

Osnovi mikroprocesorskih i mikrokontrolerskih sistema - Uvod u RISC mikroprocesore

prof. dr Ivan Mezei

04. april 2022.

Glava 5

Uvod u RISC mikroprocesore

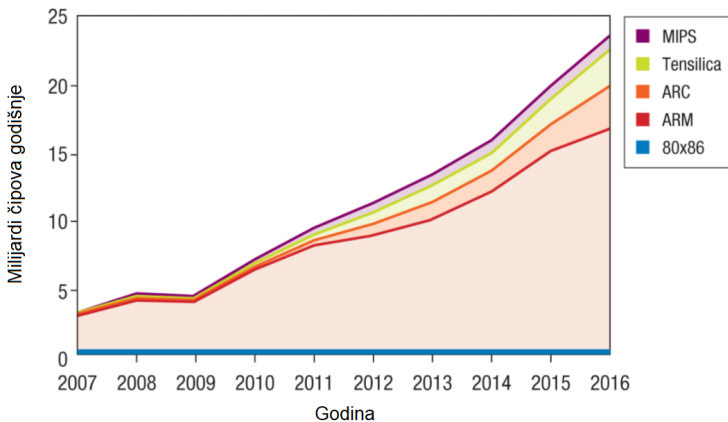
U dosadašnjim poglavljima smo se upoznali sa generalnim konceptima vezanim za mikroprocesore. Primere i zadatke koje smo obradili su uglavnom pripadali tzv. akumulatorskim arhitekturama ili njihovim varijetetima. Ove arhitekture, pored stek arhitektura, imaju više istorijski ili edukacioni karakter. Pri tome su bliže CISC konceptu¹. U ovom poglavlju ćemo dati akcenat na registarske arhitekture u domenu RISC počevši od hipotetičkih, edukacionih da bismo na kraju poglavlja obradili, koliko je to moguće uzimajući u obzir složenost tematike, dva primera realnih današnjih RISC arhitektura: ARM i RISC-V. U okviru edukacionih arhitektura ćemo pokazati jednotaktni RISC1 mikroprocesor i mikroprocesor sa protočnom obradom RISC-P. Sličan, znatno obimniji pristup se može pogledati i u knjigama [7] i [28].

5.1 Neke važne karakteristike RISC arhitektura

Pre nego što ukažemo na neke od važnih karakteristika RISC arhitektura najpre ćemo se osvrnuti na razloge zbog kojih su RISC arhitekture danas dominantne. Ovo se vidi jasno na slici 5.1 koja pokazuje da u post-PC eri proizvodnja RISC mikroprocesora je daleko veća od CISC. Neke procene su da se na svaka četiri sata proizvede 10 miliona RISC mikroprocesora. Intelova arhitektura 80x86 je praktično

¹Podsetiti se razlika RISC i CISC arhitektura obrađenih ranije.

imala monopol u eri personalnih računara. Oni su performanse poboljšavali tako što su vremenski kritične instrukcije raščlanjivali na manje koje su izvršavali u RISC maniru. Pojavom savremenih oblasti kao što je *Internet of things* i drugih gde su mala potrošnja i male dimenzije od izuzetnog značaja jasno je da arhitekture koje mnogo troše i traže velike čipove ne mogu da se koriste. U ovim oblastima arhitektura 80x86 nije mogla da ima svoje mesto. Dodatno, pogledajmo jednačinu ?? obrađenu u sekciji o performansama mikroprocesora. Iz ove jednačine se vidi da vreme potrebno za izvršenje nekog programa zavisi od broja instrukcija, CPI i od učestanosti takta. Ako pretpostavimo da se koristi ista učestanost za dati program je kod CISC mikroprocesora potrebno u proseku oko upola manje instrukcija nego kod RISC, ali je zato CPI u proseku šest puta veći [29]. Ovo jasno ukazuje na bolje performanse koje je moguće postići RISC arhitekturama.



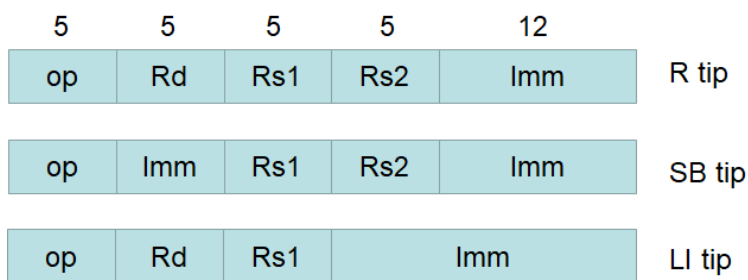
Slika 5.1: Proizvodnja RISC mikroprocesora [29]

Sledeće arhitekturne karakteristike RISC mikroprocesora su uobičajene. Umesto mnoštva specijalizovanih registara, kod RISC mikroprocesora se po pravilu koristi koncept registarskog polja (eng. *register file*) koje se sastoji od više registara opšte namene (npr. 16). Da bi se instrukcije što brže izvršavale obično se nastoji da je što manje pristupa memoriji radi zahvata operanda već da se on nalazi ako je moguće u registrima. Po pravilu aritmetičko-logičke instrukcije rade samo sa operadnima u registrima i eventualno konstantama (eng. *immediates*), a pristup za čitanje/upis u memoriju je dozvoljen samo specijalnim instrukcijama koje se najčešće označavaju mnemonicima koji su izvedeni od engleskih reči *Load*

(učitati) i *Store* (upisati) za razliku od izvedenih mnemonika kod CISC arhitektura koje se često izvode od engleske reči *Move*. Pri tome se teži što većoj regularnosti instrukcionog formata gde su sve instrukcije istih dimenzija (npr. 32-bitne), sa što manje različitih polja u formatu, a pogotovo se izbegavaju velike razlike u svrsi tih polja za različite instrukcije. Obično je podržan mali broj načina adresiranja. U upotrebi je Harvard organizacija memorije tako da je programska memorija odvojena od memorije za podatke.

5.2 RISC1

Prilikom razmatranja RISC arhitektura po pravilu se prvo polazi od tzv. osnovnih jednotaktnih ili jednociklusnih realizacija (eng. *single-cycle*) tj. realizacija kod kojih je $CPI=1$. Arhitekturu edukacionog RISC mikroprocesora ćemo u ovoj sekciji zvati RISC1. Dakle, to su arhitekture u kojima se svaka instrukcija realizuje u jednoj periodi takta i bez primene protočne obrade (o protočnoj obradi će biti reči u narednoj sekciji). Kod do sada urađenih primera u prethodnim poglavljima smo videli da je za izvršavanje instrukcija bilo potrebno više od jednog takta. Da bi se uopšte mogla realizovati jednotaktna arhitektura potrebno je da su zadovoljene određene pretpostavke, pre svega u domenu hardvera o čemu je bilo reči u prethodnoj sekciji. Pored toga, vreme trajanja tog jednog ciklusa takta će određivati ona instrukcija za koju je potrebno najduže vreme da se izvrši. Ovo vreme se obično označava sa T_c , a inverzna vrednost daje maksimalnu vrednost učestanosti takta. Prema metodologiji projektovanja mikroprocesora detaljno obrađenoj ranije,



Slika 5.2: Format instrukcija

pored hardverskih aspekata koji su izloženi u prethodnoj sekciji, potrebno je odgovoriti i na pitanja u vezi instrukcija. Radi što jednostavnijeg rada ćemo razmatrati

samo sledeće instrukcije: ADD, ADDI, XOR, LD, ST i BEQ. Format instrukcije je fiksne 32-bitne dužine i prikazan je na slici 5.2. Daljim modifikacijama se lako dobijaju po potrebi i drugi tipovi uobičajeni kod RISC mikroprocesora (npr. od LI tipa se lako dobija J tip proširenjem Imm i na Rs1 polje).

Od tipova adresiranja podržana su četiri tipa adresiranja: 1. neposredno, 2. registarsko, 3. bazno i 4. relativno u odnosu na programski brojač. R tip instrukcija se koristi kod registarskog adresiranja, LI tip kod instrukcije za čitanje iz memorije ($Rd = D\text{-}MEM[Rs1 + Imm]$) i neposrednog adresiranja, a SB tip kod instrukcije za upis u memoriju ($D\text{-}MEM[Rs1 + Imm] = Rs2$) i instrukcije uslovnog grananja. Detaljan opis instrukcija sa primerima dat je u tabeli 5.1.

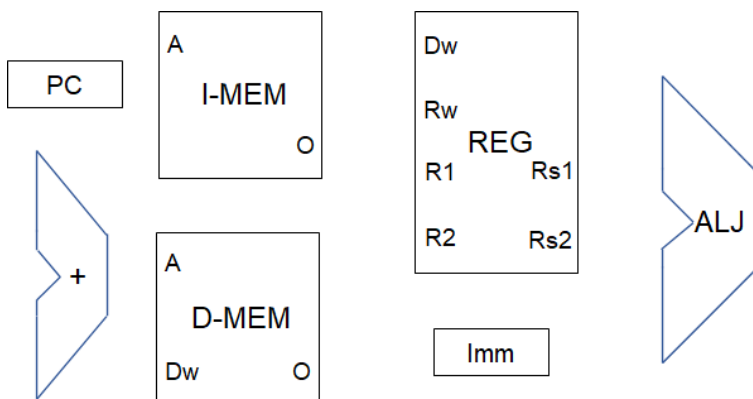
Tabela 5.1: Detaljan opis instrukcija RISC1

| Instr. | Tip | Primer | RTN | Polja instr. formata | | | |
|--------|-----|---------------|---|----------------------|---|---|-------|
| ADD | R | ADD R9,R8,R7 | $R9 \leftarrow R8 + R7$ | opadd | 9 | 8 | 7 0 |
| XOR | R | XOR R9,R8,R7 | $R9 \leftarrow R8 \oplus R7$ | opxor | 9 | 8 | 7 0 |
| ADDI | LI | ADD R9,R8,#5 | $R9 \leftarrow R8 + 5$ | opaddi | 9 | 8 | 0 5 |
| LD | LI | LD R7,R6(100) | $R7 \leftarrow M[R6 + 100]$ | opld | 7 | 6 | 0 100 |
| ST | SB | ST R5,R6(100) | $M[R6 + 100] \leftarrow R5$ | opst | 0 | 6 | 5 100 |
| BEQ | SB | BEQ R5,R6,100 | $(R5 = R6) \rightarrow$ $PC \leftarrow PC + 100$ | opbeq | 0 | 5 | 6 100 |

Nakon definisanih zahteva ovde ćemo pokazati metod evolucije mikroprocesora počevši od osnovnih unutrašnjih elemenata, a potom za odgovarajuće tipove instrukcija ćemo dodavati potrebne elemente i veze. Osnovni elementi RISC1 mikroprocesora su: 32-bitni programski brojač (PC), memorije za instrukcije i podatke (I-MEM, D-MEM), registarsko polje (REG), aritmetičko-logička jedinica (ALJ), blok za generisanje konstanti (Imm), sabirači, multiplekseri i linije za povezivanje. Glavni elementi su prikazani na slici 5.3.

Instrukciona memorija se može posmatrati kao ROM memorija tako da ona ima adresni ulaz (A) i izlaz za instrukciju (O). Memorija podataka je RAM tipa i ima iste priključke kao i instrukciona uz dodatak ulaza za upis podataka (Dw). Registarsko polje je realizovano kao 32x32 bita. Ima dva adresna porta za čitanje (R1 i R2) i jedan adresni port za upis (Rw). Adresni portovi su 5-bitni. Pročitani podaci se nalaze na izlazima Rs1 i Rs2, a podatak za upis je potrebno dovesti na ulazni port Dw. Blok Imm služi za rad sa konstantama pre svega radi očuvanja bita za znak. Drugim rečima on proširuje konstantu do 32-bitne vrednosti uz očuvanje znaka, koja se potom vodi na ALJ kao operand (drugi operand se uzima iz registra).

U evoluciji mikroprocesora RISC1 krećemo od programskog brojača. Kako je u



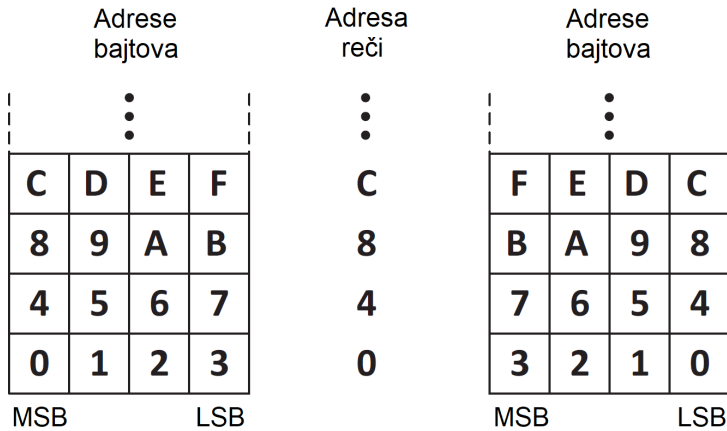
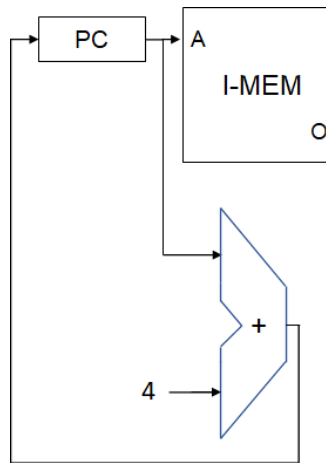
Slika 5.3: Glavni elementi RISC1

pitanju 32-bitni mikroprocesor, a memorija je adresibilna na nivou bajta, postavlja se pitanje kako su četiri bajta unutar 32-bitne reči postavljeni. Postoje dva načina za to i oni su prikazani na slici 5.4. Na levoj strani slike je prikazan tzv. veliki endian (*big-endian*) način. U ovom slučaju adrese najmanje značajnosti (*least significant byte* - LSB) pojedinačnih bajtova unutar reči kreću sa pozicije najveće značajnosti unutar reči (*most significant byte* - MSB). Na desnoj strani slike je prikazan tzv. mali endian (*little-endian*) način. Ovde je obrnuto u odnosu na veliki endian, tj. LSB adrese bajtova unutar reči kreću sa LSB pozicije u reči.

Kako je RISC1 32-bitni mikroprocesor, radi zahvata instrukcije iz memorije potrebno je uvećanje PC za četiri (jer je memorija bajt adresibilna). Izlaz PC je povezan direktno na adresni ulaz instrukcione memorije. Realizacija ovog dela RISC1 je prikazana na slici 5.5.

Dalje ćemo prvo razmotriti registarski tip instrukcija. Izlaz instrukcione memorije se vodi na registarsko polje gde se iz instrukcionog formata (vidi sliku 5.2) odgovarajuća polja vode na odgovarajuće ulaze (portove) registarskog polja. Ukoliko imamo neposredno adresiranje onda se odgovarajuća konstanta vodi na Imm blok. Prvi operand ALJ je uvek iz registra (Rs1), a drugi može biti ili iz registra (Rs2) ili konstanta pa je potreban multiplekser za izbor. Primer prve instrukcije bio ADD R20, R19, R18, a za drugu ADD R20, R19, #7. Elementi oko registarskog polja kao i ALJ su detaljnije prikazani na slici 5.6.

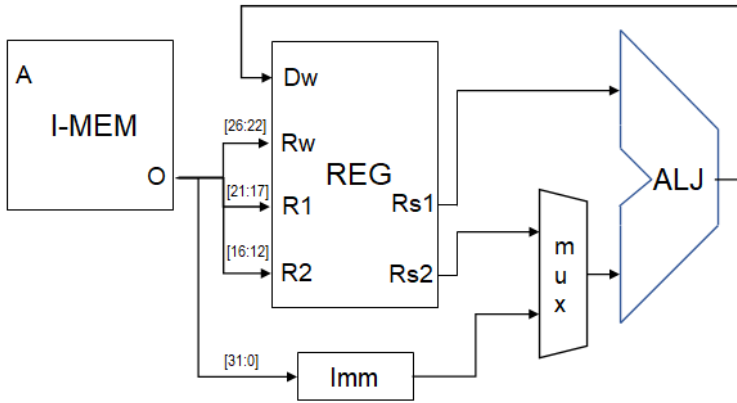
Sledeći tip instrukcija su memorijske. U dosadašnjoj arhitekturi RISC1 dodajemo i memoriju za podatke D-MEM. Na adresni ulaz se dovodi izlaz iz ALJ, a na ulaz za upis podataka (Dw) izlaz Rs2 iz registarskog polja za instrukciju upisa u

Slika 5.4: *Big-endian* i *little-endian* načini za organizaciju memorije

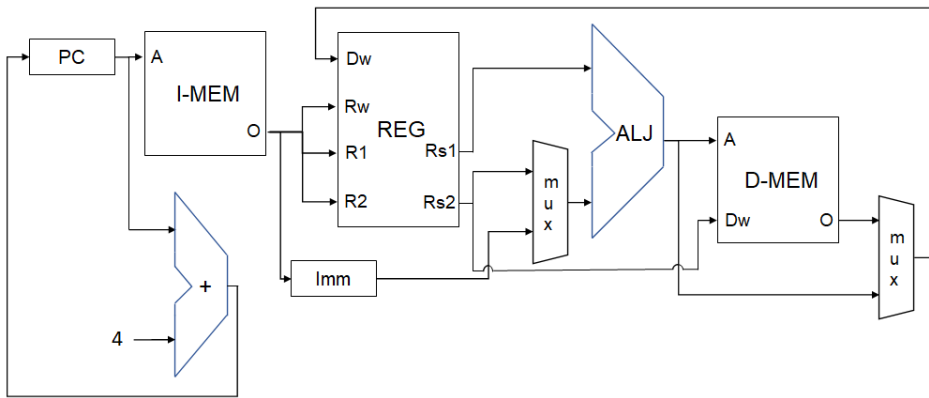
Slika 5.5: Veza programskog brojača i instrukcione memorije

memoriju ST. Povratni upis podatka je ili rezultat ALJ registrarske instrukcije ili podatak pročitani iz memorije pa je neophodan i jedan multiplekser. Na slici 5.7 su prikazani detalji.

Poslednji tip instrukcija u okviru SB tipa omogućava uslovno grananje (BEQ).

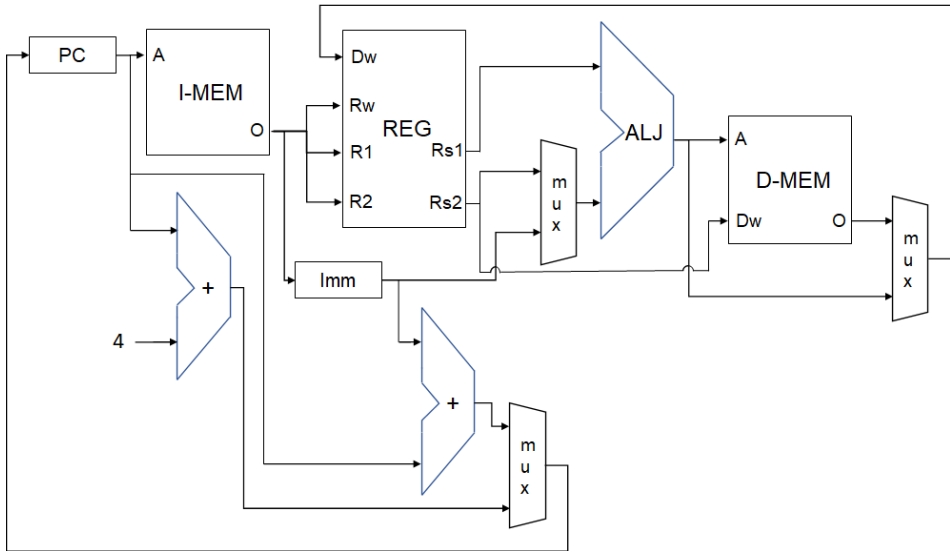


Slika 5.6: Regstarsko polje i ALJ



Slika 5.7: Dodavanje memorije za podatke D-MEM

Adresa skoka se računa u odnosu na vrednost programskog brojača, ukoliko važi $Rs1=Rs2$, tako što se dodaje 17-bitna konstanta koja je proširena na 32 bita uz očuvanje znaka. Dakle, potreban još jedan sabirač i multiplexer koji odabira da li se u PC učitava adresa uvećana za 4 ili za konstantu za skok. Celovita unutrašnja struktura bez upravljačkih signala je prikazana na slici 5.8. Detaljna analiza upravljačkih signala i upravljačke jedinice RISC1 prevazilazi okvire ovog predmeta.

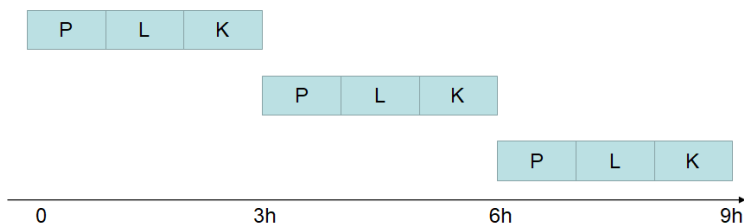


Slika 5.8: Celovita unutrašnja struktura RISC1 (bez upravljačkih signala)

5.3 Protočna obrada

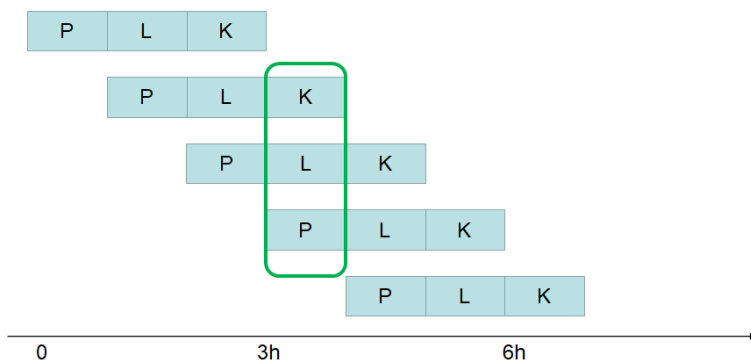
Radi objašnjenja suštine, svrhe i načina realizacije protočne obrade (eng. *pipelined*) poći ćemo od jednostavnog primera korićenja knjige. Pretpostavimo da u nekoj maloj firmi za korićenje je potrebno ukoričiti N primeraka knjige koja ima M strana. Dakle, potrebno je $M \times N$ strana i N korica. Posao se sastoji iz tri faze: 1. priprema M strana (P-priprema), 2. lepljenje M strana (L-lepljenje) i 3. dodavanje korica (K-korićenje). Radi jednostavnosti neka svaka od faza traje jednako - po jedan sat. Ukoliko se ovaj posao radi tako što se redom rade faze jedna po jedna (sekvencijalno, vidi sliku 5.9), tada je za korićenje jedne knjige potrebno tri sata, a za N knjiga $3N$ sati. Ako broj faza označimo sa F (u našem primeru je $F=3$) tada je u opštem slučaju potrebno FN sati.

Primenom principa protočne obrade je moguće ovo ubrzati tako što nakon što se obavi priprema i preda na lepljenje može već da se krene sa novom pripremom. Isto tako nakon što se obavi lepljenje i preda na korićenje, može da se radi novo lepljenje. Ovakva vrsta paralelizacije posla podrazumeva da imamo odgovarajuće resurse za to (npr. ljude i/ili mašine). U slučaju ovakve paralelizacije posle početnih 3 koraka koji su potrebni da se uposle svi resursi, tada dobijamo novu gotovu knjigu



Slika 5.9: Primer sekvencijalnog korićenja

na svakih sat vremena (označeno zelenim na slici 5.10). Potrebno vreme za korićenje N primeraka knjige je $T = N - (F-1) + 2(F-1) = N + F - 1$. Za dovoljno veliko N ovo je približno jednako N . Stoga je ubrzanje S dobijeno protočnom obradom približno jednako $S = FN/N = F$. Dakle, u slučaju idealne protočne obrade ubrzanje koje je moguće postići je jednako broju faza protočne obrade (F).



Slika 5.10: Primer protočne obrade

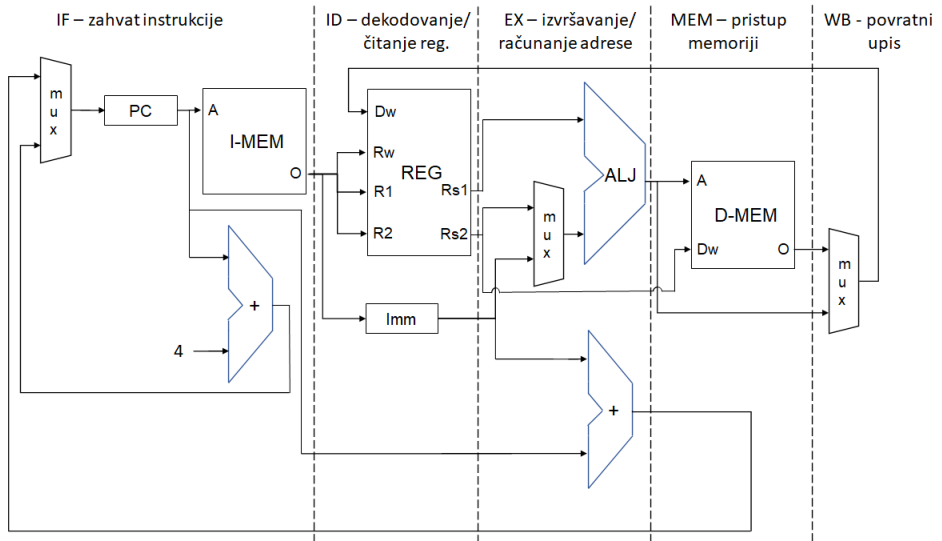
Ukoliko primer korićenja knjige zamenimo sa izvršavanjem instrukcija onda princip protočne obrade možemo primeniti i na mikroprocesore. Ako izvršenje instrukcije podelimo u 3 faze: 1. faza zahvata (umesto P), 2. faza dekodovanja (umesto L) i 3. faza izvršavanja (umesto K), onda se vidi da i izvršavanje instrukcija može imati isti tok kao na slikama 5.9 (sekvencijalno) i 5.10 (sa protočnom obradom). Naravno vreme potrebno za završetak pojedinačnih faza će biti znatno kraće (npr. 200 ps).

Realna protočna obrada uvek ima određenih problema (npr. nebalansiranost,

pojava hazarda itd.) zbog kojih se smanjuje ubrzanje koje želimo da postignemo protočnom obradom. Detaljnije bavljenje problemima kod protočne obrade i rešenjima tih problema izlazi izvan okvira ovog predmeta. U narednoj sekciji ćemo videti kako je moguće primeniti principe protočne obrade na RISC1 i napraviti verziju RISC1 mikroprocesora sa protočnom obradom.

5.4 RISC1

Da bismo mogli primeniti principe protočne obrade na mikroprocesor RISC1 potrebno je hardverske resurse organizovati na odgovarajući način da možemo da imamo više faza protočne obrade. Kod RISC arhitektura najmanji broj faza je 2 kod nekih jednostavnih mikrokontrolera, ponegde 3, a najčešće 5 do 8. Kod složenih RISC mikroprocesora ova brojka može biti i znatno veća (npr. 13 ili 19). Tada se kaže da je reč o dubokoj protočnoj obradi (eng. *deep pipeline*).



Slika 5.11: Podela RISC1 na faze

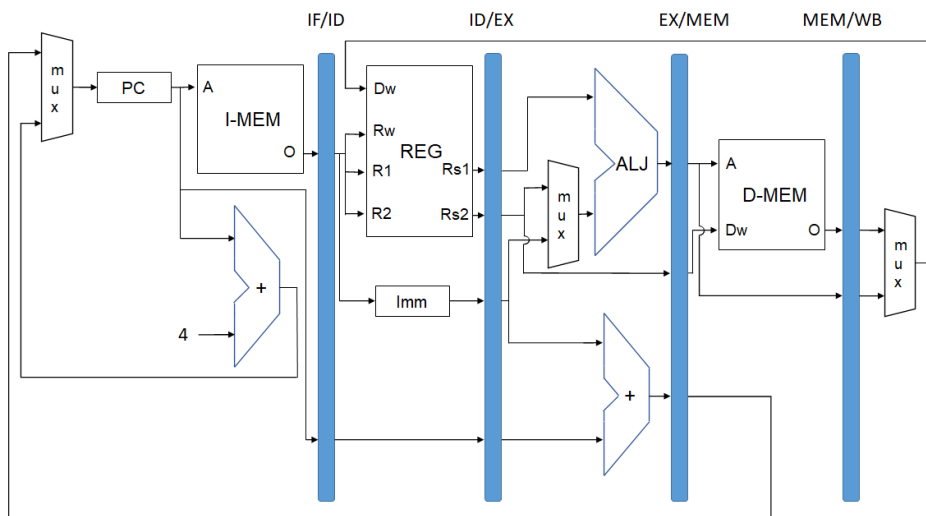
Ukoliko mikroprocesor RISC1 malo bolje grafički organizujemo i uočimo šta se u odgovarajućim celinama izvršava, onda je moguća podela na 5 faza. To su faze:

1. zahvat instrukcije (IF)

2. dekodovanje ili čitanje registara (ID)
3. izvršavanje instrukcije ili računanje adrese (EX)
4. pristup memoriji (MEM) i
5. povratni upis u registar (WB)

Ove faze su ilustrovane na slici 5.11.

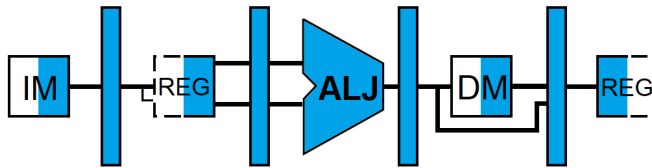
Da bi bilo moguće obrađivati podatke i prenositi iz faze u fazu potrebno je na granicama faza (isprekidane vertikalne linije na slici 5.11) dodati odgovarajuće međuregistre koji služe da čuvaju rezultat operacije pojedine prethodne faze i služi kao ulaz naredne. Kada se to uradi dobijamo arhitekturu kao na slici 5.12.



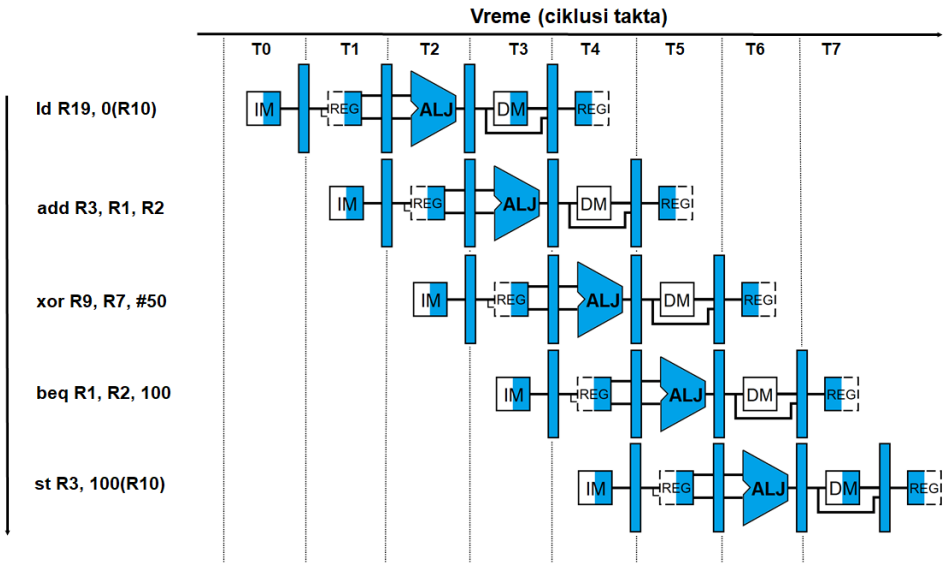
Slika 5.12: Arhitektura RISC

Ovde treba obratiti pažnju da dodati registri predstavljaju značajan hardverski resurs jer npr. širina IF/ID registra je 64 bita, a ID/EX registra je čak 128 bita! Pored elemenata DP bloka u toku daljeg detaljnog projektovanja bi trebalo obratiti pažnju na još neke detalje. Na primer, da li je povratni upis u registar pravilno realizovan na ovaj način? Dalje, koji su sve upravljački signali potrebni, kako realizovati upravljačku jedinicu itd. Ovo je ostavljeno studentima za samostalan rad ili dalje izučavanje na višim godinama studija jer izlazi izvan okvira ovog predmeta.

Prilikom analize izvršavanja više uzastopnih instrukcija unutar protočne obrade obično se koristi dva pristupa pri grafičkom prikazivanju. Prvi pristup podrazumeva multitaktni opis izvršavanja instrukcija uz upotrebu apstraktnog simbola protočne obrade prikazanog na slici 5.13. Ovaj simbol prikazuje svih pet faza protočne obrade na jednostavan grafički način. Primer analize pet uzastopnih instrukcija je dat na slici 5.14. Ovde se vidi da aritmetičko-logičke instrukcije nemaju DM fazu. Moguća je i upotreba kvadratića za svaku fazu instrukcije sa imenom faze ali je to manje ilustrativno.



Slika 5.13: Apstraktni simbol protočne obrade



Slika 5.14: Primer pet instrukcija u protočnoj obradi

Drugi pristup je tzv. jednotaktni gde se koristi crtež celokupne arhitekture mikroprocesora kao na slici 5.12 i u okviru svake faze se zapisuje koji deo koje instrukcije se izvršava. Ovaj način je najdetaljniji jer se mogu primetiti detalji koji se kod prethodnog načina nije video, ali je i grafički najsloženiji i traži najviše prostora.

Protočna obrada uvedena sa ciljem da se dobije ubrzanje ima i svojih ograničenja. Pre svega tu je problem hazarda na koji se mora obratiti pažnja. Hazardi protočne obrade dovode do neispravnog izvršenja datog programa. Postoje tri vrste hazarda: 1. hazardi podataka, 2. strukturalni hazardi i upravljački hazardi.

Hazardi podataka se javljaju kao posledica međusobne zavisnosti instrukcija. Npr. ukoliko imamo uzastopno sledeće dve instrukcije:

```
add R3,R1,R2
```

```
xor R5,R4,R3
```

tada će se javiti hazard podataka jer sadržaj R3 još nije izračunat i upisan u WB fazi u registar, a već je potreban kao operand instrukcije xor.

Strukturalni hazardi se javljaju kao posledica situacije kada dva različita izvora žele da pristupe istom resursu. Ovo je jedan od razloga zašto imamo Harvard organizaciju memorije da ne bi mogao da se desi hazard potrebe istvremenog pristupa instrukcionoj i memoriji podataka.

Upravljački hazardi se javljaju prilikom programskih grananja. Tada ukoliko se desi skok, prethodno zahvaćene instrukcije koje slede instrukciju skoka u protočnoj obradi su pogrešne.

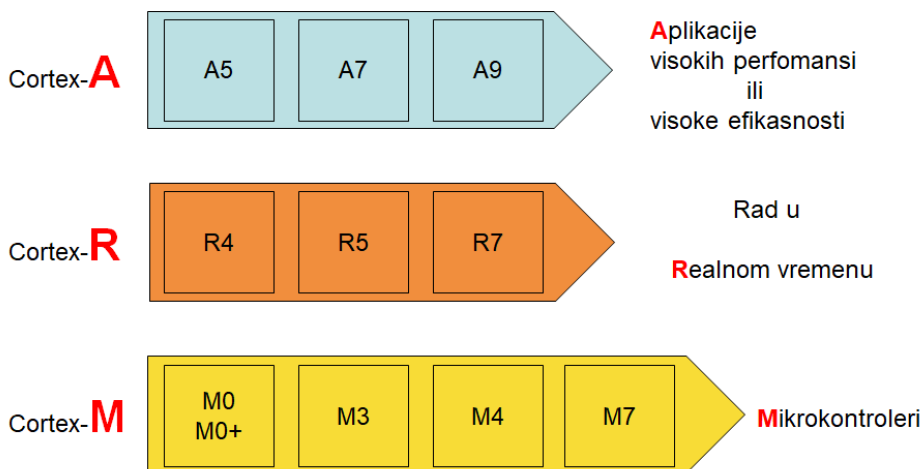
Metode rešavanja hazarda protočne obrade izlaze izvan okvira ovog predmeta.

5.5 ARM

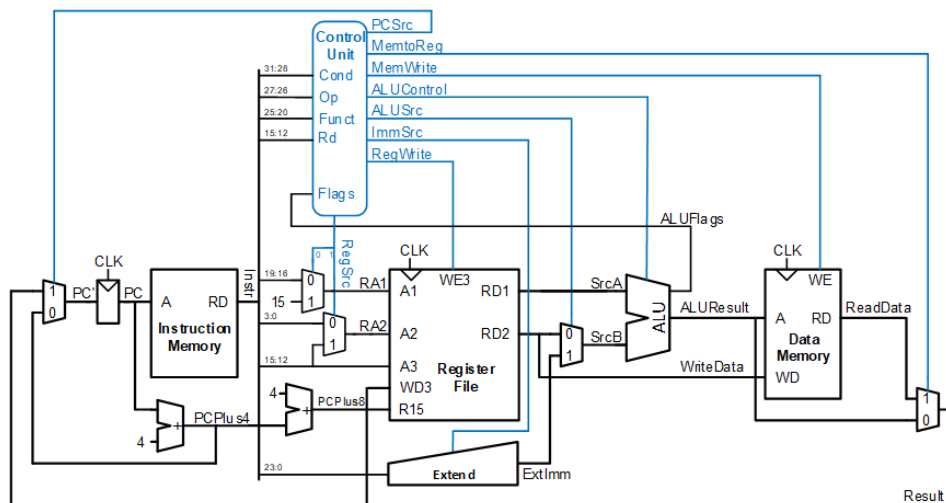
Jedno od najznačajnijih imena u svetu RISC mikroprocesora danas je kompanija ARM. Počeci ARM mikroprocesora se vezuju za 1983. godinu i kompaniju Acorn computers. Smatra se prvom komercijalnom RISC implementacijom. Kompanije Advanced RISC Machine je osnovana 1990. od strane Acorn, Apple i VLSI. Kasnije, 1998. kompanije menja ime u ARM Ltd.

Osnovni poslovni model kompanije bazira na projektovanju RISC mikroprocesora i prodaji različitih vrsta licenci i oblika intelektualne svojine za korišćenje njihovih mikroprocesora. Fabrikaciju ARM mikroprocesora rade druge kompanije i danas imamo jako mnogo proizvoda na bazi ARM mikroprocesora koje su proizvele različite kompanije (npr. Apple, Qualcomm, Broadcomm i dr.). Trenutna situacija je da ARM dominira u svetu mobilnih, portabilnih i baterijskih napajanih uređaja.

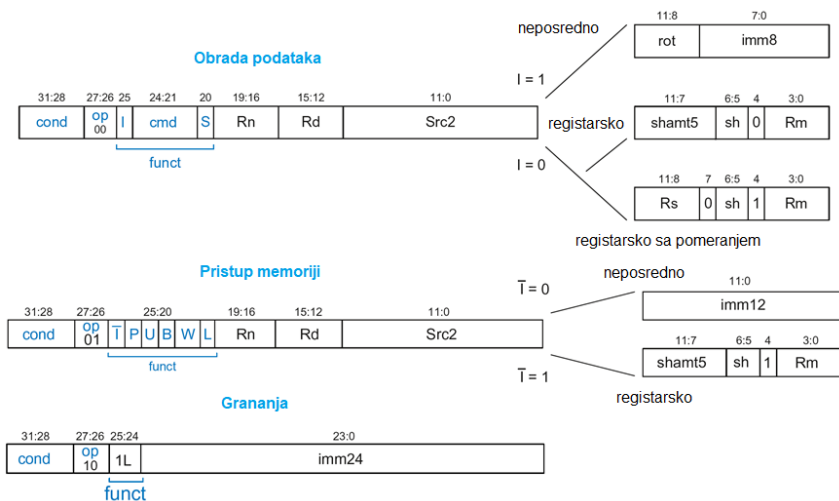
U toku svoga razvoja arhitektura skupa instrukcija ARM mikroprocesora je prešla dugačak put od verzije v1 iz 1985 do trenutno aktuelne verzije v9 iz 2021. Komercijalno ozbiljniji uspeh kreće sa pojavom verzije 4 i 5 i familija mikroprocesora ARM7/9/11. Dalji razvoj ide preko različitih serija Cortex familija mikroprocesora. Danas kompanija ARM vidi dalji put preko koncepta "ostrva". Drugim rečima kao najznačajniju specijalizaciju u primeni njihovih mikroprocesora ističu se tri oblasti: unapređena bezbednost, digitalna i vektorska obrada signala, i veštačka inteligencija i mašinsko učenje. Danas su ARM familije podeljene u tri grupe: A (aplikacioni mikroprocesori), R (mikroprocesori za rad u Realnom vremenu) i M (Mikrokontroleri). Ova podela je pokazana na slici 5.15. Aplikacioni mikroprocesori su najsnažniji mikroprocesori koji obezbeđuju mogućnosti gde su potrebne visoke performanse ili visoka efikasnost. U srednjoj kategoriji su mikroprocesori Cortex-R kategorije kojima je fokus rad u realnom vremenu. Obe ove kategorije imaju složeniju arhitekturu (npr. protočna obrada je sa 6, 8 i više faza u sličaju duboke protočne obrade) od Cortex-M. Ova grupa mikroprocesora podržava ono što se obično naziva mikrokontrolerima. Pri tome su M0 i M0+ najjeftiniji i imaju najmanju potrošnju, ali i najmanje procesorske snage uz protočnu obradu od samo 2 faze. Sa druge strane familija M7 je namenjena dogotalnoj obradi signala i ima maksimalne procesorske mogućnosti ove kategorije.



Slika 5.15: Familije ARM Cortex mikroprocesora



Slika 5.16: Arhitektura jednotaknog ARM mikroprocesora [6]

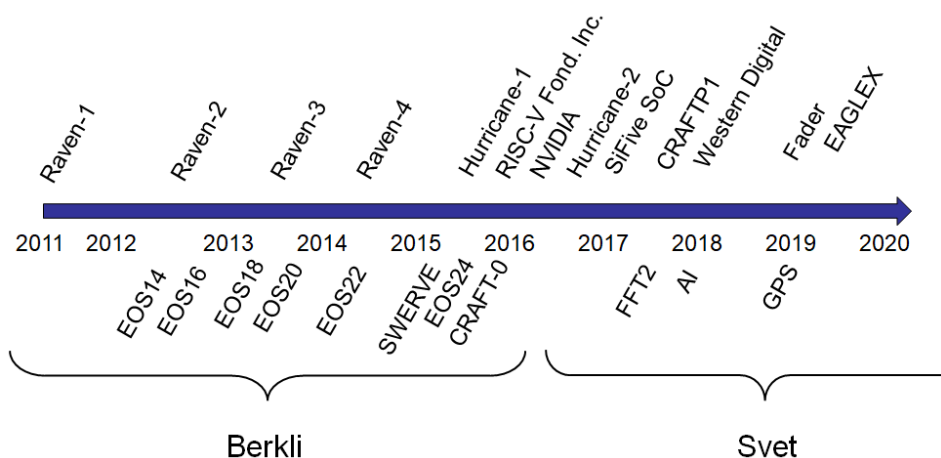


Slika 5.17: Instrukcioni formati ARMv7 mikroprocesora

otvorenog koda (eng. *open source*) tako da svako može da ga koristi bez ikakve naknade.

RISC-V je po imenu naslednik mikroprocesora RISC-I, RISC-II itd. čiji razvoj je krenuo 80-tih godina prošlog veka na Kalifornijskom Univerzitetu u Berkliju, SAD. RISC-V je počeo kao akademski projekat na istom univerzitetu 2010. godine. Nakon nekoliko godina razvoja, 2015. godine je RISC-V počeo da ima i komercijalni uspeh. Tada je osnovana RISC-V fondacija u čijem članstvu su sve veće IT kompanije današnjice.

Osim samog početka kada je za projektovanje korišćen Verilog jezik za opis harvera, na dalje je u upotrebi objektno orijentisani jezik Chisel. Od starta su radene i različite fabrikacije ovog mikroprocesora što je prikazano na slici 5.19. Ozbiljnom dosadašnjem komercijalnom uspehu RISC-V je doprinela i spin-off kompanija SiFive sa Berklija ³. Od osnivanja oni proizvode različite verzije RISC-V mikroprocesora. Slično ARM Cortex podeli na različite familije spram potreba korisnika prikazanoj na slici 5.15 i SiFive ima podelu na E (32-bitna embeded), S (64-bitna embeded) i U (64-bitna heterogena multicore) jezgra. Pri tome unutar svake kategorije postoje arhitekturne serije (u oznaci 2, 3, 5 ili 7) prilagođene potrebnom: broju faza protočne obrade (2-3/5-6/8), veličini jezgra, performansama, potrošnji itd.



Slika 5.19: Različite fabrikacije RISC-V

³<https://www.sifive.com/>

Od elemenata arhitekture RISC-V ima sledeće karakteristike. Ima bazičan 32-bitni celobrojni skup instrukcija koji je lako moguće proširiti sa nekoliko standardnih proširenja. Standardna proširenja su: množenje i deljenje, atomičke operacije, operacije u pokretnom zarezu itd. Mikroprocesor postoji u 32-bitnoj, 64-bitnoj i 128-bitnoj realizaciji. Standardan broj registara je 32 (X0-X31) pri čemu pojedini imaju specijalnu funkciju (npr. X0 uvek ima vrednost nule). Predstava podataka u memoriji je *little-endian*. Tri univerzalna principa projektovanja hardvera kojih su se držali autori RISC-V arhitekture su:

1. Jednostavnost nastaje iz regularnosti
2. Manje je brže i
3. Dobar dizajn zahteva dobre kompromise.

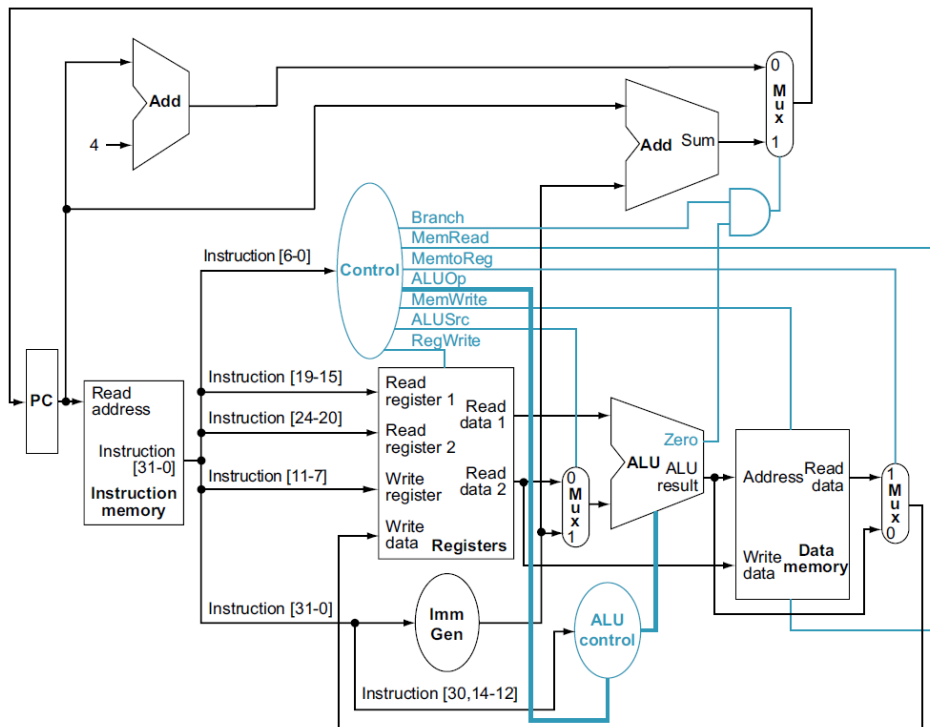
Pridržavanje ovih principa dovodi do toga da je arhitektura mikroprocesora takva da ipak se može za relativno kratko vreme razumeti i savladati. U prilog tome govori i činjenica da, primera radi, dokumentacija za arhitekturu skupa instrukcija RISC-V ima oko 250 strana, a za x86-32 preko 2000 strana. Ovo jasno govori o razlikama u složenosti ovih arhitektura.

Vrste instrukcionih formata su prikazane na slici 5.20. Očigledno je da je format prilično regularan uz malo odstupanja of funkcija pojedinih polja unutar instrukcionog formata.

| | | | | | | | | | | | | | | |
|-----------------------|----|----|----|-----|----|-----|----|--------|----|-------------|---|--------|---|-------|
| 31 | 27 | 26 | 25 | 24 | 20 | 19 | 15 | 14 | 12 | 11 | 7 | 6 | 0 | |
| funct7 | | | | rs2 | | rs1 | | funct3 | | rd | | opcode | | R-tip |
| imm[11:0] | | | | | | rs1 | | funct3 | | rd | | opcode | | I-tip |
| imm[11:5] | | | | rs2 | | rs1 | | funct3 | | imm[4:0] | | opcode | | S-tip |
| imm[12 10:5] | | | | rs2 | | rs1 | | funct3 | | imm[4:1 11] | | opcode | | B-tip |
| imm[31:12] | | | | | | | | | | rd | | opcode | | U-tip |
| imm[20 10:1 11 19:12] | | | | | | | | | | rd | | opcode | | J-tip |

Slika 5.20: Instrukcioni formati RISC-V

Realizacija jednotaktnog edukacionog RISC-V mikroprocesora prikazana je na slici 5.21, a verzije sa protočnom obradom na slici 5.22. Za više detalja pogledati odgovarajuća poglavlja iz knjige [7]. Detaljan rad sa mikroprocesorom RISC-V i njegova implementacija u VHDL jeziku za opis hardvera su predviđeni na master akademskim studijama.

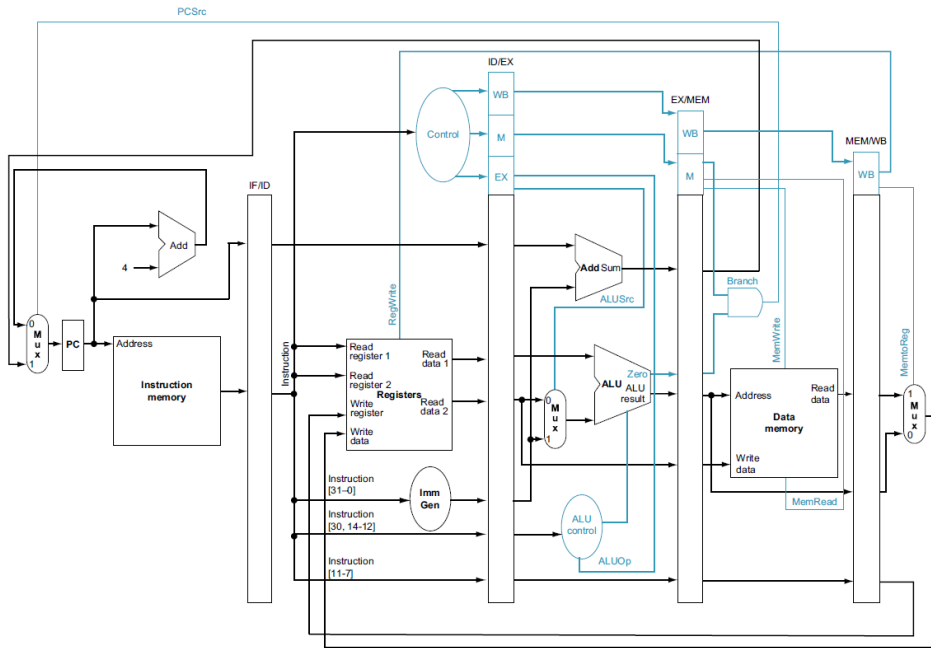


Slika 5.21: Arhitektura jednotaktnog RISC-V [7]

5.7 Principi projektovanja

Za kraj ovog poglavlja ćemo navesti nekoliko ključnih principa projektovanja savremenih arhitektura mikroprocesora. Detaljnije o tome će biti reči u predmetu Arhitekture mikroračunarskih sistema na Osnovnim akademskim studijama i u predmetu Napredni mikroprocesorski sistemi na Master akademskim studijama. Neki od principa su:

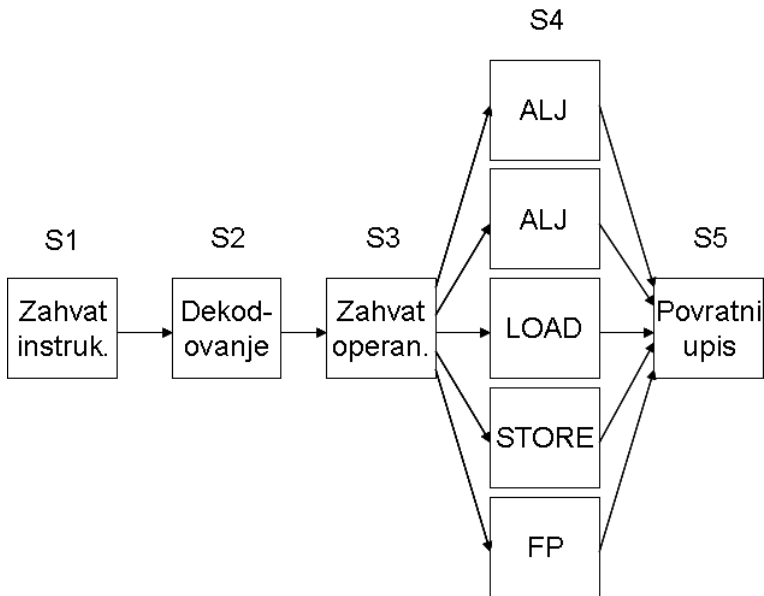
1. Sve instrukcije je poželjno da se izvršavaju direktno u hardveru. To drugim rečima znači da nema interpretiranja instrukcija. Na taj način se obezbeđuje veća brzina izvršavanja.
2. Maksimizovati brzinu pokretanja instrukcija tako da se dobije što veći broj



Slika 5.22: Arhitektura RISC-V sa protočnom obradom [7]

instrukcija po sekundi (MIPS)

- Superskalarne arhitekture koriste više funkcionalnih jedinica u paraleli. Primer sa 5 funkcionalnih jedinica u paraleli je prikazan na slici 5.23.
- Poželjno je što više registara.
- Poželjna je fiksna dužina instrukcija, mali broj polja u instrukcionom formatu, visok nivo regularnosti, što manje različitih formata i sl. Cilj je da instrukcije mogu da se jednostavno dekoduju.
- Samo Load i Store instrukcije pristupaju memoriji. Što više operacija među registrima to će brzina izvršavanja biti veća jer je manje pristupa memoriji.
- Upotreba protočne obrade je poželjna.
- Radi ubrzanja koristiti razne vrste paralelizacije.



Slika 5.23: Primer superskalarne arhitekture

9. Težnja da se sve instrukcije izvrše u jednom taktnom periodu ($CPI = 1$).

5.8 Pitanja

1. Objasniti zašto su RISC arhitekture danas dominantne na tržištu u post-PC eri.
2. Nabrojati i objasniti uobičajene arhitekturne karakteristike RISC mikroprocesora.
3. Objasniti instrukcione formate RISC1.
4. Koji tipovi adresiranja su podržani kod RISC1? Objasniti ih.
5. Uzimajući u obzir 32-bitnu arhitekturu, kako su podaci smešteni u memoriji? Nacrtati.

6. Objasniti i nacrtati razliku između *little i big endian* načina smeštanja podataka.
7. Nacrtati i objasniti arhitekturu RISC1.
8. Objasniti ulogu protočne obrade u idealnom slučaju i objasniti na nekom primeru.
9. Šta je potrebno dodati/modifikovati u arhitekturi RISC1 da bi se obezbedio rad sa protočnom obradom?
10. Koliko faza protočne obrade ima RISC1, koje su to i koja im je uloga?
11. Nacrtati i objasniti apstraktni izgled protočne obrade RISC1.
12. Šta predstavljaju hazardi protočne obrade? Navesti primer.
13. Objasniti hazardne protočne obrade.
14. Uporediti ARM i RISC-V mikroprocesore.
15. Koji su neki od principa projektovanja savremenih arhitektura CPU?
16. Objasniti šta predstavlja superskalarna arhitektura i šta se njome postiže.

Bibliografija

- [1] Arthur W. Burks, Herman H. Goldstine, and John von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument", The Institute of Advanced Study, Princeton, USA, 1946-47.
- [2] J. Biggs, J. Myers, J. Kufel et al. A natively flexible 32-bit Arm microprocessor, *Nature* 595, pp. 532–536, 2021.
- [3] F. Arute, K. Arya, R. Babbush et al. Quantum supremacy using a programmable superconducting processor. *Nature* 574, pp. 505–510, 2019.
- [4] J. Hochstetter, R. Zhu, A. Loeffler et al. Avalanches and edge-of-chaos learning in neuromorphic nanowire networks. *Nat Commun* 12, 4008, 2021.
- [5] B. B. Brey, *Intel Microprocessors*, 8th Edition, Pearson, 2009.
- [6] S. Harris, D. Hariss, *Digital Design and Computer Architecture*, Arm edition, Morgan Kaufmann, 2015.
- [7] D. Patterson, J.L. Hennessy, *Computer Organization and Design RISC-V Edition: The Hardware Software Interface*, The Morgan Kaufmann Series in Computer Architecture and Design, 2nd Edition, 2021.
- [8] M. M. Mano, C. R. Kime, T. Martin, *Logic and Computer Design Fundamentals*, Fifth edition, Pearson, 2015.
- [9] R. S. Sandige, M. L. Sandige, *Fundamentals of Digital and Computer Design with VHDL*, McGraw Hill, 2012.
- [10] L. Null, J. Lobur *Essentials of Computer Organization and Architecture*, 5th Edition, Jones and Barlett Publishers, 2018.

- [11] E.O. Hwang, Digital Logic and Microprocessor Design With VHDL with Interfacing, Cengage Learning, Second Edition, 2017.
- [12] I. Mezei "Evolution of an educational microprocessor", Computer Applications in Engineering Education, 28(5), Wiley, pp. 1265-1277, 2020.
- [13] W. Stallings, Computer organization and architecture, 9th Edition, Pearson, 2013.
- [14] M. Abd-El-Barr, H. El-Rewini, Fundamentals Of Computer Organization And Architecture, John Wiley and Sons, Inc., 2005.
- [15] C. Hamacher, Z. Vranesic, S. Zaky, N. Manjikian, Computer Organization and Embedded Systems, 6th Edition, McGraw-Hill, 2012.
- [16] Manuel Jiménez, Rogelio Palomera, Isidoro Couvertier, Introduction to Embedded Systems, Springer, 2014.
- [17] D. A. Patterson, J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, Fourth Edition, Morgan Kaufmann, 2011.
- [18] J.L. Gustafson, Moore's Law. In: D. Padua D. (eds) Encyclopedia of Parallel Computing. Springer, 2011.
- [19] S.A. McKee, R.W. Wisniewski, Memory Wall. In: D. Padua (eds) Encyclopedia of Parallel Computing. Springer, 2011.
- [20] P. Bose, Power Wall. In: D. Padua (eds) Encyclopedia of Parallel Computing. Springer, 2011.
- [21] A. Tsakyridis, T. Alexoudi, A. Miliou, N. Pleros, and C. Vagionas, "10 Gb/s optical random access memory (RAM) cell," Opt. Lett. 44, pp. 1821-1824, 2019.
- [22] W.W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques", IEEE Wescon Proc., Aug. 1970.
- [23] C.G. Bell, A. Newell, Computer structures: Readings and Examples, McGraw-Hill, 1971.
- [24] S. Brown, Z. Vranesic, Fundamentals of Digital Logic with VHDL Design, McGraw Hill, 2009.
- [25] D. L. Perry, VHDL: Programming by Example, Fourth edition, McGraw Hill, 2002.

- [26] L. Null, J. Lobur, Essentials of Computer Organization and Architecture, Jones and Bartlet Learning, 3rd Ed., 2010.
- [27] D. Patterson, J.L. Hennessy, Computer Organization and Design RISC-V Edition: The Hardware Software Interface, The Morgan Kaufmann Series in Computer Architecture and Design, 1st Edition, 2017.
- [28] S. L. Harris, D. Harris, Digital Design and Computer Architecture, RISC-V Edition, Morgan Kaufman, 1st edition, 2021.
- [29] D. Patterson, "Reduced Instruction Set Computers Then and Now" in Computer, vol. 50, no. 12, pp. 10-12, 2017. <https://doi.ieeecomputersociety.org/10.1109/MC.2017.4451206>
- [30] D. A. Patterson, J. L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, ARM edition, The Morgan Kaufmann Series in Computer Architecture and Design, 2016.
- [31] D. Kushner, "The making of arduino", IEEE Spectrum 26, 2011.
- [32] M. Banzi, "How Arduino is open-sourcing imagination", TEDtalk, Scotland, 2012. <https://www.youtube.com/watch?v=UoBUXOOdLXY> (datum pristupa: 31.01.2022.)
- [33] Sparkfun, "Arduino shields v2", <https://learn.sparkfun.com/tutorials/arduino-shields-v2>, (datum pristupa: 01.02.2022.)
- [34] Arduino Uno Rev3 schematic, https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf (datum pristupa: 02.02.2022.)
- [35] Arduino software, <https://www.arduino.cc/en/Main/Software> (datum pristupa: 02.02.2022.)
- [36] 74HC595 datasheet, Philips, 1998, <https://www.arduino.cc/en/uploads/Tutorial/595datasheet.pdf> (datum pristupa: 03.03.2022.)
- [37] <https://www.ti.com/lit/ds/symlink/sn74hc165.pdf> (datum pristupa: 03.03.2022.)
- [38] ATmega328 datasheet, Atmel, 2015, https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (datum pristupa: 15.03.2022.)

- [39] Tutorials 83, Fundamentals of RS-232 serial communications, Analog Devices, 2001. <https://www.maximintegrated.com/en/design/technical-documents/tutorials/8/83.html> (datum pristupa: 31.03.2022.)