Uvod u organizaciju i arhitekturu računara 1

- odgovori na česta pitanja sa ispita -

Mina Milošević
mi17081@alas.matf.bg.ac.rs

2017/2018

1. Navesti karakteristike svake od generacija savremenih elektronskih racunara.

I generacija racunara 1939. – 1958.

- vakuumske cevi kao prekidaci
- U/I uredjaji su bili busene kartice, papirne i magnetne trake
- unutrasnju memoriju su cinile odlozenie linije, magnetne trake i magnetni dobosi
- za programiranje se koristio masinski jezik, pri kraju perioda i assembler
- racunari su bili veliki, proizvodili su mnogo toplote
- 1957. - FORTRAN - prvi visi programski jezik
- 1958. - razvijena i prva globalna racunarska mreza

II generacija racunara 1959. – 1964.

- koriste se tranzistori kao prekidaci (od germanijuma i silicijuma)
- unutrasnja memorija se pravi od magnetnih jezgara
- kao spoljasnja memorija koriste se magnetni diskovi
- U/I uredjaji su bili busene kartice, papirne i magnetne trake
- javljaju se prvi operativni sistemi
- razvijaju se visi programski jezici (Fortran, Lisp, Algol-60, Cobol...)
- projekat Stretch

III generacija racunara 1965. – 1971.

- primena integrisanih kola umesto pojedinacnih tranzistora
- pojavljuju se SSI cipovi
- novi programski jezici razlicitih karakteristika
- unutrasnja memorija se pravi od magnetnih jezgara
- kao spoljasnja memorija koriste se magnetni diskovi
- U/I uredjaji su bili busene kartice, papirne i magnetne trake, ekrani i tastature terminala
- operativni sistemi se dalje razvijaju
- konstruisu se prva unapred planirana familija racunara - IBM S/360 i prvi miniracunar - PDP/8
- na kraju ovog perioda pojavljuje se disketa velicine 8 inca

IV generacija racunara 1972. – danas

- dalja minijaturizacija tranzistora i integrisanih kola
- unutrasnja memorija je napravljena u poluprovodnickoj tehnologiji
- pojavljuje se i prvi mikroprocesor (Intel 4004)
- kao spoljasnja memorija koriste se magnetni diskovi
- dolazi do daljeg razvoja softvera i operativnih sistema
- razvijaju se komunikacija i racunarskie mreze
- povecava se koriscenje super racunara

2. Detaljno opisati kontinualna racunska sredstva.

Komponente se medjusobno povezuju na nacin analogan nekom realnom sistemu. U/I podaci su predstavljeni preko fizickih velicina.

1. matematicke velicine se predstavljaju onom tacnoscu koja odgovara mogucnosti preciznog merenja odgovarajuce fizicke velicine
2. tacnost dobijenog rezultata zavisi od preciznosti izrade racunskog sredstva
3. nisu programabilni
4. slozenost matematickog modela ne utice na brzinu dobijanja rezultata

3. Detaljno opisati diskretna racunska sredstva.

Obavljaju operacije iskljucivo sa diskretnim podacima. Sve vrednosti se predstavljaju u obliku brojeva koji se zapisuju pomocu pojedinacnih cifara.

1. svaka cifra se registruje u odvojenom objektu kao jedno od njegovih diskretnih stanja. Diskretna stanja moraju da budu stabilna i da se medjusobno razlikuju
2. tacnost rezultata ne zavisi od preciznosti izrade racunskog sredstva
3. programabilni su
4. brzina izracunavanja zavisi od slozenosti problema koji se resava

4. Navesti karakteristike svakog perioda razvoja informacionih tehnologija.

Premehanicki period(3000g.p.n.e. - 1450g.n.e.)

• Pojava prvog alfabeta i pisma (oko 3000 g. p.n.e. u Mesopotamiji)
• Formiranje ne-slikovnih alfabeta (Fenicani, Grci, Rimljani,...)
• Koriscenje papira za pisanje (Kina, oko 100 g. n.e.)
• Pojava prvih biblioteka
• Razvoj brojcanih sistema (nepozicionih i pozicionih)
• Pojava i koriscenje abakusa kao sredstva za racunanje

Mehanicki period(1450g. - 1840g.)

• 1450. - pojava stamparske prese sa pokretnim slogovima od metala (Gutenberg)
• 1614. - uvodjenje logaritama i decimalne tacke u zapisima brojeva (Dzon Neper)
• 1622. - pojava klizajuceg lenjira - sibera (Viljem Outred)
• 1623. - konstrukcija masine za sabiranje i oduzimanje (Sikard)
• 1642. - Paskalina – mogla je da sabira i oduzima osmocifrene brojeve
• 1673. - pojava masine koja je mogla da vrsi sabiranje, oduzimanje, mnozenje i deljenje (Lajbnic)
• 1801. - prva upotreba busene kartice (Zakardov automatski razboj)
• Masine Carlsa Bebidza (diferencijska, nacrt 1822, prototip 1832, nacrt analiticke masine 1833. godine)

Elektromehanicki period (1840g. - 1939g.)

• Razvoj telekomunikacija (telegraf 1830., telefon 1876., radio 1984.)
• 1854. - pojava Bulove algebre
• 1884. - konstrukcija automatske masine za tabuliranje zasnovane na busenim karticama (Herman Holerit)
• Razvoj razlicitih elektromehanickih kalkulatora u prvoj polovini 20. veka
• Pojava elektromehanickih racunara specijalne nemene
• Obrada podataka u udaljenom okruzenju
• Konstrukcija MARK I elektromehanickog racunara (zavrsen 1944. godine)

Elektronski period (1939g. - danas)

• 1936. – Tjuringova masina – uproscena verzija savremenog racunara
• 1941. - ABC kalkulator za resavanje sistema jednacina
• 1943. - Kolos (Tjuring) – dekriptovanje nemackih sifrovanih poruka
• 1946. - ENIAC – racunar zasnovan na dekadnom sistemu (Ekert i Musli)
• 1945. - EDVAC (nacrt) – mogucnost cuvanja programa u memoriji zajedno sa podacima i kasnije izvrsavanje. Konstruisan je 1951.
• 1949. - EDSAC – konstrusian po fon Nojmanovoj arhitekturi
• 1949. - BINAC – prvi racunar sa dualnim procesorom
• 1950. - Vihor – racunar namenjen radu u realnom vremenu

5. Primeri analognih i digitalnih racunarskih sredstava u svakom periodu razvoja informacionih tehnologija.

Kontinualna (analogna):
- Antikitera (premehanicki)
- Siber (mehanicki)
- Diferencijalni analizator, Rokfelerov dif. Analizator (elektromehanicki)
- Elektronski analogni racunar (elektronski)

Diskretna (digitalna):
- Abakus (premehanicki)
- Paskalina, Lajbnicova masina, aritmometar...(mehanicki)
- Holeritov tabulator (elektromehanicki)
- EDVAC, ENIAC, EDSAC, MESM... (elektronski)

6. Opisite specificne karakteristike predstavnika racunara II generacije.

IBM 7094 - za prenos podataka je koristio kanale na koji su bili povezani U/I uredjaji. Kanal je nezavisan U/I modul koji poseduje sopstveni procesor sa posebnim skupom instrukcija. Instrukcije se cuvaju u memoriji i izvrsava ih specijalizovani procesor koji se nalazi na kanalima. Posedovao je i multipleksor u koga se slivaju sve veze iz kanala, memorije i CPU-a

BESM-6 - jednoadresni sa unutrasnjom memorijom sastavljenom od magnetnih jezgara. Procesor je bio napravljen u tranzistorskoj tehnologiji. Racunar je imao 15 indeks registara i kes. Maksimalna velicina programa je bila oko 192kb.

IBM 7030 Stretch - uvodi pojmove bajt i sistemska arhitektura. Zasnovan je na tranzistorskoj tehnologiji. Bio je prva masina kod koje je ukljuceno deljenje procesa izvrsavanja na fazu dohvatanja i dekodiranja, i fazu izvrsavanja. Koriste se magnetna jezgra za unutrasnju memoriju. Adresiranje memorije se obavljalo u prirastajima koji su odgovarali stepenima broja 2. Umesto dobosa, za spoljasnju memoriju se koriste magnetni diskovi

7. Opisite specificne karakteristike predstavnika racunara III generacije.

IBM System/360 - prva unapred planirana familija racunara:

1. svi racunari familije su imali identican ili slican skup instrukcija
2. svi racunari u familiji su imali identican ili slican operativni sistem
3. svaki od jacih modela je u odnosu na slabije za vecu cenu nudio i vecu brzinu; veci broj kanala na koje je moglo da se prikljuci veci broj U/I jedinica; vecu kolicinu unutrasnje memorije.

PDP-8 - prvi miniracunar koji se pojavio na trzistu; uveden koncept *magistrale*

8. Definisati osnovne pojmove vezane za azbuku.

Azbuka - konacan neprazan skup proizvoljnih simbola. Elementi skupa V se nazivaju slova ili simboli, a konacne niske slova se nazivaju reci nad V. Prazna rec je rec koja ne sadrzi slova. Proizvoljan skup reci nad V* se naziva jezik. Broj slova u reci P se naziva duzina reci i oznacava se sa |P|. Duzina prazne reci je po definiciji 0.

9. Definisati azbuku, jezik, pojmove kod, osnova koda, kodna rec, funkcija kodiranja, funkcija dekodiranja.

- *Azbuka* - konacan neprazan skup proizvoljnih simbola
- *Jezik* – proizvoljan skup reci
- *Kod* jezika L1 u azbuci V2 je skup vrednosti
- *Osnova koda* – broj simbola u azbuci V2
- *Kodna rec* – rec qi iz jezika L2
- *Funkcija kodiranja* - svaka funkcija f definisana sa f : L1-->L2.
- *Funkcija dekodiranja* - funkcija g : L2-->L1, $g = f^{-1}$, ako f^{-1} postoji.

10. Kada kazemo da je kod: jednoznacan, tezinski, komplementaran ili ciklican?

• Kod je *jednoznacan* akko je funkcija *f* 1-1. U suprotnom je kod *viseznacan*.
• Kod je *potpun* kada obuhvata sve reci odredjene duzine u jeziku *L*2.
• Kod je *ravnomeran* ako je duzina svih kodnih reci u jeziku *L*2 ista. U suprotnom, kod je *neravnomeran*. Duzina reci ravnomernog koda se naziva *mesnost* koda.
• Kod je *ciklican* ako se od prethodnog razlikuje za 1 bit.
• Kod je *komplementaran* ako su kodovi dekadnih cifara a i b, za koje vazi uslov a+b=9, komplementarni
• Kod je *tezinski* ako je i-toj poziciji kodne reci pridruzen broj p_i, tako da za dekadnu cifru q i njenu kodnu rec $y_3y_2y_1y_0$ vazi: $q=p_3y_3+p_2y_2+p_1y_1+p_0y_0$.

11. Kako se zapisuju znakovni podaci u računarskom sistemu? Nabrojati kodove koji se najčešće koriste za zapis znakovnih podataka i navesti njihove karakteristike.

Znakovni podaci se u računaru zapisuju pomoću binarnih kodova. Najpoznatiji, odnosno najčešće korišćeni kodovi su:
• ASCII. Ovaj kod je 7-bitni i može da predstavi 128 karaktera. Iako se za kodiranje koristi niska od 7 binarnih cifara, karakteri kodirani u ASCII kodu se skoro uvek čuvaju i prenose u grupi od 8 bitova, pri čemu se osmi bit koristi za kontrolu parnosti.
• EBCDIC. U ovom kodu se može predstaviti 256 različitih karaktera pričemu se svaki karakter predstavlja jednoznačnom niskom od 8 binarnih cifara.
• ISO-8. Ovaj kod je 8-bitni pri čemu se prvih 127 pozicija poklapa sa ASCII kodom dok su preostale pozicije popunjenje različitim kontrolnim i grafičkim karakterima. ISO-8 kontrolni karakteri su preuzeti iz ISO 6429, dok su grafički karakteri preuzeti iz ISO 8859-1 koda.
• IBM-PC. Ovaj kod je takože 8-bitni kod i u prvih 127 pozicija se poklapa sa ISO-8 kodom, dok se na ostalim mestima nalaze različiti kontrolni i grafički karakteri.
• UNICODE. U pitanju je 16-bitni kod. UNICODE predstavlja standard za univerzalno kodiranje karaktera i omogućuje razmenu, obradu i prikaz teksta pisanog u bilo kom jeziku savremenog sveta, kao i u velikom broju klasičnih jezika. UNICODE standard je kompatibilan sa ISO/IEC 10646 standardom.

12. Definisati osnovne pojmove vezane za brojcane sisteme.

• *Nepozicioni* – znak koji oznacava cifru ima istu vrednost bez obzira na poziciju u zapisu broja
• *Pozicioni* – vrednost znaka koji predstavlja cifru zavisi i od izgleda znaka i od pozicije cifre u zapisu broja

13. Navesti osnovne karakteristike pozicionih brojcanih sistema. Navesti moguce tipove osnova pozicionih brojcanih sistema.

Osnovne karakteristike:
• *Osnova brojcanog sistema* – broj razlicitih cifara pozicionog brojcanog sistema
• *Pozicija cifre* - mesto cifre u zapisu broja
• *Duzina broja* - broj cifara u zapisu broja
• *Tezina cifre* u zapisu broja zavisi od pozicije na kojoj se cifra nalazi

Moguci tipovi osnova:
• dekadni sistem: N=10, S={0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
• binarni sistem: N=2, S={0, 1}
• oktalni sistem: N=8, S={0, 1, 2, 3, 4, 5, 6, 7}
• heksadekadni sistem: N=16, S={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}
• troicni sistem: N=3, S={0, 1, 2}
• balansirani troicni sistem: N=3, S={−1, 0, 1}
• sistem sa negativnom osnovom: -N, S=[0, N − 1]

14. Zapis oznacenih brojeva:

Oznaceni brojevi su brojevi ciji zapis ukljucuje i cifru znaka. Cifra na mestu najvece tezine (krajnje leva cifra) oznacava znak broja. Postoji vise nacina za zapis oznacenih brojeva, a svima je zajednicko da se znak broja predstavlja najmanjom cifrom sistema (nulom) ukoliko je broj pozitivan, odnosno, najvecom cifrom ukoliko je negativan. Zapisi se medjusobno razlikuju po nacinu predstavljanja apsolutne vrednosti broja, to su znak i apsolutna vrednost, nepotpuni komplement, potpuni komplement i zapis sa uvecanjem k.

15. Znak i apsolutna vrednost:

Cifra na mestu najvece tezine (krajnje leva cifra) oznacava znak broja. Znak broja se predstavlja najmanjom cifrom sistema (nulom) ukoliko je broj pozitivan, odnosno, najvecom cifrom ukoliko je negativan.

Sabiranje:

• potrebno je odrediti znak zbira i apsolutnu vrednost zbira.

1) ako su zadati brojevi istog znaka, tog znaka je i rezultat. Apsolutna vrednost zbira je zbir apsolutnih vrednosti sabiraka.

2) ako su zadati brojevi razlicitog znaka, znak rezultata odgovara sabirku sa vecom apsolutnom vrednoscu. Apsolutna vrednost zbira se dobija kada se od vece apsolutne vrednosti oduzme manja apsolutna vrednost.

Oduzimanje:

• kako je $A - B = A + (-B)$ prvo se vrsi promena znaka broja B, a zatim se postupa u skladu sa pravilima za sabiranje brojeva zapisanih u znaku i apsolutnoj vrednosti.

Prekoracenje:

• javlja se ako je za zapis apsolutne vrednosti zbira potreban veci broj cifara nego za zapis apsolutnih vrednosti sabiraka.

16. Nepotpuni komplement:

Dobija se komplementiranjem cifara zapisa znak i apsolutna vrednost.

Sabiranje:

• prvo se sabiraju sve cifre zapisa, ukljucujuci i znak. Ako se javi prenos sa pozicije najvece tezine, dodaje se na poziciju najmanje tezine

Oduzimanje:

• svodi se na sabiranje komplementiranjem svih cifara umanjioca

Prekoracenje:

• moze doci do prekoracenja samo ako se sabiraju brojevi istog znaka i to ako se dobije rezultat razlicitog znaka od sabiraka ili broj koji nije cifra za znak

17. Potpuni komplement:

Dobija se dodavanjem +1 na poziciju najmanje tezine zapisa broja u nepotpunom komplementu.

Sabiranje:

• prvo se sabiraju sve cifre zapisa, ukljucujuci i znak. Ako se javi prenos sa pozicije najvece tezine, on se ignorise

Oduzimanje:

• moze da se oduzima direktno ili se svodi na sabiranje komplementiranjem svih cifara umanjioca i dodavanjem +1

Prekoracenje:

• moze doci do prekoracenja samo ako se sabiraju brojevi istog znaka i to ako se dobije rezultat razlicitog znaka od sabiraka ili broj koji nije cifra za znak

18. Visak k:

Za brojeve zapisane u kodu visak k zbir i razlika se racunaju prema pravilima koja vaze za brojeve zapisane u potpunom komplementu, a potom se dobijena vrednost azurira oduzimanjem tj. dodavanjem konstante k:

$$((x+k)+(y+k) = x+y+2\cdot k)$$
$$((x + k)-(y + k) = x-y)$$

19. Sta je komplementaciona konstanta?

Komplementaciona konstanta se odnosi samo na negativne brojeve. Najefikasnije izvršavaje operacija se dobija za sledece vrednosti K:

1) $K = N^n - 1$ komplement umanjene osnove ili N −1-vi komplement.

$$[-N^{n-1}+1, +N^{n-1}-1]$$

2) $K = N^n$ komplement u odnosu na osnovu sistema ili N-ti komplemen..

$$[-N\ n-1\ , +N\ n-1\ -\ 1]$$

20. Mnozenje neoznacenih brojeva:

Za realizaciju algoritma za mnozenje neoznacenih brojeva potrebna su nam tri registra A, P i M duzine 8 bita i jedan jednobitni registar C.

Inicijalizacija:

1. Mnozenik zapisujemo kao broj duzine 8 bita u registru M
2. Mnozilac zapisujemo kao broj duzine 8 bita u registru P. Sa P_0 cemo obeleziti najnizi bit ovog registra
3. Registar A inicijalizujemo na 0
4. Registar C inicijalizujemo na 0

Koraci algoritma:

1. Ako je $P_0 = 1$ na sadrzaj registra A dodajemo sadrzaj registra M i eventualni prenos prilikom sabiranja upisujemo u registar C. Ako je $P_0 = 0$ nista ne preduzimamo.
2. Sadrzaj registara CAP posmatran kao jednu rec logicki pomeramo za jednu poziciju udesno.
3. Prethodna dva koraka ponavljamo osam puta.

Rezultat ocitavamo iz registra AP kao broj duzine 16 bita.

21. Mnozenje oznacenih brojeva:

Za realizaciju algoritma za mnozenje oznacenih brojeva potrebna su nam tri registra A, P i M duzine 8 bita i jedan jednobitni registar P_{-1}.

Inicijalizacija:

1. Mnozenik zapisujemo u registru M kao broj u potpunom komplementu duzine 8 bita
2. Mnozilac zapisujemo u registru P kao broj u potpunom komplementu duzine 8 bita. Sa P_0 cemo ziti najnizi bit ovog registra
3. Registar A inicijalizujemo na 0
4. Registar P_{-1} inicijalizujemo na 0

Koraci algoritma:

1. Posmatramo kombinaciju bitova P_0P_{-1}
   - $P_0P_{-1} = 00$ nista ne preduzimamo
   - $P_0P_{-1} = 11$ nista ne preduzimamo
   - $P_0P_{-1} = 01$ racunamo zbir A + M i upisujemo ga u registar A
   - $P_0P_{-1} = 10$ racunamo razliku A − M i upisujemo je u registar A
2. Sadrzaj registara APP_{-1} posmatran kao jednu rec aritmeticki pomeramo za jednu poziciju desno.
3. Prethodna dva koraka ponavljamo osam puta.

Rezultat ocitavamo iz registra AP kao broj duzine 16 bita zapisan u potpunom komplementu.

22. Deljenje neoznacenih brojeva:

Za realizaciju algoritma za deljenje neoznacenih brojeva potrebna su nam tri registra A, P i M duzine 8 bita:

Inicijalizacija:

1. Deljenik zapisujemo kao broj duzine 8 bita u registru P. Sa P_0 cemo obeleziti najnizi bit ovog registra
2. Delilac zapisujemo kao broj duzine 8 bita u registru M
3. Registar A inicijalizujemo na 0

Koraci algoritma:

1. Sadrzaj registra AP kao jednu rec pomeramo za jednu poziciju ulevo
2. Uporedjujemo sadrzaj registra A sa sadrzajem registra M:
   - Ako je A-M=A, A>=0 , racunamo razliku A−M i upisujemo je u registar A, $P_0 \rightarrow 1$
   - Ako je A-M=A, A<0, u P_0 upisujemo 0
3. Prethodna dva koraka ponavljamo osam puta.

Rezultat (kolicnik) ocitavamo iz registra P, dok ostatak ocitavamo iz registra A.

23. Deljenje oznacenih brojeva:

Za realizaciju algoritma za deljenje oznacenih brojeva potrebna su nam tri registra A, P i M duzine 8 bita. Sa P_0 cemo obeleziti najnizi bit registra P.

Inicijalizacija:

1. Deljenik zapisujemo kao broj duzine 16 bita u potpunom komplementu u registru AP
2. Delilac zapisujemo kao broj duzine 8 bita u potpunom komplementu u registru M

Koraci algoritma:

1. Sadrzaj registra AP kao jednu rec pomeramo za jednu poziciju ulevo
2. Uporedjujemo prvi bit registra A sa prvim bitom registra M:
   - Ako su bitovi jednaki (vrednosti u registrima A i M su istog znaka) racunamo razliku A − M
   - Ako su bitovi razliciti (vrednosti u registrima A i M su razlicitog znaka) racunamo zbir A + M

Obelezimo ovako dobijen rezultat sa R. Ukoliko je prvi bit rezultata R jednak prvom bitu registra A , za operaciju kazemo da je uspesna i rezultat R upisujemo u registar A , 1 upisujemo u P_0.
Ukoliko prvi bit registra R nije jednak prvom bitu registra A, za operaciju kazemo da je neuspesna i rezultat R zanemarujemo, 0 upisujemo u P_0

3. Prethodna dva koraka ponavljamo osam puta.

Rezultat (kolicnik) ocitavamo iz registra P kao broj u potpunom komplementu. Ukoliko su jos deljenik i delilac razlicitih znakova, rezultat treba uzeti sa predznakom minus. Ostatak ocitavamo iz registra A kao broj zapisan u potpunom komplementu.

24. Koja je osnovna prednost modifikovanog Butovog algoritma za mnozenje celih brojeva u odnosu na obican Butov algoritam za mnozenje celih brojeva?

Osnovna prednost modifikovanog Butovog algoritma je manji ili jednak broj operacija (sabiranja ili oduzimanja) u odnosu na obican Butov algoritam. Kod modifikovanog algoritma operacija sbiranja ili oduzimanja se izvrsava najvise [n/2]+1 puta (za brojeve zapisane u polju velicine n bita), dok kod obicnog Butovog algoritma taj broj moze da bude i veci, u zavisnosti od broja alterniranja 0 i 1 u zapisu mnozioca.

25. Hartmanova metoda:

Hartmanova metoda je nacin prevodjenja brojeva iz dekadnog u heksadekadni sistem i obratno. Sastoji se iz niza sabiranja i mnozenja korektivnom cifrom. Broj se konvertuje sleva udesno; sve operacije se obavljaju u brojcanom sistemu u kojem se vrsi prevodjenje.

26. Modifikovan Butov algoritam:

Minimalizuje broj operacija prilikom mnozenja oznacenih brojeva.
Zapis brojeva - Mnozenik se zapisuje kao broj duzine 16 bita u potpunom komplementu. Mnozilac se prvo zapisuje kao broj duzine 8 bita u potpunom komplementu, a zatim se svodi na modifikovani oblik.
Svodjenje mnozioca na modifikovani oblik - Ideja je uociti uzastopne nizove jedinica u mnoˇzocu Pocetak serije jedinica treba obeleziti sa −1, a kraj serije (prvu pojavu nule) sa +1. Na svim ostalim pozicijama treba upisati nule.
Izdvajanje parova i racunanje vrednosti – parove izdvajamo s desna na levo pocevsi od pozicije najmanje tezine. Svakom paru treba da bude pridruzena odgovarajuca vrednost (-2, -1, 0, 1, 2)
Pomeranje mnozenika i racunanje medjuproizvoda - Za svaku vrednost broja k prvo treba pomeriti mnozenik za 2k bita ulevo, a zatim tako dobijeni binaran broj treba pomnoziti vrednoscu V_k para. Pritom vazi:

1. $V_k = 2$, mnozenje se svodi na pomeranje mnozenika za jednu poziciju ulevo.
2. $V_k = 1$, mnozenik se ne menja
3. $V_k = 0$, rezultat je 0
4. $V_k = -1$, mnozenje se svodi na komplementiranje i dodavanje jedinice na poziciju najmanje tezine
5. $V_k = -2$, mnozenje se svodi na komplementiranje i dodavanje jedinice na poziciju najmanje tezine, a zatim na pomeranje tako dobijenog broja za jednu poziciju ulevo.

Konacan proizvod - Konacan proizvod se dobija sabiranjem svih medjuproizvoda. Sabiranje se izvodi po pravilima koja vaze za brojeve u potpunom komplementu pa se eventualni prenosi zanemaruju.

27. Navesti i opisati karakteristike binarnih kodova dekadnih cifara.

Zbog toga sto je nekada nemoguce tacno predstaviti dekadne razlomljene brojeve u binarnom brojevnom sistemu, doslo se na ideju da se svaka dekadna cifra u zapisu broja posebno kodira pomocu binarnog zapisa.
Za kodiranje dekadnih cifara binarnim brojevima potrebne su bar cetiri binarne cifre.
Da bi binarni kod dekadnih cifara bio upotrebljiv mora da bude ispunjen uslov jednoznacnosti, odnosno da sve binarne reci koje ulaze u kod moraju da budu medjusobno razlicite. Pored ovoga, poželjne osobine koje omogucuju jednostavije izvodjenje operacija su:

- Najvecoj dekadnoj cifri (9) treba pridružiti rec koja ima najvecu vrednost (posmatrana kao binarni broj).
- Parnim i neparnim dekadnim ciframa treba da odgovaraju parni odnosno neparni binarni br.
- Kod treba da bude komplementaran i tezinski (pr. 8421, 2421, 5421, 753-6, višak 3)

28. Opisati karakteristike pakovanog i nepakovanog zapisa oznacenih BCD brojeva.

Binarno kodirani dekadni brojevi se mogu koristiti za zapis brojeva i zapis znakova.

- Kod ASCII koda u nepakovanom zapisu se u polubajt vece tezine upisuje cifra 5 koja oznacava da je u bajtu zapisana cifra, a u polubajtu manje tezine se zapisuje BCD zapis cifre u odgovarajucem kodu
- Binarno kodirani dekadni brojevi se uglavnom zapisuju u obliku znak i apsolutna vrednost. Znak se upisuje u polubajtu vece tezine poslednjeg bajta I to kao binarno zapisana heksadekadna cifra A za + ili B za -

Ukoliko se koriste samo brojcani podaci, dolazi do nepotrebnog trosenja prostora i zato se uvodi pakovani zapis.

- u svakom polubajtu (osim u poslednjem) se cuva neka cifra broja
- u poslednjem polubajtu se cuva znak broja
- u zapisu broja sa parnim brojem cifara se u vodeci polubajt upisuje 0

29. Kod 8421

U kodu 8421 svakoj dekadnoj cifri pridruzujemo odgovarajuci binarni kod.

Sabiranje:

1) sabiraju se cifre na odgovarajucim pozicijama i beleze se dobijeni medjuzbirovi i prenosi
2) na osnovu dobijenih vrednosti se vrsi korekcija kako bi svaka od cifara medjuzbira predstavljala korektnu cifru u kodu 8421

Oduzimanje - vrsi se prema slicnom algoritmu za sabiranje brojeva:

1) oduzimaju se cifre na odgovarajucim pozicijama i beleze se dobijene medjurazlike i pozajmice
2) vrsi se korekcija

Ako je medjuzbir/ralzlika > 9 ili je prenos 1, dodajemo +6 (+0110) kao korekciju.
Prekoracenja ce biti ako je neki od krajnje levih prenosa 1.

30. Kod visak 3

U kodu visak 3 svaka od cifara se prikazuje kao cetvorobitna BCD vrednost uvecana za 3.

Sabiranje:

1) sabiraju se cifre na odgovarajucim pozicijama, belezi dobijeni medjuzbir i prenos
2) na osnovu dobijenih vrednosti se vrsi korekcija kako bi svaka od cifara medjuzbira predstavljala korektnu cifru u kodu visak 3

Posto je kod visak 3 komplementaran, odredjivanje potpunog komplementa broja zapisanog u ovom kodu se vrsi komplementiranjem datog zapisa.

Oduzimanje:

- kao sabiranje u potpunom komplementu umanjenika sa umanjiocem kome je promenjen znak

Kada ima prenosa, medjurezultatu dodajemo +3 (+0011), u suprotnom -3 (+1101).
Do prekoracenja dolazi ako je neki od krajnje levih prenosa 1 (kod neoznacenih brojeva), ili ako sabiranjem dva broja istog znaka dobijemo rezultat suprotnog znaka.

31.Sta je Grejov kod i koje su njegove karakteristike i konverzije?

Grejov kod duzine $n \geq 0$ je funkcija $G(n, i)$ koja vrsi "1-1" preslikavanje celog broja $i \in [0, 2^n - 1]$ pri cemu vazi da se binarne reprezentacije $G(n, i)$ i $G(n, i+1)$ razlikuju tacno na 1 mestu.

Karakteristike:

- funkcija koja vrsi preslikavanje nije jedinstvena tako da postoji vise Grejovih kodova duzine n
- Jedna od najcesce koriscenih funkcija se moze definisati na sledeci nacin:

$$G(n, i) = {}_n i \oplus {}_n[i/2] \ , \ n>0, i \in [0, 2^n-1]$$

- Ista funkcija $G(n, i)$ se moze definisati i rekurentno:

$G(n+1, i) = 0G(n, i)$ $\quad n>0;\ i \in [0, \dots, 2^n-1]$
$G(n+1, i) = 1G(n, 2^{n+1}-1-i)$ $\quad n>0;\ i \in [2^n, \dots, 2^{n+1}-1]$
$G(1,0) = 0$
$G(1,1) = 1$

1. kodiranje (iz binarnog broja u odgovarajuci Grejov kod) - cifra najvece tezine se kopira, a ostale cifre se odredjuju po formuli:

$$g_i = a_i \oplus a_{i+1},\ i=n-2,\dots,0$$

2. dekodiranje - cifra najvece tezine se kopira, a ostale cifre se odredjuju po formuli:

$$a_i = g_i \oplus a_{i+1},\ i=n-2,\dots,0$$

32. Karakteristike Cen-Ho i DPD kodiranja?

Posto je za kodiranje jedne dekadne cifre potrebno 4 bita, za 3 cifre je potrebno 12 bitova, pa u vecini slucajeva dolazi do gubitka prostora. IBM naucnici su zakljucili da je za kodiranje tripleta dekadnih cifara dovoljno 10 bita. Ideja ova dva kodiranja je u sustini ista. Cifre su podelili na male [0, 7] i velike [8, 9]. Razlikuju sledece slucajeve:

- *sve cifre su male* - potrebno 9 bitova za cifre i 1 za kombinaciju
- *jedna cifra je velika* - 7 bitova za cifre i 3 bita za kombinaciju
- *dve cifre su velike* - 5 bitova za cifre i 5 za kombinaciju
- *dve cifre su velike* - 3 bita za cifre i 7 za kombinaciju

Kodiranje je proces u kome se 12-bitna kombinacija (abcd)(efgh)(ijkl) kodira u 10-bitnu kombinaciju(p)(qrs)(tuv)(wxy) kod Chen-Ho, odnosno u (pqr)(stu)(v)(wxy) kod DPD kodiranja.

33. Koje su prednosti Cen-Ho u odnosu na BCD? Koje su prednosti DPD u odnosu na Cen-Ho?

Osnovna prednost Cen-Ho u odnosu na BCD je u ustedi memorije i u njemu se za kodiranje tri dekadne cifre koristi 10 bita sa samo 0.34% gubitka prostora, sto daje oko 20% efikasnije kodiranje u odnosu na obican BCD kod.

DPD kodiranje je po ideji slicno Cen-Ho kodiranju ali koristi drugacije preuredjenje bitova sto donosi sledece prednosti:

1. pri kodiranju jedne ili dve cifre stedi se prostor, dok kod Chen-Ho kodiranja uvek za 3 cifre treba 10 bita
2. pri kodiranju jedne ili dve cifre vrsi se desno poravnanje tako da pri prosirivanju zapisa nije potrebno ponovo kodirati broj, dok kod Chen-Ho kodiranja je neophodno ponovno kodiranje
3. brojevi od [0, 79] su desno poravnati kao i u BCD 8241 zapisu, sto olaksava konverziju, dok su kod Chen-Ho kodiranja poravnani samo u intervalu [0, 7]

34. Sta su realni brojevi u fiksnom (nepokretnom) zarezu, kako se zapisuju i gde se upotrebljavaju?

U pitanju su realni brojevi koji se, bez obzira na velicinu broja, zapisuju uvek sa istim brojem cifara. Pri tome je i broj cifara u razlomljenom delu uvek konstantan (implicitno, odavde sledi da je konstanan i broj cifara pomocu koga se zapisuje celobrojni deo broja). Celobrojni i razlomljeni deo mogu da se zapisuju kao dekadni, heksadekadni ili binarni brojevi.

35. Sta su realni brojevi u pokretnom zarezu, kako se zapisuju i gde se upotrebljavaju?

Realni brojevi u pokretnom zarezu se koriste za zapis jako velikih ili jako malih brojeva kao i kada je potrebno predstaviti realne brojeve sa velikom preciznoscu.

Predstavljaju se pomocu osnove β (koja je uvek parna) i preciznosti p (broj cifara znacajnog dela).

Opsti slucaj: $\pm d_0, d_{-1}d_{-2}...d_{-(p-1)}\ \beta^e$

- $d_0, d_{-1}d_{-2}...d_{-(p-1)}$ – znacajni deo (mantisa, frakcija)
- e – eksponent

Niz cifrara u razlomljenom broju se naziva *frakcija*. Znacajni deo se zapisuje u sistemu sa osnovom β. Zapis broja za koji vazi da je $d_0 \neq 0$ se naziva *normalizovan*.

Zapis: znak broja + frakcija + eksponent

Eksponent se zapisuje kao binarni ceo broj sa uvecanjem. Velicina uvecanja i broj cifara u frakciji zavise kako od osnove koja se koristi za zapis frakcije tako i od broja bitova koji su na raspolaganju za zapis brojeva.

36. Definisati pojmove ulp, relativna greska i masinsko ε?

Ulp – najmanja vrednost za oju s mogu razlikovati dva broja u pokretnom zarezu zapisa u odredjenoj osnovi i sa odredjenom preciznoscu. Ako realni broj $d_0, d_{-1}d_{-2}...d_{-(p-1)}*\beta^e$ predstavlja broj z, tada je greska u zapisu $|d_0, d_{-1}d_{-2}...d_{-(p-1)} - (z/\beta^e)| * \beta^{p-1}$ ulp-a.
Relativna greska - apsolutna vrednost razlike realnog broja i njegove reprezentacije podeljena sa apsolutnom vrednoscu realnog broja. Uvek se zapisuje u terminima *masinskog ε*= $\beta^{1-p}/2$.
Ulp se koristi za odredjivanje greske zaokruzivanja brojeva, a relativna greska za analiziranje gresaka izracunavanja prema razlicitim formulama.

37. Sta je efektivna sirina?

Nacin za opis rezolucije formata je bliskost pojedinacne vrednosti u odnosu na relativno rastojanje sto moze da se predstavi vrednoscu ulp-a/vrednost broja. Relativno rastojanje moze da se konvertuje u oblik koji se naziva *efektivna sirina* koji je vrlo je slican sa preciznoscu. Za neku pojedinacnu normalnu vrednost x u pokretnom zarezu efektivna sirina: $W_\beta=\log_\beta(x/ulp)$.

38. Sta su cifre cuvari i kada se koriste?

Da bi rezultat aritmetičke operacije bio izračunat sa preciznoscu od *p* cifara, argumenti operacije treba da se zapisuju i izračunavaju sa *p+2* cifre. Te dodatne cifre se nazivaju *cifre čuvari*.
Zbog ogranicenja broja bitova koji se koriste za zapis realnih brojeva, moze se dogoditi da rezultate aritmetickih operacija nije moguce tacno zapisati. Ako se argumenti zapisu sa jednom ili vise dodatnih cifara (cifre cuvari), nad takvim argumentima izvedu operacije i tada zaokruzi dobijeni rezultat na prvobitnu tacnost, eliminisace se negativan efekat prethodnog zaokruzivanja brojeva i njihovog predstavljanja u trazenoj tacnosti.
Primer:
x=100.11, y=99.99, osnova je 10, preciznost je 4:
Ako je x=100.11, y=99.99, tada je $xa = 1.001*10^2$, $ya = 9.999*10^1$. Da bi se brojevi sabrali, moraju da se dovedu na isti eksponent. Ako se ne koriste cifre cuvari, razlika je:
$1.001*10^2 - 0.999*10^2 = 0.002*10^2 = 0.2$. Ako se pri zapisu koristi jedna dodatna cifra cuvar tada se brojevi zapisuju sa tacnoscu od 5 cifara, pa je razlika:
$1.0011* 10^2 - 0.9999 *10^2 = 0.0012 * 10^2 = 0.12$, sto je i tacna vrednost razlike x − y.

39. IEEE 754 opis standarda.

Zbog lose prenosivosti numerickih programa, lose kompatibilnosti na drugim racunarima, 1985. godine uveden je IEEE 754 standard za zapis realnih brojeva u pokretnom zarezu. On je propisivao neka pravila, algoritme za izvodjenje aritmetickih operacija, kao i zaokruzivanja tih brojeva. Zbog nemogucnosti predstavljanja svih dekadnih realnih brojeva pomocu binarne osnove, kao i zbog blizine dekadne osnove ljudima doslo se na ideju da se u standard uvede i dekadna osnova.
IEEE 754 standard propisuje:

- β=2 i β=10 kao osnove za zapis brojeva u pokretnom zarezu
- osnovne formate za zapis podataka u binarnoj i dekadnoj osnovi
- formate za razmenu podataka
- prosirene i prosirivane formate za zapis podataka
- nacin izvodjenja aritmeticih operacija
- nacin konverzije brojeva
- nacin konverzije formata brojeva i njihove spoljasnje reprezentacije
- vrste izuzetaka koji se javljaju pri radu sa brojevima i nacine njihove obrade

Osnovni tipovi zapisa:

1. kao uredjena trojka: $(-1)^{znak} X\ \beta^{eksponent} X$ znacajni_deo_broja
2. +∞, -∞
3. qNaN i sNaN

Enkodiranje preslikava ove reprezentacije u niske bitova.

Osnovni formati zapisa u IEEE 754:

1. Tri formata zapisa sa binarnom osnovom sa preslikavanjem u 32, 64 i 128bita
2. Dva formata zapisa sa dekadnom osnovom sa preslikavanjem u 64 i 128bita

Parametri	Formati			
	Binary16	Binary32	Binary64	Binary128
sirina polja (k)	16	32	64	128
bit za znak	1	1	1	1
duzina polja exp	5	8	11	15
preciznost (p)	11	24	53	113
frakcija (t)	10	23	52	113
emax	14	126	1022	16382
emin	-15	-127	-1023	-16383
uvecanje exp (E)	15	127	1023	16383

Parametri	Formati		
	Decimal32	Decimal64	Decimal128
sirina polja (k)	32	64	128
bit za znak	1	1	1
duzina polja komb	11	13	17
preciznost (p)	7	16	34
evecanje exp (E)	101	398	6176

40. Koje specijalne vrednosti postoje? Ukratko ih opisati.

1. Normalni brojevi – realni brojevi u intervalu $[\beta^{emin}, \beta^{emax} \times (\beta-\beta^{1-p})]$. Poseduju eksponent $e \in [emin, emax]$, implicitni bit je 1, dok frakcija moze da ima proizvoljnu vrednost.
2. Subnormalni brojevi - brojevi koji su po apsolutnoj vrednosti manji od β^{emin} (ali veci od 0). Uvek imaju manje od *p* znacajnih cifara. Bez subnormalnih brojeva svaki broj manji od β^{emin} je zamenjivan nulom. Sa subnormalnim brojevima granica je pomerena blize ka 0 cime se postize tzv. postepeno potkoracenje.
3. Beskonacno - predstavlja nacin za nastavak izvrsavanja programa u slucaju pojave prekoracenja i moze biti $+\infty$ ili $-\infty$. Predstavlja se u pokretnom zarezu sa eksponentom emax+1 (sve jedinice) i frakcijom koja je 0.
4. Oznacena nula - iako postojanje dve nule ima nedostataka IEEE komitet je procenio da ima vise koristi od postojanja oznacene nule nego nedostataka. Tako, kada mnozenje ili deljenje sadrzi oznacenu nulu, njen znak utice na znak krajneg rezultata. Takodje, u slucaju neoznacene nule ne bi vazila relacija $1/(1/x)=x$ za $x=\infty$. Razlog je sto i $1/-\infty$ I $1/+\infty$ kao rezultat daju 0 a $1/0=+\infty$, pri cemu se gubi informacija o znaku. 0 se predstavlja pomocu eksponenta emin-1 dok je frakcija jednaka 0.
5. NaN - problemi koji se desavaju kod izracunavanja 0/0 ili $\sqrt{-1}$ izbegavaju se uvodjenjem specijalnih vrednosti koji nisu brojevi i nazvane su NaN (not a number). Predstavljaju se kao broj u pokretnom zarezu sa eksponentom emax+1 (sve jedinice) i frakcijom koja nije 0. Postoje:

   1) Signalni NaN (sNaN) - signalizira izuzeto stanje kod aritmetickih operacija, poredjenja i konverzije kod operacija koje su deo standarda

   2) Tihi NaN (qNaN) - predstavlja pojavu nedozvoljene operacije u programu. Propagira se kroz izracunavanje tako da ostaje vidljiv na njegovom kraju, pri tome ne signalizira pojavu izuzeca.

Aritmeticke operacije:

• Ako podatak predstavlja normalan broj, subnormalan broj ili nulu, rezultat se dobija u skladu sa uobicajenim nacinom izracunavanja.

• Pravilo za odredjivanje rezultata operacije koja ima beskonacno kao operand - zameni se ∞ sa konacnim brojem x i odredi granica kada $x \to \infty$. Ako granica kada $x \to \pm\infty$ ne postoji, tada je rezultat operacije qNaN. Ako je jedan od argumenata izraza NaN, tada je vrednost celog izraza qNaN, a ako je argument ±∞ tada vrednost izraza moze biti vazeci broj u pokretnom zarezu. qNaN oznacava prisustvo neinicijalizovanih promenljivih, pogresne operacije ili prosirenja u aritmetici koja nisu deo standard. sNaN moze da oznaci postojanje informacije o pogresnim ili nepostojecim podacima.

41. Sta su izuzeta stanja, zastavice i zamke?

Pri pojavi *izuzetog stanja*, prema IEEE standardu, predaje se rezultat i nastavlja se sa radom. Tipicni predefinisani rezultati su NaN za 0/0 i ∞ ili prekoracenje za 1/0. IEEE 754 standard deli izuzeta stanja u 5 klasa:

• prekoracenje, potkoracenje, deljenje sa nulom, pogresna operacija i netacnost

Za svako od ovih izuzeca se postavlja posebna *zastavica* koju korisnik moze programski da ispituje. Standard takodje omogucuje upotreba programskih rutina za rad sa *zamkama*.

42. Sta je prosireni, a sta prosirivi zapis u IEEE-754 standardu? Koji su razlozi njihovog uvodjenja?

Prosireni zapis - format koji prosiruje podrzane osnovne formate zapisa, kako po preciznosti, tako i po opsegu eksponenta.

Prosirivi zapis - format zapisa kod koga korisnik odredjuje velicine preciznosti i opsega.

Standard preporucuje koriscenje prosirenog i prosirivog zapisa radi povecanja preciznosti u operacijama. Kako se radi sa vecom preciznoscu manja je mogucnost pojave gresaka u krajnjem rezultatu. Takodje, zbog veceg opsega eksponenta manje je prekida zbog prekoracenja u medjukoracima izracunavanja kod kojih je krajnji rezultat korektan.

parametri	Prosireni formati pridruzeni formatu				
	binary32	binary64	binary128	decimal64	decimal128
p cifara ≥	32	64	128	22	40
emax	+1023	+16383	+65535	+6144	+24576

43. Navesti nacine zaokruzivanja prema IEEE-754 standardu.

1. Zaokruzivanje na najblizu vrednost

a) *zaokruzivanje na parnu cifru* - medjurezultat izracunavanja se zaokruzuje na najblizu predstavljivu vrednost, uz zaokruzivanje na parnu cifru kada je izracunati medjurezultat na sredini intervala izmedju dve predstavljive vrednosti. Ovaj nacin zaokruzivanja je predefinisan za zapis sa binarnom osnovom

b) *zaokruzivanje na udaljeniju vrednost* - medjurezultat izracunavanja se zaokruzuje na najblizu predstavljivu vrednost, uz zaokruzivanje na vecu (po apsolutnoj vrednosti) ukoliko se nalazi na sredini intervala izmedju dve predstavljive vrednosti

2. Usmereno zaokruzivanje

a) *zaokruzivanje ka pozitivnom* (zaokruzivanje ka +∞) - rezultat se zaokruzuje na vecu vrednost. Moze se izvrsiti dodavanjem jedinice na mesto od koga se odbacuju ostale cifre za broj koji je pozitivan ili jednostavnim odsecanjem cifara sa desna

b) *zaokruzivanje ka negativnom* (zaokruzivanje ka -∞) - rezultat se zaokruzuje na manju vrednost. Moze se izvrsiti oduzimanje jedinice na poslednjem mestu od koga se odbacuju ostale cifre za broj koji je negativan ili jednostavnim odsecanjem cifara sa desna za ostale brojeve

c) *zaokruzivanje ka nuli* - rezultat je broj koji je najblizi broju, a ne veci po apsolutnoj vrednosti. Jednostavno se odseku sve cifre sa desne strane

44. Sta je enkodiranje u IEEE-754 standardu?

U standardu su predvidjene dve mogucnosti za zapis – dekadno i binarno enkodiranje. *Dekadno enkodiranje* omogucava jednostavniju hardversku, dok *binarno* omogucava jednostavniju softversku implementaciju. Skup realnih brojeva koji moze da se zapise je isti u oba kodiranja. Implementacija mora da obezbedi i mogucnost konverzije izmedju oba nacina enkodiranja.

45. Red veličine brojeva (u dekadnom sistemu) zapisanih u binarnoj i dekadnoj osnovi u jednostrukoj, dvostrukoj i četvorostrukoj tačnosti.

	jednostruka	dvostruka	cetvorostruka
binarna	$1.2x10^{-38} \leq \|X\| \leq 3.4x10^{+38}$	$2.2x10^{-308} \leq \|X\| \leq 1.8x10^{+308}$	$3.4x10^{-4932} \leq \|X\| \leq 1.2x10^{+4932}$
dekadna	$1.0x10^{-95} \leq \|X\| \leq 1.0x10^{+96}$	$1.0x10^{-383} \leq \|X\| \leq 1.0x10^{+384}$	$1.0x10^{-6143} \leq \|X\| \leq 1.0x10^{+6144}$

46. Totalno uredjenje predstavljenih vrednosti.

Poredak je zasnovan na vrednostima znakova, eksponenta i frakcije. Svaka pozitivna vrednost je veca od negativne. Ukoliko su vrednosti istog znaka vrsi se leksikografsko poredjenje vrednosti eksponenata i frakcija. Kod zapisa pomocu dekadne osnove dobija se poredak izmedju istih vrednosti koje su zapisane sa razlicitim brojem znacajnih cifara (tj. nulama na kraju).

Gledajuci samo eksponente vazi:

qNaN>sNaN>+∞>N_{max}>+konacan broj>+N_{min}>+D_{min}>0>-D_{min}>-N_{min}>-konacan broj>-N_{max}>-∞>sNaN>qNaN

47. IEEE-754 zapis sa binarnom osnovom.

Svaki broj u pokretnom zarezu zapisuje na jedan nacin. Da bi ovo bilo pravilno izvedeno frakcija se povecava dok exponent smanjujemo, sve dok e=emin ili m≥1. Ako je e=emin i 0<m<1 onda smo dobili subnormalan broj u pokretnom zarezu.

Format zapisa:

1. znak broja velicine *1 bit*
2. uvecani exponent zapisan u *w bita*
3. frakcija ili znacajan deo $d_1 d_{-2} ... d_{p-1}$ zapisan u duzini *p-1 bita*

Binary32

$\pm(1.d_{-1} d_{-2} ... d_{-23})_2 * 2^{exp}$

- *1 bit* se koristi za zapis znaka: 0 ako je pozitivan ili 1 ako je negativan
- *8 bitova* za zapis eksponenta – exp se uvek zapisuje sa uvecanjem 127, exp∈ [-126, +127]
  exp∈ [0, +254]
- *23 bita* za zapis frakcije – zapisuju se samo cifre iza decimalne tacke, a implicitna jedinica se podrazumeva

Binary64

$\pm(1.d_{-1} d_{-2} ... d_{-52})_2 * 2^{exp}$

- *1 bit* se koristi za zapis znaka: 0 ako je pozitivan ili 1 ako je negativan
- *11 bitova* za zapis eksponenta – exp se uvek zapisuje sa uvecanjem 1023, exp∈ [-1022, +1023]
  exp∈ [0, +2046]
- *52 bita* za zapis frakcije – zapisuju se samo cifre iza decimalne tacke, a implicitna jedinica se podrazumeva

Zapis specijalnih vrednosti:

1. +0	0	00...00	0000...0
2. -0	1	00...00	0000...0
3. +∞	0	11...11	0000...0
4. -∞	1	11...11	0000...0
5. sNaN	0/1	11...11	0...(≠0)
6. qNaN	0/1	11...11	1...(proizvoljno)
7. subnormalni	0/1	00...00	...(≠0) - implicitni bit frakcije je 0, exp=-126/-1022

48. IEEE-754 zapis sa dekadnom osnovom.

Kohorta je skup razlicitih reprezentacija u koje se preslikava broj u pokretnom zarezu. Ako reprezentacija konacnog ne-nula broja ima n dekadnih cifara, odgovarajuca kohorta ce imati najvise *p–n+1* clanova (p – broj cifara preciznosti). Clanovi iste kohorte mogu da se razlicito ponasaju u aritmetickim operacijama.

Kanonicki zapis - predstavljanje brojeva pomocu kanonickih dekleta. Razlog uvodjenja kanonicke reprezentacije je postojanje velikog broja razlicitih preslikavanja koja mogu da se koriste za kodiranje tri dekadne cifre u 10 binarnih cifara. Postoje 24 nekanonickih bit-kombinacija koje mogu da budu ulazne, ali ne i izlazne vrednosti iz aritmetickih operacija. Sva enkodiranja koja sadrze neku od nekanonickih kombinacija bitova se nazivaju *nekanonicka*.

DPD kodiranje (decimal32)

$\pm(d_1d_2d_3d_4d_5d_6d_7)_{10}*10^{exp}$

- *1 bit* se koristi za zapis znaka: 0 ako je pozitivan ili 1 ako je negativan
- *11 bitova* za kobinaciju – kombinacija ukljucuje zapis eksponenta i zapis prve cifre frakcije d_1. Eksponent se zapisuje kao broj duzine 8 bita sa uvecanjem 101, a od vrednosti cifre d_1 zavisi format kombinacije:
  1. d_1 mala cifra – kombinacija: prva dva bita eksponenta + kod d_1 duzine 3 bita + 6 bitova exp
  2. d_1 velika – kombinacija: bitovi 11 + prva dva bita exp + kod d_1 duzine 1 bit + 6 bitova exp
- *20 bitova* za zapis ostatka frakcije – niz cifara $d_2d_3d_4d_5d_6d_7$ se razdvaja na dve grupe od po 3 dekadne cifre $d_2d_3d_4$ i $d_5d_6d_7$, a zatim se svakoj od njih pridruzuje DPD kod duzine 10 bitova prema pravilima za kodiranje iz tablice

Zapis specijalnih vrednosti:

1. +0	0	00|000|.....	0000...0
		01	
		10	
2. -0	1	00|000|.....	0000...0
3. +∞	0	11110|.....	….(proizvoljno)
4. -∞	1	11110|.....	….(proizvoljno)
5. sNaN	0/1	11111|1|.....	….(proizvoljno)
6. qNaN	0/1	11111|0|.....	….(proizvoljno)

BID kodiranje (decimal32)

$\pm(d_1d_2d_3d_4d_5d_6d_7)_{10}*10^{exp}$

- *1 bit* se koristi za zapis znaka: 0 ako je pozitivan ili 1 ako je negativan
- *11 bitova* za kombinaciju – razlikujemo sledece slucajeve:
  1. ako je prevod frakcije $(d_1d_2d_3d_4d_5d_6d_7)_{10}$ u sistem sa osnovom 2 duzine 23 bita, kombinacija se sastoji od 8 bitova eksponenta koji se zapisuje sa uvecanjem 101 i 3 visa bita prevoda frakcije
  2. ako je prevod frakcije $(d_1d_2d_3d_4d_5d_6d_7)_{10}$ u sistemu sa osnovom 2 duzine 24 bita, kombinacija pocinje bitovima 11, zatim sledi 8 bitova eksponenta koji se zapisuje sa uvecanjem 101, a zatim 1 bit kojim se kodira cetvorobitni pocetak prevoda frakcije: ako prevod frakcije pocinje bitovima 1000 zapisuje se bit 0, a ako pocinje bitovima 1001 zapisuje se bit 1
- *20 bitova* za zapis ostatka prevoda frakcije u sistem sa osnovom 2

Uopsteno:

Znak broja velicine *1 bit*. Polje za kombinaciju velicine w+5 bita koje kodira: klasifikaciju, za konacne brojeve uvecani exponent je velicine w+2 bita, cifru najvece tezine znacajnog dela za DPD ili 3-4 cifre najvece tezine frakcije kod BID. Ostatak frakcije koji sadrzi 10*J bita, tako da je omogucen zapis 3*J+1 cifre u dekadnom sistemu.

Odredjivanje reprezentacije vrsimo pomocu 5 bitova najvece tezine u zapisu kombinacije $G_0G_1G_2G_3G_4$:

1. jednaki 11111, vrednost je NaN. Ukoliko je G_5=1 vrednost je sNaN, a inace qNaN
2. jednaki 11110, vrednost je ∞
3. izmedju 00000-11101 u pitanju je konacan broj i tu imamo razlicite mogucnosti:

(DPD) a) 5 bitova u intervalu 0xxxx-10xxx, tada su G_0G_1 dva bita najvece tezine eksponenta, $G_2G_3G_4$ kodiraju cifru na najvecoj poziciji frakcije, a ostali bitovi kombinacije cine nastavak eksponenta

b) 5 bitova u intervalu od 110xx-1110x, onda je vrednost cifre najvece tezine frakcije $100G_4$, pocetak ekponenta je G_2G_3, a nastavak frakcije krece od G_5 do kraja

c) ako je frakcija jednaka 0, a prvih 5 bita kombinacije su 00000, 01000 ili 10000, onda se radi i oznacenoj nuli

(BID) a) Ukoliko su G_0G_1 jednaki 00, 01 ili 10 onda se uvecani exp formira od prvih w+2 cifre, a ostale 3 cifre su pocetak frakcije

b) Ukoliko su G_0G_1 jednaki 11, a G_2G_3 su 00, 01 ili 10 onda se exponent formira od G_2 pa do predposlednje cifre

Nacin kodiranja ne utice na izbor velicine uvecanja exponenta i broja cifara u znacajnom delu.
Broj bitova u eksponentu i nastavku frakcije (DPD kodiranje):

- eksponent - jednostruka 8, dvostruka 10, cetvorostruka 14
- frakcija- 20, 50, 110

49. Zapis brojeva sa heksadekadnom osnovom.

Tipican predstavnik racunara koji su koristili ovakav nacin zapisa je familija S/360 i S/370. Kao i u slucaju binarne osnove znacajan deo broja se zapisuje u obliku znaka i apsolutna vrednosta; znak znacajnog dela broja je istovremeno i znak broja; upisuje se u bit najvece tezine i ima vrednost 0 za pozitivne i 1 za negativne brojeve. Frakcija znacajnog dela broja je takodje normalizovana; zapisuje se u 6 heksadekadnih cifara u 24 bita.
Za velicinu eksponenta e vazi: $-2^6<e\leq 2^6-1$
Za vrednost s kojom se predstavlja frakcija vazi: $16^{-1} \leq |s| \leq 1-16^{-6}$
Kombinacijom ovih vrednosti dobija se interval brojeva koji mogu da se zapisu:

$16^{-1}*16^{-64} \leq |x| \leq (1-16^{-6})*16^{+63}$ tj. $5{,}4*10^{-79} \leq |x| \leq 7{,}2*10^{+75}$

Jednostruka tacnost:

$\pm 0.d_1d_2d_3d_4d_5d_6*16^{exp}$

Za normalizovane brojeve vazi da je $d_1\neq 0$. Karakteristike zapisa:

- *1 bit* za znak broja – 0 ili 1
- *7 bita* za eksponent – zapisuje se sa uvecanjem 64, expϵ [-64, +63]
  expϵ [0, +127]
- *24 bita* za frakciju – zapisuje se sa 6 heksadekadnih cifara, od kojih svaka zauzima tacno 4 cifre u binarnom zapisu, a implicitna nula se ne zapisuje.

Denormalizovani brojevi u zapisu imaju sve nule u eksponentu, a u frakciji je $d_1=0$. Dekadna vrednost i normalizovanih i denormalizovanih bojeva odredjuje se na isti nacin.

50. Aritmetičke operacije u pokretnom zarezu (IEEE754 format).

Aritmetika u pokretnom zarezu

Osnovne arimtičke operacije u pokretnom zarezu se vrše u skladu sa pravimila aritmetičkih operacija za stepene jednakih osnova. Eksponent i frakcija čuvaju u odvojenim registrima, ali po završetku izračunate vrednosti eksponenta i frakcije se ponovo spajaju i formira se broj u pokretnom zarezu koji je rezultat operacije. Aritmetika sa dekadnim brojevima uključuje i izbor odgovarajuće kohorte. U slučaju da operacija sa dekadnim brojevima ne proizvodi tačan broj kao rezultat, da bi se dobio najveći broj značajnih cifara koristi se član kohorte sa najmanjim eksponentom. Ako je rezultat tačna vrednost, bira se član kohorte zasnovan na ciljanom eksponentu za rezultat operacije.

Sabiranje i oduzimanje

1. provera postojanja specijalnih vrednosti
2. oduzimanje x-y se realizuje kao x+(-y) uz prethodnu promenu znaka argumentu y
3. ukoliko je jedan od sabiraka jednak nuli, vrednost drugog sabirka je razultat sabiranja
4. svodjenje sabiraka na iste eksponente - vrši se povećavanjem manjeg eksponenta uz istovremeno pomeranje cifara frakcije udesno. Ako pri pomeranju značajni deo postane nula, tada vrednost drugog sabirka postaje vrednost rezultata
5. sabiraju se frakcije sabiraka, pri čemu se uzimaju u obzir njihovi znaci. Sabiranje se vrši po pravilima za sabiranje celih brojeva u ZAV. Ako je pri sabiranju došlo do prekoračenja, dobijeni rezulatat se pomera za jedno mesto udesno uz povećanje vrednosti eksponenta za jedan. Ako ovo povećanje vrednosti eksponenta dovede do prekoračenja, ukupan rezultat sabiranje je $+\infty$ ili $-\infty$ u zavisnosti od znaka broja.
6. traženi zbir predstavlja broj u pokretnom zarezu čiji su znak i frakcija jedanki znaku i frakciji delova sabiraka, a eksponent jednak eksponentu sabiraka. Ako u rezultatu sabiranja frakcija nije normalizovana ili nije predstavljena pomoću ciljanog eksponenta, pokušava se njegova normalizacija odnosno traži ciljani eksponent

Mnozenje

1. proverava se postojanje specijalnih vrednosti
2. ukoliko je bar jedan od činilaca jednak nuli, rezultat je 0
3. saberu se vrednosti eksponenta i od dobijenog zbira oduzme uvećanje. Ako je došlo do prekoračenja pri ovom sabiranju, krajnji rezultat je $\pm\infty$. Ako je došlo do potkoračenja rezultat je pozitivna ili negativna nula
4. množe se frakcije. Množenje se vrši prema pravilima za množenje celih brojeva zapisanih pomoću znaka i apsolutne vrednosti
5. dobijeni rezultata se normalizuje, odnosno odredi se ciljani eksponent
6. broj cifara u proizvodu je dvostruko veći od broja cifara vrednosti koje su pomnožene; cifre koje su višak se odbacuju u procesu zaokruživanja

Deljenje

1. proverava se postojanje specijalnih vrednosti
2. ako je delilac nula, tada je rezultat ili NaN ili $\pm\infty$ u zavisnosti od deljenika
3. oduzmu se vrednosti eksponenata i na dobijenu razliku doda uvećanje. Ako je došlo do prekoračenja, krajnji rezultat je $\pm\infty$. Ako je došlo do potkoračenja, krajnji rezultat je pozitivna ili negativna nula
4. podele se značajni delovi brojeva. Deljenje se vrši prema pravilima za deljenje celih brojeva zapisanih pomoću znaka i apsolutne vrednosti
5. Dobijeni rezultat se normalizuje, odnosno odredi se ciljani eksponent
6. Dobijeni količnik se zaokružuje prema pravilima za zaokruživanje

51. Brojčani sistemi sa ostacima.

Brojčani sistem sa ostacima je pozicioni brojčani sistem sa više osnova kod koga svaka pozicija u zapisu broja ima različitu osnovu. Ovaj brojčani sistem je zasnovan na relaciji kongruentnosti. Za dva cela broja a i b se kaže da su kongruentni po modulu m ako m deli bez ostatka razliku a-b. Broj m se naziva moduo ili osnova. U aritmetici ostataka, ostatak pri deljenju brojeva a i b je broj r koji se dobija kada se od a oduzme najveći mogući broj c, tako da je c = d*b, tj. r = a – d*b. Za r se kaže da je ostatak u odnosu na m i označava se sa $r = |a|_m$.

Za dati skup pozitivnih celih brojeva koji su veći od 1, RBS $(m_n|m_{n-1}|\ldots|m_1)$ označava brojčani sistem sa ostacima $m_n, m_{n-1}, \ldots, m_1$.

Da bi se izbegla višeznačnost zapisa brojeva i pojava duplikata, moduli moraju da budu uzajamno prosti brojevi $m_n, m_{n-1}, \ldots, m_1$. pri čemu važi $m_n > m_{n-1} > \ldots > m_1$. U ovom sistemu se moze predstaviti ukupno $M = m_n \cdot m_{n-1} \cdot \ldots \cdot m_1$ vrednosti.

Interval:

- za neoznacene brojeve [0, 839]
- za oznacene brojeve [-420, 419]
- [-N, P], gde vazi M=N+P+1

Primer: zapis broja 62 u RBS (8|7|5|3) je:

$r_1 = 62 \bmod 8 = 6$; $r_2 = 62 \bmod 7 = 6$; $r_3 = 62 \bmod 5 = 2$; $r_4 = 62 \bmod 3 = 2$

$(62)_{10} = (6|6|2|2)_{RBS(8|7|5|3)}$

52. Sta je aditivni, a sta multiplikativni inverz?

Aditivni inverz - za ostatak x definisan kao x+x'=0, izracunava se kao $x = |m - x|_m$ i koristi se za zapis negativnih brojeva.

Multiplikativni inverz je A^{-1}, za uzajamno proste ne-nula brojeve A i m, ako vazi $|A * A^{-1}|_m = 1$

53. Izbor modula.

Za module mozemo uzeti bilo koje cele brojeve, ali zbog zahteva koje postavlja implementacija u racunaru potrebno je izvrsiti neku optimizaciju pri izboru. Treba koristiti sto manje module, nema svrhe koristiti jedan veliki i ostale male module, jer taj veliki moduo usporava izvrsavanje operacija. Nacini izbora:

1. uzimamo redom proste brojeve dok njihov proizvod ne postane veci od postavljene granice. Na kraju mozemo pomnoziti neka dva broja, tako da dobijemo jedan veci u opsegu bita kao najveci od njih. Takodje mozemo i izbaciti neke manje brojeve tako da proizvod ostalih zadovoljava granicu.
2. bolji sistem je da redom uzimamo proste brojeve, ali pre nego sto uzmemo sledeci prost broj, ukljucimo i stepen manjih brojeva, a taj manji broj iskljucimo.
3. najbolje je koristit module koji su stepeni broja 2 ili stepeni broja dva umanjeni za jedan. Takodje ovi brojevi su uzajamno prosti i uz pomoc njih dobijamo najbolju optimizaciju.

54. Modularna aritmetika.

Efikasnost korišćenja brojčanog sistema sa ostacima zavisi od mugućnosti efikasnog izvodjenja osnovnih aritmetičkih operacija kao i poredjenja dve vrednosti.

- ***Aditivni inverz***

Za ostatak x, aditivnvi inverz $\bar{x}$ je definisan pomoću jednakosti $x+\bar{x}=0$. Aditivni inverz je jedinstven za svaki ostatak m i izračunava se kao $\bar{x}=|m-x|_m$. Njegova glavna primena je u predstavljanje negativnih brojeva i oduzimanje, i on se može primeniti na pojedinačne ostatke i na kompletan zapis broja.

Primer:

$(62)_{10} = (6|6|2|2)_{RBS(8|7|5|3)}$

$|8-6|_8 = 2$; $|7-6|_7 = 1$; $|5-2|_5 = 3$; $|3-2|_3 = 1$

$(\overline{6}|\overline{6}|\overline{2}|\overline{2})_{RBS(8|7|5|3)} = (2|1|3|1)_{RBS(8|7|5|3)}$

• ***Zapis negativnih brojeva***

Formalno, u RBS($m_n|m_{n-1}|\ldots|m_1$) negativni brojevi mogu da se predstave u zapisu pomoću komplementa. Zapis negativnog broja se dobija i nalaženjem aditivnog inverza apsolutne vrednosti broja.

Primer:

$(-62)_{10}$ u RBS(8|7|5|3)

Pomoću zapisa broja koji se dobija komplementiranjem sa komplemenatacionom konstantom 840. Kako je 840-62=778, a zapis broja 778 u RBS (8|7|5|3) je:

r_1=778 mod 8=2; r_2=778 mod 7=1; r_3=778 mod 5=3; r_4=778 mod 3=1

Odavde je $(-62)_{10} = (2|1|3|1)_{RBS(8|7|5|3)}$

• ***Sabiranje i oduzimanje***

U modularnoj aritmetici važi sledeće pravilo: $|A \pm B|_m = |\,|A|_m \pm |B|_m\,|_m$. Sabiranje i oduzimanje se izvode nezavisno nad svakom od pojedinacnih cifara. Oduzimanje se može izvesti kao sabiranje sa adititvnim invezom umanjioca.

Primer:

Neka je RBS=(8|7|5|3), i neka su brojevi $A = 26 = (2|5|1|2)_{RBS(8|7|5|3)}$ i $B = 12 = (4|5|2|0)_{RBS(8|7|5|3)}$

$C = A+B = ((2+4)\text{mod } 8|(5+5)\text{mod } 7|(1+2)\text{mod } 5|(2+0)\text{mod } 3)_{RBS(8|7|5|3)} = (6|3|3|2)_{RBS(8|7|5|3)}$

$C = A-B = ((2-4)\text{mod } 8|(5-5)\text{mod } 7|(1-2)\text{mod } 5|(2-0)\text{mod } 3)_{RBS(8|7|5|3)} = (-2|0|4|-1)_{RBS(8|7|5|3)}$

$=(6|0|4|2)_{RBS(8|7|5|3)}$

• ***Mnozenje***

U modularnoj aritmetici važi sledeće pravilo: $|A \times B|_m = |\,|A|_m \times |B|_m\,|_m$. Mnozenje se izvodi nezavisno nad svakom od pojedinacnih cifara.

Primer:

Neka je RBS=(8|7|5|3), $A = 26 = (2|5|1|2)_{RBS(8|7|5|3)}$, $B = 12 = (4|5|2|0)_{RBS(8|7|5|3)}$

$C = A \times B = ((2 \times 4)\text{mod } 8|(5 \times 5)\text{mod } 7|(1\times 2)\text{mod } 5|(2\times 0)\text{mod } 3)_{RBS(8|7|5|3)} = (0|4|2|0)_{RBS(8|7|5|3)}$

• ***Mnozenje modulom***

Za svaku celobrojnu vrednost k i moduo m važi:

1. $|k \times m|_m = 0$
2. $|A \pm k \times m|_m = |A|_m$

Na osnovu njih skraćuje se postupak sabiranja i oduzimanja u slučajevima kada je jedan od sabiraka umnožak modula, tako što se ne vrši sabiranje ili oduzimanje na poziciji tog modula.

Primer:

$A = 26 = (2|5|1|2)_{RBS(8|7|5|3)}$, $B = 14 = (6|0|4|2)_{RBS(8|7|5|3)}$

$A+B = (2+6|5+0|1+4|2+2)_{RBS(8|7|5|3)} = (0|5|0|1)_{RBS(8|7|5|3)}$

Kako je 14 = 2×7, prema prethodnim pravilima ne treba izvršiti sabiranje na poziciji koja odgovara modulu 7: $A+B = (2+6|5+4|2+2)_{RBS(8|7|5|3)} = (0|5|0|1)_{RBS(8|7|5|3)}$

• ***Multiplikativni inverz***

Za uzajmno proste ne-nula brojeve *A* i *m*, A^{-1} je multiplikativni inverz broja A u odnosu na moduo m ako važi: $|A \times A^{-1}|_m = 1$

Primer:

Neka je A = 5. Multiplikativni inverz u odnosu na moduo 11 je vrednost A^{-1} za koju važi $|5 \times A^{-1}|_{11} = 1$. Za rešavanje se može koristiti diofanska jednacina 5*x = 11*y +1. Jedno od rešenja jednacine je x=9, y=4. Odavde se dobija da 9 je multiplikativni inverz broja 5 u odnosu na moduo 11 (jer 5*9=45, 45 mod 11 = 1).

• ***Deljenje***

Deljenje je u modularnoj aritmetici najsloženija od osnovnih aritmetičkih oprecaija. Deljenje se može definisati na pojedinačnim brojevima i proširiti tako da važi za skupove n-torki koje prestavljaju zapise brojeva u RBS, ali se javljaju problemi jer količnik ne mora uvek da bude ceo broj. U brojčanom sistemu sa ostacima jednakost se može predstaviti ekvivalencijom:

$c \times b \equiv a \pmod m$. Množenjem obe strane sa multiplikativnim inverzom od b dobija se:

$c \equiv a \times b^{-1} \pmod m$. C predstavlja količnik samo u slučaju da je ceo broj.

Primer:

Neka m = 7 i neka treba izracunati kolicnik c = 6/2. Odavde se dobija:

$2*c \equiv 6 \pmod 7 \equiv 6 * 2^{-1} \pmod 7 \equiv 6 * 4 \pmod 7 \equiv 3 \pmod 7$

55. Prevođenje brojeva.

Prevodjenje iz dekadnog u RBS:

Vrsi se tako sto se zeljeni broj deli svakim modulom posebno i ostatak pri tom deljenju se upisuje redom sa leva na desno, potrebno je n cifara, gde je n broj modula.

Prevodjenje iz binarnog u RBS:

Izvodi se pomocu tabele u koju se upisu ostaci pri deljenju stepena broja 2 sa modulima za odredjeni sistem. Onda se izracunavanje vrsi tako sto se saberu sve vrednosti ostataka iz tabele koje odgovaraju poziciji jedinica u binarnom broju, ali se sabiranje vrsi po modulu.

Prevodjenje brojeva iz RBS:

Moraju se odrediti tezine svake pozicije, a to se vrsi tako sto proizvod modula na ostalim pozicijama mnozimo sa multiplikativnim inverzom u odnosu na taj moduo.

Vrednost broja iz RBS u dekadnom sistemu se dobija kada tezinu svake pozicije pomnozimo sa ostatkom na toj poziciji, pa sve saberemo po modulu koji odgovara proizvodu svih modula ovog sistema. Prevodjenje iz RBS u binarni sistem je analogno prevodjenju u dekadni, a racunari poseduju predefinisane tablice za odredjene RBS radi lakseg racunanja.

56. Navesti prednosti i nedostatke brojcanog sistema sa ostacima.

Nedostaci:

- slozenost za implementaciju
- relativno losa efikasnost testiranja broja, poredjenja, otkrivanja prekoracenja i deljenja

Prednosti:

- ne postoji problem prenosa izmedju pozicija pri sabiranju i mnozenju
- cifre su male cak i za velike brojeve
- aritmetika je jednostavna i brza
- mogucnost izolacije pojedinacnih cifara koje mogu da imaju gresku

Primena:

- u aplikacijama koje zahtevaju otpornost na greske
- u apliakcijama u kojima se uglavnom koriste mnozenje i sabiranje
- pri obradi digitalnih signala (posebno pri radu sa digitralnim filetrima)
- u aplikacijama koje treba da preduzmu odredjenu vrstu akcije u slucaju pojave greske u racunarskim sistemima

57. Na koji nacin se definisu logicke konstante, logicke promenljive i formule algebre logike?

Logicke konstante su elementi od S, tj. $\top$(tacno) i $\bot$(netacno).

Logicke promenljive algebra logike su znaci A,B,C…

Formule algebre logike se definisu na sledeci nacin:

1. Logicke konstante i logicke promenljive su formule algebra logike
2. Ako su P i Q formule algebre logike tada su $\neg P, \neg Q, (P \wedge Q), (P \vee Q)$ takodje formule algebre logike
3. Formule algebra logike se dobijaju jedino primenom pravila 1. i 2.

58. Defisati pojam logicke funkcije. Koliki je broj razlicitih logickih funkcija od n argumenata? Navesti sve logicke funkcije jednog argumenta.

Neka su A_1, A_2,..., A_n logicke promenljive. Svaka funkcija $f : (A_1, A_2,..., A_n) \rightarrow \{0,1\}$ se naziva *logicka funkcija*.

Kako svaka od logickih promenljivih koja je argument funkcije moze imati vrednost 0 ili 1, to je broj razlicitih logickih funkcija od n argumenata jednak 2^{2^n} . Postoje ukupno 4 logicke funkcije od jednog i 16 logickih funkcija od dva argument.

Sve logicke funkcije jednog argument su: nula funkcija, identitet, negacija i jedinicna funkcija.

59. Sta je pun sistem funkcija? Navesti primere punih sistema funkcija.

Neka je f_i neka od funkcija algebre logike sa jednim ili dva argumenta. Skup $F=\{f_1, f_2,..., f_n\}$ funkcija algebre logike se naziva *pun sistem funkcija* ako se proizvoljna funkcija algebre logike moze predstaviti pomocu funkcija iz ovog skupa.

Ako se pomocu funkcija iz skupa $G=\{g_1, g_2,..., g_n\}$ mogu izraziti sve funkcije nekog punog sistema funkcija, tada i G predstavlja pun sistem funkcija.

Pun sistem funkcija F se naziva *minimalan* ako skup $G=F - \{f_i\}$, dobijen izostavljanjem bilo koje funkcije, vise ne predstavlja pun sistem funkcija.

Primeri:

$F=\{\bar{}, \cdot\}$, $F=\{\bar{}, +\}$, $F=\{\uparrow\}$, $F=\{\downarrow\}$

60. Definisati sledece pojmove:

1. *Elementarna konjunkcija* je logicki izraz koji ne sadrzi operaciju disjunkcije.
2. *Elementarna disjunkcija* je logicki izraz koji ne sadrzi operaciju konjukcije.
3. *Savrsena elementarna konjukcija* je elementarna konjukcija koja sadrzi sve promenljive iz skupa promenljivih od kojih se grade logicki izrazi.
4. *Savrsena elementarna disjunkcija* je elementarna disjunkcija koja sadrzi sve promenljive iz skupa promenljivih od kojih se grade logicki izrazi.
5. *Disjunktivna forma* je logicki izraz koji se sastoji od elementarnih konjukcija medjusobno povezanih operacijama disjunkcije.
6. *Konjunktivna forma* je logicki izraz koji se sastoji od elementarnih disjunkcija medjusobno povezanih operacijama konjunkcije.
7. *Savrsena disjunktivna normalna forma* je disjunktivna forma u kojoj su sve konjunkcije savrsene elementarne konjunkcije. *(SDNF)*
8. *Savrsena konjunktivna normalna forma* je konjunktivna forma u kojoj su sve disjunkcije savrsene elementarne disjunkcije. *(SKNF)*

Funkcija je zapisana u obliku *savrseno konjunktivne normalne forme* ako je logicki izraz koji je predstavlja zapisan kao savrseno konjunktivna normalna forma pri cemu svaka od savrsenih elementarnih konjunkcija koja ulazi u njen sastav sadrzi sve promenljive koje su argumenti funkcije.

Funkcija je zapisana u obliku *savrseno disjunktivne normalne forme* ako je logicki izraz koji je predstavlja zapisan kao savrseno disjunktivna normalna forma pri cemu svaka od savrsenih elementarnih disjunkcija koja ulazi u njen sastav sadrzi sve promenljive koje su argumenti funkcije.

Primeri:

- Konjunktivne forme: $(A + B) \cdot (A + C)$, $(A + B) \cdot (\bar{A} + \bar{C})$
- Disjunktivne forme: $A \cdot B + A \cdot C$, $A \cdot B + \bar{A} \cdot \bar{C}$
- Savrsene konjunktivne normalne forme: $(A + B + C) \cdot (A + B + \bar{C})$, $(A + B + C) \cdot (\bar{A} + \bar{B} + \bar{C})$
- Savrsene disjunktivne normalne forme: $(A \cdot B \cdot C) + (A \cdot B \cdot \bar{C})$, $(A \cdot B \cdot C) + (\bar{A} \cdot \bar{B} \cdot \bar{C})$

61. Sta su logicki elementi i koja je njihova funkcija?

Logicki elementi (prekidaci)– osnovne komponente od kojih se formiraju sva digitalna logicka kola. To su fizicki objekti koji implementiraju neku od funkcija algebre logike. Argumenti funkcija odgovaraju ulaznim velicinama logickih elemenata, a vrednosti funkcija njihovim izlaznim velicinama.

62. Minimalizacija logickih funkcija. Nacini i kratak opis postupka.

Moguce je izvesti jednostavniji Bulovski izraz iz tabele istinitosnih vrednosti nego sto je reprezentacija u obliku SDNF ili SKNF. U najvecem broju slucajeva se implementacija logickih funkcija vrsi samo pomocu NI ili samo pomocu NILI elemenata. Nacini:

- algebarske transformacije – ukljucuju primenu osnovnih zakona i identiteta
- Karnoove mape – pogodne za minimalizaciju funkcija sa malim brojem promemenljivih. Mapa je niz od 2^n kvadrata koja predstavlja sve moguce vrednosti n binarnih promemenljivih
- tablicna metoda Kvin-MekKlaskog – u tri koraka:
  1. pretrazuje se funkcija radi nalazenja terma (prostih implikanata)
  2. formira se skup prostih implikanata i odredjuju bitni implikanti na osnovu formalnih tablica prostih implikanata
  3. termovi koji se ne nalaze u skupu bitnih implikanata dodaju se u logicki izraz koji minimalizuje datu funkciju ukljucivanjem dodatnih prostih implikanata

63. Sta su kombinatorne mreze, na koji nacin su logicki elementi povezani u njima i navesti primere kombinatornih mreza?

Kombinatorne mreze predstavljaju skup medjusobno povezanih logickih elemenata ciji je izlaz u nekom vremenskom trenutku funkcija koja zavisi samo od vrednosti ulaza u tom istom vremenskom trenutku. Neposredno iza pojavljivanja ulaza sledi i pojavljivanje izlaza, uz moguce kasnjenje koje zavisi jedino od kasnjenja logickih elemenata.

Kombinatorne mreze se u literaturi nazivaju jos i *kombinatorna kola*. U opstem slucaju, kombinatorne mreze se sastoje od *n* binarnih ulaza i *m* binarnih izlaza i mogu biti definisane na tri nacina:

1. preko tabele istinitosnih vrednosti koja sadrzi kombinacije za svaku od 2^n mogucih ulaznih vrednosti
2. preko povezanog skupa grafickih simbola.
3. preko logickih funkcija koje izrazavaju vezu izmedju ulaznih i izlaznih promenljivih (signala)

Primeri: multipleksori, dekoderi, programibilni niz logickih elemenata, samocitajuca memorija, sabiraci…

64. Sta su sekvencijalne mreze, na koji nacin su logicki elemnti povezani u njima i navesti primere sekvencijalnih mreza?

Sekvencijalne mreze predstavljaju skup povezanih logickih elemenata ciji je izlaz u nekom vremenskom trenutku funkcija koja zavisi od tekuceg stanja elemenata mreze i vrednosti ulaza u tom istom vremenskom trenutku.

Primeri: flip-flop elementi, registri, brojaci…

65. Struktura savremenog računarskog sistema. Fon Nojmanova mašina.

Fon Nojmanova masina se sastojala od centralne jedinice za obradu (procesora), unutrasnje memorije i kanala veze. Jedinica za obradu su cinili: aritmeticko-logicka jedinica i upravljacka jedinica. Oba dela su sadrzavala registre. Masina je izvrsavala instrukcije koje su prepoznavane u upravljackoj jedinici. Kompletan tok podataka od i ka memoriji je isao preko prihvatnog registra memorije. Takodje, svi podaci koji su se ucitavali ili stampali su prvo prenoseni u memoriju, a zatim se odatle prenosili u aritmeticko-logicku jedinicu.
Fon Nojman je 1945. objavio izvestaj sa modelom racunara EDVAC koji bi mogao da cuva program zajedno sa podacima. On treba da se sastoji od:

1. centralne aritmeticke jedinice
2. centralni organ za upravljanje
3. unutrasnje memorije
4. ulaza
5. izlaza

Savremeni racunarski sistem zasnovan na Fon Nojmanovoj arhitekturi ima sledece koncepte:

1. Postoji samo jedna memorija u kojoj se cuvaju programi i podaci a razlikuju se samo prema interpretaciji
2. Memorija je adresibilna, tj moze joj se pristupati bez obzira na to sta je u njoj smesteno
3. Izvrsavanje instrukcija se izvodi sekvencijalno

66. Navesti karakteristike, registre i tipove instrukcija IAS racunara. Od cega se sastoji instrukcioni ciklus IAS racunara?

Karakteristike:

- 1000 lokacija u memoriji koje su nazvane reci
- lokacija sadrzi 40 binarnih cifara
- brojevi su se predstavljali kao 1(znak) +39(vrednost)
- instrukcije su se predstavljale sa 20 bitova

Registri:

- prihvatni registar memorije PRM
- registar memorijskih adresa RMA
- prijemni registar instrukcija PRI
- instrukcioni registar IR
- brojac instrukcija PC
- akumulator AC i mnozilac/delilac MQ

Tipovi instrukcija:

- instrukcije za prenos podataka
- instrukcije bezuslovnog grananja
- instrukcije uslovnog grananja
- aritmeticke instrukcije
- instrukcije za modifikaciju adrese

Instrukcioni ciklus se sastojao od dva podciklusa:

- cilus pripreme → operacioni kod se smesta u IR, drugi deo instrukcije(adresa) se smesta u RMA, instrukcija se uzima iz PRM ili PRI
- ciklus izvrsavanja → vrsi se interpretacija operacionog koda i izvrsava se

67. Sistem prekida. Brzina obrade podataka.

Posto su izlazno/ulazni uradjaji daleko sporiji od procesora, pri izvrsenju neke naredbe procesor bi morao da ceka da se neka naredba izvrsi i tako gubio vreme. Zato je uveden *sistem prekida*.
Sistem prekida je mehanizam koji omogucava efikasniji rad racunara. Recimo pri prenosu podataka na stampac procesor dobije prekid i on nastavlja da radi neku drugu instrukciju, a kada ponovo dobije prekid, tj obavestenje da je rad gotov on se vraca na tu instrukciju.
Kontrola prekida se vrsi pomocu (nacini obrade visestrukih prekida):

1. *onemogucavanja prekida* - svi ostali prekidi cekaju dok se ne izvrsi zapoceta instrukcija
2. *definisanje prioriteta prekida*, tako da onaj sa vecim prioritetom moze da prekine izvrsavanje onog sa nizim prioritetom

Brzina obrade podataka:

1. MIPS - broj masinskih instrukcija koje CPU moze da obradi u jednoj sekundi
2. FLOPS - broj operacija u pokretnom zarezu koje mogu da se obrade u jednoj sekundi (koristi se za merenje brzine super racunara)
3. Merenje brzine izvrsavanja jednog instrukcijskog ciklusa

68. Organizacija centralnog procesora. Registri.

Kod prvih generacija procesora glavne komponente su bile ALU i CU.
ALU je aritmeticko-logicka jedinica i ona sluzi za obradu podataka, a *CU* je kontrolna jedinica koja vrsi kontrolu izvrsavanja i upravljala prenosom podataka i instrukcija u i iz ALU.
Da bi se operacije uspesno obavljale, potrebno je negde smestiti neke argumente, medjurezultate, dobijene vrednosti, naredna instrukcija i za to se koriste registri.
Registri cine internu memoriju procesora, a veza sa ostalim delovima racunarskog sistema se uspostavlja preko *magistrala*. Podaci koji se obradjuju u ALU se dobijaju prenosom iz registara, a rezultati, zastavice i indikatori se isto smestaju u registre. Postoji vise vrsta registara, ali neka najcesca podela je na:

1. *opste namene* - za razlicite funkcije. Korisnički program može da im pristupi, čita i menja njihov sadržaj bez ograničeja. (akumulatori, indeks registri, pokazivaci steka i segmenta)
2. *specijalizovane namene* - oni se koriste pri izvrsavanju operacija, njihovoj kontroli, za prikaz tekuceg stanja. Većina njih nije direktno dostupna korisničkim programima i koriste ih samo programi operativnog sistema, dok su drugi dostupni samo na nivou mikrokoda. (instrukcioni registar, registar memorijskih adresa, prihvatni registar memorije, brojac instrukcija, kontrolni registri...)

69. Navesti i ukratko opisati karatkeristike i tipove tranzistora.

Tranzistor je uredjaj sa tri zavrsna prikljucka koji u racunaru moze da bude prekidac ili pojacivac. Sastoji se od tri sloja poluprovodnickog materijala koji moze da provodi struju. Koriste se silicijum ili germanijum. Postoje bipolarni NPN i PNP tranzistori. Postoje i FET tranzistori kod kojih se prenos struje vrsi pomocu efekta polja.

1. *Bipolarni tranzistori* – sastoji se od dva para PN spojenih dioda koje su smestene jedna uz drugu. Postoje dva tipa bipolarnih tranzistora:

a) tranzistori sa PNP konfiguracijom, kod kojih je izmedju dva P-sloja poluprovodnika smesten jedan N-sloj poluprovodnika

b) tranzistori sa NPN konfiguracijom, kada je izmedju dva N-sloja poluprovodnika smesten jedan P-sloj poluprovodnika

2. *FET tranzistori* – elektrode FET tranzistora se nazivaju *izvor, vrata* i *odvod*. Vrata su razdvojena od povrsine poluprovodnika tankim slojem izolacionog materijala. Povrsina izmedju izvora i odliva se naziva *kanal*. Kanal je napravljen ili od N-tipa ili od P-tipa poluprovodnickog materijala. Najveci broj tranzistora u savremenim racunarima je MOSFET tipa. Ako se visoka voltaza primeni na metalna vrata dobija se provodnik elektriciteta. Ako se primeni niska voltaza, prekidac ostaje otvoren i elektricitet se ne provodi.

3. *CMOS tranzistori* - U CMOS tranzistorima, ako se na metalna vrata primeni niska voltaza, prekidac se zatvara i propusta elektricitet. U slucaju primene visoke voltaze, prekidac ostaje otvoren. Prednost CMOS tranzistora je sto za odrzavanje stabilnog stanja trose jako malo struje.

70. Detaljno opisite tehnologije izrade mikroprocesora koje poznajete.

Procesorske jedinice svih racunara se danas izradjuju u obliku mikroprocesora.
Mikroprocesor je cip koji sadrzi CPU i malu kolicinu memorije koja se koristi za specijalizovane namene. Do sredine 90-ih proslog veka za izradu mikroprocesora koriscen je silicijum, ali posto se doslo do granice na kojoj je silicijum otporan morala se naci neka druga tehnologija. Delom se modifikovala silicijumska, a delom uvele potpuno nove tehnologije.

1. *Litografija* (proces koji se koristi u proizvodnji procesorskih cipova)
   • umnozavanje matrice na ploce od silicijuma
   • matrica koja se prenosi sadrzi strukturu kompletnog cipa ukljucujuci tranzistore, njihove spojeve i ostale strukture
   • koristi se fotootpornik sastavljen od polimera
   • na svaki sloj na cipu se na povrsinu silicijumske oblande nanosi po jedan sloj fotootpornika
   • laserski zrak deluje na fotootpornik i rastvara osvetljene delove
   • na kraju procesa preostali fotootpornik se uklanja pomocu organskog rastvora, dok silicijumska oblanda ostaje sa narezanom zeljenom strukturom na povrsini
2. *Tehnologija sa bakarnim vezama*
   • IBM je prvi uveo tehnologiju koja je omogucila upotrebu bakra za povezivanje tranzistora
   • upotreba metala koji bolje provode struju je bila mnogo kompilikovanija od upotrebe aluminijuma
   • pronadjen je nacin da se postavi mikroskopska barijera izmedju bakra i silicijuma uz smanjenje broja koraka potrebnih za kompletiranje cipa
3. *Silicijum na izolatoru*
   • brzi poluprovodnik koji je dosta proucavan u poslednjih 30 godina
   • predstavlja tanak sloj silicijuma na vrhu izolatora kao sto je staklo. Na tom sloju mogu da se smestaju tranzistori koji rade brze
   • u slucaju MOS tranzistora, svaki put kada se ukljuci, prekidac mora da popuni svoj interni kapacitet pre nego sto pocne da provodi elektricitet. Vreme potrebno za operacije punjenja i praznjenja elektricnog naboja je relativno dugo u odnosu na vreme potrebno za ukljucivanje i iskljucivanje tranzistora.
   • mikroprocesorski cipovi konstruisani sa SOI imaju 20% do 25% krace vreme izvrsavanja instrukcionog ciklusa i 25% do 30% bolje performance od odgovarajucih cipova izgradjenih u cistoj CMOS tehnologiji
4. *Silicijum-germanijum tehnologija*
   • najvecu primenu imaju u telekomunikacionim uredjajima, automobilskim radarskim sistemima i klasi racunara koja se naziva licni digitalni pomocnik
5. *Niski-K dielektrik*
   • sustina ove tehnologije je u koriscenju ovog materijala za izolaciju integralnih kola. Time se smanjuje medjusobna interferencija
   • primenom ove tehnologije dobijaju se cipovi sa 30% boljim performansama.

71. CISC i RISC arhitektura mikroprocesora.

Nakon softverske krize izvrsavanje programa postalo je neefikasno. Da bi se to premostilo nastala je drugacija arhitektura novih modela racunara koji su donosili povecanje skupa instrukcija, nove nacine adresiranja i hardverdsku implementaciju pojedinih naredbi visih programskih jezika. Ove modifikacije su omogucile:

1. jednostavniju konstrukciju prevodilaca uz mogucnost podrske svih slozenijih prog. jezika
2. povecanje efikasnosti izvrsavanja

72. Pojava RISC tehnologije. Neke osobine RISC procesora.

Nakon zavrsetka IBM-ovog projekta 1975., njegovi projektanti su shvatili da ce ona biti odlicna osnova za mikroprocesor opste namene sa visokim performansama.

Najvaznije osobine masine koje su omogucavale dobar odnos njenih performansi i cene bile su:

- deljenje kesa za instrukcije i podatke
- nije bilo izvrsavanja aritmetickih operacija nad podacima u memoriji
- uniforma velicine instrukcije i jednostavna konstrukcija su omogucili vrlo kratko izvrsavanje ciklusa

Prvi prepoznatljiv pokusaj smanjenja performansi u slucaju izvodjenja skokova je IBM-ov racunar IBM 7030 (Stretch).

Velika prednost nove,eksperimentalne masine je to sto je bila u stanju da izvrsi mnogo veci broj instrukcija u jednom ciklusu u odnosu na druge masine. Veliki potencijal masine je predstavljala cinjenica da se jednostavna instrukcija izvrsava znatno brze za istu familiju integrisanih kola zbog vremena potrebnog za isvrsavanje CISC interpretatora. Masina je dobila ime 801. U narednim godinama je koriscen u velikom broju projekata.

73. Navesti karakteristike RISC i CISC procesora i navesti predstavnike.

Karakteristike CISC procesora:

- naredbe koje su se najsesce javljale u programima su bile naredbe dodele, uslovna naredba i poziv potprograma. Takodje, pronadjeno je da su poziv i povratak iz procedure operacije koje zahtevaju najvise vremena u tipicnim programima pisanim na visim programskim jezicima
- referisanje na operande se obavlja kao referisanje na lokalne skalarne vrednosti. Na osnovu toga je zakljuceno da je nuzna optimizacija procesa zapisivanja i pristupa lokalnim promenljivim
- broj reci potreban pri pozivu procedure nije veliki i treba obratiti paznju na pristup argumentima

Osobine RISC procesora:

- izvrsavanje (bar) jedne masinske instrukcije za jedan masinski ciklus. Ovim se smanjuje ili eliminise potreba za mikrokodom i kompletna masinska instrukcija moze da bude hardverski kodirana. Takva instrukcija se izvrsava brze od odgovarajucih instrukcija CISC procesora jer nema potrebe za vrsenjem mikroprogramske kontrole
- najveci broj masinskih operacija je tipa registar-u-registar sto rezultuje uproscenom upravljackom jedinicom. Takodje, ovakva arhitektura omogucuje optimizaciju upotrebe registara tako da argument kome se cesto pristupa ostaje u brzoj memoriji od koje su napravljeni registri
- upotreba relativno malog broja nacina adresiranja. Najveci broj instrukcija RISC procesora koristi registarsko adresiranje. Ostali kompleksniji nacini adresiranja se realizuju softverski. Ova osobina takodje ima uticaj na jednostavnost konstrukcije upravljacke jedinice cime se povecava brzina rada.
- upotreba jednostavnih formata instrukcija. U opstem slucaju se koristi samo nekoliko razlicitih formata instrukcija. Instrukcije su fiksne duzine i obicno su poravnate na granicu reci, sto znaci da ne prelaze granice stranica. Polja u instrukcijama su takodje fiksne duzine sto omogucuje istovremeno dekodiranje operacionog koda i pristup operandu instrukcije. Takodje, mali broj formata pojednostavljuje upravljacku jedinicu.

Najpoznatiji proizvodjaci RISC cipova su firme Motorola (88000, ..., PowerPC), Silicon Grpahics (MIPS R1000, R3000, R4000,..., R12000), Digital (Alpha), Hewlett Packard (PA-RISC 8200,...,8600), Sun Microsystems (Micro SPARC i ULTRA SPARC) i IBM (RS/6000 i PowerPC).

74. Koje su prednosti RISC u odnosu na CISC procesore?

1. *jednostavnija konstrukcija* - zbog manjeg broja instrukcija i jednostavnije strukture vreme potrebno za dizajniranje i uvodjenje takvog procesora u komercijalnu upotrebu je znatno krace
2. *Bolje performanse* - RISC cipovi poseduju znatno bolje performanse od CISC cipova koji rade na istim brzinama. Za RISC mikroprocesore je jednostavnije definisati prevodioce koji formiraju mnogo optimalniji kod.

75. Sta je brzina casovnika, a sta magistrala i koji tpovi magistrala postoje? Sta je sirina magistrale?

Brzina casovnika se odnosi na ritam sistemskog casovnika koji je obicno smesten na mikroprocesorski cip. Brzina casovnika se meri u megahercima (MHz), pri cemu je 1MHz ekvivalent milion pulseva u sekundi.

Magistrala je 'staza' ili veza kojom se prenose elektronski impulsi koji formiraju bitove u mikroprocesor i sistemsku jedinicu.

Sirina magistrale oznacavu kolicinu podataka koja istovremeno moze da se prenesu kroz magistralu.

Postoje tri osnovna tipa magistrala:

1. magistrala adresa - prenosi signale koji se koriste za odredjivanje adrese u primarnoj memoriji
2. magistrala podataka - prenosi podatke od/ka primarnoj memoriji
3. magistrala kontrole - prenosi signale koji kazu racunaru da 'cita' ili 'pise' podatke sa ili na odredjenu memorijsku adresu ulazni ili izlazni uredjaj

76. Koji faktori uticu na izbor metode za otkrivanje gresaka?

1. broj bitova sa greskom na komunikacionoj liniji - predstavlja verovatnocu u definisanom vremenskom intervalu da jedan bit ima gresku
2. tipa greske, tj. da li se greska javila na pojedinacnom bitu ili na grupi uzastopnih bitova

77. Navesti algoritme za otkrivanje gresaka u pristupu kontrole gresaka unazad.

Kontrola greske unazad - ne dozvoljava otklanjanje greske

1. *Kontrola parnosti* (*LRC*)- uz datu *n*-bitnu rec dodaje se jos jedan bit tako da ukupan broj binanih jedinica u rezultujucoj reci duzine *n*+1 bude paran (neparan). Otkrivaju se sve neparne greske.
2. *Kontrola parnosti u dve dimenzije* (*VRC*) - jedan bit parnosti se koristi za svaku jedinicu u bloku i dodatni skup bitova parnosti se racuna za svaki blok koji se prenosi. Otkrivaju se sve neparne greske, sve 2-bitne greske i najveci broj 4-bitnih gresaka.
3. *Kontrola zbira bloka* - formira se zbir svih jedinica u bloku i prenese zajedno sa porukom; dobijeni zbir se skrati na duzinu od 32 bita; primalac ponovo izracunava sumu zbira i poredi sa primljenom vrednoscu. Ova metoda ne moze da otkriva invertovane podatke, zamenu blokova bajtova, niti dodavanje i uklanjanje bajtova sa svim nulama.
4. *Ciklicka provera redundansi* (*CRC*)
   - *Kodiranje*
     i. Odabrati polinom generator $G(x)$
     ii. Izracunati $xkM(x)=G(x)$ gde je k stepen polinoma $G(x)$. Dobijeni ostatak je $R(x)$
     iii. Dodati koeficijente ostatka na kraj poruke i poruku proslediti primaocu
   - *Dekodiranje*
     i. Podeliti primljenu polinomijalnu kodnu rec sa $G(x)$
     ii. Ako je ostatak deljenja nula, nema gresaka pri prenosu

iii. Ako ostatak deljenja nije nula, greske pri prenosu postoje

78. Nabrojati nacine otkrivanja gresaka koji se koriste u pristupu kontrole greske unaped.

Kontrola greske unapred - dozvoljava otklanjanje greske

- SED-SEC (single error detection – single error correction)
- DED-SEC (double)
- TED-TEC (triple)

79. Navesti tipove gresaka koje se javljaju pri zapisu podataka u poluprovodnickoj memoriji.

Greske koje se javljaju pri zapisu podataka u poluprovodnickoj memoriji se mogu podeliti u dve kategorije:

1. *tvrde* (stalno prisutni defekti) - manifestuju se tako sto memorijska celija ne moze pouzdano da cuva podatke, vec bez ikakvog spoljasnjeg uzroka menja sadrzaj sa 0 na 1 i obratno
2. *mekane* (prolazne greske) - predstavljaju slucajne, nedestruktivne dogadjaje koji menjaju sadrzaj jedne ili vise memorijskih celija bez ostecenja same memorije

80. Sta je sindrom reci i na koji nacin se dobija?

Sindrom reci je neoznacen ceo broj koji se racuna po pravilu : $c_4c_3c_2c_1 \oplus c'_4c'_3c'_2c'_1$ i koji ima svojstva:

- ako je vrednost sindrom reci 0, 13, 14 ili 15 nije doslo do greske
- ako je vrednost sindrom reci 1, 2, 4 ili 8 (vrednost koja u binarnom zapisu ima samo jednu jedninicu) greska se javlja u jednom od kontrolnih bitova, dok je poruka u redu i ne zahteva nikakvu korekciju
- ako sindrom reci uzma vrednosti iz skupa {3, 5, 6, 7, 9, 10, 11, 12}, njena vrednost predstavlja poziciju na kojoj je doslo do greske i korekcija se sastoji u komplementiranju vrednosti odgovarajuceg bita

81. Nabrojati karakteristike i tipove masinskih instrukcija koje poznajete.

Skup različitih mašinskih instrukcija koje mogu da se izvršavaju na nekom procesoru se naziva *skup instrukcija procesora*. Instrukcije moraju da sadrže sledeće informacije:

- *operacioni kod instrukcije* – definiše operaciju koja će biti izvršena
- *referencu na operande instrukcije* – mogu biti ulazne vrednosti ili rezultati izvršavanja instrukcije
- *referencu na narednu instrukciju* - treba da se prenese u procesor po završetku izvršavanja tekuće instrukcije

Format instrukcija - sve instrukcije imaju slican format. Na pocetku se nalazi operacioni kod, zatim slede operandi instrukcije koji se mogu zadati na razlicite nacine. Skup instrukcija jednog procesora moze da sadrzi operacione kodove razlicitih duzina. Razlicite instructije mogu imati razlicit broj operanada razlicite duzine.

Tipovi instrukcija:

1. *aritmeticke* - deli se na instrukcije nad celim brojevima, instrukcije nad brojevima u pokretnom zarezu i instrukcije nad binarno kodiranim dekadnim brojevima
2. *logicke* - implementiraju logicke operacije Bulove algebre
3. *instrukcije za konverziju* - menjanje ili modifikacija tipa podatka
4. *instrukcije za prenos podataka* - koriste se za prenos podataka izmedju dve lokacije koje mogu da predstavljaju registre, vrh steka ili neku lokaciju u memoriji
5. *ulazno/izlazne instrukcije* - za komunikaciju i prenos podataka sa ulazno/izlaznih uredjaja
6. *kontrolne instrukcije* - skoro uvek su privilegovane i izvrsavaju se samo ukoliko je program u supervizorskom rezimu rada
7. *instrukcije za prenos kontrole* - menjaju sadrzaj brojaca instrukcija tako da on pokazuje na adresu u memoriji na kojoj se nalazi instrukcija koja se sledece izvrsava

Broj adresa u instrukciji

Jedan od nacina klasifikacije arhitektura procesora je podela racunara prema maksimalnom broju adresa koji moze da se pojavi u instrukciji. Minimalan broj adresa u instrukciji je 0. Takve instrukcije implicitno adresiraju svoje operande sto predstavlja veliko ogranicenje. Racunari mogu biti:

1. *jednoadresni* - instrukcije koje sadrze bar jednu adresu
2. *dvoadresni* – max broj adresa u instrukciji je dva, smanjuje se duzina programa
3. *troadresni* - instrukcija sa tri adrese

Nacini adresiranja

- *Registarsko adresiranje* - oznaka registra koji sadrzi operand je ukljucena u instrukciju
- *Direktno adresiranje* - stvarna adresa se direktno ukljucuje u instrukciju
- *Indirektno adresiranje* - adresa operanda se navodi indirektno, preko adrese lokacije na kojoj se nalazi adresa operanda
- *Indirektno registarsko adresiranje* - slican indirektnom sem sto se u adresnom polju instrukcije nalazi oznaka registra, a ne adresa lokacije u memoriji
- *Relativno adresiranje* - kao registar koji se referise implicitno koristi se brojac instrukcija. Za pocetnu adresu se uzima adresa tekuce instrukcije
- *Adresiranje sa baznim registrom i udaljenjem* - pocetna adresa se nalazi u posebnom registru procesora
- *Adresiranje sa indeks registrom* - pocetna adresa se nalazi u adresnom delu instrukcije a udaljenje u indeks registru
- *Adresiranje sa baznim registrom, indeks registrom i udaljenjem* - prosirenje adresiranja sa baznim registrom i udaljenjem
- *Neposredno adresiranje* - nacin adresiranja kod koga se operand nalazi u adresnom delu instrukcije
- *Implicitno adresiranje* - javlja se kada u procesoru postoje jedinstveni objekti odredjenog tipa tako da ih nije potrebno navoditi u instrukcijama. Drugi primer implicitnog adresiranja su instrukcije bez operanada. Operandi ovih instrukcija se nalaze na unapred poznatim lokacijama.

82. Masinski i asemblerski jezici.

Masinski jezik - skup instrukcija procesora koji poseduje semanticku interpretaciju u hardveru ili mikrokodu racunara. Sve masinske instrukcije i njihovi operandi predstavljaju nizove binarnih cifara.

Za programe koji su pisani sa menmonickim oznakama instrukcija uz koriscenje relativnog adresiranja se kaze da su pisani u *asemblerskom jeziku*, a za njihovo prevodjenje se koristi asembler. Programi se sa asemblerskog jezika prevode na masinski pomocu relativnih adresa.

Pozivi potprograma:

1. odredjivanje adrese potprograma koga treba pozvati
2. odredjivanje i pripremu argumenata potprograma koji se poziva
3. cuvanje svih relevantnih parametara trenutnog stanja programa koji se izvrsava
4. specificiranje adrese na koju se vrsi povratak iz potprograma
5. izvrsavanje instrukcije skoka kojom se izvrsavanje prenosi na naredbu potprograma
6. izvrsavanje potprograma
7. izvrsavanje naredbe povratka na zahtevanu adresu u pozivajucem programu

Argument i rezultati potprograma se prenose na tri nacina:

1. preko opstih registara
2. upisivanjem u odredjenu zonu memorije cija adresa se prenosi preko opstih registara
3. putem steka, zajedno sa adresom povrataka

83. Smestanje podataka u memoriji.

Redosled smeštanja bajtova - Pri zapisu visebajtnih skalarnih vrednosti u memoriju moguca su dva zapisa:

1. bajt najvece težine se upisuje u bajt sa najmanjom adresom u memoriji - sa levog kraja
2. bajt najmanje težine se upisuje u bajt sa najmanjom adresom u memoriji- sa desnog kraja

U oba slucaja se preostale vrednosti upisuju redom pocevsi od zapisane vrednosti ka drugom kraju broja. Prednosti jednog nacina zapisa su mane drugog i obrnuto. Prednosti i mane svakog od nacina zapisa su:

1. *sa desnog kraja* – aritmeticke operacije koje zahtevaju vecu preciznost se jednostavnije obavljaju jer se lakše odredjuje bajt najmanje težine. Konverzija 32-bitne u 16-bitnu celobrojnu adresu je jednostavnija jer ne zahteva dodatne operacije za izdvajanje bajtova najmanje težine.
2. *sa levog kraja* – omogucuje brže poredjenje karakter niski koje su zapisane tako da po poravnanju odgovaraju celobrojnim vrednostima. Celobrojne vrednosti i karakter niske se zapisuju u istom redosledu. Pri pregledu sadržaja memorije u slucaju otkrivanja grešaka sve vrednosti se štampaju s leva na desno.

Redosled smeštaja bitova - Ne postoji jedinstveni nacin smeštanja bitova unutar bajta. U zavisnosti od proizvodjaca racunara brojanje (i oznacavanje) bitova unutar bajta može biti:

1. sa desnog kraja
2. sa levog kraja

84. Unutrasnja memorija.

Karakteristike memorije:

- *stalnost zapisa* - memorije sa stalnim zapisom i privremenim zapisom
- *fizicki tip medijuma* - podela na osnovu tehnologije koja se koristi za zapis informacije
- *kapacitet* - kolicina informacija koju memorija moze da sadrzi. Izrazava se u bajtovima ili recima. Duzina reci zavisi od tipa procesora
- *jedinica prenosa* - jednaka je broju bitova koji mogu istovremeno da se procitaju iz memorije ili na nju upisu
- *adresivost* - svojstvo memorije da joj se moze pristupiti uz pomoc adrese. Memorije mogu biti: adresive, poluadresive i neadresive.
- *cena* - poredi se na osnovu ulozenog novca za odredjeni kapacitet

• *moguc nacin pristupa* - postoje 4 nacina pristupa podacima u memoriji:

- sekvencijalni pristup - podaci su organizovani u slogove koji se u memoriji medjusobno razdvajaju kontrolnim informacijama koje se koriste pri citanju podataka

- direktan pristup - zavisnost izmedju adrese sloga i negove fizicke lokacije na medijumu zato sto se na osnovu adrese direktno pristupa lokaciji gde je smesten slog ili njegovoj okolini.

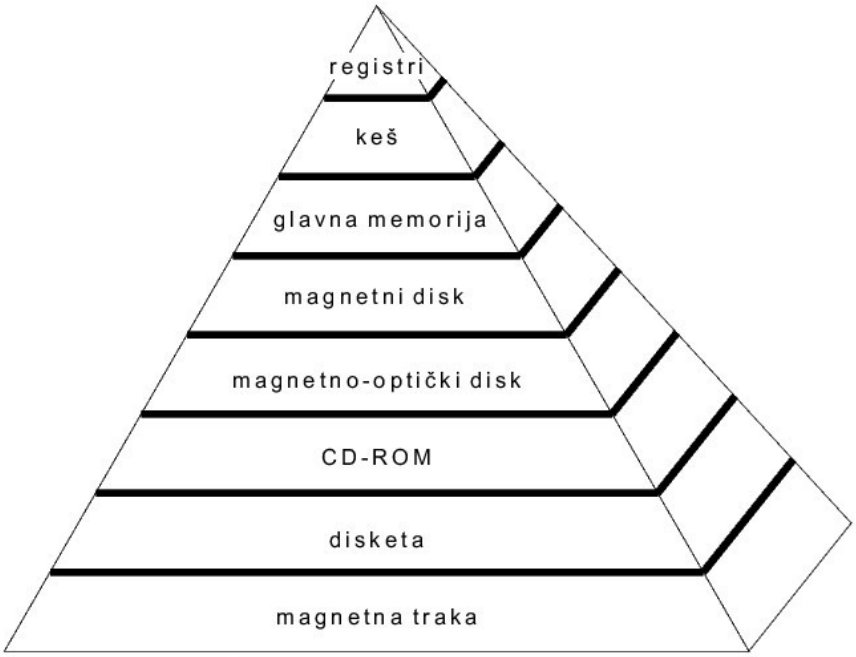
- slucajni pristup - svaka adresibilna lokacija poseduje jedinstven adresni mehanizam ugradjen u memorijski sklop

- asocijativni pristup – omogucuje poredjenja izmedju posebne maske i vrednosti odredjenih pozicija bitova u reci, pri cemu se poredjenje vrsi istovremeno za sve reci memorije

• *performanse* - najznacajnija karekteristika memorije. Odredjuju je sledeci parametri: vreme pristupa, vreme memorijskog ciklusa, brzina prenosa

• *mogucnost promene sadrzaja* - memorije ciji se sadrzaj moze promeniti zovu se upisno-citajuce, a memorije ciji se sadrzaj ne moze promeniti nazivaju se samocitajuce

Hijerahija memorija:



85. Glavna memorija:

Pravi se iskljucivo od poluprovodnickih elemenata. Osnovni element poluprovodnicke memorije je memorijska celija i ona ima sledece osobine:

- poseduje dva stabilna stanja koja se koriste da bi se predstavile 0 i 1
- njen sadrzaj se upisuje najmanje jednom
- moze da se cita proizvoljan broj puta

Tipovi glavne memorije:

- *samocitajuca memorija (ROM)* - njen sadrzaj je stalan i ne moze se promeniti. Koristi se za mikroprogramiranje, cuvanje sistemskih podataka

- *programibilni ROM (PROM)* - isti su kao i ROM cipovi samo sto su inicijalno prazni. Sadrzaj PROM-a se upisuje pomocu elektricnih impulsa. Kad se instrukcije jednom upisu PROM postaje ROM i vise ne moze da se brise.
- *izbrisivi PROM (EPROM)* - funkcionise slicno kao i PROM sem sto njegov sdrzaj moze da se izbrise ultraljubicastim zracima pa da se zameni novim sadrzajem.
- *fles memorija* - poluprovodnicka memorija koja se nalazi izmedju EPROM-a i EEPROM-a. Za brisanje kompletnog sadrzaja potrebno je par sekundi ali moguce je izbrisati i samo pojedinacne blokove memorije.
- *RAM* - sadrzaj moze da se cita i ucitava proizvoljan broj puta. Citanje i upis podataka se vrsi pomocu elektricnih signala. RAM gubi svoj sadrzaj po prestanku napajanja. Postoje dve vrste RAM-a:
  - s*taticki RAM (SRAM)* - koristi flip-flop kombinatorne mreze
  - d*inamicki RAM (DRAM)* - pravi se od celija koje cuvaju vrednosti kao naboje u kondenzatorima. Prisustvo, odnosno odsustvo elektricnih naboja se interpretira kao 1 odnosno 0

86. Kes memorija.

Svrha kes memorije je da premosti razlike izmedju brzina procesora i glavne memorije. Sastoji se od relativno male kolicine brze staticke memorije koja omogucava asocijativan pristup podacima. *Bafer memorija* je slicna kes memoriji ali dok kes memorija sadrzi kopije podataka, bufer sadrzi originale podataka koji ne postoje nigde drugde u racunaru.
Kes na procesorskom cipu predstavlja kes na prvom nivou (L1 kes), a spoljasnji kes predstavlja kes na drugom nivou (L2 kes). L1 je manja i brza od L2 kes memorije. U danasnje vreme i L2 je ugradjen u procesor, a na njegovo mesto dolazi L3 kes memorija koja je integrisana na maticnoj ploci i ima istu funkciju koju je imala L2 memorija.

87. Spoljasnja memorija.

Sadrzi podatke koji se ne koriste aktivno u nekom trenutku. Njen sadrzaj je stalan i ne gubi se prestankom napajanja. Sporija je od unutrasnje memorije ali ima veci kapacitet.
Magnetni diskovi
- sastoje se od kruznih ploca
- ploce su od metala ili plastike I prevucene supstancom koja poseduje magnetna svojstva
- podaci se upisuju preko posebnog provodnika sa navojnim kalemom koji se naziva upisno-citajuca glava

Formatiranje - proces upisa staza, sektora i kontrolnih podataka na disk. Postoje dve vrste formatiranja diska:
- formatiranje niskog nivoa - upisuje sektore na disk. Obicno se obavlja u fabrici kao deo proizvodnog procesa i omogucuje proizvodjacu da testira disk i inicijalizuje preslikavanje izmedju brojeva logickih blokova i sektora diska koji nemaju oštecenje. Moguce je izabrati razlicite velicine sektora (256, 512 ili 1024 bajta). Sto je sektor veci to je broj slogova na stazi manji pa je samim tim ima vise mesta za podatke
- logicko formatiranje - operativni sistem upisuje strukture podataka koji su mu potrebni za rad sa datotekama. Postoje i specijalni programi koji imaju mogucnost da koriste particiju diska kao veliki sekvencijalni niz logickih blokova.

Karakteristike diska:
- broj upisano/citajucih glavama
- jednostrani/dvostrani
- fiksni/imenjivi
- vreme pristupa disku – vreme trazenja, rotaciono kasnjenje
- opseg diska

RAID tehnologija – podrzava veliki broj jedinica diskova sa kontrolerskim cipom I ugradjenim specijalizovanim softverom. Istovremeno razmesta podatke preko vise paralelnih puteva I na ovaj nacin dobija krace vreme odziva.

88. Opticki diskovi.

Uvedeni su 1983.godine kao medijum za zapis muzike. Podaci se citaju tako sto se ploca diska rotira ispod mehanizma za citanje. Za cuvanje i citanje podataka koriste se dva mehanizma:
- *konstantna ugaona brzina* (CAV) - kada disk ploca rotira konstantnom brzinom onda je potrebno vise vremena da se dodje do podataka zapisanih na obodu diska nego u centru. Ova razlika u brzinama se nadoknadjuje povecanjem prostora izmedju bitova na delovima diska koji su blizi obodu sto dovodi do mogucnosti da se informacije citaju istom brzinom gde god da se nalaze.
- *konstantna linearna brzina* (CLV) - podaci se zapisuju u segmente jednake velicine. Disk rotira sporije kada se citaju podaci blizi obodu. Kapacitet staze i rotaciono kasnjenje se povecavaju kako je staza bliza obodu diska zbog cega se citanje obavlja konstantnom linearnom brzinom. Umesto vise koncentricnih staza moguca je i samo jedna spilarna staza.

Opticki diskovi se dele u 3 grupe: diskove koji su nasnimljeni i ciji sadrzaj ne moze da se menja (DO-ROM, DVD-ROM, CD-DA...), diskove na koje korisnik moze samo jednom da upise sadrzaj posle cega on ne moze da se menja (CD-R, BD-R i WORM...), diskove ciji sadrzaj moze da se pise/brise vise puta (CD-RW, DVD-RW).

89. Ulazno-izlazni podsistem. U/I moduli. Tehnike izvršavanja U/I operacija.

Ulazno-izlazni podsistem se sastoji od ulaznih i izlaznih uređaja čija je osnovna uloga efikasnije korisćenje računarskog sistema, prvenstveno tako sto omogućavaju upravljanje samim računarskim sistemom.

Ulazni uređaji prikupljaju podatke iz okoline i prevode ih u oblik pogodan za obradu u računarskom sistemu.

Izlazni uređaji preuzimaju podatke dobijene obradom i prosleđuju ih na dalju obradu i-ili prikazuju u obliku upotrebljivom za ljude.

Glavne funkcije su:

1. kontrola i usklađivanje saobraćaja - vrši koordinaciju saobraćaja između periferala i internih resursa
2. komunikacija sa procesorom - prihvata komande i adrese perifernih uređaja od CPU-a, interpretira ih i po potrebi izvršava
3. komunikacija sa uređajima – slanje i primanje podataka, kontrolnih informacija i informacija o statusu uređaja
4. prihvatanje podataka – kako je brzina perifernih uređaja relativno mala u odnosu na procesor, U/I modul prihvata i baferiše podatke iz perifernih uređaja. Procesor i memorija podatke prihvataju iz U/I bafera
5. otkrivanje gresaka – zadužen za otkrivanje i korekciju grešaka ili prosleđivanje odgovarajuće informacije CPU-u kada grešku ne moze da otkloni.

Tehnike izvršavanja U/I operacija

- *programirani U/I*
    1. kontrolna komanda
    2. test komanda
    3. komanda za čitanje
    4. komanda za pisanje
- *prekidima upravljan U/I* – primenjuje se na skoro svim racunarskim sistemima
- *DMA* – tehnika kojom se eliminise potreba za intervencijom CPU-a pri prenosu podataka izmedju memorije i U/I modula. Zahteva dodatni modul tj. DMA kontroler
- *U/I procesori i kanali* - složeni U/I moduli koji mogu da obave U/I operacije bez nadzora CPU-a. Kanal uključuje U/I procesor koji može da izvršava U/I instrukcije u memoriji. U/I modul koji ima iste mogucnosti kao i kanal i uz to poseduje sopstvenu memoriju se naziva periferni procesor.

Kod licnih racunara i radnih stanica, ako uredjaj komunicira sa racunarom preko veze u samo jednoj tacki, ta tacka prikljucka se naziva *port*. U/I port se najcesce sastoji od cetiri registra koji se nazivaju registri statusa, kontrole, primljenih i poslatih podataka.