

Osnovi mikroprocesorskih i mikrokontrolerskih sistema - Osnove arhitektura

prof. dr Ivan Mezei

25. feb. 2022.

Glava 2

Osnove arhitektura mikroprocesora, mikrokontrolera i mikroračunarskih sistema

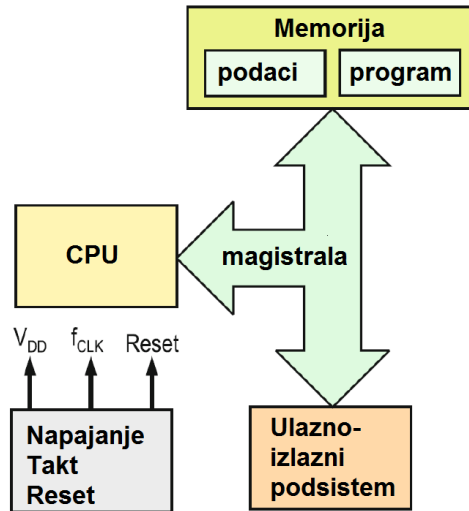
U ovom poglavlju će biti prikazane bazične postavke vezane za arhitekturu mikroračunara, mikroprocesora i mikrokontrolera. Pored toga će biti prikazani načini merenja performansi mikroprocesora i istaknuti neki ograničavajući faktori.

2.1 Arhitektura mikroračunarskih sistema

Mikroračunarski sistem (skraćeno-mikroračunar) je sistem koji se sastoji od odgovarajućih komponenata koje omogućavaju računanje. Osnovne operacije koje mikroračunar vrši su: pristup podacima, transfer podataka i operacije nad podacima. Generalna blok šema mikroračunara prikazana je na slici 2.1. Ovaj model mikroračunarskog sistema naziva se i *Von Neumann*-ov model u čast ovom naučniku koji je dao neke fundamentalne postavke današnjeg računarstva [1].

Minimalna konfiguracija mikroračunarskog sistema sastoji se iz:

1. Centralne procesorske jedinice tj. centralnog procesora (uobičajena oznaka CPU je skraćenica od engleskog *Central Processing Unit*),



Slika 2.1: Struktura mikroračunara

2. Memorije,
3. Ulazno/izlaznog podsistema, i
4. Magistrale (magistrala).

Komponente mikroračunarskog sistema međusobno su povezane pomoću skupa linija, grupisanih prema njihovoj funkciji, poznatih pod nazivom sistemske magistrale. Dodatne komponente obezbeđuju napajanje, generisanje takta, reset sistema i sl.

Bez obzira na način realizacije, svaka od komponenti mikroračunarskog sistema ima tačno definisanu funkciju:

- Centralna procesorska jedinica (**mikroprocesor**) predstavlja “srce” mikroračunarskog sistema. Zadužena je za preuzimanje instrukcija iz memorije, njihovo dekodovanje i, shodno tipu instrukcije, izvođenju odgovarajućih operacija nad podacima i/ili periferijama iz ulazno/izlaznog podsistema kako bi se obezbedila željena funkcionalnost čitavog sistema.
- **Memorije** su mesto gde su smešteni programi i podaci kojima pristupa CPU. Postoje dva osnovna tipa memorije u sistemu: programska i memorija

za podatke. Pored toga, postoje specijalizovane memorije kao što su virtualna memorija, keš memorija i druge. Detaljno o memorijama će biti obrađeno u 4. poglavlju.

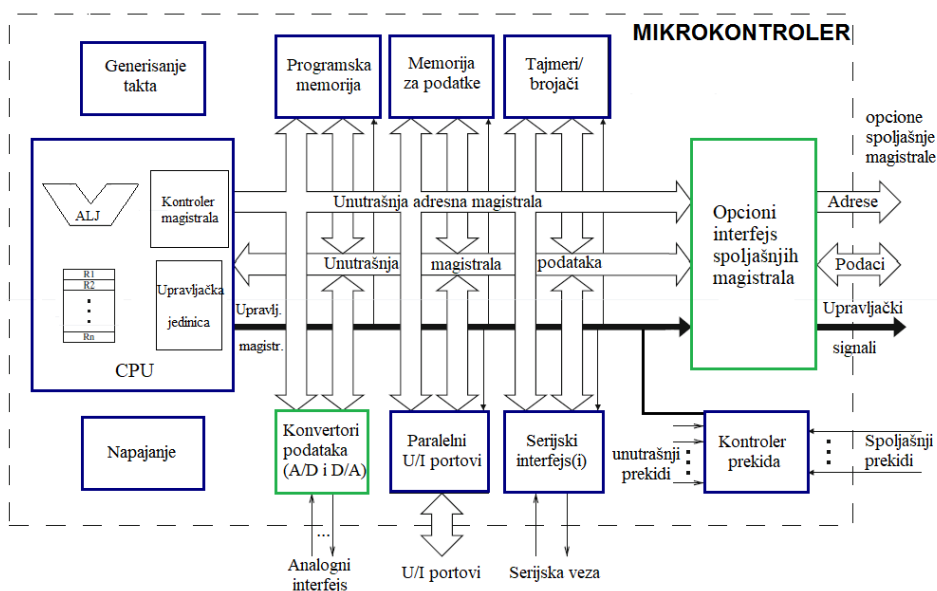
- **Ulazno/izlazni podsistem** predstavlja podsistem koji služi za povezivanje CPU sa okruženjem. On sadrži sve komponente, odnosno periferije, koje omogućavaju da CPU razmenjuje informacije sa ostalim uređajima i sistemima u svojem okruženju.
- **Sistemske magistrale** su skup linija koje povezuju CPU, memoriju i ulazno/izlazni podsistem. Različite grupe linija obavljaju različite funkcije: adretna magistrala, magistrala podataka ili kontrolna magistrala.

Komponente mikroračunarskog sistema mogu biti realizovane na razne načine. Moguće ih je realizovati korišćenjem većeg broja integrisanih kola ili modula raspoređenih na štampanoj ploči. Mogu biti sve smeštene na jednom integrisanom kolu i tada je reč o mikrokontroleru. Danas se većina embeded sistema bazira na korišćenju mikrokontrolera. Primer blok šeme mikrokontrolera prikazan je na slici 2.2.

Ovde pored mikroprocesora vidimo i druge hardverske blokove koji su uobičajeni kod mikrokontrolera. To su pre svega blok za napajanje i blok za generisanje takta. Potom memorijski blokovi (za programsku i memoriju podataka). Za povezivanje sa spoljašnjim periferijama imamo minimalno paralelne ulaze/izlaze i serijski interfejs, a za rad sa prekidima je potreban kontroler prekida. Opciono se mogu naći i sprežni interfejs za spoljašnju magistralu kao i konvertori podataka (A/D i D/A). O većini ovih blokova će biti reči u kasnijim poglavljima ove knjige.

U slučaju ostalih mikroračunarskih sistema CPU se nalazi samostalno na integrisanom kolu, a da bi se sistem oformio ostale komponente se dodaju posebno, spolja u odnosu na CPU. Najpoznatiji sistemi bazirani na mikroprocesorima su personalni računari (eng. *Personal Computer* - PC), radne stanice, serveri, klasteri, superkompjuteri, pametni telefoni, tableti i danas sve popularniji miniračunari (eng. *single-board computers*, npr. *Raspberry Pi*¹). Neki od najpoznatijih proizvođača mikroprocesora su: Intel, Arm, AMD, Samsung, Siemens itd. U oblasti personalnih računara, radnih stanica i sl. se posebno ističe Intel, dok se kompanija ARM sa projektovanjem svojih mikroprocesora ističe u embeded svetu. Pri tome je poslovni model ove dve kompanije potpuno različit jer Intel ima kompletnu podršku od projektovanja do fabrikacije u svojim fabrikama, dok ARM radi pre svega projektovanje mikroprocesora i to prodaje drugima koji će da rade fabrikaciju.

¹<https://www.raspberrypi.org/>

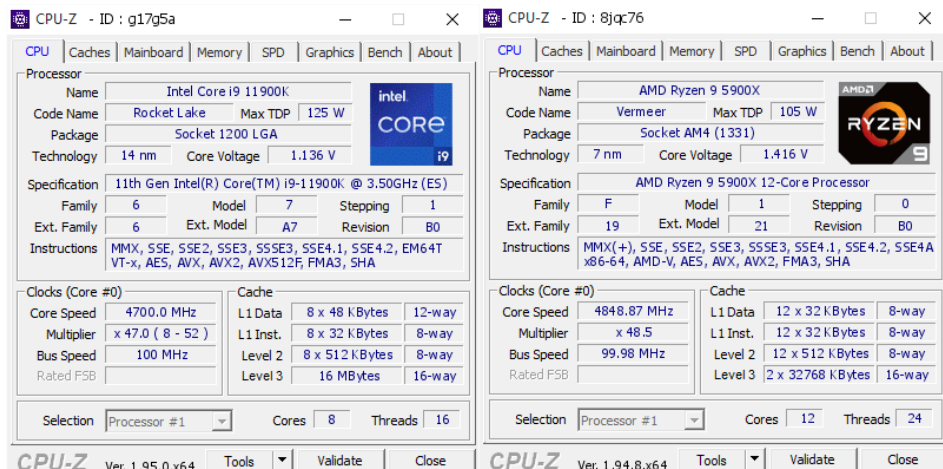


Slika 2.2: Blok šema mikrokontrolera

Razvojni put Intelovih mikroprocesora je dugačak:

- Prvi mikroprocesor kompanije Intel, 4-bitni 4004, proizveden je 1971. koristeći 10 μm tehnologiju, radio je na 108 kHz i sastojao se od 2.300 tranzistora.
- Prvi 32-bitni procesori (x386) su proizvedeni 1985. koristeći 1 μm tehnologiju, radili je na 16-33 MHz i sastojali se od 275.000 tranzistora.
- Prvi 64-bitni procesori su se pojavili 10tak godina kasnije, radili na 100 MHz i više, i sastojali se od nekoliko miliona tranzistora izrađenih u nanometarskoj tehnologiji.
- Danas imamo multijezgarne mikroprocesore koji rade na nekoliko GHz, a proizvode se u 14 nm, 10 nm i 7nm tehnologiji sa tendencijom prelaska na 5 nm. Detalji u vezi savremenih mikroprocesora Intel i9 i AMD Ryzen 9 su prikazani na slici 2.3 (u programu CPU-Z²).

²<https://www.cpubid.com/software/cpu-z.html>



Slika 2.3: Detalji mikroprocesora Intel i9 i AMD Ryzen 9

ARM mikroprocesori su se pojavili 80-tih godina prošlog veka i imali su RISC (eng. *Reduced Instruction Set Computer*) arhitekturu za razliku od Intelovih mikroprocesora koji su tada bili tipični predstavnici CISC (eng. *Complex Instruction Set Computer*) arhitekture. Više o RISC mikroprocesorima će biti reči u 5. poglavlju.

2.2 Podela arhitektura mikroprocesora

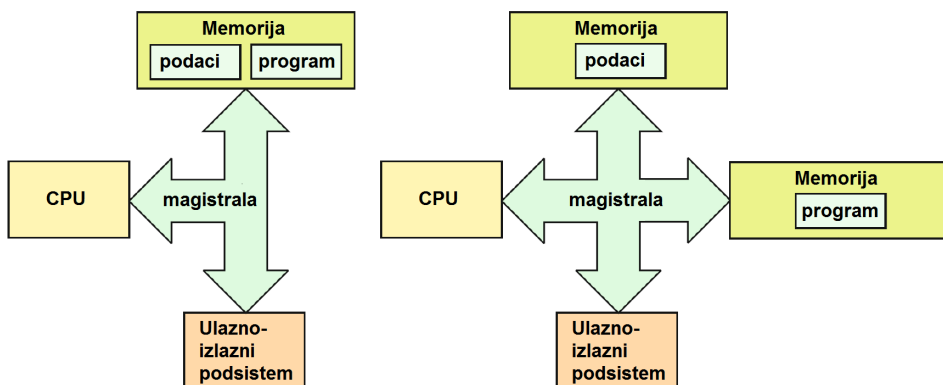
Arhitekture mikroprocesora se mogu podeliti na osnovu različitih kriterijuma, odnosno stanovišta. Ovde će biti prikazane podele po: organizaciji memorije, načinu i vremenu potrebnom za izvršavanje instrukcija, realizaciji magistrala, i skupu instrukcija.

Sa stanovišta **organizacije memorije** arhitekture mikroprocesora se dele u dve velike grupe:

- *Von-Neumann*-ova (ili *Princeton*) arhitektura i
- *Harvard* arhitektura.

Istorijski gledano prvo se pojavila *Von-Neumann*-ova arhitektura (čita se Fon Nojmanova), i kod nje se i program i podaci nalaze u *istoj* memoriji (levi deo slike

2.4). Harvard arhitektura se odlikuje fizički odvojenom memorijom podataka od programske memorije (desni deo slike 2.4). Drugim rečima postoje (bar) dve *različite* memorije u sistemu. Harvard arhitektura je danas dominantna u računarskim sistemima.



Slika 2.4: Von Neumann-ova i Harvard arhitektura memorije

Sa stanovišta **načina i vremena potrebnog za izvršavanje instrukcija** mikroprocesore delimo na jednotaktne (cela instrukcija se izvršava u toku jednog takta), više-taktne (instrukcija se izvršava u više taktova i obično se sastoji iz više mikroinstrukcija) i mikroprocesore sa protočnom obradom (eng. *pipeline*).

Putem magistrala mogu da se prenose podaci, adrese i upravljački signali, pa tako možemo da imamo magistralu podataka, adresnu magistralu i upravljačku magistralu. Sa stanovišta **realizacije magistrala** imamo sisteme sa jednom, dve ili više magistrala. Sistemi sa jednom magistralom su najjednostavniji i takve magistrale podržavaju prenos u oba smera. Takva magistrala obično objedinjuje sve potrebne signale. Ukoliko imamo bar dve magistrale podataka tada je moguća podela tih magistrala tako da jedna bude ulaznog, a druga izlaznog tipa ili da je svaka sa svojom specijalizacijom (npr. jedna za adrese i podatke, a druga upravljačka). Za više od dve magistrale je moguća dodatna specijalizacija pojedinih magistrala. Pri tome treba imati u vidu da je sprežna logika za povezivanje na magistralu značajno kompleksnija u slučaju kada je u sistemu više magistrala.

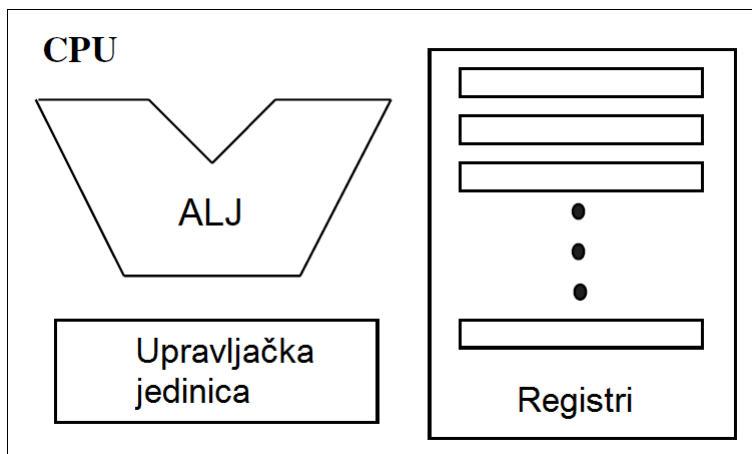
Sa stanovišta **skupa instrukcija** arhitekture se dele na RISC i CISC, pomenute u prethodnoj sekciji. U današnje vreme se komercijalno skoro isključivo koristi RISC arhitektura [23]. Osnovna osobine RISC arhitektura je da su instrukcije što jednostavnije, da ih ima mali broj i da se brzo izvršavaju. Na taj način

se performanse sistema mogu poboljšati. Iako CISC naizgled privlači činjenicom da u jednoj instrukciji možemo puno toga uraditi, pokazuje se da to često bude sporo. Pri tome, RISC verzija iste instrukcije, iako često ima mnogo više instrukcija, ukupno bude brže rešenje.

Neke od prethodnih tema i novih pojmova će biti dodatno razrađene u narednim sekcijama i poglavljima, a za više detalja čitalac se upućuje na odgovarajuća poglavlja u knjigama [6], [7] i [8].

2.3 Arhitektura mikropcesora

Centralna procesorska jedinica (CPU), tj. mikropcesor, predstavlja “srce” svakog mikroračunarskog, mikrokontrolerskog i embeded sistema. Zadužena je za izvršavanje instrukcija programa koje vrše operacije nad podacima i u njoj se instrukcije “transformišu” u signale i hardverske akcije koje upravljaju radom celog sistema. Na slici 2.5 prikazan je minimalan skup komponenata koje definišu arhitekturu CPU. U taj skup spadaju hardverske i softverske komponente.

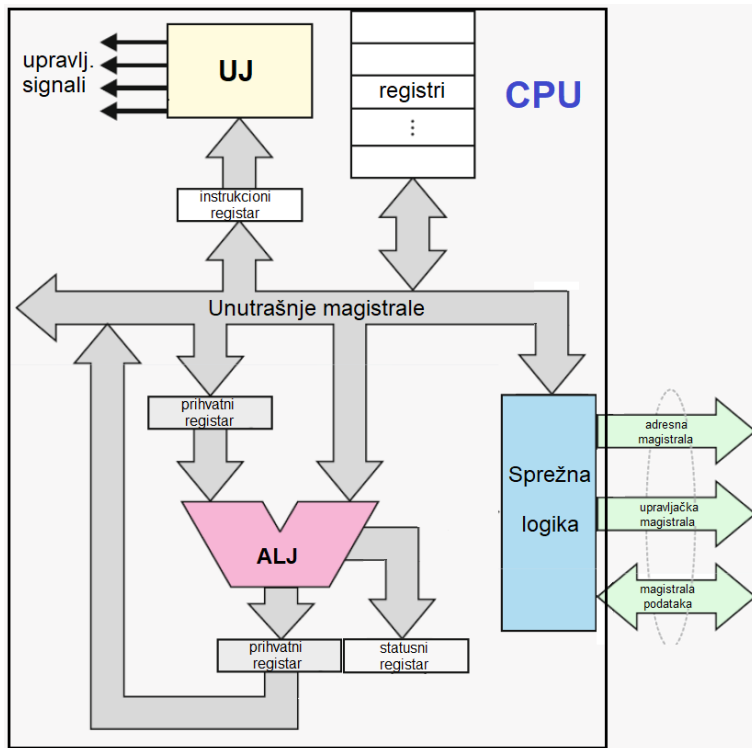


Slika 2.5: Minimalan skup unutrašnjih blokova CPU

2.3.1 Hardverske komponente arhitekture mikropcesora

Hardverske komponente arhitekture mikropcesora su:

- Skup registara
- Aritmetičko logička jedinica (ALJ, eng. *Arithmetic Logic Unit* - ALU)
- Upravljačka jedinica (UJ, eng. *Control unit* - CU),
- Unutrašnje magistrale CPU, i
- Logika za implementiranje spreznog interfejsa prema magistralama izvan CPU (sprezna logika, vidi sliku 2.6).



Slika 2.6: Hardverske komponente arhitekture CPU

Skup hardverskih komponenata koje izvršavaju obradu podataka unutar CPU

čini podsistem za obradu i tokove podataka (eng. *datapath*) procesora³. Podsistem za obradu i tokove podataka uključuje:

- Aritmetičko logičku jedinicu,
- Unutrašnje registre za smeštanje podataka,
- Unutrašnje magistrale podataka, i
- Ostale funkcionalne jedinice, kao što su npr. jedinice za obradu podataka predstavljenih u pokretnom zarezu, hardverski množači/delitelji, itd.

Hardverske komponente koje upravljaju radom mikroprocesora čine njegov upravljački podsistem (eng. *control path*). Upravljački podsistem uključuje:

- Upravljačku jedinicu koja generiše upravljačke signale,
- Logiku za implementiranje spreznog interfejsa prema magistralama, i
- Komponente zadužene za takt i vremensku sinhronizaciju.

U narednim sekcijama će biti više reči o hardverskim komponentama arhitekture mikroprocesora.

2.3.2 Softverske komponente arhitekture mikroprocesora

Softverske⁴ komponente arhitekture mikroprocesora su:

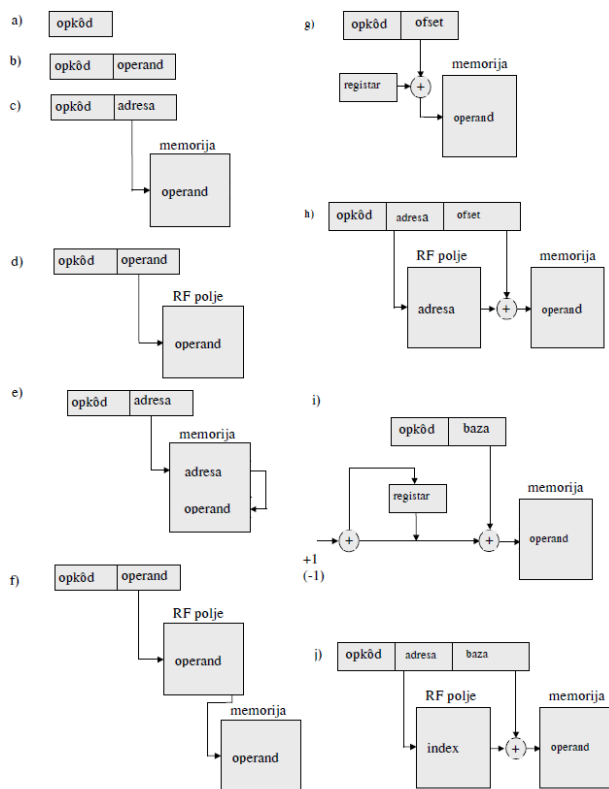
- Skup instrukcija i
- Načini adresiranja.

Skup instrukcija sačinjavaju različite grupe ili tipovi instrukcija (npr. prenoša podataka, aritmetičke, instrukcije grananja itd.). Pod pojmom instrukcija se podrazumeva elementarna celovita izvršna komanda datog mikroprocesora koja najčešće služi da vrši neku operaciju nad nekim operandom (operandima). Operandi se nalaze u memoriji ili u registrima. Da bi se omogućila različitost pristupa operandima i kasnije programiranje za dati mikroprocesor, obično je podržano nekoliko načina adresiranja. Pod pojmom adresiranje se misli na način dolaženja do podatka, tj. operanda za datu instrukciju. Na slici 2.7 su prikazani različiti načini adresiranja (a. implicitno, b. neposredno, c. memorijsko direktno, d. registarsko

³U literaturi sa našeg govornog područja se često koristi i termin 'staza podataka'.

⁴Pored termina 'softver' koji je engleskog porekla koristi se i naš termin 'programaska podrška'.

direktno, e. memorijsko indirektno, f. registarsko indirektno, g. registarsko preko implicitnog registra (obično je to PC) tj. PC relativno, h. registarsko relativno, i. indeksno sa autoinkrementiranjem/dekrementiranjem, j. bazno-indeksno). Neke od ovih tipova ćemo detaljnije objasniti kasnije. Detaljna analiza svih načina adresiranja prevazilazi okvire ove knjige.



Slika 2.7: Različiti tipovi adresiranja

2.4 Registri mikrop procesora

Registri mikrop procesora obezbeđuju privremeni smeštaj za podatke, adrese i upravljačke informacije. Predstavljaju najbrže memorijske jedinice, ali su isto-

vremeno vrlo malog kapaciteta. Sadržaj registara gubi se prilikom isključivanja napajanja. Registri mogu se podeliti u dve velike grupe:

- **Registri opšte namene** (eng. *General Purpose Registers* - GPR) nemaju specifičnu funkciju unutar procesora. Mogu se koristiti za čuvanje podataka, vrednosti promenljivih, adresa, itd., po volji programera. U zavisnosti od arhitekture, CPU može da sadrži od jednog do nekoliko desetina GPR registara.
- **Registri posebne namene** (eng. *Special Functions Registers* - SFR) imaju tačno definisanu namenu unutar procesora. Većina procesora sadrži barem sledeće SFR registre: instrukcioni registar (IR), programski brojač (PC), pokazivač steka (SP) i statusni registar (SR).

Instrukcioni registar (IR) čuva instrukciju koju je potrebno izvršiti pomoću mikroprocesora. Proces prebacivanja instrukcije iz programske memorije u IR registar zove se faza prihvata instrukcije (eng. *instruction fetch*). Kod jednostavnih mikroprocesora, IR može da čuva kôd samo jedne instrukcije. Kod složenijih mikroprocesora moguće je istovremeno izvršavanje većeg broja instrukcija ukoliko imaju više funkcionalnih jedinica.

Programski brojač (eng. *Program Counter* - PC) sadrži adresu naredne instrukcije koju je potrebno prebaciti iz memoriju u mikroprocesor radi njenog izvršavanja. Ponekad se naziva i pokazivač instrukcija (eng. *Instruction Pointer* - IP). Svaki put kada se tekuća instrukcija dekoduje i izvrši, upravljačka jedinica ažurira sadržaj PC registra kako bi on pokazivao na memorijsku lokaciju u kojoj je smeštena naredna instrukcija koju je potrebno izvršiti. Ažuriranje sadržaja PC registra najčešće podrazumeva njegovo uvećanje za unapred određenu konstantnu vrednost, ali u slučaju izvršavanja programskih skokova ili poziva podprograma sadržaj PC registra se menja potpuno novom vrednošću (adrese). Širina PC registra obično određuje maksimalnu veličinu memorije koju procesor može da adresira. Sadržaj PC registra obično ne može direktno da se menja od strane programera.

Pokazivač steka (eng. *Stack Pointer*, SP) pokazuje na "vrh" stek memorije. Stek je specijalizovani memorijski segment koji služi za privremeno čuvanje podataka, kod koga se podacima može pristupati samo u strogo kontrolisanoj sekvenci. Stek je nezamenjiv prilikom implementacije podprograma i prekidnih rutina. Većina procesora nema hardverski definisan stek već, kroz odgovarajući skup specijalizovanih instrukcija, omogućuje rad sa korisnički definisanim stekom unutar memorije podataka.

Statusni registar (SR), poznat i pod imenom programska statusna reč (eng. *Program Status Word* - PSW), fleg registar (eng. *Flag Register*) ili indikatorski

registar, sadrži skup indikatorskih bita, flegova, kao i dodatnih bita koji se odnose na ili kontrolišu status mikroprocesora. Pod indikatorom podrazumevamo jedan bit čija vrednost oslikava prisustvo ili odsustvo odgovarajućeg stanja. Broj indikatorskih bita zavisi od složenosti mikroprocesora. Većina indikatorskih bita oslikava stanje procesora neposredno nakon izvršavanja neke operacije na ALJ, iako u opštem slučaju korisnik može da manipuliše njihovim sadržajem putem programa. Indikatorski bitovi koji se najčešće mogu naći unutar procesora su: indikator nule (eng. *Zero Flag* - ZF), indikator prenosa (eng. *Carry Flag* - CF), indikator znaka (eng. *Sign Flag* - SF), indikator prekoračenja opsega (eng. *Overflow Flag* - OF), indikator prekida (eng. *Interrupt Flag* - IF), indikator parnosti (eng. *Parity Flag*) i drugi.

2.5 Aritmetičko logička jedinica mikroprocesora

Aritmetičko-logička jedinica omogućava realizaciju aritmetičkih i logičkih instrukcija. Na osnovu zahteva i specifikacije se određuje koje aritmetičke i logičke operacije će podržati. Od aritmetičkih operacija najčešće su to sabiranje i oduzimanje. Kod sabiranja se može pojaviti prenos ukoliko je rezultat veći od dimenzija podatka. Kod oduzimanja se prenos može javiti ukoliko je prvi broj manji od drugog.

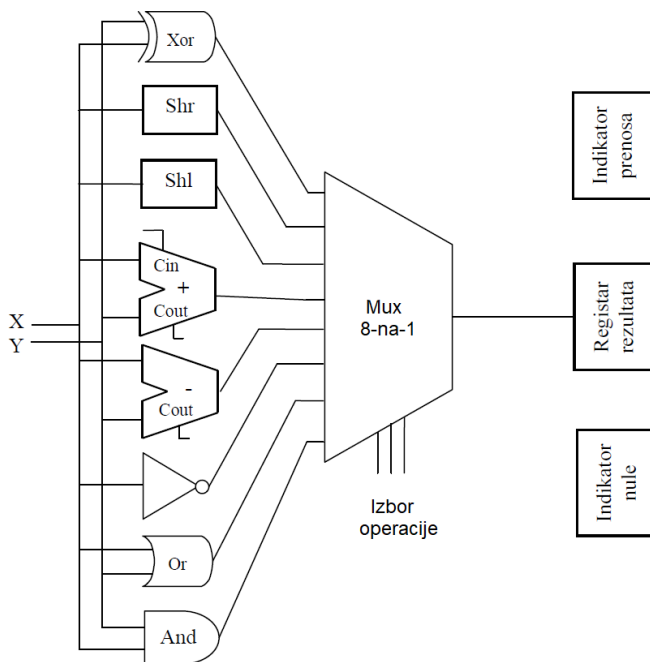
Od operacija pomeranja se često koristi pomeranje u levo ili u desno za jedan bit. Pomereni bit je obično potrebno postaviti u indikator prenosa. Ove operacije se mogu realizovati uz pomoć pomeračkih registara ili odgovarajućim slaganjem signala u prihvatni registar rezultata. Mogu da se koriste i operacije rotacija.

Primeri često korišćenih logičkih operacije su negacija, i, ili i ekskluzivno ili. Sve ove operacije se realizuju hardverski, pomoću logičkih kola.

Na osnovu rezultata operacija koje se dobijaju kao izlaz odgovarajuće kombinacije logike, se formira stanje statusnog registra, a rezultat se smešta u prihvatni registar rezultata ili direktno dalje. Primeri statusnih bita su: indikator prenosa, indikator nule, indikator parnosti, indikator preteka (eng. *overflow*) itd. Primer jednostavne ALJ je prikazan na slici 2.8.

2.6 Upravljačka jedinica mikroprocesora

Najsloženiji deo mikroprocesora predstavlja upravljačka jedinica. Ona koordinira radom celog sistema i njeno projektovanje zahteva posebnu pažnju. Svrha upravljačke jedinice je da generiše upravljačke signale u odgovarajućim trenuci-



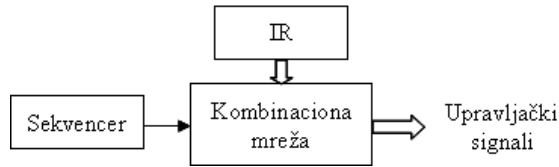
Slika 2.8: Primer jednostavne ALJ sa osam operacija i dva indikatora

ma na bazi instrukcije koja se trenutno izvršava (tj. nalazi se u instrukcionom registru), kao i statusnog registra.

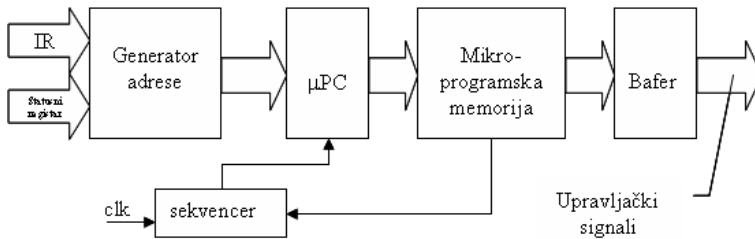
Upravljačka jedinica se može realizovati na tri načina:

1. „ožičeno“ (eng. *hardwired*),
2. mikroprogramski, ili
3. preko realizacije konačnog automata.

Takozvana ožičena realizacija upravljačke jedinice se danas ređe koristi u odnosu na druge realizacije. Razlog je što se ovakav tip upravljačke jedinice realizuje kombinacionom logikom izvedenom diskretnim logičkim kolima što je danas retkost. Blok šema ovakve upravljačke jedinice prikazana je na slici 2.9. Upravljački signali se generišu na izlazu kombinacione mreže na bazi sadržaja instrukcionog registra u kojem se nalazi tekuća instrukcija, a sekvencer upravlja radom.



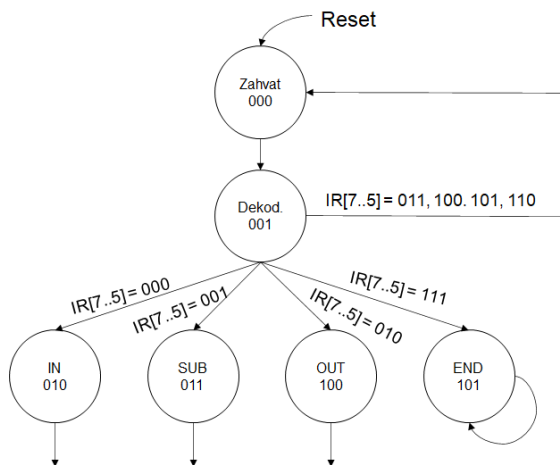
Slika 2.9: Primer "ožičene" realizacije upravljačke jedinice



Slika 2.10: Blok šema mikroprogramske upravljačke jedinice

Kod mikroprogramske upravljačke jedinice se podrazumeva da je svaka instrukcija skup više *mikroinstrukcija*. Stoga je program skup mnoštva mikroinstrukcija koje se izvršavaju određenim redosledom. Struktura mikroprogramske upravljačke jedinice prikazana je na slici 2.10. Ulazni signali upravljačke jedinice su izlazi instrukcionog i statusnog (indikatorskog) registra. Prvi blok je generator adrese (koji se nekada naziva i mapper) koji određuje početnu adresu izvršavanja za konkretno prihvaćenu instrukciju u IR registru. Ova adresa se upisuje u mikroprogramski brojač (uPC) koji adresira mikroprogramsku memoriju koja je po pravilu ROM tipa (eng. *Read Only Memory*). Detaljno o vrstama memorija i njihovim osobinama će biti obrađeno u poglavlju 4. Adresa se može odrediti i tako što se uzima kao deo upravljačke reči iz mikroprogramske memorije, što zavisi od toga koja se mikroinstrukcija trenutno izvršava. Ovim upravlja blok koji se naziva sekvencer. Glavni blok koji je potrebno projektovati je mikroprogramska memorija. Postupak se svodi na slaganje odgovarajućih bita koji reprezentuju odgovarajući upravljački signal u mikroprogramsku memoriju na osnovu mikroinstrukcije koja se izvršava i dodavanje bita koji određuju narednu adresu. Nakon izvršene instrukcije se vrši povratak na mikroinstrukciju koja predstavlja početak faze zahvata naredne instrukcije.

Upravljačku jedinicu je moguće realizovati i preko automata sa konačnim brojem stanja. Ovaj način realizacije je danas dosta čest pogotovo kada se upravljačka jedinica realizuje u programabilnoj logici gde se automati veoma lako implementiraju. Primer jednostavne upravljačke jedinice zadate dijagramom stanja dat je na slici 2.11.



Slika 2.11: Upravljačka jedinica zadata dijagramom stanja

2.7 Programerski model mikroprocesora

Sa tačke gledišta programera jasno je da programer mora da poznaje softverske komponente arhitekture mikroprocesora (skup instrukcija i načine adresiranja). Sa druge strane vrlo verovatno mu nije potrebna informacija o naponu napajanja ili kućištu CPU. Međutim da bi uspešno mogao da programira potrebno je da poznaje *programerski model* mikroprocesora.

Programerski model mikroprocesora obuhvata:

- registre,
- skup instrukcija,
- načine adresiranja i

- formate podataka.

Pod pojmom registri se ovde podrazumevaju svi registri mikroprocesora koji su vidljivi programeru. Pojam 'vidljivi' odnosi se na registre kojih programer mora da bude svestan u toku pisanja programa. Tipičan primer su registri opšte namene kojima mikroprocesor pristupa u toku izvršavanja instrukcija i u kojima programer direktno navodi imena registara. Naravno, potrebno je poznavanje i pojedinih registara koje ne navodi direktno u instrukciji. Na primer, instrukcije za rad sa stekom mogu da u svom formatu ne navode ime pokazivača steka, ali programer mora da zna da te instrukcije menjaju sadržaj tog registra i na koji način se vrši ta promena.

O skupu instrukcija i načinima adresiranja je bilo reči u jednoj od prethodnih sekcija.

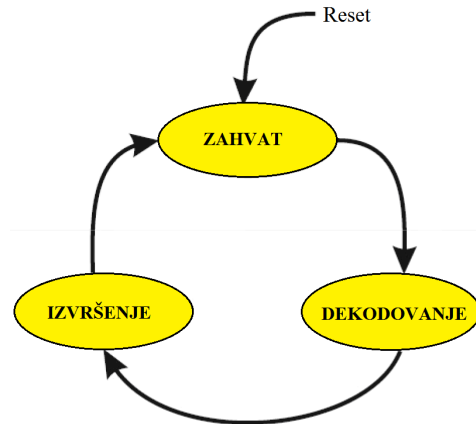
Savremeni mikroprocesori imaju mogućnosti rada sa različitim formatima podataka. Podrazumevane su operacije nad celim brojevima. Kod složenijih mikroprocesora sa više mogućnosti su dodavani novi formati podataka. Operacije nad brojevi u pokretnom zarezu su moguće ukoliko postoji hardverska jedinica za aritmetičke operacije sa brojevima u pokretnom zarezu. Često mikroprocesori imaju mogućnost rada sa pojedinačnim bitovima. Operacije sa negativnim brojevima po pravilu zahtevaju korišćenje drugog komplementa ili označenu aritmetiku i tako dalje.

2.8 Izvršavanje instrukcija mikroprocesora

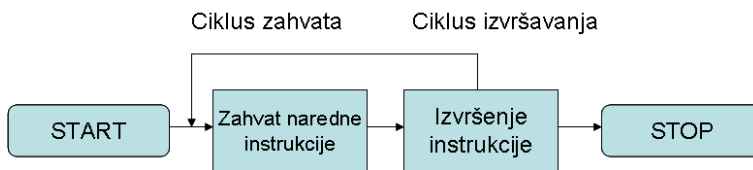
Upravljanje tokom izvršavanja instrukcija vrši upravljačka jedinica i ona obezbeđuje osnovni instrukcioni ciklus (naziva se još i mašinski ciklus) što je prikazano na slici 2.12. Ovaj ciklus započinje uključenjem napajanja koje aktivira proces reseta. Nakon tranzijentnih procesa u toku reseta na kraju se stabilizuje takt i započinje proces zahvata instrukcije koja se potom dekoduje i na kraju izvršava. Ciklus zahvata i izvršenja su prikazani na slici 2.13.

Operacije koje se izvršavaju u svakom od stanja instrukcionog ciklusa su:

- **Faza zahvata:** U ovoj fazi se nova instrukcija prenosi iz programske memorije u CPU. Programski brojač (PC) obezbeđuje adresu instrukcije koju je potrebno prihvatiti iz memorije. Prihvaćena instrukcija smešta se u instrukcioni registar (IR).
- **Faza dekodovanja:** U ovoj fazi vrši se dekodovanje koda prihvaćene instrukcije kako bi upravljačka jedinica mogla znati koju instrukciju treba da



Slika 2.12: Instrukcioni ciklus

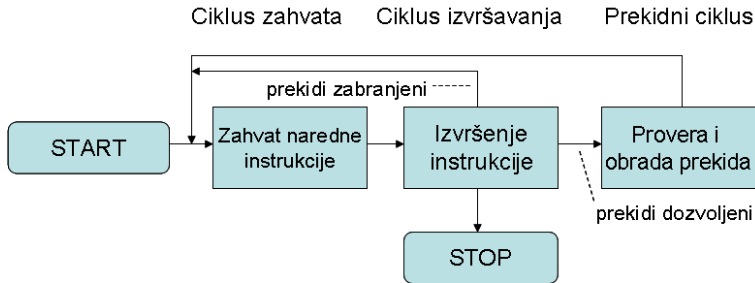


Slika 2.13: Ciklusi zahvata i izvršenja unutar instrukcionog ciklusa

izvrši. Dekodovana informacija se koristi od strane upravljačke jedinice da generiše potrebnu sekvencu upravljačkih signala kako bi se obavile akcije koje su definisane instrukcijom.

- **Faza izvršavanja:** U ovoj fazi upravljačka jedinica aktivira neophodne funkcionalne jedinice mikroprocesora (npr. ALU) u cilju izvršavanja akcija definisanih tekućom instrukcijom. Na kraju ove faze, vrednost PC registra se uvećava kako bi pokazivala na adresu naredne instrukcije koju je potrebno izvršiti. ovaj proces se naziva računanje adrese naredne instrukcije.

Instrukcioni ciklus u slučaju kada imamo i prekide je nešto drugačiji i prikazan je na slici 2.14. Ovde imamo i deo izvršavanja koji sadrži i prekidni ciklus (ukoliko su prekidi dozvoljeni). Tada nakon izvršenja tekuće instrukcije se prelazi na izvršenje i obradu prekida. O prekidima će detaljno biti reči u poglavlju ??.



Slika 2.14: Ciklus izvršavanja u slučaju prekida

Instrukcioni ciklus obično traje nekoliko taktova zavisno od tipa instrukcije i načina adresiranja. Težnja je da se instrukcija izvrši za jedan takt jer će tada ukupne performanse biti najveće ($CPI = 1!$) što će biti objašnjeno u narednoj sekciji.

2.9 Perfomanse mikroprocesora i mikroračunara

Ovde će ukratko biti objašnjene tradicionalne metode za merenja performansi (brzine) mikroprocesora [5], [6]. Polazeći od fundamentalnih aspekata, brzina procesora pre svega zavisi od učestanosti **takta**. Međutim, obzirom na to da se instrukcije različito izvršavaju na različitim arhitekturama mikroprocesora, o čemu je bilo reči u prethodnim sekcijama, podatak o učestanosti takta ne mora nužno da govori i o brzini. Jedina objektivna mera je vreme potrebno da se izvrši neki program, ali ona nije pogodna jer se odnosi na konkretan program. Zbog toga se uvode drugačije mere brzine.

Neke od najčešće korišćenih mera su:

- **CPI** (prosečan broj perioda takta po instrukciji, eng. *Cycles Per Instructions*),
- **MIPS** (milion instrukcija u sekundi, eng. *Millions of Instructions Per Second*), i
- **MFLOPS** (milion instrukcija u pokretnom zarezu po sekundi, eng. *Million Floating point Operations Per Second*).

Vreme potrebno da se izvrši neki program na datom CPU možemo definisati kao:

$$T_{izv} = Broj\ Taktova\ (za\ dati\ program) * Perioda\ takta \quad (2.1)$$

CPI (prosečan broj perioda takta po instrukciji) se definiše kao:

$$CPI = Broj\ Taktova\ (za\ dati\ program) / Broj\ instrukcija \quad (2.2)$$

pa je iz 2.1 i 2.2:

$$\begin{aligned} T_{izv} = Broj\ instrukcija * CPI * Perioda\ takta &= \frac{Broj\ Instrukcija * CPI}{Učestanost\ takta} = \\ &= \frac{Broj\ instrukcija * CPI}{f_{cpu}} \end{aligned} \quad (2.3)$$

Ako je poznata raspodela pojave određenih tipova instrukcija i koliko svaka instrukcija traje perioda takta (CPI_i), tada se može CPI definisati i kao:

$$CPI = \frac{\sum_{i=1}^n CPI_i * I_i}{Broj\ instrukcija} \quad (2.4)$$

Primer 1

Izračunati CPI za mikroprocesor A ako je poznato da je učestanost takta 200 MHz, da program ima 100 instrukcija i benchmark testovi za njega su dati u tabeli.

Tabela 2.1: Benchmark testovi mikroprocesora A

Vrsta instrukcije	Procenat pojave	Broj taktova po instrukciji
ALJ	38	1
Load i Store	15	3
Grananja	42	4
Druge	5	5

Rešenje:

Ako je izvršeno ukupno 100 instrukcija, dobija se:

$$CPI = \frac{\sum_{i=1}^n CPI_i * I_i}{Broj\ instrukcija} = \frac{38 * 1 + 15 * 3 + 42 * 4 + 5 * 5}{100} = 2.76$$

Treba primetiti da učestanost takta nema uticaj na CPI!

MIPS (milion instrukcija po sekundi) se definiše kao:

$$MIPS = \frac{Broj\ instrukcija}{T_{izv} * 10^6} \quad (2.5)$$

a veza između MIPS i CPI je:

$$MIPS = \frac{Broj\ instrukcija}{\frac{Broj\ instrukcija * CPI}{f_{cpu}} * 10^6} = \frac{f_{cpu}}{CPI * 10^6} \quad (2.6)$$

Primer 2

U tabeli su dati podaci za dva različita CPU. Odrediti:

- Koji CPU ima veću vrednost MIPS?
- Koji CPU je brži?

Tabela 2.2: Podaci za dva različita mikroprocesora

Mera	CPU a	CPU b
Broj instrukcija	10 milijardi	8 milijardi
f_{cpu}	4 GHz	4 GHz
CPI	1.0	1.1

Rešenje:

$$a. MIPS_a = 4000 > MIPS_b = 3636$$

$$b. T_{izva} = 2,5 > T_{izvb} = 2,2$$

Primer 3

Mikroprocesor B izvršava isti program kao i mikroprocesor A iz Primera 1 na istoj učestanosti takta od 200 MHz. Benchmark testovi mikroprocesora B su dati u tabeli. Uporediti MIPS i CPI za oba mikroprocesora.

Rešenje:

$$CPI_A = \frac{\sum_{i=1}^n CPI_i * I_i}{Broj\ instrukcija} = \frac{38 * 1 + 15 * 3 + 42 * 4 + 5 * 5}{100} = 2.76$$

$$MIPS_A = \frac{f_{cpu}}{CPI_A * 10^6} = \frac{200 * 10^6}{2.76 * 10^6} = 70.24$$

Tabela 2.3: Benchmark testovi mikroprocesora B

Vrsta instrukcije	Procenat pojave	Broj taktova po instrukciji
ALJ	35	1
Load i Store	30	2
Grananja	15	3
Druge	20	5

$$CPI_B = \frac{\sum_{i=1}^n CPI_i * I_i}{Broj\ instrukcija} = \frac{35 * 1 + 30 * 3 + 15 * 3 + 20 * 5}{100} = 2.4$$

$$MIPS_B = \frac{f_{cpu}}{CPI_B * 10^6} = \frac{200 * 10^6}{2.4 * 10^6} = 83.67$$

Primer 4

Neka su benchmark testovi za dva različita kompjutera dati u tabeli. Uporediti vreme potrebno za izvršenje, MIPS i CPI.

Tabela 2.4: Benchmark testovi mikroprocesora B

Vrsta instrukcije	Broj instrukcija (u milionima)	Broj taktova po instrukciji
CPU A		
ALJ	8	1
Load i Store	4	3
Grananja	2	4
Druge	4	3
CPU B		
ALJ	10	1
Load i Store	8	2
Grananja	2	4
Druge	4	3

Rešenje:

$$CPI_A = \frac{\sum_{i=1}^n CPI_i * I_i}{Broj\ instrukcija} = \frac{(8 * 1 + 4 * 3 + 4 * 4 + 2 * 4) * 10^6}{(8 + 4 + 4 + 2)10^6} \cong 2.2$$

$$MIPS_A = \frac{f_{cpu}}{CPI_A * 10^6} = \frac{200 * 10^6}{2.2 * 10^6} \cong 90.9$$

$$T_{izvA} = \frac{Broj\ instrukcija * CPI_A}{f_{cpu}} = \frac{18 * 10^6 * 2.2}{200 * 10^6} = 0.198s$$

$$CPI_B = \frac{\sum_{i=1}^n CPI_i * I_i}{Broj\ instrukcija} = \frac{(10 * 1 + 8 * 2 + 4 * 4 + 2 * 4) * 10^6}{(10 + 8 + 4 + 2)10^6} \cong 2.1$$

$$MIPS_B = \frac{f_{cpu}}{CPI_A * 10^6} = \frac{200 * 10^6}{2.1 * 10^6} \cong 95.2$$

$$T_{izvB} = \frac{Broj\ instrukcija * CPI_B}{f_{cpu}} = \frac{20 * 10^6 * 2.1}{200 * 10^6} = 0.21s$$

Dakle, iako je

$$MIPS_B > MIPS_A$$

ipak je

$$T_{izvB} > T_{izvA}$$

jer utiče i

$$CPI_B < CPI_A$$

MFLOPS (milion floating point instrukcija po sekundi) se definiše kao:

$$MFLOPS = \frac{Broj\ operacija\ u\ pokretnom\ zarezu}{T_{izv} * 10^6} \quad (2.7)$$

Pored prethodno navedenih postoje i druge koje su se ranije više koristile kao na primer DMIPS/MHz koja bazira na Dhrystone skupu sintetičkih testova. Primer mera novije generacije je CoreMark⁵ koji je baziran na testovima pri radu sa listama, matricama, automatima i CRC. Kao rezultat daje celobrojnu vrednost kao meru perfomansi CPU.

Verovatno najbolji način za procenu perfomansi nekog mikroprocesora predstavljaju *benchmark* paketi koji imaju u sebi višestruke skupove tipičnih test programa koji se u radu sa mikroprocesorima javljaju. Ovakvi alati su nastali analiziranjem realnih aplikacija. Primer ovakvog alata je SPEC⁶ (trenutno aktuelna verzija je SPEC CPU2017).

⁵<https://www.eembc.org/coremark/>

⁶<https://www.spec.org/benchmarks.html>

2.10 Neki ograničavajući faktori performansi

U ovoj sekciji će biti objašnjeni Murov zakon (eng. *Moore's law*) [9], Memorij-ski zid (eng. *Memory wall*) [10] i Zid snage (eng. *Power wall*) [11]. Svaki od njih pokazuje na neka ograničenja koja imaju uticaja na performanse.

Murov zakon

Murov zakon je termin koji je nastao u vreme kada su nastali prvi mikro-procesori. Ono što je Mur uočio i publikovao nekoliko godina ranije je bilo u eri pojave prvih integrisanih kola i on je ustanovio *da se broj tranzistora na integrisanim kolima povećava sa faktorom dva na svake dve godine*. Ovo se kasnije pokazalo kao potpuno tačno i u slučaju mikroprocesora. Često je Murov zakon bio i faktor predviđanja razvoja mikroprocesora. Murov zakon sa nekim manjim modifikacijama se koristio i u drugim domenima (npr. u vezi razvoja DRAM memorija). Bilo je i pokušaja da se performanse mikroprocesora izraze upotrebom modifikovanog Murovog zakona u formi "performanse CPU se dupliraju svakih 18 meseci", ali ovo nema veze sa osnovnim Murovim zakonom.

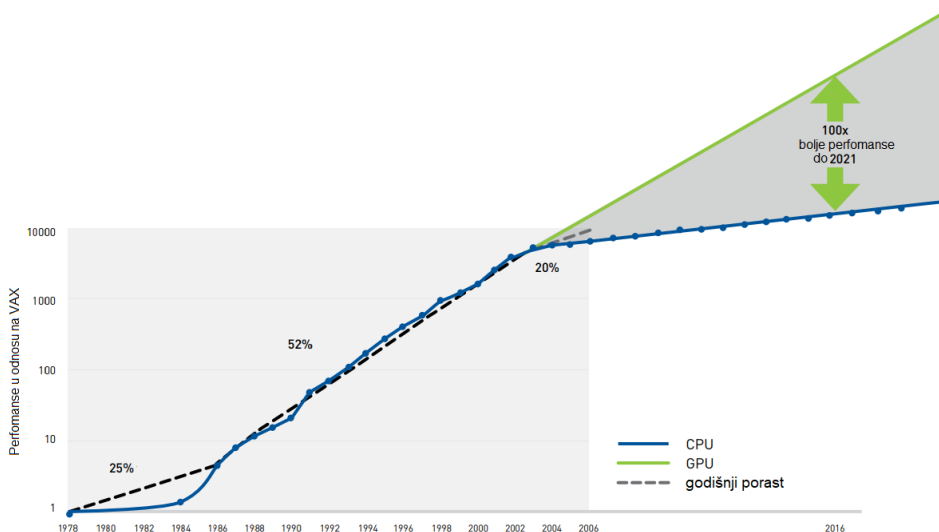
Razvoj integrisanih kola baziranih na silicijumu su tokom decenija pratili Murov zakon, a samim tim je i razvoj mikroprocesora to pratio. Kasnijim analizama se pokazuje da se i performanse mikroprocesora približno dupliraju na svake dve godine (prema nekim autorima i brže na svakih 1.5 godinu). To važi sve do početka 2000-tih kada se ovaj ovaj trend se usporava za klasične CPU i opada na 20-tak procenata godišnjeg poboljšanja. To se dešava paralelno sa ulaskom u eru višejezgaranih mikroprocesora i grafičkih procesora (eng. *Graphical Processing Unit* - GPU) koji nastavljaju prethodni trend dupliranja na svake dve godine. Na slici 2.15⁷ je prikazan razvoj performansi CPU u odnosu na VAX⁸ tokom nekoliko decenija.

Memorijski zid

Polazeći od Fon Nojmanove ili Harvard arhitekture mikroračunara prikazanih na slici 2.4, jasno je da bi mikroprocesor mogao da izvršava bilo koji program i obrađuje bilo koje podatke potrebno je da ih preuzme (tj. zahvati) iz memorije. Dakle pristup memoriji (memorijama) je jedna od osnovnih operacija koje CPU treba da uradi. Tokom decenija razvoja tehnologije proizvodnje memorija pokazalo se da je povećanje performansi memorija značajno sporije od razvoja performansi

⁷Tesla GPU brošura, <https://www.nvidia.com/docs/IO/100133/tesla-brochure-12-lr.pdf>

⁸VAX predstavlja liniju radnih stanica i superminiračunara iz 70-tih godina prošlog veka proizvođača DEC



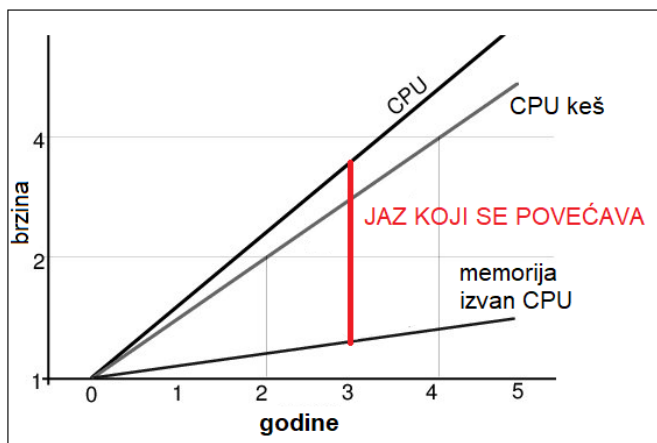
Slika 2.15: Povećanje performansi CPU u odnosu na VAX

CPU jer se brzina memorija povećavala svega 10-tak procenata godišnje. Time se stvara tzv. *memorijski zid* koji predstavlja jaz u performansama CPU i memorije u datom trenutku razvoja tehnologije. Ovaj problem je postao posebno izražen u doba paralelnog računarstva gde postoji više jezgara ili CPU, a arhitektura je npr. sa deljenom memorijom. Pokazuje se da memorije postaju usko grlo ovakvih sistema.

Postoje razna manje ili više dobra rešenja koja smanjuju efekat problema memorijskog zida. Neka od njih su: uvođenje tzv. keš (eng. *Cache*) memorija⁹, posebnih keš memorija za instrukcije i podatke, proizvodnja keš memorija na CPU čipu, više nivoa keš memorija, predviđanje izvršavanja instrukcija, tehnike smanjenja broja pristupa memoriji itd. Međutim, problem memorijskog zida ostaje i do danas prisutan u smislu ograničenja performansi sistema. Ovaj efekat je ilustrovan na slici 2.16. Ostaje da se vidi kakav će uticaj imati moderna rešenja novih memorija na ovaj problem (npr. pojava optičkih memorija [12]).

Zid snage

⁹Na našem govornom području se koristi i termin 'skrivena memorija'



Slika 2.16: Ilustracija memorijskog zida

Iako je detaljna analiza problema pod nazivom 'Zid snage' daleko izvan okvira ovog udžbenika, ovde će biti prikazane osnovne osobine radi boljeg razumevanja faktora koji ograničavaju performanse mikroracunara. Dominantna tehnologija za izradu mikroprocesora je tokom decenija bila silicijumska CMOS tehnologija. Osnovni elemenat u okviru ove tehnologije je CMOS inverter koji se sastoji od 2 MOSFET tranzistora različitih polarizacija (p i n tipa). Vrlo uprošćeno govoreći proizvodnja mikroprocesora bi se sastojala od mnoštva različitih kombinacija ovih tranzistora. Kada se postavi pitanje potrošnje snage nekog mikroprocesora onda se podrazumeva obično dinamička potrošnja snage koja se dešava na svakoj ivici takta kada se dešavaju određene promene signala sa logičkih nula na jedinicu i obrnuto. Drugim rečima, tada se neki tranzistori isključuju dok se drugi tada uključuju (razmatramo slučaj tranzistora kao prekidača u digitalnim kolima).

Može se pokazati da, ako se zanemare struje curenja i drugi uticaji, se tada potrošnja snage P_d može iskazati sledećom približnom jednačinom:

$$P_d \sim V^2 * C * f * a \quad (2.8)$$

gde su:

V - napon napajanja,

C - izlazna kapacitivnost,

f - učestanost takta na kojem radi CPU, i

$0 < a < 1$ je faktor aktivnosti.

Analizom pojedinih činilaca iz prethodne jednačine se može ustanoviti sledeće. Napon napajanja V je uslovljen tehnološki i predstavlja konstantnu vrednost, kao i kapacitet C . Faktor aktivnosti a kada uprosečimo na 0.5, na opadanje P_d utiče smanjenje učestanosti f što ne odgovara jer se time smanjuju i performanse CPU i celog mikroračunara. Sa druge strane f ne može ni da raste proizvoljno mnogo jer sa prevelikim povećanjem dolazi do izražaja talasna priroda signala i javili bi se brojni dodatno više izraženi problemi (preslušavanja, zvonjenja, refleksije itd.) pored osnovnog problema kako ohladiti CPU. Problem hlađenja se u novije vreme pored specijalnih ventilatora, rešava uvođenjem hladnjaka sa fluidima čime se pospešuje odvođenje disipirane toplote. Zaključak predhodne analize je da postoji unapred poznato ograničenje koliko maksimalno snage može da potroši/disipira određeni CPU i to predstavlja 'zid snage'.

U novije vreme se uvode tehnike štednje energije tako što se uvode različiti nivoi neaktivnosti mikroprocesora (tzv. *idle* i *sleep* modovi). Sa istim ciljem se rade i različite dinamičke tehnike skaliranja napona i frekvencije (eng. *dynamic voltage and frequency scaling (DVFS)*). Kod multijezgarnih CPU se uvode "štedljivija" jezgra za određene operacije i sl.

2.11 Pitanja

1. Od kojih komponenata se sastoji minimalna konfiguracija mikroračunarskog sistema? Objasniti.
2. Napisati i objasniti kako se dele arhitekture računara sa stanovišta organizacije memorije.
3. Objasniti podelu CPU sa stanovišta načina i vremena potrebnog za izvršavanje instrukcija.
4. Objasniti podelu CPU sistema sa stanovišta realizacije magistrala.
5. Koje su razlike između RISC I CISC skupova instrukcija?
6. Koji je hardverski minimalni skup komponenata, a koji softverski koje definišu arhitekturu CPU?
7. Šta spada u podsistem za obradu i tokove podataka (datapath) procesora?
8. Objasniti pojmove: instrukcija, skup instrukcija i načini adresiranja.

9. Objasniti sve elemente programerskog modela mikroprocesora.
10. Objasniti faze izvršavanja instrukcija.
11. Navesti vrste i objasniti namenu registara CPU.
12. Čemu služi _____ registar?
13. Na koje sve načine se može realizovati upravljačka jedinica (UJ)? Objasniti svaki.
14. Nacrtati blok šemu i objasniti mikroprogramsku realizaciju UJ?
15. Zbog čega se realizacija preko konačnih automata danas često koristi pogotovo za UJ-e realizovane u programabilnoj logici?
16. Da li veličina učestanosti takta automatski govori o brzini rada računara? Zašto?
17. Napisati izraz i objasniti kao se računa CPI.
18. Napisati izraz i objasniti kao se računa MIPS.
19. Napisati izraz za vezu CPI i MIPS i objasniti.
20. Napisati izraz i objasniti kao se računa MFLOPS.
21. Benchmark testovi su dati za 2 različita kompjutera. Izračunati i uporediti CPI i MIPS.
22. Objasniti Murov zakon.
23. Objasniti pojam 'Memorijski zid'.
24. Objasniti pojam 'Zid snage'.

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] D. A. Patterson, J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, Fourth Edition, Morgan Kaufmann, 2011.
- [6] M. Abd-El-Barr, H. El-Rewini, *Fundamentals Of Computer Organization And Architecture*, John Wiley and Sons, Inc., 2005.
- [7] R. S. Sandige, M. L. Sandige, *Fundamentals of Digital and Computer Design with VHDL*, McGraw Hill, 2012.
- [8] M. M. Mano, C. R. Kime, T. Martin, *Logic and Computer Design Fundamentals*, Fifth edition, Pearson, 2015.
- [9] J.L. Gustafson, Moore's Law. In: D. Padua D. (eds) *Encyclopedia of Parallel Computing*. Springer, 2011.
- [10] S.A. McKee, R.W. Wisniewski, Memory Wall. In: D. Padua (eds) *Encyclopedia of Parallel Computing*. Springer, 2011.

- [11] P. Bose, Power Wall. In: D. Padua (eds) Encyclopedia of Parallel Computing. Springer, 2011.
- [12] 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.
- [13] W.W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques", IEEE Wescon Proc., Aug. 1970.
- [14] C.G. Bell, A. Newell, Computer structures: Readings and Examples, McGraw-Hill, 1971.
- [15] E.O. Hwang, Digital Logic and Microprocessor Design With VHDL with Interfacing, Cengage Learning, Second Edition, 2017.
- [16] S. Brown, Z. Vranesic, Fundamentals of Digital Logic with VHDL Design, McGraw Hill, 2009.
- [17] D. L. Perry, VHDL: Programming by Example, Fourth edition, McGraw Hill, 2002.
- [18] L. Null, J. Lobur, Essentials of Computer Organization and Architecture, Jones and Bartlet Learning, 3rd Ed., 2010.
- [19] I. Mezei "Evolution of an educational microprocessor", Computer Applications in Engineering Education, 28(5), Wiley, pp. 1265-1277, 2020.
- [20] 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.
- [21] 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.
- [22] S. L. Harris, D. Harris, Digital Design and Computer Architecture, RISC-V Edition, Morgan Kaufman, 1st edition, 2021.
- [23] 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>
- [24] S. Harris, D. Hariss, Digital Design and Computer Architecture, Arm edition, Morgan Kaufmann, 2015.

- [25] 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.
- [26] D. Kushner, "The making of arduino", IEEE spectrum 26, 2011.
- [27] M. Banzi, "How Arduino is open-sourcing imagination", TEDtalk, Scotland, 2012. <https://www.youtube.com/watch?v=UoBUXOOdLXY> (datum pristupa: 31.01.2022.)
- [28] Sparkfun, "Arduino shields v2", <https://learn.sparkfun.com/tutorials/arduino-shields-v2>, (datum pristupa: 01.02.2022.)
- [29] Arduino Uno Rev3 schematic, https://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf (datum pristupa: 02.02.2022.)
- [30] Arduino software, <https://www.arduino.cc/en/Main/Software> (datum pristupa: 02.02.2022.)

