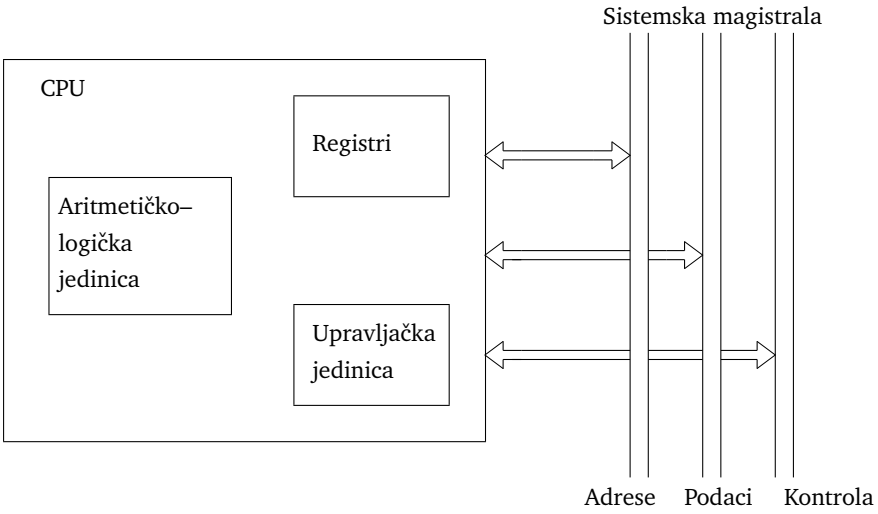
Struktura centralnog procesora

Organizacija centralnog procesora

- Kod prvih generacija procesora glavne komponente CPU-a su bile aritmetičko–logička jedinica (eng. *Arithmetic-Logic Unit, ALU*) i upravljačka jedinica (eng. *Control Unit, CU*).
- Ove i druge funkcije koje poseduju savremeni mikroprocesori se realizuju preko više funkcionalnih jedinica.
- Bez obzira na broj funkcionalnih jedinica procesora, proces izvršavanja instrukcija uključuje:
  - dohvatanje instrukcije i njenu interpretaciju
  - obezbedjivanje argumenata instrukcije
  - čuvanje vrednosti dobijenih pri izvršavanju
  - odredjivanje naredne instrukcije koja se izvršava



Slika 1: Uprošćena struktura centralnog procesora

Globalne funkcije funkcionalnih jedinica mikroprocesora su:

BCEU *Branch Control Element Unit*. Ova jedinica se koristi pri obradi zahteva za podacima i instrukcijama iz keš memorije, prevodjenje virtualnih u apsolutne adrese, pri implementaciji preklapanja instrukcija, itd. Pored toga ova jedinica sadrži i mikrokod.

IU *Instruction Unit* se koristi za dekodiranje instrukcija i odredjivanje njihovih komponenti.

AGU *Address Generation Unit* se koristi za generisanje adresa koje treba sačuvati ili sa kojih treba napuniti vrednosti.

LSU *Load/Store Unit* se koristi za prijem i čuvanje podataka pri njihovom čitanju/upisu u L1 keš za podatke. Takodje obezbedjuje poravnanje, konverziju izmedju zapisa brojeva različitih dužina, kao i (na nekim mikroprocesorima) normalizaciju brojeva u pokretnom zarezu.

FX ALU *FiXed point ALU* vrši operacije sa brojevima u fiksnom zarezu (celobojnim i BCD) kao i logičke operacije.

FP ALU *Floating Point ALU* vrši operacije sa brojevima u pokretnom zarezu.

Kod nekih mikroprocesora se ova jedinica koristi za izvršavanje dodatnih skupova instrukcija (npr MMX), dok je kod drugih mikroprocesora jedinica koja izvršava ove dodatne skupove instrukcija (npr. SSE ili AltiVec) po strukturi slična FP ALU i vrlo blisko povezana sa njom.

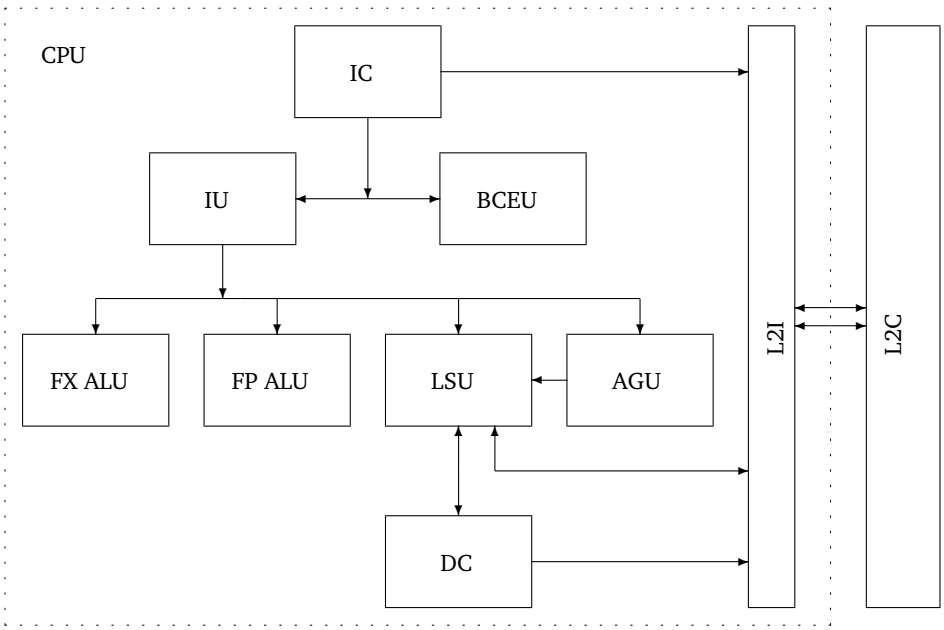
DC *Data Cache*. L1 Keš za podatke.

IC *Instruction Cache*. L1 Keš za instrukcije.

L2I *Cache Level 2 Interface*. Veza ka kešu na nivou 2.

L2C *Level 2 Cache*. Keš na nivou 2. Formalno gledano, on se nalazi van CPU-a i sa njim je povezan posebnom magistralom.

Jedinice su medjusobno povezane internim vezama koje se nazivaju *interne magistrale procesora*.



Slika 2: Uprošćena interna struktura CPU-a

Registri

Registri centralnog procesora se prema nameni mogu podeliti na

1. Registre opšte namene, i
2. Specijalizovane registre.

Registri opšte namene

Optimalan broj registara opšte namene za CISC procesore je u intervalu [8,32], dok je za RISC procesore ≥ 32, i kreće se do nekoliko stotina.

U zavisnosti od dizajna procesora i odgovarajućeg mašinskog jezika, opšti registri se mogu dodatno podeliti prema nameni, npr. na:

- *Akumulatore* za aritmetičko–logičke operacije i čuvanje znakovnih podataka.
- *Indeks registre* koji se koriste za indeksno adresiranje.
- *Pokazivače segmenata* koji sadrže početnu adresu segmenta na računarima sa segmentnim adresiranjem.
- *Pokazivače steka* koji pokazuju na početak memorije koja je odredjena za stek.

Specijalizovani registri

- Većina ovih registara nije direktno dostupna korisničkim programima.
- Neke od njih koriste samo programi operativnog sistema, dok su drugi dostupni samo na nivou mikrokoda.

Neki od specijalnih registara i njihove funkcije su:

- *Instrukcioni registar, IR* koji sadrži poslednju pročitanu instrukciju.
- *Registar memorijskih adresa, RMA* koji sadrži adresu u memoriji iz koje treba preneti sadržaj u PRM ili u koju treba upisati sadržaj iz PRM.
- *Prihvatni registar memorije, PRM*. Sadrži reč koja je poslednja pročitana iz memorije ili koju treba upisati u memoriju na adresu koja je sadržana u RMA.
- *Brojač instrukcija, PC* koji sadrži adresu sledeće instrukcije koju treba preneti u procesor.
- *Registar koji sadrži status programa koji se trenutno izvršava, PSW* (eng. *Program Status Word*). Informacije koje sadrži ovaj registar a odnose se na program koji se trenutno izvršava uključuju:
  - Uslovne kodove (eng. *Condition codes, CC*) pri izvršavaju aritmetičkih i logičkih operacija.
  - Postojanje prenosa pri aritmetičkim operacijama.
  - Prekoračenje pri aritmetičkim operacijama i vrstu podataka nad kojima je došlo do prekoračenja (celi brojevi, BCD brojevi).
  - Prekoračenje i potkoračenje pri izvodjenju operacija nad realnim brojevima u ne-IEEE 754 zapisu.
  - Indikatore da li je dozvoljen prekid kao i tip prekida koji je dozvoljen.
  - Indikator da li je tekući program u supervizorskom ili korisničkom režimu rada.
  - Indikator da li je tekuća adresa realna ili je potrebno preračunavati adrese iz programa da bi se dobile realne adrese.
  - Indikator da li je CPU u stanju čekanja kada ne izvršava ni jednu instrukciju. Ovo stanje može da se promeni jedino pojavom prekida.
  - ...
- *Kontrolni registar pri izvršavanju operacija u pokretnom zarezu, FPC* (eng. *Floating Point Control*). Ovaj registar sadrži informacije vezane za izračunvanja u pokretnom zarezu (tačnije IEEE 754 aritmetiku).
- *Kontrolne registre* koji sadrže specifične informacije za svaku mašinu: tekuće stanje mašine, vektor prekida, postojanje/nepostojanje FPU (npr. na Pentiumu ili PowerPC procesoru), rad u posebnom režimu izvršavanja instrukcija (npr. protected/real mod kod Pentium procesora), itd.

Naziv registra	Broj	Dužina	Funkcija
Opšti	8	32	Opšti registri
FPR	8	80	FPR (*floating point register*) sadrži brojeve u pokretnom zarezu
FPC	1	16	Kontrolni bitovi za operacije u pokretnom zarezu
Status	1	16	Tekuće stanje FPU
Oznake	1	16	Sadrži 2-bitne oznake sadržaja svakog od 8 FPR (važeći sadržaj, nula, specijalna vrednost, prazno)
MMX	8	64	Preslikavaju se u FPR na bitove manje težine
Segment	6	16	Koriste se pri izboru segmenta
Zastavice	1	32	Sadrži status i kontrolne bitove
Pokazivač instrukcije	1	32	Sadrži adresu tekuće instrukcije
EFLAGS	1	32	Sadrži kontrolne bitove i kodove generisane pri operacijama u procesoru, odnosno delimično vrši funkciju sličnu PSW
Kontrolni registri	5	32	Sadrže informacije vezane za operacije procesora. Drugi registar po redu se ne koristi

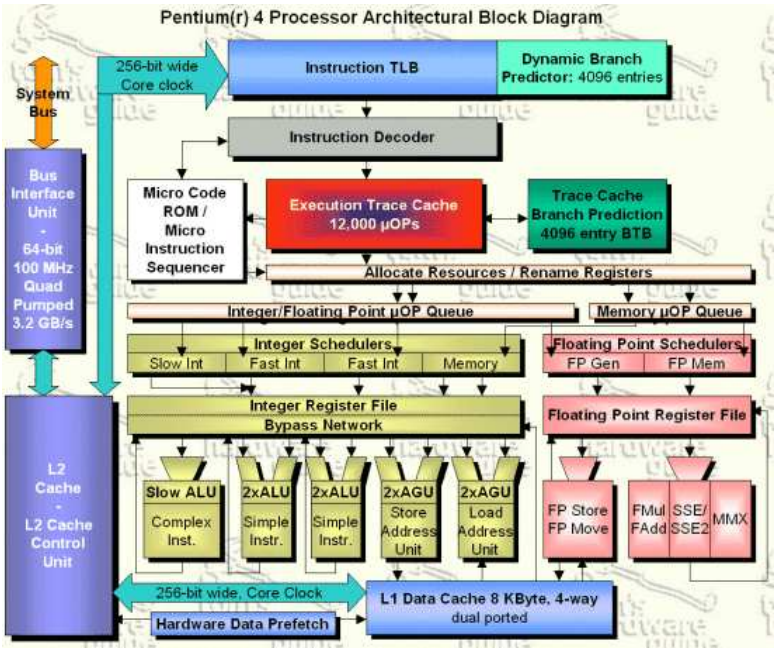
Tabela 1: Neki od registara Pentium Procesora

Naziv registra	Broj	Dužina	Funkcija
Opšti	32	64	Opšti registri
Registar izuzetaka	2	32	Sadrži vrstu izuzetka pri operaciji sa brojevima u fiksnom zarezu
FPR	32	64	Sadrži brojeve u pokretnom zarezu
FPSCR	1	32	Sadrži bitove za status i kontrolu operacije u pokretnom zarezu
Uslovni registar	1	32	Sadrži 8 četvorobitnih polja sa uslovnim kodovima
Povezivač	1	64	Koristi se u instrukcijama uslovnog grananja za indirektno adresiranje
Brojač	1	64	Omogućava kontrolu pri iterativnim ciklusima. Pored toga koristi se i za indirektno adresiranje željene adrese u instrukcijama grananja
Stanje mašine	1	64	MSR (eng. *Machine State Register*) sadrži različita stanja procesora pri izvršavanju operacije, tj. sadrži deo informacija iz PSW

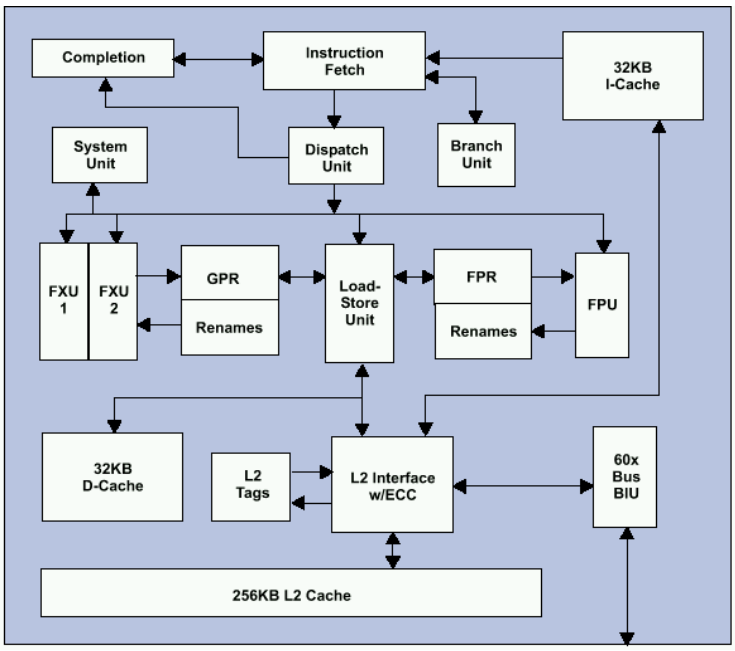
Tabela 2: Neki od registara PowerPC Procesora

Naziv registra	Broj	Dužina	Funkcija
Opšti	16	32	Opšti registri
FPR	16	64	Svaki FPR može da sadrži brojeve zapisane u pokretnom zarezu bilo pomoću binarne bilo pomoću heksadekadne osnove
FPCR	1	32	Sadrži bitove za maske, zastavice, izuzeća i način zaokruživanja
PSW	1	64	Status programa koji se trenutno izvršava
Kontrolni registri	16	32	Kontrola i manipulacija podacima van CPU-a
Pristupni registri	16	32	Sadrže indirektne specifikacije parametara u tabeli segmenata
Vektor status registar	1	64	Status i kontrola operacija u vektorskom procesoru
Prefiks	1	32	Koristi se pri odredjivanju realnih i apsolutnih adresa
TOD	1	104	TOD (eng. *Time-Of-Day*) se koristi sa merenje vremena u računaru

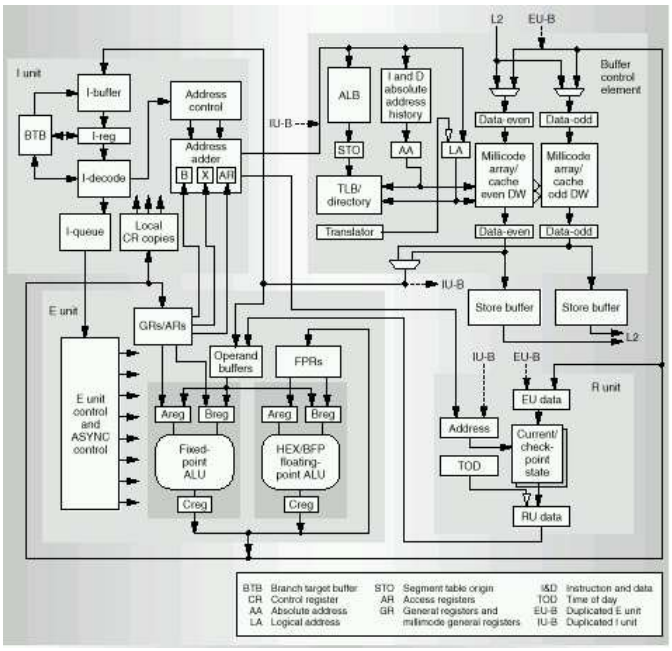
Tabela 3: Neki od registara S/390 G6 procesora



Slika 3: Struktura Pentium 4 procesora



Slika 4: Struktura PowerPC 750 procesora



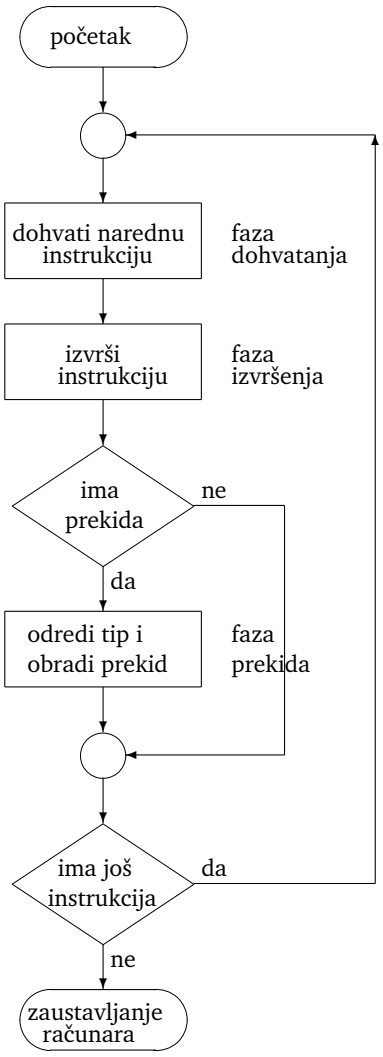
Slika 5: Struktura S/390 G5/G6 procesora

Instrukcioni ciklus

Izvršavanje svih faza mašinskih instrukcija čini proces koji se naziva *instrukcioni ciklus*

Svaka od faza mašinskog ciklusa se može raščlaniti na različit broj koraka (podfaza). Koji od ovih koraka će biti uključeni u izvršavanje instrukcije zavisi od same instrukcije. Jedno od mogućih raščlanjenja faza mašinske instrukcije je:

1. Faza dohvatanja:
    - Izračunavanje adrese instrukcije, pri čemu se odredjuje adresa naredne instrukcije koja treba da se izvrši.
    - Dohvatanje instrukcije. U ovom koraku se čita instrukcija iz memorije i prenosi u CPU.
    - Dekodiranje operacionog koda instrukcije. Na osnovu analize operacionog koda odredjuje se tip operacije i broj argumenata instrukcije.
    - Izračunavanje adrese operanada. Za svaki od eventualnih operanada instrukcije se odredjuje gde je smešten (registar ili memorija). Ako se koristi indirektno adresiranje tada odredjivanje adrese operanda uključuje dodatno obraćanje memoriji za odredjivanje adrese operanda – *faza indirekcije*.
    - Dohvatanje operanda u kome se vrši dohvatanje operanda iz memorije.
2. Faza izvršenja:
    - Izvršavanje operacije koja je zahtevana operacionim kodom.
    - Čuvanje rezultata u memoriji ili na ulazno/izlaznim uredjajima. Ako se i u ovom koraku koristi indirektno adrsiranje tada za svaki od operanada koji predstavljaju rezultat je potrebno dodatno obraćanje memoriji za odredjivanje njegove adrese.
3. Faza prekida:
    - Proveru da li se u medjuvremenu desio prekid.
    - Ako je bilo prekida, tada se vrši njegova obrada.



Slika 6: Blok dijagram instrukcionog ciklusa

Preklapanje instrukcija

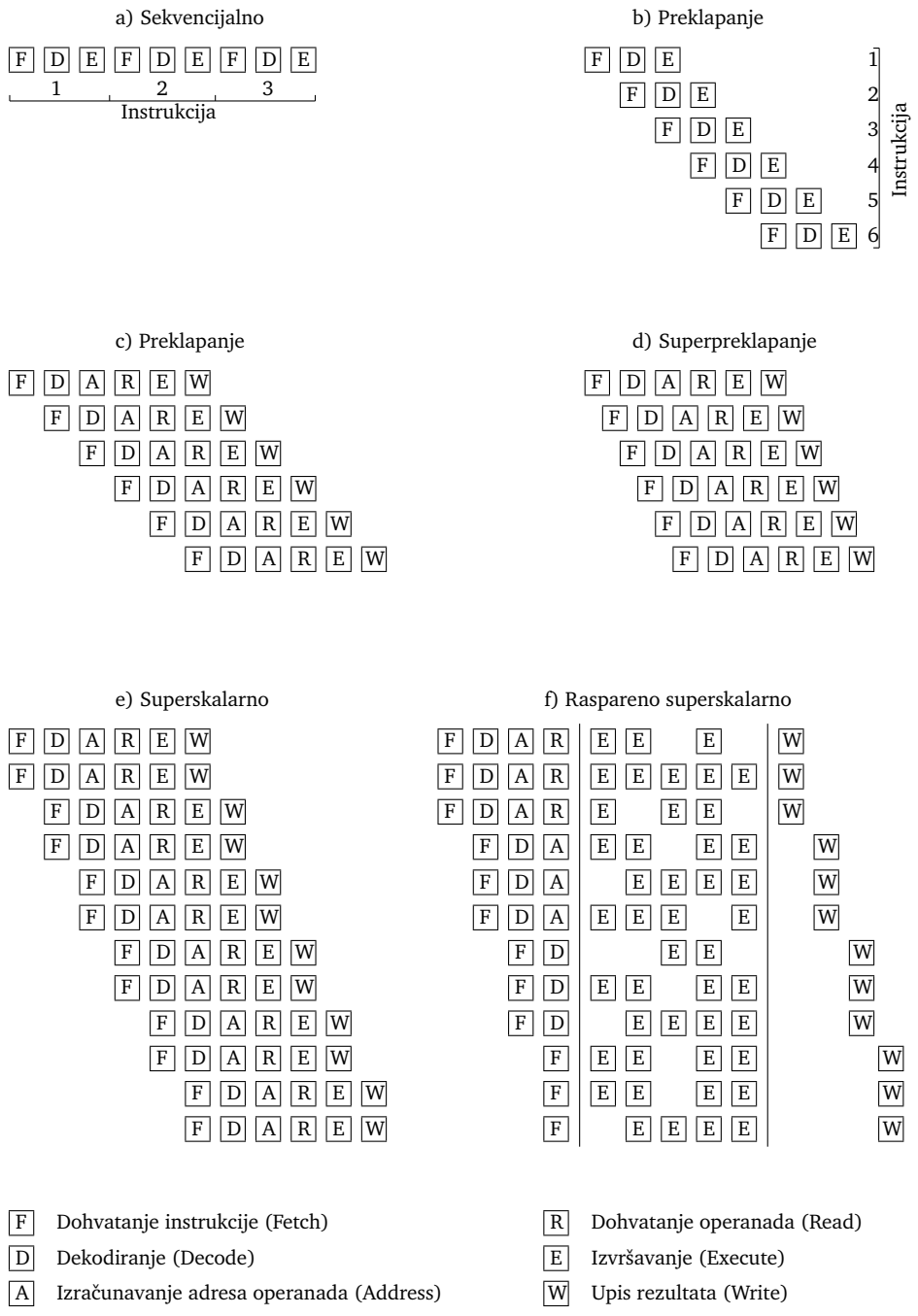
Sam termin *preklapanje* (eng. *pipelining*) označava mogućnost istovremenog izvršavanja različitih faza mašinske instrukcije.

Načini izvršavanja instrukcija

Šematski prikaz nekih tehnika izvršavanja mašinskih instrukcija je dat na slici 7. Pretpostavimo da se izvršavanje mašinske instrukcije može podeliti u 6 koraka:

1. Dohvatanje instrukcije (eng. *Fetch*)
2. Dekodiranje operacionog koda instrukcije (eng. *Decode*)
3. Odredjivanje adrese operanada (eng. *Address calculation*)
4. Dohvatanje operanada (eng. *Read operands*)
5. Izvršavanje (eng. *Execute*)
6. Upis rezultata (eng. *Write results*)

Svaki korak je na slici označen kao uokvireni kvadrat u kome je upisan naziv koraka. Radi pojednostavljenja, obrada prekida nije prikazana na slici. Takodje, pretpostavka je da su vremena potrebna za izvršavanje svih koraka jednaka i da se svaki od koraka izvršava u jednom ciklusu časovnika (ova pretpostavka ne važi u realnim situacijama).

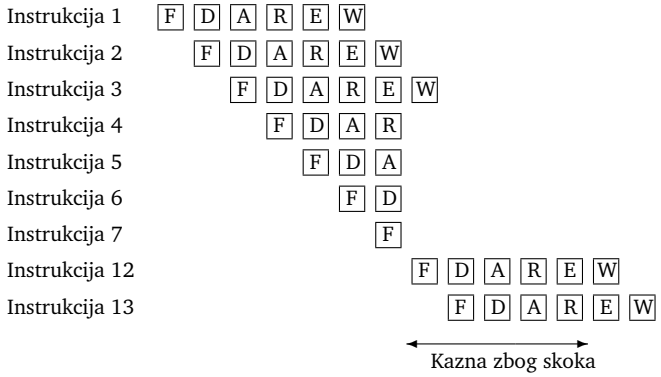


Slika 7: Tehnike izvršavanja mašinskih instrukcija

Preklapanje faza instrukcije teorijski je omogućilo proprocionalno smanjenje vremena potrebnog za njeno izvršavanje. Medjutim, u realnim situacijama ovo se ne dostiže iz najmanje dva razloga:

1. Vreme potrebno za izvršavanje je duže od vremena potrebnog za dohvatanje instrukcije.
2. Nailazak na instrukciju uslovnog grananja dovodi do toga da nije poznata adresa naredne instrukcije koju treba dohvatiti.

Prvi razlog se može donekle neutralisati podelom instrukcije na veći broj koraka (npr. slika 7–c). Procesori su, u kasnijem periodu korišćenja ove tehnologije, delili instrukciju često na 10 i više koraka.



Slika 8: Efekat izvršavanja instrukcije uslovnog grananja

Negativni efekti koji nastaju usled instrukcija uslovnog grananja se ne mogu eliminisati iz programa već samo ublažiti. Značajno povećanje broja koraka u izvršavaju instrukcijea nije ostvarivo iz sledeća dva razloga:

1. Izmedju svaka dva koraka mašinske instrukcije se odredjeni interval vremena koristi za prebacivanje podataka iz bafera u bafer i različite vrste priprema za izvršavanje narednog koraka. Zbir ovih vremena može u značajnoj meri da produži izvršavanje instrukcije.

2. Broj logičkih kola neophodan za kontrolu i razrešenje memorijskih i registarskih konflikata naglo raste sa povećanjem broja koraka u instrukciji. To dovodi do opasnosti da logička kola postanu kompleksnija (i troše više vremena) od koraka instrukcije koje treba da kontrolišu.

Zbog toga se pribegava tehnikama za predvidjanje koja će instrukcija sledeća biti dohvaćena i izvršena.

Predčitanje instrukcija

Kada se u nizu instrukcija uoči postojanje uslovnog grananja tada se unapred pročitaju i instrukcija koja sledi za uslovnim grananjem i instrukcija na koju se vrši uslovno grananje.

Upotreba više vodova instrukcija

- Postoje dva voda instrukcija
- Početni delovi oba voda su identični dok se u nastavku smeštaju obe ciljne instrukcije na koje može da se dodje izvršavanjem uslovnog grananja.

Bafer za petlje

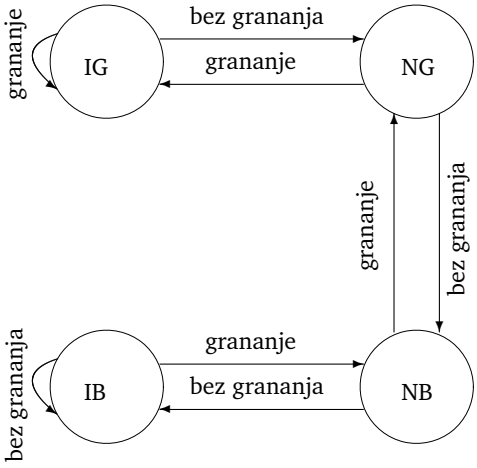
Bafer za petlje predstavlja malu količinu vrlo brze memorije koja se ažurira pri izvršavanju faze dohvatanja u instrukciji. Bafer sadrži, u redosledu dohvatanja, n poslednje dohvaćenih instrukcija.

Predvidjanje adrese grananja

Za predvidjanje adrese na koju će se izvršiti grananje se najčešće koriste sledeće tehnike:

- Statičko predvidjanje, u kome se ne konsultuje istorija izvršavanja do trenutka nailaska na instrukciju grananja.
  - *Rezultat predvidjanja se nikada ne uzima u obzir*, tj. predvidjanje se i ne vrši.
  - *Rezultat predvidjanja se uvek uzima u obzir*, tj. dohvatanje instrukcije sa ciljne adrese na koju se izvršavanje prenosi.
  - *Predvidjanje u zavisnosti od operacionog koda.*

- Dinamičko predvidjanje, u kome rezultat predvidjanja zavisi od prethodnih izvršavanja naredbi uslovnog grananja:
  - Korišćenje *prekidača Izvršen/Nije izvršen.* Uz svaku instrukciju uslovnog grananja se asocira jedan ili više bitova koji sadrže prethodnu istoriju instrukcije. Ovi bitovi se nazivaju *prekidači izvršen/nije izvršen* jer na osnovu njihovog sadržaja procesor formira odluku o dohvatanju kada se sledeći put javi instrukcija kojoj su pridruženi.



I – poslednje predvidjanje ispravno
N – poslednje predvidjanje neispravno
G – predvidjeno grananje
B – predvidjeno da neće biti grananja

Slika 9: Dijagram stanja predvidjanja grananja u instrukciji uslovnog skoka

  - *Upotreba tabele sa istorijom skokova.*

Odloženo grananje

- Vrši se reorganizacija koda radi dobijanja optimalnih performansi izvršavanja.
- Izvršavaju se instrukcije koje nemaju efekta na izvršavanje ostalih instrukcija.
- Promena redosleda se vrši da bi se obezbedilo stalno postojanje instrukcija u svakoj od faza u vodu instrukcija.
- Promena redosleda instrukcija je relativno efikasna za bezuslovno grananje kao i pozive i povratak iz potprograma.
- Za uslovno grananje promena redosleda se primenjuje samo u slučajevima kada izvršavanje prethodne instrukcije ni na koji način ne utiče na izvršavanje instrukcije grananja.

Superskalarni procesori

Za procesore koji mogu da istovremeno inicijalizuju i zatim nezavisno da izvršavaju mašinske instrukcije se kaže da poseduju superskalarnu arhitekturu.

Superskalarni procesori mogu da izvršavaju više od jedne instrukcije u svakom instrukcionom ciklusu. Zbog toga njihove funkcionalne jedinice moraju biti sposobne da istovremeno obrade više od jednog koraka u instrukcionom ciklusu.

Osobine superskalarnog izračunavanja

Nivo paralelizma instrukcija (eng. *instruction-level paralelism*) označava broj instrukcija programa koje mogu paralelno da se izvršavaju. Paralelizam instrukcija se javlja ako u programu postoje instrukcije koje se nalaze u nizu jedna za drugom pri čemu su medjusobno nezavisne tako da mogu da se izvršavaju istovremeno. Na primer,

```
LOAD   R1,R2     ;(R2)       ⟶ R1          ADD    R1,'1'    ;(R1)+1     ⟶ R1
ADD    R3,'1'    ;(R3)+1     ⟶ R3          AR     R2,R1     ;(R2)+(R1)  ⟶ R2
AR     R4,R2     ;(R4)+(R2)  ⟶ R4          STORE  R4,0(R2)  ;(R4)       ⟶ (R2)
```

nivo =3 nivo =1

Paralelizam ma ine je (hardverska) mogućnost procesora da podrži paralelizam instrukcija. Nivo paralelizma mašine je odredjen brojem instrukcija koje mogu biti dohvaćene i izvršene istovremeno, odnosno brojem paralelnih vodova instrukcija. Na nivo paralelizma mašine takodje utiču i i brzina i efikasnost algoritma koji se koristi za odredjivanje medjusobne zavisnosti instrukcija.

Za odredjivanje optimalnog nivoa paralelizma moraju se uzeti u obzir sledeće tri činjenice:

1. Redosled dohvatanja instrukcija.
2. Redosled izvršavanja instrukcija.
3. Redosled menjenja sadržaja registara i memorijskih lokacija pri eventualnom izvršavaju instrukcija u redosledu dohvatanja.

Optimalni nivo paralelizma se dostiže menjanjem redosleda izvršavanja instrukcija uz očuvanje semantike koja odgovara izvršavanju instrukcija u redosledu dohvatanja.

Ograničenja pri obradi

a) **Zavisnost od prethodnih podataka (zavisnost upis-čitanje**

```
AR      R1,R2        ;(R1)+(R2) ⟶  R1
STORE   R4,0(R1)     ;(R4)      ⟶  (R1)
```

Druga instrukcija može da bude dohvaćena i dekodirana ali ne može da bude izvršena pre prve, jer adresa upisa sadržaja registra R4 zavisi od vrednosti koja se izračunava u prvoj instrukciji.

b) **Proceduralna zavisnost**

Instrukcija koja sledi iza operacije (uslovnog) grananja zavisi od procedure grananja i ne može biti izvršena pre izvršavanja instrukcije koja sadrži grananje.

Dodatni oblik proceduralne zavisnosti je prisutan kod instrukcija promenljive dužine. Takve instrukcije moraju biti bar delimično dekodirane da bi se odredila adresa naredne instrukcije koju treba dohvatiti.

c) **Zavisnost od resursa**

Zavisnost od resursa predstavlja slučaj kada dve ili više instrukcija istovremeno zahtevaju isti resurs (npr. memoriju, keš, FPU, FXU, ...).

d) **Zavisnost od izlaza**

```
AR      R2,R1          ;(R2)+(R1) ⟶  R2
MOVE    R3,R2+1        ;(R2)+1    ⟶  (R3)
MOVE    R2,R1+1        ;(R1)+1    ⟶  (R2)
MOVE    R4,R2+R3       ;(R2)+(R3) ⟶  (R4)
```

Druga instrukcija zavisi od rezultata prve instrukcije. Sličan odnos je i izmedju četvrte i prve, četvrte i druge kao i izmedju četvrte i treće instrukcije. Medjutim, izmedju treće i prve instrukcije takodje postoji zavisnost, jer bi izvršavanje treće instrukcije pre prve dovelo do nekorektnog sadržaja u registru R3 i samim tim do nekorektnih izlaznih vrednosti prve instrukcije. Da bi ova izlazna vrednost bila korektna (jer kasnije predstavlja ulaz u drugu instrukciju) izvršavanje treće instrukcije mora da bude odloženo sve dok postoji opasnost da bi njeni rezultati mogli da budu preklopljeni izvršavanjem instrukcije koja je pre nje u redosledu dohvatanja.

e) **Antizavisnost**

```
AR      R2,R1          ;(R2)+(R1) ⟶  R2
MOVE    R3,R2+1        ;(R2)+1    ⟶  (R3)
MOVE    R2,R1+1        ;(R1)+1    ⟶  (R2)
MOVE    R4,R2+R3       ;(R2)+(R3) ⟶  (R4)
```

Pošto treća instrukcija ažurira sadržaj registra R2, ona ne sme da završi izvršavanje pre nego što druga instrukcija započne svoje izvršavanje i dohvati operande. Ovakva zavisnost se u literaturi naziva **antizavisnost** ili **zavisnost pisanje–pisanje**.

Prema izvedenim promenama, instrukcije superskalarnih procesora se mogu podeliti u tri grupe prema redosledu u kome počinje i završava obrada instrukcija:

1. Početak i završetak obrade instrukcija su u redosledu dohvatanja.
2. Početak obrade instrukcija je redosledu dohvatanja, dok je završetak u redosledu nezavisnom od redosleda dohvatanja.
3. Početak i kraj obrade instrukcija su u redosledu nezavisnom od redosleda dohvatanja.

Pri tome se mogu, u zavisnosti od dostignutog nivoa paralelizma instrukcija, u nekim instrukcijama promeniti nazivi registara.

Primer: na narednoj slici je dat uporedni prikaz različitih tehnika za izvršenje niza od šest instrukcija za koje važi:

1. instrukcija I1 zahteva za izvršavanje dva ciklusa časovnika, sve ostale instrukcije zahtevaju jedan ciklus,
2. instrukcija I1 radi sa brojevima u pokretnom zarezu, tj. zahteva za izvršavanje FPU,
3. instrukcije I2, I5 i I6 zahtevaju za izvršavanje jedinicu FXU_{int},
4. instrukcije I3, I4 zahtevaju za izvršavanje jedinicu FXU_{BCD},
5. rezultati instrukcije I4 su argumenti instrukcije I5.

Svojstva obrade instrukcija su:

- vod može istovremeno da dohvati i dekodira dve instrukcije;
- procesor poseduje tri razdvojene jedinice za obradu (npr. dve FX ALU – jednu za rad sa celim a drugu sa BCD brojevima i jednu FP ALU);
- vod instrukcija poseduje dve instance pomoću kojih može istovremeno da predaje rezultate obrade.

Ciklus	Dekodiranje		Izvršavanje			Upis vredn.	
1	I1	I2					
2	I3	I4	I1	I2			
3	I3	I4	I1				
4		I4			I3	I1	I2
5	I5	I6			I4		
6		I6		I5		I3	I4
7				I6			
8						I5	I6

a) Početak i završetak u redosledu dohvatanja

Ciklus	Dekodiranje		Izvršavanje			Upis vredn.	
1	I1	I2					
2	I3	I4	I1	I2			
3		I4	I1		I3	I2	
4	I5	I6			I4	I1	I3
5		I6		I5		I4	
6				I6		I5	
7						I6	

b) Početak u redosledu, kraj nezavisan od redosleda dohvatanja

Ciklus	Dekodiranje		Prozor	Izvršavanje			Upis vredn.	
1	I1	I2						
2	I3	I4	I1,I2	I1	I2			
3	I5	I6	I3,I4	I1		I3	I2	
4			I4,I5,I6		I6	I4	I1	I3
5			I5		I5		I4	I6
6							I5	

c) Početak i kraj nezavisni od redosleda dohvatanja

Slika 10: Tehnike obrade instrukcija superskalarnih procesora

Promena naziva registara

```
AR      R2,R1          ;(R2_a)+(R1_a) ⟶ R2_b
MOVE    R3,R2+1        ;(R2_b)+1      ⟶ (R3_b)
MOVE    R2,R1+1        ;(R1_a)+1      ⟶ (R2_c)
MOVE    R4,R2+R3       ;(R2_c)+(R3_b) ⟶ (R4_b)
```

Indeks u nazivu registra označava hardverski registar koji se alocira radi čuvanja vrednosti. Prethodni niz instrukcija se posle promene imena može napisati kao

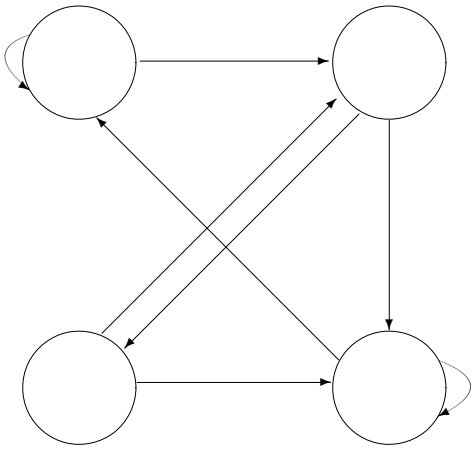
```
AR      Rb,Ra
MOVE    Rb,Rb+1
MOVE    Rc,Ra+1
MOVE    Rb,Rc+Rb
```

Kako se registri obično označavaju sa R1, R2, ... dobijeni niz instrukcija ima oblik

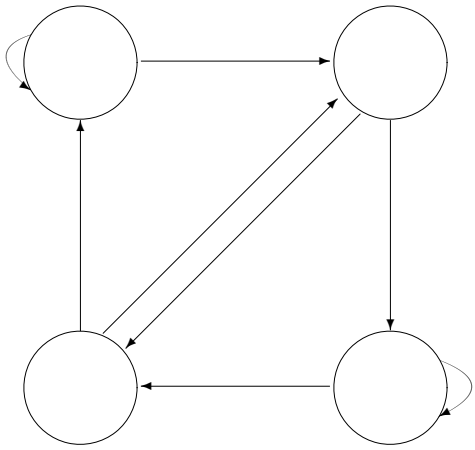
```
AR      R2,R1
MOVE    R2,R2+1
MOVE    R3,R1+1
MOVE    R2,R3+R2
```

U ovom zapisu ne postoji antizavisnost tako da prva i treća instrukcija mogu da se izvršavaju istovremeno.

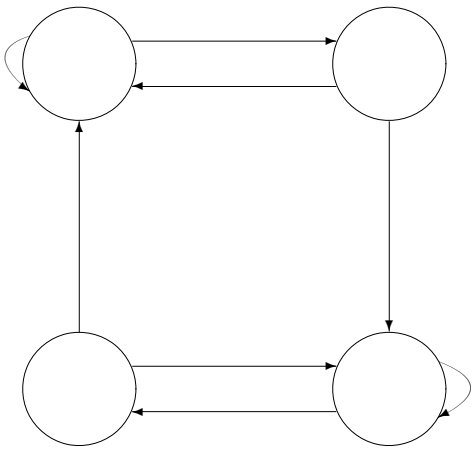
Domaći zadatak: Dati moguća tumaćenja sledećih dijagrama stanja predvidjanja grananja u instrukciji uslovnog skoka.



Slika 11: Dijagram 1



Slika 12: Dijagram 2



Slika 13: Dijagram 3