ki definira lokacijo pomnilniškega operanda znotraj segmenta. Indirektno naslavljanje pa vključuje dodatni register - bazo (BX ali BP) ali indeks (SI ali DI).

Segmenti so definirani s segmentnimi registri. Običajno je podatkovni segment definiran z DS registrom (data segment), programski z CS registrom (code segment) in skladni z SS registrom (stack segment). ES register (extra segment) običajno definira dodatni podatkovni segment. Pri indeksnem načinu naslavljanja obstajata izjemi. Če uporabljamo bazo BP, potem leži operand v skladovnem segmentu, ki je definiran z SS registrom. V primeru, ko sta uporabljena pri naslavljanju operanda baza in indeks, je segment definiran z bazo.

Opisani avtomatizem je mogoče obiti s posebno predpono (segment override prefix), s katero eksplicitno definiramo segmentni register, ki nam naj definira segment. Izjeme so naslednji primeri:
- ukazni števec (instruction pointer) je vedno odmik v segmentu, definiranim z CS registrom.
- kazalec v skladu (stack pointer) je vedno odmik v segmentu, definiranim z SS registrom.
- indirektno naslavljanje z DI registrom gre vedno preko ES registra.

3.2 Organizacija nabora ukazov

Množico ukazov procesorja 8086 lahko razdelimo na šest podmnožic:
- ukazi za prenos podatkov
- aritmetični ukazi
- ukazi za obdelavo nizov
- ukazi za prenos kontrole
- ukazi za kontrolo procesorja

Ukazi za prenos podatkov ne vplivajo na stanje zastavic (razen ukazov SAHF in POPF). Samo pri teh ukazih je lahko operand tudi segmentni register.

Aritmetični ukazi omogočajo štiri osnovne aritmetične operacije nad 8-bitnimi in 16-bitnimi operandi z ali brez predznaka.

Logični ukazi omogočajo pomikanje in rotiranje 8-bitnih ali 16-bitnih operandov za poljubno število mest. To število mora biti definirano z vsebino CL registra. Poleg tega izvršujejo logični ukazi osnovne logične operacije nad operandi (negacijo, konjunkcijo in disjunkcijo).

Ukazi za obdelavo nizov omogočajo premikanje nizov znakov ali besed po pomnilniku in primerjavo nizov. V kombinaciji z drugimi ukazi je mogoče na enostaven način realizirati kompleksne operacije nad nizi. Vsak ukaz deluje samo nad enim elementom niza (znak ali beseda). S posebno predpono REP, ki jo lahko dodamo poljubnemu ukazu iz te podmnožice, dobimo ukaz, ki deluje na cel niz. Dolžina niza mora biti seveda predhodno definirana z vsebino v CX registru:

```
REP  MOVB  PONOR,IZVOR
```

Vsi ukazi za obdelavo nizov uporabljajo register SI za naslavljanje izvirnega operanda v podatkovnem segmentu (DS) in DI register za naslavljanje ponornega operanda v posebnem segmentu (ES). Gornji ukaz bo torej preslikal niz znakov dolžine CX iz podatkovnega segmenta v posebni segment. Točna naslova pa sta definirana z SI in DI registroma. Ponavljalno predpono REP je mogoče kombinirati z LOCK predpono (segment override prefix). Zavedati pa se moramo, da se bo v primeru prekinitve kontrola vrnila samo en zlog pred ukaz in bo torej aktivna samo ena predpona.

Za kompleksnejše operacije nad nizi se običajno poslužujemo programskih zank namesto običajnega hardwarskega mehanizma. Poglejmo si na zgledu takšno kompleksnejšo operacijo:

```
        JCXZ    PRAZEN   : če je vhodni vmesnik
=                        prazen skoči na PRAZEN
NASL:   LODB    EBCBUF   :naloži v AL novi
=                        znak iz niza
        XLAT    TABELA   :pretvori v ASCII
        CMP     AL,EOT   :testiraj ali je EOT
        STOB    ASCBUF   :naloži znak v izhodni
=                        vmesnik
        LOOPNE  NASL     :če ni EOT nadaljuj
          .
          .
          .
PRAZEN:
```

Gornji program prevaja niz EBCDIC znakov v niz ASCII znakov. Prevajanje se konča z EOT znakom. Register SI je inicializiran na začetek EBCDIC niza in DI na začetek ASCII niza. BX kaže na začetek tabele za prevajanje kodov. CX vsebuje dolžino vhodnega niza.

Ukazi za prenos kontrole so razdeljeni v štiri skupine:
a) klici, brezpogojni skoki in vrnitve
b) pogojni skoki
c) kontrola ponovitev
d) prekinitve

Ukazi za kontrolo procesorja omogočajo definiranje neobičajnih režimov delovanja procesorja. Ti režimi so potrebni predvsem pri vključevanju procesorja v večje sisteme. Tako je s stališča sinhronizacije zanimiv WAIT ukaz. S tem ukazom preide procesor v stanje čakanja iz katerega ga lahko vrne zunanji signal TEST. Neobičajnejši pa je ukaz ESC. Pri tem procesor nič ne izvaja ampak samo pobira ukaze in jih daje na vodilo. Na ta način lahko drugi procesorji sprejemajo ukaze iz ukaznega niza procesorja 8086 in pri tem uporabljajo mehanizme naslavljanja procesorja 8086.

V to skupino spada tudi že prej omenjena predpona LOCK. Med izvajanjem ukaza, ki mu je dodana ta predpona, se aktivira izhodni signal LOCK. V večprocesorskih sistemih s skupnimi viri je potrebno zagotoviti vzajemno izključevanje pri uporabljanju teh virov. Osnovni mehanizem je semafor, ki ga lahko implementiramo z nedeljivo in časovno izključujočo se operacijo testiranja in dodeljevanja vrednosti skupni spremenljivki. Poglejmo si torej implementacijo take sinhronizacijske strukture:

```
TEST:  MOV     AL,1
 LOCK  XCHG    SEM,AL
       TEST    AL,AL
       JNZ     TEST
         .
         .
         .
       MOV     SEM,0
```

Ukaz za izmenjavo vsebin registra AL in pomnilniške lokacije ima predpono LOCK. Pri izvajanju tega ukaza se torej aktivira zunanji signal LOCK, ki pri ustrezni materialni povezavi, onemogoči drugim procesorjem naslavljanje spremenljivke SEM.

Pri vseh dosedanjih zgledih smo predpostavljali, da so bili segmenti že definirani. Poglejmo si zato še en kratek zgled s kompletnim programom.

```
=Program za izpis niza ASCII znakov iz
=vmesnika  dolžine 16 na naslovu 1100H
=(H-pomeni heksadecimalno številko)
=
        MOV     AX,0000H
        MOV     DS,AX     :podatkovni segment
=                         začne na naslovu 0000H
        MOV     BX,1100H  :naslov začetka
=                         vmesnika
        MOV     CX,0010H  :dolžina vmesnika
=
STR:    CALL    CHOUT     :izpis znaka
        INC     BX
        LOOP    STR
        JMP     MON       :vrnitev v monitor
=
=
=Subrutina za izpis ASCII znaka
=
CHOUT:  IN      23H       :testiranje
=                           statusa USART-a
        TEST    01H
        JZ      CHOUT
=
        MOV     AL, BX
        OUT     21H       :izpis znaka
        RET
```

Program uporablja že v monitorju definiran skladni segment. Programski segment definiramo

z direktivo za aktiviranje programa. Če se začne program na absolutnem naslovu 1000H, ga lahko aktiviramo na več načinov:

a) G 0000:1000
b) G 1000
c) G 0100:0000

Takoj za znakom za direktivo namreč definiramo vsebino CS registra. Če tega podatka ni (primer b) potem se smatra , da se začne programski segment na naslovu 0000H. Na koncu programa imamo skok (intersegmentni skok) nazaj v monitor, ki leži v drugem programskem segmentu. Novi programski segment definiramo v samem ukazu za intersegmentni skok.

4. Opis konfiguracije s procesorjem 8086

Procesor 8086 je mogoče ocenjevati z neko dovolj kompleksno konfiguracijo, ki uporablja razen ožjih članov procesorske družine 8086, ki so bili navedeni v poglavju 2, še člane procesorske družine 8080A in 8085.

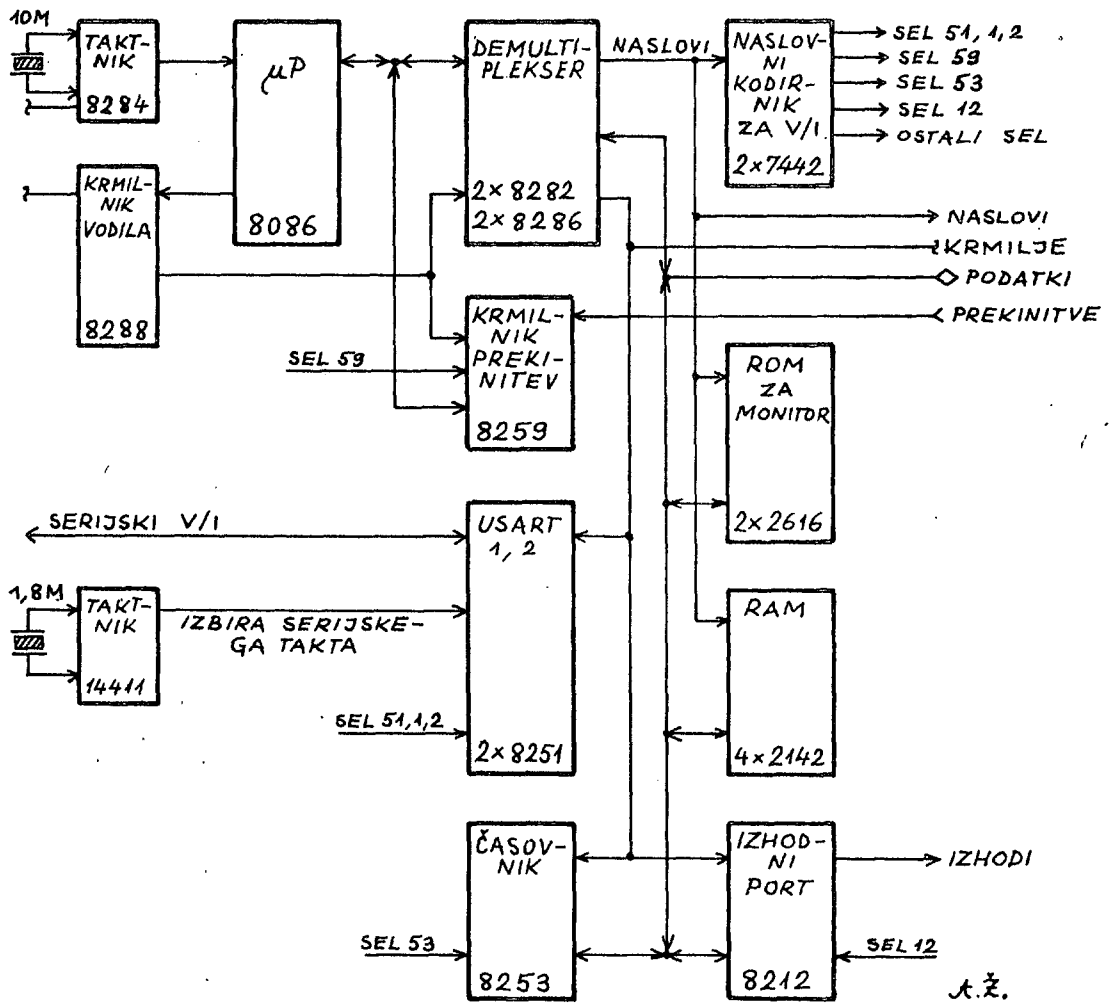
Konfiguracija na sliki 6 kaže osnovni sistem, ki ga sestavljajo:

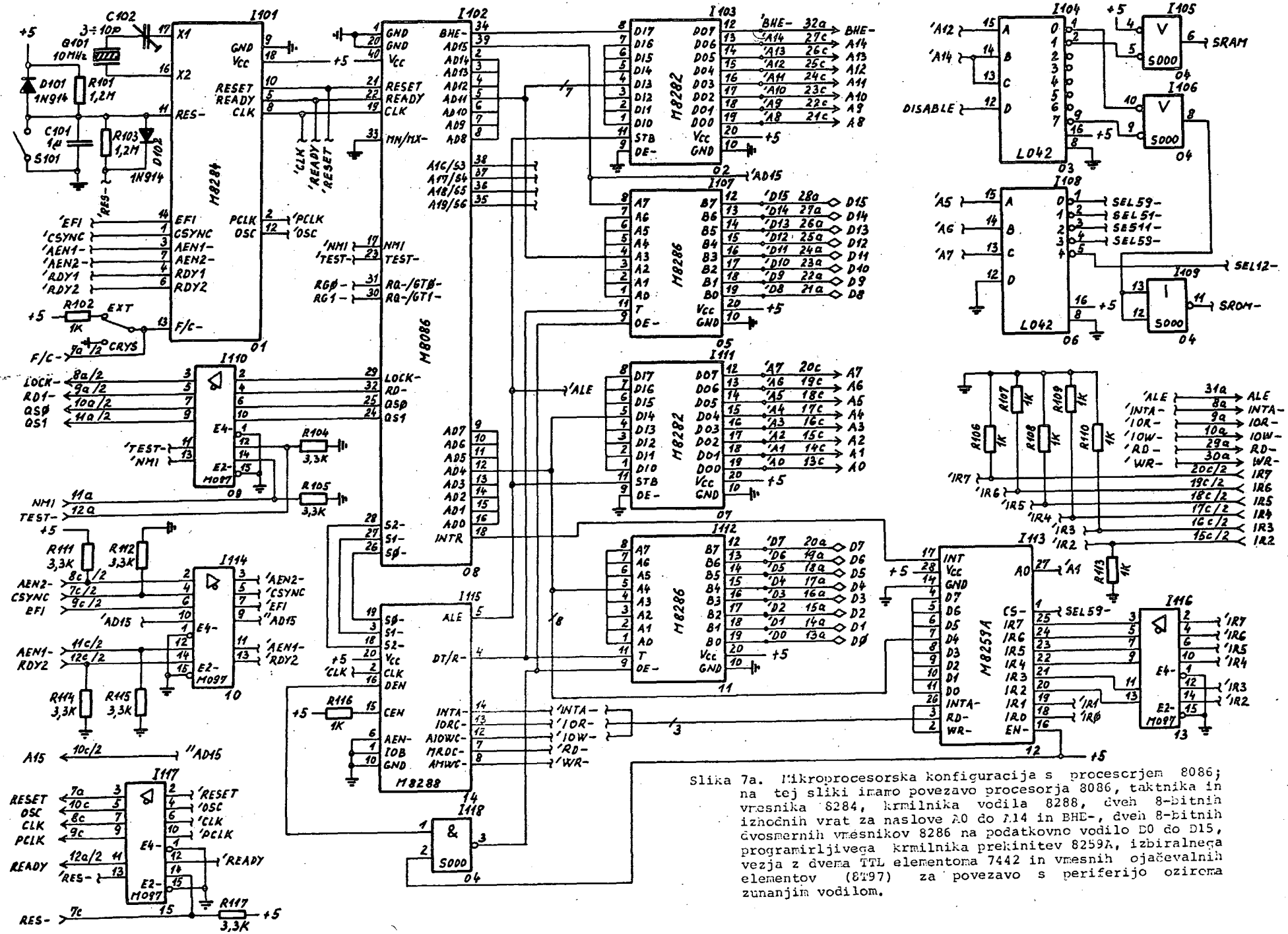
1 x 8086 : 16-bitni mikro procesor
1 x 8284 : taktnik in vmesnik za 8086;
1 x 8288 : krmilnik vodila za 8086;
2 x 8282 : 8-bitna vhodna vrata;
2 x 8286 : 8-bitni paralelni dvosmerni vmesnik za vodilo;
1 x 8259A: programirljivi krmilnik prekinitev;
2 x 7442 : izbiralno vezje 4:10;
2 x 2616 : 2K x 8-bitni PROM;
4 x 2142 : 1024 x 4-bitni statični RAM;
2 x 8251 : USART;
1 x 14411: taktni generator za USART;
1 x 8253 : programirljivi intervalni časovnik;
1 x 8212 : 8-bitna vhodna/izhodna vrata;

in še nekatera vmesna TTL in MOS vezja (ojačevalniki s tremi stanji). Na slikah 7a,b,c je prikazano celotno vezje konfiguracije na sliki 6 z vsemi podrobnostmi.

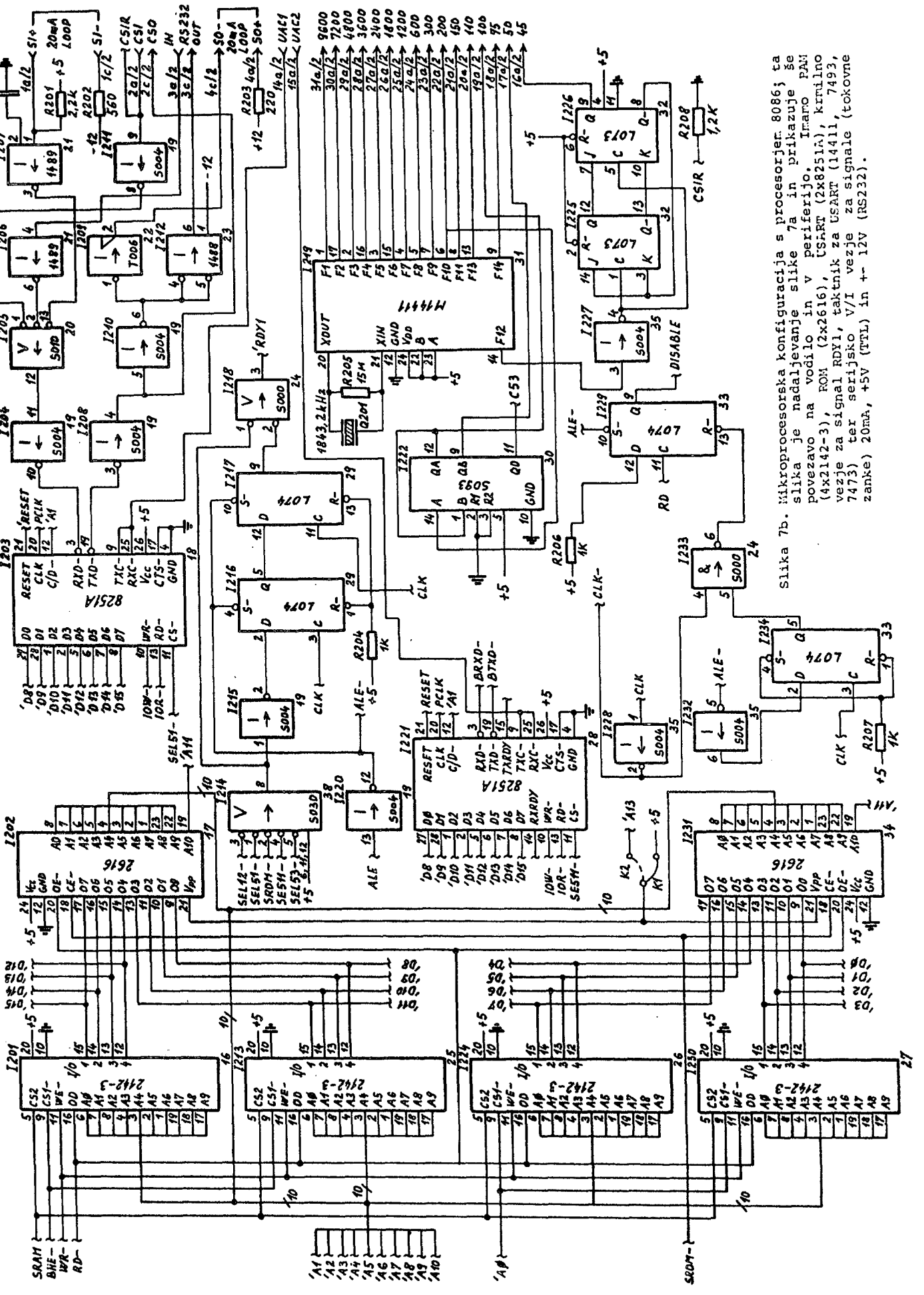
Konfiguracija ima bogato izbiro serijskih V/I kanalov, in sicer za 20 mA (ASCII teleprinter), za 40 mA (Baudotov teleprinter), za 5V nivo (TTL za pogon kasetnih naprav) in RS232 (nivo -12 do +12V). Hitrosti serijskih kanalov se nastavljajo s pretikali v razponu od 45 do 9600 Baud (hitrosti so: 45.45, 50, 75, 100, 110, 150, 200, 300, 600, 1200, 1800, 2400, 3600, 4800, 7200, in 9600 Baud). Sistem deluje z osnovnim monitorjem v ROMu (2 x 2616, tj. 4K zlogov), ima tri direktive in je opisan v naslednjem poglavju. Sistem ima 8 prednostnih prekinitev (IR0 do IR7), dve sta predvideni za povezavo v samemu mikro računalniškemu sistemu (višja prioriteta), 6 pa jih je izpeljanih na periferijo.

Slika 6. Bločni diagram konfiguracije s 16-bitnim procesorjem 8086

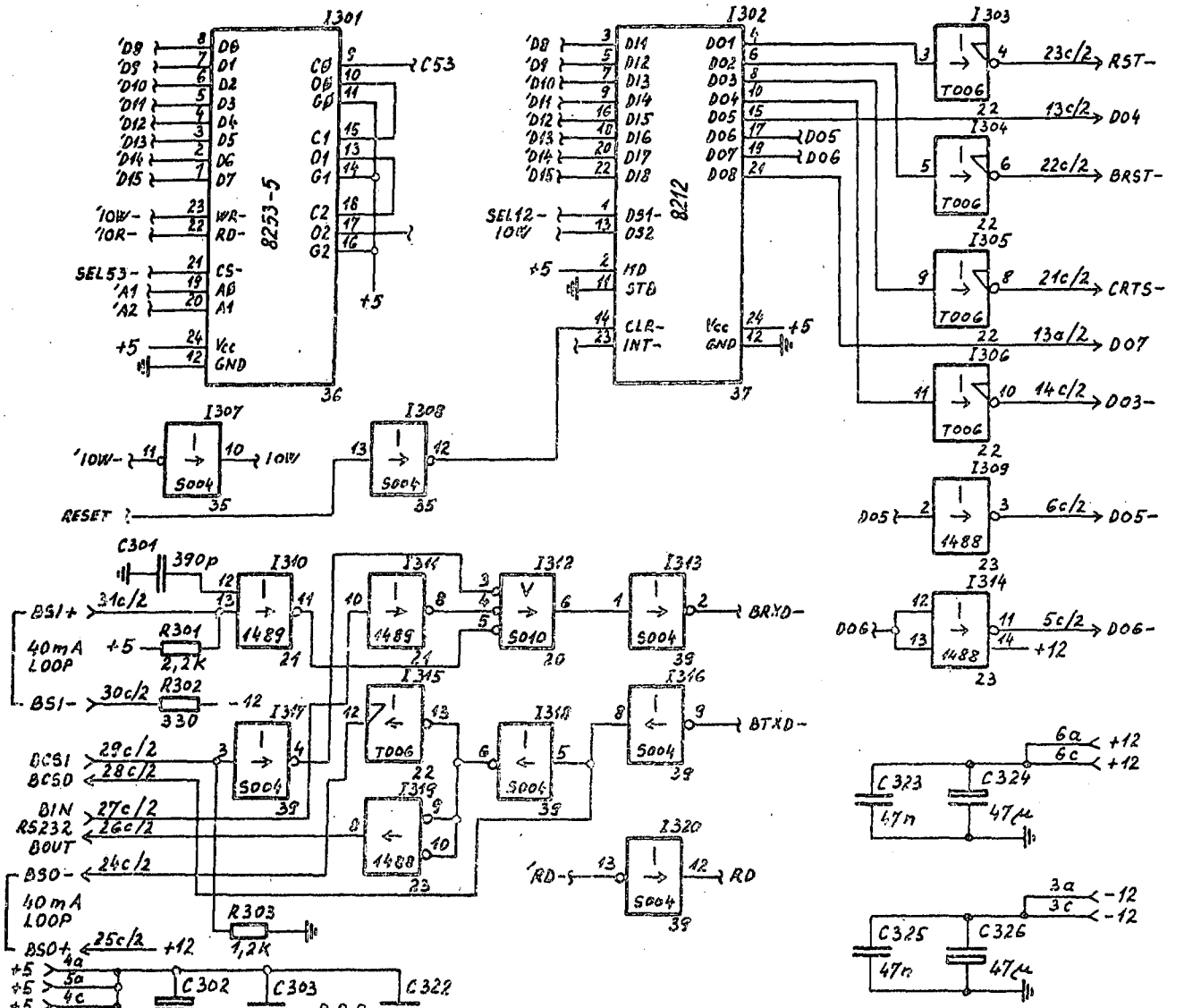




Slika 7a. Mikroprocesorska konfiguracija s procesorjem 8086; na tej sliki imamo povezavo procesorja 8086, taktnika in vmesnika 8284, krmilnika vodila 8288, dveh 8-bitnih izhodnih vrat za naslove A0 do A14 in BHE-, dveh 8-bitnih dvosmernih vmesnikov 8286 na podatkovno vodilo D0 do D15, programirljivega krmilnika prekinitev 8259A, izbiralnega vezja z dvema TTL elementoma 7442 in vmesnih ojačevalnih elementov (8T97) za povezavo s periferijo oziroma zunanjim vodilom.



Slika 7b. Mikroprocesorska konfiguracija s procesorjem 8086; ta slika je nadaljevanje slike 7a in prikazuje še povezavo na vodilo in v periferijo. Imamo PAM (4x2142-3), ROM (2x2616), USART (2x8251A), krmilno vezje za signal RDY1, taktnik za USART (14411, 7493, 7473) ter serijsko V/I vezje za signale (tokovne zanke) 20mA, +5V (TTL) in +- 12V (RS232).



slika 7c. Mikroprocesorska konfiguracija s procesorjem 8086; ta slika je nadaljevanje slik 7a in 7b ter prikazuje še nadaljno povezavo na vodilo in v periferijo. Imamo programirljivi intervalni časovnik (8253-5), izhodna vrata (8212) z ojačevalniki za različne sigalne nivoje, serijsko V/I vezje za signale 40mA (tokovna zanka za navadni teleprinter), +5V (TTL) in +- 12V (RS232). Konfiguracija lahko uporablja navadni teleprinter (npr. s 45 ali 50 Baud).

Konfiguracija obratuje z nižjim taktom (10 MHz), kot je predpisana od proizvajalca (12 MHz), tako da je mogoča uporaba standardnih integriranih vezij družine 8080A (brez sufiksov A in 5). Vezje je ožičeno na standardni, dvojni evropski kartici (mali format) in seveda preizkušeno. Ta konfiguracija je namenjena preizkušanju in ocenjevanju procesorja 8086 in jo je mogoče razširiti do maksimalnega sistema.

5. Shema monitorja za procesor 8086

Monitor DEMO-86 omogoča uporabniku vnašanje, spreminjanje in izvajanje programa v strojnem nivoju. Monitor se nahaja v ROM pomnilniku in komunicira z uporabnikom preko serijskega kanala (teleprinter, video terminal).

Monitor se nahaja v skrajnem zgornjem delu naslovljivega prostora procesorja 8086 (od FF600 do FFFFF). Javi se z znakom (.). Uporabniku so na voljo naslednje direktive:

1. Prikaz vsebine pomnilnika. Sintaksa direktive je naslednja:

D[⟨preseldek⟩][⟨začetni naslov⟩[⟨presledek⟩|⟨,⟩⟨število zlogov⟩]⟨CR⟩
⟨naslov⟩=[⟨naslov segmenta⟩⟨:⟩][⟨odmik⟩]

Monitor grupira po 16 zlogov v eno vrstivo. Na začetku vsake vrstice je naslov prevega zloga. Primer:

D FF60:00BC,25

2. Sprememba vsebine pomnilnika. Sintaksa direktive je:

S[⟨presledek⟩][⟨začetni naslov⟩]⟨CR⟩

Monitor izpiše naslov, trenutno vsebino in pomišljaj za katerim pričakuje novo vsebino. Če vsebine nočemo spremeniti odtipkamo (CR), presledek ali (,). Sicer pa definiramo novo vsebino v obliki dveh šestnajstiških številk.

Direktivo končamo z odtipkanjem poljubega znaka razen šestnajstiških znakov, (CR), presledek in (,).

3. Startanje programa. Sintaksa je naslednja:

G[<presledek>] [začetni naslov>]<CR>

Direktiva dodeli CS in IP vrednosti definirani v začetnem naslovu. Če katera vrednost ni definirana vstavi vrednost 0.

Prekinitveni vektorji kažejo v ROM pomnilnik, kjer so subrutine za izpis sporočil. Ta sporočila nas obvestijo katera prekinitvena linija je bila aktivna.

Monitor je sestavljen iz treh programskih modulov: interpretacije komand, množice prekinitvenih subrutin in tako imenovanega serijskega modula. Našteti moduli uporabljajo skupni podatkovni modul. Modul interpretacije komand vsebuje inicializacijo perifernih naprav (8259, 8251), interpretacijo vhodnih znakov in angažiranje rutin za javljanje napak. Ta modul vsebuje zunanji blok monitorja, kjer se začne izvajanje. Serijski modul je skupek podprogramov s katerimi so implementirane komandne funkcije. Podprogrami iz tega modula so zanimivi tudi za uporabnika. Grupirani so v tri dele: V/I del, uporabniški del in komandni del. V uporabniškem delu sta podprograma SIO$BLANKS za izpis specificiranega števila presledkov in SIO$CRLF za premik v novo vrstico. Iz osnovnega V/I dela pa so zanimive: DELAY za zakasnjevanje, SIOOUTCHAR za izpisovanje znaka, SIOGETCHAR za čitanje znaka in SIOOUTSTRING za izpisovanje niza znakov.

Tretji modul pa združuje prekinitvene rutine. Aktiviranjo ga prekinitve zaradi zunajih dogodkov (INTR in NMI vhoda) in prekinitve zaradi notranjih dogodkov (deljenje z vrednostjo 0, presežna vrednost (overflow)). Ustrezni podprogrami reagirajo na prekinitve z izpisovanjem sporočil na zaslon. Uporabnik mora seveda za svojo aplikacijo napisati ustrezne prekinitvene podprograme in spremeniti prekinitvene vektorje.

6. Preizkus konfiguracije

Pri preizkusu pravilnosti delovanja mikroračunalniškega sistema in pri odkrivanju napak zaradi katerih sistem sploh ne deluje, se poslužujemo kratkih tipičnih programov. Ti naj bi sprožili takšno akcijo, ki jo je mogoče opazovati. Običajno so to programske zanke, ki naslavljajo karakteristične naslove v sistemu. Z osciloskopom lahko potem ugotovimo ali se program dejansko odvija ali ne. Pri procesorju 8086 je v ta namen uporabna predpona LOCK, ki nam omogoča, da z opazovanjem signala LOCK ugotavljamo ali se program izvaja ali ne.

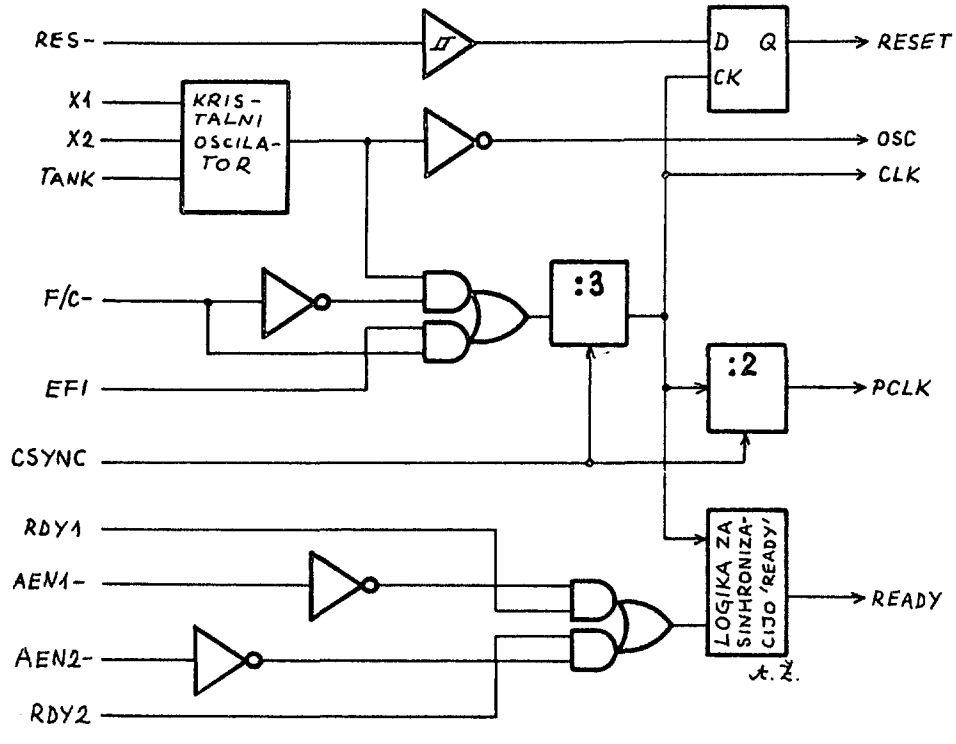
Za začetek z osciloskopom pretestiramo same signale v sistemu. Predvsem obstoj sistemske ure. Naslednji korak je ugotavljanje pravilnosti delovanja sistemskih vodil. Testiranje mora vključevati najmanjšo možno množico materialnih virov potrebnih za izvajanje programa, ker za nobenega ne vemo, če pravilno deluje ali ne. V ta test so vključeni: procesor, EPROM pomnilnik in sistemska vodila. Program, ki nam bo omogočil testiranje je naslednji:

```
LOOP:  LOCK  MOV  AL,55H
             OUT  23
             JMP  LOOP
```

Aktivira naj se s pritiskom na RESET gumb. Po aktiviranju programa opazujemo na osciloskopu signal LOCK. Če dobimo tu periodični signal pomeni, da tisti del sistema, ki ga uporablja program, pravilno deluje. Z osciloskopom lahko opzaujemo tudi podatke na podatkovnem vodilu in ugotavljamo ali se naslavlja periferno vezje z naslovom 23H .

Prehod od testiranja osnovnih sistemskih signalov do pravilnosti delovanja jedra sistema (procesor, EPROM pomnilnik, vodila) je najtežji. Pomagamo si lahko le z osciloskopom in s čimveč rutine. Najpogostejše napake so: zrcalno zamenjane linije vodila, kakšna linija v vodilu je v stiku z maso (premočno zategovanje žic pri ožičevanju), pozabljena povezava na krmilne vhode ojačevalnikov treh stanj, ipd.

Naslednji korak je testiranje perifernih



Slika 8. Zgradba taktnika in vmesnika 8284

vezij in RAM pomnilnika. To je sedaj, ko vemo, da nam del sistema pravilno deluje, že lažje. Tako npr. ugotovimo ali dela serijski kanal ali ne z naslednim programom:

```
        MOV   AL,CFH      :inicializacija USARTa
        OUT   23
        MOV   AL,05H
        OUT   23
ZAN:    MOV   AL,41H
        OUT   21
LI:     IN    23
        TEST  AL,01H
        JZ    LI
        JMP   ZAN
```

Cornji program izpisuje znak 'A' na zaslon. Sedaj pa je potrebno vključiti v program samo še delovni pomnilnik (RAM).

Po teh grobih preizkušnjah smo dali v sistem originalni monitor. Vendar se je še vedno dogajalo, da sistem občasno ni deloval. Predvsem po vklopu. Tudi to pomanjkljivost smo odpravili, ko smo kvarčni signal prestavili v neposredno bližino oscilatorskega vezja (8284).

7. Sklep

Članek obravnava zgradbo procesorja 8086, njegovo programiranje in določeno materialno konfiguracijo s tem procesorjem. Snov v članku je informativne in konkretne narave, saj so podana izdodišča in primer konfiguracije, ki nakazujejo pot za razvoj mikro računalniških sistemov s procesorjem 8086.

Procesor 8086 lahko uporablja družinske člane procesorja 8080A, če mu nekoliko znižamo taktno frekvenco (iz 4MHz na 3.33 MHz), ko uporabimo 10 MHz kristal namesto 12 MHz. Iz slike 8 je razvidna zgradba taktnika 8284, ki daje osnovno taktno in periferno taktno frekvenco, ki sta 3- in 6- krat nižji od frekvence kristala. V našem primeru imamo taktno periferno frekvenco 1.66 MHz in lahko uporabimo periferna vezaja, ki imajo mejno periferno frekvenco 2MHz. Pri frekvenci kristala 12MHz priporoča Intel uporabo perifernih vezij 8212, 8251A, 8253 ali 8253-5, 8255A ali 8255A-5, 8257 ali 8257-5, 8271, 8273, 8275, 8278, 8279 ali 8279-5, 8291, 8292, 8294 in 8295. Z uporabo READY signala, ki ga sproži naslovljena naprava, lahko imamo v pomnilnem modulu tudi 'počasna' pomnilniška vezja.

Programiranje procesorja 8086 je nekoliko drugačno, kot smo ga vajeni iz uporabe 8-bitnih procesorjev. Tu imamo več možnosti povezovanja ukazov, ko dosežemo posebne (repetitivne) učinke. Za učinkovito programiranje je seveda že v prvi fazi potreben zbirnik, ki nam avtomatično razreši nekatere dodatne probleme, ki bi jih imeli v primeru programiranja 16-bitnega procesorja 8086 v računalniškem (absolutnem) kodu.

Ta članek o možnostih uporabe procesorja 8086 in z njegovo konkretno konfiguracijo je prav gotovo bistveni prispevek k iskušnjam, ki si jih z velikimi napori pridobivamo tudi na področju 16-bitnih procesorjev. Težave so tu večkratne: ni ustrezne informacije, procesorska vezja niso na razpolago brez dodatnih naporov, vrsto lastnosti in možnosti je potrebno preizkusiti s posebnimi programi, signali in instrumentacijo, potrebno je skratka trdo delo inženirja. V časopisu Informatica bomo tudi v prihodnje obravnavali projekte, povezane z najnovejšo mikro procesorsko tehnologijo.

INFORMATICA 2/1979

POSTUPCI DETEKCIJE STANJA POTPUNOG ZASTOJA RAČUNARSKOG SISTEMA NA MODELU GRAFA SISTEMA

N. HADJINA

UDK: 681.3.02

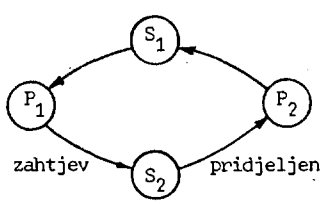
SVEUČILIŠNI RAČUNSKI CENTAR, ZAGREB

U radu je definirana situacija potpunog zastoja na modelu pridjeljivanja sredstava računskog sistema koji je definiran usmjerenim grafom, grafom sistema. Na tom modelu su istraženi nužni i dovoljni uvjeti za postojanje potpunog zastoja, te su posebno razradjeni neki specijalni slučajevi pridjeljivanja sredstava računskog sistema. Na osnovu provedenih istraživanja dani su postupci za detekciju stanja potpunog zastoja.

ALGORITHMS FOR DEADLOCK DETECTION IN COMPUTER SYSTEM BY SYSTEM GRAPH: Deadlock in system graph, computer resource allocation model, defined by directed graphs, is presented. On that model necessary and sufficient conditions for deadlock state are given, and special situations in computer resource allocations are analysed. Based on performed investigation some algorithms for deadlock detection are given.

1. UVOD

Prilikom uvodjenja paralelizma u korištenju računarskog sistema od strane više programa ili korisnika istovremeno pojavljuje se problem sigurnog i efikasnog pridjeljivanja sredstava računarskog sistema. Paralelizam u obradi procesa (programa, transakcija) može dovesti do situacije potpunog zastoja koja nastupa kada su procesi tako blokirani da se zahtjev za nekim sredstvom ne može nikada zadovoljiti bez drastične akcije u računarskom sistemu (uklanjanje procesa, prisilno oslobadjanje sredstava i dr.). Tipičan primjer potpunog zastoja nastaje kada npr. proces P_1 zahtijeva sredstvo S_2 koje je pridjeljeno procesu P_2, a taj zahtijeva sredstvo S_1 koje je pridjeljeno procesu P_1 kao na slici 1.



Slika 1. Potpuni zastoj

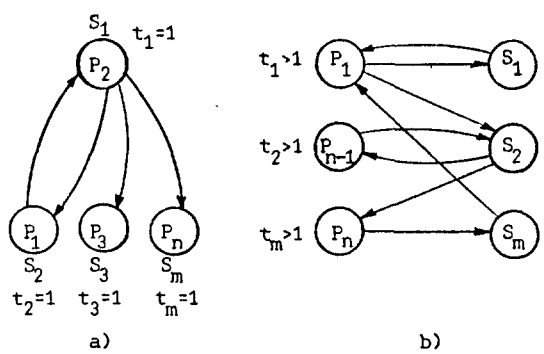
Ponašanje računarskog sistema za slučaj višestrukog korištenja sistema od strane više paralelnih procesa može se analizirati na modelu grafa.

2. GRAF SISTEMA

Model pridjeljivanja sredstava računarskog sistema može se definirati usmjerenim grafovima i to za pridjeljivanje raznovrsnih sredstava gdje je broj raspoloživih jedinica sredstava jednak 1, kao na slici 2a, te za pridjeljivanje sredstava gdje je broj raspoloživih jedinica veći od 1, kao na slici 2b.

Graf sistema definiran je, dakle, usmjerenim grafom sa slijedećom interpretacijom i ograničenjima.

Model sa slike 2a predstavljen je usmjerenim grafom $G(P,s)$ gdje je sa P označen skup procesa koji predstavljaju čvorove grafa, a sa s je označena relacija preslikavanja P u P. Relacija preslikavanja odredjena je *slijedećom operacijom: strelica usmjerena od* P_i *prema* P_j znači da proces P_i zahtijeva sredstvo S_1 koje je pridjeljeno procesu P_j (P_i čeka na oslobadjanje sredstva S_1 koje drži P_j).



Slika 2. Graf sistema

Model sa slike 2b predstavljen je usmjerenim bipartitnim grafom $G_B(\Pi,s)$ sa slijedećim svojstvima:

- Skup čvorova usmjerenog grafa Π podijeljen je u dva medjusobno odvojena podskupa, podskup čvorova procesa $P=\{P_1,P_2...P_n\}$ i podskup čvorova sredstava $S=\{S_1,S_2...S_m\}$
- Graf je bipartitan u odnosu na P i S. Svaka strelica je usmjerena izmedju čvorova P i čvorova S. Ako je strelica $s=(P_i,S_j)$, tada se s naziva strelica zahtjeva, tj. proces P_i zahtijeva jednu jedinicu od S_j. Ako je $s=(S_j,P_i)$, tada se s naziva strelicom pridjeljivanja, što pokazuje da je jedna jedinica od sredstava S_j pridjeljena procesu P_i.
- Za svako sredstvo $S_i \in S$ postoji nenegativan cijeli broj t_i koji označava broj raspoloživih jedinica

sredstva S_i.

- Da bi se stanje sistema pridjeljivanja moglo realizirati, model mora zadovoljiti slijedeća ograničenja, uz oznaku $|(a,b)|$ koja predstavlja broj strelica grafa usmjerenih od čvora a prema čvoru b:

  a) Za sredstvo S_i ne smije postojati više pridjeljivanja od t_i, tj.

  $$\Sigma_j \, |(S_i,P_j)| \leq t_i \qquad \forall\, i$$

  b) Suma zahtjeva i pridjeljivanja za bilo koji proces za pojedino sredstvo ne može prijeći broj raspoloživih jedinica, tj.

  $$|(S_i,P_j)| + |(P_j,S_i)| \leq t_i \qquad \forall\, i\ i\ j$$

Stanje sistema se mijenja samo kao rezultat operacija zahtjeva, oslobadjanja i pridjeljivanja sredstava.

Zahtjev

Ako je sistem u stanju S, a proces P_i nema zahtjeva, tada P_i može zahtijevati bilo koji broj jedinica sredstva uz gore navedeno ograničenje. Tada sistem prelazi u stanje T iz stanja S izvodjenjem procesa P_i ($S \overset{i}{\rightarrow} T$).

Pridjeljivanje

Stanje sistema S se može promijeniti ovom operacijom ($S \overset{i}{\rightarrow} T$) samo ako proces P_i ima otvorenih zahtjeva koji se svi mogu zadovoljiti, tj.

$$|(P_i,S_j)| + \Sigma_k |(S_j,P_k)| \leq t_j \ .$$

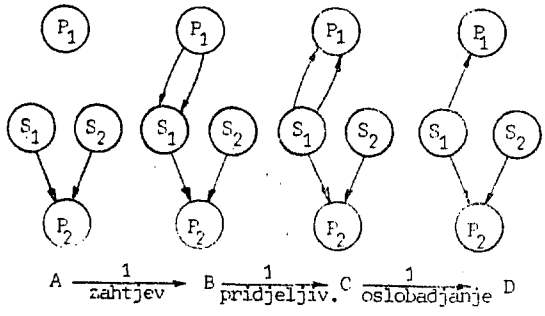
Graf je identičan grafu od zahtjeva, jedino su strelice reverzirane (od S_j na P_i) a to označava pridjeljivanje.

Oslobadjanje

Proces P_i može promijeniti stanje od S na T operacijom oslobadjanja sredstava onda i samo onda ako proces P_i nema otvorenih zahtjeva (strelica od P_i na S_j) za sredstvom. Ekvivalentna operacija na grafu je operacija uklanjanja strelice pridjeljivanja.

Navedene operacije prikazane su na modelu sa slike 3 za $P=\{P_1,P_2\}$, $S=\{S_1,S_2\}$, $t_1=3$ i $t_2=2$.

Grafovi sistema sa slike 3 predstavljaju modele za stanja sistema A, B, C i D koja nastaju izvodjenjem procesa P_1.



Slika 3. Operacije na grafu sistema

3. ODREDJIVANJE NUŽNIH I DOVOLJNIH UVJETA ZA POSTOJANJE ZASTOJA NA GRAFU SISTEMA

Da bi se došlo do tih uvjeta potrebno je uvesti još neke dodatne definicije vezane uz model grafa sistema:

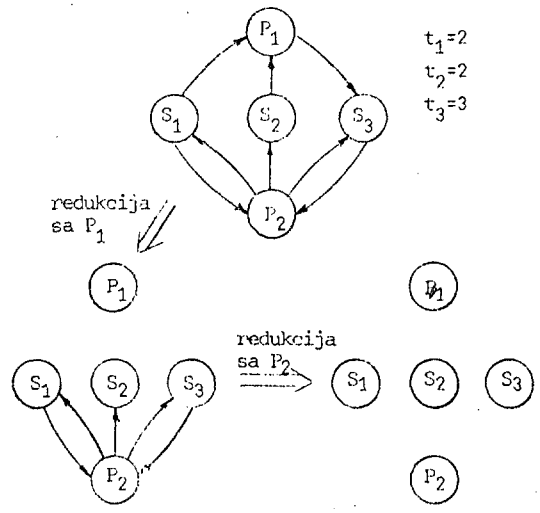
* Graf sistema se reducira procesom P_i koji nije niti blokiran niti izolirani čvor grafa, uklanjanjem svih strelica od ili prema P_i. P_i tada postaje izolirani čvor.
* Graf sistema se naziva nereducibilnim ako se ne može reducirati niti sa jednim procesom.
* Graf sistema se naziva potpuno reducibilnim ako postoji sekvencija redukcija koja briše sve strelice grafa. Slika 4 prikazuje potpuno reducibilan graf sistema, budući da graf prelazi iz stanja A u stanje B ($A \overset{*}{\rightarrow} B$) bez strelice.

Iz slike 4 se vidi da je P_2 blokiran u stanju A na sredstvu S_1, ali postaje neblokiran nakon redukcije sa P_1. Redosljed redukcije nije važan jer dovodi do istog nereducibilnog grafa, što je pokazano u slijedećoj lemi.

Lema 1

Sve sekvencije redukcije grafa sistema dovode do istog nereducibilnog grafa.

Dokaz leme dan je u [1]. Uz pomoć leme 1 moguće je izvesti nužne i dovoljne uvjete za zastoj.



Slika 4. Potpuno reducibilni graf sistema

Teorem 1

S je stanje potpunog zastoja onda i samo onda ako graf sistema nije potpuno reducibilan.

Struktura grafa sistema daje mogućnost za odredjivanje samo nužnih uvjeta ali ne i dovoljnih za potpuni zastoj. Tako je nužan uvjet za potpuni zastoj, definiran strukturom grafa, dan slijedećim teoremom.

Teorem 2

Ciklus (A.2) u grafu sistema je nužan uvjet za postojanje potpunog zastoja.

Teorem 1 i teorem 2 predstavljaju najosnovnije tvrdnje koje se mogu koristiti za detekciju zastoja u slučajevima kada ne postoje nikakva ograničenja na sistem procesa i sredstava. Detekcija zastoja je efikasna ako se ona provodi kontinuirano. Podlogu za takvu detekciju daje slijedeći teorem.

Teorem 3

Ako S nije stanje potpunog zastoja, i ako postoji $S \overset{i}{\vdash} T$, tada je T stanje potpunog zastoja onda i samo onda ako je operacija procesa P_i operacija zahtjeva i ako je P_i u potpunom zastoju u T.

Dokazi teorema 1, 2 i 3 mogu se naći u [1].

Teorem 3 pokazuje da zastoj može izazvati zahtjev koji se ne može odmah zadovoljiti. To znači da je nakon svakog postavljanja zahtjeva od strane procesa P_i potrebno provesti redukciju grafa sa procesom P_i. Ukoliko je ona moguća, nisu potrebne daljnje redukcije grafa sistema. Na osnovi gornjih tvrdnji, tj. iz strukture grafa sistema, moguće je odrediti nužne ali ne i dovoljne uvjete za postojanje zastoja. Da bi se odredili dovoljni uvjeti za zastoj, potrebno je razmatrati ograničenja i strukture sistema procesa i sredstava. To dovodi do razmatranja nekih specijalnih slučajeva koji su vrlo česta pojava u postupcima pridjeljivanja sredstava računarskog sistema.

NEPOSREDNO PRIDJELJIVANJE SREDSTAVA

Ukoliko računarski sistem odmah zadovoljava sve zahtjeve za sredstvima koji se daju realizirati, tada postoji jednostavan uvjet koji je dovoljan za zastoj. Stanje sistema u kojem su svi procesi koji imaju zahtjeve blokirani može se nazvati sebičnim stanjem.

Teorem 4

Ako je sistem u sebičnom stanju, tada je gnijezdo u grafu sistema dovoljan uvjet za potpuni zastoj.

Dokaz

Uz pretpostavku da graf sadrži gnijezdo (A.3) K, tada svi procesi u gnijezdu moraju biti blokirani na sredstvima u gnijezdu, budući da po definiciji ne postoji strelica koja izlazi iz gnijezda. Slično, sva pridjeljena sredstva u K su zadržana od procesa u K iz istog razloga. Konačno, svi procesi u K moraju biti blokirani zbog sebičnog stanja i definicije gnijezda. Prema tome, svi procesi u K moraju biti u potpunom zastoju.

PRIDJELJIVANJE SREDSTAVA SA PO JEDNOM JEDINICOM

Uz pretpostavku da svako sredstvo ima jednu jedinicu, tj. $t_i=1$, $i=1,\ldots,m$, tada ciklus postaje dovoljan uvjet za zastoj.

Teorem 5

Graf sistema, sa sredstvima koja sadrže po jednu jedinicu, je u stanju potpunog zastoja onda i samo onda ako sadrži ciklus.

Dokaz

Teoremom 2 je utvrdjena nužnost postojanja ciklusa za potpuni zastoj. Za dokaz dovoljnog uvjeta, uz pretpostavku da graf sistema sadrži ciklus, potrebno je razmatrati samo procese i sredstva na ciklusu. Budući da svaki čvor procesa mora sadržavati ulaznu i izlaznu strelicu, on mora imati nezadovoljeni zahtjev za sredstvom u ciklusu, te mora držati neka sredstva u ciklusu. Slično, svako sredstvo (sa po jednom strelicom) u ciklusu mora biti pridjeljeno nekom procesu u ciklusu. Prema tome, svaki proces u ciklusu je blokiran na sredstvu u ciklusu, koje može biti oslobodjeno jedino procesom u ciklusu. Dakle su procesi iz ciklusa u potpunom zastoju.

ZAHTJEVI ZA PO JEDNU JEDINICU SREDSTVA

U ovom slučaju proces smije zahtijevati samo po jednu jedinicu sredstva u nekom trenutku vremena. To znači da će u grafu sistema najviše jedna strelica zahtjeva biti vezana na čvor procesa. Tada za sebična stanja, gnijezdo u grafu postaje dovoljan uvjet za zastoj.

Teorem 6

Graf sistema sa zahtjevima za po jednu jedinicu sredstva u sebičnom stanju je u potpunom zastoju samo ako sadrži gnijezdo. Dokaz je dan u [1].

4. POSTUPCI ZA DETEKCIJU STANJA POTPUNOG ZASTOJA

Na osnovi do sada istraženih nužnih i dovoljnih uvjeta, detekcija stanja potpunog zastoja bit će provedena na modelu grafa sistema i to na tri načina:

- redukcijom grafa sistema;
- detekcijom ciklusa u grafu sistema i
- detekcijom gnijezda u grafu sistema.

REDUKCIJA GRAFA SISTEMA

Teorem 1 se direktno može koristiti za postupak detekcije. Potrebno je efikasno provesti postupak redukcije grafa i ako graf nije potpuno reducibilan, tada je izvorno stanje, stanje potpunog zastoja. Lema 1 dozvoljava da se redukcija provede proizvoljnim redom. Graf sistema se može predstaviti matricama ili listama, a svakoj strelici u grafu se može pridijeliti težina koja označava broj jedinica koje su tražene ili pridjeljene procesu.

Matrična reprezentacija

Graf sistema se može predstaviti sa dvije matrice dimenzije $n \times m$.

a) *Matrica pridjeljivanja* $\mathbf{A}$ čiji elementi a_{ij}, $i=1,\ldots,n$; $j=1,\ldots,m$, daju broj jedinica sredstva S_j koja su pridjeljena procesu P_i, tj.

$$a_{ij} = |(S_j,P_i)|$$

b) *Matrica zahtjeva* $\mathbf{B}$ čiji elementi b_{ij} predstavljaju broj jedinica sredstva S_j koje traži proces P_i, tj.

$$b_{ij} = |(P_i,S_j)|.$$

Struktura lista

Sredstva pridjeljena bilo kom procesu P_i su vezana na P_i,

$$P_i \rightarrow (S_x,a_x) \rightarrow (S_y,a_y) \rightarrow (S_z,a_z)$$

gdje je $a_j = |(S_j,P_i)|$.

Na sličan način se može konstruirati lista zahtjeva procesa P_i za sredstvima. Adekvatne liste se mogu upotrijebiti i za sredstva. Pridjeljivanje sredstava formira listu

$$S_i \rightarrow (P_u,b_u) \rightarrow (P_v,b_v) \rightarrow \ldots \rightarrow (P_w,b_w)$$

gdje je $b_j = |(P_j,S_i)|$.

Slična lista sadrži zahtjev za sredstvima.

Za obje reprezentacije je pogodno održavati vektor raspoloživih jedinica $(r_1,\ldots,r_m)$, gdje r_i odgovara broju raspoloživih jedinica sredstava S_i, tj.

$$r_i = t_i - \sum_k |(S_i, P_k)|.$$

Direktni postupak koji se može realizirati je da se iterativno obradjuju matrice (liste) zahtjeva procesa, provodeći redukcije grafa, gdje je to moguće, sve do nereducibilne situacije. Najlošija situacija nastupa kad su procesi poredani od $P_1 \ldots P_n$, a jedino moguća sekvencija redukcije je $P_n \ldots P_1$. Broj ispitivanja procesa je $n+(n-1)+\ldots+1=n(n+1)/2$. Ako svaki proces zahtijeva m sredstava, tada je za najlošiji slučaj vrijeme obavljanja postupka proporcionalno sa mn^2.

Brži postupak se dobiva ukoliko postoji dodatna informacija o zahtjevima. Za svaki čvor procesa definira se broj čekanja w_i, koji se sastoji od broja sredstava (ne jedinica sredstava) koja prouzrokuju da proces bude blokiran u bilo kojem vremenu. Takodjer treba održavati listu zahtjeva za sredstvima poredanih po veličini.

DETEKCIJA CIKLUSA U GRAFU SISTEMA

Detekcija ciklusa je postupak kojim se detektira zastoj na osnovi teorema 5 za sredstva sa po jednom jedinicom.

Ovdje će biti izložena dva načina detekcije:
- preko matrice povezanosti grafa i
- redukcijom čvora s izlaznim stupnjem jednakim nuli.

DETEKCIJA PREKO MATRICE POVEZANOSTI GRAFA

Grafu sistema za model sa slike 2a može se pridružiti matrica povezanosti $A=[a_{ij}]$ (A.4). Matrica $A^h=[a_{ij}]^h$ daje sve moguće puteve od procesa P_i do procesa P_j dužine h. Element matrice $[a_{ij}]^h$ daje podatak da se proces P_i nalazi na ciklusu dužine h. Na taj način može se jednostavnim potenciranjem matrice A i ispitivanjem dijagonalnih elemenata utvrditi da li postoji ciklus i koji su procesi u njemu uključeni.

Teorem 6

Sistem je u stanju potpunog zastoja onda i samo onda ako je v broj dijagonalnih elemenata matrice A^h različitih od nule, veći ili jednak od h ($v \geq h$). Za slučaj v=h u sistemu postoji jednostavan zastoj (jedan ciklus), a za v>h u sistemu postoji višestruki zastoj (više ciklusa). Ako je v<h u sistemu ne postoji zastoj.

Dokaz

Dokaz tvrdnje za v=h slijedi direktno iz definicije (A.5) da je A^h matrica svih puteva dužine h koji se mogu realizirati od čvora i do čvora j ako je $[a_{ij}]^h \neq 0$. v=h znači da na ciklusu dužine h leži upravo h procesa, što govori o postojanju samo jednog ciklusa, dakle, jednostavnog zastoja.

Dokaz tvrdnje za v>h proizlazi iz činjenice da elementi $[a_{ij}]^h$ različiti od nule odredjuju procese koji leže na ciklusu dužine h. Pošto je v>h, a ciklus mora biti dužine h, slijedi da postoje barem dva ciklusa dužine h od kojih barem jedan sadrži v=h procesa koji nisu uključeni u prvi ciklus. Situacija sa više ciklusa u sistemu naziva se višestruki zastoj.

Dokaz tvrdnje za v<h proizlazi direktno iz teorema 5 po kojem za zastoj mora postojati ciklus, a taj može postojati iz definicije ciklusa samo za v≥h, a ne za v<h.

REDUKCIJA ČVORA S IZLAZNIM STUPNJEM JEDNAKIM NULI

Neka čvor ponora bude svaki čvor čiji je izlazni stupanj jednak nuli. Algoritam za detekciju ciklusa sastoji se u uzastopnom brisanju strelica, koje ulaze u čvor ponora. Ako razmatrajući graf sadrži samo čvorove ponora, tada ne postoji ciklus u originalnom grafu, jer po definiciji ciklusa čvorovi u ciklusu ne mogu biti čvorovi ponora.

Za kontinuiranu detekciju može se upotrijebiti teorem 3, tj. treba testirati da li se proces P_i sa zahtjevom pojavljuje na ciklusu u grafu sistema. To se provodi ispitivanjem svih puteva koji počinju sa P_i na postojanje ponovne pojave čvora procesa P_i.

DETEKCIJA GNIJEZDA U GRAFU SISTEMA

Detekciju gnijezda u grafu moguće je provesti na jednostavniji način nego detekciju ciklusa. Praćenje izlaznog stupnja čvora nije neophodno za svaki čvor koji je vezan na čvor ponora, jer bez obzira na broj strelica koje izlaze iz nekog čvora, taj čvor ne može biti dio gnijezda. U slučaju kontinuirane detekcije koristi se teorem 3 kojim se odredjuje da li se proces P_i sa zahtjevom pojavljuje u gnijezdu grafa ili ne.

5. ZAKLJUČAK

Postupci detekcije stanja potpunog zastoja u biti su složeni i zahtijevaju znatan utrošak sistemskog vremena. Stoga je potrebno za svaki slučaj posebno odabrati način i vrijeme detekcije zastoja. Takodjer je potrebno, koliko je to moguće spriječiti pojavu potpunog zastoja, a tek ako to nije moguće potrebno je provesti neki od postupaka detekcije, a nakon toga i postupak oporavka iz stanja potpunog zastoja.

DODATAK A.

Krug je put $s_1, s_2, \ldots, s_q$ u kojem se početni čvor od s_1 i završni čvor s_q prekrivaju. (A.1)

Elementarni krugovi su oni koji ne koriste isti čvor više nego jednom (izuzev za početni i završni). Elementarni krug naziv se kraće *ciklusom*. (A.2)

Gnijezdo u usmjerenom grafu G(N,s) je podskup čvorova $M \subseteq N$ za koje vrijedi $a \in M$, $s(a) \in M$. (A.3)

Za graf G(N,s) moguće je izraditi *matricu povezanosti* $A=[a_{ij}]$ koja ga opisuje.

Elementi matrice A se dobivaju kako slijedi:

$a_{ij} = 1$ ako postoji strelica (n_i, n_j) u G

$a_{ij} = 0$ ako ne postoji strelica (n_i, n_j) u G.

Skup stupaca koji imaju 1 u retku n_i predstavljaju skup $s(n_i)$, a skup redaka koji imaju 1 u stupcu n_i predstavljaju skup $s^{-1}(n_i)$. (A.4)

Neka je A matrica povezanosti grafa G i neka je matrica $Y=A^h$, tada je y_{ij} ukupan broj različitih sekvenci $(n_i, \ldots), \ldots, (\ldots, n_j)$ koje imaju dužinu h, što odgovara putevima u G. Ako je $A^h=0$ za neki $h \leq n$ tada je graf G aciklički. Dokaz gornje tvrdnje može se naći u [2]. (A.5)

LITERATURA

1. N. Hadjina. *Postupci pridjeljivanja u višestruko korištenim računarskim sistemima.* Doktorska disertacija, Zagreb, 1979.
2. Berztiss, A.T. *Data Structure: Theory and Practice.* Academic Press, London, 1975.

MODULA – PROGRAMSKI JEZIK PRIHODNOSTI ZA MIKRORAČUNALNIŠKE APLIKACIJE?

D. NOVAK
M. EXEL
M. KOVAČEVIĆ
B. KASTELIC

UDK: 681.3 – 519.688

INSTITUT JOŽEF STEFAN, LJUBLJANA

Članek opisuje glavne značilnosti programskega jezika Modula. Poudarek je na uporabi Module pri programiranju programskih krmilnikov perifernih naprav. Z nekaj zgledi je ilustrirana enostavnost in preglednost modulov naprav (device module). Opisani so tudi elementi jezika za multiprogramiranje in mehanizem razvrščanja procesov.

MODULA - PROGRAMMING LANGUAGE OF THE FUTURE FOR MICROCOMPUTER APPLICATIONS ? The paper describes the main features of the programming language Modula. The use of Modula in device drivers programming for peripheral devices is emphasized. The simplicity and clearness of device module construct is illustrated by few examples. The multiprogramming facilities of the language as well as the scheduling mechanism are also described.

1. Uvod

Modula je relativno mlad programski jezik, saj ga je izoblikoval in definiral Wirth šele leta 1977 (1,2,3). Namenjen je predvsem za programiranje specializiranih računalniških sistemov, kot so na primer sprotni sistemi (real-time) za kontrolo industrijskih in laboratorijskih procesov. Zelo primeren je za programiranje krmilnih programov za vhodno-izhodne (V/I) naprave. Na tem področju naj bi zamenjal zbirni jezik. Že sam avtor je napovedal (1), da bo jezik pridobival na veljavi z razvojem mikroprocesorske tehnologije in z njim povezanim znižanjem cen mikroprocesorjev. Zaenkrat se te napovedi še niso uresničile. Za mikroračunalnike je značilno, da so običajno revni s programsko opremo in da je ta zelo težko prenosljiva iz sistema na sistem. Ta programska oprema je namreč napisana v zbirnem jeziku in je zaradi tega često slabo strukturirana in težko razumljiva. Specifičnosti sistema niso strnjene v zaključene enote (module) ampak so bolj ali manj razkropljene, kar predstavlja osnovni problem pri modificiranju obstoječe programske opreme. Ravno tu lahko pridejo do izraza prednosti jezika Module, ki omogoča združevanje od konfiguracije sistema odvisnih predmetov v kompaktne celote - module. Z moduli, ki so enote dosega (scope), te predmete ogradimo od okolice. Predmete modula je možno dosegati iz okolice samo na eksplicitno kontroliran način.

Nizka cena mikroprocesorjev je verjetno eden od glavnih razlogov, da so postali multiprocesorski sistemi vsakdanja praksa in da niso več atrakcija raziskovalnih laboratorijev. Če so bili osembitni procesorji še precej nedorasli za takšne aplikacije, pa tega ne moremo trditi za nove šestnajstbitne procesorje (8086, 68000). Že sam nabor ukazov in arhitektura procesorjev pričata, da so razvijalci imeli v mislih združevanje večjega števila procesorjev v kompleksnejše in zmogljivejše sisteme. Tudi za nekatere od takih sistemov je Modula dovolj močno orodje, saj vsebuje multiprogramske pripomočke kot so: signali (sinhronizacija), procesi (sočasno izvajanje) in vmesniški moduli (izključevanje sočasnih dostopov do skupnih oz.deljenih predmetov).

V članku bomo poskušali osvetliti bistvene značilnosti Module in podati hardwarske zahteve, ki morajo biti izpolnjene za uspešno implementacijo jezika. Posebej se bomo osredotočili na programske krmilnike (driver), kjer bomo na zgledih uporabili koncept modula naprave (device module). Za konec pa si bomo pogledali še problematiko vezano na multiprogramiranje, t.j. razvrščanje, vzajemno izključevanje in zaščito.

2. Glavne značilnosti Module

Modula je majhen koncizen jezik z močnim vplivom Pascala. Zasnova je taka, da naj bi omogočala zelo učinkovito implementacijo. Jezik ne pozna dinamičnih tabel (array), procedurni mehanizem za prenos parametrov je prevzet iz Pascala, procesi ne morejo biti vgnezdeni ter jih ni mogoče dinamično ustvarjati. Modula ne vsebuje visoko nivojskega V/I, ker je v bistvu namenjena za implementacijo visokonivojskih V/I pripomočkov; omogoča pa programiranje perifernih naprav na najnižjem nivoju. Tipičen program v Moduli sestavlja skupek medsebojno sinhroniziranih procesov; mišljeno je, da naj bi se programi v Moduli lahko izvajali na "golem" stroju, z minimalno podporo ob izvajanju. To minimalno podporo daje Modulino jedro, ki ga sestavljajo rutine razvrščevalnika (scheduler) in sinhronizacijske rutine. Jedro je napisano v zbirnem jeziku in obsega okoli 100 besed (v PDP-11 implementaciji).

Stavki Module za sekvenčno programiranje so praktično isti kot pri Pascalu. Opuščeni elementi Pascala so: oznake, GOTO stavek, neskladovno alociranje podatkov (procedura

"new"), kazalci, FOR stavek, tip množice, zapisi s spremenljivimi deli, realna aritmetika, datoteke, visokonivojski V/I in podintervalni tipi. Nekateri sintaktični konstrukti Pascala se pojavljajo v spremenjeni obliki v Moduli. Vsi kompleksnejši sintaktični konstrukti v Moduli so zaprtega tipa (glej Algol 68) - imajo eksplicitne končne simbole v nasprotju s Pascalom. Program v Pascalu:

```
IF a<b THEN
  BEGIN
    a:=b; b:=c
  END;
```

je zapisan v Moduli takole:

```
IF a<b THEN
  a:=b; b:=c
END;
```

Pascalov tip množice je v Moduli nadomeščen s tipom "bits".

Podobno kot pri Pascalu imajo v Moduli vse konstante, spremenljivke in izrazi svoj tip. Števila in literali imajo implicitno določen tip, medtem ko je tip spremenljivk določen z deklaracijo. Tip izraza določajo tipi njegovih operandov in operatorjev. Pri Moduli srečamo štiri standardne tipe: integer, Boolean, char in bits. Kot posebnost Module si oglejmo tip bits. Vrednosti tega tipa so tabele elementov Boolovega tipa. Tip bits je vnaprej deklariran z:

```
ARRAY 0:n OF Boolean
```

Konstanta n je dolžina računalniške besede manj 1 računalnika, ki gosti Modulo. Razen osnovnih tipov so na razpolago še naslednji sestavljeni tipi: naštetje (enumeration), tabela (array) in zapis (record). Oglejmo si nekaj primerov deklaracij tipov:

```
colour=(red, yellow, green, blue)
vector=ARRAY 1:100 OF colour
matrix=ARRAY 1:20,0:10 OF integer
account=RECORD x:integer;
                y:Boolean;
                z:ARRAY 0:9 OF char
         END
```

Procedure v Moduli so bloki v smislu Pascala ali Algola. Konstante, tipe, spremenljivke in procedure lahko deklariramo lokalno dani proceduri. Okolje te procedure ne ve ničesar o obstoju njenih lokalnih predmetov; procedura (oz.blok) definira doseg (scope) za svoje lokalne predmete. Bločna struktura se je izkazala za koristno, vendar ne omogoča ohranjevanja lokalnih predmetov procedure po koncu njenega izvajanja, kakor tudi ne omogoča, da bi si več procedur delilo v teh procedurah skrite lokalne predmete. Z namenom, da odstrani te pomanjkljivosti bločnih struktur je v Moduli vpeljan koncept modula (od tod izvira tudi ime jezika).

Modul je sestavljen iz deklaracij konstant, tipov, spremenljivk, procedur ter stavkov za inicializacijo teh predmetov. Predmeti modula zaživijo čim se kontrola prenese na proceduro, ki jo vsebuje modul. Modul "ograjuje" svoje lokalne predmete (definira novo dosežno območje - scope), vendar ob tem dovoljuje natančno določitev transparentnosti te "ograje". V ta namen sta v glavo modula lahko vključeni dve listi imen (identifier): lista definiranih in lista rabljenih imen. Lista definiranih imen (DEFINE) navaja imena tistih predmetov, ki so dosegljivi zunaj modula. Lista rabljenih imen (USE) navaja imena predmetov deklariranih zunaj modula, ki naj bodo dosegljivi znotraj modula. Na ta način omogočamo skrivanje predmetov, ki predstavljajo implementacijske detajle. Modul skriva predmete, ki niso pomembni za preostali del programa ali pa predmete, katerih doseganje bi bilo s stališča zaščite neustrezno.

Primer: predpostavimo, da je predmet a deklariran v okolici modula M1. Specifikacija modula M1 omogoča okolici dostop do predmetov a,b in c:

```
MODULE M1;
  DEFINE b,c;
  USE a;
  (*deklariraj d*)
  MODULE M2;
    DEFINE c,e;
    USE d;
    (*deklariraj c,e,f*)
    (*c,d,e,f so tukaj dostopni*)
  END M2;
  PROCEDURE b;
    (*deklariraj f*)
    (*a,b,c,d,e,f so tukaj dostopni*)
  END b;
  (*a,b,c,d,e so tukaj dostopni*)
END M1
```

Imena navedena v listi definiranih imen so "eksportirana"; imena, navedena v listi rabljenih imen pa so "importirana". Vrednosti spremenljivk "eksportiranih" iz modula m ne smemo spreminjati izven m. Take spremenljivke lahko zunaj m samo "čitamo".

"Sekvenčna" Modula je razširjena z minimalnim številom pripomočkov za multiprogramiranje. Ti dodatni pripomočki so procesi (process), vmesniški moduli (interface module) in signali. Procesi so podobni proceduram, vendar se za razliko od slednjih izvajajo sočasno s programom, ki jih kliče (starta) oz.inicializira. Inicializacija procesa je možna samo v glavnem programu. Procesi ne morejo biti vgnezdeni. Procesi ne morejo klicati drugih procesov. Isti proces je lahko klican večkrat (verjetno z različnimi argumenti). Med pomanjkljivosti Module lahko štejemo nezmožnost kontrole nad izvajanjem procesov. Procesa ne moremo ustaviti, kakor tudi ne moremo na enostaven način ugotoviti, kdaj se je proces ustavil. Ta dejstva onemogočajo pisanje razvrščevalnikov (scheduler) v Moduli.

Sinhronizacijo med sočasnimi procesi dosežemo z uporabo signalov. Signali so deklarirani kot spremenljivke tipa signal (signalne "spremenljivke"). Signalne "spremenljivke" lahko uporabljamo samo kot argumente operacij "wait" in "send" (čakaj na signal, pošlji signal) in Boolove funkcije "awaited", ki je res (true), če kdo čaka na signal. Pojem signala odgovarja pojmu pogoja pri Hoareju (6) in pojmu vrste pri Brinch Hansenu (7). Poglavitni aspekt signalnega koncepta je anonimnost procesa, potem ko je ta inicializiran. Vpliv na potek procesov je možen samo z uporabo signalov in skupnih oz. deljenih spremenljivk. Procesi sodelujejo s tem, da uporabljajo skupne oz. deljene spremenljivke. Dele procesov, ki operirajo nad skupnimi spremenljivkami imenujemo kritične dele (8); preprečiti je potrebno sočasno izvajanje kritičnih delov s strani procesov. V Moduli so kritični deli deklarirani kot procedure in tovrstne - imenujemo jih vmesniške procedure - procedure (ki se nanašajo na iste skupne spremenljivke) so zbrane v posebnih modulih imenovanih vmesniški moduli (interface module), ki odgovarjajo pojmu monitorja pri Hoareju in Hansenu.

Opisane splošne multiprogramske pripomočke Module dopolnjujejo še nekateri računalniško odvisni pripomočki, ki so potrebni za programiranje računalniških perifernih naprav. Registri perifernih naprav in operatorji, ki se na te registre nanašajo se pojavljajo v Moduli s specifikacijo njihovih hardwarskih naslovov in programsko definiranih tipov. Statusni registri so ponavadi deklarirani kot tip bits, kar omogoča enostavno postavljanje in brisanje posameznih bitov. Implementacija sistemskega jezika mora omogočati učinkovito uporabo računalniških prekinitvenih mehanizmov, vključno s prekinitvenim prioritetnim sistemom. Vse operacije, ki jih izvršujejo periferne naprave in tiste, ki jih izvajajo prekinitvene rutine so v Moduli združene v takozvane krmilne procese (device process). Operacije, ki jih samostojno izvršuje periferna naprava opisujemo v krmilnem procesu s stavkom "doio". Ta stavek je možno uporabiti samo v krmilnih procesih naprave in ti so lahko deklarirani le znotraj takozvanih modulov naprav (device module). Pri modulu naprave lahko definiramo tudi prekinitveno prioriteto naprave. Modul naprave je v bistvu vmesniški modul, le da za razliko od "normalnih" vmesniških modulov vsebuje še proces naprave. V modulu naprave so "skrite" vse računalniško odvisne lastnosti periferne naprave.

3. Hardwarske zahteve za implementacijo

Za uspešno implementacijo Module morajo biti izpolnjene nekatere hardwarske zahteve. Prva je zahteva po učinkovitem mehanizmu sklada, kar omogoča implementacijo ponovno-vstopnih (reentrant) procedur in procesov, kot to zahteva Modula. Druga je zahteva po večnivojskem prekinitvenem mehanizmu. Brez prekinitvenega mehanizma postanejo moduli naprav, ki smo jih označili za eno poglavitnih novosti Module, brezpredmetni. S prioritetnimi nivoji je namreč dosežena učinkovita implementacija vzajemnega izključevanja v vmesniškem modulu, kar je v bistvu modul naprave. Mehanizem za implementacijo vzajemne izključitve je v tem primeru hardwarsko implementirana onemogočitev prekinitev. Tretja zahteva pa se nanaša na naslavljanje perifernih naprav. To mora biti enolično, se pravi samo v obliki pomnilniškega naslavljanja ali naslavljanja specializiranih V/I vrat. Pri nekaterih mikroprocesorjih je namreč mogoče naslavljanje na oba načina. V tem primeru Modula ni primerno orodje.

4. Programski krmilniki perifernih naprav

Že uvodoma smo omenili, da je Modula zelo primerna za programiranje krmilnikov za V/I naprave. Za te namene je definiran poseben tip modula imenovan modul naprave (device module). To je dejansko vmesniški modul (interface module), pri katerem je eden od sodelujočih procesov krmilni proces V/I naprave in je vključen v sam modul.

Samo v krmilnem procesu znotraj modula naprave se lahko pojavi "doio" stavek, ki označuje aktivnost periferne naprave. Običajno so periferne naprave organizirane tako, da po končani akciji sprožijo prekinitev. Torej je s stališča krmilnega procesa stavek "doio" dejansko čakanje na prekinitev in v tej točki proces običajno čaka. Kot bomo kasneje iz

P1,P2-procesa
IM-vmesniški modul

P-proces
KP-krmilni proces
DM-modul naprave

Slika 1. Shema vmesniškega modula in modula naprave

primerov videli se stavek "doio" nahaja nekje v sredi opisa procesa. To nas ne sme zmesti. Naslov prekinitvene rutine, ki je pridružen vsakemu krmilnemu procesu se nanaša ravno na stavek "doio", kjer je logični začetek procesa. Ob inicializaciji, ko se poženejo vsi procesi, steče krmilni proces do "doio", kjer je njegov logični začetek.

Stavek "doio" predstavlja podobno kot "wait" in "send" singularno točko procesa, kjer se sprosti zahteva po izključujočem dostopu do skupnih spremenljivk. Te singularnosti, ki ponavadi nastopajo v vmesniških procedurah in krmilnih procesih, razbijejo proces na več kritičnih področij, v katerih je potrebno vzajemno izključevanje. Zato so to primerne točke za preklapljanje procesov v multiprogramskem okolju (primarna primernost teh točk je seveda v dejstvu, da v teh točkah procesi čakajo - se blokirajo oz., da deblokirajo druge procese). Vsakemu modulu naprave je pridružena določena prekinitvena prioriteta, ki jo označimo v glavi definicije modula.

Krmilni procesi znotraj modula naprave so v bistvu prekinitvene rutine za katere vemo, da so običajno časovno kritične, ker so onemogočene prekinitve nižjega in istega nivoja za ves čas izvajanja rutine. Zato velja za modul naprave nekaj omejitev:

a) Krmilni procesi in vmesniške procedure modula ne smejo klicati procedur izven modula.
b) Krmilni procesi niso ponovno-vstopni (reentrant).
c) Krmilni procesi ne smejo pošiljati signalov drugim krmilnim procesom.

V modulu naprave so skrite podrobnosti, odvisne od konkretne arhitekture. Okolica komunicira s perifernimi napravami preko vmesniških procedur, ki so eksportirane iz modula naprave. V modulu naprave je torej mogoče deklarirati posamezne registre periferne naprave (kontrolni, statusni, podatkovni) kot spremenljivke in navesti njihove točne absolutne naslove. To nam med drugim omogoča, da se lahko spustimo na tako nizek nivo programiranja. Poglejmo si sedaj nekaj primerov. Prvega, ki se nanaša na računalnik PDP-11 bomo kar povzeli (4).

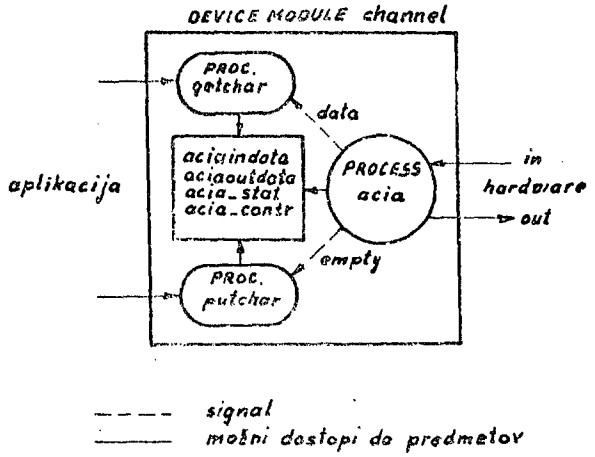
```
DEVICE MODULE timing[6];
DEFINE delay;
  VAR csr [177546B]:bits;  (*control/status*)
      tick:signal;
  PROCEDURE delay(n:integer);
    VAR left:integer;
  BEGIN
    left:=n;
    REPEAT
      wait(tick);dec(left) (*dec=decrement*)
    UNTIL left =0
  END delay;
  PROCESS clock [100B]; (*KW11-L handler*)
  BEGIN
    LOOP
      csr[6]:=true;      (*enable interrupt*)
      doio;  (*wait for clock tick*)
      csr[6]:=false;
      WHILE awaited(tick) DO send(tick)  END
    END
  END clock;
BEGIN  (*timing initialization*)
  clock (*start process clock*)
END timing;
```

V gornjem primeru imamo znotraj modula samo eno (vmesniško) proceduro in ta je tudi eksportirana. Prioriteta urnih prekinitev ima vrednost 6 in prekinitveni vektor kaže na lokacijo 100 (oktalno). Modul se inicializira ob zagonu sistema. V našem primeru se samo sproži proces "clock", ki se odvije do svojega logičnega začetka - do stavka "doio". Iz primera je razvidno kako so posebnosti, kot so: prekinitveni vektor, naslov kontrolnega registra naprave in prioriteta prekinitve zaprte v modulu. Za okolico je modul dostopen le preko "delay" procedure, ki je eksportirana. Naslednji primer se nanaša na mikroprocesor 6800. V obliki modula naprave je opisan krmilnik serijskega dupleksnega kanala realiziranega z integriranim vezjem ACIA (6850):

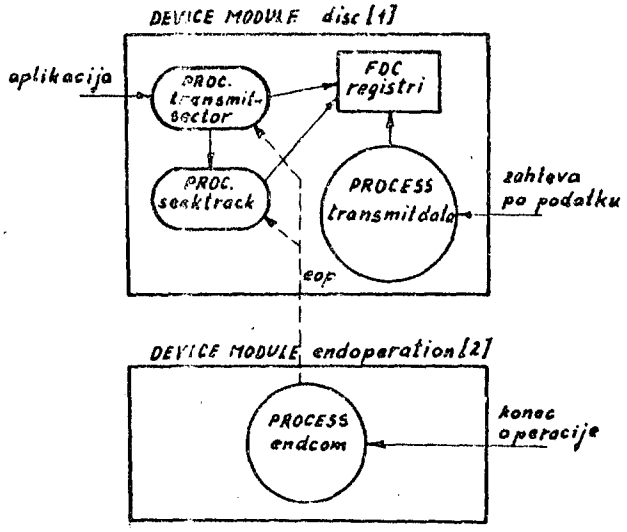
```
DEVICE MODULE channel [1];
DEFINE getchar, putchar;
  VAR aciaindata [DFEFH]:char;
      aciaoutdata [DFEFH]:char;
      acia_stat [DFEEH]:bits;
(*acia_stat[0]=true: aciaindata full*)
(*acia_stat[1]=true: aciaoutdata empty*)
      acia_contr [DFEEH]:bits;
  PROCEDURE getchar(VAR ch:char);
  BEGIN
    wait(data);
    ch:=aciaindata AND 7FH;
  END getchar;

  PROCEDURE putchar(ch:char);
  BEGIN
    IF acia_stat[1]=false THEN wait(empty)
    END;
    aciaoutdata:=ch;
  END putchar;

  PROCESS acia [100H];
  BEGIN
    LOOP
      acia_contr[5]:=true;
      acia_contr[7]:=true;
(*enable receiver and transmitter int.*)
      doio;
      acia_contr[5]:=false;
      acia_contr[7]:=false;
      IF acia_stat[0]=true THEN send(data)
        ELSE send(empty) END
    END
  END acia;

BEGIN
  acia_contr:=[0,1]; (*master reset*)
  acia_contr[0]:=false; (*clock ratio 64,
        7 bits, even parity, 2 stop bits*)
  acia
END
```

Opomba: oznake H (heksadecimalno) ni v originalni Moduli.

Oba procesa sprejemanja in pošiljanja znakov sta združena v en sam krmilni proces naprave, ker uporabljata isto prekinitveno linijo. S testiranjem statusnega registra ACIA-e (acia_stat) je mogoče ugotoviti, ali se prekinitev nanaša na sprejem ali oddajo. Sicer pa je sama struktura modula zelo pregledna in ne potrebuje posebnega pojasnila. Ob inicializaciji se izvrši resetiranje ACIA-e in nastavi format znakov. Šele potem se starta krmilni proces. Shematično je opisani modul naprave prikazan na sliki 2.



Slika 2. Modul naprave "channel" z lokalnimi predmeti, medsebojnimi povezavami in povezava z okolico.



Slika 3. Modula naprave pogona za gibki disk.

Na tretjem in zadnjem zgledu pa si poglejmo še programski krmilnik za pogon za gibki disk (kontroler 1771) za procesor 6800. Tu imamo opravka z dvema nivojema prekinitev. Na prvem nivoju je prekinitev naprave (data request), ki pomeni, da zahteva či nje ali zapisovanje znaka. Te zahteve prihajajo, kadar je v komandnem registru (fdc com) prisotna koda za čitanje ali pisanje sektorja. Na drugem nivoju pa so prekinitve, ki označujejo konec operacije; pa naj bo to iskanje sledi, pisanje ali čitanje sektorja. Na zgledu si bomo ogledali samo oba modula naprave. Struktura obeh modulov naprave je prikazana na sliki 3.

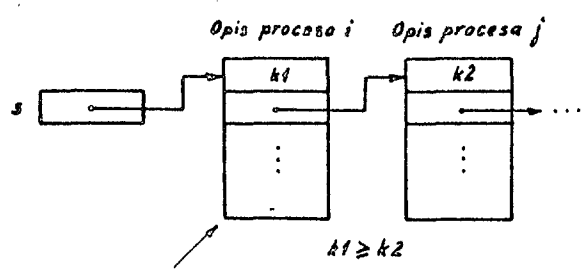
```
DEVICE MODULE disk [1];
  DEFINE transmitsector;
  USE endop,buffer;
  CONST n=128; (*buffer size*)
        read=8CH;write=ACH;seek=1CH;
  VAR fdcstat FBCCH: bits; (*status reg.*)
      fdccom FBCCH, (*command register*)
      fdcsect FBCEH, (*sector register*)
      fdcdata FBCFH: integer; (*data reg.*)
      buffer: ARRAY 1:n OF char;
      diskbusy,rw: Boolean;
      diskfree: signal;
      i: integer;

  PROCEDURE transmitsector(VAR buf: ARRAY
    1:n OF char; sector,track:integer;
    direction:Boolean);
    VAR j: integer;
  BEGIN
    j:=1;
    IF diskbusy THEN wait(diskfree) END;
    diskbusy:=true; seektrack(track);
    fdcsect:=sector; rw:=direction;
    IF rw=true THEN fdccom:=read;
    ELSE
      REPEAT
        buffer[j]:=buf[j]; j:=j+1
      UNTIL j>n;
      fdccom:=write END;(*write sector*)
    wait(endop); diskbusy:=false;
    send(diskfree);
    IF rw=true THEN
      REPEAT
        buf[j]:=buffer[j]; j:=j+1
      UNTIL j>n
    END;
  END transmitsector;

  PROCEDURE seektrack(track:integer);
  BEGIN
    fdcdata:=track;
    fdccom:=seek;
    wait(endop)
  END seektrack;

  PROCESS transmitdata [06H];
  BEGIN
    LOOP
      doio;
      IF rw=true THEN buffer[i]:=fdcdata
        ELSE fdcdata:=buffer[i] END;
      i=(i MOD n)+1
    END
  END transmitdata;

BEGIN
  i:=1; transmitdata; diskbusy:= false
END disk;


DEVICE MODULE endoperation [2];
  DEFINE endop;
  VAR endop: signal;

  PROCESS endcom [03H];
  BEGIN
    LOOP
      doio; send(endop)
    END
  END endcom;
BEGIN
  endcom
END endoperation;
```

5. Preklapljanje in razvrščanje procesov

Preklapljanje procesov se vrši samo v singularnih točkah. Te pa so "send", "wait" in "doio" ukazi. Izjema so samo preklopi ob prekinitvah, ki prekinejo izvajanje procesa, če imajo dovolj visoko prioriteto. Sistem, opisan s programom v Moduli, se sestoji iz fiksnega števila paralelnih procesov. Vsi procesi so deklarirani in inicializirani na nivoju gnezdenja 0 (kar pomeni, da procesi ne morejo biti deklarirani v procesih ali v procedurah). Proces poženemo (startamo, inicializiramo) s procesnim stavkom, ki je ime procesa z listo parametrov. Nivo gnezdenja 0 predstavlja glavni program in v tem se poženejo vsi procesi sistema. Torej Modula dovoljuje dinamično ustvarjanje procesov, vendar so možnosti ustvarjanja omejene. To omogoča, da se pomnilniški prostor za procese dodeljuje sekvenčno, v glavnem programu. Vsak proces ima svoj sklad fiksne dolžine; ker Modula dovoljuje rekurzivne procedure, mora v tem primeru biti prisoten napotek prevajalniku, ki pove maksimalno globino rekurzije.

Proces se lahko nahaja v enem od treh stanj. Lahko se izvaja (je aktiven), lahko je pripravljen na izvajanje (pogoji, na katere je čakal so izpolnjeni) ali čaka na izpolnitev določenih pogojev (izpolnitev pogoja potrjuje signal). Razvrščanje procesov se odvija na "prostovoljni" osnovi. Vsak proces z izvajanjem "wait" ali "send" stavka preide v pasivno stanje ("pripravljen na izvajanje" oz."čakajoč na signal") in avtomatsko povzroči aktiviranje procesa iz vrste procesov pripravljenih na izvajanje. Pri tem velja naslednje pravilo razvrščanja. Proces, ki pošlje signal s preide v stanje "pripravljen na izvajanje". Aktivira pa se proces, ki je prvi v čakalni vrsti na signal s. Kadar proces naprave izvede "send", se signalizirani proces postavi iz stanja "čakajoč na signal" v stanje "pripravljen na izvajanje", proces naprave pa nadaljuje z izvajanjem. Razlog za to izjemo je dejstvo, da proces naprave lahko signalizira le "normalne" procese, ti pa imajo nižjo prioriteto.

Vidimo, da je mogoče signale enostavno implementirati kot spremenljivke, katerih vrednosti so kazalci na opis prvega procesa v vrsti procesov, čakajočih na signal s (slika 4). Ob operaciji send(s) je prvi proces v vrsti na s postavljen v stanje "pripravljen na izvajanje"; ob operaciji wait(s,p) pa je tekoči proces uvrščen v vrsto procesov čakajočih na s in sicer v skladu s prioriteto čakanja p. Procesi v vrsti na s , ki imajo isti prioriteto, so medsebojno razvrščeni po FIFO zaporedju; tisti, ki je najdalje čakal, je prvi reaktiviran.



Proces, ki čaka na signal s (wait(s,k1). K1 predstavlja čakalno prioriteto.

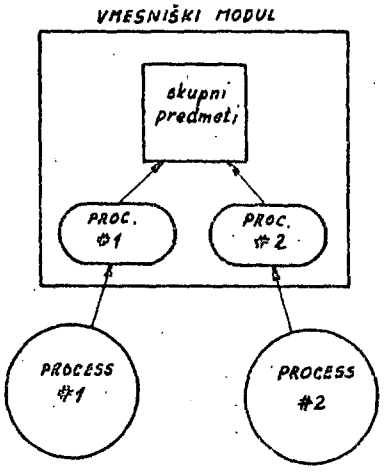
Slika 4. Signal s in čakalna vrsta procesov na s.

Zunanja prekinitev lahko edina "nasilno" prekine proces, ki se izvaja. Izvrši se preklop in aktivira se krmilni proces naprave, ki ustreza konkretni prekinitvi. Za razvrščanje krmilnih procesov naprav skrbi torej prioritetni prekinitveni mehanizem.

Krmilni proces naprave steče do singularne točke ("doio" ali "wait"), kjer pride do novega preklopa. Zaradi enostavne implementacije vmesniških modulov (glej naslednjo točko) je tedaj potrebno takoj nadaljevati prekinjeni proces.

6. Vzajemna izključenost in zaščita

V multiprogramskem in multiprocesorskem okolju mora biti izpolnjen princip izključenosti hkratnih dostopov procesov do skupnih predmetov. Vmesniški modul je struktura, ki izpolnjuje omenjeni princip. Vmesniški modul vsebuje skupne predmete, ki so definirani kot lokalni predmeti vmesniškega modula. Dostop do njih je mogoč samo preko eksportiranih procedur - vmesniških procedur (slika 5).



Slika 5. Vmesniški modul zagotavlja pravilen način uporabe skupnih predmetov (skupnih procesoma 1 in 2).

Med izvajanjem vmesniške procedure ne more noben drug proces vstopiti v vmesniški modul s klicem kakšne druge vmesniške procedure. Šele, ko prvi proces konča izvajanje vmesniške procedure ali pride v njej do singularne točke, kjer pride do preklopa, lahko drugi proces vstopi v vmesniški modul. V multiprogramskem okolju (en sam procesor) že sam mehanizem razvrščanja v Moduli - kot je bil definiran v prejšnji točki - zagotavlja vzajemno izključenost procesov pri uporabi skupnih predmetov. V multiprocesorskih sistemih in v monoprocesorskih sistemih s poljubnim mehanizmom razvrščanja (npr. s preklopi ob izteku prekinitvenih rutin) pa je očividno potreben za implementacijo vmesniškega modula še dodatni semafor (monitor gate semafor, glej Hoare (6)).

Modul naprave je tudi vmesniški modul. Tu je vzajemna izključenost dosežena kar z onemogočitvijo prekinitev. V modulu naprave se izvajajo vse lokalne procedure in krmilni proces naprave na istem prioritetnem nivoju. Skupne predmete uporabljata krmilni proces naprave in "normalni" proces. Krmilni proces naprave lahko postane aktiven samo, če so omogočene prekinitve, torej samo tedaj, ko je "normalni" proces zapustil modul naprave (medsebojno izključevanje).

Kadar vmesniške procedure vsebujejo singularne točke, lahko pride v Moduli do slučaja, da pride sredi vmesniške procedure p1 do preklopa in se nato izvaja vmesniška procedura p2, ki operira nad istimi skupnimi predmeti (p1 in p2 sta v istem vmesniškem modulu). Posledica je lahko porušitev integritete skupnih podatkov. Vzrok temu je Modulina specialna implementacija vmesniških modulov. Lep primer je vmesniška procedura "transmitsector" modula naprave "disk" iz točke 4 (kot smo že omenili, je modul naprave posebne vrste vmesniški modul). Če iz te procedure odstranimo stavke, ki se nanašajo na "diskbusy" in "diskfree" lahko pride do preklopa v wait(endop), nakar je vnovič lahko klicana ista vmesniška procedura, kar ima za posledico nepravilno operiranje nad skupnim predmetom "fdccom". Zaščito skupnega predmeta smo v tem primeru izvedli z dodatno sinhronizacijo ("diskbusy" Boolova spremenljivka in "diskfree" signal). Taka sinhronizacija se normalno rešuje s semafori. V Moduli lahko implementiramo semafor s signalom in eno spremenljivko. Poglejmo si implementacijo binarnega semafora (1):

```
INTERFACE MODULE resourcereservation;
  DEFINE semaphore,P,V,init;
  TYPE semaphore=RECORD taken:Boolean;
                        free: signal;

  PROCEDURE P(VAR s:semaphore);
  BEGIN
    IF s.taken THEN wait(s.free) END;
    s.taken:=true
  END P;

  PROCEDURE V(VAR s:semaphore);
  BEGIN
    s.taken:=false;
    send(s.free)
  END V;

  PROCEDURE init(VAR s:semaphore);
  BEGIN
    s.taken:=false
  END init;
END resourcereservation.
```

Iz zgornjega opisa implementacije semafora je razvidno, da smo skupni podatek "fdccom" v zgoraj navedenem primeru v bistvu zaščitili z operacijama P in V nad semaforom s, ki ga sestavljata "diskbusy" in "diskfree".

7. Sklep

Iz nekaj primerov je razvidno, da je Modula skoraj idealna za pisanje programskih krmilnikov. To velja seveda samo v primeru, da obstoja prekinitveni mehanizem in da so naprave priključene preko prekinitvenih vhodov. Pri vedno kompleksnejših perifernih integriranih vezjih (ki so že skoraj tako močna kot procesorji) se bodo za "doio" skrivale vedno kompleksnejše operacije. Obstojali bodo torej precej dolgi dejansko paralelni hardwarski procesi. Zato ne bo dovolj samo čakati na prekinitev.

Na primeru pogona za gibki disk smo videli, da sta bila potrebna za eno napravo dva modula naprave. V primeru naprave z večjim številom prekinitvenih zahtev različnih prioritet moduli naprav niso več naravne celote.

Med pomanjkljivosti Module lahko štejemo sicer enostaven, vendar za določene sisteme morda neprimeren mehanizem razvrščanja in preklapljanja procesov. Ta mehanizem je vgrajen v Modulino jedro in ga ni možno programirati v sami Moduli.

Modula omogoča zelo jasno strukturiranje sistemov procesov. Sistem procesov, ki je sinhroniziran s preglednim mehanizmom signalov uporablja skupne predmete, ki so zaščiteni v vmesniških modulih. Problemi, vezani na sinhronizacijo dosegov do skupnih predmetov so rešeni v vmesniških procedurah vmesniških modulov (monitorjev).

Na vsak način lahko Modula precej prispeva k boljši dokumentiranosti malih operacijskih sistemov mikroračunalnikov in sprotnih (real-time) procesov, ki se odvijajo na njih. Njena glavna novost pa je razširitev možnosti programiranja v visokem jeziku tudi na programske krmilnike naprav.

8. Literatura

1. N.Wirth: Modula: a Language for Modular Multiprogramming, Software-Practice and Experience, vol.7, 335 (1977).
2. N.Wirth: The Use of Modula, Software-Practice and Experience, vol.7, 37-65 (1977).
3. N.Wirth: Design and Implementation of Modula, Software- Practice and Experience, vol.7, 67-84 (1977).
4. J.Holden,I.C.Wand: An Assessment of Modula, York Computer Science Report No.16 (November 1978).
5. PDP11 Peripherals Handbook.
6. C.A.R.Hoare: Monitors-an Operating System Structuring Concept, Com.of the ACM, vol.17, no.10 (Oct.1974).
7. Per Brinch Hansen: The Programming Language Concurrent Pascal, IEEE Trans.on Software Eng., vol.SE1, no.2 (June 1975).
8. M.Exel: Komunikacija med sekvenčnimi procesi - pregled, del I in II, časopis Informatica, št.1 in št.2 (1977).

VERIFICATION OF STORED SERIAL DATA

R. MURN
D. PEČEK

UDK: 681.327.63 : 621.3.019.3

J. STEFAN INSTITUTE, LJUBLJANA, YUGOSLAVIA

The paper deals with problems and methods of verifying serial data while using contemporary magnetic storage units. We encounter these problems most especially in the transfer of information between microprocessors and digital cassettes or floppy disks. The method is based on the theory of cyclic codes. The basic properties of cyclic codes and realisation of suitable hardware circuity is given.

VERIFIKACIJA ZAPISA SERIJSKIH PODATKOV. V članku je opisana problematika verifikacije serijskih podatkov pri sodobnih magnetnih pomnilniških napravah. Z omenjeno problematiko se srečujemo zlasti pri izmenjavi informacij med mikroprocesorji in digitalnimi kasetami in gibkimi diski. Podana je metoda za učinkovito verifikacijo zapisanih podatkov, le-ta je osnovana na teoriji cikličnih kodov. Opisane so osnovne lastnosti cikličnih kodov in realizacija ustreznega vezja.

I. INTRODUCTION

Nowadays it is impossible to imagine the world of microcomputers without low cost mass storage units, for example the digital cassettes and floppy disk drives. The use of such units requests the reliability of the data transfer as high as possible. Therefore it is necessary to define and to work out the methods for verification of the data transfer when constructing the corresponding controller. Recent developments in error-correcting codes have contributed toward achieving the high reliability required by today's digital systems, and it is evident that the use of coding methods for error control has become an integral part in the design of modern computers and communications systems.

We encounter these problems most especially in the transfer of information between microprocessors and floppy disks. The verification is based on the theory of cyclic codes and the realisation of the necessary hardware circuitry. Cyclic codes are defined and described from a viewpoint involving polynomials. The potentialities of these codes for error detection and the equipment required for implementing error detection systems using cyclic codes are also described.

II. DESCRIPTION OF CYCLIC CODES

It is convenient to think of the binary digits as coefficients of a polynomial in the dummy variable x. For example, a message 1001101 is represented by the polynomial $1+x^3+x^4+x^6$. The polynomial is written low-order-to-high-order because these polynomials will be transmitted serially, high order first, and it is conventional to indicate signal flow as occuring from left to right. These polynomials will be treated according to the laws of ordinary algebra with one exception. Addition is to be done modulo two.

A cyclic code is defined in terms of a generator polynomial G(x) of degree n-k. A polynomial of degree less than n is a code poly-

nomial F(x), i.e. acceptable for transmission, if and only if it is divisible by the generator polynomial G(x). The message polynomial is assigned as M(x)-degree < k.

Code polynomials can be formed by simply multiplying any polynomial of degree les than k by G(x). The method has the advantage, however, that it results in a code polynomial in the high-order coefficients are message symbols and the low-order coefficient are check symbols. To encode a message polynomial M(x), we divide x^{n-k}. M(x) by G(x) and then add the remainder R(x) resulting from this division to x^{n-k}. M(x) to form the code polynomial:

$$x^{n-k} \cdot M(x) = Q(x).G(x) + R(x),$$

where Q(x) is the quotient and R(x) the remainder resulting from dividing x^{n-k}. M(x) by G(x). Since in modulo two arithmetic, addition and subtraction are the same,

$$F(x) = x^{n-k}.M(x) + R(x) = Q(x).G(x),$$

which is multiple of G(x) and, therefore a code polynomial. Furthermore, R(x) has degree less than n-k. Thus the k higher-order coefficient of F(x) are the same as the coefficients of M(x), which are the message symbols. The lower order n-k coefficients of F(x) are the coefficients of R(x), and these are the check symbols.

Example:

Consider a code for which n=15, k=10 and n-k=5, which uses $G(x) = 1+x^2+x^5+x^9$ = 1010010001. We divide $x^5.M(x)$ by G(x) and find the remainder.

```
          1110001111
(110101)/100010010100000
         110101
          101110
          110101
           110111
           110101
               100100
               110101
                100010
                110101
                 101110
                 110101
                  110110
                  110101
                      11 (remainder)
```

$x^5.(1+x^2+x^5+x^9)=(1+x^2+x^4+x^5).(1+x+x^2+x^3+x^7+x^8+x^9)+(1+x)$.

$F(x) = x^5.M(x)+(1+x) = 1+x+x^5+x^7+x^{10}+x^{14}$

F(x) = 110001010010001

III. PRINCIPLES OF ERROR DETECTION

An encoded message with errors is represented by

$$T(x) = F(x) + E(x),$$

where E(x) is polynomial which has a nonzero term in each erroneous position. The addition is modulo two, T(x) is true encoded message with the erroneous positions changed. If the received message T(x) is not divisible by G(x), then an error has occured. Since F(x) was constructed so that it is divisible by G(x), T(x) is divisible by G(x) if and only if E(x) is also. An error pattern E(x) is detectable if and only if it is not evenly divisible by G(x). To insure an effective check, the generator polynomial G(x) must be chosen so that no error pattern E(x) which we wish to detect is divisible by P(x).

To detect errors, we divide the received message T(x) by G(x) and test the remainder. If the remainder is nonzero, an error has been detected. If the remainder is zero, either no error or an undetectable error has occured.

Using our detection method, we do not require the quotient. Modulo two arithmetic has simplyfied the division considerably.

The divide principle is shown in a given example. In every step we subtract the divisor so that the first number one aligns with the highest degree number one in the rest.

The hardware to implement this algorithm is a shift register (Flip-Flops) and a collection of modulo two adders (EXCLOSIVE OR). The number of shift register positions is equal to the degree of the divisor, G(x), and the divident is shifted through high order first and left to right. As the first one (the coefficient of the high-order term of the dividend) shifts off the end we subtract the divisor. Figure 1 gives a basic register that performs a division by $1+x^5+x^{12}$.

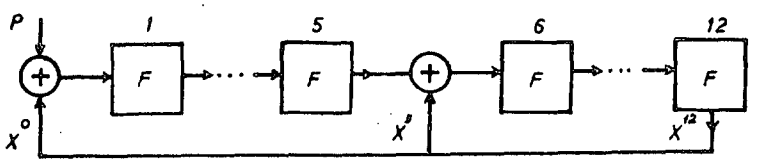


Figure 1. A shift register for dividing by $1+x^5+x^{12}$.

It is necessary to devote much of the effort when selecting the polynomial generator G(x). The size of the G(x) must enable efective discovering of the single, multiple and group faults (Burst). The cyclic codes detailed analysis (11,12,13) obtains the evaluation G(x). We shall point out the most important conclusions.

a) A cyclic code generated by any polynomial G(x) with more than one term detects all single errors;
b) Every polynomial divisible by $1+x$ has an even number of terms. It follows that the code generator by $G(x)=1+x$ detects not only any single error, but also any odd number of errors;
c) A code generator by the polynomial G(x) detects all single and double errors if the length n of the code is no greater than the exponent e to which G(x) belongs. (A polynomial G(x) is said to belong to an exponent e if e is the least positive integer such that G(x) evenly divides x^e-1);
d) Any cycle code generated by a polynomial of degree n-k detects any burst-error of length n-k or less;
e) The fraction of bursts of length $b > n-k$ that are undetected is $2^{-(n-k)}$ if $b > n-k+1$, $2^{-(n-k-1)}$ if $b=n-k+1$.

the sectors is a constant or can be defined by the program as well /4/. Let us take a short look to a standard sector format (IBM 3740). Each sector consists of ID field and data filed (fig. 2). Both fileds have additional bytes for CRC verification. Data fields involves 128 bytes and can be multiplied by 2^N (N=0,1,2, ...). Concerning the polynomial generator theory, four term generators are found to be most economical. $G(x)=1+x^5+x^{12}+x^{16}$ discovers single faults, all the faults if the number of faults is odd, the double faults if the size of the data stream is less or equal 32767. It also discovers all the burst faults if the size is 16 or less. If the size is 17 or more the discovering probability is $(1-12^{16})=99.998\%$. The same results are obtained, by following generators: $G(x) = 1+x^2+x^{15}+x^{16}$, $1+x+x^{14}+x^{16}$, $1+x^4+x^{11}+x^{16}$.

Fig. 3 represents the realisation for a polynomial expression $G(x)=1+x^5+x^{12}+x^{16}$ and the checking hardware.

The check characters are generated by dividing the data by the encoding polynomial. The remainder resulting from the division is then appended to the message stream as a check character (CRC). During reception of the data it is again divided by the same polynomial. If

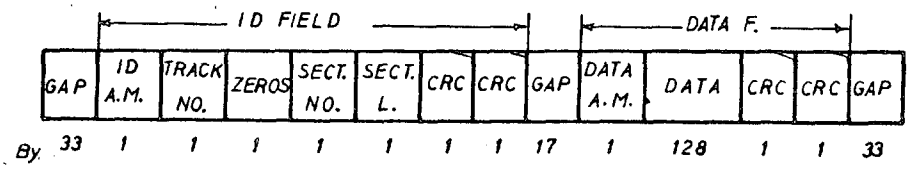


Figure 2. IBM 3740 Sector Format

IV. FLOPPY DISK DATA TRANSFER VERIFICATION

The hardware, that performs the data transfer verification is a functional part of the floppy disk controller. The polynomial generator must be adapted to a specific diskette write format. Most often, the discette has 77 tracs, each trac consists of 26 sectors. The size of

no errors have occured in transmission, the result of this division should be zero since adding the check character (remainder) to the message has the effect of making the received message evenly divisible by the code polynomial. This will be indicated by an all zeros output.

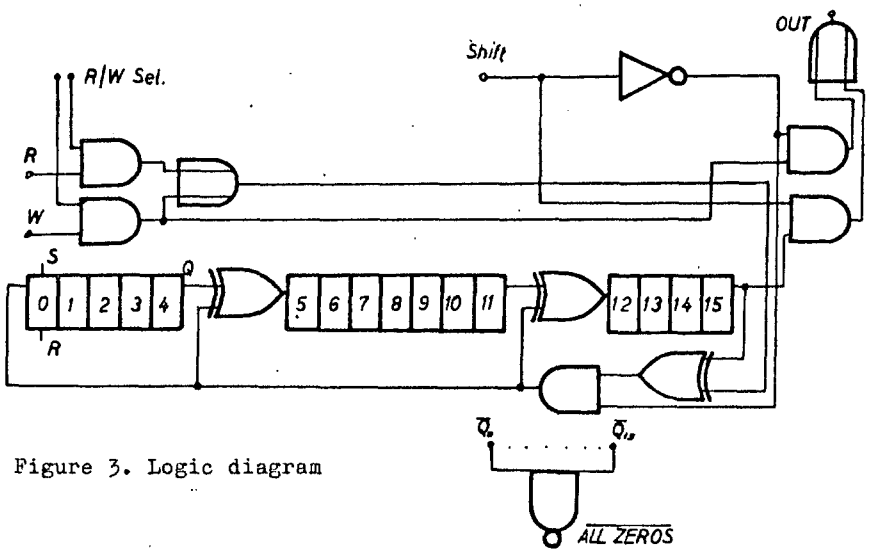


Figure 3. Logic diagram

In the Write operation the sequence begins with presetting the 16 registers (D flip-flop) to logical 1. The circuit generates the check characters by entering each bit of the data stream into the Write data in terminal. This process effectively divides the data stream by the polynomial. After the last data bit is entered, the check character is stored; it is then added to the data stream by shift control. In the Read operation, sequence is the same as the Write operation except that shift is not used.

V. CONCLUSION

The CRC method is found most effective as the verification tool, and very simple for realisation, especially when dealing with FDD and digital cassettes. Indipendently of the CRC method we can add another method that is called a PLL (Phase Locked Loop) method. CRC and PLL form together unique and effective system for data transfer between computer and FDD or digital cassette.

When dealing with professional magnetic drives, the aditional methods for detection and fault localisation are added.

REFERENCES

/1/ I.E. Meggitt; Error Correcting Codes for Correcting Bursts of Errors, IBM J.Res. Dev., 4. July 1960.

/2/ Peterson, W.Wesley; Error-Correcting Codes, MIT Press, Cambridge, Mass., 1965.

/3/ D.T.Tang, R.T.Chien; Coding for Error Control, IBM, Syst. J., No. 1, 1969.

/4/ R.Murn, D.Peček; Gibki diski, Informatica, No. 1, 1978.

A SCENARIO FOR COMPUTER NETWORK RESEARCH

D. L. A. BARBER,
T. KALIN

EIN PROJECT, TEDDINGTON, UK.

UDK: 681.324

INSTITUTE JOŽEF STEFAN, LJUBLJANA, YUGOSLAVIA

SCENARIJ ZA RAZISKAVE NA PODROČJU RAČUNALNIŠKIH MREŽ. Članek proučuje vpliv tehnoloških sprememb na področju mikroelektronike in predlaga novo arhitekturo računalniških mrež in protokolov primernih za prihodnost. Na koncu je shematično podan načrt raziskav, ki bi bile potrebne na področju računalniških mrež.

The impact of the technological changes in microelectronics upon the networking field is examined. A new architecture for computer networks and suitable for the future are proposed, followed by an outline of a scenario for future computer network research.

1.o. Introduction

Today, the pace of technological change is such that attitudes and ideas can very soon become outdated. For example, much has been said in the past about the reactionary attitudes of some postal and Telecommunications Authorities towards new data networks. This understandably stemmed from their heavy investment in electro-mechanical telephone plant that was being rendered obsolete by new digital systems. The many public data networks announced recently indicate that this situation is rapidly changing as the benefits to be gained by adopting digital equipment become apparent.

The major computer manufacturers who were once at the forefront of computer technology are just beginning to adopt, in their turn, similar reactionary postures now that large Mainframe systems are becoming absolescent. For there is a strong possibility that, within five years, informed users may no longer be ordering the conventional general purpose Mainframe as we know it at the present time.

Of prime concern in this paper, however is the realisation of the danger that our research programmes in Computer Science and Computer Networks may also be rendered ineffectual and misdirected by the most recent technological advances. This is particularly worrying because, by definition, research should be pointing our way into the Future.

It is arguable that fundamental research more often allows us to eliminate wrong answers than to the identify right ones, though this, perhaps, is seldom appreciated by those who pay for it. But often, a breakthrough suggests a change of direction that is clearly desireable, even if it makes past efforts seem rather wasted. Such changes must be made if we are to maximise the return on the capital invested in our research programs.

The advent of micro-electronics has been a breakthrough of such a great magnitude that it will change most aspects of human affairs. This is already appreciated at the highest political levels in some countries, because of the profound effects on individuals and the jobs they will have to do in the world of tomorrow. Computer networks, themselves a formidable factor for shaping the future, are being radically changed by intelligent terminals, based on new microelectronics techniques. These terminals exploit cheap storage and processing power to provide each user with the power of today's small mainframe systems, at roughly the cost of a teletype. The impact of this on network research programmes has so far been minimal.

2.o. Present Research Activities

Some years ago, layered protocols were rightly seen as the solution to promoting freedom of connection in a heterogenous network of mainframes, which support many non-intelligent terminals attached through their own local networks. This, now classic, network architecture, shown in figure 1 ,

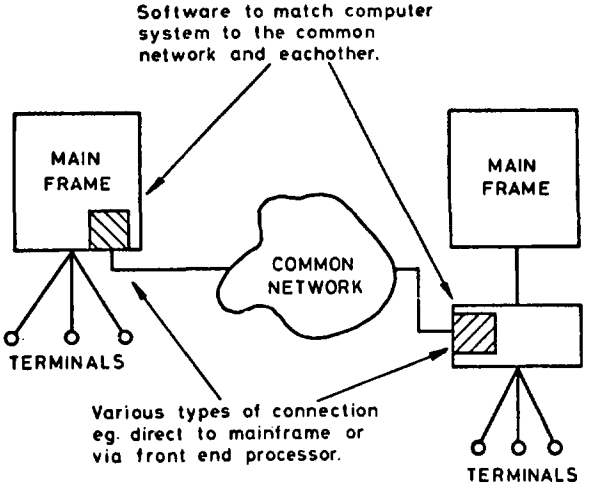


Figure 1: TYPICAL CURRENT NETWORK ARCHITECTURE

gave rise to the Transport Station, Virtual Terminal, File Transfer and other protocols, shown in figure 2.

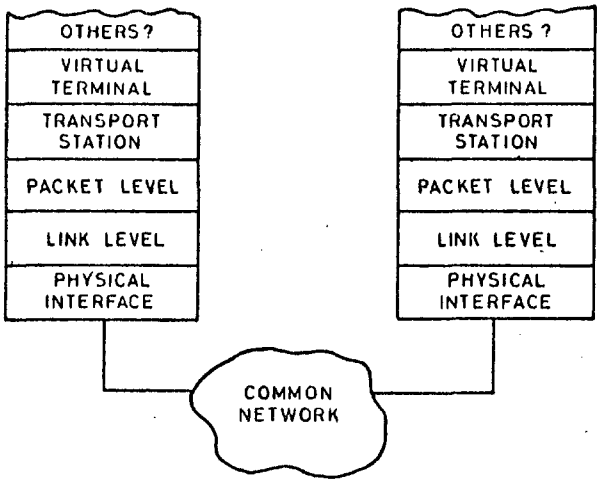


FIGURE 2

These have, or are being, implemented in various research networks, (ref.1) and are generally thought of as forming the basis of standards for 'open working' in the future.

Open working is the situation where a number of computer systems connected by a Public data network are able to intercommunicate in some way. But, unfortunately, the variations between existing systems of different manufacture render it virtually impossible for them to interact meaningfully unless some modifications are introduced somewhere. At the level of the physical connection to a data link, the standard interfaces introduced by CCITT for modems have forced all systems to seem similar; possibly it is this that gave birth to the idea of introduction of corresponding levels of protocols, built above the physical interface, in all of the computers attached to a network, as shown in figure 2.

The structured protocol levels are supposed to make the computer systems look sufficiently alike to allow them to usefully intercommunicate. But a number of difficult questions arise. Firstly, how many, and what, levels are needed; secondly, where are these levels to be located and, thirdly, is it possible that levels can be implemented independently andaccurately enough to achieve the required result? Indeed perhaps the most difficult question of all is how to agree on what is the required result; this is not mentioned at all, or is discussed only in the most general and vague terms, in most papers describing protocol development. Usually these papers concern themselves with the technicalities of the particular protocol under consideration, and tend to ignore the environment within which it has to operate.

We shall certainly not address these questions here, for our purpose is to argue that a change of direction is required at the present time. However, it is interesting to note the long delay between the emergence of the idea of layered protocols in a research atmosphere, and its serious consideration by standards bodies.

The concept was first described in papers written about a decade ago, when systems centered on a Mainframe dominated the computing scene (ref 2). But only now, when these systems are threatened with extinction, are the same simple ideas appearing before the arganisations responsible for standardisation (ref. 3), where the levels of protocol typically under consideration are shown in figure 3.

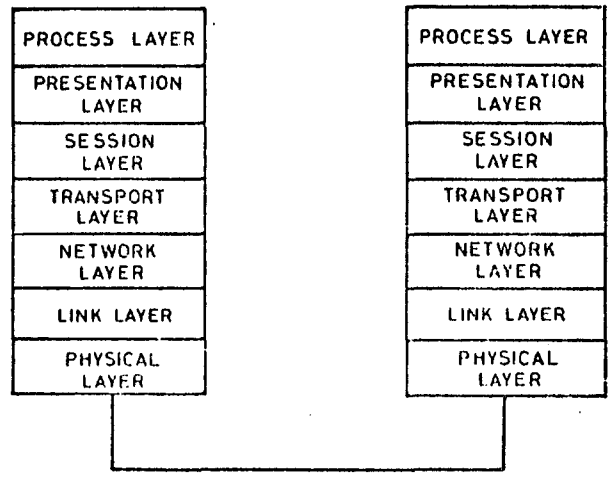


FIGURE 3 LAYERS OF PROTOCOL BEING DISCUSSED BY STANDARDS COMMITTEES

This is most unfortunate because standards based on today's network architectures are likely to be obsolete before they are published. If they are to be timely, standards planned now should match the network architecture of the future. Of course, much of the experience gained in designing and implementing today's protocols can still be utilised, but in the context of the new style of architecture discussed in Section 4, rather than in relation to the orthodox architecture hitherto accepted as the norm.

3.0. The Economics of Micro-electronics

The traditional falling cost of hardware is common knowledge, but the startling reductions made possible by the latest microprocessors and semiconductor stores are less well appreciated. Existing technology has already led to the 'personal' computer (ref. 4), which is a sufficiently low price to create a volume market. This will allow many of the existing terminal devices to be written off at a much earlier stage than was intended, when they were purchased. Early amortisation is likely to be even more justified if the cost of maintenance and their unattractiveness to the user are taken into account. For this reason there is no doupt that the personal computer or home terminal will also have a profound effect on the business world, where the replacement

of electric typewriters by competitively priced intelligent terminals could bring office automation into being very quickly. Indeed, the covergence of office automation and data processing is now proceeding at an over-increasing rate.

The high volume sales and the ensuing competition between intelligent-terminal manufacturers will probably bring enhanced features, rather than further cost reductions; these features will include mass local storage on mini discs and bubble devices, flexible I/O facilities for connections to local equipment. The already powerful processors such as the Intel 8086 and the Zilog 8000 (ref. 5), now being designed into new equipment, are even now being overshadowed by those in prospect. Appropriate software will be provided as packages (ref 6) sold competitively on the open market, so that user's preferences will determine which are successful. Thus, for example, a particularly effective file editor package establish a de facto standard for the user interface, through common acclaim.

There will be a corresponding impact of microelectronics on Mainframe design. This will provide increased reliability by the introduction of multiprocessor architectures, and will allow storage and processing features to be more closely associated in a distributed array processing architecture. This will transform the economics of large data base systems (ref 7).

As in the case of terminals, existing general purpose mainframes will be written-off at an accelerated pace, if the economics of new systems are realistically evaluated. The exceptions will be the bigger, newer systems at large establishments that have a high residual value; however, these systems will need to be reprogrammed because the widespread use of intelligent terminals will simplify the task of terminal handling, rendering many features of present mainframe architecture redundant. As a result smaller executives can be written to support the new kinds of data base services that will be required by users of intelligent terminals, in the light of the new architecture described below.

4.0. A New Architecture for Networks

When sufficient storage and processing power resides in each user's personal terminal, there will be a drastic change in the type of traffic and connections that have to be established through a public data network. The situation will be analagous to that of private cars and public transport, where the convenience of the former is displacing the latter, except for long trips where rail and air transport becomes attractive. Because all interactive data entry, preparation and editing functions will be done locally, the present character-oriented traffic will disappear and there will no longer be need for the PAD facility (ref 8), which is now performed by some public networks and is relatively complex compared with their basic transportation functions.

Frequently used data will be held locally, so the data network will be used only for access to data which is rarely consulted, or to obtain changing data that is centrally updated by a data bank service. There is a strong similarity here with a typical office which has current business files readily to hand, and obtains any other information that may be needed when the occasion arises; albeit with more or less difficulty, delay and expense. In the case of the data network too, there will be trade-offs to be made between the cost of local storage and that of using the network, taking into the delay and diminished value of out-of-date information.

The ability for users of intelligent terminals to construct files locally allow them to exchange messages directly with other users' terminals, rather than communicate through a common Mainframe system. These exchanges may be interactive so that users can conduct a dialogue, or messages may be left at an unattended terminal to be read by its owner later. Research projects have studied the intelligent terminal as an aid to network users (ref. 9) and no doubt the result of further development will be a kind of automated secretary, assisting its user in many routine tasks and cheap enough to be available to everyone.

The above considerations lead to the idea of a new architecture for networks of the future, illustrated in figure 4, which shows an intelligent terminal accessing a database service (1) or another intelligent terminal (2). However, a third type of connection (3) could be required between two database services. This might be established at the request of their operators, for example, to update any common files, or it might be made following a request of a user wishing for data not held in a local data base. For example, a subscriber to a public data network might call directory enquiries about a subscriber who happened to be located in another country. His local directory could then call for the desired data, on his behalf.

It is in the context of these three types of connection that we examine, in the next section, the new protocols that need to be defined as the basis for future standards. Also, the realisation that figure 4

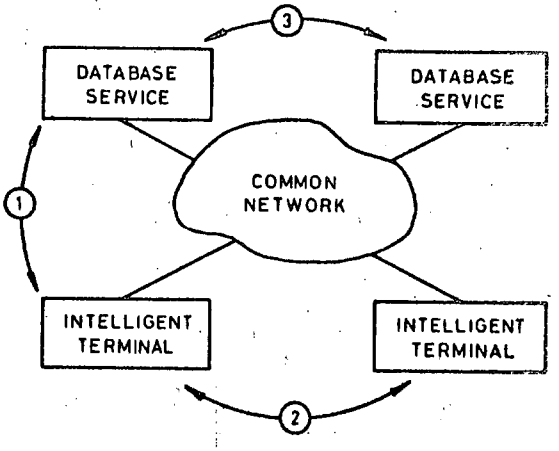


FIGURE 4

represents the network of the future has important implications for the topics that should properly be considered together in a future program of computer network research which we discuss later.

5.o. Protocols of the future

A great deal of effort today is directed towards devising protocols for handling simple 'dumb' terminals; these protocols seek to provide standard ways to cope with arbitrary differences between terminals,e.g. line length, the number of lines on a screen, the delay in returning a carriage, etc. The intelligent terminal requires no such protocols, because it exchanges complete messages with other systems. Sadly however, to interact with todays computers intelligent terminals often have to act 'dumb'.

In a future scheme, using intelligent devices, purely local conventions will suffice for creating, editing and displaying messages in any form required by the user. This will mean that transmitted and received messages will contain no embedded editing commands that have to be passed and obeyed by their recipient, before messages can be considered to decipher their meaning. This is in contrast with, say, the string of characters sent from a typical dumb terminal to an applicationsprogram in a main frame, where several of the characters present in the string may have been introduced by mistyping and its subsequent correction, in any of the ways sommon in present day systems.

What is urgently required for the future, therefore, is a message exchange protocol (refs. lo,ll). Because the terminals using it are intelligent it can be based on different principles from those of existing Virtual Terminal Protocols; furthermore, it needs to be designed with specific applications in mind, and three simple ideas have to be considered. They are:

1. commands and escape sequences should be character strings, not bit strings;
2. message formats and content should be designed for easy comprehension by people;
3. data structures based on human document standards should be adopted.

The adoption of these ideas will simplify the design, understanding and implementation of protocols at the expense of increased use of communication, processing and storage resources. But these all continue to fall in cost measured in real terms, while the value of the people who use them steadily increases.

A protocol designed along these new lines could, for example, support the submission of computer programs to a compiler, as well as the exchange of typical business letters between one company and another. Of course, other protocols will be required for particular applications, and for covering basic kinds of interaction between people, between people and computers, and between computers. But all these could and should be based on the proposed message exchange protocol for intelligent terminals.

There is a grey area where the highest levels of protocol merge into procedures or even into the command languages which allow complex tasks to be carried out. The message exchange protocol must also serve as a vehicle for these, but in the same sense that the 15o standard A4 page now serves as a vehicle for the majority of transactions between people. The information exchanged is limited by the area of the page, but any number of such pages may be classed together. We could therefore hope to see the message exchange protocol featured in future standards discussions and, indeed, the mailbox service now beginning to attract attention of the computer and communication industries will require the agreement of such a protocol.

6.o. A Future Research Program

The foregoing discussions suggest that there are four aspects that are crucial to a program of research in computer networks. The first of these is to keep abreast of developments in microprocessor systems, which are the basis of intelligent terminals as well as of advanced database systems. The second is the organisation and management of intelligent databases implemented on one or more stand-alone large storage systems, possibly using distributed processing methods. The third is the development of new protocols suitable for interactions between two intelligent terminals, and netween such terminals and intelligent databases. The fourth is the study of what interfaces must be presented by new services to their users, to allow ordinary people to make better use of computer systems.

However, there is a fifth aspect of computer networks that has hitherto received scant attention, namely, the question of security of information, and the authentication of users attempting to gain access to it. Developments in this area will also depend on new microelectronic techniques and we would therefore suggest there is a need for a future program of network research in five main associated areas.

6.1. Multi-microprocessor systems

Because the microprocessor is revolutionising computing and networks, its impact must be given cognisance in any research project. Groups of microprocessors cooperating together offer high reliability and increased processing capability, so a multi-microprocessor architecture is likely to be appropriate for most new systems and it is essential to include a study of multi-microprocessor systems in any future informatics research program.

6.2. Intelligent Databases

The cost per unit of data stored is lower in a large store because overheads of all kinds are more widely shared. An informatics network can exploit really large stores by providing access to them from many users. New techniques for stand-alone mass-storage should therefore be considered in the context of future networks, where a high degree of intelligence needs to be built into the store control mechanism. So, special attention should be paid to access problems, protocols, reliability and security aspects, and to the possibilities for evolution and expansion as techniques and user requirements develop.

6.3. Intelligent Protocols

Experience with research networks has shown the need for agreed procedures, known as high level protocols, to facilitate intercommunication between a network, the attached computers and terminals, and the people using them. The earlier research work has led to the proposals recently submitted to standards bodies where activity in this area is rapidly intensifying. Much remains to be done to promote understanding and agreement of new protocols, particularly applications-oriented protocols, rather than the architecturally based protocols studied to date.

6.4. Aids for Users

The growing availability of data networks has allowed access to a variety of computing systems, and has reveled an appaling lack of uniformity in their interfaces to users. The exploitation of informatics networks by people who are not computer specialists depends on the study, agreement and implementation of reasonably simple and uniform interfaces that allow similar tasks to be performed in the same way by all users. An important possible future development is the provision of a directory database that can be accessed by the users' intelligent terminal, in order to load itself with information needed to act on the user's behalf, when interacting with the network facilities he has selected.

6.5. System Integrity and Authentication

The advent of networks has nighlighted the need for reliability both of the networks and of the associated systems and terminals. An unreliable system is an insecure one, so a related aspect is the security of information in the network or in attached databases, and the prevention of its corruption and access by unauthorised parties. This requires a reliable means of user authentication.

These issues are of increasing concern to network users and need to be studied in the context of future informatics systems to allow uniform procedures to be devised. These procedures must ensure that users can interinteract as freely as they desire, without fear of interference or eavesdropping by third parties.

A properly rounded information network research program should study techniques for ensuring reliable and secure transactions and investigate methods of authentication in order to make specific proposals for practical solutions and to develop software and hardware for trials of these solutions with the intention of making available results to users and standards bodies.

Hitherto, system security and user authentication have been almost entirely neglected in the context of informatics networks. But these topics are now being seen as of growing concern to everyone. There is a strong relationship between research in this area all the others proposed. Microprocessors facilitate the use of cryptography; storage system design must take account of the encrypted information; users need to be aware of possibilities and bitfalls, while new protocols must respect the requirements of reliable, secure system design.

7.o. Conclusions

This paper has been presented a case for a a network research program that looks ahead in the light of the impact of present technological developments, so that these can be properly exploited when they mature.

Such a program implies a break with some of the present lines of research as has been discussed, but this is a manifestation of the impact that microelectronics is having on Society in general and merely means that computer scientists, too, must learn to adapt to a more rapidly changing environment.

8.o. References

1. a ARPA Current Network Protocols NIC 7104 Network Information Centre, Stanford Research Institute
1. b The EIN End to End Protocol EIN Report No 76/003
1. c Magnee, F and Endrizzi, A Virtual Terminal Protocols - A Survey, Commission of European Communities Research Contract 881-78-06
1. d A Network Independent File Transfer Protocol INWG Protocol Note No 86 Dec 1977
2. Carr, S et al Host-Host Communication Protocols in the ARPA Network Proc AFIPS Conf. Vol 36 p589 (1970)
3. a CCITT Study Group VII Temp.Doc.39 Raporteurs Meeting on High Level Protocols
3. b ISD IC97 SC16 N34 Washington March 1978
4. Coll J PET 2001 Review Personal Computer World, Vol 1, No2,1978
5. Katz, B J et al 8086 Micro Computer Bridges the Gap between 8 and 16 Bit Designs Electronics, Vol 51, No 4,1978
6. Paterson, J et al The Micro Computer in the Construction Industry Personal Computers in Business, On-Line 1978
7. Todd, S. Hardware Design for High Level Databases Proc. Conf. Computer Systems and Technology IERE, Sussex 1977
8. CCITT Recomendations X3 X25 X28 X29 International Telecommunications Union 1978
9. a Alsberg, P.A. et al, Intelligent Terminals as User Agents Trends and Applications 1976. IEEE 76CH1101-5o, Gaithersberg, MD 1976.
   b Anderson, R.H. Advanced Intelligent Terminals as a User's Network Interface Proc. 11th. IEEE Computer Society Conference, Fall 1975.
10. CCITT S.G. VIII Contribution No 29 Teletex Raporteurs Group November 1977
11. Barber, D.L.A. A Simple Text Transfer Protocol to Aid Network Development EIN internal paper May 1977

INFORMATICA 2/1979

PROGRAM ZA PRORAČUN POUZDANOSTI RAZLIČITIH MEMORIJSKIH SISTEMA

L. BUDIN
Ž. NOŽICA
U. PERUŠKO
D. VUKOVIĆ

UDK: 621.3.019.3 : 681.3.067

ZAVOD ZA ELEKTRONIKU
ELEKTROTEHNIČKI FAKULTET, ZAGREB

U članku se opisuje pristup izradi programa za proračun pouzdanosti različitih memorijskih sistema. Posebna pažnja usmjerena je na memorijske sisteme koji su posebno projektirani da bi zadovoljili visoke zahtjeve za pouzdanošću. Program odredjuje pouzdanost memorijskog sistema polazeći od zadane pouzdanosti blokova memorije. Pouzdanost bloka memorije računa se u programu prema MIL-HDBK-217B.

Reliability Analysis Program for Memory Systems. Although memory elements are greatly expanded in nearly all areas of current digital system design, these components have not yet reached the degree of reliability required in large memory systems. Therefore various schemes that improve reliability of memory systems are developed. In following text computer program, based on developed models and models from MIL-HDBK-217B, that allows comparison of memory systems is described

1. UVOD

Buran razvoj poluvodičkih elemenata visokog stupnja integracije te mikro računarske tehnike, uvjetovao je i nagli rast proizvodnje i potrošnje poluvodičkih memorija, bilo kao operativnih bilo kao masovnih memorija. Interes korisnika da poveća memoriju svog sistema doveo je usprkos svakodnevnom tehnološkom napretku koji uzrokuje pad cijena po bitu do porasta udjela cijene memorije unutar računarskog sistema.

Razvoj tehnologije istodobno je uticao da se glavni problem projektiranja računala pomakae od centralnog procesora k drugim područjima medju kojima je memorija jedan od vrlo značajnih za cijeli sistem. S jedne strane, premda je tehnologija producirala i vrlo brze memorije, nije dala uz ekonomičan i brz procesor i ekonomične i brze memorije. S druge strane, iako su se snažno širile na sva područja projektiranja digitalnih sistema, memorijske komponente nisu dostigle zadovoljavajući nivo pouzdanosti. To posebno vrijedi za neke tehnologije (npr. nabojski vezani elementi) koje su zbog niske cijene vrlo interesantne za izgradnju velikih memorija.

Rezultat nastojanja da se prevlada kako jaz izmedju brzine memorije i brzine obrade tako i uticaj nepouzdanosti memorijskih komponenata na rad cijelog sistema ogleda se u razvoju čitavo niza arhitektura i organizacija memorija. Predmet istraživanja kojima ovaj članak doprinosi su postupci zaštite memorijskog sistema koji mora ostvariti visoku pouzdanost u radu. Tehnike i postupci ostvarivanja takvih sistema razvijaju se niz godina i danas postoji veliki broj sistema koji su, koristeći dodatne sklopove, kodove i/ili programsku podršku, vrlo pouzdani [5-9]. Na osnovi dosadašnjih istraživanja sa sigurnošću se može reći da pogodnost tih sistema vrlo ovisi o veličini memorije, brzini, pouzdanosti kontrolne logike i sl. Program koji se opisuje u ovom tekstu pogodan je za upoređbu memorijskih sistema upravo po tim parametrima, i tako omogućava projektantu lakši izbor optimalnog sistema.

2. PREGLED SISTEMA I MODELA

2.1. Model memorije

Pretpostavke:

(1) Kvar ćelije je tipa s-a-α, $\alpha \in \{0,1\}$.
(2) Memorija se sastoji od k ravnina bita.
(3) Kvarovi se dogadjaju nezavisno.

Iz pretpostavke (3) proizlazi da se i greške kao direktne posljedice kvarova dogadjaju nezavisno. Analizom vjerojatnosti pojave grešaka u grupi bita (riječi) pokazalo se da je distribucija ovih vjerojatnosti binomna. Odatle slijedi da će distribucija vjerojatnosti m grešaka i n maskiranih kvarova u grupi bita imati oblik:

$$\text{binm}(m;P_e,m+n)\cdot\text{binm}(m+n;u,k)$$

Ovdje je P_e vjerojatnost pojave greške na poziciji bita na kojoj postoji kvar. Ova vjerojatnost je veoma ovisna o rasporedu jedinica i nula u memoriji. Analizom uzorka bita u memoriji moguće je odrediti P_{Di}, $i = 1,\ldots,k$. Ukoliko se dogodio kvar, vjerojatnost da taj kvar uzrokuje grešku bit će:

$$P_e = P_{SO}\cdot P_D + P_{S1}\cdot(1-P_D)$$

gdje je radi jednostavnosti pretpostavljeno $P_{Di} = P_D$, za svaki i.

Kako se m grešaka može pojaviti uz bilo koji broj kvarova veći ili jednak m, vjerojatnost m grešaka u grupi bita bit će:

$$\Psi(k,P_e,u,m)=\sum_{i=m}^{k}\text{binm}(m;P_e,i)\cdot\text{binm}(i;u,k)$$

a vjerojatnost m ili manje grešaka u grupi bita:

$$\Theta(k,P_e,u,m)=\sum_{q=0}^{m}\Psi(k,P_e,u,q)$$

Vjerojatnost da sve grupe u memoriji sadrže m ili manje grešaka je:

$$P^*(k,P_e,u,m,w)=[\Theta(k,P_e,u,m)]^w$$

2.2. Modeli pouzdanosti za pojedine memorijske sisteme

2.2.1. Simpleks memorijski sistem (slika 2.)

Pretpostavke:

(1) Greška u bilo kojoj memorijskoj riječi uzrokuje neispravnost memorijskog sistema.
(2) Kvar u kontrolnom sklopu uzrokuje neispravnost memorijskog sistema.

Iz ovih pretpostavki slijedi da je pouzdanost simpleks memorijskog sistema:

$$R_{SMP}= P^*(k,P_e,u,0,w)\ R_m$$

2.2.2. Ispisna memorija (slika 3.)

Ispisna memorija je strukturalno slična simpleks memoriji osim što se jednom upisani podatak u njoj ne mijenja. Izraz za pouzdanost simpleks memorije je stoga primjenjiv i za ispisnu memoriju (ROM), uz dobar izbor P_e. Iz razloga navedenih u [1] odabrano je za ovu analizu P_e=1 za memorije promjenjivog sadržaja i P_e=0.5 za ispisne memorije.

2.2.3. Memorijski sistem sa N-modularnom zalihosti (slika 4.)

Pretpostavke:

(1) Memorija se sastoji od N simpleks memorija vezanih paralelno.
(2) N je neparan, tako da sistem može podnijeti (N-1)/2 grešaka.
(3) Svaki bit pročitane riječi dobija se većinskim nadglašavanjem bita iz N modula.
(4) Greška u kontrolnom sklopu bilo koje od N simpleks memorija uzrokovat će neispravnost memorijskog sistema.

Pouzdanost ovog memorijskog sistema bit će:

$$R_{NMR}= P^*(N,P_e,u,\tfrac{N-1}{2},kw)\cdot P^*(N,P_{ev},u_v,\tfrac{N-1}{2},k)\cdot R_m^N$$

2.2.4. Memorijski sistem sa rezervnim modulima (slika 5,)

Pretpostavke:

(1) Memorija se sastoji od N simpleks memorija sa paralelnim upisom.
(2) Prije upisa podaci se kodiraju kodom za korekciju grešaka i detekciju j grešaka.
(3) Podatak se najprije pročita iz jednog modula i provodi se kontrola grešaka. Ako riječ sadrži ℓ ili manje grešaka, bit će korigirana i prenesena. Ako riječ sadrži više od ℓ a manje od j grešaka, preklopnik izlazne sabirnice će se prekopčati na slijedeći modul i pokušava se čitanje iste riječi iz tog modula.
(4) Kvar u preklopniku izlazne sabirnice, sklopovima za detekciju i korekciju greške ili kontrolnom sklopu uzrokovat će neispravnost pročitane riječi.

Pouzdanost ovog sistema je:

$$R_{SS-N}=\Big\{\sum_{v=0}^{N-\ell}\binom{N}{v}\big[P^*(k,P_e,u,j,1)-P^*(k,P_e,u,\ell,1)\big]^v\big[P^*(k,P_e,u,\ell,1)\big]^{(N-v)}\Big\}^w\cdot R_s\cdot R_e\cdot R_m^N$$

2.2.5. Memorijski sistem sa korekcijom jednostruke greške i rezervnim ravninama bita (slika 6.)

Pretpostavke:

(1) Podatak je kodiran kodom za korekciju jednostruke i detekciju dvostruke greške.

(2) Osim k aktivnih ravnina postoji s rezervnih ravnina bita.

(3) Kvar u kontrolnom sklopu uzrokovat će neispravnost pročitane riječi.

Pouzdanost ovog sistema je:

$$R_{SEC-S} = P^*(k+s,P_e,u,s+1,w)\cdot R_e\cdot R_s\cdot R_m + P^*(k,P_e,u,1,w)\cdot(1-R_s)\cdot R_e\cdot R_m + P^*(k,P_e,u,0,w)\cdot(1-R_e)\cdot R_m$$

2.2.6. Memorijski sistem sa korekcijom M grešaka i rezervnim ravninama bita (slika 7.)

Pretpostavke:

(1) Organizacija ovog sistema je slična prethodnom, SEC-S sistemu, osim što su podaci zaštićeni kodom za korekciju M grešaka.

Pouzdanost ovog sistema je:

$$R_{MEC-S} = P^*(k+s,P_e,u,s+M,w)\cdot R_e\cdot R_s\cdot R_m + P^*(k,P_e,u,M,w)\cdot(1-R_s)\cdot R_e\cdot R_m + P^*(k,P_e,u,0,w)\cdot(1-R_e)\cdot R_m$$

Dosada navedeni sistemi analizirani su detaljnije u literaturi [1], pa su umjesto detaljnog izvoda navedeni samo konačni izrazi za pouzdanost. U analizu iz [1] nije uključen dupleks memorijski sistem pa je ovdje naveden kompletan izvod za njegovu pouzdanost.

2.2.7. Dupleks memorijski sistem (slika 8.)

Pretpostavke:

(1) Dupleks sistem sastoji se od dva simpleks sistema vezana paralelno.

(2) Podaci se upisuju u obje jedinice i zaštićeni su paritetno.

(3) Prilikom čitanja čita se iz obje jedinice i na sabirnicu se propušta podatak iz jedinice koja nije indicirala paritetnu grešku.

(4) Svaka jedinica ima zasebni kontrolni sklop i sklop za kontrolu pariteta.

(5) Svaka jedinica sadrži s rezervnih ravnina bita i sklop za njihovo prekapčanje.

Elementarni dogadjaji:

A - Prva jedinica ne sadrži više od s+1 grešaka, druga jedinica ne sadrži više od s grešaka.

B - Prva jedinica ne sadrži više od s grešaka, druga jedinica ne sadrži više od s+1 grešaka.

C - Sklop za prekapčanje ravnina bita u prvoj jedinici je ispravan.

D - Sklop za prekapčanje ravnina bita u drugoj jedinici je ispravan.

E - Prva jedinica ne sadrži više od jedne greške, druga jedinica ne sadrži niti jednu grešku.

F - Prva jedinica ne sadrži niti jednu grešku, druga jedinica ne sadrži više od jedne greške.

Vjerojatnosti elementarnih dogadjaja:

$P(A) = P^*(k+s,P_e,u,s+1,1)\cdot P^*(k+s,P_e,u,s,1)$

$P(B) = P(A)$

$P(C) = P(D) = R_s$

$P(E) = P^*(k,P_e,u,1,1)\cdot P^*(k,P_e,u,0,1)$

$P(F) = P(E)$

Ove vjerojatnosti izražene su na nivou jedne riječi (w=1), jer se radi o podacima koji se u obje jedinice nalaze na istim memorijskim lokacijama. U slučajevima kad se pojedini složeni dogadjaj odnosi na cijelu memoriju bit će to označeno indeksom w.

Teorem 1.:

A i B nisu disjunktni dogadjaji.

Dokaz:

Dokaz ovog teorema može se provesti na taj način da se pronadje skup $X = A\cap B$.

$A\cap B = \{x\in X \mid$ niti jedna jedinica ne sadrži više od s grešaka$\}$

Teorem 2.:

C i D nisu disjunktni dogadjaji.

Dokaz:

$C\cap D = \{x\in X \mid$ oba sklopa za prekapčanje ravnina bita su ispravna$\}$

Teorem 3.:

E i F nisu disjunktni dogadjaji.

Dokaz:

$E\cap F = \{x\in X$ niti jedna jedinica ne sadrži grešku$\}$

Iz pretpostavke o nezavisnosti kvarova i grešaka mogu se izvesti slijedeći izrazi:

$P(A\cap B) = P(A)\cdot P(B)$

$P(C\cap D) = P(C)\cdot P(D)$

$P(E\cap F) = P(E)\cdot P(F)$

Složeni dogadjaj S1 :

Sklopovi za prekapčanje rezervnih ravnina bita u obje jedinice su ispravni, jedna jedinica ne sadrži više od s+1 grešaka, druga jedinica ne sadrži više od s grešaka.

$$S1 = (A \cup B)_w \cap (C \cap D)$$

$$P(S1) = [P(A)+P(B)-P(A)P(B)]^w \cdot P(C) \cdot P(D)$$
$$= [2P(A)-P(A)^2]^w \cdot P(C)^2$$

$$P(S1) = \{2P^*(k+s,P_e,u,s+1,1)P^*(k+s,P_e,u,s,1) - [P^*(k+s,P_e,u,s+1,1) \cdot P^*(k+s,P_e,u,s,1)]^2\}^w \cdot R_s^2$$

Složeni dogadjaj S2 :

Sklopovi za prekapčanje rezervnih ravnina bita su neispravni bilo u prvoj, drugoj ili u obje jedinice, jedna jedinica ne sadrži više od jedne greške, druga jedinica ne sadrži niti jednu grešku.

$$S2 = (E \cup F)_w \cap (\bar{C} \cup \bar{D})$$

$$P(S2) = [P(E)+P(F)-P(E)P(F)]^w \cdot [1-P(C)P(D)]$$
$$= [2P(E)-P(E)^2]^w \cdot (1-R_s^2)$$

$$P(S2) = \{2P^*(k,P_e,u,1,1) \cdot P^*(k,P_e,u,0,1) - [P^*(k,P_e,u,1,1) \cdot P^*(k,P_e,u,0,1)]^2\}^w (1-R_s^2)$$

Teorem 4.:

Složeni dogadjaji S1 i S2 su disjunktni.

Dokaz:

$$S1 \cap S2 = [(A \cup B) \cap (C \cap D)] \cap [(E \cup F) \cap (\overline{C \cap D})]$$
$$= (A \cup B) \cap (E \cup F) \cap (C \cap D) \cap (\overline{C \cap D})$$

Kako je $X \cap \bar{X} = \emptyset$ i $Y \cap \emptyset = \emptyset$, slijedi

$$S1 \cap S2 = (A \cup B) \cap (E \cup F) \cap \emptyset = \emptyset$$

Složeni dogadjaj S3 :

Memorijski sistem je operativan ako se dogodi S1 ili S2, te ako su istovremeno ispravni sklop za detekciju greške i kontrolni sklop u obje jedinice i sklop za prekapčanje izlazne sabirnice podataka.

$$S3 = (S1 \cup S2) \cap (S_{m1} \cap S_{m2}) \cap (S_{d1} \cap S_{d2}) \cap S_p$$

$$R_{DUP-S} = [P(S1)+P(S2)] \cdot R_m^2 \cdot R_d^2 \cdot R_p$$

gdje je:

$R_m = P(S_{m1}) = P(S_{m2})$ -pouzdanost kontrolnog sklopa

$R_d = P(S_{d1}) = P(S_{d2})$ -pouzdanost sklopa za detekciju greške

$R_p = P(S_p)$ -pouzdanost sklopa za prekapčanje izlazne sabirnice podataka

Dupleks memorija bez rezervnih ravnina bita:

Složeni dogadjaj S4:

Memorijski sistem je operativan ako jedna jedinica ne sadrži više od jedne greške, druga jedinica ne sadrži niti jednu grešku, te ako su istovremeno ispravni kontrolni sklop i sklop za detekciju greške u obje jedinice i sklop za prekapčanje izlazne sabirnice podataka.

$$S4 = (E \cup F)_w \cap (S_{m1} \cap S_{m2}) \cap (S_{d1} \cap S_{d2}) \cap S_p$$

$$R_{DUP} = P(S4) = [P(E)+P(F)-P(E)P(F)]^w \cdot R_m^2 \cdot R_d^2 \cdot R_p$$
$$= [2P(E)-P(E)^2]^w \cdot R_m^2 \cdot R_d^2 \cdot R_p$$

3. PROGRAM ZA RAČUNANJE POUZDANOSTI

Na osnovu opisanih modela izradjen je program za digitalno računalo pomoću kojega se računaju pouzdanosti opisanih memorijskih sistema. Uopćeni dijagram toka programa prikazan je na slici 9. Program je napisan u višem programskom jeziku FORTRAN V i implementiran je u sistemu UNIVAC 1110 Sveučilišnog računskog centra (SRCE) u Zagrebu.

Format podataka na ulaznim karticama je slijedeći:

A	B	C	D	E	F
COM	XV	NB	NWR	BK	NH
A4	A2	I2	I4	A1	I5

Opis podataka po poljima :

A) Naredba ulaznog jezika; može biti jedna od slijedećih:

- POUZ - Izračunavanje pouzdanosti memorijskog sistema sa parametrima iz polja B,C,D,E i F.
- MTBF - Izračunavanje pouzdanosti memorijskog sistema (Mean Time Between Failures - srednje vrijeme izmedju pogrešaka) sa parametrima iz polja B,C,D,E i F.
- ALTR - Paralelni ispis u alternativnu datoteku za štampanje. Ostala polja nisu značajna.
- PRMS - Ulazak u rutinu za promjenu parametara pouzdanosti.
- PLOT - Aktiviranje potprograma za generiranje izlaznih rezultata u formatu pogodnom za grafički prikaz.
- BUGS - Aktiviranje potprograma za praćenje izvodjenja programa. Namjena ovog potprograma je prvenstveno bila za kontrolu rada u toku razvoja programa, ali može poslužiti ako se žele dobiti neki rezultati koji se inače ne štampaju na izlaznoj listi.
- KRAJ - Završetak obrade.

B) Nezavisna varijabla; može biti jedna od slijedećih:

T - vrijeme
U - nepouzdanost memorijske ćelije
RM - pouzdanost kontrolnog sklopa
RE - pouzdanost sklopa za detekciju i korekciju greške

C) Duljina memorijske riječi u bitima
D) Kapacitet memorije u riječima
E) Oznaka jedinice za kapacitet memorije. Ako je u ovom polju upisano K kapacitet će biti prihvaćen u K-riječima (K = 1024 riječi).
F) Vrijeme za koje se računa pouzdanost. Ovo polje nema značaja ukoliko je nezavisna varijabla vrijeme.

Sva polja na ulaznoj kartici su uvjetna osim polja A i B. Program sadrži dva osnovna skupa parametara:

(1) Parametri pouzdanosti. To su frekvencije kvara pojedinih blokova sistema te vjerojatnosti P_{SO}, P_{S1}, P_e i sl.
(2) Parametri memorijskih sistema. To su kapacitet memorije, duljine riječi, broj rezervnih ravnina bita, broj rezervnih modula i sl.

Parametri pouzdanosti mogu se promijeniti naredbom PRMS, dok se parametri memorijskih sistema učitavaju uz naredbe POUZ i MTBF. Ukoliko se polja za parametre ostave prazna, program se izvodi sa slijedećim parametrima:

duljina riječi		
SMP,NMR,ROM		16
SEC-0,SEC-1		22
2EC-0		23
SS-2,DUP-0,DUP-1		17
kapacitet memorije		1K
l,j za SS-2		0,1
λ memorijske ćelije		10^{-9}
λ sklopa za korekciju	0.95	10^{-6}
λ sklopa za prekapčanje	0.15	10^{-6}
λ kontrolnog sklopa	(k+s)0.3	10^{-6}
λ sklopa za glasanje	0.5	10^{-6}
P_e (ROM,NMR)		0.5
P_e (svi ostali)		1.0

Izlazna lista prikazuje pouzdanosti pojedinih memorijskih sistema u zavisnosti od odredjene nezavisne varijable. Rasponi u kojima se kreću pojedine nezavisne varijable su slijedeći:

T - 1 do 20000 sati
U - 10^{-11} do 10^{-2}
RM - 0.82 do 1.0
RE - 0.82 do 1.0

Kako računarski sistema SRCE-a ne uključuje grafičku stanicu, izlazni podaci za grafički prikaz buše se u posebnom formatu na kartice, tako da se mogu posebnim programom iscrtati na crtaču krivulja sistema IBM 1130 (Elektrotehnički fakultet - Zagreb). Program je predvidjen za interaktivni rad na terminalu. U slučaju da se želi ispis rezultata na štampaču, naredbom ALTR postiže se paralelan ispis na terminalu i u alternativnu datoteku za štampanje. Ova datoteka se po završetku programa upućuje na štampanje.

4. RAČUNANJE FREKVENCIJA KVAROVA BLOKOVA NA OSNOVU PRIRUČNIKA MIL-HDBK-217B

Program opisan u prethodnom poglavlju računa pouzdanost memorijskih sistema na osnovu frekvencija kvara pojedinih blokova sistema koje su zadane kao parametri. Ovi parametri se moraju ili pretpostaviti ili izračunati. Da bi se izračunala frekvencija kvara bloka koji se obično sastoji od većeg broja raznovrsnih komponenti, potrebno je:

(1) Sastaviti listu komponenti bloka čiju frekvenciju kvara odredjujemo.
(2) Generirati podatke zavisne o primjeni (radna temperatura, okolina, naprezanja i nivo kvalitete).
(3) Obezbijediti informacije o svakoj komponenti kao što su kompleksnost integriranog sklopa i granice maksimalnog naprezanja.
(4) Izvršiti proračun frekvencije kvara prema tablicama iz MIL-HDBK-217B [10].

Da bi se ovaj naporni posao olakšao izradjen program za digitalno računalo koji zamjenjuje korak (4) ovog postupka. Ovaj program je tek u početnoj fazi, tako da zasada računa frekvencije kvara integriranih elektroničkih sklopova, a daljnji razvoj predvidja uključivanje i ostalih komponenti zastupljenih u MIL-HDBK-217B. Također je u planu uspostavljanje datoteke sa kataloškim podacima o pojedinim komponentama. Na taj način bi se i korak (3) zamijenio automatskim pretraživanjem kataloga.

Model za računanje frekvencije kvara integriranog sklopa prikazan je izrazom:

$$\lambda_P = \pi_L \cdot \pi_Q \cdot (C_1 \cdot \pi_T + C_2 \cdot \pi_E)$$

gdje je :

λ_P - frekvencija kvara integriranog sklopa izražena u $1/10^6$ sati
π_L - faktor učenja komponente (izražava tzv. početnu nepouzdanost kod komponenti u početnoj proizvodnji)
π_Q - faktor kvalitete integriranog sklopa

π_T - temperaturni faktor, ovisi o tehnologiji proizvodnje integriranog sklopa

π_E - faktor okoline u kojoj se integrirani sklop primjenjuje

Faktori $\pi_L, \pi_Q, \pi_T, \pi_E$, C_1 i C_2 dobiju se iz tablica MIL-HDBK-217B.

Dijagram toka programa prikazan je slikom 10. Format ulaznih podataka je slijedeći:

A	B	C	D	E	F	G	H	I
A3	A1	I8	A2	A1	A2	F4.0	I3	I10

Značenja pojedinih polja su:

A) Tip integriranog sklopa:
- DMS - digitalni, niskog ili srednjeg stupnja integracije
- DLS - digitalni, visokog stupnja integracije
- RAM - memorija
- ROM - ispisna memorija
- LIN - linearni

B) Tehnologija proizvodnje sklopa:
- A - TTL i DTL
- B - bipolarni i MOS linearni, bipolarni ECL i "beam lead", ostali MOS sklopovi

C) Broj logičkih sklopova

D) Nivo kvalitete komponente:
- AO - Mil-M-38510, klasa A (JAN)
- BO - Mil-M-38510, klasa B (JAN)
- B1 - MIL-Std-883, metoda 5004, klasa B
- B2 - Vendor ekvivalent Mil-Std-883, metoda 5004, klasa B
- CO - Mil-M-38510, klasa C (JAN)
- DO - komercijalni sklopovi, izvan vojnih standarda

E) Faktor učenja

F) Radna okolina:
- GB - zemaljski, laboratorijski uvjeti
- SF - svemirski let
- GF - zemaljski nepokretni
- AI - zrakoplovstvo, područje s posadom
- NS - mornarica
- GM - zemaljski, pokretni
- AU - zrakoplovstvo područja bez posade
- NU - mornarica, otežani uvjeti
- ML - satelitske i raketne primjene

G) Radna temperatura

H) Broj izvoda na kućištu integriranog sklopa

I) Broj komponenti u bloku

Izlazna lista sadrži preglednu tabelu svih komponenti i parametara sa pojedinačnim frekvencijama kvara. Na kraju tabele štampa se ukupni broj komponenti i ukupna frekvencija kvara bloka.

5. ZAKLJUČAK

Dosadašnja istraživanja na području ostvarivanja pouzdanog rada memorijskih sistema rezultirala su u velikom broju različitih koncepcija Prilikom odabiranja odredjenog sistema mora se voditi računa o odnosu veličina cijene, brzine i pouzdanosti koje karakteriziraju sistem. Da bi se odabrao optimalni sistem s obzirom na pouzdanost, mora se odrediti uticaj takvih parametara memorije kao što su npr. dužina memorijske riječi, veličina memorije, pouzdanost upotrebljenih elemenata, očekivane dužine rada i sl. [1 - 4].

Programi koji su opisani omogućuju analizu uticaja pojedinih parametara memorije na pouzdanost cijelog sistema. Programi u računanju pouzdanosti memorijskog sistema polaze od modela koji su izvedeni iz pouzdanosti pojedinih blokova memorije. Pouzdanost pojedinog bloka memorije računa se prema priručniku MIL-HDBK-217B.

DODATAK : Objašnjenje pojmova i oznaka korištenih u tekstu

kvar - odstupanje fizikalnih karakteristika komponente od karakteristika odredjenih početnim dizajnom

greška - Logička varijabla čija se vrijednost razlikuje od dizajnom predvidjene vrijednosti, uslijed djelovanja kvara

maskirani kvar - logička varijabla čija se vrijednost ne razlikuje od dizajnom predvidjene vrijednosti, iako je na nju djelovao kvar

grupa bita - skup bita unutar kojega promatramo vjerojatnost pojave greške u datom trenutku, Kod NMR sistema to je skup bita koji se dovode na ulaze sklopa za glasanje a kod ostalih je to memorijska riječ.

s-a-α - (engleski: stuck-at-α), tip kvara kada logička varijabla zadržava trajno istu vrijednost $\alpha, \alpha \in \{0,1\}$.

$\text{binm}(i;p,N) = \binom{N}{i} p^i \cdot (1-p)^{N-i}$

P_{Di} - dio memorijskih riječi koje na poziciji i-tog bita sadrže jedinicu.

k - duljina memorijske riječi

w - kapacitet memorije u riječima

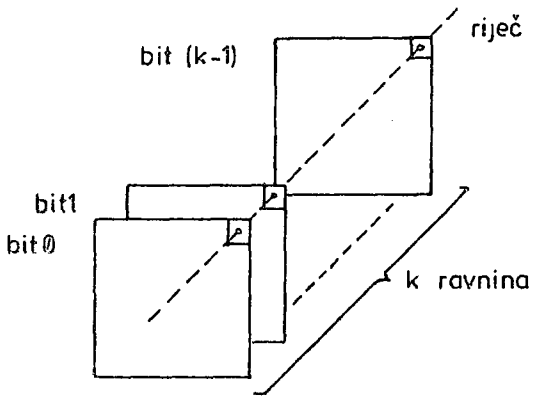
P_{S0}, P_{S1} - vjerojatnosti kvarova s-a-0 i s-a-1

u - nepouzdanost ćelije

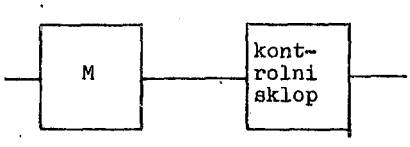
u_v - nepouzdanost sklopa za glasanje

s - broj rezervnih ravnina bita

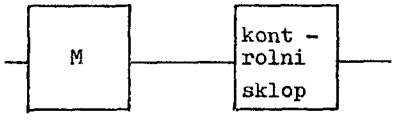
λ - frekvencija kvara



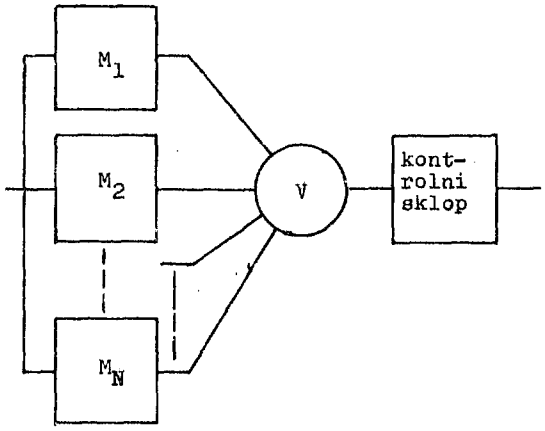
Slika 1.: Organizacija memorije



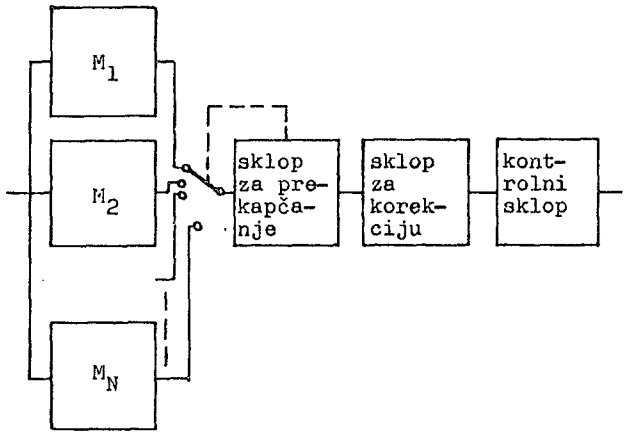
Slika 2.: Model pouzdanosti simpleks memorijskog sistema (SMP)



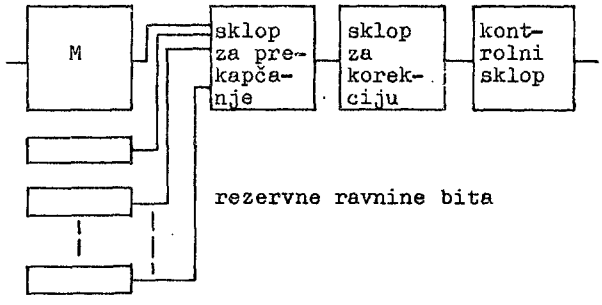
Slika 3.: Model pouzdanosti ispisne memorije (ROM)



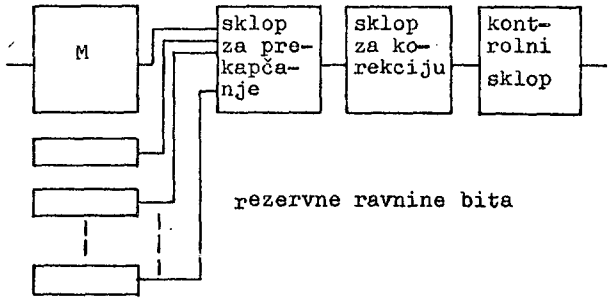
Slika 4.: Model pouzdanosti memorijskog sistema sa N-modularnom zalihosti (NMR)



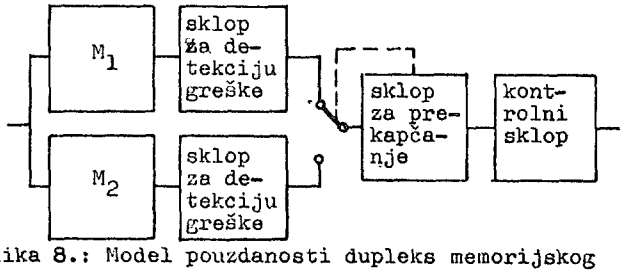
Slika 5.: Model pouzdanosti memorijskog sistema sa rezervnim modulima (SS-N)



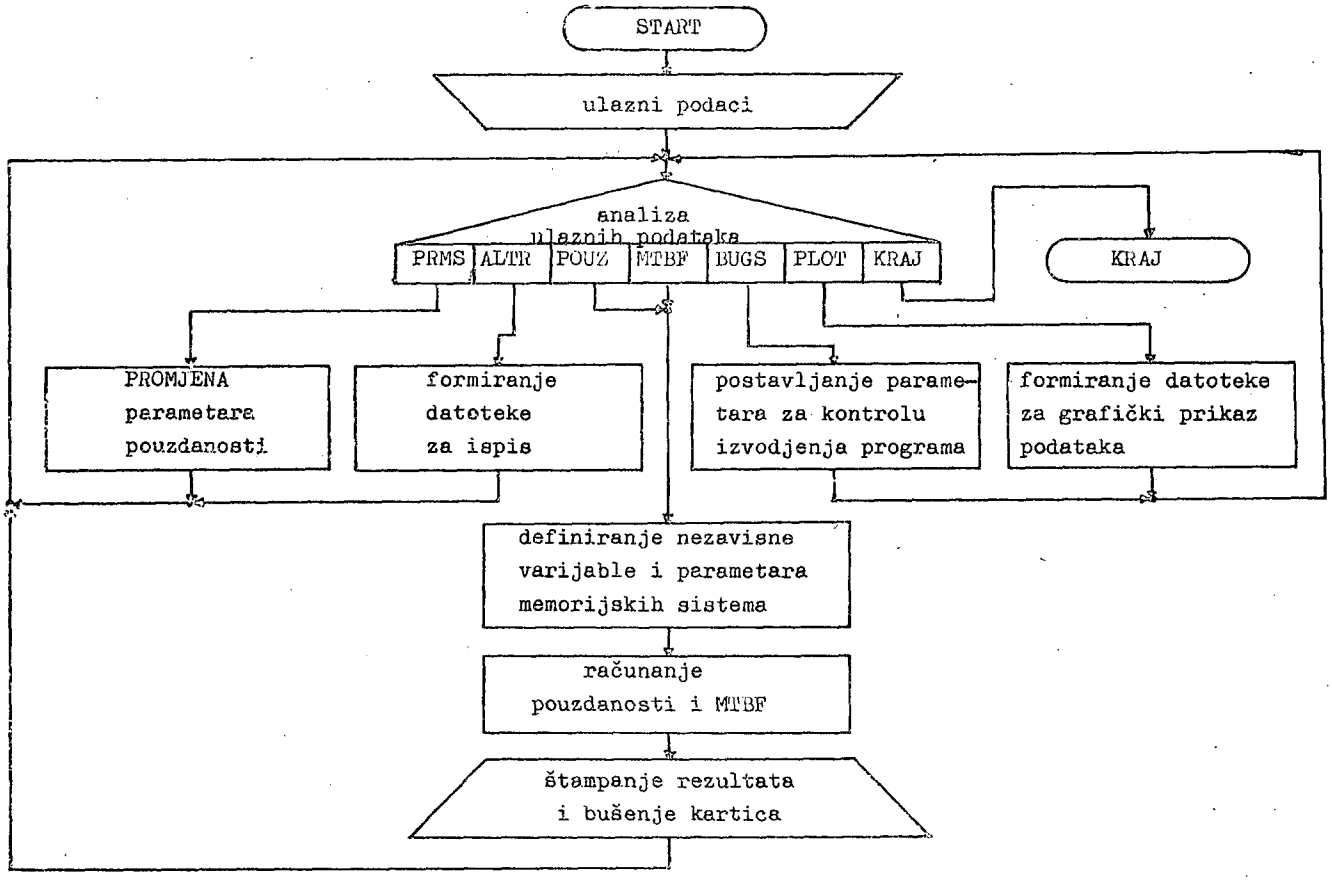
Slika 6.: Model pouzdanosti memorijskog sistema sa korekcijom jednostruke greške i rezervnim ravninama bita (SEC-S)



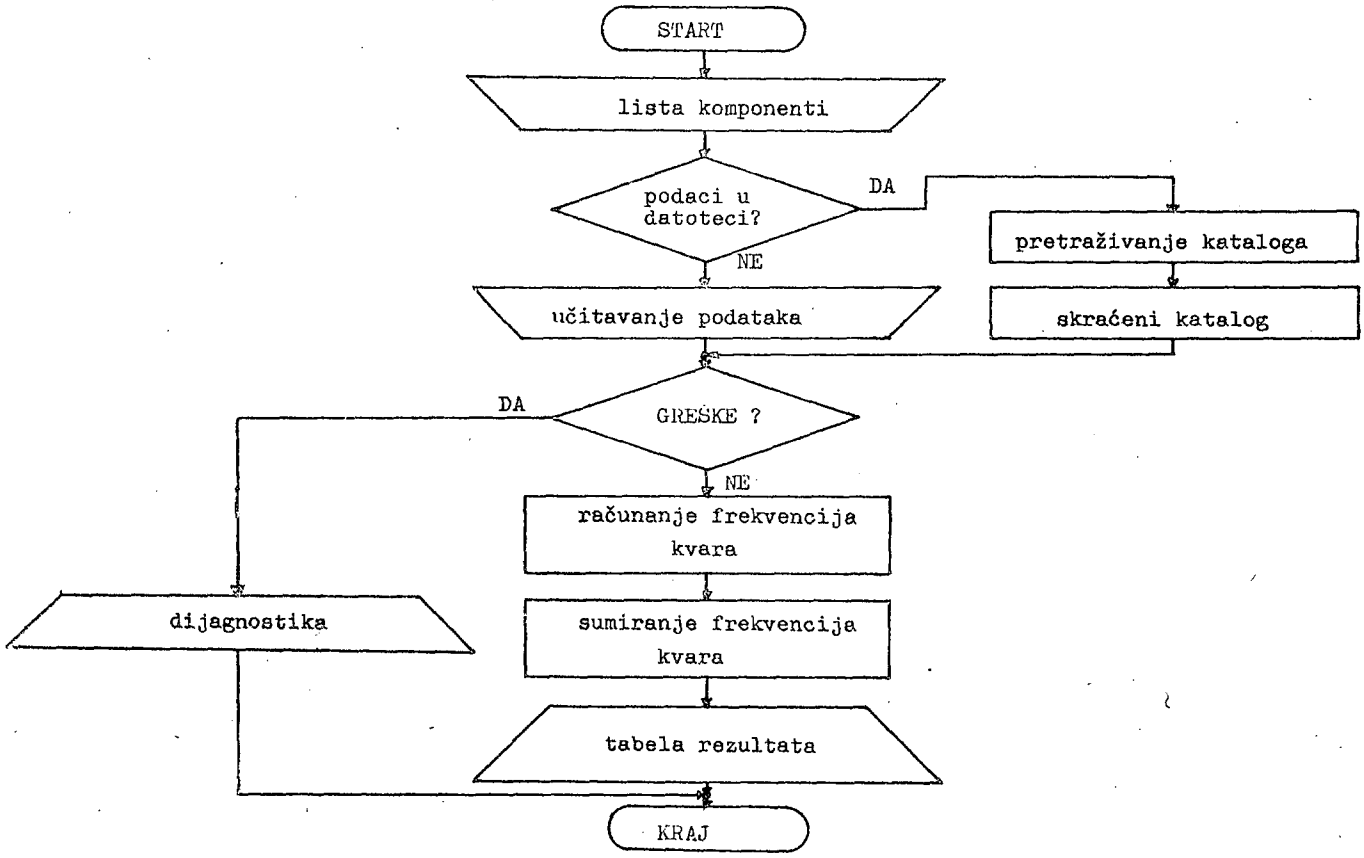
Slika 7.: Model pouzdanosti memorijskog sistema sa korekcijom M grešaka i rezervnim ravninama bita (MEC-S)



Slika 8.: Model pouzdanosti dupleks memorijskog sistema (DUP-S)



Slika 9.: Dijagram toka programa za računanje pouzdanosti memorijskih sistema



Slika 10.: Dijagram toka programa za računanje frekvencija kvara blokova

Literatura

1 Glen W.Cox,B.D.Carroll,"Reliability Modeling and Analysis of Fault-Tolerant Memories,"IEEE Trans.Reliability,vol R-27,1978 April,pp 49-54.

2 D.Siewiorek,S.Elkind,"The Effect of Semiconductor memory chip failure modes on system reliability and performance,"1978 International Symposium on Fault-Tolerant Computing,Toulouse,France,1978 June 21-23, pp 150-155.

3 D.Siewiorek et al.,"A Case study of C.mmp, Cm,and C.vmp:Part II-Predicting and calibrating reliability of multiprocessor systems,"Proceedings of the IEEE,vol.66, No.10,1978 Octobar,pp 1200-1220.

4 L.Levine,W.Meyers,"Semiconductor memory reliability with error detecting and correcting codes,"Computer,vol 9,No.10,1976 October,pp 43-50.

5. W.T.Hartwel,C.W.Hoffner,W.N.Toy,"A fault-tolerant memory for duplex systems," IEEE Trans .Reliability,vol R-27,1978 April,pp 134-138.

6. J.Goldberg et al.,"An organization for a highly survivable memory,"IEEE Trans. on Computers,vol C-23,No.7,July 1974,pp 693-705.

7. W.C.Carter,C.E.Mc Carthy,"Implementation of an experimental fault-tolerant memory system,"IEEE Trans.on Computers,vol C-25 , No.6,June 1976,pp 557-568.

8. W.C.Carter at al.,"Lookaside techniques for minimum circuit memory translators,"IEEE Trans.on Computers ,vol C-22,No.3,March 1973 pp 283-289.

9. W.G Bouricius at al,"Modeling of a bubble memory organization with self-Checking translators to achieve high reliability,"IEEE Trans on Computers,vol C-22, March 1973,pp 269-275

10. Mil-Std-Hdbk-217B, Military Standardization Handbook: Reliability Prediction of Electronic Equipment, September 1974.

SOFTVERSKI SISTEMI ZA SIMULACIJU MULTI–MIKROPROCESORSKIH RAČUNARA

V. OTOVIĆ

SOUR „RUDI ČAJAVEC", RO PROFESIONALNE ELEKTRONIKE
OOUR RAČUNARSKO RADARSKE TEHNIKE, BEOGRAD, JUGOSLAVIJA

UDK: 681.3.012 : 519.682.6

Ovaj rad izlaže probleme i specifičnosti vezane za softversku simulaciju multi-mikroprocesorskih računara, kao i strategiju projektovanja ovakvih sistema. Izloženi su osnovni elementi i moduli jednog ovakvog sistema kao i funkcije i karakteristike svakog od modula. Razmatran je i problem medjusobne sprege modula, kako u okviru samog sistema za simulaciju tako i sa stanovišta procesa koji se odvijaju tokom simulacije na "host" računaru. Kao moguća rešenja dato je par konfiguracija koje su uslovljene složenošću sistema i mogućnošću "host" računara.

SOFTWARE SYSTEMS FOR MULTI-MICROPROCESSOR COMPUTER SIMULATION: This paper presents the problems associated with the design of multi-microprocessor computer simulators, as well as the general modeling strategie for such systems. Basic elements and modules are described, as well as the characteristics and functions of each module. A discussion of module linkage, from the point of view of the processes taking place on the host computer, is also given. A few configurations are presented as possible solutions, all of which are a function of the system complexity and host computer capabilities.

UVOD

Poslednjih nekoliko godina mogućnosti mikroprocesora su se znatno povećale, a samim tim i polje njihove primene. Medjutim, i pored znatnih poboljšanja, neke od primena su bile preambiciozne za samo jedan mikroprocesor. Da bi se zadovoljile potrebe počinje se sa ugradnjom više mikroprocesora u sisteme, a u poslednje vreme se na tržištu pojavljuju pravi multi-mikroprocesorski (MMP) računari.

Očigledno da se odmah javlja problem analize rada ovih sistema, kao i problem projektovanja i provere softvera za MMP računare. Sistemi za modeliranje za MMP računare još ne postoje. Medjutim, i da postoje ne bi bili od veće koristi za detaljniju analizu rada. Klasičan sistem za modeliranje bi možda mogao da pruži potrebnu kontrolu, ali ne i obilje statistika potrebnih da bi se dobila jasna slika o radu sistema. Pored toga sistem za modeliranje bi ograničio korisnika na jednu konfiguraciju.

Rad je referiran na XIV. Simpoziju Informatica 79, Bled, 1-6. oktobra 1979.

U takvoj situaciji najlakši izlaz je pribeći softverskoj simulaciji sistema. Softverski sistem bi obezbedio potrebnu kontrolu, omogućio prikupljanje statistika, i ne bi vezao korisnika za samo jednu konfiguraciju. Obezbedilo bi se relativno lako i efikasno projektovanje sistema u koji MMP računar treba da se ugradi.

KARAKTERISTIKE SISTEMA ZA SIMULACIJU

Prilikom projektovanja softverskih sistema za simulaciju MMP računara moguće je voditi se nekim uopštenim strategijama i metodama. Takav jedan sistem bi trebalo da poseduje karakteristike koje omogućuju što efikasniju upotrebu. Zahtevi na koje sistem treba da odgovori kao i neka pravila prilikom projektovanja, u osnovi definišu karakteristike sistema za simulaciju. Osnovne karakteristike ovakvog sistema su:

1) Sistem treba da bude projektovan modularno sa precizno definisanim ulazima (izlazima)

2) Sistem treba da ima mogućnost da se što je moguće lakše prilagodjava novoj konfiguraciji MMP računara.
3) Skup komandi sistema za simulaciju mora da pruži punu kontrolu nad simulacijom.
4) Sistem za simulaciju mora da obezbedi sve potrebne statistike o radu svih mikroprocesora kao i računara u celini.
5) Radi što veće kontrole sistem mora da bude interaktivan.

MODULI

Potreba za izmenama u sistemu za simulaciju se može javiti usled rekonfiguracije MMP računara kao i usled nekih specifičnosti novog "host" računara na koji se sistem za simulaciju ugradjuje. Da bi se omogućila što lakša izmena, sistem se projektuje modularno. Moduli se organizuju u logičke celine i to tako da se veza izmedju modula svede na minimum neophodnih podataka. Ako je ovaj "interfejs" modula precizno definisan, bilo kakve izmene u okviru jednog modula ne bi imale efekta na ostale module u sistemu, pod pretpostavkom da se "interfejs" ne menja. U daljem tekstu su date sve karakteristike modula kao i definicije njihovih ulaza/izlaza.

Osnovni moduli sistema za simulaciju su:
1)Ulazni izlazni (U/I) modul
2)Modul (P) za obradu komandi
3)Kontrolni (K) modul
4)Modul (M) simulatora mikroprocesora
5)Statistički (S) modul
6)Modul (D) podataka

Ulazno izlazni modul obezbedjuje komunikaciju izmedju korisnika i sistema. Modul P vrši prevodjenje upravljačkih komandi sistema. Kontrolni K modul sinhronizuje rad M modula, periferije i memorije u skladu sa njihovom konfiguracijom u MMP računaru i obezbedjuje simulaciju realnog vremenskog preklapanja dogadjaja. Simulator mikroprocesora, M modul, simulira skup instrukcija mikroprocesora. Statistički S modul prikuplja i raspolaže svim podacima o stanju računara. Modul podataka (D) nije izvršni modul već je modul koji sadrži sve elemente računara i njihova stanja.

U/I MODUL

Ulazno izlazni modul obezbedjuje komunikaciju izmedju korisnika i sistema za simulaciju. Neposredna komunikacija sa korisnikom se odvija preko terminala "host" računara. Ulaz se koristi za unošenje upravljačkih komandi. Ovaj ulaz je relativno jednostavan jer format upravljačkih komandi može unapred da se odredi. Nakon unošenja, komanda se u izvornom formatu prosledjuje modulu za obradu komandi (P mod.). Izlaz se koristi za prikazivanje stanja elemenata u MMP računaru, i zato je znatno složeniji s obzirom na veliki broj mogućih izlaznih podataka i potrebu za njihovim reformatiranjem prilikom prikazivanja.

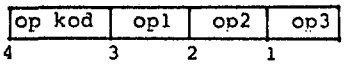
Izlazne podatke treba grupisati u niz podmodula. Svaki podmodul bi sadržao informaciju o tipu i formatu podatka koji treba da prikaže, kao i sve potrebne izlazne instrukcije. Izlaz bi se jednostavno inicirao pozivanjem odredjenog podmodula. Kako podmoduli mogu medjusobno da se pozivaju, kombinacije izlaznih podataka bi bile izuzetno velike. Grupisanje izlaznih podataka i medjusobno pozivanje podmodula se odredjuje unapred ali se isto tako može veoma brzo promeniti ako se U/I modul organizuje na ovaj način.

P MODUL

Modul P vrši prevodjenje izvornih upravljačkih komandi. Upravljačke komande omogućuju upravljanje sistemom, ispitivanje i modifikaciju stanja svih elemenata u MMP računaru. Modul P prima komande od U/I modula u nekom od alfanumeričkih kodova (ASCII, EBCDIC), zavisno od "host" računara. Moguć format izvorne komande je dat u slici 1.a.

komanda / op1 / op2 / op3

Slika 1.a Izvorna komanda (80 bajta)



Slika 1.b Prevedena komanda (4 bajta)

Komanda se zatim prevodi u pogodniji format, koji ostali moduli prepoznaju. Moguć format prevedene komande je dat u slici 1.b. Za prevodjenje komande mogu se koristiti standardne metode upotrebljene u jezičkim prevodiocima. S obzirom da je skup komandi relativno mali i da je format praktično isti za sve komande, te metode se mogu znatno uprostiti i svesti na najneophodnije operacije potrebne za prevodjenje. Izlaz iz P modula je komanda u formatu

datom u slici 1.b. Komanda se prosledjuje ka S ili K modulu, zavisno od toga da li je komanda aktivna ili pasivna. Pod pasivnom komandom se podrazumevaju komande koje ne aktiviraju simulator računara (K i M module), već samo menjaju ili prikazuju stanje pojedinih elemenata u računaru. Ove komande se prosledjuju ka S modulu. Pod aktivnom komandom se podrazumevaju komande koje aktiviraju simulator računara. Ove komande se prosledjuju ka kontrolnom K modulu. Tipičan skup komandi za jedan ovakav sistem je dat u tabeli 1.

TABELA 1

Tipične komande sa primerima mogućih polja operanda

Komanda	op1	op2	op3	Tip	Opis
RUN	A	-	-	A	Startuj procesor A
HALT	B	-	-	A	Zaustavi procesor B
LOAD	-	-	-	A	Puni memoriju
DISPLAY	M	0100	010F	P	Prikaži sadržaj mem. od 0100_{16} do $010F_{16}$
MODIFY	M	051A	EFFF	P	Promeni sadržaj mem. lokacije 051A u EFFF
STAT	-	-	-	P	Prikazivanje željenih statistika o radu

Tip: A - aktivna komanda
P - pasivna komanda

K MODUL

Kontrolni K modul je verovatno najvažniji modul u čitavom sistemu. Modul K objedinjuje i sinhronizuje rad svih simulatora mikroprocesora, odnosno M modula, kao i svih periferika. Razlučuje memorijske konflikte izmedju M modula onako kako bi to stvarni sistem činio. Najvažniji zadatak K modula je da sinhronizuje procese tokom simulacije, odnosno, da daje osnovni takt i da simulira vremensko preklapanje istovremenih procesa. Kod "sinhronih" računara (računari sa istim mikroprocesorima i istim osnovnim taktom) problem se može rešiti simulacijom osnovnog takta u K modulu. Svaki n-ti takt bi se redom dodeljivao prvom, drugom pa i-tom M modulu. Zatim bi se proces ponovio za n+1, n+2 takt i tako redom.

Za "asinhrone" MMP računare(računare sa različitim mikroprocesorima i različitim taktom) problem je nešto složeniji i može se rešiti programiranjem u realnom vremenu. K modul u tom slučaju mora da sadrži planer procesa koji će preko časovnika stvarnog vremena da inicira procese u odgovarajuće vreme. S obzirom da će simulator verovatno biti sporiji od stvarnog sistema, sva vremena se moraju skalirati kako bi relativan odnos izmedju procesa ostao isti. K modul mora da poseduje i podmodul za prihvatanje komandi koje dolaze iz P modula. Ovaj podmodul treba da inicira izvršenje komande i da ustanovljava kraj izvršenja komande.

M MODUL

Modul M simulira izvršenje kompletnog skupa instrukcija za dati mikroprocesor u sklopu MMP računara. U MMP računaru koji ima iste mikroprocesore poželjno je raditi ovaj modul u "reentrant" kodu kako bi se izbeglo dupliciranje M modula za svaki mikroprocesor u računaru. Kod MMP računara sa različitim mikroprocesorima potreban je po jedan M modul za svaki mikroprocesor u sistemu. Sinhronizovan rad svih M modula u sistemu kontroliše K modul.

S MODUL

Statistički (S) modul raspolaže svim podacima o radu računara kao i podacima o stanju svih elemenata u računaru. S modul izvršava pasivne komande, odnosno, prikazuje ili menja stanje elemenata u računaru. Ovaj modul ima pristup svim podacima u D modulu, i preko U/I modula inicira prikazivanje elemenata iz D modula. Prikazivanje se inicira time što se U/I modulu prenosi informacija o broju podmodula koji vrši izlaz,zajedno sa podatkom koji treba da se prikaže. Kombinacijom brojeva podmodula iz U/I modula, S modul može da daje i različite kombinacije izlaznih podataka.

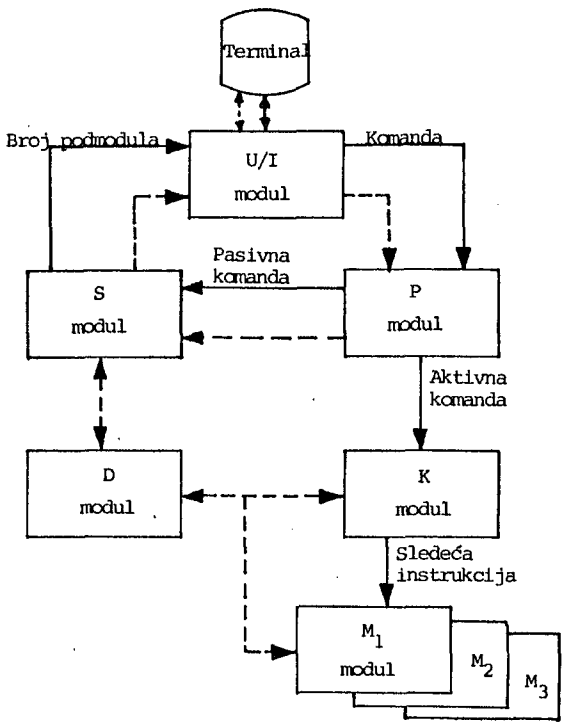
D MODUL

Modul podataka (D) nije izvršni deo sistema za simulaciju već je samo blok zajedničkih podataka. Ovaj modul sadrži sve iformacije o stanju elemenata u računaru. U D modulu su podaci o stanju sadržaja memorije, registara, brojača lokacija, ulazno izlaznih registara i raznih kontrolnih signala.

KONFIGURACIJA SISTEMA ZA SIMULACIJU

Slika 2 prikazuje blok-šemu sistema za simulaciju MMP računara. Isprekidanim linijama je prikazan tok podataka, a punim linijama tok kontrole. Kao što se vidi, komande sa terminala se prosledjuju P modulu na prevodjenje. U slučaju da je komanda pasivna, P modul je šalje S modulu na izvršenje. Zavisno od komande, S

modul ili menja stanje u računaru ili poziva U/I modul radi prikazivanja podataka na terminalu. U sličaju aktivne komande P modul je prosledjuje kontrolnom, K, modulu.



Slika 2 Blok-šema sistema

K modul aktivira M module, sinhronizuje njihov rad i proverava uslove završetka tekuće komande. Po završetku komande deaktivira sve M module i čeka novu komandu. Da bi simulacija bila što realnija izuzetno je važna konfiguracija celokupnog sistema sa stanovišta procesa koji se odvijaju na "host" računaru. Najjednostavnija konfiguracija je kada je kompletna šema data u slici 2 predstavljena kao jedan jedinstven proces na "host" računaru. Kontrola teče iz jednog u drugi modul i korisnik nama mogućnost da utiče na simulaciju dokle god se kontrola ne vrati u U/I modul i omogući mu da unese sledeću komandu. To bi ograničilo stepen kontrole i znatno umanjilo realnost simulacije. Pored toga moglo bi da se dodje u situaciju da vreme odziva sistema postane nedozvoljeno dugo.

U slučaju da "host" računar ima mogućnost multiprogramiranja i komunikacije izmedju procesa, sistem se može organizovati u daleko povoljniju konfiguraciju. Moduli K, M i D predstavljaju sistem koji se simulira dok moduli U/I, P i S su na neki način upravljački moduli. Mnogo povoljnija konfiguracija bi bila kad bi se sistem organizovao u te dve logičke celine kao dva različita procesa na "host" računaru, koji se odvijaju paralelno, ali koji i komuniciraju. Stepen slobode prvog procesa (K,M i D) bi sada bio daleko veći. Mogućnost simulacije prekida i raznih drugih stohastičkih procesa, koji utiču na rad računara, bi bila daleko veća od mogućnosti koje pruža prva konfiguracija. Ovakva konfiguracija bi znatno iskomplikovala P i K module zbog mehanizma izmene poruka (komandi) ali bi se dobilo u realnosti simulacije.

Mehanizam izmene poruka (komandi) se može organizovati na više načina. Moguće je da P modul prekida K modul, nakon čega bi K modul primao poruku. Druga mogućnost je da P modul ostavlja poruku u nekoj zajedničkoj lokaciji u memoriji koju bi K modul povremeno ispitivao. Mehanizama ima raznih,ali su obično prilikom projektovanja sistema ograničeni mogućnošću računara na kojem se sistem projektuje.

ZAKLJUČAK

Opisana je jedna moguća strategija projektovanja softverskog sistema za simulaciju MMP računara. Dati su osnovni elementi i karakteristike takvog sistema. Projektovanje konkretnog sistema je složen posao i optimalnost jednog takvog sistema je kompromis izmedju zahteva, ekonomičnosti, složenosti MMP računara i mogućnosti računara na kojem se sistem projektuje. Modularni pristup, opisan u ovom radu, pokušava da generalizuje ovakve sisteme i da pruži odredjenu fleksibilnost i preglednost u projektovanju. Samostalnost modula, kao i precizno definisani ulazi/izlazi modula daju sistemu odredjenu fleksibilnost i mogućnost rekonfiguracije kako bi se zadovoljili zahtevi projektovanja konkretnog sistema.

BIBLIOGRAFIJA

1. Enslow H.P. : Multiprocessors and Parallel Processing John Wiley, N.Y., 1974
2. Martin F.F. : Computer Modeling and Simulation John Wiley, N.Y., 1968
3. Deland C.D., Bekey G.A. : Interactive Computer Simulation, Instruments and Control Systems, Oktober,1972
4. Shore J.E. : Software Simulation of an Associative Processor, N.R.L.,Washington D.C.,December,1971
5. Mueller R.A.,Johnson G.R. : Generator for Microprocessor Assemblers and Simulators, Proc. of IEEE, vol. 64, No. 6, June,1976

KONVOLUCIJSKI KODIRNI IN DEKODIRNI POSTOPEK PRI MIKRORAČUNALNIKU ISKRA DATA 1680

M. KAPUS
G. DEVIDE
B. HORVAT

UDK: 681.325.3

VISOKA TEHNIŠKA ŠOLA, MARIBOR

POVZETEK - Brez dvoma je eden najmodernejših in atraktivnih dekodirnih postopkov v prenašanju in zavarovanju digitalne informacije Viterbijev algoritem. V naslednjem desetletju pričakujemo v komunikaciji podatkov ta algoritem realiziran v integrirani obliki s pomočjo materialne opreme. Zaradi visoke redundance se tak postopek koristi predvsem pri prenosu podatkov FEC brez povratnega komunikacijskega kanala. Pri komuniciranju med procesorji in v želji po čim večji zanesljivosti smo koristili ta kodni in dekodni postopek in ga realizirali s pomočjo programske opreme.
Kodirnik je preprost pomikalni register povratno povezan s seštevalniki po m = 2. Dekodiranje je bolj zapleteno in je bistvo članka. Simulirali smo celoten prenosni sistem za ocenitev prenosa. Komunikacijski kanal smo ponazorili kot normalno porazdelitev z generatorjem naključnih števil. Rezultati so obetavni in kažejo na veliko zanesljivost prenosa kljub množici motenj v kanalu.

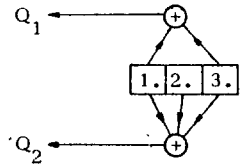
CONVOLUTIONAL CODING AND DECODING PROCEDURE BY MICROCOMPUTER ISKRA DATA 1680 - Without doubt is in communication the Viterbi algorithm one of the up to date decoding procedure. In next decade we expect that the procedure be realized in integrated form on the chip. The procedure take more use in FEC transmission and need for deplexing the same number information as control bits. By communication between processors to achieve more transmission reliability we use this coding and decoding procedures with software suport.
The coder is simple shift register feedback conected with exlcusive or elements. The decoding procedure is more complex and the mathematical structure which we take is explained in this article. We simulate the communication channel with a random number generator, to value the transmission system. The decoding procedure has shown succesfull transmission with a great degree of reliability on channel errors.

1. Uvod

Če želimo dvigniti zanesljivost prenaša nih sporočil v digitalnih sistemih, pri katerih ni povratnega kanala, se odločamo najpogosteje za popravljanje napak na sprejemni strani. Tak način prenosa zahteva dokajšno redundanco prenešenega sporočila v obliki kontrolnih bitov, ki jih privesimo informacijskim bitom. Ravno tako moramo računati s povečanim procesorskim časom in povečano pomnilniško kapaciteto. Želeli smo ustvariti poseben minimalni procesorski modul, ki bi bil kot komunikacijski procesor priključen na večji računalniški sistem in bi tako zagotavljal povečano zanesljivost vsem onim vhodno/izhodnim vodilom, kjer ni možnosti za povraten kanal, torej za prenos ARQ sistema. Takšni komunikacijski kanali so pogosto vsi prenosi podatkov na pomnilniške medije in vse prenosne poti, kjer pogosto ponavljanje informacijskih paketov ne bi zavrlo prepustnost informacijskega pretoka. Razumljivo je, da komunikacijska pot mora imeti določeno kvaliteto in nenavadne neprevelike aditivne napake kanala nam dekodirni proces izloči. Sam kodirni postopek je zelo preprost in ne zahteva procesorskega časa in ga najpreprosteje rešimo v materialni opremi. Dekodirni proces je mnogo bolj zamotan in podana je njegova programska rešitev. Izbrali smo minimalno konfiguracijo registrov in si s tem zagotovili manjši čas procesiranja in manjšo pomnilniško kapaciteto.

2. Opis kodirnega in dekodirnega algoritma

Pri sinhronem prenosu sporočil nimamo povratne kontrole, ali je bil prenos pravilen. Zato moramo uporabiti načine kodiranja, kjer bo šifra čim manj občutljiva na motnje. Eden takih načinov je konvolucijsko kodiranje. Odločili smo se za kodirnik z delovno besedo dolžine treh bitov.



informacijski biti

Informacijske bite vodimo skozi pomikalni register in vsakokrat izračunamo Q_1 in Q_2. Namesto sekvence informacijskih bitov oddajamo sekvenco urejenih parov (Q_1, Q_2). Ta je dvakrat daljša od prvotne in implicitno vsebuje razen osnovne informacije še odnose med zaporednimi tremi biti. Zaradi tega je prenos zanesljivejši.

Ker ne vemo, koliko se je informacija pri prenosu pokvarila, jo lahko dekodiramo le z verjetnostnim računom. Seveda ni mogoče zajeti vseh motenj, ki se lahko pojavijo, saj bi potrebovali ogromen spomin. Zato zajamemo le najverjetnejše, to je minimalne motnje. Rezultat teh motenj je zaporedje, ki se kar najmanj razlikuje od oddane sekvence.

Na tem principu sloni Viterbijev algoritem.
Izmed informacijskih bitov, ki so trenutno v postopku kodiranja, bosta 2. in 3. bit skupaj z naslednjim informacijskim bitom I določala vrednost Q_1, Q_2 v naslednjem koraku. Označimo urejeni par 2. in 3. bita z

x_i ; x_i a = 00, b = 01, c = 10, d = 11

S sprejemom novega inf. bita dobimo par x_{i+1}. Pri tem so možni prehodi, kot jih kaže tabela 1.

$x_i \quad {}^{I}{}_{i+1} \quad x_{i+1}$

Urejena trojica (x_i, I_{i+1}) določa par $(Q_1, Q_2)_{i+1, x_i}$. Ta se pri prenosu moti in dejansko sprejmemo par $(Q_1, Q_2)_{i+1}$, ki se od para $(Q_1, Q_2)_{i+1}, x_j$ razlikuje na m_{i+1, x_i} mestih. $m_{i+1, x_i} \quad 0,1,2$

Definirajmo funkcijo poti $P_{i+1, x_{i+1}}$.

$$P_{i+1, x_{i+1}} = 0 == x_i \quad a,b$$
$$1 == x_i \quad c,d$$

					$(Q_1,Q_2)_{i+1}$			
					00	01	10	11
x_{i+1}	${}^{I}{}_{x_{i+1}}$	x_i	$(Q_1,Q_2)_{i+1,x_i}$	$P_{i+1,x_{i+1}}$	m_{i+1,x_i}			
a	0	a	00	0	0	1	1	2
		c	11	1	2	1	1	0
b	1	a	11	0	2	1	1	0
		c	00	1	0	1	1	2
c	0	b	10	0	1	2	0	1
		d	01	1	1	0	2	1
d	1	b	01	0	1	0	2	1
		d	10	1	1	2	0	1

Tabela 1

Sprejeli smo sekvenco Q_i parov $(Q_1, Q_2)_j$; j = 1,...,i. Iz nje lahko sklepamo, kakšna je bila sekvenca inf. bitov S_i. Sklepamo ločeno za primere, ko se S_i končuje z a, b, c oziroma d. Tako dobimo štiri sekvence S_{i,x_i}; $x_i \quad a,b,c,d$. Vsaka sekvenca S_{i,x_i} nam da sekvenco Q_{i,x_i}, ki se od dejansko sprejete sekvence Q_i razlikuje na M_{i,x_i} mestih.

$$S_{i,x_i} == Q_{i,x_i}$$

$$Q_i - Q_{i,x_i} = M_{i,x_i}$$

V naslednjem koraku sprejmemo par $(Q_1, Q_2)_{i+1}$ in za vsako končnico x_{i+1} spet poiščemo najverjetnejšo vhodno sekvenco $S_{i+1,x_{i+1}}$. Nove sekvence so seveda nastale z dodajanjem naslednjega informacijskega bita starim vhodnim sekvencam.

$$S_{i+1,x_{i+1}} = S_{i,x_i} \; , \; I_{x_{i+1}}$$

Odnos med x_{i+1}, x_i, ${}^{I}{}_{x_{i+1}}$ že poznamo iz tabele 1.

$$S_{i+1,x_{i+1}} == Q_{i+1,x_{i+1}} \; ; \quad Q_{i+1,x_{i+1}} = Q_{i,x_i},$$

$$(Q_1,Q_2)_{i+1,x_i}$$

$$Q_{i+1}-Q_{i+1,x_{i+1}} = Q_i-Q_{i,x_i} + (Q_1,Q_2)_{i+1}-(Q_1,Q_2)_{i+1,x_i} =$$

$$= M_{i,x_i} + m_{i+1,x_i} = M_{i+1,x_{i+1}}$$

Ker sta za vsak x_{i+1} možna dva x_i, izberemo verjetnejšega, to je tistega, kjer je $M_{i+1,x_{i+1}}$ manjša.

$$x_i: \; M_{i+1,x_{i+1}}(x_i) = \min.$$

Izbrani x_i določa funkcijo poti:

$$P_{i+1,x_{i+1}} = P_{i+1,x_{i+1}}(x_i)$$

Izmed štirih sekvenc $S_{i+1,x_{i+1}}$ vzamemo kot pravilno tisto, kjer je

$$x_{i+1}: \; M_{i+1,x_{i+1}}(x_{i+1}) = \min.$$

Ker vse sekvence S_{i,x_i} težijo k isti najverjetnejši sekvenci S, se začenjajo deloma ujemati v bitih, ki so nastali prej in so šli že večkrat skozi postopek preverjanja verjetnosti. S poskusi so ugotovili, da je to dovolj pogosto že na mestu (i - D.4), strožje na mesti (i - 5.D), kjer je dolžina delovne besede kodirnika, v našem primeru je D = 3. Delov sekvenc S_{i,x_i}, ki so skupni, ni več mogoče izboljšati, zato predstavljajo rezultat dekodiranja. Torej lahko predstavimo sekvence v registrih z dolžino 4.D, bit, ki vsakokrat izpade iz registra, pa je rezultat. Program smo realizirali na MC 6800, ki ima 8-bitne celice, zato smo uporabili 16-bitne registre.

Za S_{0,x_0} lahko izberemo poljubno zaporedje, prav tako za M_{0,x_0} poljubne vrednosti, saj bodo te zanemarljive napram vsoti m_{i,x_i} višjih redov. Mi smo izbrali $S_{0,x_0}=0^{16}$, $M_{0,x_0} = 0$.

Za naslednji korak so pomembne le relativne razlike med M_{i,x_i}, zato vsakokrat odštejemo minimalno.

$$M_{i,x_i} \quad M_{i,x_i} - M_{i,x_i}(x_i) \min.$$

Pri tem se pokaže, da so vse možne kombinacije naslednje:

M_a	0	1	0	0	0	1	0	0	1	1	1	0	0	1	2	0	2	1	1	0	1	2	2
M_b	0	0	1	0	0	1	0	1	0	1	1	0	0	2	1	2	0	1	1	1	0	2	2
M_c	0	0	0	1	0	0	1	1	1	0	1	1	2	0	0	1	1	0	2	2	2	0	1
M_d	0	0	0	0	1	0	1	1	1	1	0	2	1	0	0	1	1	2	0	2	2	1	0
M	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Tabela 2

M_a	0	2	2	2	0	2	3	3
M_b	2	0	2	2	2	0	3	3
M_c	2	2	0	2	3	3	0	2
M_d	2	2	2	0	3	3	2	0
M	23	24	25	26	27	28	29	39

Vsak niz M_a, M_b, M_c, M_d nadomestimo z njegovo zaporedno številko M; M = 0, 1, ..., 30

Zahteve pri programiranju:
1. Čim več rezultatov izračunamo vnaprej in jih po potrebi le pokličemo iz spomina.
2. Uporabimo čim več spomina tipa EPROM in čim manj RAM-a.

Za vsak M izračunamo tabelo prehodov.

M_i	Q_1	Q_2	M_{i+1}	M	P_{i+1}	J_{i+1}	S	S
a	0	0	a	d	a	d		
	0	1				J	J_x	
	1	0						
d	1	1						

Primer za M = 19.

19

0	0	0	0222	23	0x00	1	1	0	0
1	0	1	0010	3	0010	0	0	1	0
2	1	0	0001	4	0001	0	1	1	0

2

Pri starem stanju M_i sprejmemo bita Q_1, Q_2. Dobimo minimalne razdalje - nabor M_{i+1} z imenom M, s tem da smo uporabili optimalne poti iz nabora P_{i+1}.

P_{i+1} = 0, če smo šli po zgornji poti, n.pr. a v a .

= 1, če smo šli po spodnji poti, n.pr. c v a .

= X, če sta poti enakovredni.

Mestom M_{i+1,x_i} priredimo naslednje dvojiške vrednosti:

$M_{i+1,a}$: 11 ; $M_{i+1,b}$: 10 ; $M_{i+1,c}$: 01 ; $M_{i+1,d}$:00.

J_{i+1} je oznaka minimalne $M_{i+1,x_{i+1}}$, to je registra, iz katerega bomo prebrali rezultat.

1. Če je M_{i+1} min. ena sama, pišemo njeno pozicijo v stolpec $J.J_x$= 00.
2. Če sta M_{i+1} min. dve, se poziciji vedno razlikujeta le po enem bitu. V stolpec J_x vpišemo skupni bit z njegovo pravo vrednostjo, bit se razlikuje, pa označimo z X.J = 00.
3. Če so M_{i+1} min. tri, dve sosednji vpišemo kot pod točko 2, tretjo pa kot pod točko 1.
4. Če so M_{i+1} min. štiri, je J_x = XX in J = 00.
5. S = 1 natanko v primeru 3, ko se moramo odločati med J in J_x z verjetnostjo p(j) : $p(J_x)$ = 1 : 2.
6. S = 0 pomeni, da smo se odločili za J. To je v primeru 1, po naši prosti izbiri pa tudi v primeru 3, ko se S posebej izračuna v RAM-u.
7. S = 1, če se odločimo za J_x, to je v primerih 2 in 4.

Vsaki kombinaciji M, Q_1, Q_2 priredimo tri spominske celice v EPROM-u. Naj bo R najnižji uporabljeni naslov.

Potem najdemo na naslovu

$R + 4.M_i+(Q_1Q_2)_2$ celico T,

$R + 4.M_i+(Q_1Q_2)_2+4.31$ celico U,

$R + 4.M_i+(Q_1Q_2)_2+8.31$ celico V.

0	M_{i+1}	0	0	T
J_{i+1}	J_x	P_{i+1}		U
S	S	$X(J_x)$	$X(P_{i+1})$	V

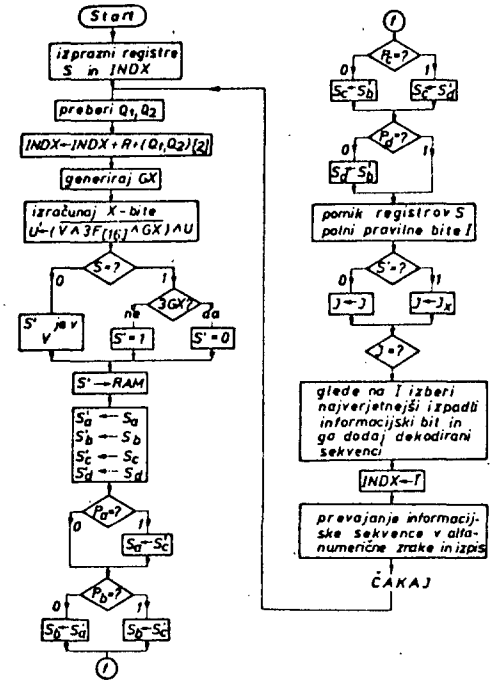
Biti v $X(J_x)$ in $X(P_{i+1})$ imajo vrednost 1 na mestih, kjer v J_x in P_{i+1} nastopa X, sicer so 0.

Prva celica je del naslova za naslednji cikel. R še ni upoštevan, zato je možno vse tri bloke spomina prestavljati na poljubne naslove brez sprememb. V začetku vsakega cikla se izračuna naslov celice T in se shrani v INDEX registru. Celice U in V so dosegljive z indeksiranim naslavljanjem.

Za polnjenje X bitov potrebujemo psevdogenerator naključnih števil GX. Glede na GX (mod 3) tudi določamo S , kadar je S = 1.

Potrebne celice RAM-a:

- GX,
- štirje dvocelični pomikalni registri za S_a, S_b, S_c, S_d,
- štirje dvocelični spomini za spravljanje registrov S, ko jih je treba premešati,
- celica za informacijske bite, ki izpadejo iz registrov,
- celica za S ,
- celica za U, ko se izračunajo X-biti,
- celica za dekodirano sekvenco informacijskih bitov.



3. Sklep

Opisani algoritem smo preizkusili na simuliranem prenosnem sistemu. Prenos sporočil je zanesljiv do določene kvalitete kanala. Pri izbiri daljših registrov, bi po pričakovanju prenosni sistem iskazal še boljšo zanesljivost. Koristili smo majhen pomnilniški prostor in razmeroma malo procesorskega časa. V vsakem primeru bomo ob uporabi integriranih vezij s tem dekodirnim algoritmom dvignili zanesljivost tudi nad mediji, kjer je zanesljivost zelo slaba in hkrati povečali prepustnost komunikacijskega kanala ter bo odpadla potreba po povratni liniji.

LITERATURA

1) A.J. Viterbi: Error Bounds for Convolutional Codes and a Asymptotically Optimum Decoding Algorithm; IEEE Trans. on Information Theory Nov.1970 Vol. IIT-16 No 6.

2) J. Justesen: Convolutional Code Construction; IEEE Trans. on Information Theory Vol. I.T - 1973.

3) B.Horvat in sodelavci: Prenos gospodarnih in vernih informacij II. Naloga SBK 785/762502, januar 1978.

4) M 6800 Microprocessor Programming Manual Motorola Inc 1975.

KOMUNIKACIJSKI PROTOKOLI

M. KAPUS
A.P. ŽELEZNIKAR

UDK: 681.324

VISOKA TEHNIŠKA ŠOLA, MARIBOR
IJS, LJUBLJANA

Članek podaja pregled najbolj razširjenih formatov za prenos informacij. Podrobno obravnava HDLC sinhrone protokole in komuniciranje z informacijskimi paketi.

COMMUNICATION PROTOCOLS
In the paper the most commonly used formats for the information transfer are described. Details are given about the HDLC and the packet level communications.

1. UVOD

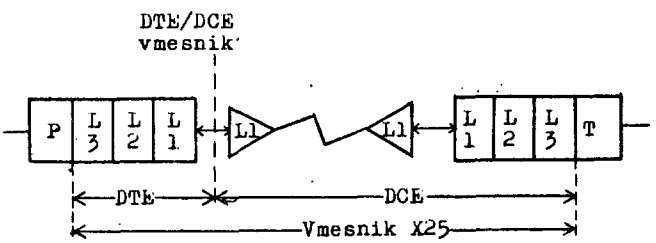
Leta 1976 je CCITT izdal nekaj novih smernic za prenos informacij v priporočilu X25, ki podaja način komuniciranja med terminali oz. DTE (Data Terminal Equipment) in DCE (Data Circuit-terminating Equipment)z informacijskimi paketi.

Komuniciranje lahko obravnavamo na treh nivojih. Prvi nivo podaja električne karakteristike linij in fizikalne procedure za njihovo vzpostavljanje.

Drugi nivo je kontrola linije z digitalnimi sekvencami, ki omogoča zanesljiv prenos informacij v realnih tokokrogih.

Tretji nivo obravnava manipulacijo z informacijskimi paketi v stalnih ali začasnih logičnih kanalih.

X25 podrobno definira tretji logični nivo. Za ilustracijo drugega nivoja podaja uporabo HDLC protokola z asinhronim načinom odgovora v dupleksni izvedbi. Prvi nivo je opisan v priporočilu X21. Članek je pregled protokolov za drugi in tretji nivo.



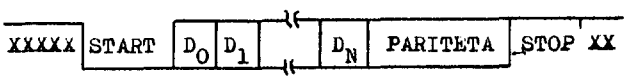
L1 - fizikalni vmesnik
L2 - logični vmesnik na nivoju nabora
L3 - logični vmesnik na nivoju paketa
P - uporabnikovi procesi
T - transportna funkcija mreže
▷ X21 kontrolor linije ali
V24 modem

2. SEKVENČNI NIVO

2.1. Uvod

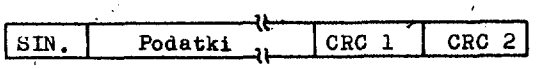
Za prenos informacij se uporablja več formatov.

Asinhroni format



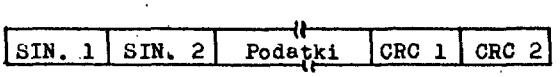
X - sekvenca prostega teka
Pariteta - liha, soda ali ni prisotna
N - 5,6,7 ali 8
Ustavitveni (stop) znak - 1, 1.5 ali 2 bita

Sinhroni format z enim zlogom za sinhronizacijo



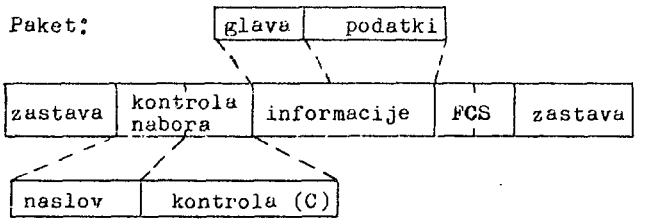
SIN. - sinhronizacijski zlog
CRC - kontrola pravilnosti prenosa z metodo krožečega ostanka

Sinhroni format z dvema zlogoma za sinhronizacijo



Format pri zunanji sinhronizaciji

Za asinhrone formate je značilna mala dolžina informacijskega polja po pravilu en znak - en nabor. Sinhroni formati uporabljajo nabore različne dolžine, ki vključujejo več zlogov (byte). S tem lahko prenašamo kontrolne ukaze v samem naboru, manj je redundančnih startnih in ustavitvenih bitov, če pošiljamo dolge sekvence. Zato so sinhroni formati idealna oblika prenosa večje količine informacij. Najbolj dovršen format uporabljajo HDLC protokoli.



2.2. HDLC - High Level Data Link Control

je skupno ime več standardiziranih protokolov za sinhroni prenos podatkov. Ti protokoli vzpostavljajo in prekinjajo zvezo, kontrolirajo pravilnost prenosa, sami popravljajo nekatere napake, o drugih pa obveščajo višje nadzorne strukture. Razlikujejo se po namenu za različne konfiguracije sistema.

Pri vseh je osnovna prenašana enota nabor, sestavljen iz začetne in končne zastave, naslovnega zloga, kontrolnega zloga, informacijskega polja spremenljive dolžine in dveh zlogov za FCS (Frame Checking Sequence). Eden izmed bitov kontrolnega zloga je P/F bit, ki ureja odnos hierarhije med partnerjema in s tem neposredno določa vrsto protokola.

2.2.1. P/F cikel

Na vsaki liniji ima eden od partnerjev primarno, drugi pa sekundarno vlogo. Primar oddaja ukaze, sekundar odgovarja. V ožjem pomenu le primar oddaja informacije, vendar to ni nujno. Če je terminal priključen na več linij, je lahko za nekatere primar, za druge sekundar. P/F bit imenujemo v naborih, ki jih oddaja primar, P (Poll) bit, v naborih, ki jih oddaja sekundar, pa F (Final) bit.

Primar, ki zahteva od sekundarja odgovor, to označi s P = 1. Sekundar konča odgovor z naborom, ki ima F = 1. Primar lahko oddaja nabore s P´, toda šele P - nabor aktivira sekundar. Ko sekundar odda F - nabor, ne sme oddati nobenega nabora več, dokler ne sprejme P - nabora. Sekundar lahko odda F - nabor takoj ali pa za nekaj F´ - nabori. Prvi način imenujemo asinhroni način odgovora (ARM - Asynchronous Response Mode), drugi pa normalni ali sinhroni način odgovora (NMR - Normal Response Mode ali SDLC - Synchronous Data Link Control).

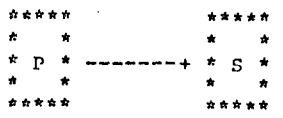
ASINHRONI NAČIN		SINHRONI NAČIN	
Primar	Sekundar	Primar	Sekundar
P´--+		P´--+	
P --+		P --+	
	+--F		+--F´
P --+		P´--+	+--F´
	+--F	P´--+	+--F
P´--+		P´--+	

Vidimo, da je pri NRM možen enakovreden pretok informacij v obeh smereh, pri ARM pa sekundar v glavnem le potrjuje pravilen sprejem.

2.2.2. Uporaba asinhronih protokolov

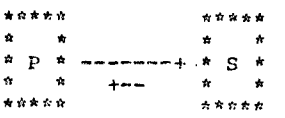
Z ARM lahko izvedemo simpleksni prenos (simpleks), neuravnoteženi dupleksni in uravnoteženi dupleksni prenos (dupleks).

Za simpleksni prenos imamo:



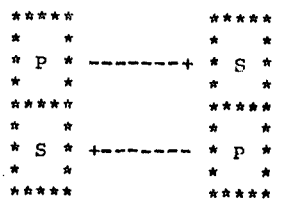
Primar oddaja informacije, sekundar jih potrjuje (ali zavrača) z nabori brez informacijskega polja. Pretok informacij je le v eno smer.

Za neuravnoteženi dupleks (Unbalanced Duplex) je:



Primar oddaja informacije, sekundar odgovarja z F-nabori, ki vsebujejo informacijsko polje. Ker je število F-naborov omejeno s številom P-naborov, število P´-naborov pa ne, in ker je dolžina informacijskega polja navzgor omejena, je pretok informacij v smeri od primarja k sekundarju močnejši.

Pri uravnoteženem dupleksu (Balanced Duplex) imamo:



Vsaka postaja na liniji ima vlogo primarja in sekundarja. Tako imamo simpleks v dveh smereh.

2.2.3. Uporaba SDLC

Pri SDLC je pretok informacij zaradi narave P/F cikla lahko vselej dupleksni. Zato je SDLC uporaben tudi za bolj kompleksne konfiguracije sistema, npr. multipoint in loop.

Javlja se vprašanje izbire polovičnega in polnega dupleksa. (Polovični - half - dupleks poteka po eni sami žici izmenično v obeh smereh. Polni - full - dupleks ima za vsako smer poseben vod.) Uporaba polnega dupleksa je smiselna pri sinhronih protokolih, ko sta na liniji lahko hkrati P´ in F´ nabor, pri ARM pa je polni dupleks uporaben le kot uravnoteženi dupleks.

2.3. Podroben opis nabora pri HDLC

2.3.1. Zastava (Flag)

Zastava označuje začetek in konec nabora in rabi kot znak za sinhronizacijo. Njena oblika je 01111110. Da ne bi sprejemnik enakega vzorca znotraj nabora prebral kot zastavo, se avtomatično vstavi 0 za vsakimi zaporednimi petimi 1. Pri sprejemu se 0 odstrani.

2.3.2. Naslovni zlog

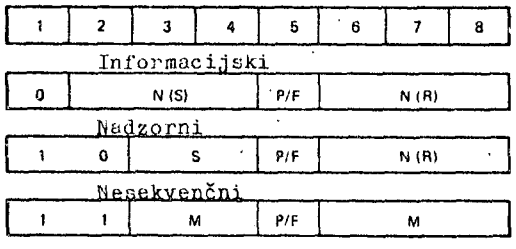
Ne glede na smer pretoka informacij je v naslovnem zlogu vedno naslov sekundarja. Kadar vsi partnerji vsebujejo primarno in sekundarno funkcijo, je dogovorjeno, da sta v naslovih sekundarja na DTE najniže utežena bita b 0 = 1, b 1 = 1, na DCE pa b 0 = 1, b 1 = 0. Tu je skrito nekaj redundance.

2.3.3. FCS zloga

rabita za kontrolo pravilnosti prenosa polja med začetno zastavo in zlogoma FCS. Pri tem se uporablja metoda CRC (Cycling Redundancy Checking) v različnih oblikah. Več o tem je napisanega v (5).

2.3.4. Formati kontrolnega zloga (C)

C zlog ima tri formate: I (information), S (supervisory) in NS (nonsequenced).



2.3.4.1. Vlogo P/F bita poznamo. V I in S formatu pa se pojavljata N(R) in N(S). N(R) je sprejemno sekvenčno število (Recieve Sequence Number), N(S) je oddajno sekvenčno število (Send Sequnce Number). Informacijski nabori (I) se štejejo po modulu 8 ločeno za obe smeri. Kadar postaja A odda postaji B informacijski nabor, je v njem N(S) zaporedno število nabora v smeri A --+ B, N(R) pa število naslednjega I-nabora, ki bo prišel v smeri B --+ A.

Vsak partner na liniji ima svojo V(R), spremenljivko sprejemnega stanja (Recieve State Variable),in V(S), spremenljivko oddajnega stanja (Send State Variable). V(R) je število naslednjega I-nabora, ki ga bo postaja sprejela, V(S) pa zaporedno število naslednjega oddanega nabora. Ko postaja odda I-nabor, vpiše V(S) v N(S) in V(R) v N(R). Potem se V(S) poveča za 1.

V(S) +-- (V(S) + 1)(mod 8)

Ko se sprejme I-nabor, se mora V(S) ujemati z N(R) in V(R) z N(S), saj je V(R) postaje A enak V(S) postaje B in obratno. Če je I-nabor pravilno sprejet, se V(R) poveča za 1.

V(R) +-- (V(R) + 1)(mod 8)

Nabori NS in S formata se ne štejejo v sekvenco. N(R) v S-naboru je le kontrolna informacija.

2.3.4.2. Nadzorni format S (Supervisory)

ima pri ARM vlogo odgovora, pri NRM pa vlogo ukaza in odgovora. Ti so:

RR (Recieve Ready). Potrjuje pravilen sprejem naborov do vključno N(R) - 1 in sporoča, da je postaja pripravljena na sprejem.

RNR (Recieve Not Ready). Opozarja, da je postaja preobremenjena in ne more sprejeti nobenega nabora, ki zahteva prostor v vmesniku. Potrjuje sprejem naborov do N(R) - 1.

REJ (Reject, zavrnitev). Zahteva oddajo ali ponovno oddajo nabora s sekvenčnim številom N(R) in potrjuje sprejem naborov do N(R) - 1. Zahteva REJ se briše, ko postaja pravilno sprejme nabor N(R).

Format	prvi bit	Binarna konfiguracija	zadnji bit	Vsebina	Ukaz	Odgovor	Informacijsko polje prepovedano	Resetira N(R) in N(S)	Potrjuje nabore do N(R) - 1
NS	000	P/F	0011	NSI	—	—			
	000	F	0111	RQI		—	—		
	000	P	0111	SIM	—		—	—	
	100	P	0011	SNRM	—		—	—	
	000	F	1111	ROL		—	—		
	010	P	0011	DISC	—		—		
	011	F	0011	NSA		—	—		
	100	F	0111	CMDR		—			
	001	1	0011	ORP	—		—		
S	N(R)	P/F	0001	RR	—	—	—		—
	N(R)	P/F	0101	RNR	—	—	—		—
	N(R)	P/F	1001	REJ	—	—	—		—
I	N(R)	P/F	N(S)0	I	—	—			—

2.3.4.3. NS (Nonsequenced) format

imenovan tudi U (Unnumbered), pozna različne ukaze pri ARM in NRM. Pri NRM so to:

NSI (Nonseqenced Information). Služi za prenos informacij v naboru zunaj redne sekvence informacijskih naborov. Sprejema ni treba potrjevati.

SNRM (Set Normal Response Mode). Podredi sekundar primarju. Sekundar lahko odgovarja le na zahtevo primarja.V(R) in V(S) se resetirata (na 0). Pričakovani odgovor je NSA. Sekundar ostane v NRM do sprejema DISC ali SIM ukaza.

DISC (Disconect). Postavi sekundar v stanje off-line. Sekundar ne more več oddajati I-naborov, dokler ne sprejme SNRM ali SIM ukaza. Pričakovani odgovor je NSA.

NSA (Nonsequenced Acknowledgement) je pritrdilni odgovor na SNRM, DISC ali SIM ukaz. Nadaljni prenos naborov sproži primar.

RQI (Request for Initialization). S tem sporoči sekundar primarju, da želi SIM ukaz. Vsak ukaz razen SIM sproži ponovitev RQI.

SIM (Set Initialization Mode). Sproži proceduro za vzpostavitev normalnih funkcij na nivoju linije. V(S) in V(R) se resetirata.

ROL (Request On-Line). S tem sporoča sekundar, da je v stanju off-line.

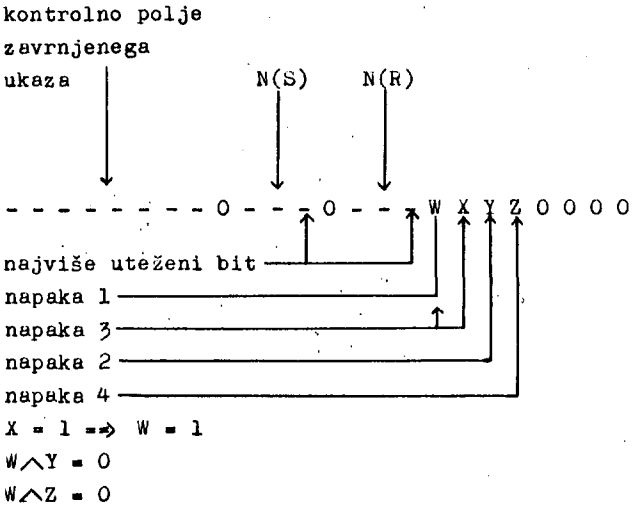
CMDR (Command Reject). Ta odgovor odda sekundar v stanju NRM, če sprejme napačen ukaz. To je v primerih:

1. Kontrolno polje ni definirano.

2. Informacijsko polje je predolgo za vhodne vmesnike.

3. Nabor vsebuje informacijsko polje, ki je pri danem formatu kontrolnega zloga prepovedano.

4. N(R) se ne ujema z V(S) sekundarja.



Vrsta napake je označena v informacijskem polju, kot kaže slika. Odgovor CMDR se ponavlja do sprejema SNRM, DISC ali SIM.

Protokoli za ARM poznajo ukaze:

SARM (Set Asynchronous Response Mode), kot SNRM pri NRM protokolih, oblika 000 P 1111; UA (Unnumbered Acknowledgement) = NSA; CMDR in DISC.

2.4. Abortiranje (razveljavljanje)

Če želi oddajnik napol oddani nabor razveljaviti, odda osem zaporednih enic. Če jih doda še sedem, linija preide v prosti tek in jo je treba ponovno inicializirati. Lahko pa osmim enicam doda zastavo in s tem ostane na liniji. Če abortira sekundar, ne sme več oddajati, dokler ga primar ne pokliče.

2.5. Kontrola zanke (loop) pri SDLC

Normalno sekundarji ponavljajo ukaz primarja po zanki. Primar dobi status sekundarjev z ukazom ORP (Optional Response Poll), oddanim na skupni, sistemsko določeni naslov. Z ukazom GA (go ahead) da pravico sekundarju z najvišjo prioriteto, da začne oddajati lastne informacije. Ostali ukazi so vsi prej našteti.

2.6. Sistemski parametri

2.6.1. Vsakokrat, ko primar aktivira sekundar s P-bitom, vključi časovnik T1. Primar ukaz ponovi, če časovnik izteče pred odgovorom sekundarja. To lahko naredi N2-krat. Kasneje je potrebna akcija na višjem logičnem nivoju.

Pri tem primar ne ve, koliko njegovih informacijskih naborov je bilo potrjenih. Zato shrani svojo V(S) na notranjo spremenljivko x, V(S) pa dobi vrednost zadnje N(R), ki jo je sekundar potrdil. V primeru, da kasneje pride odgovor z N(R), ki ni v območju med vključno novo V(S) in x, to pomeni, da se je sekundar zmedel v štetju naborov in primar odda REJ.

2.6.2. Maksimalno število zlogov v naboru (N1) je določeno z maksimalno dolžino informacijskega polja.

2.6.3. Maksimalno število informacijskih naborov (k), ki jih postaja lahko odda, preden jih sprejemnik potrdi, je manjše ali enako sedem.

3. PAKETNI NIVO

3.1. Uvod

Paket je zaključena logična enota, ki se prenaša v informacijskem polju naborov po logičnih kanalih. Logični kanal je kanal brez motenj, ki ga dobimo iz realnega kanala tako, da uvedemo kontrolo na drugem nivoju (na nivoju binarne sekvence).

Paketi so sestavljeni iz kontrolne glave (header) in podatkov v ožjem pomenu besede. Glede na glavo jih delimo v 14 tipov.

VRSTA PAKETA	
Od DCE k DTE	Od DTE k DCE
Vzpostavljanje zveze in čiščenje	
INCOMING CALL	CALL REQUEST
CALL CONNECTED	CALL ACCEPTED
CLEAR INDICATION	CLEAR REQUEST
DCE CLEAR CONFIRMATION	DTE CLEAR CONFIRMATION
Prenos podatkov in prekinitev	
DCE DATA	DTE DATA
DCE INTERRUPT	DTE INTERRUPT
DCE INTERRUPT CONFIRMATION	DTE INTERRUPT CONFIRMATION
Kontrola pretoka in reset	
DCE RR	DTE RR
DCE RNR	DTE RNR
	DTE REJ
RESET INDICATION	RESET REQUEST
DCE RESET CONFIRMATION	DTE RESET CONFIRMATION
Restart	
RESTART INDICATION	RESTART REQUEST
DCE RESTART CONFIRMATION	DTE RESTART CONFIRMATION

3.2. Procedure pri prenosu

3.2.1. Prenos informacij med dvema DTE se odvija preko DCE:

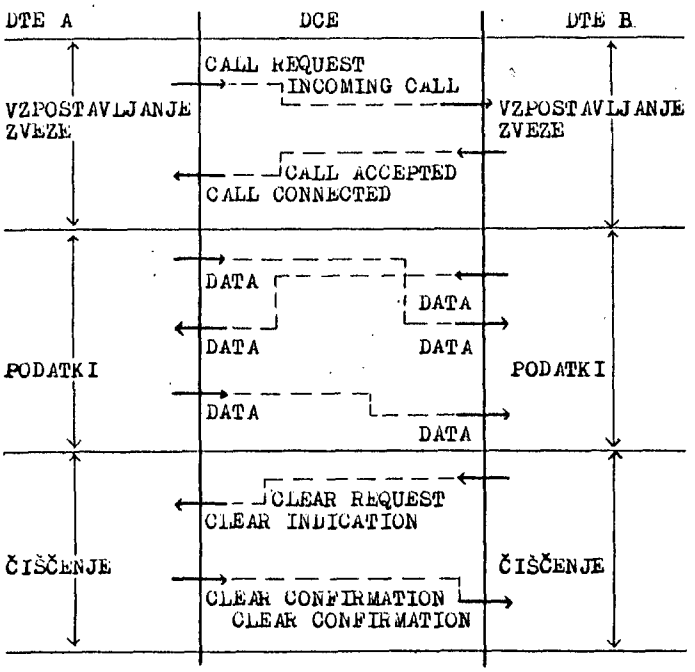
DTE A ---+ DCE ---+ DTE B

Pri tem se uporabi poljuben prost (READY) logični kanal, navadno najniže oštevilčeni.

Logični kanali so razporejeni v največ 15 skupin. V vsaki je največ 255 kanalov. Kanal je enolično določen s številko skupine in številko kanala znotraj skupine.

3.2.2. Vzpostavljanje in prekinjanje zveze

DTE A zahteva zvezo z DTE B s paketom CALL REQUEST. DCE o tem obvesti DTE B z INCOMING CALL. DTE B potrdi vezo s CALL ACCEPTED. DCE obvesti DTE A s CALL CONNECTED. Če je veza že vzpostavljena ali če takrat, ko DCE oddaja INCOMING CALL, DTE B hkrati zahteva isto zvezo na istem logičnem kanalu, DCE takoj odda CALL CONNECTED.



Če DTE B noče sprejeti zveze, odda CLEAR REQUEST. DCE obvesti DTE A s CLEAR INDICATION z navedbo, da je DTE B zaposlen. DTE A potrdi z DTE CLEAR CONFIRMATION. DCE obvesti DTE B z DCE CLEAR CONFIRMATION. Če DCE ne more vzpostaviti zveze zaradi drugih vzrokov, to označi v paketu CLEAR INDICATION. Prekinitev zveze lahko zahteva kadarkoli.

3.2.3. Prenos podatkov

Podatki se prenašajo v DTE in DCE DATA paketih. Če se nadaljujejo v naslednjem paketu, to označimo z bitom M (more data). Glede na to delimo DATA pakete v tri skupine:

a) podatkovno polje krajše od maksimalnega dovoljenega, brez M,

b) podatkovno polje maksimalno, brez M,

c) podatkovno polje maksimalno, z M.

Dolžina podatkovnega polja je omejena na 16, 32, 64, 128 (najpogosteje), 256 (255), 512 ali 1024 oktetov (8 bitov).

DATA paketi se štejejo po modulu 8 ali 128 za vsako smer vsake linije. V ta namen uvedemo P(R), sprejemno sekvenčno število paketa, in P(S), oddajno sekvenčno število paketa, ki služita za kontrolo okna. Širina okna W je največje število DATA paketov, ki se lahko hkrati prenesejo brez vmesne kontrole, in ni večje od 7 oziroma 127. Določa se za vsak DTE in vsako smer posebej glede na dejansko potrebo po prenosu informacij. Prvi prenašani paket ima P(S) vedno enak P(R) sprejemne postaje.

V enem oknu so paketi lahko vidno ločeni v dve skupini z bitom Q (data qualifier), vendar ne smejo biti pomešani.

Prenos podatkov poteka po istih zakonitostih kot na nivoju binarnih naborov in lahko ugotovimo naslednjo ekvivalenco pojmov:

DCE DATA paket	-	I-nabor
DTE DATA paket	-	I-nabor
DCE RR paket	-	RR-nabor
DTE RR paket	-	RR-nabor
DCE RNR paket	-	RNR-nabor
DTE RNR paket	-	RNR-nabor
DTE REJ paket	-	REJ-nabor
P(S)	-	N(S) oz. V(R) ali V(S)
P(R)	-	N(R) oz. V(S) ali V(R)
W	-	parameter k

3.2.4. Reset in restart

Z RESET in RESTART paketi odstranimo z izbranega kanala vse DATA in INTERRUPT pakete. Z RESTART paketi resetiramo vse kanale, ki so vezani na pobudnika restarta. Reset (restart) lahko sproži DCE in vsak DTE s paketom RESET (RESTART) INDICATION oziroma RESET (RESTART) REQUEST. Potrdilna paketa sta DTE in DCE RESET (RESTART) CONFIRMATION. Procedura je enaka kot pri CLEAR paketih.

3.2.5. Interrupt (prekinitev)

INTERRUPT paketi omogočajo prenos podatkov izven redne sekvence DATA paketov, po funkciji torej ustrezajo NSI-naborom na nivoju dva. Lahko jih uporabljamo, kadar v kanalu ni RESET in RESTART paketov, zveza pa je vzpostavljena. DTE pošlje podatke DCE v DTE INTERRUPT paketu. DCE potrdi z DCE INTERRUPT CONFIRMATION. DCE pošlje podatke v DCE INTERRUPT paketu. DTE potrdi z DTE INTERRUPT CONFIRMATION.

3.2.6. Posebni dogovori o načinu komuniciranja

Vsakokrat pri vzpostavljanju zveze lahko DTE v CALL paketu zahteva določen način komuniciranja. Specificira npr. širino okna W ali maksimalno dolžino podatkovnega polja. Če teh specifikacij ni, dobita parametra maksimalno možno vrednost v mreži. DTE lahko zahteva Reverse Charging (povratno odgovornost). Ta dogovor mu omogoča, da lahko zahteva oddajo informacij z drugega terminala.

Druga skupina dogovorov velja za določeno zvezo DTE/DCE trajno za dogovorjeni čas. Npr.:

Omogočitev povratne odgovornosti (Reverse Charging Acceptance). DCE se obveže, da bo sprejela Reverse Charging Request, če se pojavi na tej liniji.

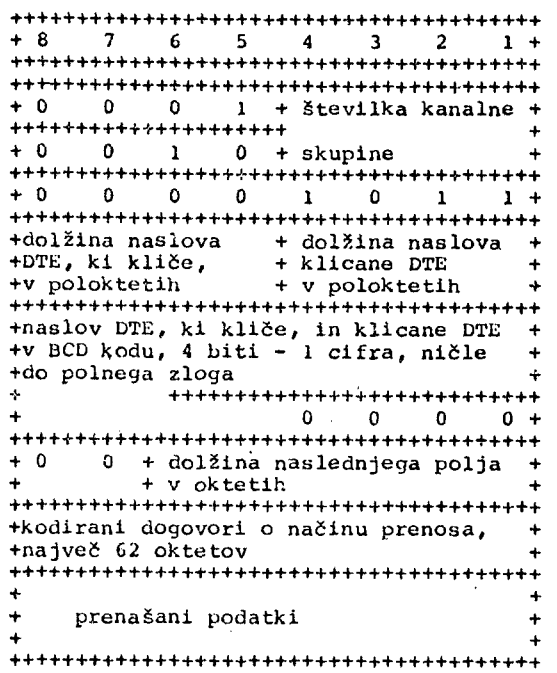
Enosmerni logični kanal (One-way Logical Channel). Kanalu se pripiše ena od smeri DTE/DCE ali DCE/DTE.

Ponovna oddaja paketa (Packet Retransmission). Terminal dobi pravico zahtevati od DCE ponovno oddajo paketa s paketom DTE REJ.

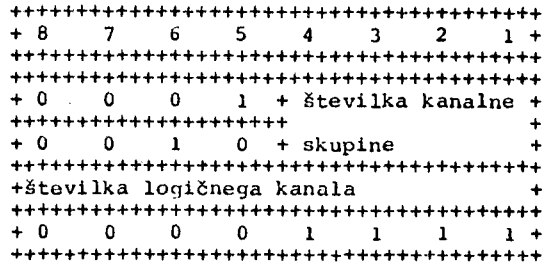
DCE označi te dogovore v CALL INDICATION paketu. Seveda je možnih dogovorov še veliko. Za vse velja, da se kodirajo v paru oktetov, v prvem oktetu vrsta dogovora, v drugem pa pripadajoči parametri.

3.3 Formati paketov

CALL REQUEST in INCOMING CALL
paketni format



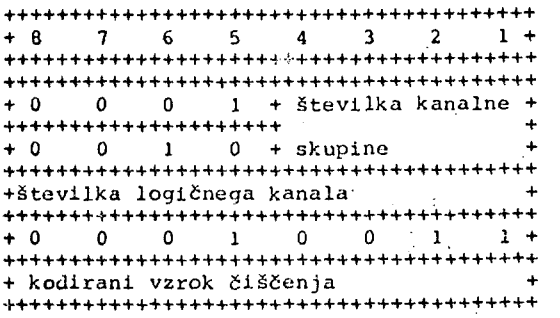
CALL ACCEPTED in CALL CONNECTED
paketni format



Dodatni komentarji

Vse vrednosti, pri katerih ni posebej označeno, so podane binarno, tako da je najniže oštevilčeni bit najniže utežen. Isto pravilo velja pri BCD vrednostih tudi za razpored decimalnih cifer. Biti 8-5 prvega zloga zavzemajo zgornjo vrednost, če se DATA paketi štejejo po modulu 8, in spodnjo, če se štejejo po modulu 128.

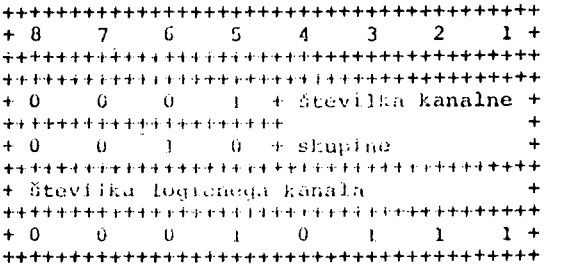
CLEAR REQUEST in CLEAR INDICATION
paketni format



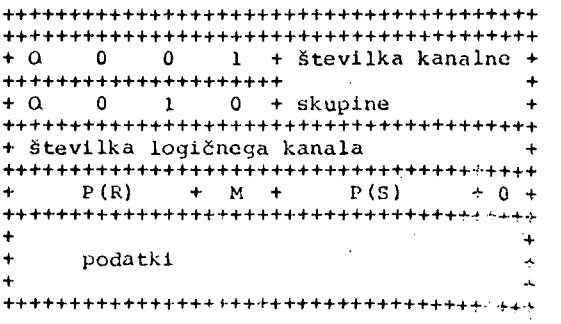
Kodiranje vzroka čiščenja v
CLEAR INDICATION paketu

	87654321
DTE čiščenje	00000000
postaja je zasedena	00000001
postaja ne deluje	00001001
procedurna napaka na drugi strani zveze	00010001
postaja noče sprejeti povratne odgovornosti	00011001
napačen klic	00000011
dostop je blokiran	00001011
lokalna procedurna napaka	00010011
mreža je zasedena	00000101
mreža ne deluje	00001101

DTE in DCE CLEAR CONFITMATION
paketni format



DTE in DCE DATA
paketni format



M = MORE DATA
Q = DATA QUALIFIER

DTE in DCE RR
paketni format

8	7	6	5	4	3	2	1
0	0	0	1	številka kanalne skupine			
0	0	1	0				
številka logičnega kanala							
P(R)			0	0	0	0	1

DTE in DCE RNR
paketni format

8	7	6	5	4	3	2	1
0	0	0	1	številka kanalne skupine			
0	0	1	0				
številka kanala							
P(R)			0	0	1	0	1

DTE REJ paketni format

8	7	6	5	4	3	2	1
0	0	0	1	številka kanalne skupine			
0	0	1	0				
številka logičnega kanala							
P(R)			0	1	0	0	1

DTE in DCE INTERRUPT
paketni format

8	7	6	5	4	3	2	1
0	0	0	1	številka kanalne skupine			
0	0	1	0				
številka logičnega kanala							
0	0	1	0	0	0	1	1
podatki							

DTE in DCE INTERRUPT CONFIRMATION
paketni format

8	7	6	5	4	3	2	1
0	0	0	1	številka kanalne skupine			
0	0	1	0				
številka logičnega kanala							
0	0	1	0	0	1	1	1

RESTART REQUEST in RESTART INDICATION
paketni format

8	7	6	5	4	3	2	1
0	0	0	1				
0	0	0	0				
0	0	1	0				
0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	1
vzrok za restart							

Kodiranje vzroka za restart
v RESTART INDICATION paketih

	87654321
lokalna procedurna napaka	00000001
mreža je zasedena	00000011

DTE in DCE RESTART CONFIRMATION
paketni format

8	7	6	5	4	3	2	1
0	0	0	1				
0	0	0	0				
0	0	1	0				
1	1	1	1	1	1	1	1

RESET REQUEST in RESET INDICATION
paketni format

8	7	6	5	4	3	2	1
0	0	0	1	številka kanalne skupine			
0	0	1	0				
številka logičnega kanala							
0	0	0	1	1	0	1	1
vzrok za reset							
diagnostična koda							

Kodiranje vzroka za reset v
RESET INDICATION paketih

	87654321
DTE reset	00000000
okvara	00000001
procedurna napaka na drugi strani veze	00000011
lokalna procedurna napaka	00000101
mreža je zasedena	00000111

Diagnostična koda v
RESET INDICATION paketih

Diagnoza ni določena	00000000

DTE in DCE RESET CONFIRMATION
paketni format

8	7	6	5	4	3	2	1
0	0	0	1	Številka kanalne skupine			
0	0	1	0				
Številka logičnega kanala							
0	0	0	1	1	1	1	1

4. SKLEP

Današnji komunikacijski protokoli so le abeceda protokolov prihodnosti. Že bežen pogled pokaže, da je v sekvencah še veliko redundančnih bitov, ki z njimi danes nimamo kaj početi. Kljub temu je smotrno, da so prisotni, tako kot so v začetku tega stoletja perspektivna mesta gradila železniške postaje za petdeset let naprej.

Tu so odprta vrata razvoju. Kdaj bomo napolnili okvire, ki smo si jih začrtali, je odvisno od naraščanja komunikacijskih potreb, torej od napredka človekovih dejavnosti.

5. LITERATURA

1.CCITT: FINAL REPORT ON THE WORK OF STUDY GROUP VII,PART III: PROPOSALS FOR NEW AND REVISED SERIES X RECOMMENDATIONS. GENEVA, 1976.

2.DR.C.SOLOMONIDES: X25 INTERFACE DEFINITION. THE NATIONAL COMPUTING CENTRE LIMITED, MANCHESTER, APRIL 1977.

3.IBM SYNCHRONOUS DATA LINK CONTROL, GENERAL INFORMATION.

4.MOSTEK TECHNICAL MANUAL FOR SERIAL I/O CONTROLLER MK3884/3885.

5.GLUŠAC D.: POUZDANOST PRENOSA PODATA IZMEDJU MAGNETNIH MEDIJUMA I CRC-KONTROLA, INFORMATICA 1977/3.

DMA PROCESORI U MIKRO RAČUNARSKIM SISTEMIMA

M. KOVAČEVIĆ
D. NOVAK
B. KASTELIC
A.P. ŽELEZNIKAR

UDK: 681.3 – 181.4

INSTITUT JOŽEF STEFAN, LJUBLJANA

Članak daje pregled osnovnih principa djelovanja direktnog pristupa memoriji te pregled savremenih DMA komponenti iz nekoliko različitih mikroprocesorskih familija. Istaknute su prednosti komuniciranja centralnog procesora sa perifernim jedinicama u DMA režimu pred nekim drugim načinima. Članak, takodjer, detaljno prikazuje kako koristimo DMA procesore u konfiguracijama sa gipkim (floppy) diskom.

DMA processors in Microcomputer Systems. The article gives a survey for direct memory access pronciples. An overview of some new DMA components from various microprocesor producers is also given. The advantages of communication using DMA against other methods are emphasized. The usage of DMA processors in configurations with floppy disk is described in details.

1. Uvod

Savremena LSI tehnologija, uz brojne prednosti pred klasičnim rješenjima u pogledu cijene, sigurnosti, male potrošnje energije i jednostavnosti pruža velike mogućnosti za širu primjenu tehnički zahtjevnijih i savršenijih metoda obrade i prenosa podataka. Sa pojavom kompletnih mikro procesorskih familija dolazimo u priliku gdje jednostavnim povezivanjem nekoliko LSI komponenti iste familije ostvarujemo univerzalni ili specijalizirani sistem velikih mogućnosti. Jedna od takvih LSI komponenti sa vrlo širokim spektrom aplikacija je procesor za direktni pristup memoriji (DMA). Savremeni CRT kontroleri, disk kontroleri, kontroleri za sinhroni prenos podataka visoke brzine, kao i mnogi drugi kontroleri za komuniciranje sa perifernim napravama, sadrže DMA procesore. Isto tako, komunikacija izmedju dva procesora može teći preko DMA procesora pri čemu je isključena mogućnost konfliktnih situacija pri dijeljenju zajedničke memorije.

Razvoj sistemske programsku opreme upotrebom programskih jezika visokog nivoa za sisteme sa DMA procesorima još uvijek predstavlja teškoću, iako razvoj programske opreme na asemblerskom nivou predstavlja znatno olakšanje u odnosu na programiranje u sistemima sa klasičnim načinom prenosa podataka.

2. Principi DMA procesiranja

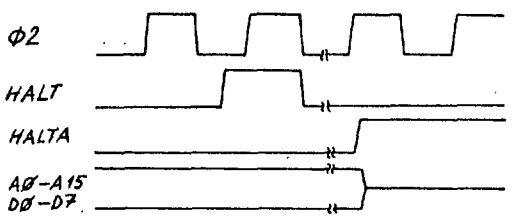
Direktni pristup memoriji teče mimo centralne procesne jedinice (CPU) tako da ne utiče na njen rad, odnosno da ne prekida izvodjenje tekućeg programa. Sve što CPU mora učiniti za DMA procesor je inicijalizacija prenosa. Po incijalizaciji teče prenos podataka izmedju CPU jedinice i periferne naprave na zahtjev periferne naprave. Prenos može biti organiziran pojedinačni ili blokovno. Po prenosu odredjenog broja slogova (byte), koji predstavljaju blok, DMA procesor može prekidnim signalom ili statusnim bitom obavijestiti CPU jedinicu o završenom prenosu bloka.

Pri inicijalizaciji DMA procesora potrebno je specifirati sljedeće elemente:

- početna adresa memorijskog bloka za DMA pristup
- broj slogova bloka koji treba da se prenese
- smjer prenosa i vrsta operacije (čitanje, pisanje, verifikacija ...)
- prioritet DMA kanala ako ih ima više
- izbor drugih eventualnih mogućnosti (dekrementiranje adrese, inkrementiranje adrese, ciklično ponavljanje prenosa bloka i slično).

Prenos sloga u DMA režimu može biti realiziran na dva osnovna načina uz neke varijacije. Prvi način prenosa je takozvani HALT način. Ovaj način predstavlja jednostavniji oblik DMA procesiranja za kojega se može slobodno reći da ne predstavlja striktno realizirane zahtjeve DMA procesiranja. Naime, pri HALT načinu DMA prenosa podataka izmedju memorije i perifernih jedinica dolazi do zaustavljanja rada CPU jedinice često, kada je u pitanju prenos obimnijih blokova, za neprihvatljivo dug vremenski period. I pored tog značajnog nedostatka vrlo često susrećemo HALT način DMA procesiranja. Glavne prednosti ovog načina su jednostavnost i kompatibilnost sa većinom mikro procesorskih CPU jedinica.

DMA procesor , djelujući u HALT režimu, komunicira sa CPU jedinicom preko signala HALT i HALTA (ponegdje HOLD i HOLDA). Signal HALT predstavlja ulazni signal u CPU jedinicu. Njegov aktivni nivo zaustavlja rad CPU jedinice sve dotle dok se stanje tog signala ne promijeni. Istovremeno CPU jedinica gubi kontrolu nad adresnim, podatkovnim i kontrolnim vodilom tako da sve linije CPU jedinice prelaze u stanje visoke impedanse. Svoju spremnost za DMA djelovanje potvrdjuje CPU jedinica signalom HALTA (halt acknowledge) prelazeći istovremeno u halt stanje pošto završi tekući mašinski ciklus, kako prikazuje slika 1.

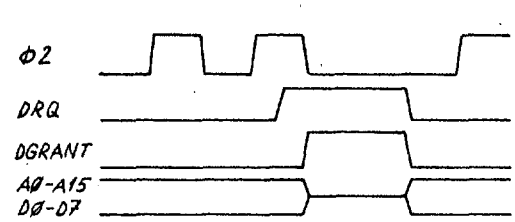


Slika 1: HALT tehnika DMA procesiranja.

U trenutku kada se pojavi aktivni nivo signala HALTA, kontrolu nad vodilom preuzima DMA kontroler, tako da omogući prenos podataka kako je pri inicijalizaciji predvidjeno.

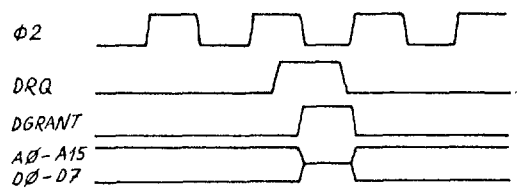
Drugi način DMA prenosa upotrebljava tehniku kradje ciklusa (cycle stealing). Ova tehnika ima prednosti pred HALT tehnikom pošto u znatno manjoj mjeri "smeta" CPU jedinici pri njenom radu. U ovom slučaju je u potpunosti ispunjen zahtjev DMA procesiranja da CPU jedinica ne smije "osjećati" prisustva DMA operacija. Medjutim, tehnika kradje ciklusa ipak utiče na rad CPU jedinice tako što je usporava. Pri svakom prenosu jednog sloga produži se osnovni ciklus sistemskog kloka dva puta.

Komunikacija izmedju DMA procesora i CPU jedinice teče, u ovom slučaju, preko signala DRQ i DGRANT. DMA ciklus započinje tako što DMA procesor , na zahtjev periferne naprave, postavi signal DRQ u aktivno stanje. Taj signal djeluje na generator sistemskog kloka, koji na početku sljedeće periode odgovori aktivnim nivoom signala DGRANT. Signal DGRANT djeluje na CPU jedinicu tako da sve linije adresnog, podatkovnog i kontrolnog vodila prelaze u stanje visoke impedanse. DMA kontroler, nato, preuzima kontrolu nad vodilom tako da omogući prenos podataka izmedju memorije i periferije. Pošto je jedan slog prenesen, kontrolu nad vodilom ponovo preuzima CPU jedinica. Ova metoda u suštini djeluje tako da produžava prvu poluperiodu sistemskog kloka onda kada se zahtijeva DMA operacija. Slika 2 pokazuje opisani način DMA procesiranja.



Slika 2: Tehnika kradje ciklusa DMA procesiranja.

Poboljšana metoda DMA procesiranja tehnikom kradje ciklusa, poznata kao skriveni DMA, djeluje tako da dijeli cijelu periodu sistemskog kloka namjenski na dvije poluperiode. U prvoj poluperiodi se na zahtjev vrši DMA, a u drugoj regularno djelovanje CPU jedinice. Ovaj način DMA procesiranja je moguć u onim mikro računarskim sistemima sa dvofaznim sistemskim klokom, odnosno tamo gdje je perioda sistemskog kloka sastavljena iz neaktivne i aktivne poluperiode. Diagram na slici 3 pokazuje vremenski tok dogadjaja pri skrivenom DMA procesiranju.



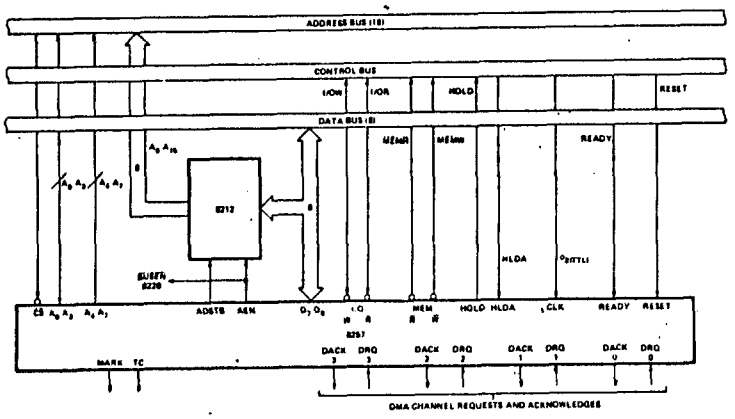
Slika 3: Vremenski tok dogadjaja pri skrivenom DMA procesiranju.

Važno je napomenuti još i problem istovremenog djelovanja DMA procesora tehnikom kradje ciklusa sa kontrolerom za osvježavanje dinamičnih memorija. Naime i DMA procesor i kontroler za osvježavanje dinamičnih memorija upotrebljavaju tehniku kradje ciklusa što stvara mogućnost konflikta. Ovaj problem rješavamo uključivanjem arbitražnog logičnog kola , kao što je opisano u (1).

3. Mikro računarske DMA komponente

Većina modernih mikro računarskih familija sadrži i DMA procesor kao samostalno integrirano kolo, koje se uz minimalnu dodatnu materijalnu opremu uključuje u mikro računarske konfiguracije. Pogledajmo strukturu DMA procesora iz mikro računarskih familija Intel 8080, MC6800 te Z80.

DMA procesor familije 8080 nosi oznaku 8257. To je četverokanalni DMA procesor, specijalno projektiran za jednostavniji prenos podataka velikom brzinom u Intelovim mikro procesorskim sistemima. Njegova primarna funkcija je generiranje memorijskih adresa, što omogućava perifernim jedinicama čitanje ili pisanju podataka direktno iz odnosno u memoriju. Zahtjev za sistemsko vodilo se ostvaruje putem HOLD funkcije mikro procesora 8080. Procesor 8257 ima logiku za rješavanje problema prioriteta, pri čemu generira komponiran HOLD zahtjev CPU jedinici. Isto tako ovaj procesor vrši brojanje DMA ciklusa za svaki kanal sa mogućnošću generiranja kontrolnog signala koji obavještava perifernu napravu o završenom prenosu programiranog broja DMA ciklusa. Drugi izlazni kontrolni signali pojednostavljuju prenos sektoriranih podataka te širenje na druge 8257 procesore u sistemima koji zahtijevaju više od četiri DMA kanala. Za djelovanje procesora 8257 je neophodno integrirano kolo 8212 iz iste familije. Naime, četrdeset nogica kola 8257 ne omogućava paralelnu prisutnost svih 16 adresnih linija te svih podatkovnih linija. Zbog toga upotrebljavamo ulazna/ izlazna paralelna vrata 8212 pomoću kojih iz podatkovnih linija , upotrebom specijalnih kontrolnih signala AEN i ADSTB, proslijedimo viših osam bitova adrese na adresno vodilo kako prikazuje slika 4.



Slika 4: Priključenje DMA procesora 8257 na vodilo mikro procesora 8080.

Procesor 8257 poznaje tri različita načina djelovanja. To je DMA čitanje iz memorije, DMA pisanje u memoriju i DMA verifikacija pri kojoj ne dolazi do prenosa podataka. U trenutku kada je 8257 u stanju "slave" njegova posebna logika za čitanje i pisanje prihvata I/OR- ili I/OW- signal iz sistemskog vodila, dekodira najniža četiri bita adrese (A0-A3) te, nato, piše sadržaj podatkovnog vodila u adresirani registar ili postavlja sadržaj adresiranog registra na podatkovno vodilo. U toku DMA ciklusa logika za čitanje i pisanje generira I/O signale za čitanje i pisanje (I/OR-,I/OW-) te signale za čitanje iz memorije (MEMR-) ako je u pitanju DMA read ciklus ili signal za pisanje u memoriju (MEMW-) ako je u pitanju DMA write ciklus. Poseban blok kontrolne logike procesora 8257 generira 16-bitnu adresu te odgovarajuće kontrolne signale. Signal READY sinhronizira djelovanje DMA procesora u sistemima sa sporim memorijama. HRQ i HLDA predstavljaju komunokacijske signale izmedju DMA procesora i CPU jedinice. ADSTB je signal koji omogućava upis viših osam bitova podatkovnog vodila u ulazni/ izlazni port 8212. AEN upotrebljavamo za isključenje CPU jedinice sa sistemskog vodila kao i za proslijedjenje sadržaja porta 8212 na linije adresnog vodila A8-A15.

Procesor 8257 ima kontrolni registar, statusni registar te četiri para 16-bitnih registara za DMA adresu i za brojanje DMA ciklusa. Pri tome bit 15 brojačkog registra svojom aktivnom vrednošću inicijalizira DMA read operaciju, dok bit 14 inicijalizira DMA write operaciju.

Bitovi kontrolnog registra imaju sljedeće značenje:

bit 0 do bit 3 -izbor aktivnog DMA kanala
bit 4 - izbor rotacijskog prioriteta pri kojem se poslije svakog DMA ciklusa promijeni prioritet svakog DMA kanala.
bit 5 - aktivna vrijednost omogućava djelovanje u sistemu sa sporom memorijom.
bit 6 - aktivna vrijednost ovog bita onemogući DMA zahtjeve, pošto je programirani broj DMA ciklusa izvršen.
bit 7 - omogućava ciklično ponavljanje prenosa odredjenog bloka DMA ciklusa u kanalu 2

Statusni registar daje informacije o izvršenom prenosu bloka podataka za svaki kanal te pomoćne informacije pri djelovanju procesora u režimu cikličnog ponavljanja.

Sljedeći program inicijalizira četvrti kanal DMA procesora tako da poslije 256 DMA ciklusa generira TC signal, odnosno prestaje da prihvata DMA zahtjeve od strane periferne jedinice. U programu je predvidjeno DMA čitanje iz memorije sa početnom adresom 0000.

```
-- -- --
MVIA  0
OUT   DMAADD   upis 16-bitne adrese (0)
OUT   DMAADD   u DMA adresni registar
MVIA  FFH      upis broja DMA ciklusa
OUT   DMATC    u brojački registar
MVIA  80H      operacija čitanja
OUT   DMATC
MVIA  48H      aktiviramo kanal 4 te
OUT   DMACNT   izberemo način u
               kojem se izvrši samo
               odabran broj DMA ciklusa
-- -- --
```

Dma procesor familije MC6800 sa oznakom MC6844 pruža veće mogućnosti za DMA procesiranje pošto poznaje dva načina djelovanja i to HALT i kradja ciklusa. Osim toga znatno je jednostavnija ugradnja ovog elementa u sistem pošto ne zahtijeva dodatne elemente za regularno djelovanje. Kao kod 8257 procesora i u ovom slučaju imamo četvero kanalni sistem sa mogućnošću programiranja prioriteta. Logika za komuniciranje ovog elementa sa sistemskim vodilom uključuje logiku za selektiranje , čitanje i pisanje, prekidnu logiku, prenosnu logiku zahtjeva i odobrenja te logiku koja omogućava prenos podataka preko dvosmjernog podatkovnog vodila. Funkcionalna konfiguracija ovog procesora je programirana preko podatkovnog vodila. Svaki od četiri kanala može, za razliku od procesoora 8257, istovremeno djelovati u posebno programiranoj funkcionalnoj konfiguraciji. Programirljivi kontrolni registri omogućavaju kontrolu nad lokacijom prenosa i dužinom, načinom prenosa, prioritetom servisiranja DMA zahtjeva, cikličnim ponavljanjem prenosa te prekidnim signalima. Statusne i kontrolne linije omogućavaju kontrolu nad perifernim napravama. Procesor MC6844 uključuje sljedeće osobine:

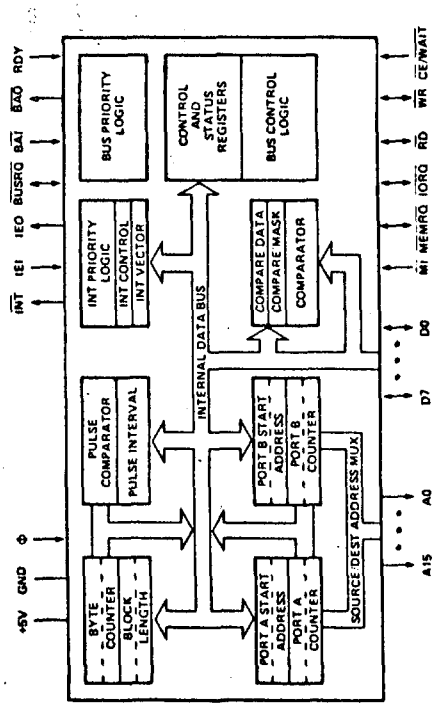
- *četiri DMA kanala, od kojih svaki ima 16-bitni adresni registar , 16-bitni registar za brojanje DMA ciklusa te kontrolni i statusni registar*
- *1 M slog/sec maksimalna brzina prenosa*
- *selekcija fiksnog ili rotacijskog sistema prioriteta servisiranja DMA zahtjeva*
- *odvojeni kontrolni biti za svaki kanal*
- *mogućnost djelovanja sa inkrementiranjem ili dekrementiranjem adrese*
- *programirljivi prekidni te DMA-end signali.*

Svih 15 registara koje sadrži procesor MC6844 su tipa čitanje/ pisanje pri čemu su samo neki statusni biti tipa samo čitanje. Pored adresnih, brojačkih i kontrolnih registara (za svaki kanal po jadan) ovaj procesor sadrži još i sljedeće registre:

- Kontrolni prioritetni registar omogućava rješavanje prioritetnih problema DMA zahtjeva različitih DMA kanala. Biti 0-3 svojom aktivnom vrijednošću uključuju odgovarajuće DMA kanale u sistem. Bit 7 pruža mogućnost izbora izmedju statičnog prioritetnog sistema (kanali su razvrstani po sljedećem prioritetu: 0,1,2,3) i rotirajućeg prioritetnog sistema, gdje se poslije svakog DMA ciklusa mijenja prioritetna lista.
- Kontrolni prekidni registar. Biti 0-3 svojom aktivnom vrijednošću omogućavaju generiranje prekidnog signala iz pojedinog kanala. Bit 7 je tipa samo čitanje, a označava to da je DMA procesor poslao prekidni signal CPU jedinici pošto je obavio svoj zadatak.
- Kontrolni registar cikličnog ponavljanja. Preko ovog registra je omogućeno ciklično ponavljanje prenosa programiranog bloka u kanalima 0,1 i 2. Kanal 3 nemože djelovati u ovom režimu. Bit 0 registra za ciklično ponavljanje omogućava djelovanje procesora u režimu cikličnog ponavljanja. Biti 1 i 2 adresiraju kanal koji djeluje u ovom režimu. Pri cikličnom ponavljanju se sadržaj adresnog i brojačkog registra kanala 3 prenosi u odgovarajuće registre selektiranog kanala prije svakog ponavljanja.

Očito je da DMA procesor MC6844 predstavlja jednu od najsposobnijih DMA komponenti koje se danas mogu naći na tržištu.

Posobnost u svijetu mikro računarskih DMA procesora predstavlja Z80DMA procesor iz mikro računarske familije Z80. Ovaj procesor je jednokanalna naprava koja generira sve potrebne adresne, kontrolne i vremenske signale za prenos podataka izmedju dva porta u sistemu sa procesorom Z80. Oba porta mogu biti ili glavna memorija ili periferna ulazna/ izlazna jedinica što se programsko odredjuje. To je osobina po kojoj se Z80DMA procesor bitno razlokuje od drugih, koji većinom imaju sposobnost prenosa samo izmedju memorije i neke od ulaznih/ izlaznih perifernih naprava.



Slika 5: Z80DMA blok dijagram.

Ovaj procesor poznaje tri načina djelovanja i to način pri kojem imamo samo prenos podataka, način u kojem se podatci samo traže te kombinirani način. U toku DMA coklusa se generiraju dvije adrese, jedna za izvorni port i druga za ponorni. Prenos je moguć na četiri načina i to slog po slog, neprekidno sve dok su oba porta spremna za prenos, neprekidno tako da se CPU jedinica isključi dok se prenos ne završi te transparentni način pri kojem se kradu ciklusi osvježavanja. Vremenske osobine svih signala mogu biti programirane tako da se DMA procesor prilagodjava vremenskim zahtjevima svakog porta (izvora ili ponora). Procesor može biti programiran tako da generira prekide po prenesenom programiranom bloku, po uspješnom traženju ili po identifikaciji stanja "spremen". Cijeli prethodni ciklus operacija može biti jednostavno ponovljen. Procesor može signalizirati trenutak kada je odredjeni broj slogova prenesen bez potrebe zaustavljanja DMA prenosa. Kanal može biti aktiviran, deaktiviran ili resetiran pod programskom kontrolom. Status kanala je uvijek dostupan za CPU jedinicu. Procesor omogućava maksimalnu brzinu prenosa od 1,2 M sloga u sekundi.

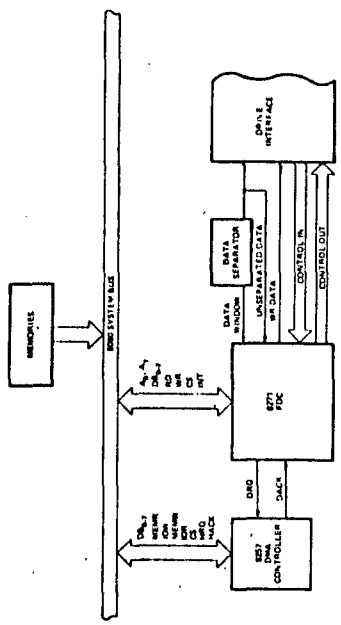
Procesor Z80DMA je sastavljen iz nekoliko logičnih podsklopova od kojih svaki ima posebnu funkciju. To su medjuspoj za vodilo koje obezbjedjuje potrebna ulazna/ izlazna linijska pojačala, kontrolna logika i registri pomoću kojih se realizira programirana operacija, podsklop za generiranje adresa , brojanje slogova i generiranje drugih signala koji ima zadatak da održava odgovarajuće adrese za oba porta uz mogućnost dekrementiranja i inkrementiranja. Tu je još i podsklop za oblikovanje vremenskih osobina signala za čitanje i pisanje za oba porta, podsklop za traženje te podsklop za kontrolu prekida i DMA zahtjeva.

Programer ima na raspolaganju sljedeće registre:

- Kontrolni registar: drži DMA kontrolne informacije o operaciji i načinu izvršenja te o zahtjevima za generiranje prekidnih i drugih signala.
- Registar za odredjivanje oblika signala za čitanje i pisanje za oba porta.
- Registar prekidnog vektora: drži osam bitni prekidni vektor čiji se sadržaj ispisuje na podatkovno vodilo pri generiranju prekida.
- Registar za specifikaciju dužine bloka: sadrži broj slogova koji treba da bude pretražen ili prenesen.
- Brojač slogova: sadrži broj već prenesenih ili pretraženih slogova pri prenosu ili pretraživanju bloka.
- Uporedjivački registar: sadrži slog koji treba da se traži u programiranom bloku.
- Maskirni registar: sadrži osam bitnu masku koja odredjuje bitove u uporedjivačkom registru koji treba da se usporedjuju pri traženju.
- Registar startne adrese (port A i B): Sadrže startnu adresu za oba porta za DMA prenos. Pri djelovanju u režimu "samo traženje" mora biti odredjena samo jedna adresa. Ako je jedan od portova neka I/O naprava onda je odgovarajuća adresa fiksna.
- Kontrolni registar za statusne signale: drži informaciju o dužini bloka poslije kojeg se pojavljuje signal na INT nogici.
- Statusni registar: sadrži statusne informacije u vezi sa traženjem , prenosom bloka, spremnosti za rad i slično.

4. Programiranje DMA procesora

Programiranje DMA procesora, slično programiranju svih perifernih kontrolera, se odvija upisivanjem kontrolnih i podatkovni vrijednosti u registre procesora. Pri tome je često potrabno čitati statusne podatke iz drugih registara. Pogledajmo kako je potrebno inicijalizirati DMA procesor 8257 za djelovanje u konfiguraciji sa kontrolerom za gipki disk, kao što pokazuje slika 6.



Slika 6: Kontroler za gipki disk sa DMA procesorom

Za djelovanje DMA procesora u ovoj konfiguraciji je potreban vrlo kratak inicijalizacijski program za čitanje i pisanje više zaporednih sektora (128 slogova) na gipki disk. Svi drugi programski segmenti, potrebni za pogon diska se odnose na disk kontroler 8271.

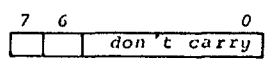
Komunikacija izmedju osnovnih kontrolnih programa za pogon diska i okoline može da teče preko ovakvog bloka parametara:

TRAKNO
SECNO
NOSEC
BUFFH
BUFFL
DRSEL
OPER

Pri tome TRAKNO sadrži adresu traga na disketi nad kojim treba da se izvrši operacija, SECNO sadrži adresu prvog sektora na tragu TRAKNO nad kojim treba da se izvrši operacija, BUFFH i BUFFL sadrže adresu memorijskog bloka, DRSEL sadrži informaciju o izboru diskovnog pogona, te OPER odredjuje vrstu operacije. Inicijalizacijski program za DMA procesor upotrebljava NOSEC, OPER te BUFFH i BUFFL parametre iz opisanog bloka parametara. Inicijalizacijski program može izgledati ovako:

```
--  --  --
LDA   BUFFL   SET LOW BYTE OF
OUT   DMAADD  ADDR.  OF DMA BUFFER
LDA   BUFFH   SET HIGH BYTE OF
OUT   DMAADD  ADDR.  OF DMA BUFFER
LDA   NOSEC   LOAD NO.  OF SECTORS
RLC           MULTIPLY BY 128
RLC
RLC
RLC
RLC
RLC
RLC
PSHA  PSW     SAVE A AND PSW
ANI   80H     SEPARATE LOW BYTE
OUT   DMATC   SET TERMINAL COUNT REG
POP   PSW     LOW BYTE
ANI   7FH     SEPARATE HIGH BYTE
LXIH  OPER    INCLUDE OPERATION
ORAM
OUT   DMATC   SET TERMINAL COUNT
MVIA  41H     SET DMA MODE REG
OUT   DMACNT  GO CHANEL 0, TC MODE
--  --  --
```

Ovaj inicijalizacijski segment predpostavlja sljedeći format parametra OPER:



bit 6=1 ⇒ operacija DMA čitanje
bit 7=1 ⇒ operacija DMA pisanje

Pri tome se predpostavlja hardwareska veza izmedju disk kontrolera 8271 i kanala 0 DMA procesora (u DMA mode registar je upisana vrijednost 41H).

U programu je izvršeno upisivanje adrese DMA memorijskog bloka te broja slogova koji trebaju da se prenesu, pošto smo to izračunali (množenjem NOSEC sa 128). Ovakav segment se mora uključiti u podprogram za čitanje ili pisanje više zaporednih sektora sa odnosno na disk. Dodatne asemblerske naredbe tog podprograma se odnose na upis parametara i kontrolnih slogova u registre disk kontrolera, našto započinje prenos podataka izmedju diska i memorije pod kontrolom DMA procesora. Kraj prenosa označava prekidni signal kojega može generirati disk kontroler ili DMA procesor.

Posebno zanimljiv problem se pojavljuje pri programiranju u sistemima sa DMA procesorom u multiprogramskom ili multi procesorskom režimu. Najnoviji jezici, konstruirani za multiprogramske i multi procesorske aplikacije, te aplikacije programiranja u realnom vremenu ne sadrže potrebne konstrukte za rješavanje problema sinhronizacije i uzajamne isključenosti u sistemima sa DMA procesorima. Naime, takvi jezici kao na primjer Modula (2) u velikoj mjeri zasnivaju rješenja tih problema na pojmovima prekida, prekidnih programa te prioriteta prekida. U sistemima sa DMA procesorima ne postoji mogućnost realizacije operacija u vezi sa signalom (3,4) (send, wait) pošto DMA prenos jednog sloga izmedju memorije i periferne jedinice nije vezan na prekidni program niti na bilo kakav program izuzev inicijalizacijskog. Osjeća se nedostatak mogućnosti definiranja hardwareskih objekata (npr. registri) kao objekata koji su po prirodi sposobni vršiti operacije u vezi sa signalima.

5. Zaključak

DMA procesori su predstavljali težak problem za projektante i proizvodjače mikro procesorskih LSI komponenti, što dokazuje

njihova dosta kasnija pojava na tržištu. To ujedno može da dočara kompleksnost, a isto tako i sposobnost DMA procesora koji su već na raspolaganju. Veću primjenu DMA procesora možemo u budućnosti očekivati u najrazličitijim aplikacijama.

Literatura

(1) D.Novak: Uporaba taktnega generatorja MC6875 v sistemih z dinamičnim pomnilnikom in DMA prenosom, Informatica 4, 1978.

(2) N. Wirth: Modula: A language for modular multiprograming, Software- Practice and Expereience, 7, 1 (Jan. 1977), 3-35.

(3) B. Hansen: Operating System Principles, Prentice Hall, Englewood Cliffs, N.J. 1973.

(4) D. Novak, M. Exel, M. Kovačević, B. Kastelic: Modula: jezik prihodnosti za mikro računalniške aplikacije, Informatica 2, 1979.

LITERATURA IN SREČANJA

3-5 dec. San Diego, California , ZDA

1979 WINTER SIMULATION CONFERENCE
Informacije: Mitchell G. Spiegel , FEDSIM/NA, Washington, DC 20330, USA

10-12 dec. Pacific Grove, California, ZDA

7TH SYMPOSIUM ON OPERATING SYSTEM PRINCIPLES
Informacije: Conf.chm. Michael D. Schroeder, Xerox Palo Alto Research Centre, 3333 Coyote Hill Road, Palo Alto, CA 94304, USA.

10-12 dec. London, Velika Britanija

LONG-LIFE SOFTWARE PLUS STATE OF THE ART TUTORIAL USER PARTICIPATIVE SYSTEMS
Organizator: INFOTECH
Informacije: Special Events Division, Infotech International Limited, Nicholson House, Maidenhead, Berkshire SL6 1LD, England

LETO 1980

9-11 jan. Oxford, Velika Britanija

1980 CONFERENCE ON PATTERN RECOGNITION
Informacije: Josef Kittler, Nuclear Physics Laboratory, Keble Road, Oxford OX1 3RH, England

28 jan.- 1 feb. Caracas, Venezuela

PANEL 80 EXPODATA
Informacije: Secretaria General PANEL 80, Coordinacion De Computacion, Universidad Simon Bolivar, Sartanija, Baruta, Edo. Miranda, Apartado Postal No. 80659, Caracas 108, Venezuela

30 jan.-1. feb. Monterey, California, ZDA

INTERNATIONAL SYMPOSIUM ON MICROCOMPUTERS AND THEIR APPLICATION
Informacije: Secretary, MIMI-80 (Monterey), Box 2481, Anahaim, CA 92804.

12-14 feb. Kansas City, ZDA

ACM COMPUTER SCIENCE SONFERENCE
Informacije: Conf. chm. Earl J. Schweppe, Dept. of Computer Science, University of Kansas, Lawrence, KS 66044

4-6 marec Zurich, Švica

1980 INTERNATIONAL ZURICH SEMINAR ON DIGITAL COMMUNICATIONS
Informacije: Secretariat 1980 Internationa Zurich Seminar, D. Hug, Dept. ENF, BBC Brown, Boveri and Co. Ltd., CH-5401 Baden, Switzerland

12-14 marec, Versailles, Francija

INTERNATIONAL SYMPOSIUM ON DISTRIBUTED DATA BASES
Organizator: IRIA
Informacije: IRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78150 Le Chesnay, France

12-14 marec Kiel, ZRN

GI-NTG CONFERENCE ON COMPUTER ARCHITECTURE AND OPERATING SYSTEMS
Informacije: Prog. chm. G. Zimmermann, Institut für Informatik und Prakt. Math., Universität Kiel, D-2300 Kiel, Germany

17-21 marec Dunaj, Avstrija

6TH INTERNATIONAL CONGRESS ON DATA PROCESSING IN EUROPE
Organizator: Arbeitgemeinschaft für Datenverbeitung
Informacije: Sekretariat, 6 Internationaler Kongress Datenverbeitung im Europäischen Raum, c/o Interconvention, P.O.Box 35, A-1095 Vienna, Austria

24-28 marec Jahorina, Jugoslavija

IV. BOSANSKOHERCEGOVAČKI SIMPOZIJUM IZ INFORMATIKE "JAHORINA 80"
Organizator: Elektrotehnički fakultet Sarajevo
Informacije: Elektrotehnički fakultet Sarajevo, Odsjek za informatiku, za simpozijum, 71000 Sarajevo, Toplička cesta bb, telef. 071 521-677/132

31 marec-2 april Brighton, Velika Britanija

CAD 80 - 4th INTERNATIONAL CONFERENCE ON COMPUTERS IN ENGINEERING AND BUILDING DESIGN
Informacije: Conf. chm. Gareth Jones, IPC Science and Technology Press Ltd., 32 High st., Guildford, Surrey, England GU1 3EW

8-11 april Dunaj, Avstrija

5TH EUROPEAN MEETING ON CYBERNETICS AND SYSTEMS RESEARCH
Organizator: Austrian Society for Cybernetic Studies
Informacije: Austrian Society for Cybernetic Studies, Schottengasse 3, A-1010 Vienna, Austria

16-20 april Skopje, Jugoslavija

BIOMEDICINSKA KIBERNETIKA 1980
Organizator: Društvo za biokibernetiko
Informacije: M. Kon-Popovska, Matematički fakultet, pp 504, 91000 Skopje

30 april-2 maj Pittsburgh, ZDA

11TH ANUAL PITTSBURGH CONFERENCE ON MODELING AND SIMULATION
Organizator: School of Engineering, University of Pittsburgh
Informacije: William G. Vogt or Marlin H. Mickle,Modeling and Simulation Conference, 348 Benedum Engineering Hall University of Pittsburgh, Pittsburgh, Pennsylvania 15269, USA

28-30 maj Shiraz, Iran

IFAC/IFIP CONFERENCE ON SYSTEM APPROACH AND COMPUTER APPLICATIONS FOR DEVELOPMENT
Organizator: Iran Society of Automatic Control Engineers
Informacije: Secretary of IFAC/IFIP Conference, Iran 1980, PO Box 737, Shiraz, Iran

16-19 jun. Warsaw, Poljska

3RD IFAC/IFORS CONFERENCE ON MODELLING AND CONTROL OF NATIONAL ECONOMIES
Organizator: Systems Research Institute -Polish Academy of Sciences
Informacije: Dr. M. Lipiec, Systems Research Institute Polish Academy of Sciences, 6 Newelska st., 01-477 Warsaw, Poland

23-27 jun. Brussels, Belgija

WORLD FORUM OF INTERNATIONAL TRANSNATIONAL ASSOCIATIONS
Organizator: Union of International Associations (UAI)
Informacije: UAI, rue aux Laines 1, 1000 Brussels, Belgium

23-27 jun. Roma, Italija

IBI WORLD CONFERENCE ON TRANSBORDER DATA FLOW POLICIES
Organizator: Intergovernmental Bureau for Informatics
Informacije: IBI, Viale Civilta del Lavoro 23, POB 10253, 00144 Rome, Italy

30 jun.-4 jul. Bratislava, ČSSR

INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELIGENCE AND INFORMATION CONTROL SYSTEMS OF ROBOTS
Organizator: The Scientific Board for Artificial Inteligence of the Systems Analysis Committee of the Presidium of the Academy of Sciences of the USSR and the Institute of Tecnhnical Cybernetic of the Slovak Academy of Sciences
Informacije: Institute of Technical Cybernetic of the Slovak Academy of Sceinces, 80931 Bratislava, Dubravska 9, ČSSR

16-19 sept. Lausanne, Švica

EUSIPCO 80
Organizator: Swiss Federal Institut of Technology
Informacije: Mrs Stehlé EUSIPCO-80, Swiss Federal Institut of Technology, 16, Chemain de Bellerive, CH-1007 Lausanne, Switzerland

29 sept.-4 okt. Tokyo, Japonska

MEDINFO 80: 3RD WORLD CONFERENCE ON MEDICAL INFORMATICS
Organizator: IFIP TC4
Informacije: IFIP Foundation, 40 Paulus Potterstraat, 1071 DB Amsterdam, Netherlands

oktober, Kyoto, Japan

CONFERENCE ON MAN-MACHINE COMMUNICATIONS IN CAD AND CAM
Organizator: IFIP WG5.2, 5.3
Informacije: IFIP Secretariat, 3 rue du Marché, CH-1204, Geneva, Switzerland

1-3 okt. Kyoto, Japonska

10TH INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING
Organizator: The Technical Committee on Fault-Tolerant Computing of the IEEE Computer Society
Informacije: Secretariat of FTCS-10, Dept. of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Kyoto 606, Japan

6-12 okt. Bled, Jugoslavija

INFORMATICA 80, 15. JUGOSLOVANSKI MEDNARODNI SIMPOZIJ O OBRAVNAVANJU PODATKOV
Organizator: Slovensko društvo Informatika
Informacije: Slovensko društvo Informatika, Jamova 39 61000 Ljubljana

14-17 okt. Dusseldorf, ZRN

6TH IFAC/IFIP INTERNATIONAL CONFERENCE ON DIGITAL COMPUTER APPLICATIONS TO PROCESS CONTROL
Organizator: VDI/VDE-Gesellschaft Mess-und Regelungstechnik
Informacije: VDI/VDE, Postfach 1139, D-4000 Düsseldorf 1 Germany.

VABILO K SODELOVANJU

Iduće godine tj. 16.-20.4. 1980, za vreme održavanja V. Medžunarodne izložbe "SAVREMENA MEDICINA 1980" u Skopju u organizaciji Društva za biokibernetiku SR Makedonije održat će se:

Simpozij BIOMEDICINSKA KIBERNETIKA 1980

Teme simpozija

1. Modeli bioloških i medicinskih sistema
2. Biomedicinska instrumentacija
3. Računari i mikroprocesori u biologiji i medicini
4. Veštačka inteligencija i robotika

Ovim vas pozivamo da svojim radovima doprinesete uspjehu simpozija. Predvid eno je da se radovi štampaju u posebnom Zborniku simpozija.

Molimo vas da obratite pažnju na slijedeće rokove:

1.12.1979 Rok za dostavu kratkog sadržaja na jednom od jugoslovenskih jezika

30.12.1979 Rok do kojeg će autori biti obavešteni o prihvatanju radova

15.2.1980 Rok za dostavu radova

Za detaljnije informacije obratite se na adresu:

mgr. Margita Kon Popovska
Matematički fakultet
poštanski pret. 504
91000 Skopje

25. JUBILARNI GODIŠNJI SKUP JUREMA

Održat će se 14. do 17. travnja/aprila 1980. u Zagrebu pod mottom

MJERENJE I AUTOMATIZACIJA = ŠTEDNJA ENERGIJE I SIROVINA, POVEČANJE PROIZVODNOSTI

Zamišljeno je da se ovim skupom predoči dvadesetpet godina intenzivnog rada ne prijenosu znanja i tehnologije i istakne doprinos JUREMA razvoju mjerenja, automatike, teorije sistema i kibernetike i njihovoj primjeni.

Pozivamo stručnjake da do 15.listopada/oktobra 1979. prijave referat. Sažetak mora biti dužine najviše 200 riječi na jednom od jezika jugoslovenskih naroda i na engleskom jeziku. Prvi izbor radova provest će se na temelju sažetaka, pa će do 1. studenoga/novembra 1979. autori biti izvješteni o prihvaćanju i zamoljeni da pripreme referat u skladu s uputama. Pripremljeni rad treba dostaviti najkasnije do 10. veljače/februara 1980.

25. jubilarni godišnji skup JUREMA obuhvatit će:

SAVJETOVANJE:

STANJE I PERSPEKTIVE RAZVOJA AUTOMATIZACIJE

Predmet rasprava na ovom savjetovanju bit će teme iz ovih područja:

PRIRODNA VRELA I OKOLIŠ

- pridobivanje nafte i plina
- rudarstvo
- opskrba vodom
- zaštita okoline

IZVORI I PRIJENOS ENERGIJE
PROIZVODNI PROCESI I OPERACIJE

- kemijska i farmaceutska industrija
- industrija nafte i petrokemija
- prehrambena industrija
- metalurgija
- industrija papira, gume, plastike
- industrija cementa, gradjevnog materijala
- tekstilna industrija

EKONOMSKI I SOCIJALNI ASPEKTI

- optimiranje
- odnos čovjek - stroj
- transfer znanja i tehnologije.

Naglasak je dakle na primjeni automatizacije, ali se očekuju i prilozi koji će obraditi pitanja proizvodnje elemenata i uredjaja za automatizaciju u nas.

3. SAVJETOVANJE O METROLOGIJI:

STANJE I RAZVOJ MJERITELJSTVA

Savjetovanje je zamišljeno kao mjesto na kojem će se široj stručnoj javnosti učiniti dostopni rezultati vlastitog rada na području teorije i primjene mjerenja. Očekuju se radovi iz ovih područja:

OPĆA PITANJA

- teorija mjerenja
- zakonska metrologija
- mjerna služba

PRIMJENA

- mjerni postupci i uredjaji
- mjerni uredjaji s računalom
- laboratorijska mjerna sredstva i uredjaji
- mjerna sredstva i uredjaji za motrenje zagadenosti okoline
- procesna mjerna oprema
- mjerna sredstva i oprema za rad u posebnim uvjetima

ZNANSTVENI RAD I OBRAZOVANJE

2. SIMPOZIJ O ELEKTROMAGNETSKOJ KOMPATIBILNOSTI

Predvidjeno je da se na ovom simpoziju obuhvate ove teme:

Utjecaj elektromotornih pogona na izvore električne energije:

- utjecaj na izmjeničnu mrežu/harmonici napona i struje, jalova snaga, fluktuacije napona, visokofrekventne i radiofrekventne smetnje/,
- utjecaj na istosmjernu mrežu /valovitost, prenaponi, VF i RF smetnje/,

Postizavanje elektromagnetske kompatibilnosti elektromotornih pogona i okoline:

- potiskivanje smetnji na izmjeničnoj i istosmjernoj mreži /filteri za smanjenje harmonika struje i napona, VF i RF smetnji, te kompenzatori jalove snage/,
- potiskivanje smetnji koje ometaju prijenos informacija,
- postizavanje elektromagnetske kompatibilnosti više elektromotornih pogona na istoj mreži.

Prihvatit će se i referati vezani i uz druga pitanja elektromagnetske kompatibilnosti.

2. SIMPOZIJ

AUTOMATIZACIJA I SIGURNOST ŽELJEZNIČKOG PROMETA

posvečen je temi

Informacijski sistemi za upravljanje prometom na pruga-
ma sa većim brzinama
pa su obuhvačena ova pitanja:

SIGNALIZIRANJE I GRANIČNA PODRUČJA

UPRAVLJANJE PROMETOM

- upravljanje saobraćajem
- vodenje vlakova
- uredjaji neophodni za djelovanje sistema
- problemi kompatibilnosti postojećih i novih sistema

TEHNIČNO RJEŠENJE UREDJAJA

- nadzor i upravljanje elementima i uredjajima
- prenos i upravljanje informacijama
- upravljanje prometom u staničnom području
- upravljanje prometom na pružnom odsjeku.

POUZDANOST

- prikupljanje podataka i ocjena pouzdanosti
- planiranje i održavanje pouzdanosti
- primjena podataka o pouzdanosti pri projektiranju i organizaciji održavanja

DEFINICIJE I STANDARDIZACIJA

Napomena: Pod većom brzinom podrazumivanje se brzine 120 do 200 km/sat.

6. SKUP O NASTAVI MJERENJA, AUTOMATIKE I TEORIJE SISTEMA

Kao i na dosadašnjim skupovima i na ovom će jedan od sekcija biti posvećena laboratorijskoj nastavi, a kao temeljna tema adabrano je aktualno pitanje:

Primjena mikroprocesora i mikroračunala u nastavi mje-
renja i automatike.

Posebno će biti još obradene teme povezanosti nastavnog i znanstvenog rada.

VABILO K SODELOVANJU

Iduće godine tj. 24.-28. marta 1980 održat će se

IV. Bosanskohercegovački simpozijum
iz informatike "JAHORINA 80"

Teme simpozija

1. ARHITEKTURA I HARDVER RAČUNARSKIH SISTEMA

- Arhitektura sistema
- Mikroprogramiranje
- Uticaj mikroprocesorskih i LSI kola na arhitekturu
- Multiprocesorski sistemi
- Mreže računara i distribuirani računarski sistemi
- Modeliranje i ocjena performansi računarskih sistema

2. SOFTVER

- Operativni i programski sistemi
- Programski jezici i prevodioci
- Razvojna sredstva i discipline
- Softver za upravljanje podacima
- Softver mikro računara
- Protokoli i komunikacioni softver
- Modeliranje i ocjena performansi programskih sistema

3. OBRAZOVANJE I INFORMATIKA

- Metodološki uticaji informatike na obrazovanje
- Računari u školama
- Terminologija i standardi
- Informatika na univerzitetu

4. INFORMACIONI SISTEMI I BAZE PODATAKA

- Teoretski aspekti
- Metode (jezici, modeli, tehnike) razvoja informacionih sistema
- Arhitektura informacionih sistema i baze podataka
- Modeliranje baze podataka
- Distribuirani informacioni sistemi: aspekti razvoja, tehnologije, organizacije i upravljanja
- Primjeri realizacije informacionih sistema

5. RAZVOJ DRUŠTVENOG SISTEMA INFORMISANJA

- Opšta načela društvenog sistema informisanja
- Priroda, funkcija i organizacija
- Zajedničke osnove i elementi jedinstva društvenog sistema informisanja
- Obrazovanje i istraživanje za potrebe razvoja društvenog sistema informisanja

6. RAČUNARSKO UPRAVLJANJE PROCESIMA

- Uticaj novih tehnologija na koncepte računarskog nadzora i upravljanja procesima
- Distribuirano računarsko upravljanje procesima
- Operativni sistemi za rad u realnom vremenu
- Prenos podataka - komunikacije u distribuiranim sistemima
- Primjeri konkretnih realizacija

7. PRIMJENA RAČUNARA U NAUCI I TEHNICI

- Numeričke metode u elektroenergetici
- Projektovanje uz pomoć računara
- Računari u medicini
- Aplikacije teorije grafova
- Primjena računara u drugim oblastima, (automatika, gradjevinarstvo, saobračaj, mašinstvo, itd)

AKTUELNA TEMA

SISTEMNI ASPEKTI PROBLEMA ENERGIJE

- Planiranje razvoja energetskih resursa, konverzija energije i upotrebe
- Aspekti modela i modeliranja u energetskim sistemima
- Problematika naučnih istraživanja: organizacija i sadržaj
- Informacioni aspekti upravljanja energetskim resursima

PRATEĆE MANIFESTACIJE

- Izložba stručne literature
- Okrugli sto
- Panel diskusije
- Prezentiranje novih proizvoda

Molimo vas, da obratite pažnju na slijedeće rokove:

15.11.1979 Rok za prijavo referata

10.1.1980 Rok za dostavu konačnog teksta

15.2.1980 Rok do kojeg će autori biti obavešteni o prihvatanju radova.

Za detaljnije informacije obratite se na adresu:

Elektrotehnički fakultet Sarajevo
Odsjek za informatiku (za simpozijum)
71000 Sarajevo, Topličká c. bb
telef. (071)521-677/132

SIMPOZIJ IN SEMINARJI INFORMATICA 79

Kot že vrsto let je bil Bled v prvem tednu meseca oktobra v znamenju osrednjega jugoslovanskega srečanja teoretikov in praktikov s področja računalništva in informatike. Srečanje je pripravilo Slovensko društvo Informatika v sodelovanju z našim Institutom ter Fakulteto za elektrotehniko v Ljubljani.

Na 14. simpoziju Informatica, so poročila strokovnjakov o raziskavah in rešenih problemih zajela skoraj vsa področja informatike in računalništva. Seminarji so bili s področja multiprocesiranja z mikroprocesorji, računalniških mrež ter načrtovanja softwara za novi 16 bitni mikroprocesor MC.

V ponedeljek 1.oktobra je bil v Festivalni dvorani na Bledu uraden začetek simpozija, ki ga je odprla in pozdravila vse prisotne, predsednica organizacijskega odbora mgr.Borka Džonova-Jerman-Blažič. Sledil je pozdravni govor predsednika Slovenskega društva Informatika dr.A.P.Železnikarja ter otvoritveni govor dr.I. Winklerja, predsednika Republiškega komiteja Izvršnega sveta SR Slovenije za raziskovalno dejavnost.

"Ne samo v najbolj razvitih državah, tudi pri nas postaja vse bolj očitno, kako pomembno je, da dobe delavci v temeljnih organizacijah združenega dela, delovnih organizacijah, krajevnih skupnostih in občinah pravočasne in natančne podatke, ki jih potrebujejo pri svojem delu. Odločitve, ki jih morajo sprejemati, so le tako lahko pravilne. Dobro razvitega in razvejanega sistema informiranja pa ni moč vpeljati brez sodobne računalniške opreme. Dosedanji razvoj domače računalniške industrije in znanosti s tega področja je namreč že pokazal, kolikšne dosežke lahko ponudi gospodarstvu in negospodarskim dejavnostim tako na domačem kot na tujem trgu", je med drugim v svojem govoru dejal dr.I.Winkler.

Nato so sledila povabljena predavanja prof.dr.S.Hana iz Instituta za informatiko, Novi Sad, dr.D.Barberja, direktorja European Informatics Network iz Velike Britanije, dr.R.J.Buhra, Carleton University iz Kanade, dr.M.Vukobratoviča, Institut Mihailo Pupun, Beograd, prof.S. Dobrenića, Ekonomski fakultet, Zagreb ter mgr.Komunjera iz Elektrotehne, Ljubljana.

V petih dneh simpozija so se vrstila predavanja, kratki referati in strokovna poročila, debate, formalni, neformalni pogovori in srečanja na naslednjih programskih področjih: programska oprema, računalniška oprema, teoretični in matematični vidiki obravnavanja podatkov, sistemi za upravljanje in administracijo, upravljanje procesov, razne uporabe v znanosti in tehniki, vzgoja, družboslovje, humanistika, medicina.

Program simpozija je bil izredno obsežen.V petih dneh simpozija je bilo predstavljenih 309 prispevkov, od katerih jih je bilo 90 v obliki predavanj. Ostali prispevki so bili predstavljeni kot posterji. Novi način predstavitve prispevkov je bil zelo ugodno sprejet med poslušalci in med avtorji. Na ta način so organizatorji simpozija omogočili udeležencev spremljanje večjega števila prispevkov, saj so paralelne sekcije odpadle. Neformalni pogovori z avtorjem ob posterju so se izkazali prikladnejši od klasičnega predavanja in časovno omejenih diskusij.

V okviru simpozija so bili organizirani tudi pogovori ob okrogli mizi z naslednjimi temami: "Vloga računalnika v ustvarjalnem zaznavanju umetniške grafike" - vodja pogovora je bil akademski slikar iz Trsta E.Zajc; "Družbeni sistemi informiranja"-vodji pogovora sta bila prof S.Han in prof.S.Dobrenić in "Robotika" vodja pogovora je bil prof.M.Vukobratović.

Izredno zanimanje so udeleženci pokazali za temo "Družbeni sistem informiranja", saj je v TV zaslonu hotela Park, ki sprejme 100 ljudi zmanjkalo prostora za vse zainteresirane. Zanimanje za ta pogovor ter pripombe udeležencev, je organizatorje opozorilo, da je potrebno tudi v naslednjih letih vključevati in organizirati v program simpozija pogovore s širšimi temami.

Široko zstavljen program simpozija je poleg obravnavanih ozko strokovnih vprašanj, omogočil tudi povezovanje s področij, ki na prvi pogled nimajo veliko skupnega z razraščajočo se računalniško tehnologijo. V spremljevalnem programu simpozija je bila vključena razstava slik "Scherzzo za matriko in figure" ter "Logični trenutki v barvah II", slikarja Edvarda Zajca, tržaškega Slovenca, ki sodi med vodilne ustvarjalce na področju računalniške grafike.

Edvard Zajc že več kot 10 let raziskuje tiste sestavine v računalnikovem jeziku in mišljenju, ki bi lahko v likovni umetnosti postale nov vir ustvarjalnih pobud. Slike odkrivajo posamezne trenutke določenega ustvarjalnega procesa in so kot take le informacija gledalcem o poti, ki sta jo z računalnikom prehodila.

Razstavo računalniško generiranih slik je spremljala razstava računalniške literature dveh založniških hiš ter razstava računalniške opreme. Svoje izdelke so predstavili ter pripravili demonstracijo delovanja opreme: Iskra TOZD Računalniki, Iskra TOZD Procesna tehnika, Tovarna meril Slovenj Gradec, Eurodia iz Avstrije ter Columbia iz Velike Britanije.

Simpozija se je udeležilo preko 400 strokovnjakov iz vseh jugoslovanskih republik ter iz 18 evropskih držav, ZDA in Kanade.

Letošnji simpozij je bil pomemben prispevek k medsebojnemu povezovanju ter izmenjavi izkušenj in informacij, saj je bilo ob izredno delovnih sekcijah in obsežnem programu le malo časa za izvenprogramske aktivnosti.

Tudi letos so sodelavci IJS, Fakultete za elektrotehniko in študenti izvedli ves organizacijsko-tehnični del posvetovanja. Ponovno je bil v veliko pomoč organizatorjem simpozija računalnik PDP 11/34.

Ob koncu gre zahvala vsem, ki so kakorkoli pripomogli k organizaciji letošnjega simpozija v želji, da prispevajo k čimboljši predstavitvi in informiranju o naši dejavnosti na področju računalništva in informatike.

RAZISKOVALNE NALOGE, PRIJAVLJENE NA RSS V LETU 1979

V tej rubriki objavljamo kratke povzetke raziskovalnih nalog, ki jih financira Področna raziskovalna skupnost za avtomatiko, računalništvo in informatiko, ki so s področja računalništva in informatike.

Naslov naloge: Sodobni koncepti upravljanja v industrijskih procesih

Projekt: Računalniška avtomatizacija industrijskih procesov

Nosilec naloge: Anton Čižman, Institut "Jožef Stefan", Ljubljana

Program raziskave:

- Primerjava in uporabnost matematičnih metod in postopkov za računalniško identifikacijo multivariabilnih dinamičnih sistemov in parametrov z referenčnim modelom.
- Identifikacija in poenostavljanje zveznih sistemov z metodami za identifikacijo in poenostavljanje diskretnih sistemov.
- Razvoj interaktivnega programskega paketa, ki bo omogočal laboratorijsko in industrijsko uporabo razvitih metod na konkretnih aplikacijah.
- Testiranje razvite programske opreme na problematiki kemijskih reaktorjev.

Naslov naloge: Verifikacija delovanja sistemov (II.faza)

Projekt: Računalniška tehnika in proizvodnja

Nosilec naloge: Janez Korenini, Institut "Jožef Stefan", Ljubljana

Program raziskave:

- Raziskava zakonitosti v računalniško vodenem procesu v zvezi z verifikacijo.
- Sinteza verifikacije v krmilnem delu pri upoštevanju celotnega sistema.
- Optimizacija glede na sistemski hardware in ekonomiko ter primer.

Naslov naloge: Informacijski sistemi in baze podatkov

Projekt: Informacijski sistemi

Nosilec naloge: Peter Tancig, Institut "Jožef Stefan", Ljubljana

Program raziskave:

- Razvoj, preizkušanje in ovrednotenje participativne metodologije analize in načrtovanja IS s posebnim poudarkom na sodelovanju vseh prizadetih, kar pogojujejo naša družbeno-politična načela.
- Izdelava okvirja dela in smernic za delo na področju projekta "Informacijski sistemi" ob široki vključenosti uporabnikov in izvajalcev.
- Preizkušanje in ovrednostenje švedske moderne metodologije (ISAC) razvoja IS.
- Izdelava formalnega modela analize in sinteze infološke strukture IS.
- Instaliranje sistema INGRES za obravnavanje relacijskih baz podatkov na računalnik PDP-11 in njegov preizkus na manjši bazi podatkov.
- Izdelava metodologije za oceno, izbor in za določitev za uporabo baz podatkov v našem okolju.
- Začetek raziskav na področju uporabe matematične logike pri delu z bazami podatkov (modeliranje sveta).
- Začetek dela na problematiki realizacije bolj naravnih spraševalnih jezikov za preiskovanje baz podatkov.

Naslov naloge: Računalniška analiza signalov v nevrofiziologiji

Projekt: Individualne naloge

Nosilec naloge: Bogdan Oblak, Institut za klinično nevrofiziologijo, Medicinska fakulteta in Klinični center v Ljubljani

Program raziskave:

- Prilagoditev celotne programske zbirke KOMB 9 za operacijski sistem RTE 4 (doslej smo uporabljali sistem DOS-III).
- Poenotenje klicnih sekvenc.
- Dopolnitev programov za draženje in za zbiranje.
- Preučiti želimo variabilnost cenik parametrov statokenizigrama tudi s pomočjo simulacije.

Naslov naloge: Modeli informacijskih sistemov v industriji

Projekt: Informacijski sistemi

Nosilec naloge: Saša Dekleva, Visoka šola za organizacijo dela, Kranj

Program raziskave:

- Predvidevamo, da bo raziskava v 3 fazah. Za vsako fazo planiramo 1 leto dela.

- 1. faza:
- analiza tipov industrijskih poslovnih sistemov,
- priprava hipoteze,
- anketiranje,
- opredelitev značilnih tipov industrijskih poslovnih sistemov,
- študij za informacijski sistem relevantnih predpisov

- 2. faza:
- opredelitev najpotrebnejših informacijskih procesov za doseganje poslovne uspešnosti pri posameznih tipih industrijskih poslovnih sistemov,
- opredelitev modelov računalniško odprtih poslovnih informacijskih sistemov za posamezne tipe industrijskih poslovnih sistemov,
- anketiranje,
- ocenitev efektov uvedbe računalniško podprtih poslovnih informacijskih sistemov.

- 3. faza:
- analiza in ocena razpoložljive programske opreme za izgradnjo gornjih informacijskih sistemov,
- opredelitev potreb po razvoju manjkajoče programske opreme v Jugoslaviji z določitvijo njenih funkcij,
- ocenitev količine dela in stroškov pri razvoju manjkajoče programske opreme v Jugoslaviji.

Naslov naloge: Podsestavi avtomatskih industrijskih Merilnih sistemov IV.

Projekt: Avtomatski industrijski merilni modularni sistemi

Nosilec naloge: Peter Šuhel, Fakulteta za elektrotehniko Univerze v Ljubljani

Program raziskave:

Modularnost in programiranje merilnega sistema:
- programska in aparaturna oprema za programiranje na magnetnem disku.

Obdelava izmerjenih parametrov:
- on line računalnik
- of line računalnik
- minimizacija izvedenih parametrov.

Sistemi merilnih robotov:
- študija inteligence sistema.

NOVICE IN ZANIMIVOSTI

DRUŽINA INTEGRIRANIH VEZIJ Z-8000

AMERIŠKO PODJETJE ZILOG JE S SVOJIMI KOOPERANTI (AMERICAN MICRO DEVICES, SGS/ATES) PRIPRAVILO PROIZVODNJO NOVE 16-BITNE PROCESORSKE DRUŽINE Z-8000 Z NASLEDNJIMI ČLANI:

Z-CPU CENTRAL PROCESSING UNIT
Z-MMU MEMORY MANAGEMENT UNIT
Z-MBU MICROPROCESSOR BUFFER UNIT
Z-FIFO BUFFER MEMORY
Z-CIO COUNTER AND PARALLEL I/O DEVICE
Z-SIO SERIAL I/O DEVICE
Z-UPC UNIVERSAL PERIPHERAL CONTROLLER
Z-BUS RAM

OGLEJMO SI PODROBNEJE POSAMEZNA INTEGRIRANA VEZJA.

O Z-8000-CPU SMO V ČASOPISU INFORMATICA ŽE POROČALI. TA MIKRO PROCESOR IMA NOVO ARHITEKTURO, UKAZE, KI SO ENAKI ALI MOČNEJŠI OD ONIH V MINI RAČUNALNIKIH, NASLOVI 8 M ZLOGOV V LOČENIH POMNILNIH SEKTORJIH ZA UKAZNI KOD, PODATKE IN SKLADE, PODPIRA PROGRAMIRANJE ZA SEDEM PODATKOVNIH TIPOV OD BITOV DO BESEDNIH NIZOV, IMA MOŽNOST RAZDELJEVANJA VIROV V MULTIMIKROPROCESORSKIH SISTEMIH TER IMA NOVO PREKINITVENO STRUKTURO. Z-CPU NUDI POSEBNO PODPORO ZA PARALELNE PROCESORSKE SISTEME, OPERACIJSKE SISTEME IN KOMPILATORJE. IMA ŠESTNAJST 16-BITNIH SPLOŠNIH REGISTROV, OSEM UPORABNIŠKIH NAČINOV NASLAVLJANJA, SEDEM GLAVNIH PODATKOVNIH TIPOV (BITI, BCD ŠTEVILKE, ZLOGI, BESEDE, DOLGE BESEDE, BESEDNI NIZI) IN 110 RAZLIČNIH UKAZNIH TIPOV. REGULARNOST REGISTRSKE ORGANIZACIJE, PODATKOVNIH TIPOV, UKAZOV IN NAČINOV NASLAVLJANJA POENOSTAVI PROGRAMIRANJE IN SKRAJŠA DOLŽINO PROGRAMOV. TAKO SE 16-BITNI REGISTRI LAHKO UPORABLJAJO KOT AKUMULATORJI IN KOT INDEKSNI REGISTRI. Z NEKAJ IZJEMAMI SE LAHKO ZLOGI, BESEDE IN 32-BITNE BESEDE UPORABIJO V VSEH UKAZIH. PODOBNO VELJA ZA UPORABO NAČINOV NASLAVLJANJA.

POMNILNIŠKE ZAHTEVE SE ZMANJŠAJO ZARADI KOMPAKTNEGA UKAZNEGA FORMATA IN ZARADI ARHITEKTONSKE REGULARNOSTI. Z-CPU JE DO DESETKRAT HITREJŠI OD 8-BITNIH MIKRO PROCESORJEV PA TUDI PETKRAT HITREJŠI OD VRSTE 16-BITNIH MIKRO PROCESORJEV IN MINI RAČUNALNIKOV. PROCESOR IMA 4 MHZ TAKTNIK IN OMOGOČA UPORABO CENENIH POMNILNIKOV. KOMPILATORJI, OPERACIJSKI SISTEMI IN DRUGA SISTEMSKA PROGRAMSKA OPREMA SE UČINKOVITO IZVAJA NA Z-CPU. KOMPILATORJI SE NPR. PODPIRAJO Z VEČKRATNIMI SKLADI, VELIKIM POMNILNIM PROSTOROM, S KONSISTENTNO UKAZNO ZALOGO TER S KOMPILATORSKO PRIREJENIMI UKAZI. ZA OPERACIJSKE SISTEME SO NA VOLJO POSEBNI UKAZI S KOMPLEKSNO STRUKTURO PREKINITEV IN PASTI, ZA SISTEMSKE IN NORMALNE NAČINE TER ŠE POSEBNI SISTEMSKI IN NORMALNI PODPROSTORI ZA SKLADE. Z SISTEMSKIMI IN NORMALNIMI NAČINI SE DOSEŽE LOČITEV VODILNIH IN UPORABNIŠKIH PROGRAMOV IN S TEM IMPLEMENTACIJA BOLJ KVALITETNEGA IN ZANESLJIVEGA OPERACIJSKEGA SISTEMA. PREKINITVENA STRUKTURA NUDI VISOKO STOPNJO AVTOMATIZACIJE, KOT SO AVTOMATIČNA REŠITEV PROGRAMSKEGA STATUSA, HITRA REŠITEV REGISTROV IN HITER ZAČETEK IZVAJANJA PREKINITVENEGA PROGRAMA. ŠTIRI PASTI PODPIRAJO DELOVANJE OPERACIJSKEGA SISTEMA.

GLEDE NA POMNILNIŠKE ZAHTEVE STA NA VOLJA DVE RAZLIČICI Z-CPU. ŠTIRIDESET- NOŽIČNA NESEGMENTIRANA RAZLIČICA NASLOVI DIREKTNO 64K ZLOGOV NASLOVNEGA PROSTORA, 48-NOŽIČNA SEGMENTIRANA RAZLIČICA PA NASLOVI NEPOSREDNO 8 M ZLOGOV NASLOVNEGA PROSTORA. NA SLIKI 1 IMAMO IMENA SIGNALOV IN NJIHOVO FUNKCIJO NA PODNOŽJU Z-CPU.

- - - - -

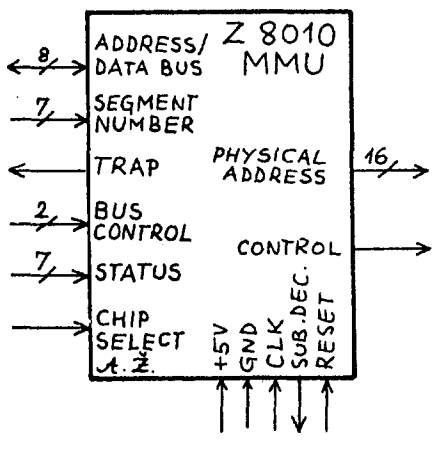
Z-MMU JE INTEGRIRANO VEZJE ZA UPRAVLJANJE POMNILNIKA. TO VEZJE IZVAJA DINAMIČNO SEGMENTNO PREMEŠČANJE, S ČEMER POSTANEJO PROGRAMSKI NASLOVI NEODVISNI OD NASLOVOV FIZIČNEGA POMNINLNIKA. VEZJE IZVAJA SEGMENTNO UPRAVLJANJE IN ZAŠČITO, PODPIRA ORGANIZACIJO SPREMENLJIVE SEGMENTNE DOLŽINE IN VEČ VEZIJ Z-MMU LAHKO PODPIRA VEČ PREVAJALNIŠKIH TABEL V POLJUBNEM DELU NASLOVNEGA PROSTORA Z-CPU. S KOMPONENTO Z-MMU SE UVAJAJO V SISTEM Z-8000 LASTNOSTI, KI JIH IMAJO LE NAJBOLJ RAZVITI RAČUNALNIŠKI SISTEMI.

Z-MMU PODPIRA DINAMIČNO SEGMENTNO PREMEŠČANJE KOT TUDI SEGMENTNO ZAŠČITO IN DRUGE SEGMENTNE UPRAVNE FUNKCIJE. SEGMENTNO PREMEŠČANJE NAREDI UPORABNIŠKE PROGRAMSKE NASLOVE NEODVISNE OD NASLOVOV FIZIČNEGA POMNILNIKA IN TAKO RAZBREMENI UPORABNIKA, KI MU NI POTREBNO NAVAJATI LOKACIJSKIH INFORMACIJ ZA FIZIČNI POMNILNIK. SEGMENTNA ZAŠČITA ONEMOGOČA ILEGALNO UPORABO LOGIČNIH SEGMENTOV IN PISANJE V ZAŠČITENE CONE NI MOGOČE. Z-MMU UPRAVLJA SEGMENTNO PREMEŠČANJE ZA UPORABNIKA S POMOČJO NASLOVNEGA PREVAJALNEGA MEHANIZMA, KI JE TRANSPARENTEN GLEDE NA UPORABNIŠKE PROGRAME. TA MEHANIZEM PREVEDE 23-BITNE LOGIČNE NASLOVE V PROGRAMIH V 24-BITNE NASLOVE FIZIČNEGA POMNILNIKA.

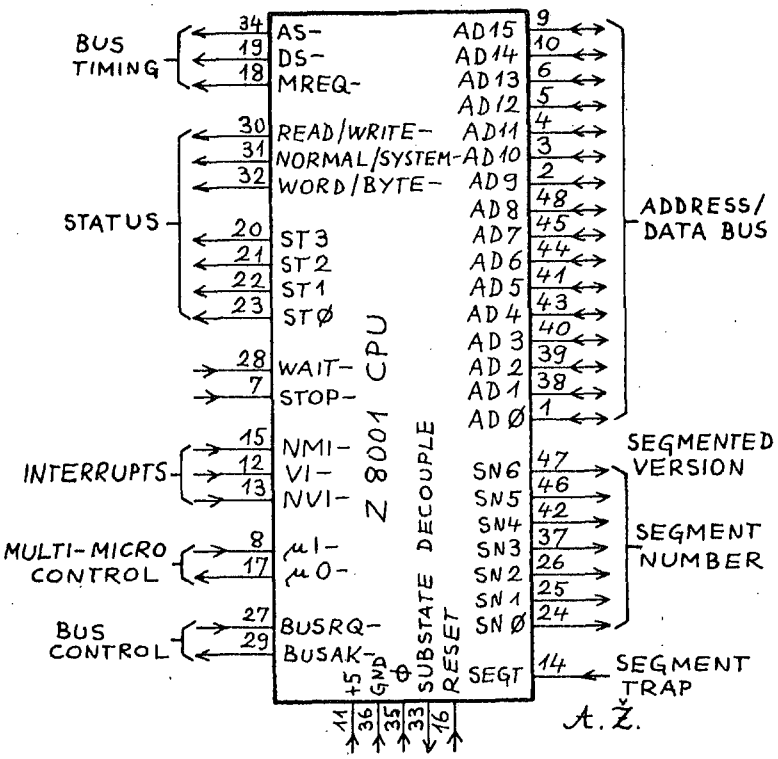
VSAK 23-BITNI LOGIČNI NASLOV JE SESTAVLJEN IZ 7-BITNE ŠTEVILKE SEGMENTA IN IZ 16-BITNEGA ODMIKA. S POMOČJO PREVAJALNE TABELE V Z-MMU SE 7-BITNI ŠTEVILKI LOGIČNEGA SEGMENTA PRIDRUŽI BAZNI NASLOV SEGMENTA FIZIČNEGA POMNILNIKA. NATO SE 16-BITNI ODMIK PRIŠTEJE K BAZNEMU NASLOVU IN SE DOBI 24-BITNI FIZIČNI NASLOV. SISTEM DINAMIČNO PRENAŠA PREVAJALNE TABELE V ODVISNOSTI OD NALOG, KI NASTAJAJO, SE UKINJAJO IN MENJAJO.

VSAKEMU SEGMENTU JE PRIREJENO DOLOČENO ŠTEVILO PRIDEVKOV, KO SE NA ZAČETKU SEGMENT VSTAVI V Z-MMU. KO SE POJAVI POMNINLNIŠKA NAVEDBA, PREIZKUSI PREKINITVENI MEHANIZEM PRIDEVKE GLEDE NA INFORMACIJSKI STATUS IZ Z-CPU. PRI NEUJEMANJU PRIDEVKOV SE GENERIRA PAST, KI PREKINE DELOVANJE Z-CPU. POTEM LAHKO Z-CPU PREIZKUSI STATUSNE REGISTRE Z-MMU IN UGOTOVI VZROKE. SEGMENTNI PRIDEVKI SO MED DRUGIMI TUDI OBSEG SEGMENTA IN TIP SEGMENTA (NPR. ČITALNI, SISTEMSKI, IZVRŠILNI, Z NEVELJAVNIMI NAVEDBAMI ITN.). DRUGE SEGMENTNE UPRAVNE FUNKCIJE SO NPR. OPOZORILNI SIGNAL ZA PISALNO CONO PRI OPERACIJAH S SKLADI, STATUSNI REGISTER Z ZGODOVINSKO INFORMACIJO O SPREMEMBI IN NAVEDBI SEGMENTA.

VSAKO VEZJE Z-MMU LAHKO SHRANI 64 SEGMENTNIH VSTOPOV, VSAK VSTOP PA JE SESTAVLJEN IZ SEGMENTNEGA BAZNEGA NASLOVA, NJEGOVIH



SLIKA 2. Z8000-MMU JE VEZJE ZA UPRAVLJANJE POMNILNIKA, IN SICER: DINAMIČNO SEGMENTNO PREMEŠČANJE, SEGMENTNA ZAŠČITA ITN.



SLIKA 1. CENTRALNA PROCESNA ENOTA Z8000-CPU, KI IMA OZNAKO Z8001. TO JE T.I. SEGMENTIRANI TIP PROCESORJA, KI NASLOVI 48M ZLOGOV. NESEGMENTIRANA RAZLICICA IMA OZNAKO Z8002 IN LAHKO NASLOVI LE 64K ZLOGOV. VEZJI IMATA 48 IN 40 NOŽIC. IZ SLIKE JE JASNO RAZVIDEN SEGMENTIRANI DODATEK: TO SO SPONKE SNØ DO SN6 IN SPONKA SEGT. PROCESOR Z8000 PREDSTAVLJA S SVOJO DRUŽINO VEZIJ TRENUTNO NAJMOČNEJSI 16-BITNI MIKRORAČUNALNIŠKI SISTEM.

PRIDEVKOV, IZ OBSEGA IN STATUSA. SEGMENTNI OBSEG JE SPREMENLJIV OD 256 ZLOGOV DO 64K ZLOGOV V INKREMENTIH PO 256 ZLOGOV. V OKVIRU NASOVNEGA PROSTORA SE LAHKO UPORABI TUDI VEČJE ŠTEVILO VEZIJ Z-MMU Z VEČ PREVAJALNIMI TABELAMI. NA SLIKI 2 JE PRIKAZANO PODNOŽJE VEZJA Z-MMU Z IMENI SIGNALOV.

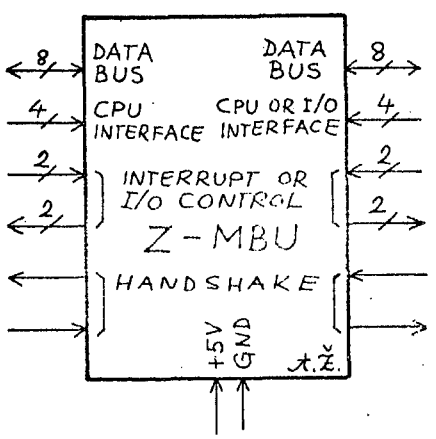
- - - - -

Z-MBU JE MIKROPROCESORSKA VMESNA ENOTA, KI IMA ASINHRONI, DVOSMERNI FIFO, Z ORGANIZACIJO 256 KRAT 8 BITOV. TA ENOTA JE RAZŠIRLJIVA NA 16-BITNE BESEDE IN JO JE MOČ KASKADIRATI DO POLJUBNE GLOBINE. Z-MBU IZOLIRA CENTRALNO PROCESNO ENOTO V PARALELNI PROCESORSKI KONFIGURACIJI IN SE LAHKO UPORABI V RAZLIČNIH MIKROPROCESORSKIH SISTEMIH. VEZJE IMA LOGIKO ZA RAZPOZNAVANJE VZORCEV IN LAHKO USTAVI PRENOS PODATKOV TER SPROŽI PREKINITVE.

Z-MBU PREDSTAVLJA SPLOŠNO UPORABEN ELASTIČEN VMESNIK MED ASINHRONO DELUJOČIMI CENTRALNIMI PROCESNIMI ENOTAMI V PARALELNI PROCESORSKI MREŽI ALI PA TUDI VMESNIK MED CPU IN PERIFERNIMI VEZJI (KRMILNIKI ZA DISK). Z-MBU POVEZE Z-VODILO Z DRUGIM MIKRO PROCESORJEM ALI PERIFERNIM VEZJEM. TA ENOTA PREDSTAVLJA KLJUČNI ELEMENT ZA PARALELNO DELOVANJE PROCESORJEV V SISTEMIH Z-8000, SAJ POVEZUJE KOMPONENTE IN PODSISTEME, KI OBRATUJEJO Z RAZLIČNIMI HITROSTMI. TA DVOSMERNA NAPRAVA SPREJEMA PODATKE IN JIH HRANI DOTLEJ, KO JIH DRUGA NAPRAVA V SISTEMU LAHKO SPREJME. NA TA NAČIN SE NE ZMANJŠA HITROST DELOVANJA SISTEMA, KO SE SPREJEMNA NAPRAVA PRIPRAVLJA ZA SPREJEM PODATKOV.

Z-MBU ZNATNO POVEČUJE PRETOČNOST SISTEMA, KO OPRAVLJA PRENOSE SPREMENLJIVIH BLOKOV PODATKOV V REŽIMU DIREKTNEGA POMNILNIŠKEGA DOSTOPA (DMA) ALI V REŽIMU PREKINITVE, KO SE POVEZUJEJO HITRO DELUJOČA PERIFERNA VEZJA. Z-MBU DAJE TUDI POPOLNO STATUSNO INFORMACIJO PRI UPORABI V REŽIMIH ODTIPAVANJA PROCESNE OKOLICE. NA SLIKI 3 VIDIMO PODNOŽJE VEZJA Z-MBU Z IMENI SIGNALOV.

- - - - -

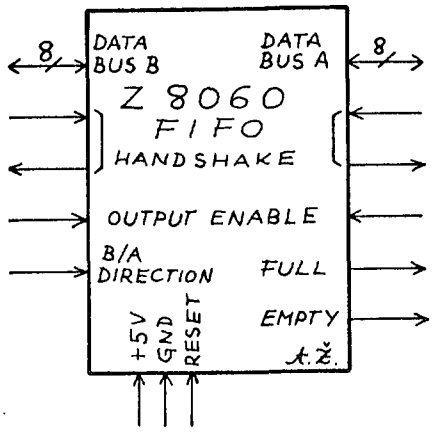


SLIKA 3. Z-MBU: ASINHRONI, DVOSMERNI FIFO ZA MULTIPROCESORSKO KONFIGURACIJO.

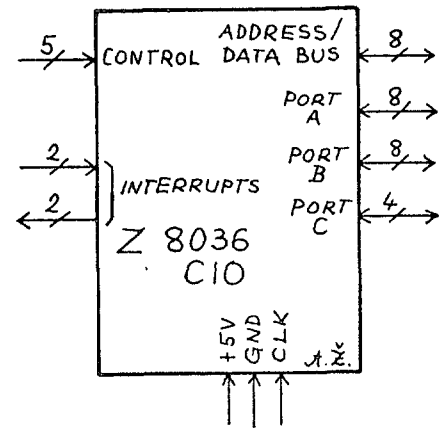
Z-FIFO JE KLASIČNI POMNILNIK TIPA "VRSTA" (FIFO), KI RAZŠIRJA ENOTO Z-MBU DO POLJUBNE GLOBINE, IMA ORGANIZACIJO 256 KRAT 8 BITOV, INDIKACIJSKI NOŽICI ZA STANJI "VRSTA JE POLNA" IN "VRSTA JE PRAZNA" IN IZHODE PREKO OJAČEVALNIKOV S TREMI STANJI. SEVEDA SE Z-FIFO LAHKO UPORABI TUDI KOT SPLOŠNI POMNILNIK FIFO KJERKOLI. S KASKADIRANJEM IN V POVEZAVI Z Z-MBU SE DOSEŽEJO ŠTEVILA LOKACIJ Z INKREMENTI PO 256 CELIC. ČEPRAV Z-FIFO NI POVEZLJIV Z Z-VODILOM PA IMA MOŽNOSTI ROKOVANJA Z VSEMI KOMPONENTAMI, PRIKLJUČENIMI NA Z-VODILO. UPORABLJA SE LAHKO KOT VMESNIK MED V/I VRATI (PORTI) VEZIJ Z-UPC, Z-CIO ALI Z-MBU IN UPORABNIŠKIMI NAPRAVAMI.

Z-FIFO JE KASKADNO RAZŠIRLJIV, ŠIRIMO PA GA LAGKO TUDI DO POLJUBNE DOLŽINE BESEDE. NJEGOVI IZHODNI SPONKI ZA INDIKACIJO STANJ "POLN" IN "PRAZEN" OMOGOČATA UGOTAVLJANJE POLNOSTI IN PRAZNOSTI SISTEMA KOMPONENT Z-FIFO. PRI TEM IMA Z-FIFO MOŽNOST DVOSMERNEGA PRENOSA PODATKOV S KRMILJENJEM NA SPONKI B/A, KOT KAŽE SLIKA 4.

- - - - -



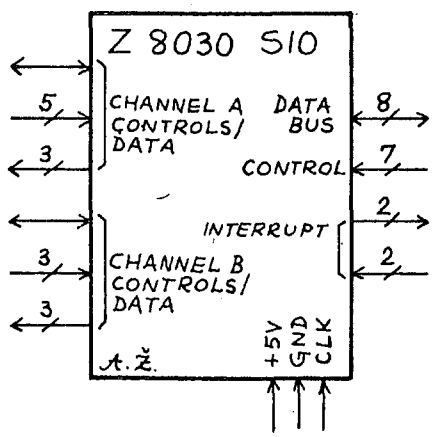
SLIKA 4. Z8060-FIFO JE KLASIČEN FIFO S STANJI "POLN" IN "PRAZEN".



SLIKA 6. Z8036-CIO IMA PARALELNA V/I VRATA, ŠTEVNIKE IN ČASOVNIKE.

Z-SIO JE VEZJE ZA SERIJSKI V/I Z DVEMA NEODVISNIMA POLNODUPLEKSNIMA KANALOMA S KRMILJENJEM MODEMA. HITROST SERIJSKIH KANALOV JE OD Ø DO 800K BITOV NA SEKUNDO, ZA ASINHRONE NAČINE PRENOSA JE MOGOČE PROGRAMIRATI PET DO OSEM PODATKOVNIH BITOV Z ENIM, ENIM IN POL ALI DVEMA USTAVITVENIMA BITOMA. MOGOČE JE PROGRAMIRATI TUDI DELILNI TAKTNI FAKTOR (ZA HITROST SERIJSKIH KANALOV), DETEKTIRATI PREKINITEV IN JO GENERIRATI. NADALJNE LASTNOSTI OZIROMA MOŽNOSTI VEZJA SO ŠE: DETEKCIJA PARNOSTI, PRESTOPA IN NAPAKE, BISYNC NAČIN Z NOTRANJO ALI ZUNANJO SINHRONIZACIJO ZNAKOV, EDEN ALI DVA SYNC ZNAKA, CRC GENERIRANJE IN PREIZKUŠANJE. VEZJE GENERIRA IN SPREJEMA FORMATA SDLC IN HDLC (AVTOMATIČNO VSTAVLJANJE NIČEL IN NJIHOVO BRISANJE ITN.).

NAŠTETE FUNKCIJE TEGA VEZJA IZPOLNJUJEJO VRSTO ZAHTEV, POVEZANIH S SERIJSKIM PRENOSOM PODATKOV. VEZJE OPRAVLJA SERIJSKOPARALELNO IN OBRATNO PRETVORBO Z VRSTO PROGRAMIRLJIVIH FUNKCIJ, KI SO POTREBNE PRI PODATKOVNEM KOMUNICIRANJU. Z-SIO OBDELA ASINHRONE FORMATE, SINHRONO IN ZLOGOVNO SESTAVLJENE PROTOKOLE, KOT SO IBM BISYNC IN SINHRONE PROTOKOLE, KOT STA HDLC IN IBM SDLC. SEVEDA PA TO VEZJE LAHKO PODPIRA KATERIKOLI DRUGI PROTOKOL, NPR. V PRIMERU UPORABE KASETNE NAPRAVE, GIBKEGA DISKA ITN. Z-SIO GENERIRA IN PREISKUŠA CRC-KODE V VSAKEM SINHRONEM NAČINU IN GA JE MOČ PROGRAMIRATI ZA PREIZKUŠANJE PODATKOV PRI RAZLIČNIH NAČINIH. KADAR KRMILJENJA MODEMOV NE



SLIKA 5. Z8030-SIO JE DVOKANALNI V/I ZA ZNANE PODATKOVNE FORMATE.

POTREBUJEMO, SE LAHKO USTREZNI VHODI IN IZHODI UPORABIJO ZA DRUGE V/I FUNKCIJE. PODNOŽJE VEZJA Z-SIO JE PRIKAZANO NA SLIKI 5 Z IMENI SIGNALOV. TO VEZJE MOČNO SPOMINJA NA Z-80-SIO, KI SE UPORABLJA V SISTEMIH Z Z-80 ZA ENAKE NAMENE.

- - - - -

Z-CIO JE KOMBINIRANO INTEGRIRANO VEZJE S ŠTEVNIKI, ČASOVNIKI IN PARALELNIMI V/I VRATI. IMA DVOJE NEODVISNIH 8-BITNIH OJAČEVALNIH DVOSMERNIH V/I VRAT IN ŠE ENA 4-BITNA V/I VRATA, ŠTIRI NAČINE ROKOVANJA VKLJUČNO Z IEEE-488, ČAKALNOPRENOSNI VMESNIK ZA HITRI PRENOS PODATKOV, TRI NEODVISNE 16-BITNE ŠTEVNIKE, LOGIKO ZA RAZPOZNAVANJE VZORCEV IN PREDNOSTNI KRMILNIK PREKINITEV.

TO UNIVERZALNO PERIFERNO VEZJE LAHKO ZADOSTI VRSTI ZAHTEV ZA PARALELNI PRENOS PODATKOV IN POTREB, POVEZANIH S ŠTETJEM IN MERJENJEM ČASA. TO VEZJE ZDRUŽUJE TAKO FUNKCIJE DVEH VEZIJ, IN SICER Z-80-PIO IN Z-80-CTC. BREZ ZUNANJIH POVEZAV JE MOGOČE OBLIKOVATI TUDI 32-BITNI ŠTEVNIK/ČASOVNIK. NA SLIKI 6 JE PRIKAZANO PODNOŽJE TEGA VEZJA.

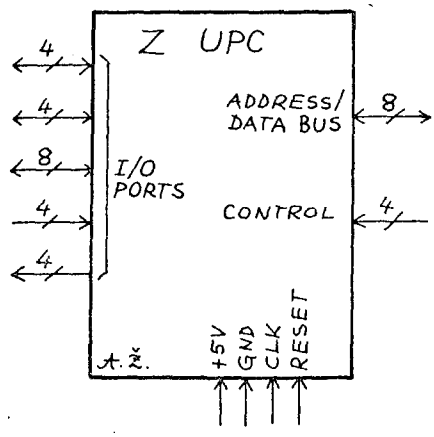
- - - - -

Z-UPC JE UNIVERZALNI PERIFERNI KRMILNIK, KI IMA STRUKTURO IN UKAZE MIKRO RAČUNALNIKA V ENEM VEZJU Z OZNAKO Z-8. TO VEZJE VSEBUJE 2K-ZLOŽNI ROM, 144-ZLOŽNO REGISTRSKO ZBIRKO, UART ZA POLNI DUPLEKS, DVA ŠTEVNIKA/ČASOVNIKA TER 24 BITOV ZA V/I Z ROKOVANJEM ALI BREZ ROKOVANJA.

Z-UPC JE RAZLIČICA MIKRO RAČUNALNIKA V ENEM VEZJU Z-8 ZA UPORABO NA Z-VODILU. TOREJ JE NEKE VRSTE INTELIGENČNI V/I ZA POVEZAVO Z UPORABNIŠKO PERIFERIJO. IMA 47 UKAZNIH TIPOV IN 9 NAČINOV NASLAVLJANJA. PREKINITVENI MEHANIZEM VSEBUJE ŠEST VEKTORIRANIH PREKINITEV Z MASKAMI IN PREDNOSTJO. 144-REGISTRSKA ZBIRKA VSEBUJE 124 SPLOŠNIH REGISTROV, 4 V/I REGISTRE VRAT IN 16 KRMILNIH IN STATUSNIH REGISTROV. TUDI UART V VEZJU IMA VSE STANDARDNE LASTNOSTI IN GA LAHKO POVEZEMO S ČASOVNIKI VEZJA Z-UPC. ŠTEVNIŠKI/ČASOVNIŠKI KANALI IMAJO PO 14 BITOV (6 PLUS 8). VEZJE JE SHEMATIČNO PRIKAZANO NA SLIKI 7.

- - - - -

Z-BUS RAMI SO 4K KRAT 8-BITNI PSEVDO STATIČNI POMNILNIKI TIPA RAM, 2K KRAT 8-BITNI TAKTNI STATIČNI RAMI BREZ DODATNIH TTL VEZIJ ZA NJIHOVO UPORABO NA Z-VODILU. TI POMNILNIKI IMAJO NEKATERE PREDNOSTI NA Z-VODILU IN SO NAMENJENI KOT DODATEK PORAZDELJENIM INTELIGENČNIM SISTEMOM V OBLIKI LOKALNIH POMNILNIKOV. GRE ZA BISTVU ZA DINAMIČNE



SLIKA 7. Z-UPC JE UNIVERZALNI PERIFERNI KRMILNIK S PROCESORJEM Z8.

POMNILNIKE BREZ ZUNANJEGA DINAMIČNEGA OSVEŽEVANJA IN CELOTNA KOLIČINA CELIC SE NAHAJA V ENEM VEZJU. 2K KRAT 8-BITNI POMNILNIK PA JE STATIČEN, HITER (150 NS) IN Z MAJHNO PORABO ENERGIJE. ŽE 3V ZADOSTUJEJO ZA OHRANITEV INFORMACIJE V STANJU MIROVANJA, KO JE TOK 10 MA.

- - - - -

PODJETJE ZILOG RAZVIJA ŠE DRUGE Z-8000 KOMPONENTE, OD KATERIH SO PRAV GOTOVO ZANIMIVI DMA VEZJE, KRMILNIK ZA DISK (RIGID DISK), CRT KRMILNIK IN PROCESOR ZA PLAVAJOČO VEJICO. NAŠTETE KOMPONENTE BO PROIZVAJALO TUDI ITALIJANSKO PODJETJE SGS-ATES, ELETTRONICI SPA, VIA C. OLIVETTI 2, 20041 AGRATE BRIANZA, ITALIJA. DISTRIBUTOR TEH ELEMENTOV PA BO TUDI PODJETJE RAPIDO, VIA G. CORSI 4, TRST.

A.P.ŽELEZNIKAR

MEHURČNI POMNILNIK NA TISKANI PLOŠČI

TEXAS INSTRUMENTS UVAJA NA TRŽIŠČE TISKANE PLOŠČE Z MEHURČNIMI POMNILNIKI. V ZVEZI S TEM PROIZVAJA TI INTEGRIRANA VEZJA, TJ. VMESNIKE ZA POGON 256K-ZLOŽNIH MEHURČNIH POMNILNIH ENOT; TI POMNILNIKI NOSIJO OZNAKO TIB0303. PERIFERNA VEZJA ZA MEHURČNI POMNILNIK PA SO:

SN 75385 VMESNIK ZA NAVITJA
SN 75384 FUNKCIJSKI VMESNIK
SN 75282 REGISTRIRNI (VHODNI) OJAČEVALNIK

V PRIPRAVI PA JE TUDI UNIVERZALNO VMESNO VEZJE TMS9922, KI ZDRUŽUJE VSE FUNKCIJE ZA UPORABO MEHURČNIH POMNILNIKOV. TO VEZJE JE KRMILNIK ZA 256K-ZLOŽNE MEHURČNE POMNILNE ENOTE. "TI" BO V NAJKRAJŠEM ČASU PROIZVAJAL TUDI MEHURČNE POMNILNE ENOTE Z MILIJON ZLOGI.

A.P.ŽELEZNIKAR

16-BITNI PROCESOR MC 68000

DOLGOPRIČAKOVANI 16-BITNI MIKRO PROCESOR MOTOROLA MC 68000 JE KONČAL DRUGI SILICIJEV PREHOD IN TA DRUGA ITERACIJA SO ŠE VEDNO VZORČNI PRIMERKI IZ SERIJSKE PROIZVODNJE, KI SE POŠILJAJO UPORABNIKOM NA OSNOVI POSEBNEGA SPORAZUMA. VZORCI NOSIJO OZNAKO XC 68000 IN DODANA JIM JE LISTA NAPAK V INTEGRIRANEM VEZJU. REDNA PROIZVODNJA BO STEKLA ŠE LETOS, VENDAR BO MOTOROLA V EVROPI PRODAJALA LETOS LE SISTEME S TEM MIKRO PROCESORJEM. POSAMEZNA INTEGRIRANA VEZJA BODO NAJBRŽ V PRODAJI NASLEDNJE LETO (1980).

A.P.ŽELEZNIKAR

RAZVOJNI CENTER PODJETJA INTEL NA JAPONSKEM

PODJETJE INTEL SE JE ODLOČILO DA ODPRE MARCA NASLEDNJE LETO (1980) BLIZU TOKIJA RAZVOJNI CENTER ZA INTEGRIRANA VEZJA Z VISOKO GOSTOTO. TO BO PETI RAZVOJNI CENTER PODJETJA INTEL IN DRUGI IZVEN ZDA; PODOBNI RAZVOJNI CENTER ZE DELUJE V IZRAELU. CENTER BO RAZVIJAL LE ZELO KOMPLEKSNA INTEGRIRANA VEZJA ZA CELOTNO SVETOVNO TRŽIŠČE IN NE SAMO ZA JAPONSKO. DIREKTOR TEGA CANTRA BO M. SHIMA, KI JE RAZVIJAL 16-BITNI PROCESOR Z-8000, PRED TEM PA JE SODELOVAL V RAZVOJU PROCESORJEV 4004 IN 8080.

A.P.ŽELEZNIKAR

SISTEM /38: PRVA ZAKASNITEV PODJETJA IBM

ZAKASNITEV PODJETJA IBM PRI DOBAVI SISTEMOV /38 ZARADI PROBLEMOV S PROGRAMSKO OPREMO POMENI, DA NASTOPAJO NOVI ELEMENTI TUDI V TRŽNEM SISTEMU IZDELEK/RAZVOJ. SISTEM /38 JE MALI POSLOVNI SISTEM IN ZAKASNITEV NJEGOVE DOBAVE POMENI NEKAJ NOVEGA, KER SE JE TO ZGODILO TUDI PRI PODJETJU IBM. RAZVOJ SISTEMSKE PROGRAMSKE OPREME ZA OPERACIJE, KOMUNIKACIJE IN PODATKOVNO UPRAVLJANJE PREDSTAVLJA VEČJI IN TEŽJI DEL V PRIMERJAVI Z MATERIALNO OPREMO SISTEMA. RAZVOJ TAKE OPREME POSTAJA VSE BOLJ KOMPLEKSEN IN NAPOREN IN KOT POSLEDICA TEGA SE SPREMINJA TUDI TRŽNA STRATEGIJA V POVEZAVI Z ELEMENTOM IZDELEK/RAZVOJ. IBM-OVO OPRAVIČILO ZA ZAKASNITEV NAVAJA POSEBEN ČAS, KI JE POTREBEN ZA INTEGRACIJO IN PREIZKUS SISTEMSKIH PROGRAMSKIH ELEMENTOV, DA BI SE DOSEGLA NAČRTOVANA ZMOGLJIVOST SISTEMA.

NAJVEČJE TEŽAVE PRI RAZVOJU SISTEMA /38 SE POJAVLJAJO PRI NOVEM KRMILNEM PROGRAMU SISTEMA, V MEHANIZMIH INTERAKTIVNIH PODATKOVNIH BAZ TER PRI KOMPILATORJU ZA JEZIK RPG III. PRI TEM SISTEMU GRE NAMREČ ZA POPOLNOMA NOVO ARHITEKTURO IN OPERACIJSKI SISTEM, KI POMENI KONČNI ODSKOK OD SISTEMA /360, KI GA JE IBM VPELJAL LETA 1964. TA SISTEM OBLJUBLJA ENORAVNINSKO SHEMO POMNILNIŠKEGA NASLAVLJANJA Z OBJEKTNO USMERJENIMI POMNILNIŠKIMI NAVEDBAMI, DO 280 TRILIJONOV ZLOGOV VIRTUALNEGA POMNILNIKA IN EN SAM KRMILNI JEZIK ZA OPERATORJE, PROGRAMERJE IN UPORABNIKE TERMINALOV. POMNILNIŠKE AMBICIJE SO TU ZARES VELIKE IN NIHČE DOSLEJ NI POSKUSIL PODOBNEGA NA PODROČJU ARHITEKTURE IN OPERACIJSKIH SISTEMOV.

DANAŠNJE RAČUNALNIŠKO TRŽIŠČE SEVEDA NE DOVOLJUJE ZAMRZNITVE PROGRAMSKE OPREME, NJENE DOKONČNE DEFINICIJE IN ASORTIMANA. PRI PODJETJU IBM JE NAROČENIH 35.000 SISTEMOV /38 IN PODPORA TAKEMU ŠTEVILU SISTEMOV V UPORABI BO ZAHTEVALA NOVE DIMENZIJE PRI VZDRŽEVANJU, DOPOLNJEVANJU IN POVEZOVANJU SISTEMOV.

A.P.ŽELEZNIKAR

NAPOVEDI RAZVOJA INTELIGENČNE ELEKTRONIKE

POROČILO A.D.LITTLE NAPOVEDUJE, DA BO TRŽIŠČE INTELIGENČNIH ELEKTRONSKIH NAPRAV ŽE V LETU 1987 DOSEGLO ZNESEK 40 MILIJARD DOLARJEV. ELEKTRONSKA INTELIGENCA BO SPREMENILA NALIČJE

POSLOVANJA, VPRAŠANJE PA JE V KAKŠNI MERI IN NA KATERIH PODROČJIH. V/I PROBLEMI BODO ODLOČILNI ZA PRIHODNOST VRSTE TRŽIŠČ. ŽE SEDAJ PREDSTAVLJAJO SENZORJI, AKTUATORJI IN PERIFERNE NAPRAVE BISTVENI IZDATKOVNI DELEŽ V INTELIGENČNIH ELEKTRONSKIH IZDELKIH IN V PRIHODNOSTI BO TA DELEŽ ŠE NARASTEL.

TRŽIŠČE INTELIGENČNE ELEKTRONIKE JE MOGOČE RAZDELITI NA ŠTIRI DELE:

AVTOMOTIVNO
POSLOVNO/KOMUNIKACIJSKO
POTROŠNIŠKO
INDUSTRIJSKO

VSAKO OD TEH PODROČIJ JE RAZDELJENO NA PODPODROČJA S SVOJIMI TRŽIŠČI, ŽIVLJENSKIMI CIKLI IN RAZLIKAMI MED AMERIŠKIM IN EVROPSKIM TRŽIŠČEM.

AVTOMOTIVNO PODROČJE

AVTOMOTIVNO TRŽIŠČE BO V ZAPADNI EVROPI IN V ZDA DOSEGLO PRODUKT 7 MILIJARD DOLARJEV V LETU 1987. TO STANJE BODO PODPIRALI EKONOMIJA V PORABI GORIVA IN PREDPISI O ONESNAŽEVANJU. VEČINA TEH PREDPISOV BO ZAČELA VELJATI ŽE PRIHODNJE LETO (1980). EVROPSKO TRŽIŠČE BO DOSEGLO POLOVIČNI PRODUKT AMERIŠKEGA NA TEM PODROČJU V LETU 1987. FAKTOR RASTI V NASLEDNJIH DEVETIH LETIH NAJ BI ZNAŠAL NA TEM PODROČJU OD 20 DO 40.

PODROČJE POSLOVANJA IN KOMUNIKACIJ

V LETU 1987 NAJ BI PRODUKT NA TEM PODROČJU DOSEGEL ŽE 13 MILIJARD DOLARJEV. TO PODROČJE NAJ BI POVEČALO PREDVSEM UČINKOVITOST MENAŽERSKEGA DELA IN SKRAJŠALO ČAS ZA UPRAVNE ODLOČITVE, SKRAJŠALO NAJ BI ČAS PRENOSA INFORMACIJ OD ZBIRALNIKOV DO UPORABNIKOV IN POVEČALO IZREDNO AKTUALNO PRODUKTIVNOST V ZDA. UVEDBA TAKO IMENOVANEGA INTEGRIRANEGA URADA NAJ BI SE ZAČELA V LETU 1985, KO BODO NA VOLJO MULTIMEDIJSKA KOMUNIKACIJSKA SREDSTVA, SISTEMI ZA PROCESIRANJE TEKSTOV, FAKSIMILNI IN KOPIRNI SISTEMI, ELEKTRONSKI TELEFONI, MEDSEBOJNA MENJAVA POSLOV, ZASEBNI KOMUNIKACIJSKI SISTEMI, MIKROGRAFIČNE NAPRAVE IN KALKULATORJI. PERSPEKTIVA TEGA PODROČJA JE IZREDNO ŠIROKA IN ZAHTEVNA IN PORAST PRODUKTA DO LETA 1987 BO DOSEGEL 250 PROCENTOV.

POTROŠNIŠKO PODROČJE

V LETU 1987 NAJ BI BILO PRODANIH VEČ KOT 400 MILIJONOV INTELIGENČNIH ELEKTRONSKIH MODULOV NA PODROČJE ŠIROKE PORABE. PRI POVPREČNI CENI $ 50 ZA MODUL BO DOSEŽEN PRODUKT 20 MILIJARD OZIROMA POVEČANJE ZA 1000 PROCENTOV. DOLOČENA POTROŠNIŠKA TRŽIŠČA SE BODO HITRO POJAVLJALA IN IZGINJALA. VENDAR BO TAKO DINAMIČNO TRŽIŠČE ŠE VEDNO PRIMERNO ZA MAJHNA PODJETJA, KI BODO REALIZIRALA NOVE ZAMISLI IN NE BO MOGLO BITI KONTROLIRANO S STRANI VELIKIH PODJETIJ. PRI TEM VELJA OMENITI, DA BO IZREDNO NARASLA POTROŠNJA MIKRO PROCESORJEV, PRAV TAKO ZA 1000 PROCENTOV DO LETA 1987.

INDUSTRIJSKO PODROČJE

INDUSTRIJSKI IZDELKI, KI BODO UPORABLJALI INTELIGENČNO ELEKTRONIKO BODO NA TRŽIŠČU V LETU 1987 DOSEGLI PRODUKT 10 MILIJARD DOLARJEV. INDUSTRIJSKO PODROČJE JE MOČ RAZDELITI V KRMILJENJE PROCESOV, AVTOMATIZIRANO PROIZVODNJO, ANALITIČNE INSTRUMENTE, AVTOMATIČNE PREIZKUŠEVALNE NAPRAVE IN V KONSTRUKCIJSKE AVTOMATIČNE SISTEME. OD TEH SEKTORJEV IMA NAJVEČ MOŽNOSTI KRMILJENJE PROCESOV. INTELIGENCA STROJA BO POMENILA ZA UPORABNIKA VEČ KOT ZNIŽANJE PROIZVODNIH STROŠKOV IN NAPOČIL BO ČAS KRMILJENIH OBDELOVALNIH STROJEV, ROBOTOV IN DRUGIH NAPRAV ZA OBDELAVO MATERIALA. KER JE AVTOMATIZACIJA KAPITALNO INTENZIVNA, BO ODLOČILNO ŠTEVILO FUNKCIJ NA ENOTO CENE. NARASEL BO OBSEG RAČUNALNIŠKEGA NAČRTOVANJA IZDELKOV IN DIAGNOSTIKA V PROIZVODNJI.

ŠTUDIJO, KI VSE TO NAPOVEDUJE, SO IZDELOVALI DVE LETI, CENA ZANJO PA JE ZNAŠALA 1,5 MILIJONA DOLARJEV. NASLOV ŠTUDIJE JE : STRATEGIC IMPACT OF INTELLIGENT ELECTRONICS IN THE UNITED STATES AND EUROPE - 1977 TO 1987.

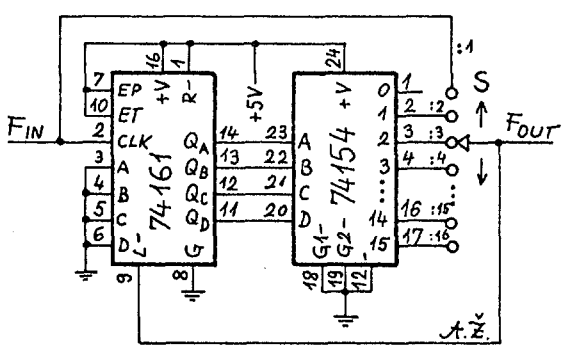
A.P.ŽELEZNIKAR

DELILNIK ZA TAKT

V MIKRORAČUNALNIŠKIH VEZJIH VEČKRAT POTREBUJEMO DELILNIKE TAKTA (URE), NPR. PRI NASTAVLJANJU HITROSTI SERIJSKEGA KANALA. ČEPRAV OBSTAJAJO POSEBNA DELILNA VEZJA (NPR. MC 14411) PA SI LAHKO VEČKRAT POMAGAMO TUDI Z USTREZNO POVEZAVO TTL ALI CMOS VEZIJ. SLIKA 1 PRIKAZUJE DELILNIK ZA RAZMERJA 1, 2, ... ,16.

VEZJE 74161 JE SINHRONI BINARNI ŠTEVNIK, KI IMA MOŽNOST PARALELNE NALOŽITVE (VHODI A, B, C, D S SIGNALOM L-). VSEBINO TEGA ŠTEVNIKA KORAKOMA POVEČUJEMO S SIGNALOM FIN (NA VHODU CLK). IZHODI QA, QB, QC IN QD SO POVEZANI Z VHODI A, B, C, D KODIRNIKA 4-V-16, TJ. Z VEZJEM 74154, KOT KAŽE SLIKA 1.

VEZJE 74154 JE UPORABLJENO KOT HEKSADECIMALNI DEKODIRNIK IN NA IZHODIH TEGA VEZJA SE POJAVLJAJO NIČLE NA VSAKIH N VHODNIH IMPULZOV (N = 2, 3, ... ,16). S PRETIKALOM S IZBEREMO DOLOČENO DELILNO RAZMERJE, TO JE N IN OČITNO VELJA

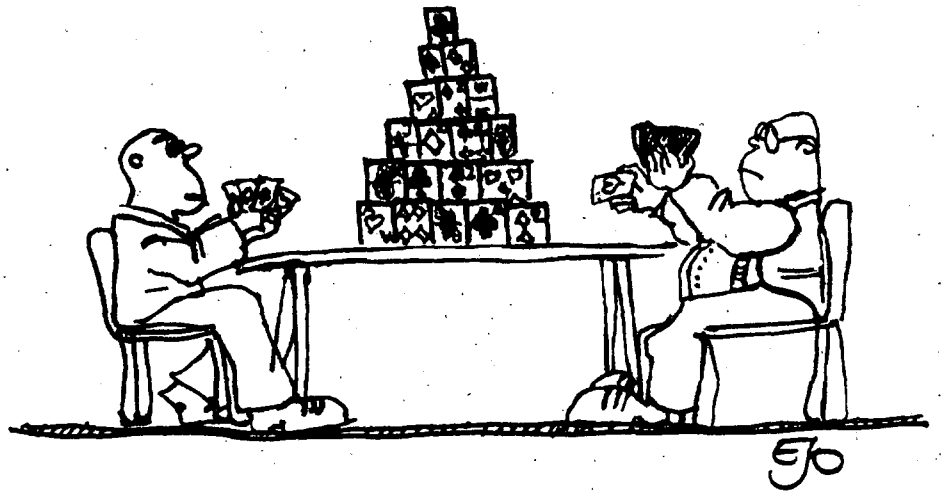


SLIKA 1. BINARNI ŠTEVNIK 74161 IN KODIRNIK "ŠTIRI-V-ŠESTNAJST" 74154 SESTAVLJATA DELILNIK S SPREMENLJIVIM MODULOM. DELILNO RAZMERJE OD 1 DO 16 JE NASTAVLJIVO S PRETIKALOM S.

FOUT = FIN/N

NAMESTO KODIRNIKA 74154 LAHKO UPORABIMO TUDI KODIRNIK 4-V-10 Z OZNAKO 7442 IN DOBIMO RAZMERJA 1, 2, ..., 10.

A.P.ŽELEZNIKAR



"BUSINESS GAMES IN THE SYSTEM FOR MANAGEMENT OF CONSTRUCTION."

RAMIĆ ESAD
78350 BOS. NOVI
ul. M. TITA 6
81952 (079)

SLOVENIJALES-RO "LESNOSPER"

ORGANIZATOR EOP

CENIK OGLASOV

Ovitek - notranja stran (za letnik 1979)

2 stran	20.000 din
3 stran	15.000 din

Vmesne strani (za letnik 1979)

1/1 stran	9.600 din
1/2 strani	6.000 din

Vmesne strani za posamezno številko

1/1 stran	3.600 din
1/2 strani	2.400 din

Oglasi o potrebah po kadrih (za posamezno številko) 1.200 din

Razen oglasov v klasični obliki so zaželjene tudi krajše poslovne, strokovne in propagandne informacije in članki. Cena objave tovrstnega materiala se bo določala sporazumno.

ADVERTIZING RATES

Cover page (for all issues of 1979)

2nd page	1100 $
3rd page	880 $

Inside pages (for all issues of 1979)

1/1 page	660 $
1/2 page	440 $

Inside pages (individual issues)

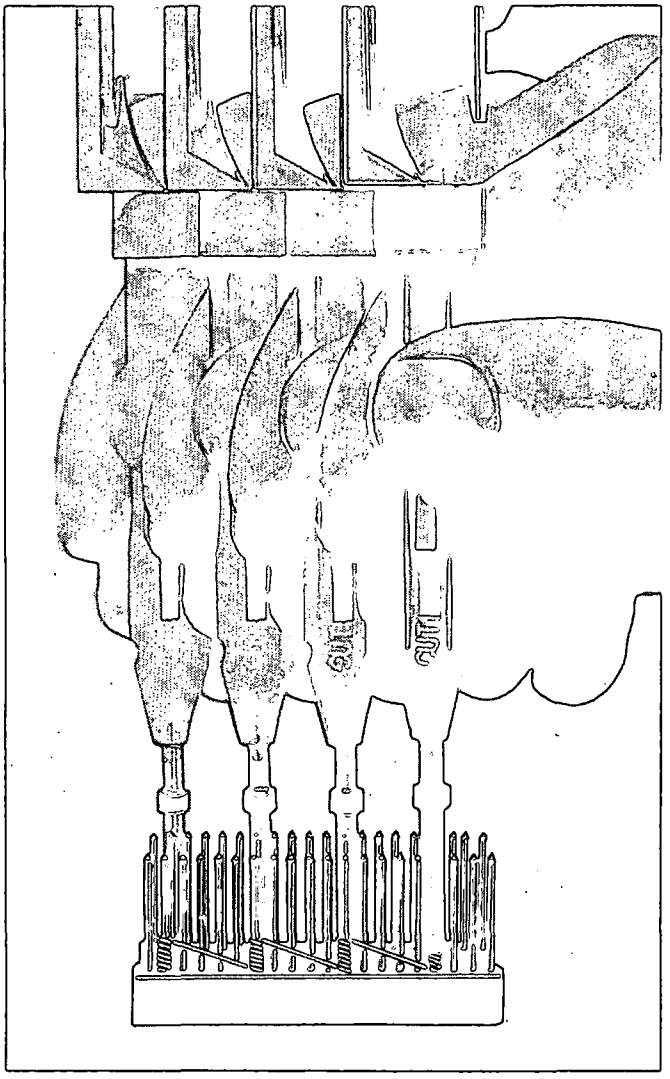
1/1 page	220 $
1/2 page	165 $

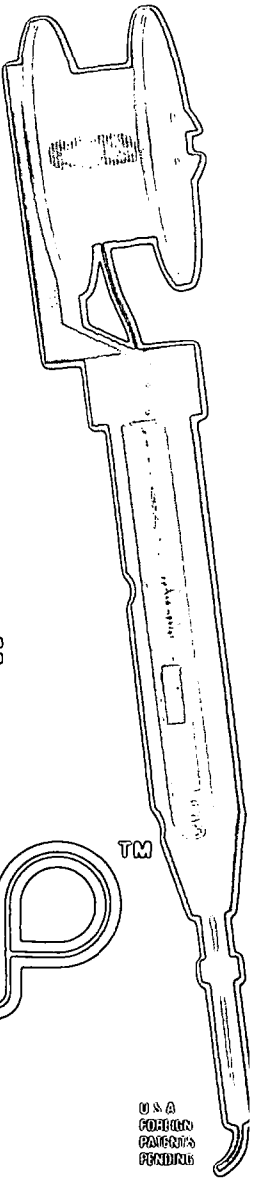
Rates for classified advertizing:

each ad 55 $

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

NEW!





WHY CUT?
WHY STRIP?
WHY SLIT?

WHY NOT...

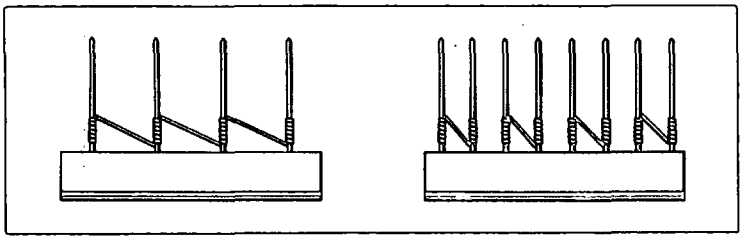
JUST WRAP™

WIRE-WRAPPING TOOL

U.S.A.
FOREIGN
PATENTS
PENDING

- AWG 30 Wire
- .025" Square Posts
- Daisy Chain or Point To Point
- No Stripping or Slitting Required
  ....JUST WRAP™....
- Built In Cut Off
- Easy Loading of Wire
- Available Wire Colors:
  Blue, White, Red & Yellow

	COLOR	PART NO.
JUST WRAP TOOL WITH ONE 50 FT. ROLL OF WIRE	BLUE	JW-1-B
	WHITE	JW-1-W
	YELLOW	JW-1-Y
	RED	JW-1-R
REPLACEMENT ROLL OF WIRE 50 FT.	BLUE	R-JW-B
	WHITE	R-JW-W
	YELLOW	R-JW-Y
	RED	R-JW-R



DAISY CHAIN

POINT TO POINT

ok MACHINE & TOOL CORPORATION 3455 CONNER ST., BRONX, N.Y. 10475 (212) 994-6600/TELEX 125091

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, Institut Jožef Stefan, Jamova 39, Ljubljana

Naročam se na časopis INFORMATICA. Predplačilo bom izvršil po prejemu vaše položnice.

Cenik: letna naročnina za delovne organizacije 300,00 din, za posameznika 100,00 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 opproximate 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 addîtional 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.

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.

Časopis INFORMATICA
Uredništvo, Institut Jožef Stefan, Jamova 39, Ljubljana

Please enter my subscription to INFORMATICA and send me the bill.

Annual subscription price: companies 300,00 din (for abroad US $ 18), individuals 100,00 din (for abroad US $ 6)

Send journal to my ☐ home address ☐ company's address.

Surname.....

Name.....

Home address

Street.....

Postal code _____ City.....

Company address

Company.....

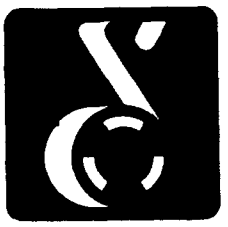
.....

Street.....

Postal code _____ City.....

Date..... Signature

.....



delta sistemi

ELEKTROTEHNA LJUBLJANA, TOZD za računalništvo Digital proizvaja in prodaja naslednje standardne računalniške konfiguracije:

DELTA 700/80

- DELTA 700 centralna procesna enota
- 512 KByte centralni pomnilnik s paritetno kontrolo,ki se lahko razširi do 4 MBytov
- 2 KByte vmesni pomnilnik spomina (cache)
- ura realnega časa
- konzolni terminal s kontrolno enoto
- dve diskovni enoti s kapaciteto po 80 MByte s kontrolno enoto
- dve magnetni tračni enoti 800/1600 b/i, 45 i/s, 9 kanalni zapis s kontrolno enoto
- asinhroni komunikacijski vmesnik
  (8 linij: EIA/CCITT modemski izhod)
  (8 linij: 20 mA tokovna zanka)
- 600 linijski tiskalnik
- KOPA 1000 alfanumerični video display terminal (2 kom)

DELTA 340/80

- DELTA 340 centralna procesna enota
- 256 KByte centralni pomnilnik s paritetno kontrolo
- 2 KByte vmesni pomnilnik (cache)
- ura realnega časa
- konzolni terminal s kontrolno enoto
- enota za baterijsko napajanje pomnilnika
- procesor s plavajočo vejico (floating point processor)
- dve diskovni enoti s kapaciteto po 80 MByte s kontrolno enoto
- dve magnetni tračni enoti (1600 b/i, 75 i/s, 9 kanalni zapis), s kontrolno enoto
- asinhroni komunikacijski vmesnik
  (8 linij EIA/CCITT modemski izhod)
  (8 linij 20 mA tokovne zanke)
- 600 linijski tiskalnik
- KOPA 1000 alafanumerični video display terminal
  (2 kom.)

DELTA 340/5

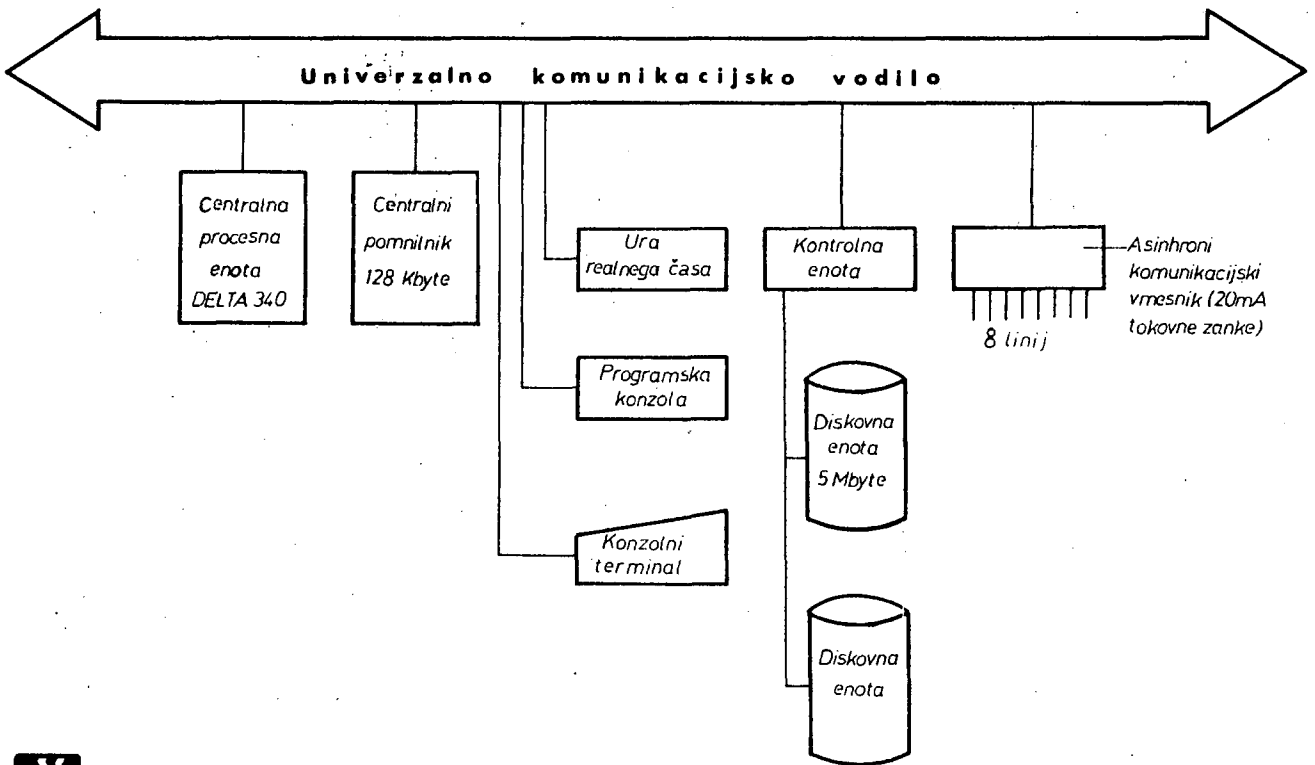
- DELTA 340 centralna procesna enota
- 128 KByte centralni pomnilnik s paritetno kontrolo, ki se lahko razširi do 256 KBytov
- ure realnega časa
- konzolni terminal s kontrolno enoto
- dve diskovni enoti s kapaciteto po 5 MByte s kontrolno enoto
- asinhroni komunikacijski vmesnik
  (8 linij: 20 mA tokovne zanke)

DELTA 340/40

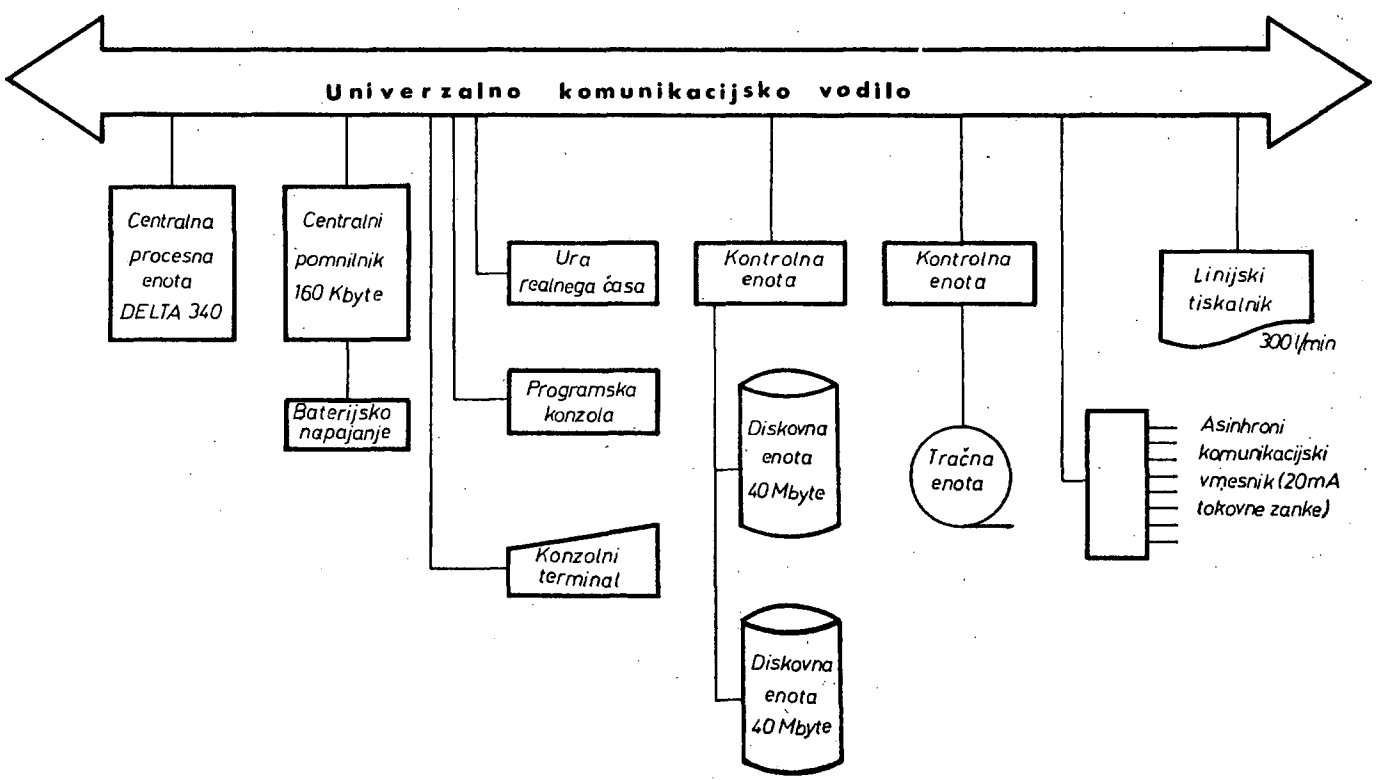
- DELTA 340 centralna procesna enota
- 160 KByte centralni pomnilnik s paritetno kontrolo do 256 KBytov
- ure realnega časa
- konzolni terminal s kontrolno enoto
- enota za baterijsko napajanje pomnilnika
- dve diskovni enoti s kapaciteto po 40 MByte s kontrolno enoto
- ena magnetna tračna enota (1600 b/i, 75 i/s, 9 kanalni zapis) s kontrolno enoto
- asinhroni kumunikacijski vmesnik
  (8 linij: 20 mA tokovne zanke)
- 300 linijski tiskalnik

NAŠTETE STANDARDNE KONFIGURACIJE LAHKO RAZŠIRITE S PRIKLJUČEVANJEM NOVIH VHODNO-IZHODNO ENOT, POVEČANJEM POMNILNIKA IPD.

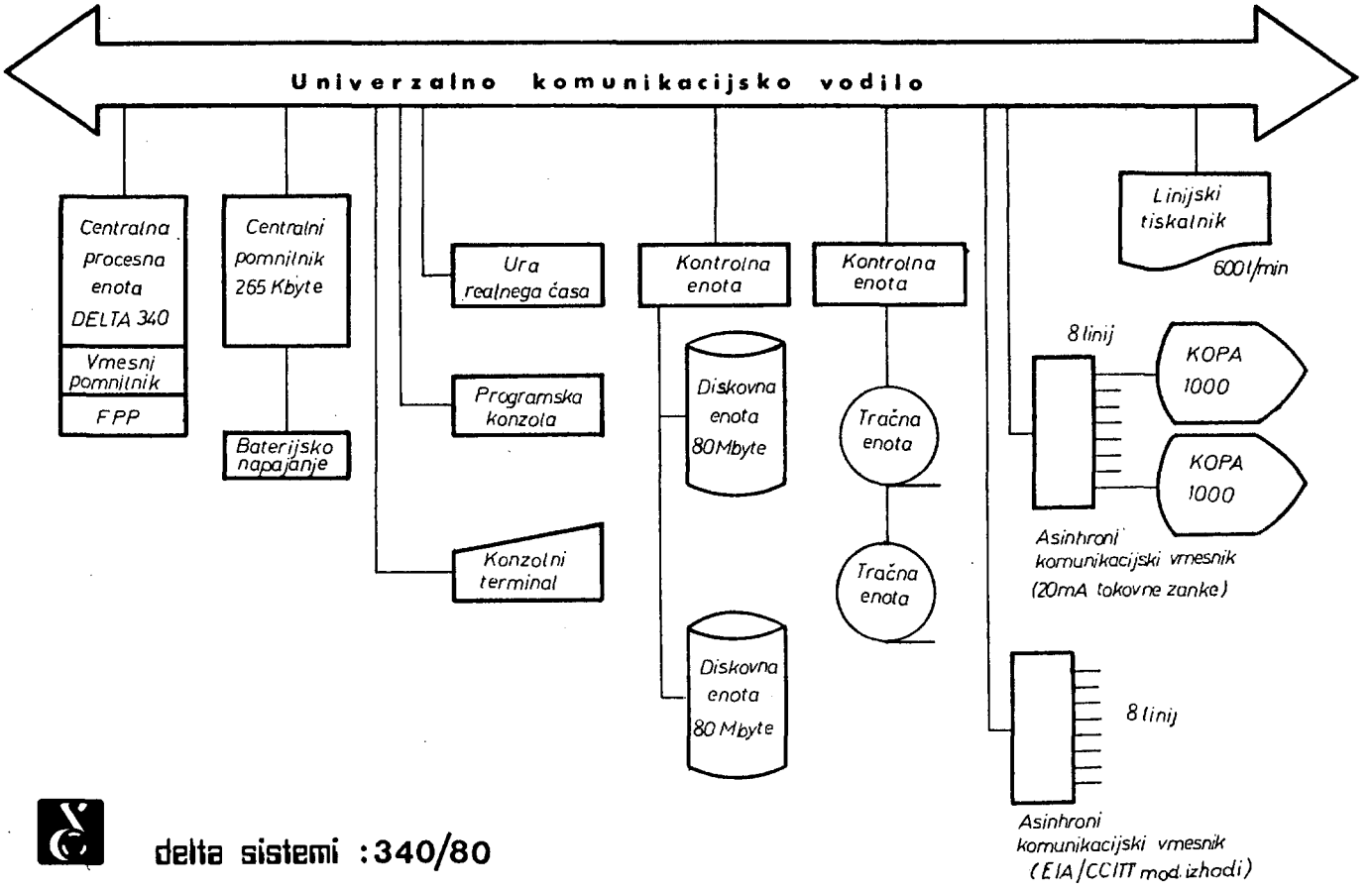
SISTEMSKI PAKETI DELTA 700/80, 340/80, 340/40 IN 340/5 VKLJUČUJEJO TUDI: OPERACIJSKI SISTEM DELTA/M S PREVAJALNIKI IN APLIKATIVNIMI PROGRAMI, ŠOLANJE V LASTNEM IZOBRAŽEVALNEM CENTRU, POMOČ PRI UVAJANJU PROGRAMSKE OPREME, INSTALACIJO RAČUNALNIŠKEGA SISTEMA IN ENOLETNO GARANCIJO ZA STROJNO IN PROGRAMSKO OPREMO.



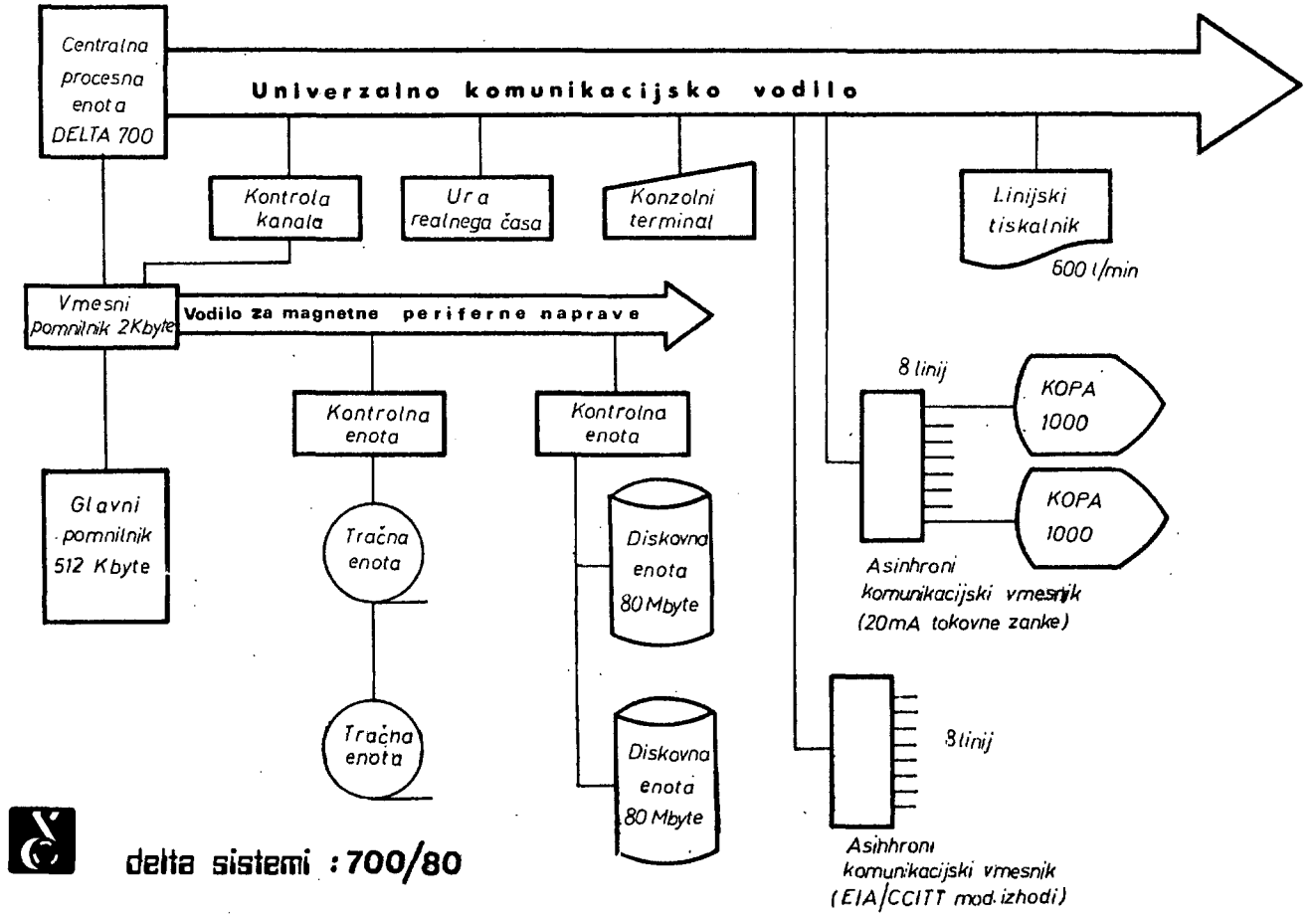
delta sistemi : 340/5



delta sistemi : 340/40



delta sistemi :340/80



delta sistemi :700/80

PROGRAMSKA OPREMA DELTA SISTEMOV

Osnovo sistemske programske opreme predstavlja DELTA/M operacijski sistem, ki je namenjen za delo v realnem času in časovno dodeljevanje resursov do 256 uporabnikom, ki lahko istočasno uporabljajo sistem. Glavna karakteristika DELTA/M sistema je interaktivnost. Človek in sistem komunicirata preko posebne enote, ki je običajno video terminal. Vsak monitorski ukaz se lahko vnese preko poljubnega terminala, če le uporabnikovo geslo zadošča ustrezni stopnji tajnosti. To pomeni z vidika uporabnika enake možnosti, kot da bi delal sam na sistemu.

Večuporabniško okolje zahteva zaščito med uporabniki samimi, saj bi lahko napaka enega uporabnika povzročila težave vsem drugim. Zaradi tega obstaja med uporabniki zaščita na nivoju programske opreme in na nivoju strojne opreme. Vsak disk je razdeljen v več logičnih področij, od katerih jih vsak uporabnik lahko nekaj uporablja . Praktično to pomeni, da lahko briše samo svoje nize in bere nize drugih uporabnikov, če mu le-ti to dovolijo. Elektronsko pa je zaščiten adresni prostor programov in uporaba instrukcij, ki bi lahko porušile integriteto sistema. Te lahko uporablja samo izvajalni sistem.

Multiprogramiranje je realizirano na nivoju sistema kot celote in na nivoju posameznega terminala. Tako ima lahko vsak uporabnik lastni multiprograming. To je važno predvsem za programerje, saj lahko istočasno razvijajo (prevajalnik, povezovalnik) in testirajo (izvajajo) programe.

Velika hitrost procesorja in perifernih enot ter učinkovito oblikovana programska oprema omogočata gospodarno uporabo vseh komponent DELTA računalnika.

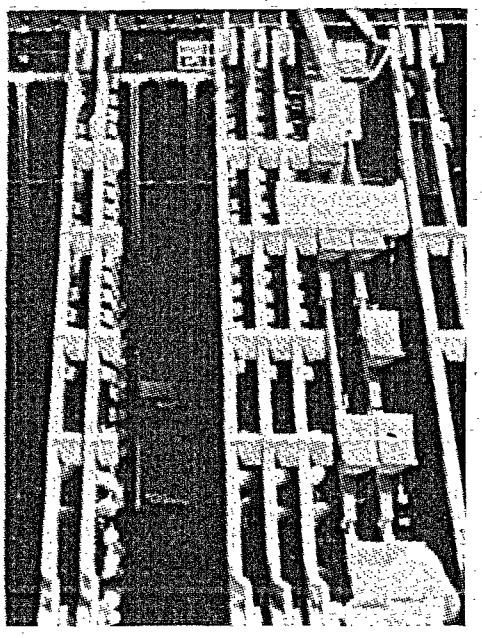
Sistem lahko istočasno upravlja industrijski proces(visoka prioriteta - realni čas), interaktivne poslovne aplikacije(srednja prioriteta), razvoj novih programov v poljubnih programskih jezikih(standardna prioriteta) in paketne obdelave (nizka prioriteta).

Aplikacijski programi se lahko pišejo v MACRO zbirnem ali enem od višjih programskih jezikov:

- FORTRAN IV
- FORTRAN IV PLUS
- BASIC 11
- RPG II
- COBOL (ANSI 74 standard)
- BASIC-PLUS-2
- PASCAL
- DATATRIEVE 11

Na tržišču ugotavljamo velike potrebe po kvalitetni komunikacijski opremi, zato posvečamo veliko pozornost prav temu področju. Komunikacijska programska oprema na DELTA/M je eden od poslov, ki se odvija v multiprogramingu in omogoča povezavo z računalniki: DELTA, PDP-11, VAX, DEC-10, DEC-20, CDC-6600, IBM 360/370, UNIVAC-11.

DELTA sistemi so namenjeni splošni uporabi. Zato je v osnovni paket vedno vključena samo tista programska oprema, ki je potrebna vsem uporabnikom. Vsak pa si lahko izbere dodatno sistemsko ali aplikativno programsko opremo. DELTA/M namreč ohranja popolno kompatibilnost navzdol z RSX-11/M operacijskim sistemom firme DEC. Ta operacijski sistem je zelo razširjen, zato je tudi ponudba ELEKTROTEHNE, DEC-a in drugih proizvajalcev zelo velika.



PODROBNE INFORMACIJE O NAKUPU DELTA SISTEMOV NUDI ELEKTROTEHNA LJUBLJANA, TOZD ZA RAČUNALNIŠTVO DIGITAL:

LJUBLJANA
Linhartova 62a
tel. (061) 323-585

ZAGREB
Aleja Borisa Kidriča 2
tel. (041) 516-690

BEOGRAD
Karadordev trg 13
tel. (011) 694-537