-Organizacija računara-

Skripta – I test

Generacija računara

-6 generacija računara, razlika je pre svega u tehnološkim osnovama

-4., 5. i 6. generacija se preklapaju, nema jasnih granica

1. Generacija

-Karakteristike: Elektronske cevi, releji, otpornici, velika potrošnja struje, ogromne dimenzije

-**1944**. - prvi računar opšte namene **Mark I** i korišćen je na harvardu 15 godina. Imao je 150 000 različitih komponenti i težio je 5 tona.

-**1946.** – **ENIAC** (Electronic Numerical Integrator and Computer). Sastojao se iz 18 000 elektronskih cevi i 70 000 otpornika.

-**1951.** – **UNIVAC** (Universal Automatic Computer). Karakteristike: programi su bili smešteni u memoriju, uređaj spoljne memorije je bio na magnetnoj traci, postojali su ulazno/izlazni uređaji (tastatura, bušene kartice, štampač), sastojao se od 54 000 elektronskih cevi.

2. Generacija

-Pojavili su se tranzistori što je dovelo do smanjenja dimenzija računara, smanjenja potrošnje električne energije i povećanja brzine

-Predstavnik: **IBM 1401**; imao je 4KB magnetne memorije. Ulaz u računar su bile bušene kartice, trake, teleprinterska kola.

3. Generacija

-Pojavila su se integrisana kola (1960.) koja su dovela do smanjenja dimenzija računara, smanjenja potrošnje struje i ubrzanja rada računara.

-Predstavnik: **IBM System/360**. Imao je tastaturu, diskove i 8-bitnu memoriju s magnetnim jezgrima veličine do 6MB.

4. Generacija

-Pojavila su se VLSI (Very Large Scale Integration) kola koja su sadržala milione komponenti u integrisanim kolima, mirkoprocesor je bio izdvojen u jednom integrisanom kolu (Intel 4004) i njima su se smanjile dimenzije računara i potrošnja struje.

-Predstavnik: IBM PC računar. Karakteristike: 8086 mikroprocesor, 64KB memorije (max 640), 360KB flopi disk, 10MB hard disk.

5. Generacija

-Dosta se primenjuje veštačka inteligencija, virtuelna realnost i intenzivan je razvoj računarskih mreža.

6. Generacija

-Počinje se sa primenom optike, biočipovi (OLED)

-Dramatični razvoj računarskih mreža koji dovodi do poboljšanja u domenu mrežne opreme i brzina prenosa informacija u komunikacionim linijama.

Osnovni pojmovi

-**Hardver** (tehnička podrška) je oprema/oruđe koja predstavlja fizičku (materijalnu) realizaciju bilo kog sistema koji obavlja određene funkcije.

-**Softver** (programska podrška) predstavlja sveukupnost instrukcija, programa i drugog "operativnog oruđa" koji se generiše, aktivira i koristi na razne načine da bi omogućio hardveru da reši dati problem i obavi željene poslove. On zapravo upravlja hardverom.

Elementi hardvera: (PITANJE ZA TEST!)

1. Memorija
2. Procesor
    a. Upravljačka jedinica
    b. Aritmetičko-logička jedinica
    c. Registri
    d. Ultra brza memorija (keš)
3. Ulazno/izlazni podsistem (kontroleri)
4. Periferijski uređaji (miš, tastatura, štampač,..)

1. Memorija je mesto gde čuvamo i predajemo podatke i programe.
-**Memorijski element** je element koji pamti elementarnu informaciju (veličina – 1bit)

-**Memorijska lokacija (ćelija)** je više udruženih memorijskih elemenata.

-**Memorijski modul (blok)** je skup memorijskih ćelija. Memorijski modul je najčešće **adresabilan** gde **adresa** predstavlja jedinstveni prirodan broj dodeljen svakoj ćeliji.

-Memoriju delimo po različitim aspektima:

- Sa aspekta pristupa:
    - Sekvencijalni pristup (magnetna traka, bušena traka..)
    - Ciklični pristup (hard disk, floppy disk, CD-ROM)
    - Proizvoljni pristup (RAM – Random Access Memory)
- Sa aspekta mogućnosti promene:
    - Promenljiva memorija (RAM memorija)
        - Statička (pamti sadržaj dok ima struje)
        - Dinamička (pamti sadržaj dok ima struje, ali mora se osvežavati)

    - Polupromenljiva memorija
        - PROM – Programmable Read Only Memory,
        - EPROM – Erasable PROM
        - EEPROM – Electrically EPROM)
        - flash memorija
    - Stalna memorija (ROM – Read only memory) – danas je skoro niko ne koristi
- Sa aspekta smeštanja sadržaja
    - Adresne (pristup memorijskoj ćeliji preko adrese)
    - Bezadresne (asocijativna memorija – hash mapa (ključ : vrednost), stek memorija,...)

-Memorija u računaru se deli na: operativnu memoriju, ultra brzu memoriju (cache) i jedinicu spoljne memorije (masovnu memoriju) što su npr. hard disk, USB pocket drive, DVD, CD-ROM, floppy disk…

2. Procesor: izvršava operacije obrade podataka definisanje programom i vrši upravljanje računarskim procesima i interakcija između pojedinih jedinica računara (mozak računara za izvršavanje programa)

a. Upravljačka jedinica – upravlja pojedinim koracima u obradi podataka i to na osnovu informacija sadržanih u instrukciji koju upravljačka jedinica zahvata iz memorije. Sinhronizuje U/I jedinice, memoriju i aritmetičko-logičku jedinicu. Postoje dva pristupa: direktan (hardverski) i mikroprogramski. Ona izbacuje adresu lokacije koju procesor traži iz memorije i kad dobija te podatke počinje sa izvršavanjem programa.
b. Aritmetičko-logička jedinica - izvršava aritmetičke operacije (sabiranje, oduzimanje,..), logičke operacije (konjukcija, disjunkcija, negacija, ...), pomeranje bitova i drugo. Elementi:
    1) kombinacione mreže (sabirači)
    2) registri u kojima se čuvaju operandi, međurezultati i rezultati operacija
    3) pomoćni registri (statusni registar, i dr.)
c. Registri – imamo više vrsta npr. radni (Akumulator,..), upravljački (PC, MAR, MBR,..)
d. Ultra brza memorija – integrisana u procesor, veoma je brza i skupa. Ova memorija se nalazi između operativne memorije i procesora, Registri procesora takođe se mogu smatrati za ultra brzu meoriju. Ovaj koncept se primenjuje i kod sporijih periferijskih jedinica (bafera)

3. Ulazno/izlazni podsistemi: omogućavaju razmenu poruka između računara i spoljašnjosti. Postoji više načina za njegovu realizaciju:

- Programiranji U/I – procesor stalno proverava status periferijske jedinice i uzima i šalje podatke iz i ka periferijske jedinice. Mana je ta što troši puno procesorskog vremena i zato se dosta retko koristi.
- Sistem prekida (interrupt) – omogućava da se regularni program prekine ako se pojavi određeni događaj i onda upravljanje procesa prenese na program za obradu tog događaja i po završetku tog programa nastavi sa izvršenjem regularnog programa. Koristi se kod prenosa manje količine podataka, ali je dosta sporo kod velike količine podataka.
- Direktan memorijski pristup - periferijska jedinica “sama” upisuje podatke u operativnu memoriju (ili čita iz nje), bez intervencije procesora. Završetak posla se prijavi procesoru upotrebom prekida. Korisno kod prenosa velikog broja podataka (tu su i programirani U/I i sistem prekida spori)

4.Periferijske jedinice: služe za ulaz/izlaz podataka . Dele se na: ulazne jedinice (tastatura, miš,..), izlazne jedinice (štampač, monitor, ploter,..) i ulazno/izlazne jedinice (hard disk, mrežna i zvučna kartica).

-Softverske karakteristike računara:

1) Aplikacioni (korisnički) programi - skup programa koji se obično koriste i/ili razvijaju korisnici za rešavanje svojih specifičnih problema. Neke klase: DTP, CAD/CAM, Word processors, Internet (WWW, Email, Chat,..)
2) Sistemski (upravljački) programi – programi za organizaciju i upravljanje procesima rada računarskog sistema. Vrste: operativni sistemi, uslužni programi (dijagnostika, ..), simulatori računarskih sistema, programski sistemi za programiranje,…

Matematičke osnove rada računara

Brojni sistemi

-Može biti nepozicioni i pozicioni.

-Nepozicioni brojni sistem je rimski brojni sistem (rimski brojevi)

-Kod pozicionog brojnog sistema svaka cifra ima zadatku težinu. Opšti oblik:

$$a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \ldots + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + a_{-2} \cdot b^{-2} + \ldots + a_{-m} \cdot b^{-m}$$

a – cifra
b – baza (osnova)
n+1 – broj celobrojnih cifara
m – broj decimal

-Cifre brojnog sistema su u opsegu: 0 – baza-1

-Primer:

- Binarni (b=2) – 1, 0
- Oktalni (b = 8) – 0, 1, 2, 3, ...,7
- Decimalni (b = 10) – 0, 1, 2, 3, ..., 9
- Heksadecimalni (b = 16) – 0, 1, 2, 3, 4, ..., 9, A, B, C, D, E, F

-Računske operacije u binarnom brojnom sistemu:

- Sabiranje – samo kada su oba sabirka 1, tada postoji prenos i iznosi 1
- Oduzimanje – samo kada od 0 oduzimamo 1, postoji prenos i iznosi 1
- Množenje – samo kada su oba činioca 1 rezultat je 1, u ostalim slučajevima je proizvod 0
- Deljenje – nulom nije dozvoljeno, a jedinicom je trivijalno

-Konverzija brojnih sistema:

Opšta formula:

-celobrojni deo:

celobrojni deo (a) u novu bazu b:

a : b = r1 i ostatak o1

r1 : b = r2 i ostatak o2

r2 : b = r3 i ostatak o3

...

rn : b = 0 i ostatak on

---

rezultat: on ... o3 o2 o1

– razlomljeni deo:

razlomljeni deo (a) u novu bazu b:

a · b = c1,r1 tj. celobrojni deo c1 i razlomljeni deo r1

r1 · b = c2,r2 tj. celobrojni deo c2 i razlomljeni deo r2

r2 · b = c3,r3 tj. celobrojni deo c3 i razlomljeni deo r3

...

rn · b = cn,0 tj. celobrojni deo cn i razlomljeni deo 0

Rezultat: c1 c2 ... cn

*Na kraju spojimo celobrojni i razlomljeni rezultat i to je konačni rezultat

*Pogledati zašto je problem ako razlomljeni deo ne bude 0!

-Konverzija iz binarnog u oktalni i heksadecimalni brojni sistem: drugačije grupisanje bitova:

– po tri bita za oktalni brojni sistem

– po četiri bita za heksadecimalni brojni sistem

<u>Predstavljanje celobrojnih brojeva u računaru:</u>

-Svaka memorijska ćelija u računaru ima 8 bitova – jedan bajt. U jedan bajt se može smestiti broj u rasponu od 0 – 255. Ako je celobrojna vrednost veća od 255, uzme se više bajtova:

– dva bajta – 16 bita: 0 – 65535

– četiri bajta – 32 bita: 0 – 4.294.967.295

-..uvek se uzima broj bajtova koji su stepen broja 2!

-Negativni brojevi se u računaru predstavljaju na dva načina: preko znaka i apsolutne vrednosti i preko komplemenata. Algoritam za znak i apsolutnu vrednost je dosta kompleksan za operacije sabiranja i oduzimanja, pa se stoga ovi brojevi predstavljaju preko komplemenata.

-Potpuni komplement (u binarnom brojnom sistemu se još zove i komplement dvojke). **Nepotpuni komplement** (u binarnom brojnom sistemu se još zove i komplement jedinice). U oba sistema se poslednja cifra koristi za znak broja (pozitivan ili negativan).

-Računanje komplementa:

-Potpuni komplement: invertovati sve bitove i dodati 1.

Primer: 00102 => 11012 + 1 = 11102

-Nepotpuni komplement: invertovati sve bitove.

Primer: 00102 => 11012

-Sabiranje sa potpunim komplementom je zapravo oduzimanje.

-**Prekoračenje (overflow)** se javlja kada se prilikom sabiranja dva broja dobije rezultat koji ne može da stane u zadati broj bitova. Pravilo: ako se prilikom sabiranja dva pozitivna ili dva negativna broja dobije broj suprotnog znaka, dogodilo se prekoračenje.

Predstavljanje realnih brojeva u računaru

-U nepokretnom zarezu: fiksna pozicija decimalnog zareza.
-U pokretnom zarezu (floating point): brojevi se predstavljaju u obliku: $\mathbf{m \cdot b^e}$

m – mantisa
b – baza
e – eksponent

-U memoriji računara se pamte mantisa i eksponent kao celobrojne označene vrednosti, najčešće sa bazom 2

-Pokretni zarez (floating point)

- Sabiranje odn. oduzimanje - pre sabiranja (oduzimanja) brojevi se svedu na isti eksponent:
  $m1 \cdot be + m2 \cdot be = (m1 + m2) \cdot be$
- Množenje, odn. deljenje:
  $(m1 \cdot be1) \cdot (m2 \cdot be2) = (m1 \cdot m2) \cdot b(e1+e2)$
- Svođenje eksponenata na istu vrednost se svodi na povećanje manjeg eksponenta, uz istovremeno deljenje mantise bazom – u računaru se deljenje matise bazom 2 svodi na pomeranje desno bitova.
- U praksi se normalizacija mantise svodi na zapis: 1,xxxx, gde se 1 podrazumeva i tada je preciznost najveća

-Pokretni zarez u računarima u nekim situacijama nije dovoljno precizan, a razlog je taj što je baza 2, pa konverzija decimalnih brojeva u oblim $m * 2^e$ ne daje okrugao broj. Npr. 0,1 se može zapisati i sačuvati, s tim što će greška biti tamo negde na 15toj decimali, što znači da ovaj je ovaj broj u samom startu u našem računaru netačan! Zbog ovog problema se u bankama ne koristi floating point.

Kodiranje alfanumeričkih informacija

- Alfanumerički simboli: numerički simboli (0, 1, ..., 9), slovni simboli (A, B, ..., Z), inteprunkcijski znakovi (, . ; : " ...), specijalni simboli (#, $, %, ...)

- Standardi: ASCII (American Standard Code for Information Interchange), ISO 8859-1, Windows CP 1250, Unicode

-Unicode: Jedno slovo/karakter je u Unicode standardu trenutno 20-bitno i zove se code point. Postoje tri reprezentacije:

- UTF-8 gde je svaki bajt za neki raspon vrednosti
- UTF-16 gde se svaki code point prikazuje code unitima, koji su 16bitni
- UTF-32 gde se za svako slovo uzima 4 bajta

Kodovi za detekciju i korekciju grešaka

-Sve informacije će biti kodirane binarno.

-Najstariji tip detekcije je dodavanje jednog bita (bit parnosti). Ovde se može detektovati greška, ali se ne može korigovati. Problem: greške od više od jednog bita mogu proći nedetektovane.

-Rešenje: dvodimenzionalno dodajemo bitove (i u vrsti i u koloni) – ni ovo nije idealno, ali ovde je moguće detektovati i korigovati grešku.

-CRC (Cyclic Redundancy Character) kod:

- Poruka se kao niz bitova deli sa nekim unapred dogovorenim brojem, rezultat se odbacuje a ostatak pri deljenju se doda uz poruku.
- Na prijemnoj strani se primljena poruka deli istim brojem i ostatak se poredi sa primljenim ostatkom.

Digitalna elektronika

-Komponente se mogu podeliti na pasivne (diode, otpornici, kondenzatori..) i aktivne (tranzistori, integrisana kola,…).

-Pasivne komponente imaju fiksan način rada (predefinisan), dok aktivne komponente mogu da menjaju naponske i strujne karakteristike kola u koje su ubačene odnosnu mogu menjati svoju funkciju na varijabilan način.

-Tranzistori su poluprovodničke komponente koje mogu da pojačaju signal tj. pojačavaju protok struje kroz sebe. Oni su osnovna jedinica građe cele modern elektronike danas. Digitalne komponente se prave od tranzistora.

-Najgrublja podela digitalnih komponenti je na kombinatorne i sekvencijalne komponente.

-Kombinatorne komponente su kombinacija digitalnih komponenti kod kojih stanje izlaznih signala zavisi isključivo od ulaznih signala.

-Sekvencijalne komponente su kombinacija digitalnih komponenti kod kojih stanje izlaznih signala zavisi ne samo od ulaznih signala, već i od tekućeg stanja memorijskih komponenti.

-Svaki računar danas je kombinacija i kombinatornih i sekvencijalnih komponenti.

-Razvijanje digitalnih kola se može odvijati:

1) Faza prototipa – ako nije previse komplikovano, možemo ih spajati preko proto board-a što predstavlja plastiku koja ima rupe koje su međusobono povezane po nekim grupacijama
2) Faza prototipa – wire wrapping
3) Faza prototipa/konačni proizvod – programibilna logička kola koja služe za najsloženije stvari i njeni tipični predstavnici su FPGA unutar kojih se nalazi na desetine hiljada komponenti koje su potpuno programibilne
4) Izrada proizvoda – finalni proizvod zahteva izradu štampane ploče koja predstavlja plastičnu ploču koja ima bakarne rupe kroz koje prolazi sve što je potrebno
5) Proizvodnja kola

Logička kola

-Logička kola realizuju funkcije Bulove algebre (true/false) . U kompjuteru se sve svodi na 0, ili ako je više volti na 1.

-Koriste se da bi se kreirale složenije mreže. Niko ne pravi računare sa dekadnim brojnim sistemom jer je to užasno teško, uvek se radi sa binarnim brojnim sistemom i za to se koristi Bulova algebra.

-Ima oko 16 vrsta funkcija, a mi ćemo koristiti sledeće navedene: AND, OR, NOR, XOR (osnovne), NAND (NI – negirano i), NOR, XNOR

*Na prezentacij pogledati funkcije AND, OR, NOT i njihove notacije. Ova tri kola su dovoljna da napravimo sve od digitalne elektronike.

-Ekskluzivno ili je vrlo slično sa OR, samo što kada su oba ulaza jedinice, izlaz je 0. Generalno pravilo je da kada su ulazi jednaki, izlaz je uvek 0. (notacija na slajdu)

-Funkcija NAND (NI) je negacije konjukcije i tehnološki se lakše pravi od I kola. (notacija na slajdu)

-Funkcija NOR (NILI) je negacija disjunkcije i tehnološki se lakše pravi od ILI kola. (notacija na slajdu)

-Funkcija NXOR (isključivo NILI): pogledati na slajdu.

-Ljudi su napravili ove dodatne funkcije jer one zapravo prirodno nastaju, a one koje smo naveli kao osnovne I i ILI nastaju invertovanjem NI i NILI kola.

-Bafer je kolo koje služi za unificiranje naponskih i strujnih karakteristika ulaza i izlaza. Ono što je na ulazu, prosleđuje se na izlaz tj. pojačava ga što znači da daje jaču struju. (notacija na slajdu)

-Tristate kola imaju tri stanja: 0, 1 i isključenost (visoka impedansa). Služe da kada se priključi veći broj logičkih kola na zajednički vod (sabirnicu, magistralu, bus).
-U stanju isključenosti, kada bi hteli da propustimo nešto, zahvaljujuči kvačici OE (output enable) to možemo i uraditi: ako je ulaz 1, kolo propušta ulaz, ako je 0, kolo ima isključen izlaz. (notacija na slajdu)

-Kombinatorna kola-

-Vrste kombinatornih mreža:

1) Koder
2) Dekoder
3) Multiplexer
4) Demultiplexer
5) Polusabirač
6) Sabirač

1) **Koder** – stanje logičkih jedinica na jednom od 2^n ulaza reprezentuje se binarnom reči dužine n. Numeracija ulaza počinje od nula. Problem nastaje kada nijedan ulaz nije aktivan ili je aktivan nulti ulaz, to onda daje isti izlaz. Ovo se rešava tako što ćemo imati jedno ILI kolo koje prvo gledamo u slučaju da je nastala neka promena, da bi znali da li je vredno gledati rezultat izlaza. (notacija slajd)
2) **Dekoder** – obrnuto od kodera. To je binarna reč od n bitova na ulazu koja izaziva aktivaciju jednog od 2^n izlaza. (notacija slajd)
3) **Multiplexer** – zadatak je da jedan od 2^n informacionih ulaza sprovede direktno na izlaz. Koji će ulaz proći određeno je pomoću n adresnih ulaza odnosno kontrolnim selektivnim ulazima. Uvek imamo jedan izlaz! (notacija slajd)
4) **Demultiplexer** – zadatak je da jedan ulaz preusmeri na jedan od 2^n izlaza, a to se takođe kontroliše pomoću n adresnih ulaza. (notacija slajd)
5) **Polusabirač** – obavlja funkciju sabiranja dva bita (notacija slajd)
    - Ulaz: 2 bita
    - Izlaz: rezultat i prenos (carry)
    - Ovo ne rešava problem potpunog sabiranja brojeva jer nema ulazni prenos
    - Ulazni prenos je bitan jer ako treba da sabiramo 2bitne, 8bitne i veće brojeve, ne možemo to realizovati polusabiračem jer nemamo taj ulazni prenos
6) **Sabirač** – obavlja funkciju sabiranja dva bita (notacija slajd)
    - Ulaz: bit 1, bit 2 i ulazni prenos
    - Izlaz: rezultat i prenos (carry)
    - Pomoću ovog jednobitnog punog sabirača možemo napraviti višebitni sabirač
    - Višebitni sabirač – ako sabiramo dva binarna broja, svaka dva bita sabiramo jednim 1-bitnim sabiračem, a prenos iz nižeg bita ubacujemo kao ulazni prenos višeg bita

-Sekvencijalna kola-

-Vrste sekvencijalnih mreža:

1) Leč (RS)
2) Flip-flop (RS flip-flop, JK flip-flop, D flip-flop)
3) Brojači
4) Registri
5) Pomerači

1) **RS leč** – osnovna memorijska jedinica koja ima dva ulaza: R (reset) i S (set) i ima dva izlaza.
   - Leč i flip flop su memorijski elementi (memorijska kola), samo je razlika u načinu na koji mi postavljamo podatak u memoriju.
   - Lečevi nemaju clock takt, a flip flop ima.
   - Leč čuva informaciju tako što dogle god ima napajanja u njemu, on će se puniti.
   - Lečevi se danas ne koriste jer nam je potreban takt koji će definisati tempo promena i prelaz iz jednog stanja u drugo.
   - Iz RS leča su nastavi RS flip-flopovi (okidan nivoom)
2) **Flip-flop**
   a. **RS flip-flop** – uveden je takt što znači da se promene na leču događaju samo u skladu sa taktom. Promena stanja je dirigovana CL ulazom. Samo u slučaju kada je CL 1, prihvata se promena stanja R i S ulaza. Čim CL padne na nula, flip flop je u hold stanju. *Clock je zapravo smišljen da sinhronizuje promene svih logičkih kola u istom trenutku.
   b. **JK flip-flop** (okidan ivicom) – ispravlja nedostatak RS flip-flopa odnosno pri istovremenom stanju ulaza J i K na 1, menja stanje u komplement
   c. **D flip-flop** (okidan ivicom) – osnovni gradivni element svakog računara. Bez obzira na prethodno stanje, šta god da je na ulazu, prelazi na izlaz u sinhronizaciji sa taktom.
3) **Brojači** – obično broji ivice takta ili signala. Kapacitet brojača: broj različitih stanja u jednom ciklusu brojanja – jedan ciklus je broj impulsa posle kojeg se brojač nađe u istom stanju. Postoje dve realizacije:
   a. **Redni** – ima 4 bita oi unutra su JK flip-flopovi. Koliko promena na ulazu izbroji, taj broj će binarno ispisati na izlazu. Kapacitet brojača je 2^n. Mana mu je kašnjenje jer se sporo menjaju vrednosti i gotovo nikad nemamo potpuno tačno pravo stanje.
   b. **Paralelni** – ispravlja manu rednog brojača jer se izlaz svih bitova dobija istovremeno. Clock je zajednički za sve flip-flopove istovremeno i na njegov udarac dobijamo konzistentno stanje. Ovim brojačem se minimizuje kašnjenje.
4) **Registri –** svi registri se sastoje od jako puno flip-flopova naslaganih redom za svaki bit i oni služe za privremeno memorijsanje podataka. Registar neće primiti informaciju sve dok load ne postane 1. U jednom trenutku samo jedan OE (output enable) sme imati vrednost 1 i o tome vodi računa upravljačka jedinica!
   a. **Registar sa magistralom** – clock dolazi do svih D flip-flopova i svi ste istovremeno okidaju
5) **Pomerački registar** – omogućuje pomeranje bitova u levo (to realizuje množenje te promenljive sa bazom) ili desno (realizuje deljenje te promenljive sa bazom). 4-bitni pomerački registar je kombinacija dva 2bitna, s tim što je izlaz (najviši bit) istovremeno serijski ulaz za sledeći bit.

- Potencijalni problem bi bio da ako nešto treba da se shiftuje 8 puta, treba nam 8 taktova a to znači da što više treba da shiftujemo, više moramo da čekamo da se ta operacija izvrši u procesoru.
- Zbog ovog problema su i smišljenji 4-bitni flip flopovi koji momentalno shiftuje više bitova u jednom taktu procesora, a tih više bitova je unapred zadato. Ovakva realizacija se radi samo uz multiplexere, nema flip-flopova i ovako se ugrađuje u današnje procesore!
- Flip-flopovi se u računarima koriste:
    - ❖ U registrima – svaki registar je zapravo D flip-flop. Npr. 16 D flip-flopova ima zajednički load i clock.
    - ❖ U ultra brzoj memoriji (cache) – statička memorija je napravljena od D flip-flopova; ova memorija je 1000x manja od cele statičke memorije.

Procesor

-Procesor se sastoji iz:

- aritmetičko-logičke jedinice
- upravljačke jedinice
- registara
- keš memorije

-Sva komunikacija između procesora i ostatka računara se vodi preko:

- adresne sabirnice (address bus)
- sabirnice podataka (data bus)
- upravljačkih signala (control bus)

-Vrste registara:

- radni (vidljivi iz programa) – mi manipulišemo njima, smeštamo i čitamo promenljive iz njih; brži su od memorije i bolje ih je koristiti
- upravljački (kontrolni i statusni) – unutar procesora, ne možemo ih direktno koristiti; indirektno se koriste pisanjem i izvršavanjem instrukcija od strane kontrolnih jedinica i specijalnih programa

-**Radne registre** možemo podeliti na:

- specijalnizovane – akumulator
- opšte namene – promenljive, učestvuju u operacijama
- za podatke – smeštanje podataka, ne učestvuju u operacijama
- za adresiranje – segmentni, adresni, SP (Stack Pointer)
- statusni (flegovi) – bitovi se automatski postavljaju prilikom izvšenja operacija, a iz programa se uglavnom može samo pročitati vrednost

-Dileme su uglavnom oko toga koje je registre bolje imati u većoj količini, recimo bolje je da imamo više registara opšte namene nego specijalizovanih. Ova dilema je zapravo sa stanovništva realizacije jer ako hoćemo više registara opšte namene, potrebno je onda da ugradimo veći broj tranzistora, a to je nekad tehnički neizvodljivo.

-**Upravljački registri** – da bi našao željeni program, on treba na magistralu da izbaci binarni broj koji predstavlja lokacijsku memoriju programa koji želi čitati. Imamo:

- Programski brojač (PC – Program Counter)
- Instrukcijski registar (IR – instruction register)
- Memorijski adresni registar (MAR – memory address register)
- Memorijski bafer registar (MBR – memory buffer register)
- Statusni registar (PSW – program status word)
    - znak (sign – znak poslednje aritmetičke operacije)
    - nula (zero – ako je rezultat 0)
    - prenos (carry – ako imamo prenos kod aritmetičkih operacija)
    - jednaki (equal – rezultat poređenja dve vrednosti)
    - prekoračenje (overflow – aritmetičko prekoračenje) – kada se kao rezultat aritmetičke operacije dobije broj koji ne može da stane u registar
    - prekidi (interrupt enable/disable – uključuje/isključuje prekide)
    - privilegovani/korisnički način rada (supervisor/user mode – neke instrukcije se mogu izvršiti samo iz privilegovanog režima rada

-**Instrukcijski ciklus** – kako procesor nabavlja i izvršava različite instrukcije. Tok ove radnje:

- prvo dobavlja prvu instrukciju (nakon reseta) iz memorije – **fetch instruction**
- kada to uradi, onda intepretira naredbu odnosno treba da je dekodira kako bi utvrdio koju akciju treba da izvrši
- kreće da je izvršava, i u slučaju da treba da dobavi neke podatke, on to i radi – **fetch data** (ovo je opciono, ne mora da se desi, samo ako za to postoji potreba)
- procesiranje podataka – izvršavanje AL operacija nad podacima (ako je potrebno)
- upisivanje podataka - rezultat se upiše u memoriju ili U/I jedinicu (ako je potrebno)

*U Asembleru naredba koja ne radi ništa je **no (no operation)** – ona ima samo prve dve stavke od ovih prethodno navedenih.

-Registar u kome se čuva adresa sledeće/tekuće instrukcije je **PC (Program Counter).**

-Svaka mačinska instrukcija ima sledeće elemente:

- svoj kod (kod operacije) – šifra instrukcije (binarni broj)
- referencu na izvor – ko učestvuje u operaciji (registri, U/I, memorija)
- referencu na rezultat – gde se smešta rezultat
- referencu na sledeću instrukciju – uglavnom se ona dobija sa memorijske lokacije koja sledi nakon tekuće, osim u slučaju kada imamo skokove ili grananja

-Prema broju adresa, instrukcije delimo na:

- 0 adresne – npr stavljanje na stek (SP će se sam pomeriti); bez adrese
- 1 adresne – učitavanje nekog broja u akumulator
- 2 adresne – npr. sabiranje vrednosti 2 registra i smeštanje te vrednosti u jedan od ta dva registra
- 3 adresne – npr. dva registra i rezultat operacije smeštamo u neki treći (2 ulaza i 1 izlaz, različiti)

-Tipovi instrukcija:

1) **Smeštanje podataka** – u/iz memorije; npr move, store (procesor -> memorija), load (memorija -> procesor), exchange (zamena sadržaja), clear, set, push, pop
2) **Procesiranje podataka** – aritmetičke (sabiranje, oduzimanje, množenje, deljenje, apsolutna vrednost, negacija, inkrement, dekrement) i logičke (AND, OR, NOT, XOR, testiranje, poređenje, šiftovanje, rotacija, bitwise operacije) operacije
3) **Kontrola toka izvršavanja** - jump, jump conditional, call subroutine, return from subroutine
4) **U/I operacije** – read, write, start I/O, test I/O
5) **Konverzije** – translate, convert

1) Smeštanje podataka

- Moguće kombinacije:
  - load (iz memorije u registar)
  - store (iz registra u memoriju)
  - move (iz memorije u memoriju)
  - iz registra u registar
- U registar može biti smešten broj ili sadržaj memorijske U/I lokacije
- Načini adresiranja: neposredno (immediate), indirektno (indirect), registarsko (register), registarsko indirektno (register indirect), bazno (displacement) I stack.

2) **Procesiranje podataka –** aritmetičko – logička jedinica izvodi ove operacije

- **Aritmetičke operacije**
  - osnovne matematičke operacije nad celobrojnim i floating point podacima (sabiranje, oduzimanje, množenje, deljenje, apsolutna vrednost, negacija, inkrement, decrement)
- **Logičke operacije**
  - operacije nad bitovima registra (AND, OR, NOT, XOR, testiranje, poređenje, shiftovanje (logičko, aritmetičko), rotacija)

*kod aritmetičkog shiftovanje ulazi sign bit kada se pomera u desno, a u levo sign bit ostaje očuvan, pretposlednji se izbacuje

3) Kontrola toka

- Skok (jump, branch) – ažurira se PC (Program Counter) npr. jmp 123 – stavi u PC 123
- Uslovni skok (conditional jump, conditional branch) – u zavisnosti od stanja flegova u statusnom registru izvršava se skok (ili ne). Primeri: jp, jnp, jz, jo
- Uslovni skok – proveriti uslov i skočiti na zadatu adresu ako je uslov zadovoljen (ili nije)
- Poziv podprograma (call subroutine) - Scenario poziva potprograma:
  - procesor nailazi na instrukciju za poziv potprograma (obično CALL adresa)
  - adresa sledeće instrukcije (posle poziva potprograma) se sačuva
  - skoči se na adresu potprograma – po povratku iz potprograma (obično instrukcija RET ili RETURN) se preuzme adresa sledeće instrukcije
  - skoči se na tu lokaciju

*****adresa sledeće instrukcije se čuva na steku!

- Smeštanje povratne adrese:
  - u registar – najbrže je; komplikacija ako slučajno izmenimo sadržaj registra u kojem se čuvala povratna adresa onda se taj registar se ne sme koristiti i komplikovano kod višestrukih poziva iste procedure (rekurzija)
  - u memoriju (odvojimo specifičnu memorijsku lokaciju za čuvanje povratne adrese) - ne mora da se brine da li ćemo pokvariti povratnu adresu nehotičnom izmenom registra, mada je idalje komplikovano kod višestrukih poziva iste procedure (rekurzija)
  - na stek – najsigurnije je i dobro radi za višestruke pozive
- Smeštanje parametara i lokalnih varijabli – analogno sa smeštanjem povratne adrese, najbolje je na stek.

4) U/I operacije

- Imamo dva pristupa:
  - odvojeni U/I – in i out operacije (Intel)
  - memorijski mapirani U/I – žrtvovana je neka memorijska lokacija kako mi se tu smestio neki ulazno/izlazni podsistem

5) Konverzija

- Translacija – na osnovu tabela korespodenata
- Konverzija – iz NBDC (koristi se u bankama jer tu trebaju precizni brojevi) u binarni

-Byte ordering – rešava problem reprezentacije binarnih reči u memoriji računara ako reč zauzima više od jednog bita. Postoje dva koncepta:

- little endion (Intel) – na nižoj adresi stavljamo niže bitove, a na višoj adresi više
- big endion (ARM) – na nižoj adresi stavljamo više bitove, a na višoj adresi niže

-Načini adresiranja:

1) neposredno (immediate) – npr. upisujemo broj
   - podatak je kodiran u instrukciju ili se nalazi neposredno posle instrukcije
   - prednost je ako je broj smešten u instrukciju jer onda ne mora dvaput da se pristupa memoriji, a s druge strane je mana što time dobijamo mali opseg brojeva
   - broj onda smeštati odmah nakon instrukcije
2) indirektno (indirect) – npr. pročitamo sadržaj na lokaciji 1000 i taj sadržaj upisujemo u memoriju
   - operand je u memoriji
   - pored pribavljanja instrukcije, još jednom se obraća memoriji zbog operanda
3) registarsko (register) – operand je sadržaj registra
4) registarsko indirektno (register indirect) – operand je na adresi na koju ukazuje sadržaj registra
5) bazno (displacement) – zbir sadržaja registara i adrese se koristi da bi se pristupilo odredišnoj adresi (npr. ovo se koristi za nizove u Javi)
6) stek – ne navodi se eksplicitna adresa, već se operand ili stavlja na stek ili vadi iz njega

Aritmetičko-logička jedinica

-Osnovne operacije kod:

- celobrojnih brojeva – komplement, konverzija različitih širina reči, sabiranje, oduzimanje, množenje, deljenje
- realnih brojeva – sabiranje, oduzimanje, množenje, deljenje

-Reprezentacija celobrojnih brojeva-

-Koristimo jedan ili više bajtova

-Problem reprezentacije negativnih brojeva rešava se korišćenjem komplementa dvojke – uradi se inverzija svih bitova i sabere se sa 1
Primer: $0010_2 => 1101_2 + 1 = 1110_2$

-Konverzija različitih širina reči-

-Iz šire u užu reč – gubi se informacija

-Iz uže u širu:

- ako je broj pozitivan, višak bitova je nula
- ako je broj negativan, višak bitova je jedan

-**Sabiranje** – vrši se bit po bit (potpisivanjem). U najgorem slučaju može doći do prekoračenja a to je kada rezultat ne može da stane u zadati broj bitova.

-**Oduzimanje** – svodi se na sabiranje sa potpunim komplementom i to nam daje REZULTAT + PRENOS

- ako je prenos = 1 onda je rezultat korektan
- ako je prenos = 0 onda je rezultat negativan (stvarni rezultat je potpuni complement od rezultata sa negativnim predznakom)
- i ovde može da se dogodi prekoračenje

*Znaćemo da se desio overflow prilikom neke operacije ukoliko prilikom sabiranja dva pozitivna ili dva negativna broja dobijemo broj suprotnog znaka

-**Množenje neoznačenih brojeva** – svodi se na sabiranje parcijalnih umnožaka gde je rezultat suma parcijalnih umnožaka, gde je svaki anredni pomeren za jedan bit u levo.

-**Množenje označenih brojeva** - imamo više varijanti:

- **Varijanta A** - konvertovati oba broja u pozitivne i obaviti množenje neoznačenih brojeva i ako je potrebno, uraditi komplement rezultata
- **Butov algoritam** je zamišljen da množi i pozitivne i negativne brojeve i da uvek daje tačan rezultat. Jedina "mana" mu je da izvršava u onoliko koraka koliko ima bitova sa kojim računamo. (Na kolokvijumu će to biti 4 koraka jer ćemo raditi sa 4bitnim brojevima)
- **Varijanta C –** kod modernih procesora, množenje se završava u nekoliko taktova

-Celobrojno deljenje – bazira se na klasičnom deljenju

- Ovde ne možemo da radimo sa označenim brojevima, već kakve god brojeve da dobijemo, mi moramo raditi sa pozitivnim. Na kraju ćemo onda raditi complement da proverimo rezulat.
- Algoritam:
  - Ulaz: apsolutna vrednost delioca u M i apsolutna vrednost deljenika u (A,Q) (proširen u dva registra (A,Q) dodavanjem nula u A)
  - Izlaz:
    - rezultat u Q (**ukoliko su bitovi znaka isti za deljenika i delioca, rezultat je tačan, u suprotnom uraditi potpuni complement od Q**)
    - ostatak u A (**ukoliko bitovi znaka deljenika i ostatka nisu isti, uraditi potpuni komplement od A**)
- Moderni CPU imaju brži algoritam za deljenje od ovoga, ali i dalje ne tako brz kao što je to slučaj kod množenja jer je gotovo nemoguće toliko optimizovati algoritam da bi se dobile bolje performanse.

<u>-Reprezentacija realnih brojeva u računaru-</u>

1) Fixed point (ograničen) – jedan deo za celobrojni deo, jedan za decimalni
2) Floating point – oblik $\mathbf{m \times b^e}$
   - m je mantisa
   - b je baza
   - e je eksponent

   *u računarima je baza obično 2, a eksponent i mantisa su celobrojne vrednosti
   - oblik **+/- mantisa x 2^{exponent}**
     - znak mantise – 1 bit
     - mantisa je normalizovana (1, xxx), podrazumeva se da je 1, pa se jedinica levo od decimalnog separatora se ne zapisuje, ako smo izdvojili 23 bita za mantisu, zapravo imamo 24 bita (**1 se podrazumeva**)
     - eksponent je u obliku "višak xxx" (excess/biased xxx) – 8 bita

-Gustina floating point brojeva je opseg / preciznost. Ukupan broj različitih brojeva je i dalje 2^{32}. Ako povećamo samo broj bitova za eksponent, povećavamo opseg, ali smanjujemo preciznost. Ako povećamo samo broj bitova za mantisu, povećavamo preciznost, ali ne i opseg. Rešenje: povećati broj bitova i za mantisu i za eksponent – dvostruka preciznost, odnosno 64 bita ukupno za mantisu i eksponent.
-Problemi sa floating point formatom: reprezentacija nule, overlow (prekoračenje eksponenta, prekoračenje mantise), underflow (eksponenta i mantise) – brojevi između dva opsega.

-Postoje standardi za ovo:

- IEEE 754:
  - Jednostruka preciznost (1 + 8 + 32) bita – float
  - Dvostruka preciznost (1 + 11 + 52) bita – double
  - Postoji mogućnost upotrebe dodatnih bitova u mantisi
    - mantisa: >30 za jednostruku preciznost i >62 za dvostruku
    - eksponent: >10 za jednostruku preciznost i >14 za dvostruku
    - koriste se kod računanja međurezultata - smanjuje se greška zaokruživanja, smanjuje se verovatnoća pojave prekoračenja u međurezultatu, čime bi propao konačan rezultat

-**Aritmetičke operacije:**

- **sabiranje/oduzimanje**
  - proveri nule
  - izjednači eksponente - manji eksponent se poveća, a njegova mantisa pomeri u desno zadati broj puta; ovako se manje gubi nego da smo veći eksponent smanjivali i množili njegovu mantisu sa 2
  - saberi/oduzmi mantise
  - normalizuj rezultat - mantisa se pomera u levo (dok MSB ne bude 1), a eksponent smanjuje
  - zaokruži
- **množenje/deljenje**
  - proveri nule
  - dodaj/oduzmi eksponente - oduzmi/dodaj višak (bias)
  - pomnoži/podeli mantise
  - normalizuj
  - zaokruži
  - svi međurezultati bi trebalo da su u većoj preciznosti
- **zaokruživanje** – međurezultati imaju veći broj bitova nego konačan rezultat

-Zadaci na testu-

1) Konverzije decimalnog broja u heksadecimalni ili oktalni
2) Konverzije iz oktalnog/heksadecimalnog u onaj drugi, preko binarnog brojnog sistema
3) Oduzimanje upotrebom potpunog komplementa
4) Množenje označenih celobrojnih brojeva upotrebom Butovog algoritma
5) Deljenje neoznačenih celobrojnih brojeva
6) Izvođenje minimalnog i maksimalnog floating point broja na osnovu zadatog broja bitova za mantisu i eksponent i zadati bias

Ivana Milutinović