ADVANCED MICROPROCESSORS AND HIGH-LEVEL LANGUAGE COMPUTER ARCHITECTURE

Knjiga z naslovom ADVANCED MICROPROCESSORS AND HIGH-LEVEL LANGUAGE COMPUTER ARCHITECTURE je zbirka najpomembnejših člankov, poročil in ekspertiz, ki jih je zbral in uredil avtor Veljko Milutinović za potrebe poučevanja predmeta z enakim naslovom na Purdue University v ZDA.

Knjiga je učbenik za študente ter hkrati tudi dober pripomoček načrtovalcem novih računalniških sistemov in vodjem razvoja, ki potrebujejo znanje o sodobnih trendih v računalniških arhitekturah. Knjiga sestoji iz 39 člankov, ki so razdeljeni v 7 delov in 14 poglavij ter ima skupno skoraj 600 strani.

Knjiga govori o arhitekturah, ki temeljijo na visokoprogramskih jezikih, to je o HLL (High Level Language) računalniških arhitekturah. Avtor deli HLL računalniške arhitekture na dve osnovni skupini:

- arhitekture z indirektnim izvajanjem ter
- arhitekture z direktnim izvajanjem.

Pri arhitekturah z indirektnim izvajanjem je za izvajanje programa potrebno izvorni kod prevesti v izvajalni kod. Pri arhitekturah z direktnim izvajanjem pa računalnik lahko direktno izvaja izvorni kod.

Nadalje deli Milutinović arhitekture z indirektnim izvajanjem v dva podrazreda:

- reducirane arhitekture in
- kompleksne arhitekture.

Kompleksne arhitekture se nadalje delijo v

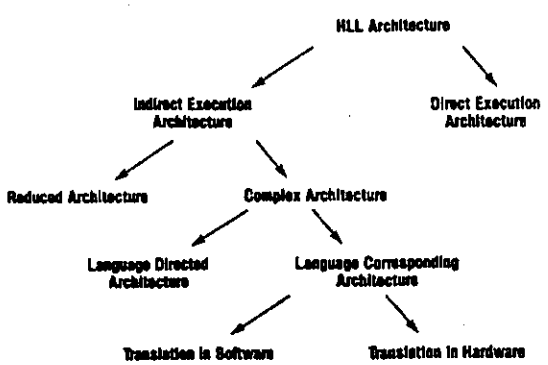
- jezikovno vodene arhitekture in
- jezikovno odvisne arhitekture.

Jezikovno vodene arhitekture lahko definiramo kot arhitekture, kjer se prožijo konstrukcije strojnega jezika na relativno visokem nivoju toda ne na nivoju HLL sintakse. Pri jezikovno odvisnih arhitekturah pa imamo direktno enolično odvisnost med HLL konstrukcijo in konstrukcijo strojnega jezika.

Na koncu so razdeljene jezikovno odvisne arhitekture še v:

- arhitekture s programskimi prevajalniki ter
- arhitekture s strojnimi prevajalniki.

Slika 1 kaže razdelitev HLL arhitektur.



Slika 1: Razdelitev HLL arhitektur.

Vtis o knjigi si najhitreje ustvarimo s pregledom kazala knjige, ki podaja poleg naslovov vseh člankov tudi strukturo knjige in razdelitev vsebine v posamezne dele in poglavja.

Table of Contents

V prvem poglavju knjige so 4 uvodni članki svetovno poznanih avtorjev (Flynn, Wulf, Myers,

Ditzel), ki obravnavajo odnos med programskim jezikom in arhitekturo računalnika. Hkrati predstavljajo izbrani članki pregled področja ter omogočajo bralcu, da na podlagi bibliografije razširi svoje znanje v posameznih specialnostih.

Drugo poglavje vsebuje 2 članka o vlogi VLSI tehnologije na računalniške arhitekture. Prvi članek (Hennessy) obravnava VLSI tehnologijo, ki temelji na klasični silicijevi tehnologiji, drugi članek (Milutinović) pa obravnava novo in hitro prodirajočo tehnologijo vezij na osnovi galijevega arzenida GaAs. Pristop pri načrtovanju računalniške arhitekture v mnogočem zavisi od izbrane tehnologije. Tako silicijeva kot tudi GaAs tehnologija imata vsaka svoje značilnosti, ki postavljajo specifične zahteve pri načrtovanju optimalne računalniške arhitekture. Poglavje obravnava reducirane arhitekture in sicer RISC računalnik iz Berkeley, MIPS iz Stanforda in ostale.

Prvi članek drugega poglavja, katerega avtor je Hennessy, primerja dva osnovna pristopa v arhitekturi, to je arhitekturo, ki zajema kompleksen nabor ukazov in arhitekturo z reduciranim naborom ukazov, kjer je obakrat procesor realiziran v VLSI tehnologiji na osnovi silicija. Članek analizira učinkovitost izvajanja prevedenega koda visokoprogramskega jezika.

Drugi članek je napisal Milutinović in predstavlja GaAs kot možno osnovo za implementacijo VLSI procesorjev. Pri tem pa nam nova tehnologija narekuje novo arhitekturo vezij in rešitve, ki so bile razvite za silicijevo okolje na splošno niso uporabne pri vezjih, ki temeljijo na GaAs. Kaže, da je pri vezjih, ki temeljijo na GaAs učinkovita samo arhitektura z reduciranim naborom ukazov.

Poglavja 3, 4 in 5 so posvečena kompleksnim arhitekturam, to je jezikovno odvisnim in jezikovno vodenim arhitekturam. V tretjem poglavju je narejen pregled različnih pristopov pri zasnovi in implementaciji novih mikroprocesorjev in HLL arhitektur ter analiza performans različnih sistemov.

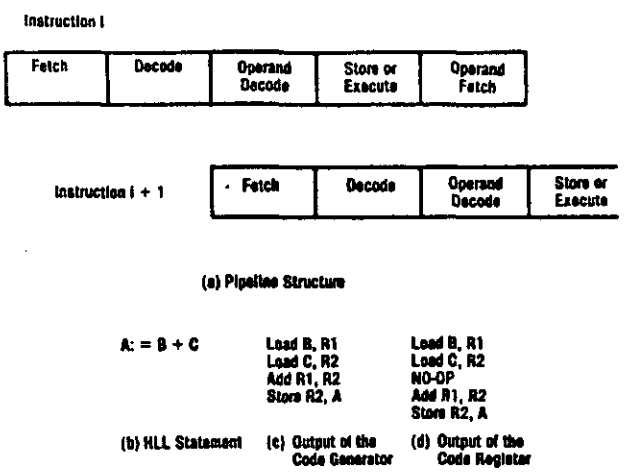
Četrto poglavje je posvečeno RISC (Reduced Instruction Set Computer) arhitekturam. Ukvarja se z izborom reduciranega nabora ukazov in nekaterimi specifičnimi implementacijami, ki temeljijo na reduciranem naboru ukazov. Posebej je v prvem članku (Petterson) obdelan projekt University of California at Berkeley UCB-RISC, ki omogoča hitrejše izvajanje programa napisanega v visokoprogramskem jeziku. Bistvo rešitve problema je v metodi kako zasnovati *arhitekturo*, da bo čim bližje tistim konstrukcijam strojnega jezika, ki se pri visokoprogramskih jezikih najpogosteje uporabljajo. Hkrati naj bo arhitektura računalnika zasnovana tako, da bo zmanjšana potreba po komunikaciji z ostalimi vezji . Tako ima RISC I arhitektura le 31 ukazov od katerih večina opravlja le preproste ALU in pomične (shift) operacije na registrih.

Drugi članek v tem poglavju (Petterson) obravnava implementacijo VLSI vezja za cache ukaze za UCB-RISC računalnik. To VLSI vezje je neobčutljivo na napake (fault tolerant), vsebuje poseben programski števec, izvaja komprimiranje kode in omogoča razširljivost. Končni cilj skupine v Berkeleyu je združiti ukazni cache pomnilnik ter podatkovni cache pomnilnik s centralno procesno enoto v eno vezje (chip).

Naslednji članek v četrtem poglavju (Tamir) se ukvarja z različnimi strategijami in metodami, ki so povezane z vodenjem večkenskih registerskih datotek v računalnikih kot je na primer UCB-RISC. Pri visokoprogramskih jezikih je klicanje procedure časovno najbolj zamudna operacija. V splošnem imajo lahko RISC programi še več klicev procedur kot običajni računalniki saj so kompleksni ukazi, ki jih najdemo v CISC arhitekturah realizirani kot podprogrami v RISC arhitekturi. Zato morajo biti klici procedur pri RISC računalnikih kar se da hitri. To je realizirano z registerskimi okni. Večokenska registerska datoteka je eden od tistih pristopov, ki bistveno pripomore k zmanjšanju potreb po komunikaciji z drugimi vezji. Ideja je v tem, da v vezje vgradimo določeno število delno prekrivajočih se registerskih oken, pri čemer je ob vsakem trenutku dostopno prevajalniku ali programerju v zbirnem jeziku samo posamezno okno. Ostala okna vsebujejo spremenljivke za ostale procese. Eden od problemov pri tem pristopu je kako ravnati, če pride pri registerski datoteki do presežka (overflow). Avtorja članka ugotavljata, da je optimalno število registrov, ki shranjujejo presežek enako ena.

Zadnji članek v četrtem poglavju (Ungar) opisuje zasnovo in implementacijo mikroprocesorja, ki je namenjen za jezik Smalltalk. Ta procesor temelji na reduciranem naboru ukazov, a se bistveno razlikuje od UCB-RISC procesorjev. To nam potrjuje domnevo, da se vsaka reducirana arhitektura navezuje na določeno aplikacijo zato različno aplikativno okolje vodi do različnih konkretnih rešitev RISC procesorja.

Peto poglavje vsebuje štiri članke in je posvečeno MIPS pristopu. MIPS je okrajšava za Microprocessor without Interlocked Pipeline Storage. Ta koncept so razvili na Stanford University. MIPS pristop je primeren za vezja, ki temeljijo na tehnologiji silicija, kot tudi za vezja ki temeljijo na tehnologiji GaAs. Če primerjamo MIPS arhitekturo z arhitekturo UCB-RISC vidimo, da je za MIPS arhitekturo značilna nizka kompleksnost VLSI vezij, zahteva pa kompleksnejši način programiranja in prevajanja. Eno osnovnih prednosti MIPS arhitekture je dejstvo, da je cevna sinhronizacija izvedena s programskim pristopom in ne več v materialni računalniški opremi. To je ugodno predvsem zaradi lažje implementacije VLSI vezij in tudi hitrosti procesorja. Seveda pa mora prevajalnik sedaj generirati kod, ki je brez cevnih konfliktov. Generator koda zadosti tem zahtevam tako, da vstavi ukaze NO-OP povsod tam, kjer je to potrebno. Optimizator koda poskuša nato zamenjati čim več ukazov NO-OP s kodo od kjerkoli drugje pod pogojem, da na ta način ne vpliva na odvisnost podatkov. Tu vidimo potrebo po kompleksnejši tehnologiji prevajalnikov.



Slika 2.: Primer seštevanja z MIPS arhitekturo, ki kaže izhod generatorja koda in reorganizatorja koda.

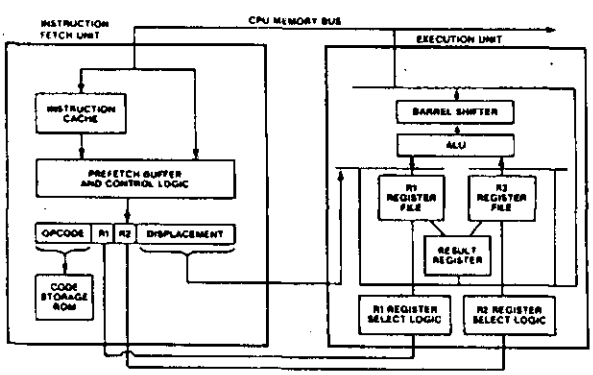
Naslednji članek (Przybylski) je tehnično poročilo Stanford University, ki opisuje VLSI implementacijo MIPS. Ta članek obravnava področja kot so nabor ukazov, cevna organizacija, ukrepanje pri izjemah, podpora za virtualni pomnilnik, podpora operacijskega sistema in visokoprogramskega jezika itd. Članek vsebuje tudi podatke o merjenju performans sistema.

Naslednji članek (Gross) se osredotoča na probleme, ki so povezani s programsko implementacijo aritmetike s plavajočo vejico za MIPS mikroprocesorje. Pri izbiri reduciranega nabora ukazov se pojavlja vprašanje, kako je tak reduciran nabor primeren za aritmetiko plavajoče vejice in za aritmetiko na sploh. Nabor ukazov za MIPS arhitekturo vsebuje ukaz, ki ustreza množenju po Booth algoritmu, kar je dalo relativno dobre performanse.

Zadnji članek v petem poglavju (Hennessey) analizira številne činitelje, ki so povezani z optimizacijo koda v MIPS okolju. Članek definira tip cevnih konfliktov do katerih lahko pride v MIPS mikroprocesorju. Govori o razlikah pri optimizaciji koda, ki se izvaja po alokaciji registrov ali sočasno z alokacijo. MIPS optimizator koda uporablja prvi pristop. Podana je teorija MIPS kodnega optimizatorja in podatki o oceni performans.

6. poglavje je posvečeno arhitekturam za direktno izvajanje. Vsebuje štiri članke, ki obravnavajo najzanimivejše procesorje z reduciranim naborom ukazov. Trije taki procesorji izhajajo iz industrije, to so: IBM (IBM 801), Ridge (Ridge-32) in Pyramid (Pyramid 90X), četrti procesor pa je bil razvit na University of Reading v Angliji. Angleški procesor je narejen v VLSI tehnologiji (RIMMS) ostali trije pa temeljijo na hitri SSI/MSI tehnologiji. Intenzivne raziskave in razvoj procesorjev z reduciranim naborom ukazov tečejo tako v industrijskem okolju (Fairchild, Hewlett-Packard, RCA, TRW, Inmos) kot tudi na univerzah (Caltech, Purdue, ULCA, Wisconsin in drugje).

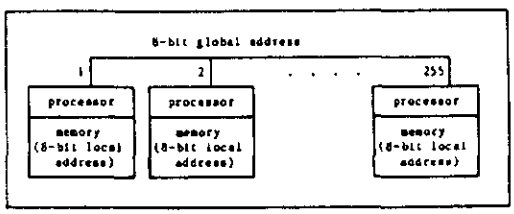
Nekateri tovrstni projekti so se pričeli že sredi sedemdesetih let. Tak je na primer projekt za miniračunalnik IBM 801. Članek, katerega avtor je Radin, govori o tem projektu in podaja nekatere značilnosti samega procesorja in tudi sistema kot celote. Opisuje na primer nabor ukazov, ki je realiziran popolnoma v materialni računalniški opremi, hierarhičnost pomnilnika, organizacijo V/I komunikacije, ki omogoča centralnemu procesorju izvajanje ukaza v skoraj vsakem ciklu in drugo. Projekt je bil poleg tega usmerjen k realizaciji sposobnejših prevajalnikov.



Slika 3.: Zgradba Ridge-32 procesorja.

Basart in Folger v svojem članku predstavljata zasnovo in implementacijo superminiračunalnika Ridge-32, ki je verjetno prvi komercialno dobljiv RISC računalnik. Članek se ukvarja z osnovnimi cilji arhitekture, z izborom nabora ukazov, z implementacijo sistema in oceno performans. Ta računalnik je narejen za hitro izvajanje zahtevnih grafičnih aplikacij. Slika 3 kaže strukturo Ridge-32 procesorja, ki je implementiran s komercialno Schottky bipolarno logiko. Procesor ima ločeno dostavno (fetch) enoto in izvajano enoto. Dostavna enota vedno vsebuje naslednji ukaz, ki se bo izvedel. Dostavna enota in izvajalna enota tvorita štiristopenjsko cevno arhitekturo, ki omogoča prekrivanje izvajanja ukazov. Ridge-32 vsebuje tudi začasen pomnilnik (cache), kjer se hranijo vsi pred kratkim izvedeni ukazi.

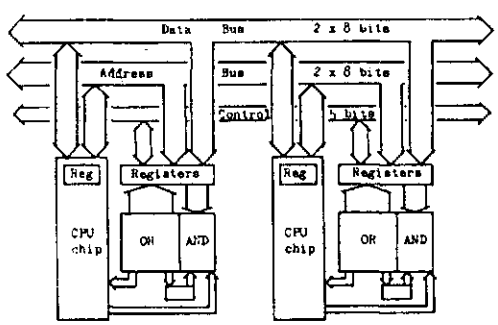
Foti s sodelavci v četrtem članku šestega poglavja opisuje projekt RIMMS mikroračunalnika (Reduced-Instruction set Multi-Microprocessor System project). RIMMS sestoji iz linearnega polja 255 mikroračunalnikov, ki komunicirajo preko skupnega vodila kot kaže slika 4.



Slika 4.: Zgradba RIMMS mikroračunalnika z 255 procesorji.

Vsak mikroračunalnik ima preprost procesor in 256 zlogov lokalnega pomnilnika.

Ta mikroprocesor ima skrajno reduciran nabor ukazov in je namenjen uporabi v multimikroprocesorskem okolju. Zato so posebno pozornost posvetili strojni računalniški podpori za medprocesorsko komunikacijo. Vodilo sestoji iz 16 bitnega naslova, 16 bitnega podatka in dodatnih bitov za delo s pomnilnikom, kar omogoča izvajanje NO-OP posega po pomnilniku. Slika 5 kaže organizacijo in kontrolo vodila ter pomnilnika za RIMMS mikroračunalnik.



Slika 5.: Nadzor vodila in pomnilnika za RIMMS mikroračunalnik.

Med najzanimivejše lastnosti tega mikroprocesorja sodijo metode, ki jih uporabljajo za podporo programskih konstrukcij kot so FORK, JOIN, REMOTE LOAD in REMOTE STORE. Razvili so jezik za paralelno procesiranje imenovan BASAL, ki podpira osnovno idejo mikroračunalnika z

reduciranim naborom ukazov za multimikroprocesorsko okolje.

Ragan-Kelley opisuje razvoj superminiračunalnika pri Pyramid Technology Corporation, ki prav tako temelji na reduciranem naboru ukazov. Cilj projekta je sposoben računalnik, ki podpira UNIX operacijski sistem in visokoprogramske jezike kot so C in Pascal. Računalnik je namenjen večuporabniškem okolju. Na razvoj tega projekta je verjetno vplival projekt RISC iz Berkeley-ja, kljub temu pa vsebuje ta projekt številne nove zamisli.

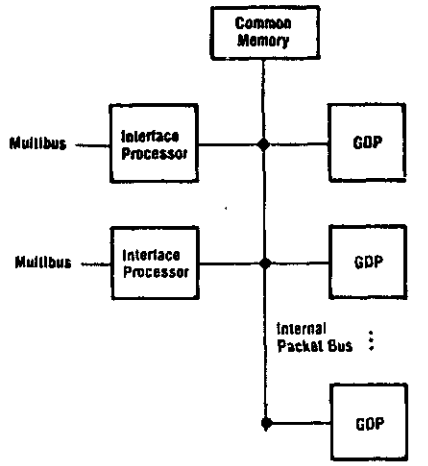
Sedmo poglavje se ukvarja s skladovnimi (stack) računalniki. Sklad omogoča učinkovito organizacijo preklaplanja konteksta kot tudi aritmetiko. S skladovno arhitekturo se posebej ukvarjajo Burroughs, Hewlett-Packard, Microdata, Intel, Xerox in Zilog. Prvi članek v tem poglavju opisuje osnovno idejo skladovne arhitekture, še posebej povezovanje in kontrolo podprogramov ter evaluacijo izrazov. Podana je primerjava skladovnih računalnikov in tradicionalnih računalnikov, ki temeljijo na splošnih registrih.

Blake je avtor članka, ki je uvod v optimalno zasnovo skladovnega računalnika in podaja pristop firme Hewlett-Packard. Članek analizira potrebe in značilnosti programov pisanih v visokoprogramskih jezikih in njihov vpliv na sklad. Posebna pozornost je posvečena strojni opremi skladovnih računalnikov, naboru ukazov, naslovnem prostoru za procese, spremljanju procesov itd.

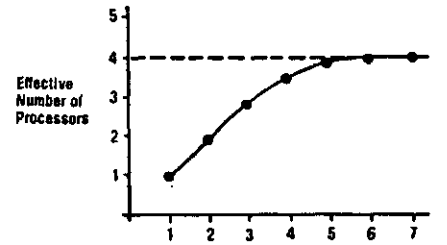
Earnest v tretjem članku sedmega poglavja obravnava več skladovnih računalnikov, ki so jih razvili pri Burroughs v zadnjih 20 letih. Eden glavnih razlogov, da se nekateri proizvajalci osredotočajo na skladovne računalnike je dejstvo, da je relativno lahko pisati prevajalnike za skladovne računalnike in, da je ta koda ponavadi precej kompaktna.

Osmo poglavje se ukvarja s sodobnimi mikroprocesorji za kompleksen nabor ukazov, ki jih pogosto imenujemo CISC (Complex-Instruction-Set Computers). Poglavje vsebuje 8 člankov, ki govorijo o 32 bitnih procesorjih, ki so namenjeni delu z visokoprogramskimi jeziki. Imajo bogat nabor ukazov, od katerih nekateri zelo spominjajo na konstrukte, ki so značilni za moderne visokoprogramske jezike. Raziskave in razvoj na tem področju hitro napredujejo in novi ter popolnejši 32 bitni mikroprocesorji se stalno pojavljajo v ZDA, na Japonskem in v Evropi. Uveljavljeno je prepričanje, da so CISC računalniki bolj primerni za podporo programske zanesljivosti kot pa RISC računalniki. Prav tako imajo močnejše sposobnosti obdelave aritmetičnih izrazov. Saveda pa so CISC računalniki v primerjavi z RISC računalniki počasnejši pri izvajanju prevedenega programa, ki je pisan v visokoprogramskem jeziku. Ker so CISC procesorji kompleksnejši, zahtevajo daljši čas za razvoj procesorja. Dejstvo pa je, da so novi CISC procesorji zelo sposobni, uporabljajo visoko stopnjo cevnega paralelizma in vsebujejo že na čipu cache pomnilnik ter mehanizme za delo s pomnilnikom. Končna odločitev in tekma med RISC in CISC mikroprocesorji je še pred nami in ni jasno kdo bo zmagovalec.

Gupta v svojem članku primerja 4 zgodnje 32 bitne mikroprocesorje. Tehnično poročilo firme Intel podaja bistvene značilnosti procesorja iAPX 432. Dobra lastnost procesorja iAPX 432 je razširjivost sistema. Večjo sposobnost sistema dosežemo brez spreminjanja programske opreme tako, da povečamo število procesorjev v sistemu na največ 5 GDP (General Data Processors). Pet takih procesorjev si deli isto vodilo in imajo zato performanse, ki so enake vsoti treh neodvisnih GDP. Razširitev iAPX v multiprocesorski sistem in performanse iAPX 432 multiprocesorskega sistema kaže slika 6.



(a) An iAPX 432 Multiprocessor System



(b) The Efficiency Function

Slika 6.: iAPX 432 multiprocesorski sistem in njegove performanse kot funkcija števila procesorjev.

Beyers v svojem članku podaja notranjo organizacijo procesirja HP-FOCUS. McGregor s sodelavci obravnava bistvene značilnosti Motorolinega procesorja MC68020. Mateosian opisuje glavne značilnosti procesorja NS32032 proizvajalca National Semiconductor. Na koncu poglavja pa je Patel podal pregled Zilogovega procesorja Z80,000. Vsi ti članki so zgolj pregledni članki.

Deveto poglavje podaja nekaj zgodnjih poskusov na področju HLL arhitektur s programskimi prevajalniki. To so arhitekture z naborom strojnih ukazov, ki je v enoličnі (eden proti enemu) zvezi z ukazi značilnega visokoprogramskega jezika. Prevajanje je izvršeno programsko. Pogosto te arhitekture imenujemo jezikovno skladne arhitekture s programskim prevajanjem. Zanimanje za te arhitekture se je začelo že leta 1960, ko se je pojavila potreba po računalniku, za katerega bi bilo lažje pisati sistemski in uporabniški kod. Glavni problem tega pristopa pa je kompleksnost materialne računalniške opreme.

To poglavje vsebuje 2 članka. Prvi, katerega avtor je Hassit et al., opisuje enega prvih tovrstnih eksperimentov pri IBM. Rezultat tega eksperimenta je bil računalnik, ki je bil namenjen za jezik APL. To je bil pravzaprav računalnik IBM S/360 M/25, ki je bil mikropro-

gramiran in je generiral APL kod. Osnovni razlog za ta razvoj je bila želja razviti učinkovito arhitekturno podporo za jezike kot so APL ali SNOBOL. Ti jeziki ponavadi potrebujejo interpreter in konvencionalen pristop ima lahko za posledico znatno zmanjšanje hitrosti izvajanja programa.

Nielsen predstavlja v drugem članku devetega poglavja glavne rezultate študije, katere cilj je bil načrt računalniške arhitekture in programskega jezika za vesoljske aplikacije.

Oba članka iz tega poglavja podajata rezultate eksperimenta. Računalnika, ki sta pri tem nastala sta danes zastarela, omenjena pa sta v tej knjigi zato, ker sta s svojimi idejami in eksperimentalnimi rezultati vplivala na sodobne tokove pri zasnovi novih računalniških arhitektur.

Deseto poglavje opisuje nekatere sodobne raziskave na MIT. Vsebuje dva članka, ki se navezujeta na jezikovno skladne arhitekture. Te so zanimive tako z gledišča VLSI načrtovanja vezij, kot tudi z gledišča mikroprogramiranja. Večina računalnikov, ki temelji na tem pristopu vsebuje mikroprogram za podporo konstrukcij visokoprogramskega jezika. Trenutno stanje razvoja VLSI lahko učinkovito podpira velike ROM pomnilnike za mikroprogram na čipu. Vemo pa, da je VLSI tehnologija premalo močna za podporo kompleksnih visokoprogramskih jezikov na popolni enolični korespondenci. Torej lahko z VLSI tehnologijo podpiramo bodisi samo izbrano podmnožico visokoprogramskega jezika ali pa uporabimo tehnologijo, ki ni VLSI in podpiramo celotni visokoprogramski jezik.

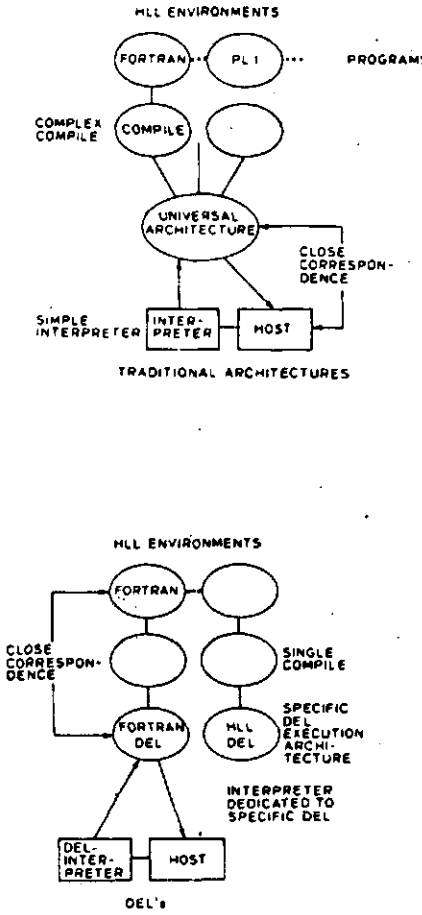
Prvi članek, katerega avtor je Sussman s sodelavci, opisuje zasnovo in implementacijo mikroračunalnika na enem vezju, ki direktno interpretira Scheme-79, to je dialekt jezika Lisp. Skupina na MIT je razvila interpreter v obliki mikrokoda, ki so ga realizirali v strojni računalniški opremi. Pri tem so uporabili dodatne in nekonvencionalne hardverske pripomoč ke ter s tem povečali učinkovitost računalnika.

Drugi članek desetega poglavja, katerega avtor je Batali et al., opisuje vezje Scheme-81, ki je naslednik vezja Scheme-79. Scehme-81 je namenjen za okolja, kjer so računalniki specializirani in kjer velike skupine računalnikov sodelujejo pri reševanju posamezne zahtevne naloge.

V 11. poglavju je govora o dveh zanimivih eksperimentih. Prvi izhaja iz šestdesetih let in opisuje Symbol računalniški sistem, to je računalnik, ki ima nekonvencionalno arhitekturo in o katerem je bilo v preteklosti veliko diskusij. Symbol je računalnik, ki ima splošen programski jezik in time-sharing operacijski sistem implementiran v materialni računalniški opremi. Vzpodbuda za tak eksperiment temelji na dejstvu, da so sčasoma postali prevajalniki zapleteni in jih je bilo težko napisati. Po drugi strani je cena materialne računalniške opreme padala, kompleksnost načrtovalnih orodij za razvoj vezij pa se je boljšala.

Drugi članek enajstega poglavja (Burkle) predstavlja laboratorijski eksperiment, ki je rezultiral v računalnik z imenom Abacus. Abacus je nastal pod vplivom računalnika Symbol, ima pa vrsto originalnih novih rešitev kot na primer kontrolo poslov (job control), učinkovito delo s podatkovnimi tipi in drugo.

Dvanajsto poglavje vsebuje dva članka, ki obravnavata DEL (Directlay Executed Language) pristop to je arhitekturo za direktno izvajanje jezika. To so, po razdelitvi iz slike 1, HLL arhitekture s strojnimi prevajalniki. Pri teh arhitekturah uporablajmo za programiranje običajen visokoprogramski jezik. S predprocesiranjem izvornega koda se prevede izvorni program v DEL obliko, kar je optimalni vmesnik med posameznim visokoprogramskim jezikom in izvajalnim računalnikom (Slika 7).



Slika 7: Primerjava med DEL arhitekturo in tradicionalno arhitekturo.

Prvi članek v tem poglavju, katerega avtor je Hoevel, podaja pregled različnih metod, ki se uporabljajo za generiranje DEL arhitekture.

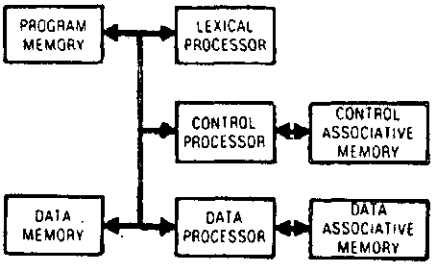
Flynn je avtor drugega članka in opisuje DELtran eksperiment, ki se navezuje na okolje programskega jezika Fortran II. Izvajalni računalnik pri tem eksperimentu je mikroprogramibilni računalnik EMMY z univerze Stanford. Članek podaja primerjavo v performansah, ko se uporablja DELtran ali pa druge konvencionalne arhitekture kot na primer IBM 370. Prihranek pri velikosti programa in naslavljanju pomnilnika je pri uporabi DELtran-a znatem.

Flynn in Hoevel sta s svojim delom močno vplivala na razvoj celotnih HLL arhitektur, vključno z RISC arhitekturami.

Trinajsto poglavje opisuje v dveh člankih arhitekturo računalnika za direktno izvajanje, ki so jo zasnovali na University of Maryland. Bistvo pristopa univerze Maryland je dejstvo, da arhitektura omogoča direktno interpretiranje izvornega koda programov. Na ta način je semantični prepad med programskim jezikom in arhitekturo popolnoma premoščen.

Prvi članek v trinajstem poglavju (Chu) se ukvarja s sistemskimi vprašanji interaktivnega visokoprogramskega jezika in z arhitekturo za njegovo direktno izvajanje. Članek obravnava razliko med sistemom z interaktivnim prevajanjem in sistemom z interaktivnim interpetiranjem.

Drugi članek trinajstega poglavja (Chu) analizira interakcijo med procesorjem in visokoprogramskim jezikom pri arhitekturh za direktno izvajanje (Slika 8). Procesor za to arhitekturo sestoji iz treh glavnih delov, to je slovarskega dela procesorja, nadzornega dela procesorja in podatkovnega dela procesorja. V članku je prikazana interakcija teh treh delov procesorja na primerih, ko uporabljamo visokoprogramski jezik.



Slika 8: Organizacija računalnika za direktno izvajanje.

Knjiga se konča s 14 poglavjem, ki obravnava pregledne članke o razvoju HLL arhitektur v različnih razvojnih centrih po svetu. Poglavje vsebuje dva članka, ki predstavljata raziskovalne in razvojno dosežke na področju HLL računalniških arhitektur v Japonski in v Evropi. Yamamoto je v članku, ki daje pregled teh raziskav na Japonskem, prikazal strukturo PL/I procesorja, konfiguracijo Cobol računalnika, več različnih Lisp računalnikov, NEC-ov single-chip Pascal procesor in druge zanimivosti.

Drugi članek zadnjega poglavja (Sylbey) daje pregled raziskav s področja računalniških arhitektur v Evropi. Omenjeni so dosežki s področja reducirnih arhitektur z University of Kent, univerze v Bonnu in Gesellschaft fur Mathematik und Datenverarbeitung. S podatkovno vodenimi arhitekturami se ukvarjajo na University of Manchester, Katholieke Universiteit Leuven in na univerzi v Dortmundu. Z reduciranimi arhitekturami se ukvarjajo na University of Reading. Jezikovno vodene arhitekture razvijajo na Tehnični univerzi Berlin in na Univerzi Dortmund. HLL arhitekture razvijajo na univerzi Paul Sabatier v Toulouse ter na univerzah v Dortmundu, Frankfurtu ter Kaiserslautern. Arhitekture za direktno izvajanje razvijajo na Švedskem in v Franciji. Seveda je centrov, kjer raziskujejo in razvijajo HLL računalniške arhitekture v Evropi še mnogo več. V tem članku je izbranih samo nekaj najbolj zanimivih. Na koncu je priložen še spisek evropskih revij in simpozijev s področja HLL računalniških arhitektur.

Pripravil: Saša Prešern