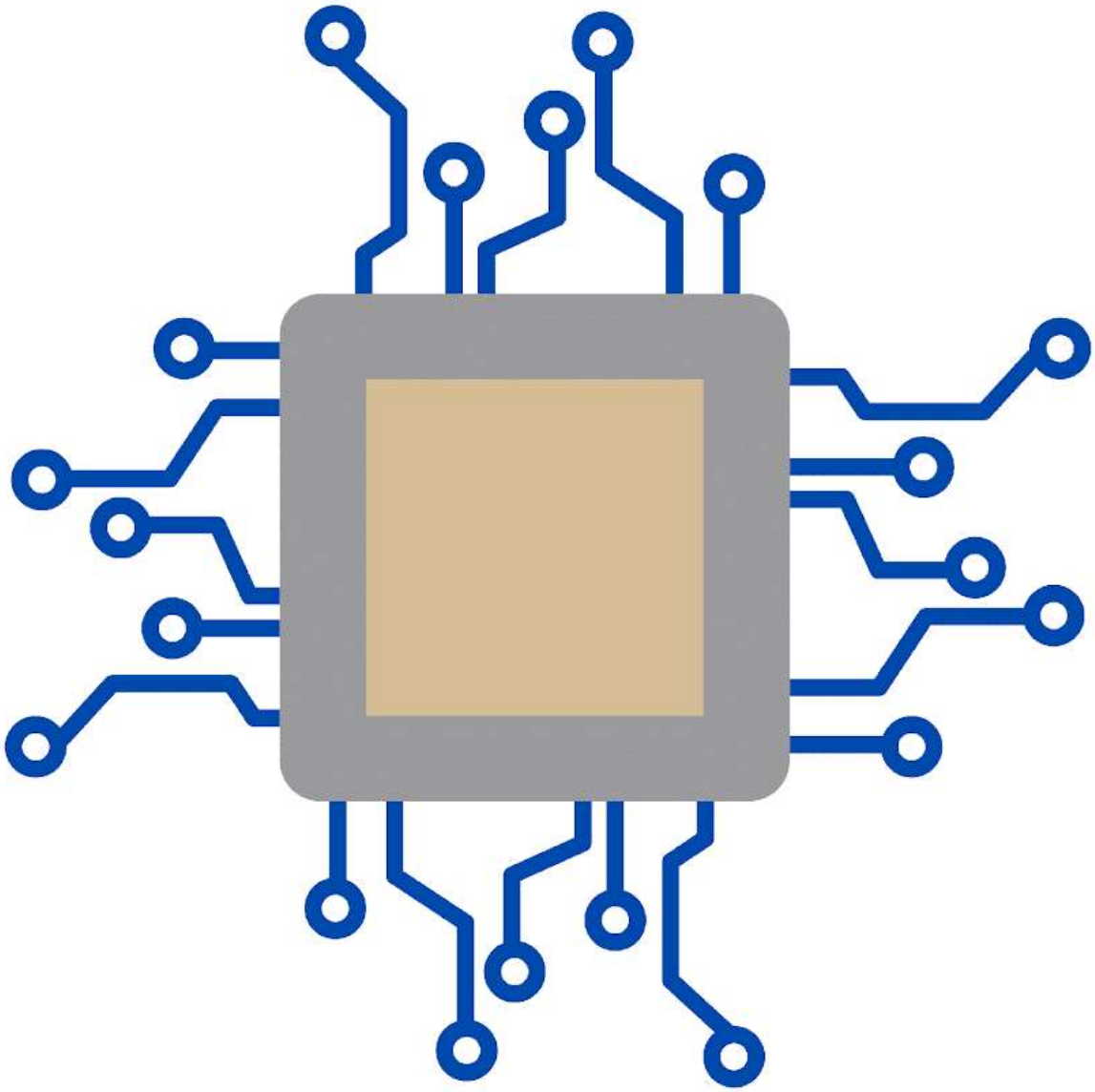
ARHIKTETURA RAČUNARA, MEMORIJA, DMA, CPU
LITTLE ENDIAN, DBUS



NAMENSKI RAČUNARSKI SISTEMI

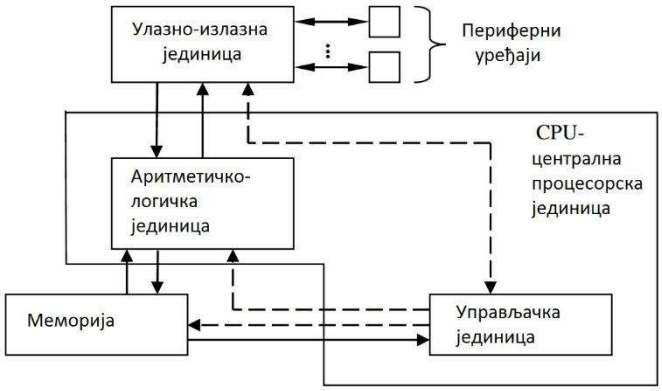
Jun 2022.

I ARHIKTETURA RAČUNARA

1. Opisati arhikteturu računara. Objasniti organizaciju komponenti i nacrtati skicu osnovnih komponenti

- **Arhitektura računara** je unutrašnja struktura digitalnog računara koju sačinjava dizajn i raspored skupa instrukacija i registara za skladištenje podataka. Određena računarska arhitektura se bira imajući u vidu tip softvera koji će se na njemu izvršavati.

  Glavne komponente ili podsistemi jedne računarske arhitekture su:
  - **ulazno/izlazni** uređaji,
  - **skladištenje**
  - **komunikacija**,
- **kontrola** i
- **obrada.**



Slika 1: Arhiktetura računara

Za svaki se može reći da poseduje vlastitu, posebnu arhitekturu.

2. **Instrukcije kao osnova projektovanja računara**

- Obrada podataka se vrši od strane procesora. Procesor služi za upravljanje i obradu koji pod programskim upravljanjem vrši obradu podataka unutar računara.

  Računar je konstruisan tako da može direktno u procesoru izvršavati određeni broj prostih operacija koje se nazivaju ***mašinske operacije***.

  Ove operacije mogu biti:
  - ***aritmetičke*** (sabiranje, oduzimanje, množenje i deljenje),
  - ***logičke*** (upoređivanje dve , ...),
  - ***prenos podataka*** iz jednog dela računara u drugi,
  - ***učitavanje podataka*** iz okruženja, ili
  - ***slanje podataka*** na periferiju.

  Izraz, koji određuje mašinsku operaciju zajedno sa operandima, tj. sa vrednostima ili mestima na kojima se te vrednosti nalaze, naziva se ***mašinska instrukcija*** ili ***naredba***.

ADD mašinska operacija sabiranja, onda ta operacija sa konkretnim operandima tj. vrednostima (npr. 2 ADD 3) je već mašinska instrukcija.

Niz ovih mašinskih instrukcija koje saopštavaju računaru, korak po korak, kako se rešava neki problem, naziva se ***mašinski program***

Mašinska operacija je beskorisna, nema svrhu, jer joj nismo pridružili konkretne operande - kada operaciji pridružimo operande, operacija dobija smisao, i tada se naziva mašinska instrukcija.

3. Fon Nojman arhiktetura kao inicijalna ideja računara

- Originalna **Fon Nojmanova mašina** je imala pet osnovnih delova:
  - **memoriju,**
  - **aritmetičko-logičku jedinicu,**
  - **upravljačku jedinicu,**
  - **ulaznu** i
  - izlaznu opremu.

Unutar aritmetičko-logičke jedinice je bio jedan specijalni unutrašnji registar nazvan **akumulator**. Tipične aritmetičke ili logičke operacije unutar aritmetičko-logičke jedinice su se izvršavale korišćenjem ovog akumulatora.

Pri izvršavanju te operacije (tačnije, instrukcije), u akumulatoru se nalazio jedan od operanada (podatak) a drugi recimo u operativnoj memoriji. Posle izvršavanja operacije, aritmetičko-logička jedinica je rezultat smestila u akumulator, brišući njegov prethodni sadržaj. Ovaj podatak u akumulatoru je bio ponovo korišćen za sledeću mašinsku instrukciju, ili je bio premešten u operativnu memoriju. Naravno, redosled izvršavanja mašinskih instrukcija unutar mašinskog programa je kontrolisala upravljačka jedinica.

Osnovni koncepti Fon Neumann-ove arhitekture su sledeći:

- **Mašinske instrukcije** (tj. program) i podaci se čuvaju u istoj memoriji
- Svi podaci su predstavljeni u **binarnom obliku**
- Mašinske instrukcije slede jedna za drugom u memoriji računara
- Računar **razmenjuje podatke** između memorije i aritmetičko-logičke jedinice preko akumulatora
- Mašinske instrukcije unutar mašinskog programa se izvršavaju jedna za drugom dok se redosled eksplicitno ne promeni **naredbom za skok.**

4. **CPU struktura**
    - Jedinica koja je sposobna realizovati aritmetičke i logičke operacije naziva se aritmetičko-logička jedinica (**ALU**) sastoji se od **registara** i elektronskih kola koja izvode **aritmetičke operacije** (sabiranje, oduzimanje, množenje, deljenje) i **logičke operacije** (upređivanje dve vrednosti po veličini i određivanje da li je izraz istinit ili nije).

      U početku su se ove operacije izvodile samo sa **celim** brojevima, dok su se operacije sa realnim brojevima izvodile softverski. Kasnije je aritmetičko-logičkoj jedidinici pridodata posebna jedinica za izvođenje operacija sa **realnim brojevima** i izračunavanje trigonometrijskih i drugih funkcija (floating point processor), koja je u početku bila realizovana kao posebna jedinica (**coprocessor**), dok su kod savremenih računara obe jedinice realizovane u okviru istog čipa.

5. **CPU (Šta je kod instrukcije i koji tipovi adresiranja postoje, čemu služe)?**
    - Osnovna jedinica svakog računara je procesor ili centralna procesorka jedinica. Procesor je integrisano kolo i u njemu se realizuju sve računske i logičke operacije i izvršavaju instrukcije koje su zadate programom tokom rada računara.

      Funkcija centralne procesorske jedinice (CPU)
      - Izvršavanje programskih instrukcija
      - Osiguravanje programskim instrukcijama da budu izvršene u pravilnom redosledu
      - Izvršavanje operacija sa pokretnim zarezom

      Karakteristike procesora su određene arhitekturom i to su:
      - **brzina procesora** koja se izražava u milionima operacija koje on obavlja u jednoj sekundi MIPS-ovima (Milion Instruction Per Second) ili MFLOPS-ima (Milion Floating Point Operations Per Second)
      - **dužina procesorske reči** je broj bitova koji se istovremeno prenose i obrađuju unutar procesora. Danas se upotrebljavaju 32-bitni i 64-bitni procesori
      - **radni takt** je učestalost impulsa koji generiše sat (clock) - specijalno elektronsko kolo kojim se iniciraju operacije procesora. Procesor preko jedne linije na kontrolnoj magistrali dobijaju takt signal (pravougaone impulse odrđene učestanosti).
      - **Učestanost** tog takt signala je u stvari učestanost sistemskog takta matične ploče. Samo jezgro savremenih mikroprocesora

radi na znatno većoj učestanosti internog takta. Ta učestanost je određena takozvanim množiocem, to jest brojem kojim treba pomnožiti učestanost sistemske magistrale da bi se dobila interna učestanost na kojoj radi jezgro mikroprocesoora. Radni takt se meri u GHz. Veći radni takt omogućava veću brzinu procesora pa se sve češće GHz upotrebljava kao merna jedinica za brzinu procesora.

- **Broj jezgara** na samoj pločici (core) **–** uglavnom novije generacije imaju više jezgarnu organizaciju za paralelno procesiranje instrukcija i signala.
- **Nominalna snaga –** definiše disipaciju (dyulove gubitke) mrežno opterećenje napojne jedinice preko matične ploče.
- **Keš memorija –** L1, L2 L3
- **Procesorska reč –** 16 bitni, 32 bita ili 64 bita za poslednje trenutne generacije
- **Radna magistala –** 8, 16, 32 i 64, 128, 256, 512 bitna
- **Radna temperatura –** Dozvoljena temperatura do koje će procesor normalno raditi i izvršavati instrukcije.

6. Kako izgleda proces izvršavanja instrukcije (fetch, ...) ?

Slika 2: Dijagram izvršavanja instrukcije

- Instrukcioni ciklus se deli na dve faze:
  - **faza pribavljanja** instrukcije
  - **faza izvršenja** instrukcije

Faza pribavljanja instrukcija je univerzalna i odnosi se na sve tipove instrukcija koje se prikupljaju i dekodiraju.

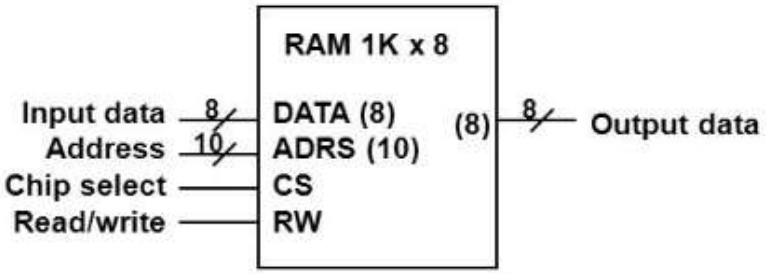
Faza izvršenja instrukcija dosta je složenija i njeni međusobni interni koraci mogu se dosta razlikovati, što je kao fazu čini dosta složenijom.

Vreme potrebno za pribavljanje, dekodiranje i **izvršenje instrukcije** kao i vreme trajanja svih operacija procesora naziva se **vremenski ciklus procesora.**

Recipročna vrednost vremenskog ciklusa procesora naziva se **takt procesora** i sam takt zavisi od tehnologije u kojoj je sam procesor izrađen.

7. Struktura, opis i način rada memorijskog modula

- Memorija je zadužena za čuvanje informacija, nezavisno kog su tipa informacije (instrukcije ili podaci). Sama memorija je sekvencijalno kolo sastavljeno od niza memorijskih čipova (registara) koji predstavljaju jednu memorijsku reč - **word**.



Slika 3: Shema generičkog memorijskog modula

- Interna organizacija memorijskog modula je matrica dimenzija N x L, koja čuva bit po bit informacije. L je oznaka za dužinu reči u bitima, dok N je broj reči odnosno memorijskih registara.

  Sprega sa memorijom obuhvata kontrolu pristupa (**R**, **W**, **CS**), adresne (**ADRS**) i linija podataka (**DATA**). **Broj adresnih linija** određen je kapacitetom memorije, dok je broj podataka jednak dužini memorijske reči.

  Dužina reči može biti različita krećući se u opsegu od **8** do **64** bita. Iako se instrukcije učitavaju kao jedna reč, procesor može uzeti **celu** reč, **polovinu reči** ili samo **jedan bajt** od cele reči.

Dve osnovne instrukcije sa rad sa memorijom su:

- **Load** (uzimanje odnosno učitavanje informacije iz memorije)
- **Store** (smeštanje informacije u memoriju)

Obe operacije međusobno razmenjuju sadržaj između **memorijske lokacije** i **registara u procesoru**.

Memorijski Adresni Registar (MAR) u toku pristupa čuva adresu memorijske lokacije kojoj je pristupio.

Memorijski Bafer Registar (MBR) je polazište/odredište podataka u zavisnosti od toga da li se nad memorijom vrši upis ili se iščitava sadržaj.

Kontrolu rada i podešavanja kontrolnih signala postavlja **upravljačka jedinica**.

Kako bi se postiglo ubrzanje došlo se na ideju da se između **procesora** i **memorije** uvede **sprega preko spoljne magistrale**. Takvim korakom potreba za **MAR** je potpuno istisnuta, dok bi postavljanjem posebnih signala na spoljnu magistralu počelo **izvršenje instrukcije**.

Sve je češća praksa da se u modernim mikroprocesorima ugrađuje **Memory Managment Unit (MMU)** koji na sebe preuzima teret **specijalizovanog upravljanja memorijom** i **preslikavanja logičkih adresa** u **fizičke adrese**.

8. Ulazno/Izlazni podsistem

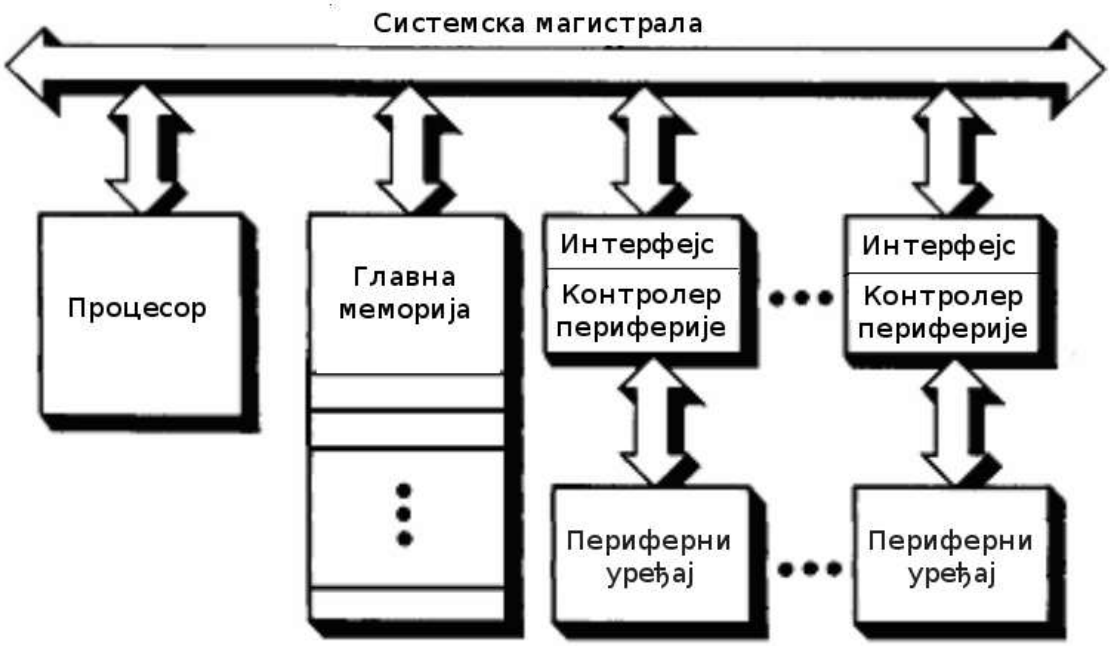
- **Ulazno-izlazni podsistem** ostvaruje prenos podataka između računara i njegovog spoljašnjeg okruženja.

  Okruženje se sastoji od četiri grupe uređaja:
  - **perifernih uređaja za komunikaciju sa korisnikom**,
  - **spoljne memorije**,
  - **uređaja za prenos podataka** preko komunikacionih linija i
  - **uređaje za vezu sa objektom** nad kojim se upravlja.

  Perifernim uređajima nazivaju se svi uređaji koji se preko U/I podsistema mogu priključiti na sistem radi ulaza ili izlaza podataka.

  U/I kontroler upravlja radom perifernih uređaja, shodno naredbama

koje izdaje CPU. Jedan U/I kontroler može biti namenjen za kontrolu više U/I uređaja.



Slika 4: Struktura U/I podsistema

Koliko je određena periferna jedinica složena diktira opis samog kontrolera, koji naizgled jako slični mogu imati potpunu različitu fizičku stranu, dok im je **digitalna strana UVEK okrenuta ka CPU**.

9. Tehnike rukovanja U/I zahtevima

- Kako je sam U/I podsistem poprilično složen, tako i same načine rukovanja nad njim možemo podeliti na tri grupe.

Programirani (polling) način rukovanja zahtevima podrazumeva da se **periodično očitavaju stanja** svake od U/I jedinica, na osnovu čega se odlučuje o daljim akcijama opsluživanja zahteva.

Rukovanje pomoću prekida (interrupt) predstavlja uvođenja sistema prekida i tabele prekida, sa idejom da sam podsistem obavesti procesor o pojavi događaja koji zahteva pažnju samog procesora. Po prijemu zahteva, procesor prekida izvršavanje tekuće instrukcije, čuva informaciju o mestu prekida, i prenosi izvršenje na **prekidnu rutinu** koja obrađuje dati događaj, te nakon obrade

vrednost **programskog brojača** se **restaurira** i vraća se nastavku izvršavanja prethodne naredbe.

Direktan pristup memoriji (DMA) obezbeđuje značajno manje kašnjenje usluge, ali ne skraćuje ukupno vreme prenosa. Razlog tome je da je procesor **dosta brži** u odnosu na periferije, te je krajnja brzina ogračena ne najbržim uređajem, već onim najsporijim - periferije.

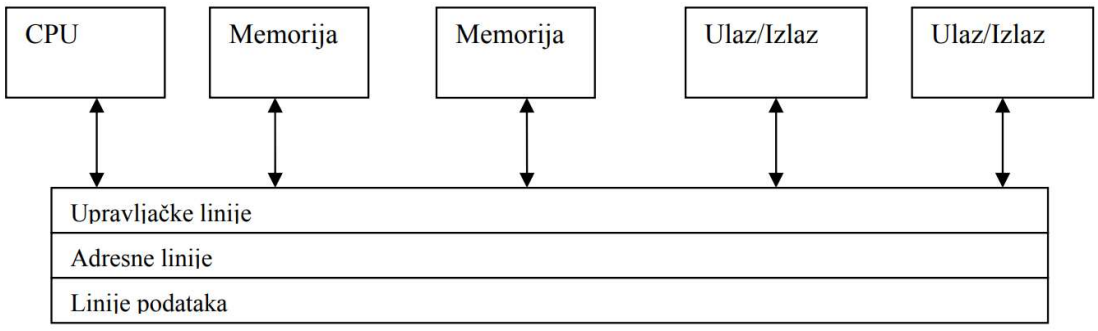
10. Sistemska magistrala

- **Sistemska magistrala (BUS)** predstavlja prenosni put za prenos adresnih signala, signala podataka i kontrolnih signala između procesora, memorije i U/I uređaja.

Sistemska magistrala međusobno povezuje funkcionalne jedinice te takav spoj formira računarski sistem.

Magistrala je direktno proširenje procesora jer prenosi signale koje kontroliše procesor bez posrednika.

Uskom specijalizacijom sistemske magistrale, ona se može proširiti na **PCI** i **USB** magistrale koje su značajno složenije sprežne strukture prisutne u modernim računarima.



Slika 5: Uloga i organizacija sistemske magistrale

Osnovna aktivnost na magistrali je **prenos podataka**:

Transakcija ima dva učesnika:

- **inicijatora prenosa**
- **onog koji odgovara na zahtev**

Inicijator je uvek **centralni procesor** koji postavlja adresu i smer prenosa podataka, uz angažovanje **dodatnih signala neophodnih za kordinaciju** predajne i prijemne strane i nesmetan tok podataka.

Izvor i odredište podataka uvek su registri u procesoru, memoriji, U/I.

11. Logičko projektovanje procesora (Veza arhikteture i organizacije)

- Logičko projektovanje procesora na najvišem nivou jeste postupak koji na početku ima definiciju **arhikteture** (formata podataka, skupa registara i skupa instrukcija), a na svom kraju daje logičku šemu (organizaciju) datog procesora.

  Projektovanje savremenih mikroprocesora je složen proces koji je uslovljen kompeksnošću njihove kompleksnosti.

  Kompleksnog se može podeliti na:
  - **obimnost** skupa instrukcija i adresnih režima i
  - **primena tehnika ubrzanja** obrade koje unose ogračinjena.

II CENTRALNI PROCESOR

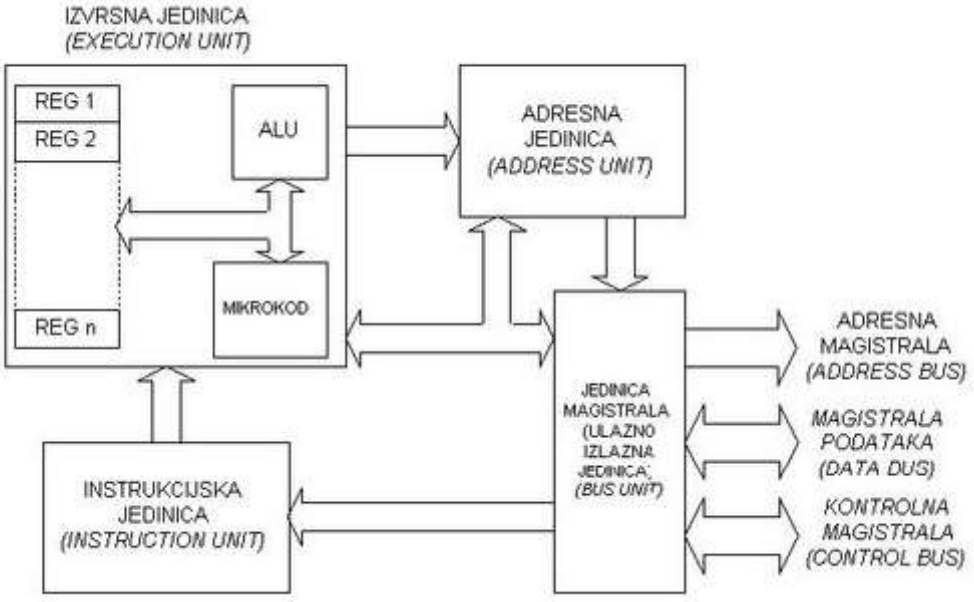
12. **Opis i struktura mikroprocesora**

- Procesor se sastoji iz nekoliko delova, od kojih su osnovni:
  - **upravljačka jedinica** (CU – Control Unit),
  - **aritmetičko-logička jedinica** (ALU – Arithmetic Logic Unit) i
  - **registri**.

Upravljačka jedinica kontroliše rad računara. Ona vrši **dekodiranje** operacija u programu, **upravlja izvršenjem programa**, **memorijom** i ostalim delovima računarskog sistema. Sa razvojem računara ide se ka tome da procesor bude što više rasterećen ovih funkcija, pa tako posebni kontroleri povezani na matičnu ploču preuzimaju deo ovih poslova.

Aritmetičko-logička jedinica jeste deo procesora koji obavlja aritmetičke i logičke operacije nad celim brojevima i pojedinim bitovima. Kao dodatak, procesori danas poseduju i jedinicu za računanje sa brojevima u pokretnom zarezu.

Registri su posebne memorijske lokacije dostupne unutar samog procesora. U njima su smešteni podaci sa kojima procesor upravo radi.



Slika 6: Struktura mikroprocesora

13. Programski jezik centralnog procesora - Mašinski jezik

- Računari su napravljeni tako da koriste samo **binarnu azbuku** (**0** i **1**), a jezik koji se gradi nad ovom azbukom i na kojem računar izvršava naredbe naziva se **mašinski jezik**. Pošto mašinski jezik nije pogodan za čoveka konstruisani su veštački jezici koji se nazivaju programski jezici.

  Mašinski jezik je interni jezik svakog računara. Programi napisani na mašinskom jeziku izvršavaju se brže od programa napisanog na nekom drugom programskom jeziku, što je njihova osnovana prednost. Naredba napisana u mašinskom jeziku može imati sledeći oblik: ***1110 0010 1110 0111***.

  Prvih osam bita (niži bajt) predstavlja adresu, a drugih osam bita (viši bajt) samu naredbu. Ovakvi programi su nerazumljivi i dugi, tako da je pisanje programa na mašinskom jeziku podložno greškama.

  Pored toga mašinski jezici različitih računara (tj. procesora) su različiti, što znači da se programi pisani za jedan procesor ne mogu izvršavati na drugom procesoru.

14. Format mašinske instrukcije

- Svaka instrukcija predstavlja **niz bitova** (0 i 1) određene dužine. Sam format instrukcije definiše značenje bitova u instrukciji.

  Formatom instrukcije definišu se:
  - operacija koju treba izvršiti
  - tip podatka nad kojim se izvršava operacija i
  - izvorišni i odredišni operandi.

  Dužina formata instrukcije zavisi od:
  - veličine memorije
  - memorijske organizacije
  - magistrale i
  - složenosti i brzine procesora.

  Operacija predstavlja neku od operacija iz skupa instrukcija koje procesor može da izvrši. **Tip operanda** određuje iz koliko **susednih memorijskih lokacija** treba pročitati operand i kako ga interpretirati. **informacija o operaciji** i **tipu podatka** specificira se poljem u formatu instrukcije koje se naziva **kod operacije**.

Bitno je napomenuti da ako se ista operacija primenjuje nad različitim tipovima podataka, mora se koristiti veći broj kodova operacija!

Interpretacija formata:

- **2 adresna polja** predstavljaju adrese izvorišnih operanada
- **1 adresno polje** predstavlja adresu odredišnog operanda

Slika 6: Format instrukcije

Tipovi adresiranja:

- **registarsko adresiranje**
- **neposredno adresiranje operanada (memorije)**

Kod **registarskog adresiranja**, operandi su u registrima koji se adresiraju pomoću bita unutar adresnog sloga instrukcije i koji se mogu smatrati proširenjem koda operacije.

U slučaju **neposrednog adresiranja**, vrednost operanada se zapisuje u memorijsku lokaciju neposredno iza prve instrukcione reči sa kodom operacije. Izuzetak su instrukcije koje specificiraju polureč, odnosno konstantu koja može stati u slobodni deo instrukcione reči.

15. Klasifikacija računara prema skupu instrukcija (prednosti i mane)

- Arhitektura procesora je određena sa:
  - **skupom registara,**
  - **skupom tipova podataka koje procesor podržava,**
  - **formatom instrukcija,**
  - **skupom instrukcija,**
  - **načinima adresiranja,**
  - **mehanizmom prekida**.

CISC mašine, odnosno, **mašine sa složenim skupom instrukcija** imaju veći broj formata instrukcija, instrukcije različitih dužina, veći izbor načina adresiranja i veću složenost hardvera.

RISC arhitektura:

- postoji više prostora za registre procesora,
- više prostora za keš memoriju,
- korišćenje pipelining-a je jednostavnije.

Glavni nedostaci RISC arhitekture su da. programi imaju veću veličinu i potrebno je više memorije za izvršavanje programa.

RISC vs. CISC

CISC	RISC
Emphasis on hardware	Emphasis on software
Multiple instruction sizes and formats	Instructions of same set with few formats
Less registers	Uses more registers
More addressing modes	Fewer addressing modes
Extensive use of microprogramming	Complexity in compiler
Instructions take a varying amount of cycle time	Instructions take one cycle time
Pipelining is difficult	Pipelining is easy

Slika 7: Prednosti i mane CISC/RISC arhikteture

16. Tipovi instrukcija

- Klasifikacija instrukcija obično se vrši u skladu sa kriterijumima:
  - **lokacije operanada** i
  - **funkciju instrukcije**.

Instrukcije kod kojih su svi operandi smešteni u registrima nazivaju se **registarskim instrukcijama**, zbog čega je njihovo izvršenje vrlo brzo.

Ukoliko se jedan od operanada instrukcije nalazi u **memoriji**, njegovo pribavljanje zahteva pristup memoriji, te se ovakve instrukcije nazivaju **memorijskim instrukcijama**.

Instrukcije se mogu podeliti na:

- **ARITMETIČKE OPERACIJE** množenje sabiranje, oduzimanje, uvećanje, smanjenje, negacija, zaokruženje, apsolutna vrednost i množenje.

- **LOGIČKE OPERACIJE** I, ILI, EXCLUSIVNO ILI, NE
- **POMERANJE** (aritmetičko / logičko)
- **ROTACIJA**
- **POREĐENJE** većina procesora definiše bit stanja koji informiše o rezultatu aritmetičke operacije Neki procesori imaju instrukciju poređenja **COMPARE**, koja u stvari obavlja oduzimanje bez modifikacije referentne vrednosti.
- **PETLJE HARDVERSKE PETLJE**
- **GRANANJE POZIV POTPROGRAMA I POVRATAK** branch, goto, jump jump-to-subroutine
- **GRANANJE** uslovno i bezuslovno, zakašnjeno / više ciklično

17. Interna predstava podataka - Format podataka

- Mašinske instrukcije različitog tipa koriste različite tipove podataka. Stoga je neophodno definisati njihov format - internu predstavu u računaru, tačnije način binarnog kodiranja podataka, takav da omogući i olakša njihovo čuvanje i obradu u digitalnom računaru.

  Svi podaci u računaru UVEK su predstavljeni nekim binarnim kodom, nekom binarnom sekvencom.

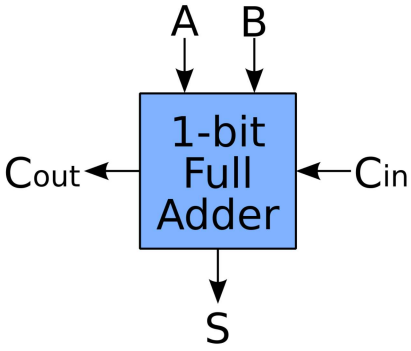
  Prvi i osnovni tip podataka u računaru su **numeričke vrednosti** u jednoj od dve forme:
  - **celobrojni** i
  - **realni brojevi**.

18. Interna predstava podataka - Celi brojevi

- **Celi označeni brojevi** (**integers**) su osnova funkcionalnosti digitalnih računara. Njihov format je određen tako da olakša izvođenje osnovnih aritmetičkih operacija, sabiranja i oduzimanja. Samim tim, olakšana je i realizacija **ALU** jedinice.

  Celobrojne vrednosti u računaru su predstavljene **znakom** i **vrednošću**, gde je znak po pravilu bit najveće težine (**MSB**).

  Nula u bitu znaka (S - sign) označava **pozitivan broj**, **jedinica negativan**. Preostali biti definišu vrednost celog broja, pri čemu se kod negativnih brojeva vrednost zadaje njenim dvostrukim komplementom. Ovo je vrlo važno, jer se tako oduzimanje celih brojeva svodi na njihovo sabiranje.



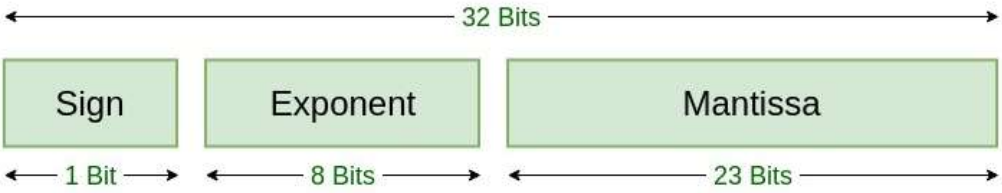
Slika 8: 1-bitni puni sabirač

19. Interna predstava podataka - Realni brojevi

- **Format realnih brojeva**, ili brojeva u tekućem zarezu (real, floating point), određen je standardom IEEE 754.

  Binarna predstava realnog broja obuhvata tri polja:
  - **znak**,
  - **eksponent** i
  - **mantisu**.

  Varijante formata (razlikuju se po dužini polja **eksponenta** i **mantise**)
  - **jednostruke** i
  - **dvostruke preciznosti**.



Slika 9: IEE754 Standard

Znak je bit **najviše težine**, iste pozicije i istog značenja kao kod celih brojeva.

Eksponent je pomeren za konstantnu vrednost, jednaku polovini predviđenog opsega (127 lli 1023), u zavisnosti od preciznostl broja čime je eliminisana potreba čuvanja znaka eksponenta.

Mantisa je normalizovana, pri čemu se vodeća jedinica ne čuva.

Aritmetička kola koja realizuju operacije sabiranja, oduzimanja, množenja i deljenja brojeva u pokretnom zarezu su značajno složenija u odnosu na kola za binarnu aritmetiku.

Savremeni procesori uglavnom poseduju numeričku jedinicu ovog tipa, poznatu kao numerički koprocesor ili floating-point unit.

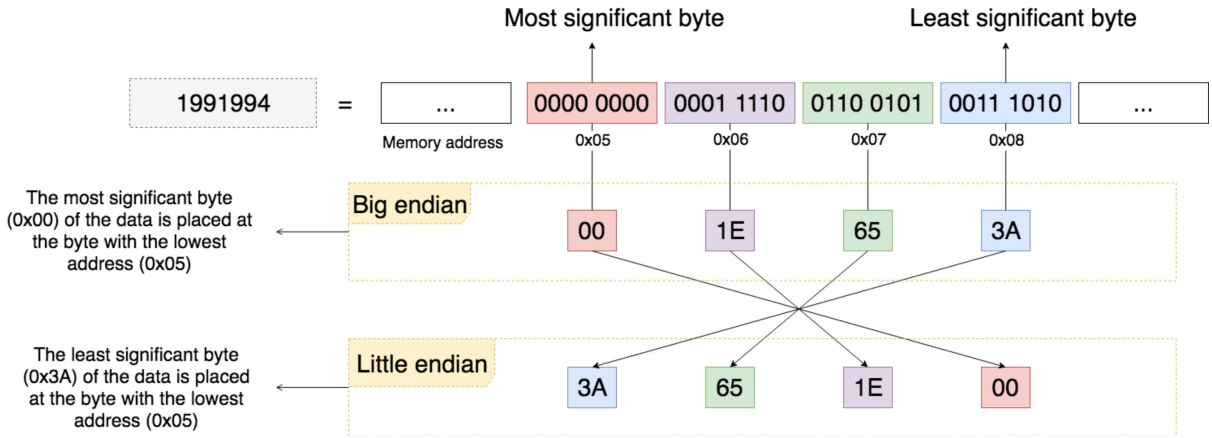
20. Little Endian/Big Endian

- Ukoliko je neki podatak (tipa int, float ili double) duži od jednog bajta, raspoložive su dve opcije za njihovo slaganje u memoriji, u jednoj ili više uzastopnih reči.

  Prva podrazumeva da se u **prvi bajt memorije smesti bajt najveće težine**, pa potom svi ostali. Kako je na najmanjoj adresi u memoriji smešten bajt najveće težine, ovo rešenje se označava kao **Big Endian**.

  Druga opčija je suprotna, jer **polazi od bajta najmanje težine**, te se naziva **Little Endian**.

  Iako oba rešenja imaju neke prednosti u odnosu na ono drugo, ne postoji ubedljiv razlog niti dogovor oko toga koje je rešenje bolje.



Slika 10: Little Endian i Big Endian

21. Organizacija registrara (vrste, namena, ...)

- **Registri opšte namene** su registri čija je bitska dužina jednaka procesorskoj, samim tim i memorijskoj reči. Zbog toga su oni primarno namenjeni čuvanju podataka, te se zato označavaju i kao registri podataka. U slučaju da su adrese iste dužine kao i podaci, što

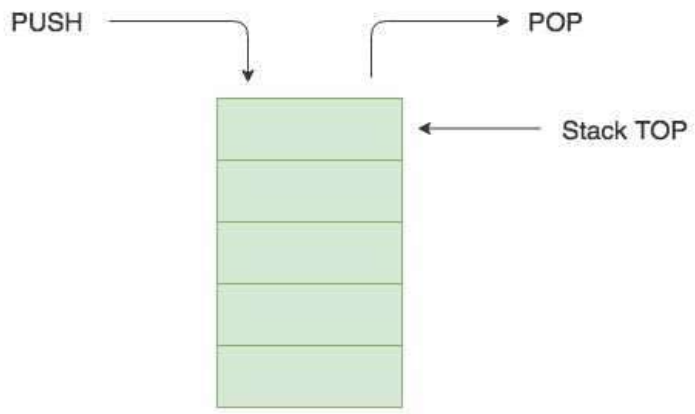
je uglavnom tačno kod **32-bitnih** i **64-bitnih** mikroprocesora, isti registar može prihvatiti oba tipa informacije.

Adresni registri su neophodni ako su adrese duža u odnosu na podatke, što je tipično kod starijih i skromnijih mikroprocesora. Kod 8-bitnih i 16-bitnih procesora adrese su uglavnom široke 16, 20 ili 32 bita.

Programski brojač (Program Counter, PC) je specijalni registar koji čuva adresu tekuće i sledeće instrukcije. Pre učitavanje tekuće instrukcije ukazuje na njenu lokaciju, nakon čega se uvećava, i zato u toku izvršne faze ukazuje na sledeću instrukciju. Realizuje se kao brojački registar, koji pored standardnog uvećanja sadržaja ima i mogućnost paralelnog upisa novog sadržaja. Na taj način se u njega može uneti proizvoljna adresa, čime se realizuju programski skokovi.

Instrukcioni registar (Instruction Register, IR) se u fazi pribavljanja instrukcije puni sadržajem učitanim iz memorije sa lokacije na koju ukazuje programski brojač - služi za dekodiranje učitane Instrukcije. **Instrukcioni izvor** je izvor ključnih informacija za upravljačku jedinicu.

Ukazivač steka (Stack Pointer, SP) je registar koji ukazuje na trenutnu poziciju steka, segmenta memorije kojim se upravlja u maniru LIFO liste (Last In First Out), koju prvi napušta element koji je zadnji upisan. Rukovanje stekom podrazumeva upotrebu **registra** (**SP**) i dve specijalne instrukcije (**push** I **pop**),



Slika 11: Organizacija stek memorije

Registar SP ukazuje na prvu slobodnu lokaciju steka, tj. na njegov vrh, na kome se isključivo vrši dodavanje (push) ill skidanje (pop) sadržaja (informacionih elemenata).

Prilikom poziva potprograma, na stek se smeštaju **povratna adresa** i **argumenti pozvane rutine**. Pozvana rutina može na njemu sačuvati sve registre koje će koristiti (izmeniti), kao što na steku može odvojiti prostor za svoje lokalne promenljive. Pri povratku, treba **restaurirati izmenjene registre**, napuniti **povratnu adresu** u **programsk brojač**, i **vratiti ukazivač steka na početnu poziciju**.

Registar stanja (Status Register, SR) smešta programsku reč stanja procesora, koja se sastoji od kontrolnih i indikatorskih bita. Kontrolni biti se postavljaju od strane programa u cilju omogućavanja izvesnih režima rada centralnih procesora. Standardan je bit dozvole prekida (Interrupt Enable), čijim postavljanjem se omogućuje ili sprečava pojava U/I prekida. Pored njega tu su često biti koji postavljaju režim rada procesora (Kernel/User, Little/Big Endian) ili memorije (Memory Mode), i slično.

Indikatorski biti se postavljaju automatski na osnovu nekog **događaja** u toku izvršenja aritmetičkih i logičkih operacija, zbog čega se uz ALU dodaje poseban registar.

Minimalan registar stanja obuhvata četiri indikatorska bita koji služe za detekciju i naknadnu proveru pojave prenosa, aritmetičkog prekoračenja, znaka ili nulte vrednosti rezultata.

Biti stanja:

- **Z-Zero Bit** koji pokazuje da li je rezultat nula (tada je Z=1).
- **S-Sign Bit** koji pokazuje znak rezultata (**1 - ne.**, **0 - poz**).
- **C-Carry Bit** označava prekoračenje fizičkog opsega registra.
- **V-Overflow Bit** označava prekoračenje kod označenih brojeva

Trenutna vrednost svakog od bita može se proveriti pomoću instrukcija uslovnog grananja, zbog čega se registar stanja nekada naziva i **registrom uslovnog skoka**.

U minimalnoj varijanti, ove instrukcije realizuju uslovna grananja direktno nad registrom stanja (BC-branch if carry, BZ-branch if zero, Itd.).