RAZVRSTITEV NOVOGENERACIJSKIH RAČUNALNIŠKIH ARHITEKTUR

UDK: 681.3.02

Borut Robič in Jurij Šilc
Institut Jožef Stefan, Ljubljana

V članku predstavljava razvrstitev računalniških arhitektur, porojeno iz treh načinov vodenja računanja: krmilnega, podatkovnega in vodenja z zahtevo. Opisani so ustrezni računski modeli ter njihove realizacije na različnih organizacijah stroja.

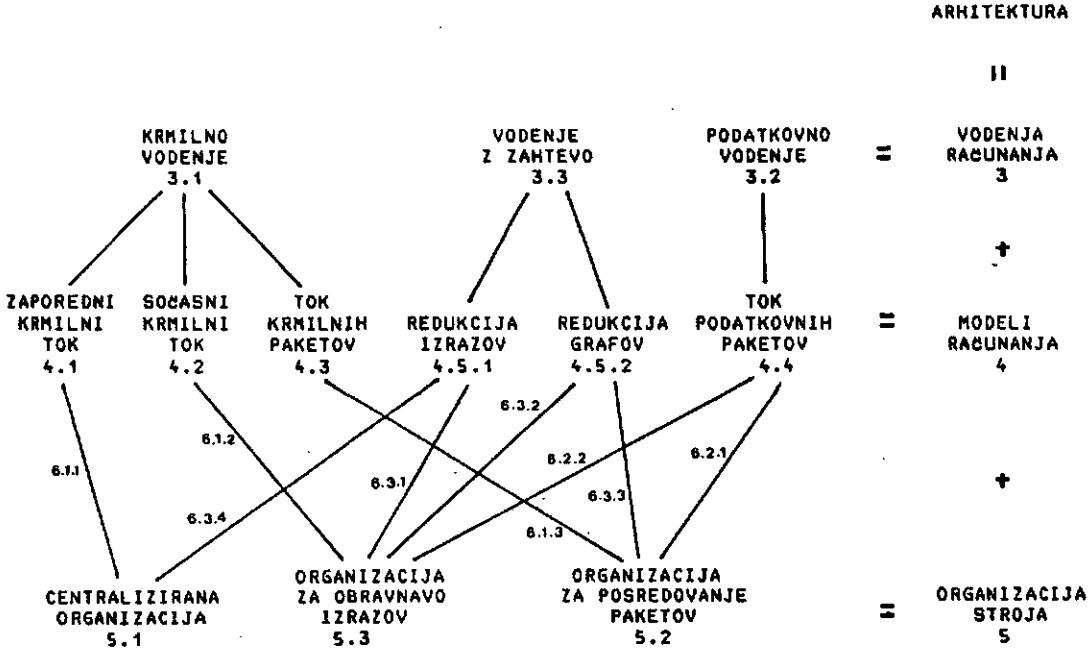
Classification of New Generation Computer Architectures - Classification of new generation computer architectures based on control driven, data driven, and demand driven computation is described. Also, we present several simple operational models and their implementations on different machine organizations.

1. Prolog

Ideja o računalnikih pete generacije je bila prvič predstavljena na mednarodni konferenci v Tokyu oktobra 1981. Peta generacija bo združevala rezultate raziskav na področjih umetne inteligence, programirnega inženiringa, VLSI (ULSI) tehnologije ter predvsem računalniških arhitektur. Slednje bodo zanesljivo prešle od zaporednega na sočasno izvrševanje. Na resnost teh raziskav kaže tudi dejstvo, da so od tokijske konference pa do leta 1984 ustanovili pet velikih nacionalnih oziroma regionalnih raziskovalnih programov:

- japonski, katerega usmerja ICOT (Institute for New Generation Computer Technology),
- ameriška DARPA (Defense Advanced Research Project Agency) in MCC (Microelectronic and Computer Technology Corporation),
- britanski Alveyjev program in
- ESPRIT (European Strategic Research Programme in Information Technology) v EGS.

Raziskave novogeneracijskih računalniških arhitektur so intenzivne in so porodile vrsto novih, ne-von Neumannovih principov računanja ter pridruženih organizacij stroja. V članku prikazujeva možno razvrstitev arhitektur, kot je bila predlagana v delu [Trel82a].



Slika 1. Elementi računalniške arhitekture.

2. Uvod

Rezultat raziskav na področju novejših računalniških arhitektur so tri osnovne družine arhitektur, ki se razlikujejo po načinu vodenja računanja [Trel81], [Trel82a]. Ustrezne arhitekture se imenujejo krmilno vodene, podatkovno vodene oziroma arhitekture, vodene z zahtevo. Znotraj vsake družine pa se arhitekture ločijo še po izbranem modelu računanja ter seveda po organizaciji stroja. Izraz računalniška arhitektura je torej določen s tremi elementi

⟨arhitektura⟩ ::= (⟨vodenje računanja⟩,
⟨model računanja⟩,
⟨organizacija stroja⟩).

Družina krmilno vodenih arhitektur združuje tradicionalne von Neumannove računalnike skupaj z njihovimi paralelnimi različicami, kot sta na primer SIMD in MIMD arhitektura. Ostali dve družini pa združujeta novejše, visoko paralelne ne-von Neumannove arhitekture.

V nadaljevanju bo prikazana podrobnejša razdelitev računalniških arhitektur ter razlike med njimi. Bralcu bo v pomoč slika 1, kjer se oznake ob posameznih elementih arhitekture nanašajo na ustrezna poglavja v članku.

3. Vodenja računanja

Računanje lahko smatramo kot zaporedno ponavljanje treh faz:
a) izbiranja /selection/,
b) preverjanja /examination/ ter
c) izvrševanja /execution/.

a) Izbiranje: v tej fazi se določi skupina ukazov programa, ki so možni kandidati za izvršitev. Izvršijo se lahko le izbrani ukazi, vendar izbor ukaza še ne zagotavlja njegove takojšnje izvršitve. Pravilo, po katerem se ukazi izbirajo, imenujemo pravilo izbiranja. Ločima tri pravila izbiranja:
- brezpogojno /imperative selection/,
- notranje /innermost selection/ ter
- zunanje /outermost selection/.

Pri brezpogojnem izbiranju se izbere ukaz ne glede na njegovo lego v programu. Izbor se izvrši na podlagi posebnega registra (programskega števca) ali pa na podlagi prisotnosti vseh krmilnih paketov. Notranje izbiranje izbere najgloblje gnezdene ukaze izraza, zunanje izbiranje pa ukaze, ki ne gnezdijo v nobenem izrazu.

b) Preverjanje: v tej fazi se ugotovi, če so izbrani ukazi izvršljivi, kar pomeni, da se preveri, če so prisotne vse potrebne vrednosti operandov v izbranih ukazih. Pravilo, po katerem poteka preverjanje, imenujemo pravilo izvršitve /firing rule/. Izvršljive ukaze prevzame tretja faza računanja, izvrševanje. Če ukaz ni izvršljiv, pa ga faza preverjanja bodisi odloži, dokler ga neka kasnejša faza izbiranja zopet ne izbere, ali pa ga poskuša napraviti izvršljivega s tem, da zahteva določitev vrednosti njegovih argumentov.

c) Izvrševanje: v tej fazi se izvršljivi ukazi dejansko izvrše.

Ločimo krmilno vodeno /control driven/ računanje, podatkovno vodeno /data driven/ računanje ter računanje vodeno z zahtevo /demand driven/.

Čeprav bodo vsa tri vodenja računanja podrobno opisana v nadaljevanju, že tu omenimo, da pri podatkovnem vodenju prisotnost vseh potrebnih operandov sproži izvrševanje ukaza, medtem ko pri vodenju na zahtevo zahteva po rezultatu sproži izvrševanje tistega ukaza, ki ta rezultat lahko priskrbi.

3.1. Krmilno vodenje

Krmilno vodeno računanje ima sledeče značilnosti:
- izbiranje ukazov je brezpogojno,
- preverjanje ukazov se ne izvaja in
- izvrševanje ukazov sledi neposredno po izbiranju.

Torej se ukazi izvrše takoj, ko so izbrani.

3.2. Podatkovno vodenje

Značilnosti podatkovnega vodenja lahko strnemo v naslednje:
- izbiranje poteka tako, da se vsakemu ukazu dodeli procesor,
- preverjanje sestoji iz sprotnega ugotavljanja prisotnosti vseh vhodnih operandov ukaza,
- izvrševanje pa sestoji iz izvršitve vseh ukazov, katerih vsi potrebni operandi so prisotni.

Torej pri podatkovno vodenem računanju ukazi pasivno čakajo na svoje vhodne operande.

3.3. Vodenje z zahtevo

Vodenje z zahtevo pa ima sledeče značilnosti:
- ukaz se izbere (navadno po pravilu zunanjega izbiranja) tedaj, ko nek prej izbrani ukaz zahteva njegov rezultat,
- v fazi preverjanja se ugotovi, če so izbrani ukazi izvršljivi, torej če so znane vrednosti njihovih operandov. Če vrednost operanda še ni znana, se posreduje zahteva tistemu ukazu, ki ga lahko izračuna,
- izvrševanje je sestavljeno iz izvršitve vseh tistih ukazov, katerih potrebni operandi so znani.

Torej se pri vodenju na zahtevo ukaz izvrši tedaj, ko je bila podana zahteva po njegovem rezultatu.

4. Modeli računanja

Vodenja računanja realiziramo z različnimi modeli računanja.

Krmilno vodenemu računanju pripadajo sledeči modeli:
- z zaporednim krmilnim tokom /sequential control flow/,
- s sočasnim krmilnim tokom /parallel control flow/,
- s tokom krmilnih paketov /control flow with control tokens/.

Podatkovno vodenemu računanju pripada le model:
- s tokom podatkovnih paketov /data flow/.

Računanje, ki je vodeno z zahtevo pa je moč realizirati z dvema modeloma:
- z redukcijo izrazov /string reduction/,
- z redukcijo grafov /graph reduction/.

V vsakem modelu računanja se zrcali izvirni način vodenja računanja v obliki sprožitvenega ter podatkovnega mehanizma.

Sprožitveni mehanizem /control mechanism/ določa način, kako izvršitev nekega ukaza vpliva na izvršitev ostalih ukazov in s tem vodenje izvrševanja programa. Sprožitveni mehanizem je lahko:

- zaporeden /sequential/: ukazi se izbirajo drug za drugim in se po vrsti izvršujejo,
- sočasen /parallel/: kjer se na nek način ugotavlja prisotnost vhodnih operandov ter povzroči začetek izvrševanja tistih ukazov, ki imajo na voljo vse potrebne operande,
- rekurziven /recursive/: kjer se posreduje zahtevo po operandih ter sproži izvrševanje tistega ukaza, katerega rezultat je bil zahtevan.

S podatkovnim mehanizmom /data mechanism/ pa je določen način, po katerem si ukazi delijo skupne operande. Pri tem si lahko ukazi delijo operand s pomočjo:

- vstavljene vrednosti /literal/: če je vrednost operanda znana že pred pričetkom računanja, se takoj vpiše v vse ukaze, ki jo uporabljajo,
- sprotne vrednosti /value/: kopije operanda, ki se je izračunal, se posredujejo vsem ukazom, ki jim je operand skupen,
- reference /reference/: tu vsebujejo vsi ukazi referenco (naslov) na skupen operand.

Zvezo med obema mehanizmoma ter različnimi modeli računanja prikazuje tabela 1.

SPREMENJIVI MEHANIZEM		PODATKOVNI MEHANIZEM: vstavljene in sprotne vred.	PODATKOVNI MEHANIZEM: vstavljene vred. in reference
	zaporeden		zaporedni krmilni tok
	sočasen	tok podatkovnih paketov	sočasni krmilni tok
			tok krmilnih paketov
	rekurziven	redukcija izrazov	redukcija grafov

Tabela 1. Modeli računanja glede na sprožitveni in podatkovni mehanizem.

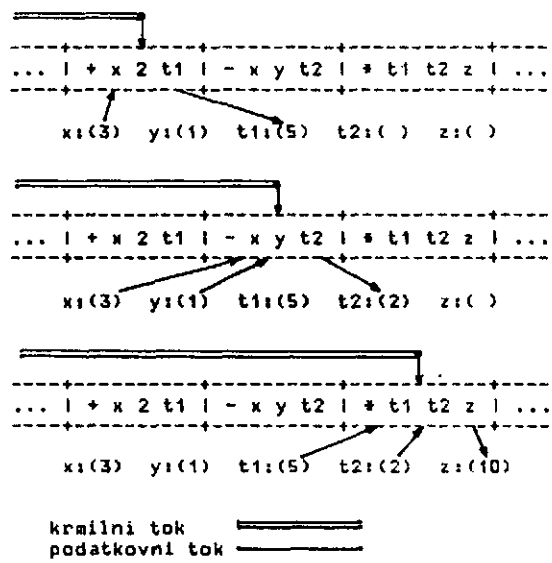
4.1. Zaporedni krmilni tok

Model z zaporednim krmilnim tokom ima naslednje značilnosti:

- krmilno vodenje računanja,
- zaporedni sprožitveni mehanizem,
- podatkovni mehanizem s pomočjo vstavljenih vrednosti in referenc.

Model z zaporednim krmilnim tokom uporablja en sam procesor za računanje in programski števec, s katerim se izbere naslednji ukaz. Torej obstaja en sam krmilni tok, ki prehaja z ukaza na ukaz in temelji na zaporednem sprožitvenem mehanizmu. Ko krmilni tok doseže ukaz, se razrešijo vse reference, kar pomeni, da se dostavijo vrednosti operandov iz naslovljenih pomnilniških lokacij. V modelu z zaporednim krmilnim tokom se torej uporablja podatkovni mehanizem z vstavljenimi vrednostmi in referencami. Ukaz se nato takoj izvrši, saj fazi (brezpogojnega) izbiranja neposredno sledi faza izvrševanja - računanje je torej krmilno vodeno. Rezultat izvršitve se vpiše v skupno lokacijo. Po izvršitvi ukaza se programski števec bodisi neposredno ali posredno ažurira, s čimer krmilni tok preide na naslednji ukaz.

Izvrševanje v modelu z zaporednim krmilnim tokom ponazorimo na stavku z := (x+2)*(x-y), ki je predstavljen z zaporedjem ukazov, pri čemer vsak vsebuje operacijo, kateri sledijo operandi (Slika 2). Ti so bodisi vstavljene vrednosti ali pa naslovi lokacij, kjer so shranjene vrednosti operandov. Prenašanje vrednosti med ukazi se vrši s pomočjo skupnih lokacij v pomnilniku.



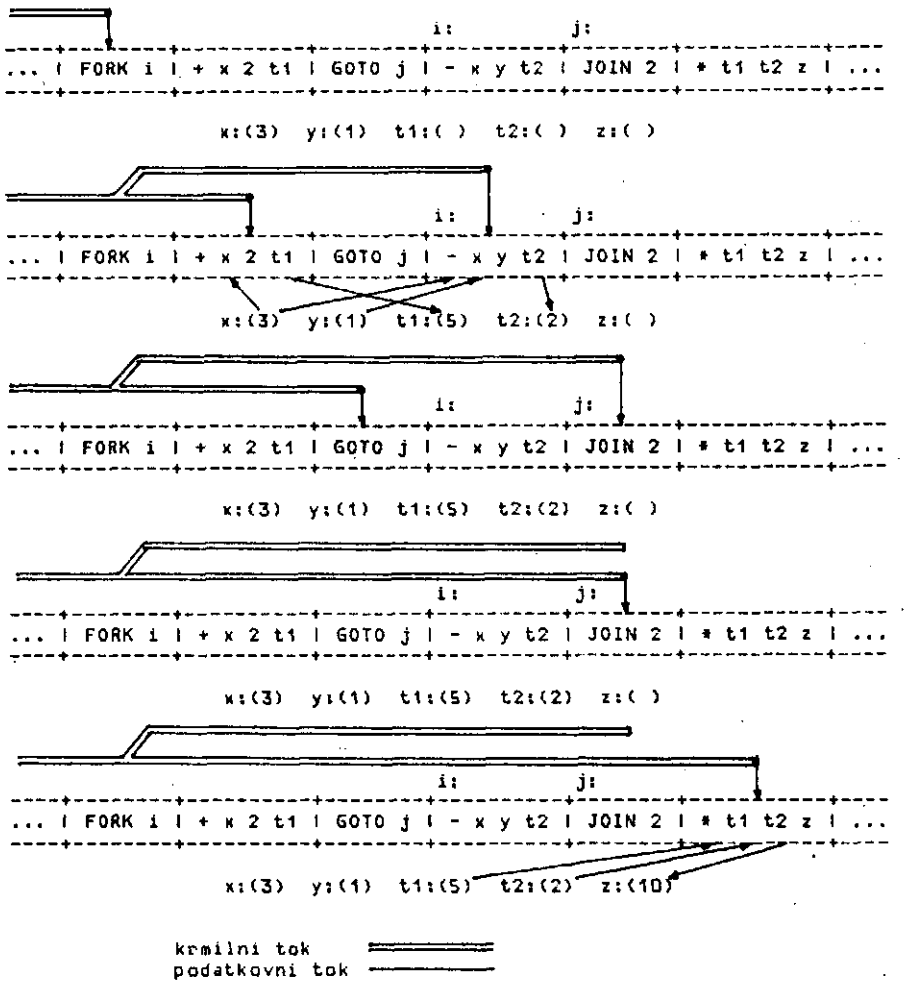
Slika 2. Računanje v modelu z zaporednim krmilnim tokom.

4.2. Sočasni krmilni tok

Model sočasnega krmilnega toka ima sledeče značilnosti:

- krmilno vodenje računanja,
- sočasni sprožitveni mehanizem,
- podatkovni mehanizem s pomočjo vstavljenih vrednosti in referenc.

V modelu s sočasnim krmilnim tokom uporabljamo ukaza FORK za začetek sočasnega računanja ter JOIN za sinhronizacijo. Ukaz FORK i povzroči nastanek novega zaporednega krmilnega toka s pričetkom pri ukazu z naslovom i. Ukaz na naslovu i ima tedaj na voljo že vse potrebne vhodne operande. Sočasni sprožitveni mehanizem v tem modelu ne zahteva preverjanja prisotnosti potrebnih operandov, saj je pričetek novega krmilnega toka eksplicitno določen z ukazom FORK. Krmilni tok, ki je izvršil ukaz FORK i, nadaljuje z izvrševanjem naslednjega ukaza (implicitni krmilni tok). Ukaz JOIN n zaustavi prvih n-1 krmilnih tokov, ki so prispeli do njega; ko prispe do tega ukaza zadnji, n-ti krmilni tok, nadaljuje z izvrševanjem naslednjega ukaza. Ker se v tem modelu sočasno vrši nekaj zaporednih krmilnih tokov, je njegov podatkovni mehanizem enak mehanizmu v modelu z zaporednim krmilnim tokom. Seveda zahteva tak model uporabo večjega števila procesorjev. Primer izračuna stavka z := (x+2)*(x-y) opisuje slika 3.



Slika 3. Računanje v modelu s sočasnim krmilnim tokom.

4.3. Tok krmilnih paketov

Model toka s krmilnimi paketi ima sledeče značilnosti:

- krmilno vodenje računanja,
- sočasni sprožitveni mehanizem,
- podatkovni mehanizem s pomočjo vstavljenih vrednosti in referenc.

Tudi pri modelu s tokom krmilnih paketov poteka računanje sočasno, zato ta model zahteva večje število procesorjev. Vsak ukaz hrani poleg operacije ter operandov še naslov svojega naslednika, ki bo uporabil njegov rezultat, ter prostore za krmilne pakete, ki jih prejme od svojih predhodnikov po njihovi izvršitvi. Ko ukaz zbere vse krmilne pakete, se sproži njegovo izvrševanje, pri čemer se najprej razrešijo vse reference ter nato izvrši operacija. Na koncu se rezultat shrani v skupno lokacijo, krmilni paket pa se posreduje vsem naslednikom ukaza. Vidimo, da je izvrševanje krmilno vodeno s sočasnim sprožitvenim mehanizmom, ki ugotavlja prisotnost potrebnih krmilnih paketov. Podatkovni mehanizem pa temelji na vstavljenih vrednostih in referencah. Potek računanja stavka z := (x+2)*(x-y) je opisan na sliki 4.

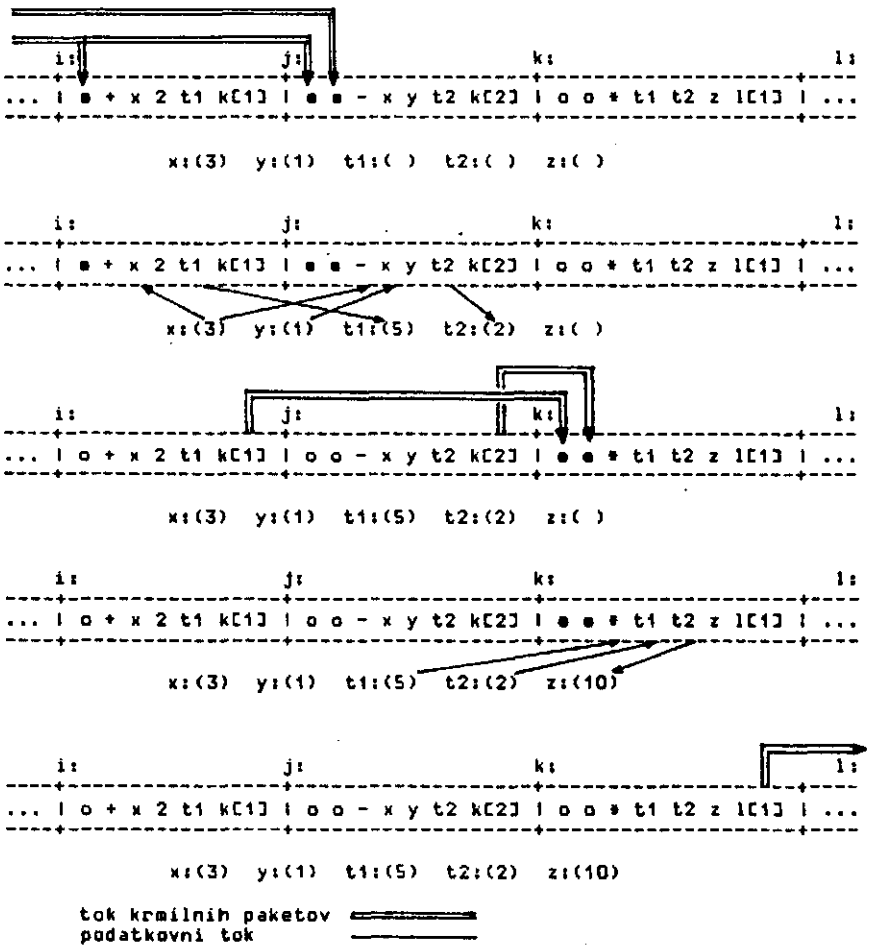
4.4. Tok podatkovnih paketov

Model s tokom podatkovnih paketov ima naslednje značilnosti:

- podatkovno vodenje računanja,
- sočasni sprožitveni mehanizem,
- podatkovni mehanizem s pomočjo vstavljenih in sprotnih vrednosti.

Ta model je podoben modelu s tokom krmilnih paketov, le da se tu izvrševanje odvija s pomočjo podatkovnih paketov. Ukazi sestojijo iz operacij, mest za operande ter oznak vseh ukazov, kamor naj se posreduje rezultat. Operandi so bodisi vstavljene ali pa sprotne vrednosti, s čimer je določen podatkovni mehanizem. Ukaz se lahko prične izvrševati takoj, ko mu je zadnji izmed njegovih neposrednih predhodnikov posredoval podatkovni paket s sprotno vrednostjo, na katero čaka. Izvrševanje je podatkovno vodeno, saj sočasni sprožitveni mehanizem v tem modelu ugotavlja prisotnost vseh potrebnih vrednosti vhodnih operandov.

V primeru toka podatkovnih paketov so programi navadno predstavljeni s pomočjo usmerjenih grafov, s katerimi je opisan tok podatkov med ukazi. Točke programskega grafa so ukazi,



Slika 4. Računanje v modelu s tokom krmilnih paketov.

povezave pa skrbijo za prenos vrednosti (podatkovnih paketov) med njimi. Točka se lahko prične izvrševati šele tedaj, ko so prisotni podatkovni paketi v vseh njenih vhodnih povezavah. Ob pričetku izvrševanja točka absorbira vse svoje vhodne podatkovne pakete (s čimer ti izginejo iz vhodnih povezav), po izvršitvi pa postavi rezultat (izhodni paket) istočasno v vse svoje izhodne povezave. Izhodni paketi točke so hkrati vhodni paketi neposrednih naslednikov.

Pri modelu računanju s tokom podatkovnih paketov je k vsakemu ukazu pridružen procesor, ki pasivno čaka na potrebne vhodne vrednosti. Faza izbiranja je torej kar pridružitev procesorjev vsem ukazom. Faza preverjanja sestoji iz ugotavljanja, če so prisotne vrednosti vseh vhodnih operandov. Če te vrednosti niso prisotne, ostane procesor neaktiven, v nasprotnem pa jih sprejme, preide v fazo izvrševanja ter zatem posreduje rezultate vsem svojim naslednikom. Izvršitev stavka z := (x+2)*(x-y) v tem modelu kaže slika 5.

4.5. Redukcija

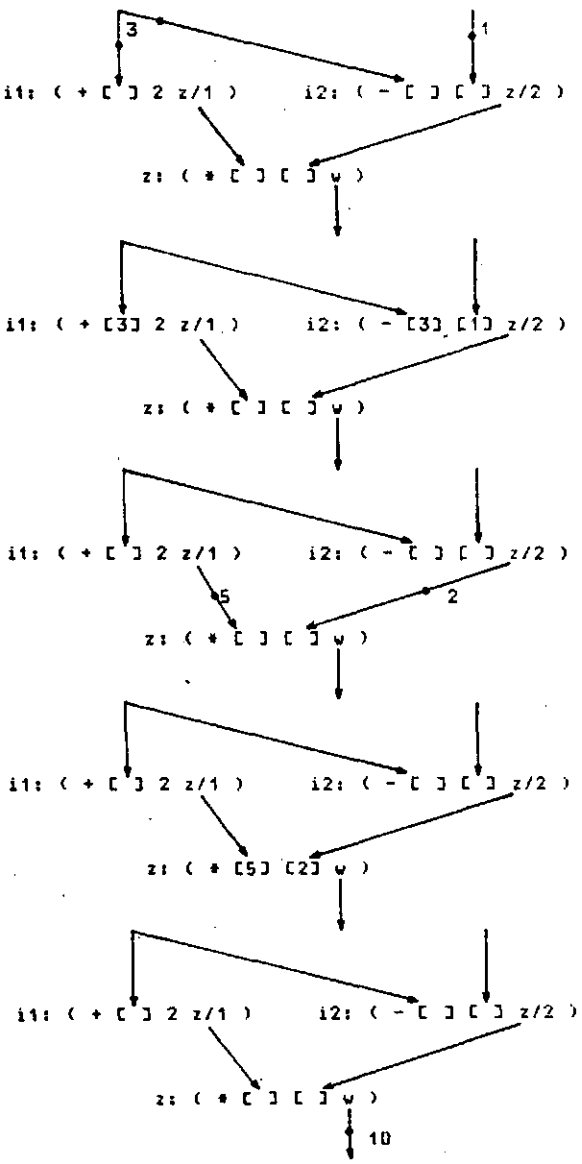
Krmilno in podatkovno vodeni modeli računanja uporabljajo ukaze, katerih velikost se ne spreminja. Modela računanja z redukcijo pa uporabljata gnezdene izraze, katerih velikost se med računanjem spreminja. Izvršitvi ukaza v krmilno ali podatkovno vodenem modelu ustreza pri redukciji uporaba /aplication/ funkcije nad njenimi argumenti. Programi v redukcijskem modelu so izrazi oblike <f><a1,...,aN>, kje so tako funkcija kot njeni argumenti bodisi atomi (npr. + ali 2) ali pa izrazi, sestavljeni iz atomov in izrazov. Uporaba funkcije je sestavljena iz dveh korakov:

- zahteve za dodelitev vrednosti njenim neznanim argumentom ter
- izračuna funkcije nad njimi.

Izračun funkcije se torej odloži, dokler niso dodeljene vrednosti vsem argumentom. Na primer, zahteva za dodelitev vrednosti argumentu a1, kjer je a1 definiran z a1 = <g><b1,...,bM>, sproži nadaljno uporabo funkcije g nad argumenti b1,...,bM. Takšne zahteve po uporabi funkcij lahko porodijo nove zahteve, s čimer je določen rekurziven sprožitveni mehanizem.

Očitno je, da mora biti vsak argument definiran z eno samo definicijo, kar imenujemo pravilo o enkratni prireditvi /single assignement rule/. Pravimo, da se dani argument sklicuje /reference/ na svojo definicijo. Poleg tega definicija vrne pri vsaki uporabi vedno enako vrednost kar imenujemo referenčna transparentnost redukcije.

Pri redukciji poteka faza izbiranja navadno



Slika 5. Računanje v modelu s tokom podatkovnih paketov.

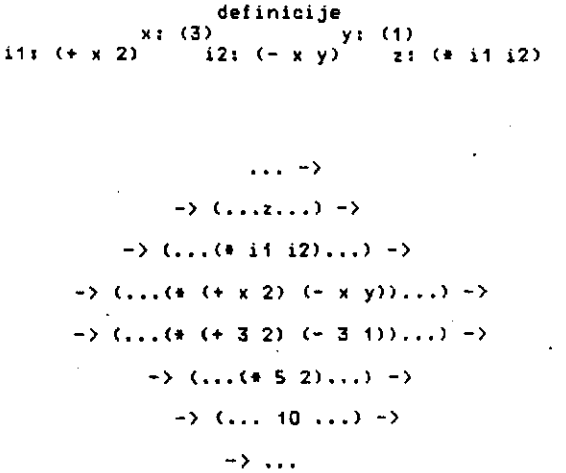
po pravilu zunanjega izbiranja. Procesorji se pridružijo ukazom v času faze izbiranja. Faza preverjanja pregleda argumente ukaza ter ugotovi, če je izračun možen. Če je, preide računanje v fazo izvrševanja, v kateri se ukaz izvrši ter nadomesti s svojim rezultatom. V nasprotnem primeru pa faza preverjanja izda zahtevo po nadomestitvi neznanih argumentov z znanimi vrednostmi. Takšna zahteva povzroči nastanek novih faz izbiranja, preverjanja in izvrševanja tistih ukazov, ki lahko vrnejo zahtevane vrednosti. Vsaka taka nova faza preverjanja seveda lahko porodi zopet nove faze računanja. Ko se vsa ta povzročena računanja končajo in vrnejo prvotno zahtevane vrednosti, začetni ukaz končno preide v fazo izvrševanja.

Ločimo dve vrsti redukcije, ki se razlikujeta po načinu obravnavanja argumentov. Pripadajoča modela računanja se imenujeta redukcija izrazov oziroma redukcija grafov.

4.5.1. Redukcija izrazov

Model z redukcijo izrazov ima sledeče značilnosti:
- vodenje računanja z zahtevo,
- rekurzivni sprožitveni mehanizem,
- podatkovni mehanizem s pomočjo vstavljenih in sprotnih vrednosti.

Zahteva za dodelitev vrednosti argumentom poteka tako, da se neatomarni argument nadomesti s kopijo definicije, na katero se sklicuje. Podatkovni mehanizem zato temelji na vstavljenih in sprotnih vrednostih. Če funkcija vsebuje več neatomarnih argumentov, se vsi nadomeste sočasno. Če so argumenti funkcije atomarni, se funkcija izračuna in nadomesti z rezultatom. Izračuni funkcij potekajo sočasno. V tem modelu je zelo pomembno dejstvo, da se postopek nadomeščanja izvrši vsakokrat, ko se zahteva izračun argumenta. Primer redukcije izrazov pri izvrševanju stavka z := (x+2)*(x-y) je opisan na sliki 6.

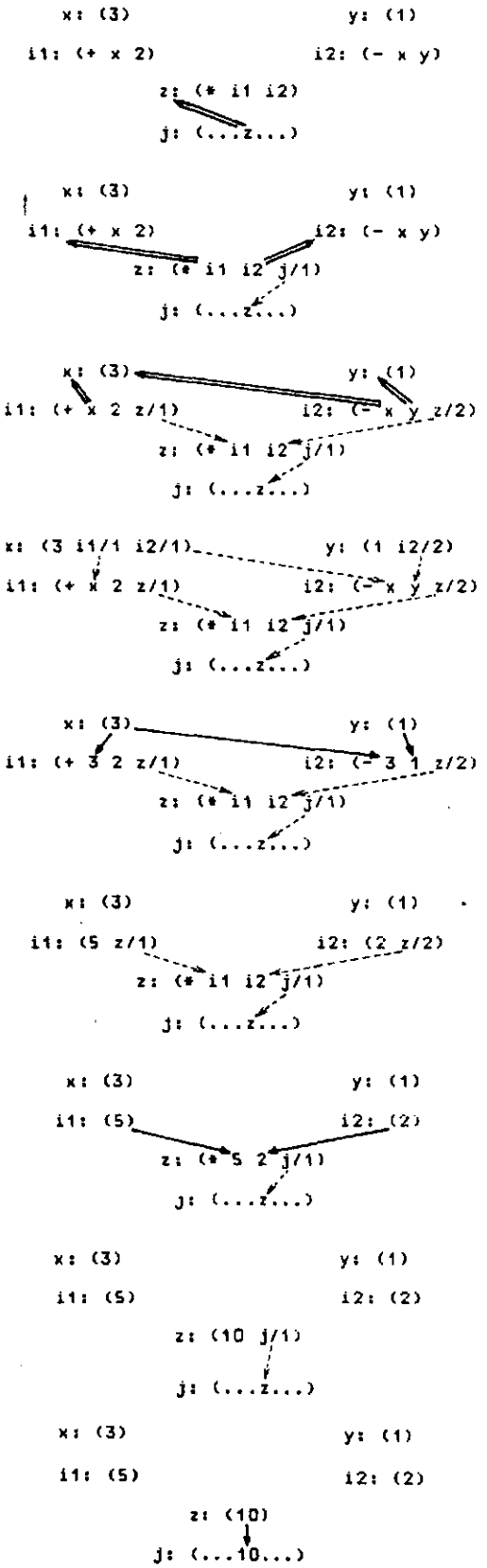


Slika 6. Računanje v modelu z redukcijo izrazov.

4.5.2. Redukcija grafov

Model z redukcijo grafov pa ima sledeče značilnosti:
- vodenje računanja z zahtevo,
- rekurzivni sprožitveni mehanizem,
- podatkovni mehanizem s pomočjo vstavljenih vrednosti in referenc.

Pri redukciji grafa se definiciji, na katero se sklicuje argument, doda oznaka tega argumenta. To pomeni, da podatkovni mehanizem temelji na vstavljenih vrednostih in referencah. Po uporabi funkcije v tej definiciji se njena vrednost posreduje čakajočemu argumentu, hkrati pa se tudi definicija nadomesti s to vrednostjo. Ob kasnejšem sklicevanju na tako definicijo se takoj vrne njena vrednost. To je temeljna prednost redukcije grafa napram redukciji izrazov. Primer izračuna stavka z := (x+2)*(x-y) je opisan na sliki 7.



Slika 7. Računanje v modelu z redukcijo grafov.

4.6. Povzetek lastnosti modelov računanja

Zaporedni in sočasni krmilni tok imata sledeče skupne značilnosti:
- prenos podatkov med ukazi poteka preko referenc na skupne pomnilniške lokacije,
- vrednosti operandov so lahko shranjene v samem ukazu (vstavljene vrednosti); s tem se pohitri njegova izvršitev,
- krmilni tok je po svoji naravi zaporeden, vendar lahko kreiramo sočasne krmilne tokove z uporabo posebnih ukazov FORK in JOIN,
- krmilni tok in tok podatkov sta ločena,
- potek računanja je popolnoma določen s krmilnim tokom.

Lastnosti računanja s tokom podatkovnih paketov so:
- sprotne vrednosti se posredujejo med ukazi neposredno, v obliki podatkovnih paketov,
- možna je uporaba vstavljenih vrednosti,
- ob pričetku izvrševanja prevzame ukaz svoje vhodne pakete, s čimer ti izginejo iz vhodnih povezav, zato jih kasneje ne more več uporabiti,
- pri izvrševanju se ne uporabljajo nikakršne skupne pomnilniške lokacije za prenos podatkov, kot je to primer pri krmilnem vodenju,
- izvrševanja programa je popolnoma določeno s tokom podatkov.

Redukcija pa ima sledeče značilnosti:
- vse programske strukture, funkcije in argumenti so gnezdeni izrazi,
- pri prenašanju vrednosti se ne uporabljajo nikakršne skupne pomnilniške lokacije,
- izvršitev ukaza vrne bodisi atomaren ali pa sestavljen izraz,
- po izvršitvi definicije se le-ta lahko nadomesti z izračunano vrednostjo,
- potek računanja je popolnoma definiran s tokom zahtev po izvršitvah.

Vse opisane modele računanja in njihov odnos glede na način vodenja opisuje tabela 2.

NAČINI VODENJA

MODELI RAČUNANJA	krmilno	podatkovno	z zahtevo
MAORDUENLAINJA	zaporedni krmilni tok	tok podatkovnih paketov	redukcija izrazov
	sočasni krmilni tok		redukcija grafov
	tok krmilnih paketov		

Tabela 2. Načini vodenja in ustrezni modeli računanja.

5. Organizacije stroja

Kot je bilo uvodoma omenjeno, pripadajo načinom vodenja računanja ustrezne družine računalniških arhitektur, ki se imenujejo krmilno vodene arhitekture, podatkovno vodene arhitekture ter arhitekture, vodene z zahtevo. Omenjene družine izvirajo iz načinov vodenja računanja, vendar pa lahko posamezne družine razdelimo na poddružine glede na izbrani model računanja ter organizacijo stroja /machine organization/. Pod izrazom organizacija stroja razumemo način sestavljanja in medsebojnega povezovanja osnovnih računalniških materialnih

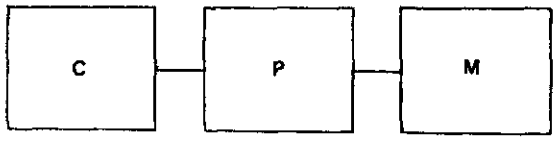
virov /resources/, kot so procesorji, pomnilniki in komunikacijske enote.

Razlikujemo tri osnovne organizacije stroja in sicer:
- centralizirano organizacijo /centralized organization/,
- organizacijo za posredovanje paketov /package communication organization/,
- organizacijo za obravnavo izrazov /expression manipulation organization/.

V nadaljevanju bomo najprej opisali vse tri omenjene organizacije stroja.

5.1. Centralizirana organizacija

Centralizirano organizacijo sestavljajo en procesor, pomnilnik ter vodilo med njima (Slika 8). Takšna organizacija ima v danem trenutku v programu en sam izvršujoč se ukaz. Po izvršitvi takšnega ukaza se prične izvrševanje njegovega natančno določenega naslednika, izbranega s programskim števcem. To je v resnici tradicionalna von Neumannova arhitektura.



P ... procesor
C ... komunikacijska enota
M ... pomnilnik

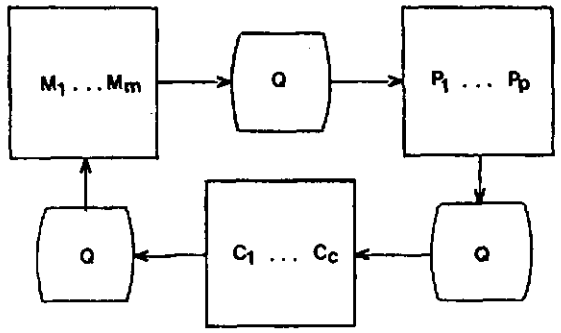
Slika 8: Centralizirana organizacija stroja.

5.2. Organizacija za posredovanje paketov

Organizacijo sestavljajo tri glavne enote: procesna enota, ki združuje večje število procesorjev, pomnilniška enota ter komunikacijska enota. Enote so krožno povezane, kot to prikazuje slika 9. Med enotami se nahajajo še vrste /pools of work/, ki opravljajo nalogo začasnega skladiščenja paketov, kadar je to potrebno. Izvrševanje programa v takšni organizaciji poteka v obliki enosmernega krožnega toka paketov skozi njene enote. Enote delujejo sočasno, torej je izvrševanje programa cevljeno /pipelined/. Paketi, ki so pripravljeni za obdelavo v eni izmed enot, čakajo v ustrezni vrsti, vse dokler jih ustrezna enota ni pripravljena sprejeti. Ko jih ta enota obdela, shrani tako spremenjene pakete v vrsto pred naslednjo enoto.

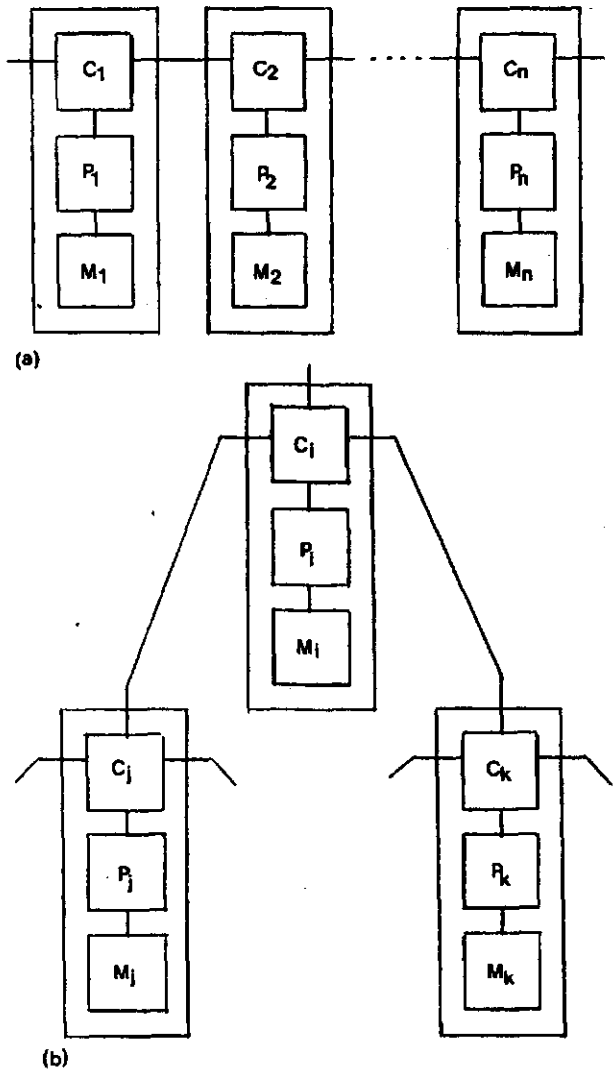
5.3. Organizacija za obravnavo izrazov

Takšno organizacijo sestavljajo enaki osnovni gradniki, med seboj povezani v pravilno strukturo /regular structure/. Vsak gradnik sestavljajo tri enote: procesor, pomnilnik in komunikacijska enota. Slednja skrbi za povezavo med procesorjem, pomnilnikom ter sosednjimi gradniki. Dve značilni strukturi takšne organizacije, vektorsko in drevesno, prikazuje slika 10. V takšni organizaciji se obravnava program kot gnezden izraz. Podizrazi so pridruženi gradnikom, pri čemer se podizraz pridruži tistemu gradniku, ki zrcali lego podizraza v izrazu. Med izvrševanjem vsak gradnik pregleda svoj pridruženi podizraz ter ugotovi, kaj mu je storiti.



P ... procesor
C ... komunikacijska enota
M ... pomnilnik
Q ... vrsta

Slika 9. Organizacija stroja za posredovanje paketov.



P ... procesor
C ... komunikacijska enota
M ... pomnilnik

Slika 10. Organizacija stroja za obravnavo izrazov; a) vektorska, b) drevesna.

6. Arhitekture

V tem poglavju bomo pokazali, kako lahko različne organizacije stroja podpirajo posamezne modele računanja. Povedano drugače, tri osnovne družine arhitektur bomo razdelili na podružine, kot je prikazano v tabeli 3.

družine arhitektur	model računanja	organizacija stroja
krmilno vodene	zaporedni krmilni tok	centralizirana
	sočasni krmilni tok	obravnava izrazov
	tok krmilnih paketov	posredovanje paketov
podatkovno vodene	tok podatkovnih paketov	posredovanje paketov
	tok podatkovnih paketov	obravnava izrazov
vodene z zahtevo	redukcija izrazov	obravnava izrazov
	redukcija grafov	obravnava izrazov
	redukcija izrazov	centralizirana
	redukcija grafov	posredovanje paketov

Tabela 3. Razdelitev računalniških arhitektur.

6.1. Krmilno vodene arhitekture

6.1.1. Zaporedni krmilni tok na centralizirani organizaciji

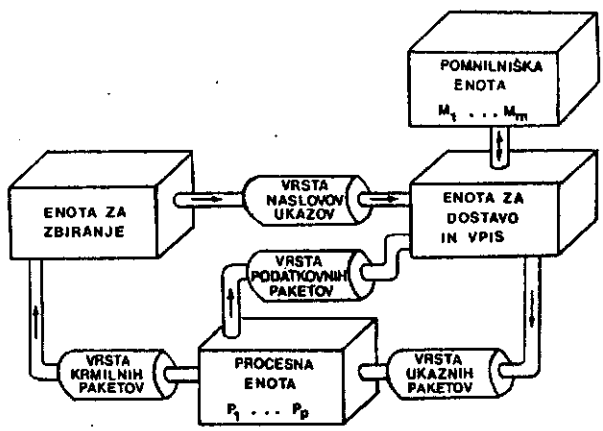
Najnaravnejši kandidat za podporo zaporednega krmilnega toka je centralizirana organizacija stroja. Krmilni tok določa programski števec v procesorju, ki zaporedno naslavlja ukaze, ki so na vrsti za izvršitev. Procesor takšne ukaze drugega za drugim, tj. zaporedno, izvršuje. Ker ta zveza določa znano von Neumannovo arhitekturo, je na tem mestu ne bomo podrobneje opisovali.

6.1.2. Sočasni krmilni tok na organizaciji za obravnavo izrazov

Sočasni krmilni tok, pri katerem uporabljamo ukaza FORK in JOIN, podpira organizacija za obravnavo izrazov. Zaporedni krmilni tokovi, ki se vršijo sočasno, potekajo v različnih gradnikih te organizacije. Krmilni tok, ki poteka v enem gradniku, lahko porodi nov krmilni tok v nekem drugem gradniku. Ko gradnik A prične izvrševati ukaz FORK i, sporoči (preko komunikacijske enote) tistemu gradniku B, v katerega pomnilniku se nahaja ukaz z naslovom i, naj prične z izvrševanjem svojega programa pri tem ukazu. V primeru, ko v B že poteka nek krmilni tok, A odloži izvršitev operacije FORK i, dokler se krmilni tok v B ne izteče. Tedaj se lahko izvrši ukaz FORK i, torej se prične krmilni tok v B pri ukazu i, gradnik A pa lahko nadaljuje izvrševanje, narekovano z lastnim krmilnim tokom. Gradnik, v katerem je krmilni tok prispel do ukaza JOIN n, se zaustavi vse dotlej, dokler se v n-1 gradnikih ne izteče zadnji med porojenimi krmilnimi tokovi.

6.1.3. Tok krmilnih paketov na organizaciji za posredovanje paketov

Krmilno voden model računanja s tokom krmilnih paketov pa je moč realizirati na organizaciji za posredovanje paketov. Arhitektura ima zgradbo, kot jo prikazuje slika 11.



Slika 11. Tok krmilnih paketov na organizaciji za posredovanje paketov

Program se nahaja v pomnilniški enoti /memory unit/. Enota za zbiranje /matching unit/ hrani za vsak ukaz programa:
- množico doslej zbranih krmilnih paketov ukaza
- naslov ukaza v pomnilniški enoti.

Ko enota za zbiranje zbere vse potrebne krmilne pakete danega ukaza, pošlje njegov naslov preko vrste naslovov ukazov /instruction address pool/ v enoto za dostavo in vpis /fetch & update unit/. Ta enota prenese iz pomnilniške enote naslovljeni ukaz, razreši reference ter sestavi ukazni paket. Le-ta vsebuje operacijo, vrednosti operandov ter naslove ukazov, ki pričakujejo rezultat. Ukazni paket se preko vrste ukaznih paketov /executable instructions pool/ prenese v procesno enoto. Ko se ukaz izvrši, se sestavijo podatkovni ter krmilni paketi. Podatkovni paketi nosijo rezultat ter naslov ukaza, ki čaka nanj in se preko vrste podatkovnih paketov /data pool/ pošljejo v enoto za dostavo in vpis, ki rezultate vpiše v naslovljene lokacije pomnilniške enote. Krmilni paketi pa se preko vrste krmilnih paketov /control token pool/ posredujejo enoti za zbiranje.

6.2. Podatkovno vodene arhitekture

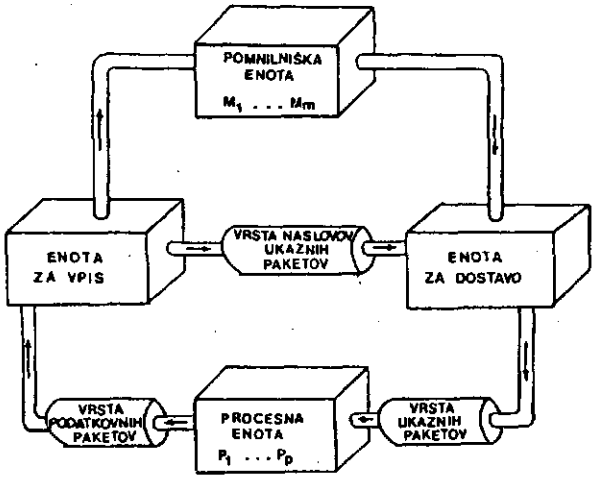
6.2.1. Tok podatkovnih paketov na organizaciji za posredovanje paketov

Model računanja s tokom podatkovnih paketov običajno realiziramo na organizaciji za posredovanje paketov. Takšno arhitekturo imenujemo podatkovno pretokovna arhitektura /data flow architecture/. Ločimo dve vrsti podatkovno pretokovnih arhitektur:

- arhitektura s shranjevanjem paketov /token store/,
- arhitektura z zbiranjem paketov /token matching/.

Prikazani sta na sliki 12 in sliki 13.

6.2.1.1. Podatkovno pretokovna arhitektura s shranjevanjem paketov



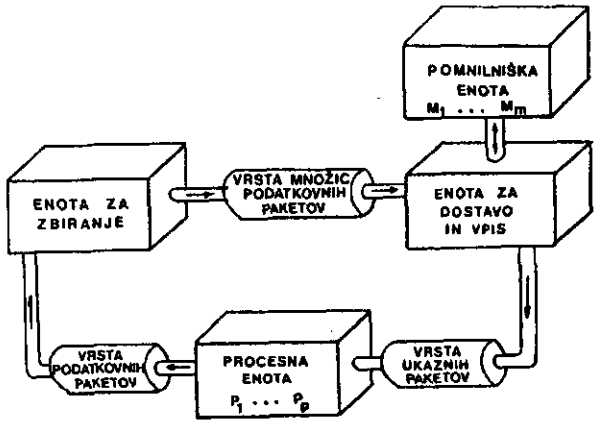
Slika 12. Podatkovno pretokovna arhitektura s shranjevanjem paketov.

Pomnilniška enota /memory unit/ vsebuje ukazne pakete, ki vsebujejo operacijo, prostore za operande ter naslove na njegov rezultat čakajočih ukaznih paketov. Enota za vpis /update unit/ hrani naslov vsakega ukaznega paketa ter število operandov, ki jih mora še prejeti, da bo postal izvršljiv. Pri tej arhitekturi se podatkovni paketi, ki jih sestavi procesna enota /processing unit/, posredujejo preko vrste s podatkovnimi paketi /data token pool/ v enoto za vpis. Vsak podatkovni paket vsebuje poleg rezultata tudi naslov čakajočega ukaznega paketa v pomnilniški enoti. Enota za vpis shrani rezultat prispelega podatkovnega paketa v čakajoči ukazni paket v pomnilniški enoti ter zmanjša število operandov, ki jih mora ta paket še prejeti. Če enota za vpis ugotovi, da je ukazni paket prejel zadnjega izmed potrebnih operandov, pošlje naslov paketa preko vrste naslovov ukaznih paketov /instruction address pool/ enoti za dostavo /fetch unit/. Ta enota nato takoj prenese ukazni paket preko vrste ukaznih paketov /executable instruction pool/ v procesno enoto, kjer se ukaz izvrši. Podatkovni paketi, ki so rezultat izršitve ukaza, se nato zopet posredujejo enoti za vpis.

Opomba: Izvrševanje, pri katerem enota za dostavo prenese izvršljiv ukazni paket v procesno enoto takoj, ko iz enote za vpis prejme njegov naslov, imenujemo takojšnje izvrševanje. S predhodno analizo programskega grafa pa lahko dobimo določeno informacijo, ki služi enoti za dostavo pri izbiranju in prenašanju izvršljivih paketov v procesno enoto. Enota za dostavo lahko tedaj počaka s prenosom na trenutek, ki je najbolj ugoden. S tem je lahko omogočeno izvrševanje dosti 'večjih' programov, ne da bi se minimalni možni čas, potreben za njihovo izvrševanje, podaljšal. Omenjena analiza je podrobno opisana v [Robi86a, Robi86b].

Projekti: Podatkovno pretokovno arhitekturo s shranjevanjem paketov so realizirali na MIT-ju pod vodstvom J.B. Dennisa [Denn83]. Druga realizacija te arhitekture je DDP (Distributed Data Processor), izdelan v Texas Instruments [Corn79]. Zanimivo je, da podpira DDP programirni jezik Fortran 66. Omenimo še francoski projekt LAU (Language à Assignation Unique), ki poteka v CERT laboratoriju, Toulouse [Comt79, Comt80].

6.2.1.2. Podatkovno pretokovna arhitektura z zbiranjem paketov



Slika 13. Podatkovno pretokovna arhitektura z zbiranjem paketov

Pri tej arhitekturi se podatkovni paketi, ki so prispeli iz procesne enote, ne shranijo neposredno v ukazni paket, kateremu so namenjeni. Namesto tega se podatkovni paketi nalagajo v množice, ki se nahajajo v enoti za zbiranje /matching unit/ paketov. Vsaka množica je preko pridruženega naslova prirejena ukaznemu paketu, ki se nahaja v pomnilniški enoti /memory unit/. Ko se množica napolni (prejme zadnji potrebni podatkovni paket), se preko vrste množic podatkovnih paketov /token sets pool/ prenese v enoto za dostavo in vpis /fetch & update unit/. Ta enota prenese iz pomnilniške enote kopijo naslovljenega ukaznega paketa, ga dopolni s prispelo množico podatkovnih paketov, ter ga preko vrste ukaznih paketov /executable instruction pool/ pošlje v procesno enoto /processing unit/. Tu se ukaz izvrši, podatkovni paketi, ki so rezultati izvršitve, pa se zopet prenesejo v enoto za zbiranje paketov.

Opomba: Podobno kot v prejšnji arhitekturi, lahko tudi tu izvršljive ukazne pakete zadržujemo, le da v tem primeru vrši zadrževanje enota za zbiranje.

Projekti: Realiziranih je več projektov, med katerimi omenimo Manchester Data Flow Computer, ki so ga izdelali pod vodstvom I. Watsona in J.R. Gurda na univerzi v Manchestru [Wats79, Gurd 83, Gurd85]. Razširjena različica EXMAN (EXtended MANchester Data Flow Computer) je predlagana v [Patn86]. Tretji projekt poteka na univerzi v Newcastlu, kjer gradijo računalnik JUMBO [Trel82a]. Omenimo še Id (Irvin Dataflow Machine) projekt, začet na kalifornijski univerzi v Irvinu, ki se trenutno nadaljuje na MIT-ju [Arvi80] in ga vodi Arvind.

6.2.2. Tok podatkovnih paketov na organizaciji za obravnavo izrazov

Tok podatkovnih paketov je moč realizirati tudi na organizaciji za obravnavo izrazov. Ko procesor v nekem gradniku prejme podatkovni paket preko komunikacijske enote iz drugega gradnika, vstavi vrednost, ki jo nosi paket v ukaz, kateremu je namenjena. V naslednjem koraku procesor ugotovi, če je ukaz izvršljiv, ter ga v tem primeru izvrši ter pošlje podatkovni paket, v katerem se nahaja rezultat, preko komunikacijske enote tistemu gradniku, v katerem se nahaja ukaz, ki ta rezultat pričakuje. Če pa ukaz še ni izvršljiv, se procesor povrne v stanje čakanja.

Projekti: Data-Driven Machine #1 (DDM1) je primer arhitekture, ki temelji na toku podatkovnih paketov, realiziranem na drevesni organizaciji za obravnavo izrazov [Davi78]. DDM1 so izdelali pod vodstvom A. Davisa na Burroughs Interactive Research Center, La Jolla, California.

6.3. Z zahtevo vodene arhitekture

6.3.1. Redukcija izrazov na organizaciji za obravnavo izrazov

Model z redukcijo izrazov je naravno podprt z organizacijo za obravnavo izrazov [Trel82b]. Kot smo že omenili, so programi v takšnem modelu računanja gnezdeni izrazi. Glavni gnezdeni izraz je porazdeljen po pomnilniških enotah gradnikov organizacije tako, da sta sosedna podizraza vpisana v pomnilniških enotah dveh sosednih gradnikov. Med izvrševanjem se izraz bodisi širi (pri vstavljanju definicij, na katere se operandi sklicujejo) ali pa krči (pri nadomestitvah izračunljivih podizrazov z njihovimi rezultati). Ko procesor pregleda podizraz v lastni pomnilniški enoti ter najde v njem nek še neizračunan operand, zahteva preko komunikacijske enote prenos definicije, na katero se operand sklicuje. Po prihodu zahtevane definicije (preko komunikacijske enote) se povzroči premik ostalega izraza vstran, s čimer se napravi prostor za vstavitev prispele definicije. Pri takšnih premikih lahko seveda preidejo podizrazi iz ene pomnilniške enote v drugo (v drug gradnik). Če pa procesor ugotovi, da je podizraz v njegovi pomnilniški enoti reducibilen (da so vsi operandi operacije znani), ga izračuna ter nadomesti z rezultatom. Posledica takšne redukcije je lahko krčenje celotnega izraza ter s tem premik delov izraza iz ene pomnilniške enote v drugo. Funkcija, ki jo pri računanju opravlja zaporedje pomnilniških enot je torej podobna funkciji dvosmernega premikalnega registra. Ker ima lahko nek podizraz več operandov, ki se sklicujejo na svoje definicije, in ker je ugodno, če se takšni operandi nadomeste z njimi sočasno (kar pa se lahko izvrši le v različnih gradnikih), morajo biti kapacitete pomnilniških enot primerno majhne. Visoka stopnja sočasnosti pri računanju je torej zagotovljena ob zadostnem številu gradnikov ter primerno majhni kapaciteti pomnilniških enot.

Projekti: Omenimo dve realizirani arhitekturi in sicer Newcastle Reduction Machine [Trel80] ter North Carolina Cellular Tree Machine [Mago80], ki ima drevesno organizacijo stroja.

6.3.2. Redukcija grafov na organizaciji za obravnavo izrazov

Tudi model z redukcijo grafov je moč realizirati na organizaciji za obravnavo izrazov [Trau85]. Graf se pred izvrševanjem smiselno porazdeli na gradnike (pomnilnike). Izračuni definicij potekajo v gradnikih, kjer so definicije argumentov, rezultati pa se posredujejo gradnikom, ki so posredovali zahtevo. Zato gradnik skupaj z zahtevo pošlje tudi svoj naslov, ki se vpiše v definicijo argumenta.

Za realizacijo te arhitekture je primernejša organizacija stroja, kjer vsebujejo gradniki le procesor in komunikacijsko enoto, pomnilnik pa je združen. Deli pomnilnika so bodisi vnaprej dodeljeni gradnikom, ali pa je celoten pomnilnik skupen za vse gradnike.

6.3.3. Redukcija grafov na organizaciji za posredovanje paketov

Pri redukciji je možno uporabiti organizacijo za posredovanje paketov le v modelu z redukcijo grafov. Izvrševanje programa na takšni organizaciji poteka z dvema vrstama paketov:

- paketi z zahtevo /demand token/,
- paketi z rezultatom /result token/.

Paket z zahtevo vsebuje:

- naslov ukaza, ki lahko izračuna zahtevano vrednost,
- naslov ukaza, ki je zahteval to vrednost.

Paket z rezultatom pa vsebuje:

- rezultat izvršitve ukaza,
- naslov ukaza, ki je ta rezultat zahteval.

Če operandi ukaza, katerega izvršitev je bila predhodno zahtevana, še niso znani, pošlje tak ukaz pakete z zahtevo vsem tistim ukazom, ki lahko te vrednosti izračunajo. Ko ukaz sprejme paket z zahtevo, se izvrši (pred tem se lahko postopek pošiljanja paketov z zahtevami ponovi), ter pošlje paket z rezultatom ukazu, ki ga je zahteval. Tedaj se lahko tudi ta ukaz izvrši. Sočasni sprožitveni mehanizem je določen z dvema pravilomi:

- za sprožitev ukaza je potreben natanko en paket z zahtevo,
- za izračun ukaza morajo biti na voljo vsi potrebni paketi z rezultati.

Projekti: Značilen predstavnik je AMPS (Utah Applicative Multiprocessing System), zgrajen na univerzi v Utahu [Kell79]. Omenimo tudi projekta na Imperial College v Londonu [Darl81] ter na univerzi East Anglia [Slee81].

6.3.4. Redukcija izrazov na centralizirani organizaciji

Pri realizaciji modela z redukcijo izrazov imajo enote v centralizirani organizaciji sledeče značilnosti:

- pomnilniška enota združuje določeno število skladov /stack/, med katerimi se ponavljajoče prenaša izraz,
- procesna enota skrbi za prenos trenutnega izraza iz začetnega /source/ sklada v končni /sink/ sklad. Pri prenašanju izraza procesna enota razpoznava njegove reducibilne podizraze, jih sproti izračunava (reducira) ter namesto njih shranjuje v končni sklad njihove rezultate. Po prenosu celega izraza prevzame končni sklad naloge začetnega in obratno. Prenašanje se ponavlja vse dokler ni izraz atomaren,

- komunikacijska enota je vodilo, po katerem se prenašajo izrazi iz začetnega sklada v procesno enoto in iz te v končni sklad.

Vsak izraz (in s tem tudi glavni izraz - program) je zapisan v prefiksni obliki, pri čemer so v taki obliki zapisani tudi vsi njegovi podizrazi. Pri prenosu izraza iz začetnega sklada v končni se mora takšna oblika ohraniti, zato navadno prenos poteka preko dodatnega, vmesnega /intermediate/ sklada.

Projekti: Primer takčne arhitekture je GMD (Gesellschaft für Mathematik und Datenverarbeitung), ki so jo realizirali v Bonnu, ZRN [Klug79].

7. Zaključek

Predstavljena razdelitev novogeneracijskih arhitektur je le ena iz vrste možnih razdelitev. Kot je že bilo omenjeno, je pri tej razdelitvi osnovno vodilo vodenje računanja. Marsikatera realizirana arhitektura težko najde mesto v eni sami predlagani poddružini (poglavje 6), saj je v novejših arhiteturah pričakovati različne kombinacije podatkovnega in sprožitvenega mehanizma, s tem pa tudi mešanih modelov računanja ter vodenja. Omenimo vzorčno vodenje /pattern driven/ računanja, kjer se ukaz prične izvrševati, ko pride do ujemanja vzorca, ki je pogoj za pričetek izvrševanja [Trel84]. Pojavljajo se tudi kompleksnejše organizacije stroja. Kot ilustracijo navedimo le dva najnovejša podatkovno pretokovna računalnika: SIGMA-1 [Shim86] in večprocesorski podatkovno pretokovni sistem na osnovi procesorjev μPD7281 [Jeff85, Silc86]. V obeh primerih gre za organizacijo stroja, ki je podobna organizaciji za obravnavo izrazov, le da so gradniki podatkovno pretokovni procesorji, katerih organizacija temelji na posredovanju paketov.

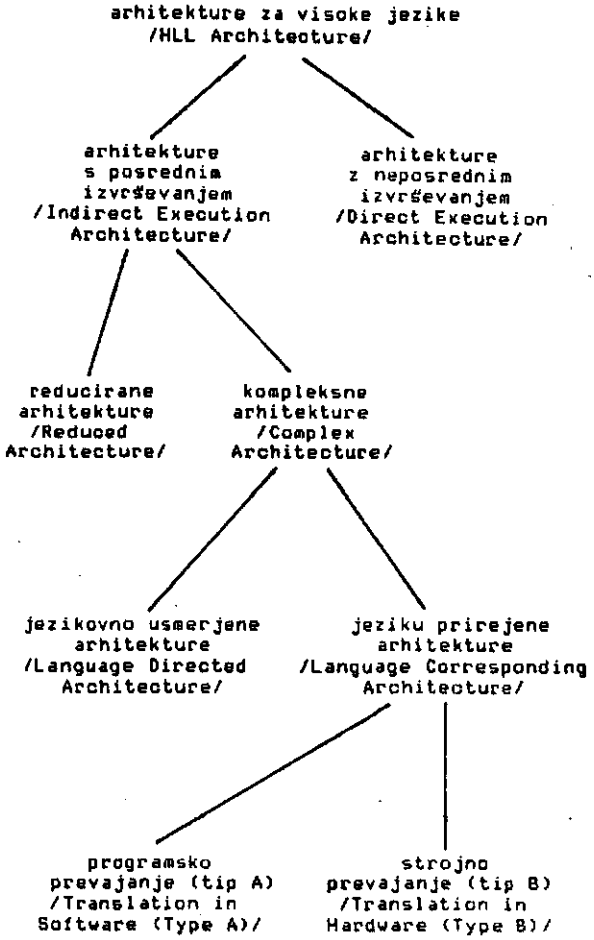
Računalniške arhitekture je moč razdeliti tudi z dveh drugih, diametralnih zornih kotov: z vidika jezikov oziroma strojne opreme.

Tako obstaja razdelitev novejših računalniških arhitektur, temelječih na visokih programirnih jezikih /high level language computer architectures/. Takšna razdelitev je podana v [Milu86] in jo prikazuje slika 14.

Čeprav ni namen članka predstavitev takšne razdelitve arhitektur, pa vendarle omenimo skupino reduciranih arhitektur, med katere sodijo IBM 801, RISC (UC Berkely), MIPS (Stanford), RIMMS (univerza Reading), Transputer (Inmos) ter VM (Purdue). Osnova teh arhitektur je strojno realiziran nabor najpogosteje prisotnih ukazov v prevedeni kodi programov (RISC nabor). Redkeje uporabljani ukazi se nadomestijo z ustreznimi zaporedji ukazov iz RISC nabora [Patt85].

Za konec omenimo še dve možni delitvi arhitektur z drugega zornega kota, to je na osnovi strojne opreme. Morda najpogostejša delitev računalniških arhitektur je po Flynnu [Flyn72], ki jih deli v štiri skupine:

- SISD (en ukazni tok in en podatkovni tok) /single instruction stream & single data stream/,
- SIMD (en ukazni tok in več podatkovnih tokov) /single instruction stream & multiple data stream/,
- MISD (več ukaznih tokov in en podatkovni tok) /multiple instruction stream & single data stream/ ter
- MIMD (več ukaznih tokov in več podatkovnih tokov) /multiple instruction stream & multiple data stream/.



Slika 14. Razdelitev računalniških arhitektur, temelječih na visokih prog. jezikih.

Shoreova delitev [Shor73] pa obsega šest razredov računalniških arhitektur glede na združevanje materialnih virov, kot so procesne enote PE /processing units/, krmilne enote KE /control units/, ukazni pomnilniki UP /instruction memories/ in podatkovni pomnilniki PP /data memories/. Teh šest razredov je:

- stroj I: so klasične von Neumannove arhitekture, z možnostjo cevljenja v PE,
- stroj II: je podoben stroju I, le da se sočasno dostavijo bitne rezine /bit slice/ vseh besed iz PP; torej v nasprotju s strojem I, stroj II obdeluje besede 'prečno' in ne 'vzdolžno',
- stroj III: je kombinacija prvih dveh strojev, zato lahko obdeluje besede bodisi 'prečno' ali 'vzdolžno',
- stroj IV: je arhitektura, ki preko ene KE združuje množico parov PE in PP, kjer PE-ji nimajo možnosti medsebojne komunikacije,
- stroj V: je podoben stroju IV, pri čemer lahko vsak par PE in PP komunicira s svojima sosednima paroma, ter
- stroj VI: za razliko od strojev I do V, kjer je PE ločen od PP, so tu funkcije PE in PP združene v eni sami enoti /logic-in-memory array/.

8. Ključne besede

- podatkovnih paketov /data token */, 6.1.3, 6.2.1.1, 6.2.1.2
- ukaznih paketov /executable instruction */, 6.1.3, 6.2.1.1, 6.2.1.2

Pripis: Pri prevajanju angleških izrazov sva nekajkrat ubrala nekoliko svobodnejšo pot, predvsem takrat, ko bi dobesedni prevod slabo opisoval dotični predmet. Upava, da sva imela pri izbiri slovenskih izrazov srečno roko.

9. Literatura

[Arvi80] Arvind, Kathail V., Pingali K. "A Processing Element for a Large Multiprocessor Dataflow Machine", Proc. Int'l. Conf. Circuits and Computers, New York, Oct. 1980.

[Back78] Backus J. "Can programming be liberated from the von Neumann style? A functional style and its algebra of programms", Comm. ACM Vol.21, No.8, Aug. 1978, pp.613-641.

[Berk71] Berkling K.J. "A computing machine based on tree structure", IEEE Trans. Computer, Vol.C-20, No.4, Jan. 1971, pp.404-418.

[Bish86] Bishop P. Fifth Generation Computers, Concepts, implementations and uses, Ellis Horwood, 1986.

[Braj86] Brajak P. "Paralelno procesiranje: arhitektura 90-tih godina, Zbornik jugoslavenskog savjetovanja o novim generacijama računala, MIPRO 86, Rijeka, Maj 1986, str.33-46.

[Corn79] Cornish M. "The TI Data Flow Architectures: The Power of Concurrency for Avionics", Proc. 3rd Conf. Digital Avionics Systems, Fort Worth, Texas, Nov. 79, pp.19-25.

[Comt79] Comte D., Hifdi N. "LAU Multiprocessor: Microfunctional Description and Technological Choices", Proc. 1st European Conf. Parallel and Distributed Processing, Toulouse, France, Feb. 1979, pp.8-15.

[Comt80] Comte D., Hifdi N., Syre J.C. "The Data Driven LAU Multiprocessor System: Results and Perspectives", Proc. IFIP 80 Congress, Oct. 1980.

[Davi78] Davis A.L. "The Architecture and System Method of DDM1: A Recursively Structured Data Driven Machine", Proc. 5th Ann. Symp. Comp. Arch., Palo Alto, Calif., April 3-5, 1978, pp.210-215.

[Davi82] Davis A.L., Keller R.M. "Data Flow Program Graphs", Computer, Vol.15, No.2, Feb. 1982, pp.26-41.

[Darl81] Darling J., Reeve M. "ALICE: A Multiprocessor Reduction Machine for the Parallel Evaluation of Applicative Languages", Proc. Int'l Symp. Functional Programming Languages and Computer Architecture, June 1981, Göteborg, Sweden, pp.32-62.

[Denn74] Dennis,J.B., Misunas,D.P. "A Computer Architecture for highly parallel signal processing", Proc.1974 Nat. Computer Conf., AFIPS Press, Arlington, Va.,1974,pp.402-409.

[Denn83] Dennis J.B., Lim W.Y-P., Ackerman W.B. "The MIT Data Flow Engineering Model", Information Processing 83, R.E.A. Mason (ed.), Elsevier Scince Publishers B.V. (North-Holland), 1983.

[Flyn72] Flynn M.J. "Some Computer Organizations and their Effectiveness", IEEE Trans. Comp. Vol.C-21, No.9, Sept. 1972, pp.948-960.

[Gilo83] Giloi W.K. "Towards a Taxonomy of Computer Architecture Based on the Machine Data Type View", The 10th Ann. Int'l Symp. Comp. Arch., June 13-17, 1983, Stockholm, Sweden, pp.6-13.

[Gurd83] Gurd J.R., Watson I. "Preliminary Evaluation of a Prototype Dataflow Computer", Information Processing 83, R.E.A. Mason (ed.), Elsevier Scince Publishers B.V. (North-Holland), 1983.

[Gurd85] Gurd J.R., Kirkham C.C., Watson I. "The Manchester Prototype Dataflow Computer", Comm. ACM, Vol.28, No.1, Jan. 1985, pp.34-52.

[Jeff85] Jeffery T. "The µPD7281 Processor", Byte, November 1985, pp.237-246.

[Kapa84] Kapauan A., Field J.T., Gannon D.B., Snyder L. "The Pringle Parallel Computer", The 11th Ann. Int'l Symp. Arch., June 5-7, 1984, Ann Arbor, Michigan, pp.12-20.

[Kell79] Keller R.M. et.al. "A Loosely Coupled Aplicative Multiprocessing System", Proc. Nat. Comp. Conf., 1979, pp.861-870.

[Mago80] Magó G.A. "A Cellular Computer Architecture for Functional Programming", Proc. IEEE COMPCON 80, Feb. 1980, pp.179-187.

[Milu86] Milutinović V. Tutorial on Advanced Microprocessors and High-Level Language Computer Architecture, IEEE Comp. Soc. Press, 1986.

[Patn86] Patnaik L.M., Govindarajan R., Ramadoss N.S. "Design and Performance Evaluation of EXMAN: An EXtended MANchester Data Flow Computer", IEEE Trans. Comp. Vol. C-35, No.3, March 1986, pp.229-244.

[Patt85] Patterson D.A. "Reduced Instruction Set Computers", Comm. ACM, Vol.28, No.1, Jan. 1985, pp.8-21.

[Prei85] Preiss B.R., Hamacher V.C. "Data Flow on a Queue Machine", The 12th Ann. Int'l Symp. Comp. Arch., June 17-19, 1985, Boston, Massachusetts, pp.342-351.

[Robi85a] Robič B., Šilc J., Mihovilovič B. "Funkcionalno programirni sistemi", Informatica 4/85, str.366-370, Nova Gorica, 1985.

[Robi85b] Robič B, Šilc J.: "Jeziki in arhitekture sodobnih računalniških sistemov", IJS DP-4058, Ljubljana, November 1985.

[Robi86a] Robič B., Šilc J., "On Choosing a Plan for the Execution of Data Flow Program Graph", Informatica 3/86, pp.11-17.

[Robi86b] Robič B., Šilc J., "Analiza statičnih podatkovno pretokovnih programskih grafov", Elektrotehniški vestnik 86/2, str.53-56.

[Shim86] Shimada T., Hiraki K., Nishida K., Sekiguchi S. "Evaluation of a Prototype Data Flow Processor of the SIGMA-1 for Scientific Computations", Proc. 13th Int'l Symp. Comp. Arch., June 1986, pp.226-234.

[Shor73] Shore J.E. "Second Thoughts on Parallel Processing", Comput. Elect. Eng., Vol.1, 1973, pp.95-109.

[Slee81] Sleep M.R., Burton F.W. "Towards a Zero Assignment Parallel Procesor", Proc. 2nd Int'l Conf. Distributing Computing, April 1981.

[Snyd84] Snyder,L. "Supercomputers and VLSI: The Effect of Large-Scale Integration on Computer Architecture", Advances in Computers, Vol.23, Marshall C. Yovits, Ed.,Academic Press, 1984.

[Ston75] Stone,H.S. Introduction to Computer Architecture, SRA,Chicago 1975.

[Silc84] Šilc J., Robič B. "Računalniki krmiljeni s tokom podatkov", IJS DP-3579,Ljubljana, Oktober 1984.

[Silc85a] Šilc J., Robič B.: "Osnovna načela OF sistemov", Informatica 2/85, str.10-15.

[Silc85b] Šilc J., Robič B., Mihovilovič B. "Podatkovno vodene računalniške arhitekure", Informatica 4/85, str.371-376.

[Silc86] Šilc J., Robič B. "Procesor s podatkovno pretokovno arhitekturo",Informatica 4/86.

[Traub85] Traub K.R. "An Abstract Parallel Graph Reduction Machine", The 12th Ann. Int'l Symp. Comp. Arch., June 17-19, 1985, Boston, Massachusetts, pp.333-341.

[Trel80] Treleaven P.C., Mole G.F. "A Multi-processor Reduction Machine for User-defined Reduction Languages", Proc. 7th Int'l Symp. Comp. Arch., May 6-8, 1980, pp.121-130.

[Trel81] Treleaven P.C., Hopkins R.P. "Decentralized Computation", The 8th Ann. Symp. Comp. Arch., May 12-14, 1981, Minneapolis, Minnesota, pp.279-290.

[Trel82a] Treleaven P.C., Brownbridge D.R., Hopkins R.P. "Data-Driven and Demand-Driven Computer Architecture", Comp. Surv., Vol.14, No.1, March 1982.

[Trel82b] Treleaven P.C., Hopkins R.P. "A recursive Computer Architecture for VLSI", The 9th Ann. Symp. Comp. Arch., Apr. 26-29, 1982, Austin, Texas, pp.229-238.

[Trel82c] Treleaven P.C., Hopkins R.P., Rautenbach P.W. "Combining Data Flow and Control Flow Computing", Comput.J., Vol.25, No.1, Feb. 1982.

[Trel84] Treleavan P.C., Lima I.G. "Future Computers: Logic, Data Flow,...,Control Flow?", Computer, Vol.17, No.4, March 1984, pp.47-57.

[Wats79] Watson I., Gurd J. "A Prototype Data Flow Computer with Token Labeling", Proc. Nat. Computer Conf., New York, N.Y., June 4-7, 1979, pp.623-628.

[Zele85] Železnikar A.P. "Mednarodna konferenca o računalniških sistemih pete generacije v Tokyu", Informatica 85/3, str.68-70.