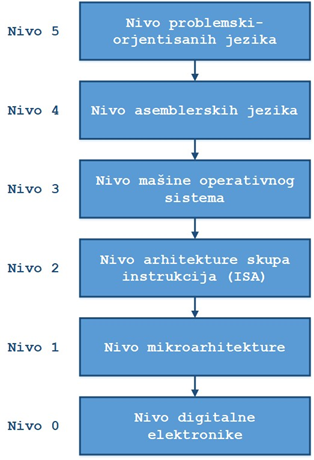
**1. Opisati ukratko savremeni višenivoovski računarski sistem. Nacrtati dijagram.**

Većina savremenih računara sastoji se od dva ili više nivoa. Na slici je prikazana jedna takva mašina, koja ima šest nivoa. Na nultom nivou nalazi se pravi hardver mašine. Elektronska kola izvršavaju mašinski programski jezik napisan na nivou L1. Čak se može reći da postoji i još jedan nivo ispod nivoa 0, koji se naziva nivoom uređaja, na kojem projektant radi sa pojedinačnim tranzistorima.

**2. Opisati nivo digitalne logike I nivo mikroarhitekture kod savremenog nišenivoovskog računarskog Sistema.**

*- Nivo digitalne logike*

Iako se prave od analognih komponenti (tranzistora), digitalna kola se mogu modelovati kao digitalni uređaji. Svako kolo ima jedan ili više digitalnih ulaza (0 ili 1) i računa izlaz iz kola kao jednostavna logička funkcija, ILI (en. OR) ili I (en. AND). Svako digitalno kolo sastoji se od nekolicine tranzistora. Takođe, ova kola mogu poslužiti za skladištenje jednog bita memorije (skladištenje 0 ili 1).

Ove jednobitne memorije se mogu grupisati grupe od npr. 16, 32 ili 64 da formiraju registar. Svaki registar može skladištiti binarni broj do neke vrednosti. Digitalna kola se takođe mogu kombinovati da formiraju i samo jezgro računanja u računaru.

*- Nivo mikroarhitekture*

Na nivo mikroarhitekture imamo skup registara (npr. skup od 8 ili od 32 registra) koji formiraju lokalnu memoriju ili komplikovanije kolo koje se naziva aritmetičko-logička jedinica (en. Arithmetic Logic Unit, ALU), koja služi za jednostavne aritmetičke operacije.

**3. Opisati ISA nivo I nivo mašine operativnog Sistema kod savremenog nišenivoovskog računarskog Sistema.**

*- Nivo arhitekture skupa instrukcije skupa (ISA)*

ISA (Instruction Set Architecture) predstavlja nivo apstrakcije arhitekture računara koji definiše skup instrukcija koje je procesor sposoban da izvršava.

Svaki proizvođač računarske opreme ovog tipa objavljuje i uputstvo za ISA sloj, bez nižih slojeva.

Kada proizvođač opisuje skup instrukcija računara, zapravo se opisuju instrukcije koje se izvršavaju interpretiranjem mikroprograma ili samog hardvera.

*- Nivo masine operativnog sistema*

Nivo mašine operativnog sistema predstavlja najniži nivo apstrakcije u organizaciji računara i operativnih sistema.

Na ovom nivou se bave konkretnim hardverom i pružaju osnovne funkcionalnosti za upravljanje hardverskim resursima i izvršavanje programa.

Npr.: Upravljanjem procesorom, memorijom, I/O operacijama, datotekama…

Neke instrukcije na nivou 3 se interpretiraju od strane operativnog sistema, a neke direktno od strane mikroprograma ili hardvera. Najniža tri nivoa nisu za „svakodnevne“ programere. Zapravo, ovo su nivoi koji se primarno koriste za izvršenje interpretera i prevodioca za podršku viših nivoa.

**4. Opisati nivo asemblerskih jezika i nivo problemski-orjentisanih jezika kod savremenog nišenivoovskog računarskog Sistema.**

- Nivo asemblerskih jezika

Nivo asemblerskog jezika je nivo apstrakcije između mašinskog jezika i višeg nivoa programiranja. Asemblerski jezik je zapravo tekstualna reprezentacija mašinskog jezika koja je čitljiva ljudima.

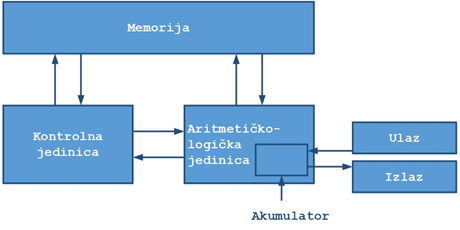
Nivo asemblerskih jezika predstavlja simboličku formu nekih od jezika na nižem nivou. Ovaj nivo pruža mogućnost da ljudi pišu programe na nivoima 1, 2 i 3 u formi koja nije tako neprijatna kao mašinski jezik tih virtuelnih mašina.

- Nivo problemski-orjentisanih jezika

Peti nivo najčešće se sastoji od programskih jezika koji služe za rešavanje konkretnih problema.

Ovi programski jezici se nazivaju viši programski jezici (en. high-level languages), i to su jezici kao što su C, C++, Java, Python, JavaScript i sl.

Programi koji su napisani u ovim jezicima često su prevedeni na nivo 3 ili nivo 4 koristeći prevodilac koji se naziva kompajler (en. compiler), ali postoje i interpretirani jezici.

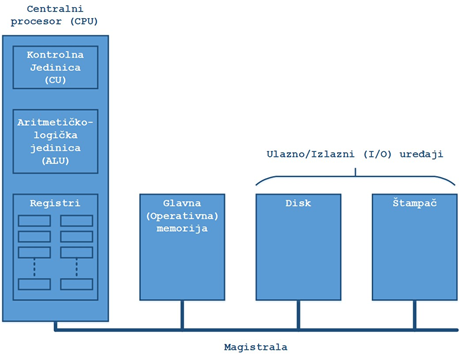
**5. Opisati nov Neumann mašinu. Nacrtati dijagram.**

Arhitektura računarskih mašina razvijala se po principima koje je postavio Džon von Njuman tako da je arhitektura računara nazvana von Njuman mašina. Von Neumann mašina sadrži pet delova: memoriju, ALU jedinicu, kontrolnu jedinicu, i ulazno-izlazni podsistem.

U savremenim računarskim sistemima ALU i kontrolna jedina su kombinovani u jedno integrisano kolo nazvano centralni processor.

**6. Šta predstavlja RADIX brojni sistem? Opisati opšti RADIX sistem.**

RADIX brojni sistem je matematički sistem za predstavljanje brojeva koji se zasniva na određenom broju baza, poznatom i kao radix ili osnova. U opštem RADIX sistemu, broj number se može predstaviti kao suma proizvoda cifara d (sa nulom) i baze sistema b na stepenu koji ide od -k do n.

**7. Opisati ukratko sastavne delove centralnog procesora. Nacrtati dijagram.**

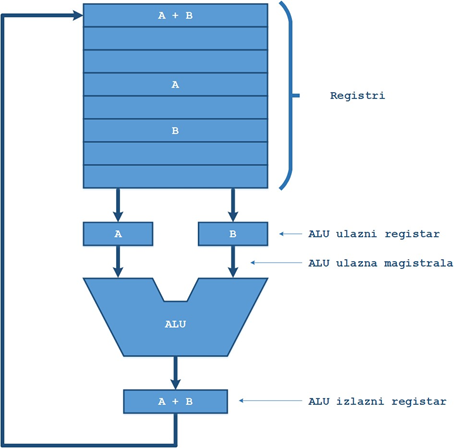
CPU se sastoji od nekoliko različitih delova.

Kontrolna jedinica je odgovorna za preuzimanje instrukcija iz glavne memorije i određivanje tipa instrukcije.

Aritmetičko-logička jedinica obavlja operacije kao što su sabiranje, pomeranje, kao i logičke operacije kao što su AND ili OR.

Registri su male memorijske jedinice velike brzine koji se koriste za privremeno

skladištenje rezultata, ili za skladištenje određenih kontrolnih informacija. Svaki ima određenu veličinu tj. kapacitet, ali i funkciju. Najvažniji registar je Programski brojač koji ukazuje na sledeću instrukciju koja se preuzima za izvršenje.

**8. Opisati tzv. putanju podataka (datapath) kod von Neumann mašine. Nacrtati dijagram putanje podataka.**

Na slici je prikazana unutrašnja organizacija dela jednostavnog von Njumanov-og CPU-a, sa fokusom na tzv. putanji podataka koja se sastoji se od registara (obično 1 do 32), ALU i nekoliko magistrala koje sve to povezuju.

U ovoj arhitekturi postoje dva ulazna registra koji su direktno povezani na ALU, označena sa A i B na slici. U ovim registrima smešteni su operandi dok ALU izvršava neko izračunavanje.

Podatak iz registra se može upisati ili uskladištiti u operativnu memoriju.

**9. Opisati sedam koraka ivršavanje instrukcija od strane CPU-a.**

CPU izvršava instrukcije u nekoliko malih koraka, koje su u opštem slučaju sledeće:

1. Pribavi sledeću instrukciju iz memorije i smesti je u instrukcijski registar

2. Promeni programski brojač da pokazuje na sledeću instrukciju

3. Odredi tip pribavljene instrukcije

4. Ukoliko instrukcija koristi podatke iz memorije, pronađi taj deo memorije

5. Pribavi deo memorije, ukoliko je potrebno, i smestiti u CPU registar

6. Izvrši instrukciju

7. Vrati se na korak 1 i krenuti sa izvršenjem sledeće instrukcije

**10. Opisati fetch-decode-execute ciklus, prefetch i pipeline ciklus izvršenja instrukcija. Nacrtati dijagram za pipeline ciklus izvršenja instrukcija.**

CPU izvršava instrukcije u nekoliko malih koraka, koje su u opštem slučaju sledeće:

1. Pribavi sledeću instrukciju iz memorije i smesti je u instrukcijski registar,

2. Promeni programski brojač da pokazuje na sledeću instrukciju,

3. Odredi tip pribavljene instrukcije,

4. Ukoliko instrukcija koristi podatke iz memorije, pronađi taj deo memorije,

5. Pribavi deo memorije, ukoliko je potrebno, i smestiti u CPU registar,

6. Izvrši instrukciju,

7. Vrati se na korak 1 i krenuti sa izvršenjem sledeće instrukcije.

Ovaj niz koraka naziva se pribavi-dekoduj-izvrši (en. fetch-decode-execute). Svi procesori rade na ovim principima.

Koncept protočnog izvršenja instrukcija (en. **pipelining**) radi na principu podela instrukcijskog ciklusa na više delova, od kojih svaka je faza (en. stage) dodeljena jedinstvenom hardveru, od kojih svaki mogu raditi paralelno.

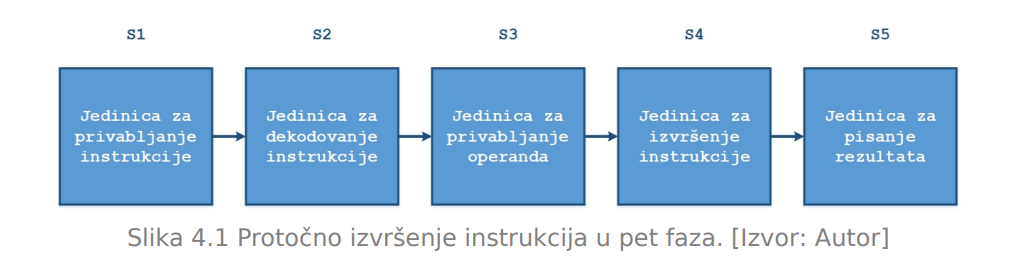
Faza 1 preuzima instrukciju iz memorije i smešta je u bafer dok ne bude potrebna.

Faza 2 dekoduje instrukciju, određujući njen tip i operande koje su potrebni toj instrukciji.

Faza 3 pronalazi i preuzima operande, bilo iz registara ili iz operativne memorije.

Faza 4 zapravo obavlja posao izvršavanja instrukcije kroz prethodno objašnjenu putanju podataka. Konačno,

faza 5 upisuje rezultat nazad u odgovarajući registar.



**11. Opisati razlike između CISC i RISC računarskih sistema.**

RISC (Računar sa smanjenim skupom instrukcija) je arhitektura procesora koja se karakteriše jednostavnim skupom instrukcija. Osnovna ideja RISC arhitekture je da se koriste jednostavne i osnovne instrukcije koje se mogu izvršiti brzo.

CISC (Računar sa složenim skupom instrukcija) je arhitektura procesora koja se karakteriše složenim skupom instrukcija. CISC procesori imaju širok spektar instrukcija, uključujući kompleksne operacije koje obavljaju više koraka u jednoj instrukciji.

**12. Koje si razlike između analognog i digitalnog signala, a koje između digitalnog i binarnog digitalnog signala?**

Analogni signal je signal koji u proizvoljnom trenutku može imati bilo koju vrednost u okviru datog opsega.

Digitalni signal u svakom vremenskom trenutku može imati jednu od nekoliko unapred određenih diskretnih vrednosti.

Binarni digitalni signal tokom vremena može imati samo dve vrednosti: visoku vrednost, koja se naziva logičkom jedinicom i nisku vrednost, koja se naziva logičkom nulom.

**13. Opisati de Morganove zakone kod logičkih funkcija.**

De Morganovi zakoni govore da invertovanjem izlaza bilo kojeg logičkog kola rezultovaće suprotnom tipu funkcije (AND u OR i obrnuto) sa pojedinačno invertovanim ulazima.

**14. Dati primer standardne disjunktivne i standardne konjunktivne forme logičke funkcije sa četiri ulaza.**

DNF - (A' \* B' \* C' \* D') + (A' \* B \* C' \* D) + (A \* B' \* C \* D') + (A \* B \* C \* D)

KNF - (A + B + C + D) \* (A + B + C' + D') \* (A' + B + C' + D) \* (A' + B' + C + D')

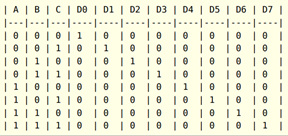
**15. Dati primer potpune standardne disjunktivne i porpune standardne konjunktivne forme logičke funkcije sa tri ulaza.**

PDNF - (A' \* B' \* C') + (A' \* B' \* C) + (A' \* B \* C') + (A' \* B \* C) + (A \* B' \* C') + (A \* B' \* C) + (A \* B \* C')

PKNF - (A + B + C) \* (A + B + C') \* (A + B' + C) \* (A + B' + C') \* (A' + B + C) \* (A' + B + C') \* (A' + B' + C')

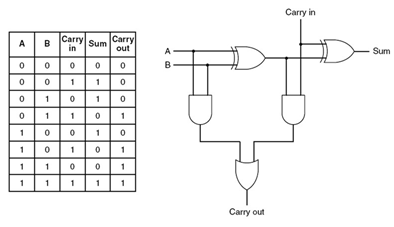
**16. Opisati način rada dekodera. Napisati tablicu istinitosti dekodera 3/8.**

Dekoder predstavlja logičko kolo koje uzima n-bitni broj kao ulazni signal I postavlja tačno jedna od 2n izlaza koji postavlja na jedinicu.

Dekoder se koristi, između ostalog, pri selektovanju tačno određenog čipa na RAM memoriji.

**17. Opisati način rada multipleksera. Realizovati multiplekser 4-1 kroz logička kola.**

Multiplekser predstavlja logičko kolo sa 2n ulaza, jednim izlazom I n kontrolnih ulaza koji selektuju određeni izlaz. Multiplekser se često koristi kao konvertor paralelnog prenosa u serijski prenos podataka.

**18. Opisati rad potpunog sabirača. Napisati tablicu istinitosti i nacrtati realizaciju kroz logička kola.**

Sabirač predstavlja aritmetičko kolo koje sabira dva bita, daje na izlazu sumu i, ukoliko postoji, i prenet bit. Ukoliko sabirač ima poseban ulaz za preneti bit iz drugog sabirača, onda je u pitanju potpuni sabirač.

**19. Šta predstavlja clock signal kod logičkih kola?**

Clock signal je impuls za sinhronizaciju koji kontroliše promene internih stanja u diskretnim vremenskim trenucima kod sinhronih kola.

Kod sinhronih kola interna stanja se menjaju u diskretnim vremenskim trenucima pod kontrolom impulsa za sinhronizaciju koga nazivamo taktni signal ili clock signal.

**20. U čemu je razlika između kombinatornih i sekvencijalnih logičkih kola?**

Kombinatorna logika se odnosi na dizajn i analizu logičkih kola u kojima izlazna vrednost zavisi samo od trenutnih ulaznih vrednosti. Sekvencijalna logika se odnosi na dizajn i analizu logičkih kola koja imaju unutrašnje stanje i pamte prethodne ulazne vrednosti.

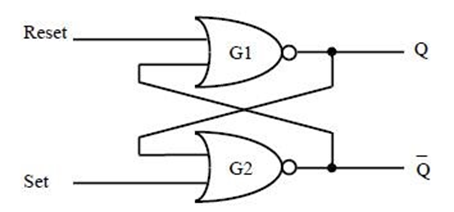
**21. Koje su prednosti, a koje su nedostaci sekvencijalnih kola u odnosu na kombinatorna?**

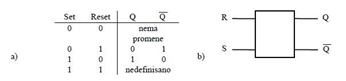
- Memorija: Sekvencijalna kola imaju sposobnost pamćenja prethodnih ulaznih vrijednosti i unutrašnjeg stanja.

- Složenija logika: Sekvencijalna kola mogu obavljati složenije logičke operacije od kombinatornih kola.

- Veća složenost: Sekvencijalna kola su složenija od kombinatornih kola jer moraju skladistiti logiku za pamćenje i upravljanje unutrašnjim stanjem.

- Vreme odziva

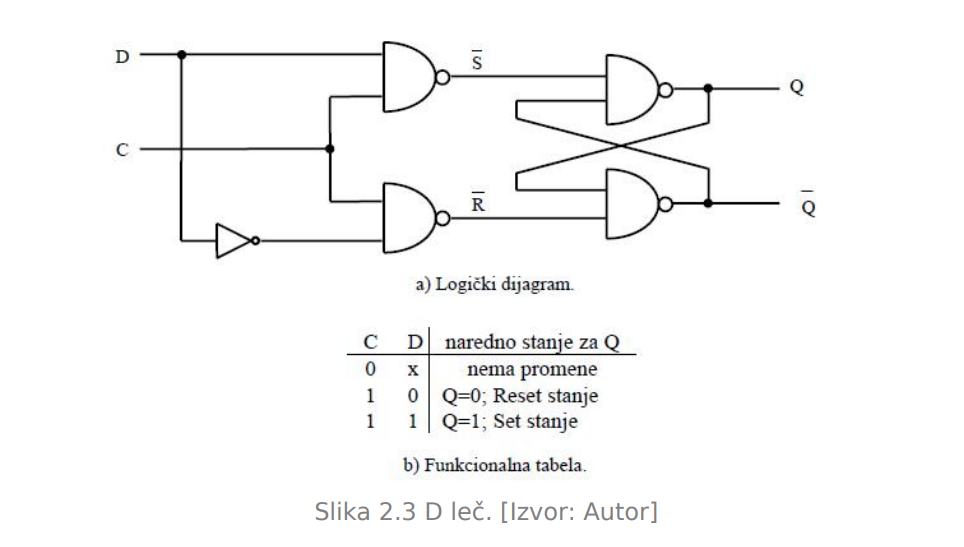
**22. Ukratko opisati rad SR-Latch kola. Nacrtati realizaciju kroz logička kola i tablicu istinitosti.**



SR leč je najjednostavniji memorijski element koji se koristi za projektovanje digitalnih sistema. SR leč čine dva unakrsno spregnuta NOR kola.

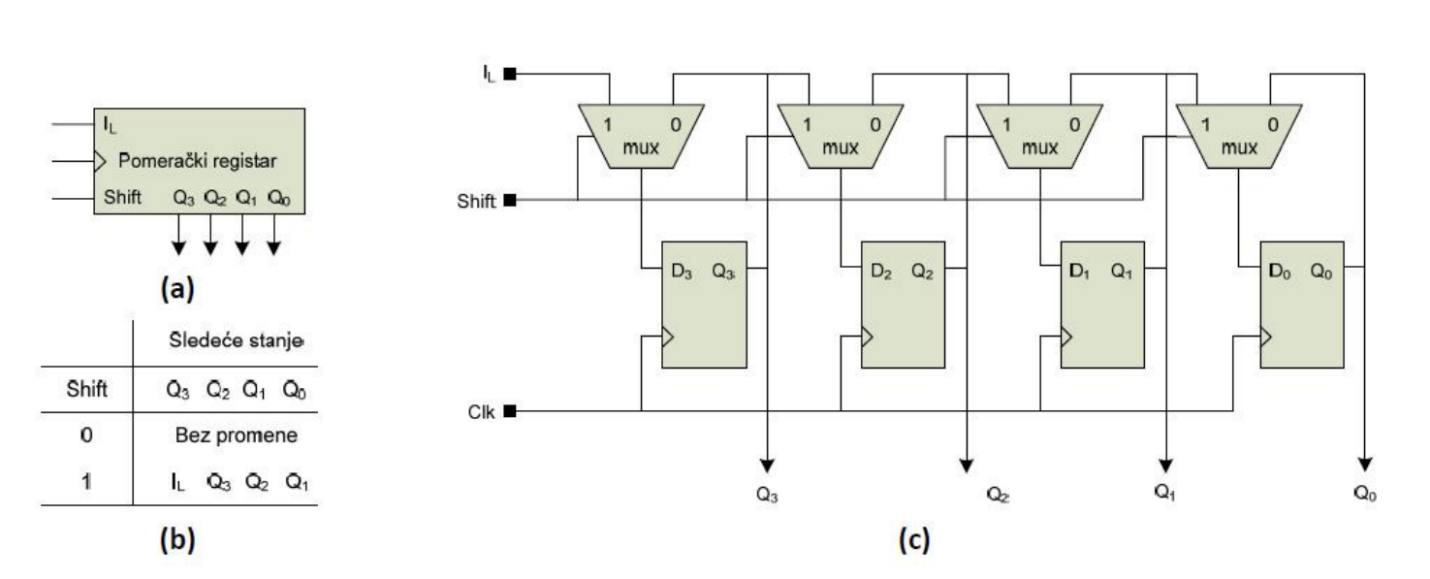
**23. Ukratko opisati rad D-Latch kola. Nacrtati realizaciju kroz logička kola i tablicu istinitosti.**

D-Latch (ili Data Latch) je jednostavna sekvencijalna logička kola koja se koriste za skladištenje i prenošenje jednog bita podataka. Leč ima samo dva ulaza: D (Data - podaci) i C (Control - upravljački).

Jedan od načina da se eliminiše neželjeno nedefinisano stanje kod SR leča je da se obezbedi da ulazi S i R ne budu nikada istovremeno jednaki 1. Ovo se izvodi kod D leča kako je to prikazano na slici 3.

**24. Ukratko opisati rad pomeračkog registra sa dozvolom. Nacrtati realizaciju 8-bitnog pomeračkog registra sa dozvolom.**

Pomerački registaa omogućava pomeranje upisanog sadržaja za jednu bit‐poziciju.



**25. Opisati rad stack memorije, kao i operacije nad stack memorijom.**

Stek je memorijska struktura koja se često koristi kako u softveru tako i u hardveru. Po definiciji, stek je memorija sa ograničenim pristupom. Podacima zapamćenim u steku pristupa se isključivo preko jedne lokacije: vrh steka.

PUSH – Podatak se smešta na vrh steka i pri tome se svi prethodno upisani podaci spuštaju za jednu poziciju niz stek.

POP - Podatak se sklanja sa vrha steka i pri tome se svi ostali podaci podižu za jednu poziciju naviše uz stek. Na

**26. Opisati ulogu upravljačke jedinice centralnog procesora.**

Upravljačka jedinica centralnog procesora igra ključnu ulogu u izvršavanju instrukcija i upravljanju radom procesora. Glavna uloga upravljačke jedinice je koordinacija svih operacija i resursa unutar procesora kako bi se obezbedilo pravilno izvršavanje programa.

**27. Opisati pojam mikrooperacije. Od čega se sastoje mikrooperacije?**

Operacije unutar mikroprocesora (često se nazivaju mikrooperacijama) sinhronizovane su generatorom takta. To je najmanja jedinica operacije koju procesor može izvršiti i obično je povezana s upravljanjem pojedinim elementima unutar procesora.

**28. Šta predstavljaju magistrale i koje su tri vrste magistrala?**

Magistrale u računarstvu predstavljaju puteve ili kanale kojima se podaci, komande i kontrolne informacije krecu.

Adresna magistrala - specificira jednu memorijsku lokaciju ili U/I port koji komunicira sa CPU-om

Magistrala za podatke - koristi se za prenos podataka

Upravljačka magistrala - upravljački signali pomoću kojih se sinhronizuje prenos podataka

**29. Kako se dele magistrale po nameni, a kako od tipa resursa?**

Deljenje magistrala po nameni:

- Sistemski magistrala (System Bus): Ova magistrala se koristi za komunikaciju između glavnih komponenti računarskog sistema, uključujući procesor, memoriju i ulazno-izlazne uređaje.

- Lokalna magistrala (Local Bus): Ova magistrala se koristi za komunikaciju unutar određene podskupine komponenti računarskog sistema, kao što su procesorska jezgra i keš memorija.

Deljenje magistrala po tipu resursa:

- Adresna magistrala, Magistrala za podatke, Upravljacka magistrala

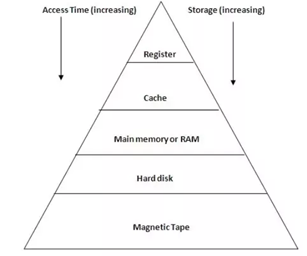
**30. Opisati razlike između sinhrone i asinhtone magistrale.**

Sinhron prenos - svi događaji se dešavaju u fiksnim vremenskim intervalima. Sve komponente, kod ispravnog prenosa, mora da rade sa brzinom koja je diktirana od strane glavnog takta.

Asinhron prenos - signali u sekvenci se generišu u proizvoljnim vremenskim intervalima. Kod asinhronih magistrala sinhronizacija prenosa nije diktirana glavnim taktom. Metod asinhronog prenosa podataka se više primenjuje kod magistrala mikroračunarskih sistema.

**31. Šta predstavlja ciklus na magistrali?**

Ciklus na magistrali je jedna kompletna sekvencija radnih impulsa koja se koristi za prenos podataka između uređaja na magistrali. To je osnovna jedinica vremena koja se koristi za organizaciju prenosa podataka na magistrali.

**32. Šta predstavlja keš memorija i gde se nalazi u hijerarhiji memorijskog sistema?**

Hijerarhijski koncept memorije podrazumeva nivo brze memorije ili keš memorije kojoj procesor pristupa u jednom taktu, i nivoe sporije memorije koja je znatno većeg kapaciteta.

Procesor pristupa informacijama, naredbama i podacima, koje su samo u keš memoriji.

Ukoliko informacija nije u keš memoriji, potrebno ju je prebaciti iz sporije u brzu memoriju.

**33. Šta predstavlja princip lokaliteta kod keš memorije? Opisati dva tipa lokaliteta kod keš memorije.**

Princip lokaliteta jeste činjenica da program u nekom vremenskom intervalu pristupa relativno uskom memorijskom području.

Prostorni lokalitet - ukoliko se u jednom trenutku pristupilo jednoj memorijskoj lokaciji da je velika verovatnoća da će u sledećim trenucima pristupati njoj susednim lokacijama.

Vremenski lokalitet - ukoliko se pristupi jednoj memorijskoj lokaciji da je velika verovatnoća ponovnog pristupa istoj unutar kratkog vremenskog intervala.

**34. Šta predstavljaju funkcije mapiranja kod keš memorije?**

Funkcije mapiranja kod keš memorije određuju način na koji se blokovi podataka smeštaju i pristupaju u keš memoriji.

- Direktno mapiranje - Kod direktnog mapiranja određeni blok uvek ide u određenu liniju u keš memoriji. Ovo je određeno LSB (en. least significant bit) indeksom bloka.

- Asocijativno mapiranje - Asocijativno-mapirana keš memorija ima svojstvo da se svaki memorijski blok može se smestiti u proizvoljnu liniju u keš memoriji.

- Set asocijativno mapiranje - Set-asocijativno-mapirana keš memorija ima svojstvo da se svaki memorijski blok može se smestiti u proizvoljnu liniju u unutar određenog skupa u keš memoriji.

**35. Ukratko opisati look-aside keš memoriju.**

Ova keš memorija radi paralelno sa centralnom procesorskom jedinicom (CPU) i koristi se za skladištenje podataka koji se često koriste ili podaci koji zahtevaju brz pristup. Kontroler keša je smešten izvan CPU-a i radi paralelno sa CPU-om kako bi omogućio efikasan pristup podacima.

**36. Ukratko opisati look-through keš memoriju.**

Ova keš memorija radi paralelno sa centralnom procesorskom jedinicom (CPU) i koristi se za skladištenje podataka koji se često koriste ili podaci koji zahtevaju brz pristup. Kontroler keša je integrisan u CPU i omogućava efikasan pristup podacima, pri čemu se prvo proverava keš pre pristupa glavnoj memoriji.

**37. Opisati postupke upisa u keš memoriju write through i write back.**

Write-through (pisanje preko): Svaka operacija upisa u keš memoriju odmah se ažurira i u glavnoj memoriji.

Write-back (pisanje natrag): Ažuriranja podataka se prvo obavljaju samo u kešu, a zatim se podaci sinhronizuju i ažuriraju u glavnoj memoriji samo kada je to potrebno.

**38. Ukratko opisati PCI magistralu.**

PCI je magistrala koja se koristi za povezivanje različitih komponenti unutar računara, kao što su grafičke kartice, zvučne kartice, mrežne kartice i drugi periferni uređaji. PCI magistrala omogućava brz prenos podataka između ovih komponenti i centralnog procesora.

PCI magistrala je paralelna, i ima master uređaj ili inicijator i slave ili ciljni uređaj. Obično se operacije upisa ili čitanja obavljaju kroz više ciklusa magistrale

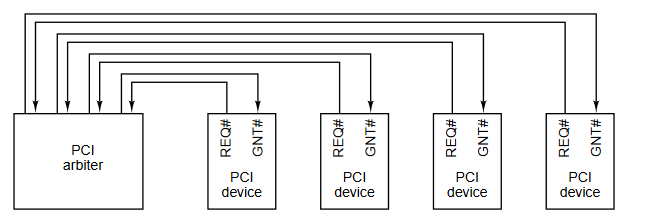
**39. Opisati arbitraciju na PCI magistrali**

PCI magistrala ima centralni arbitratorski uređaj koji odlučuje koji će PCI uređaj dobiti magistralu.

Prva linija, REQ#, ide od uređaja ka arbitratoru i preko nje PCI uređaj šalje zahtev za arbitraciju.

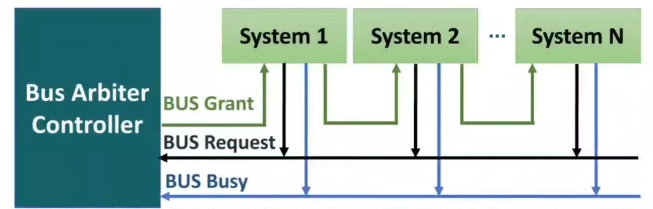
Druga linija, GNT#, ide od arbitratora ka uređaju i preko nje arbitrator šalje dozvolu za korišćenje PCI magistrale.

**40. Opisati arbitraciju na magistrali kad je procesor kao arbitar. Opisati jedan scenario arbitracije po želji. Nacrtati dijagram i opisati tok događaja.**

Arbitracija na magistrali je postupak kojim se određuje koji uređaj ima pravo da koristi magistralu u određenom trenutku. Ako je procesor arbitar, on ima zadatak da upravlja pristupom do magistrale sa ciljem da se izbegnu konflikti i obezbedi fer pristup, procesor. Kada jedan uređaj želi da koristi magistralu, on mora da pošalje zahtev procesoru. Procesor tada procenjuje sve zahteve i odlučuje koji uređaj može da koristi magistralu.

Na primer, imamo tri komponente (A, B i C) koje treba da koriste magistralu da bi prenele podatke. U ovom scenariju, procesor je taj koji odlučuje ko od njih ima pristup magistrali.

1. Komponenta A počinje sa slanjem zahteva procesoru da koristi magistralu. Istovremeno, i komponente B i C imaju podatke za prenos.
2. Procesor prima zahteve od svih komponenti. Kako je procesor arbitar, odlučuje ko će prvi imati pristup magistrali. Recimo da procesor odabere komponentu A.
3. Komponenta A sada prenosi svoje podatke preko magistrale. Tokom tog vremena, komponente B i C čekaju.
4. Nakon što je komponenta A završila sa prenosom, procesor daje pristup magistrali komponenti B, a zatim komponenti C.

**41. Opisati arbitraciju na magistrali kada postoji serijska arbitracija. Navesti prednosti i nedostatke ovakve arbitracije. Nacrtati dijagram.**

Arbitracija na magistrali, kada se koristi serijska arbitracija, znači da uređaji na magistrali zahtevaju pristup do magistrale na osnovu određenog redosleda ili prioriteta.

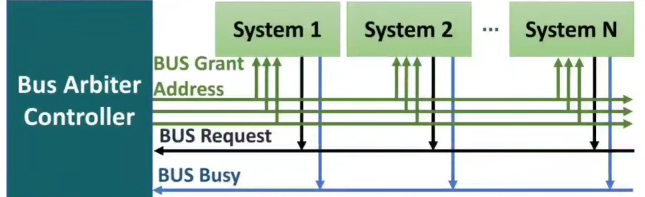
Serijska arbitracija se često implementira koristeći takozvani "daisy chaining" princip. Svaki uređaj je povezan sa prethodnim uređajem na magistrali u obliku lanca. Kada jedan uređaj završi sa korišćenjem magistrale, on prosleđuje kontrolu sledećem uređaju u lancu. Prvi uređaj u lancu ima najveći prioritet za pristup magistrali.

Prednosti serijske arbitracije su:

* Jednostavnost dizajna: Jednostavno je dodati ili ukloniti uređaje iz lanca.
* Determinizam: Redosled pristupa magistrali je jasno definisan.

Nedostaci serijske arbitracije su:

* Kašnjenje: Uređaji na kraju lanca mogu dugo čekati na pristup magistrali.
* Tačka neuspeha: Ako jedan uređaj u lancu ne funkcioniše, svi uređaji nakon njega su blokirani.

**42. Opisati arbitraciju na magistrali kada postoji paralelna arbitracija sa prozivkom. Navesti prednosti i nedostatke ovakve arbitracije. Nacrtati dijagram.**

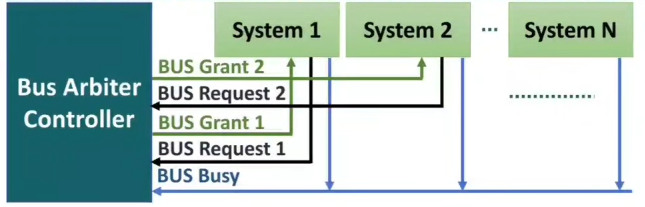
Arbitracija na magistrali sa paralelnom arbitražom sa prozivkom omogućava da svaki uređaj na magistrali može direktno da zahteva pristup magistrali. Kontroler magistrale "proziva" svaki uređaj na magistrali da vidi da li zahteva pristup.

Prednosti ovog pristupa uključuju:

* Brzina: Pošto svaki uređaj može direktno da zahteva pristup, ne postoje kašnjenja kao kod serijske arbitracije.
* Robustnost: Ako jedan uređaj ne funkcioniše, to ne utiče na mogućnost drugih uređaja da pristupe magistrali.

Nedostaci ovog pristupa uključuju:

* Kompleksnost: Paralelna arbitracija sa prozivkom zahteva kompleksniju logiku u kontroleru magistrale i veću količinu kablaže.

**43. Opisati arbitraciju na magistrali kada postoji nezavisna paralelna arbitracija. Navesti prednosti i nedostatke ovakve arbitracije. Nacrtati dijagram.**

Nezavisna paralelna arbitracija je sistem arbitracije gde svaki uređaj ima svoj paralelni put do kontrolera magistrale i može nezavisno zahtevati pristup. Ova metoda ne zahteva kontrolu prioriteta od strane centralnog kontrolera magistrale, već svaki uređaj nezavisno odlučuje kada će zahtevati pristup.

Prednosti nezavisne paralelne arbitracije:

1. Jednostavnost: Pošto svaki uređaj nezavisno odlučuje kada da zahteva pristup, ovaj sistem je relativno jednostavan za implementaciju.
2. Skalabilnost: Dodavanje novih uređaja je relativno jednostavno, jer svaki uređaj zahteva svoj paralelni put do kontrolera magistrale.

Nedostaci nezavisne paralelne arbitracije:

1. Potencijalni sukobi: Bez centralnog kontrolera koji odlučuje o prioritetima, postoji potencijal za sukobe kada više uređaja istovremeno zahteva pristup.
2. Potencijalno neefikasno korištenje magistrale: Uređaji mogu neefikasno koristiti magistralu ako nisu pravilno programirani da zahtevaju pristup samo kada je to potrebno.

**44. Opisati arhitekturu sa više magistrala sa north i south bridge čipovima.**

Arhitektura sa više magistrala sa North i South Bridge čipovima je tradicionalna organizacija unutar računarskog sistema koja omogućava efikasnu komunikaciju između procesora, memorije, periferija i drugih komponenti sistema.

North Bridge je čip koji obezbeđuje visokokapacitivne veze između CPU-a i ostalih glavnih komponenti sistema, kao što su RAM memorija, grafička kartica i magistrale koje povezuju CPU sa perifernim uređajima.

South Bridge takođe poznat kao I/O Controller Hub (ICH), je čip koji obezbeđuje komunikaciju između CPU-a i spoljnih perifernih uređaja, kao što su hard diskovi, USB portovi, mrežna kartica i zvuk.

**45. Opisati PCIe magistralu. Navesti šta je putanja (lane). Kakva je arbitracija na PCIe magistrali?**

PCIe magistrala je visokoperformantna serijska magistrala koja se koristi za brzu komunikaciju između različitih komponenti računarskog sistema.

Putanja (lane) u PCIe magistrali predstavlja pojedinačni kanal za prenos podataka. PCIe magistrala može imati različite konfiguracije sa različitim brojem putanja. Najčešće korišćene konfiguracije su x1, x4, x8 i x16.

**46. Opisati slojeve PCIe magistrale.**

Najniži je fizički sloj. Bavi se premeštanjem bitova od pošiljaoca do primaoca preko point-to-point veze.

Sloj veze se bavi sa prenosom paketa.

Transakcioni sloj upravlja akcijama magistrale.

Softverski sloj povezuje PCI Express sistem sa operativnim sistemom.

**47. Šta predstavlja kontroler I/O uređaja?**

Kontroler I/O uređaja je komponenta ili sklop koji upravlja komunikacijom između centralnog procesora (CPU) i perifernih uređaja. I/O (Input/Output) se odnosi na ulazne i izlazne operacije podataka, kao što su slanje i primanje podataka preko tastature, miša, monitora, štampača, diskova, USB uređaja i drugih perifernih uređaja.

**48. Kako se I/O uređaji klasifikuju?**

1. Prema vrsti podataka koje obrađuju:
   1. Ulazni uređaji (Input Devices): Ovi uređaji omogućavaju unos podataka u računar. Na primer, tastatura, miš, skener.
   2. Izlazni uređaji (Output Devices): Ovi uređaji prikazuju ili izbacuju podatke iz računara. Na primer, monitor, štampač, zvučnici.
2. Prema načinu prenosa podataka:
   1. Serijski uređaji: Podaci se prenose serijski, tj. jedan bit po jedan u nizu. Na primer, serijski port, USB.
   2. Paralelni uređaji: Podaci se prenose paralelno, tj. više bitova odjednom. Na primer, paralelni port, šina.
3. Prema tipu podataka koje obrađuju:
   1. Tekstualni uređaji: Ovi uređaji obrađuju tekstualne podatke. Na primer, tastatura, čitač bar-kodova.
   2. Grafički uređaji: Ovi uređaji obrađuju i prikazuju grafičke podatke. Na primer, monitor, grafička tabla.
4. Prema načinu interakcije sa korisnikom
5. Prema načinu povezivanja sa računarom

**49. Opisati svojstvo vremema pristupa kod I/O uređaja.**

Vreme pristupa je svojstvo koje se odnosi na brzinu kojom I/O uređaj može pristupiti ili izvršiti operaciju sa podacima.

Latencija - Ovo je vreme koje je potrebno da I/O uređaj započne operaciju nakon što je primio zahtev.

Propusnost - Propusnost se odnosi na brzinu kojom I/O uređaj može preneti podatke jednom kada je operacija započeta.

Vreme prenosa - Ovo je vreme koje je potrebno za prenos podataka između I/O uređaja i CPU-a.

**50. Opisati svojstvo vremema za prenos podataka kod I/O uređaja.**

Vreme prenosa podataka kod I/O uređaja je svojstvo koje se odnosi na vreme potrebno za prenos podataka između uređaja i računara.

Obim bloka podataka određuje koliko podataka se prenosi odjednom. Neki uređaji, prenose podatke jedan po jedan (bajt po bajt), dok drugi uređaji prenose blokove podataka.

Brzina prenosa podataka je faktor koji utiče na vreme prenosa. Brzina prenosa podataka kod I/O uređaja često je manja u poređenju sa brzinom prenosa između dve memorije u računaru.

**51. Opisati svojstvo verovatnoće greške kod I/O uređaja.**

Svojstvo verovatnoće greške kod I/O uređaja se odnosi na mogućnost pojave grešaka tokom prenosa podataka između uređaja i računara. Ovo svojstvo je važno jer prenos podataka može biti podložan različitim faktorima koji mogu dovesti do grešaka u podacima. Zbog fizičkog kretanja medijuma na kome je upisana informacija, velike gustine pakovanja (bitovi/inču), čestica u vazduhu, amortizacije…

**52. Kod mehaničkih hard diskova, šta predstavlja vreme punog a šta vreme srednjeg pozicioniranja?**

Kod mehaničkih hard diskova, vreme punog (maksimalnog) pozicioniranja (eng. Full Seek Time) predstavlja vreme potrebno da se glava čitača/pisača hard diska premesti sa jednog kraja diska na drugi.

Vreme srednjeg pozicioniranja (eng. Average Seek Time) predstavlja prosečno vreme koje je potrebno za pozicioniranje glave čitača/pisača sa jedne ciljne pozicije na drugu ciljnu poziciju na disku.

Oba ova vremena su važna u kontekstu performansi hard diska.

**53. Kod mahaničkih hard diskova, šta predstavlja vreme linearnog pozicioniranja, šta vreme rotacionog kašnjenja, a šta vreme akrivacije glave za upis/čitanje?**

Vreme linearnog pozicioniranja (eng. Linear Seek Time): Ovo vreme predstavlja vreme potrebno da se glava čitača/pisača hard diska premesti sa trenutne pozicije na željenu ciljnu poziciju duž prečnika diska.

Vreme rotacionog kašnjenja (eng. Rotational Latency): Ovo vreme predstavlja vreme čekanja koje nastaje zbog rotacije diska.

Vreme aktivacije glave za upis/čitanje (eng. Head Access Time): Ovo vreme predstavlja vreme potrebno da se glava čitača/pisača aktivira za čitanje ili upisivanje podataka.

**54. Kako se dele kontroleri hard diskova?**

Unutrašnji (integrisani) kontroleri: Ovi kontroleri se nalaze direktno na samom hard disku. Oni su integrisani u hard disk kao deo njegove elektronike. Unutrašnji kontroleri obavljaju različite zadatke, uključujući upravljanje komunikacijom između hard diska i računara.

Spoljašnji (eksterni) kontroleri: Ovi kontroleri se nalaze van samog hard diska i koriste se za povezivanje hard diska sa računarom putem odgovarajućeg interfejsa.

**55. Ukratko opisati kompaktne kontrolere hard diskova.**

Kompaktni kontroleri: Kompaktni kontroleri su integrisani kontroleri koji kombinuju više funkcija i komponenti na jednom čipu ili pločici. Oni obavljaju sve osnovne funkcije kontrolera hard diska.

**56. Ukratko opisati baferske kontrolere hard diskova.**

Baferski kontroleri: Baferski kontroleri su specijalizovani kontroleri koji se koriste za upravljanje baferom (međuprostorom) između računara i hard diska. Oni omogućavaju privremeno skladištenje podataka u baferu pre nego što se podaci zapravo zapisuju na sam hard disk.

**57. Ukratko opisati keš kontrolere hard diskova.**

Potpuni keš kontroleri: Potpuni keš kontroleri su napredniji kontroleri koji uključuju veliki keš (memoriju visoke brzine) na samom kontroleru. Keš se koristi za skladištenje često korišćenih podataka ili instrukcija, čime se poboljšava brzina pristupa podacima i smanjuje opterećenje na samom hard disku.

**58. Šta predstavlja ISA model?**

ISA model obuhvata tipove instrukcija koje su podržane, kao i format koji će biti korišćen

za kodiranje tih instrukcija. ISA model takođe opisuje kako će mikroprocesor obrađivati i

izvršavati instrukcije, kao i druge funkcije koje su podržane.

**59. Opisati svojstva ISA modela.**

Arhitektura skupa instrukcija (ISA) najčešće se bavi najnižim slojem

softverske i najvišim slojem hardverske hijerarhije.

- Format instrukcija: ISA definiše strukturu instrukcija, uključujući broj operanada, njihove veličine i način adresiranja. Ovo omogućava programerima da napišu program koristeći konkretne instrukcije razumljive procesoru.

- Podržane operacije: ISA specificira skup operacija koje procesor može izvršavati, kao što su matematičke operacije, logičke operacije, upravljanje grananjem, pristupanje memoriji i druge operacije specifične za arhitekturu.

- Registri: ISA definiše registre koji su dostupni programerima. Registri se koriste za čuvanje podataka tokom izvršavanja programa i obavljaju brze operacije. Ovo uključuje registre za podatke, adresiranje, kontrolu i druge svrhe.

- Način adresiranja: ISA model opisuje različite načine adresiranja memorije, kao što su direktno adresiranje, indirektno adresiranje, relativno adresiranje, indeksno adresiranje i drugi. Ovi načini omogućavaju programerima da pristupaju podacima u memoriji na efikasan način.

- Način izvršavanja instrukcija: ISA specificira redosled izvršavanja instrukcija, kao i način upravljanja kontrolom toka programa, kao što su uslovi grananja, petlje i skokovi.

- Podržane arhitekture: ISA model može biti dizajniran za određenu arhitekturu, kao što su x86, ARM, MIPS i druge. Svaka arhitektura ima svoje specifičnosti u pogledu podržanih instrukcija, registara i načina izvršavanja.

**60. Opisati sledeće registre:**

a. Programski brojač je registar koji čuva trenutnu adresu instrukcije koja se izvršava. On se koristi za praćenje redosleda izvršavanja instrukcija i pokazuje na sledeću instrukciju koja će biti izvršena.

b. Adresni registar memorije je registar koji sadrži adresu u memoriji na koju se odnosi trenutna operacija. On se koristi za čitanje ili upisivanje podataka na tačno određenu lokaciju u memoriji.

c. Prihvatni registar podataka je registar koji privremeno čuva podatke koji se čitaju ili upisuju iz/na memoriju.

d. Prihvatni registar instrukcije je registar koji privremeno čuva trenutno izvršenu instrukciju.

e. Prihvatni registar izvorišnih operanada je registar koji privremeno čuva vrednosti izvorišnih operanada (npr. podaci ili adrese memorije) potrebnih za izvršavanje instrukcija.

f. Prihvatni registar rezultata je registar koji privremeno čuva rezultat izvršene operacije. Ovaj registar često sadrži rezultat aritmetičke, logičke ili memorijske operacije pre nego što se taj rezultat upiše u memoriju ili prenese na drugi registar.

**61. Za šta se koristi indeksni registar, za šta registri opšte namene, a za šta PSW registar?**

Indeksni registar: Indeksni registar se koristi za efikasno adresiranje elemenata u nizu ili strukturi podataka. On čuva vrednost koja se koristi kao pomak (indeks) za pristupanje određenim elementima u memoriji.

Registri opšte namene: Registri opšte namene su registri koji mogu biti korišćeni za različite svrhe u skladu sa zahtevima programa. Oni se koriste za skladištenje privremenih vrednosti, adresa, rezultata operacija i drugih podataka koji su neophodni tokom izvršavanja programa.

PSW registar je poseban registar koji čuva informacije o stanju procesora i izvršavanju programa.

**62. Ukratko opisati instrukcije skoka i navesti podelu ovih instrukcija.**

Bezuslovni skokovi: Ove instrukcije omogućavaju programu da neprekidno preskoče na drugu lokaciju u programu bez obzira na bilo kakav uslov.

Uslovni skokovi: Ove instrukcije omogućavaju programu da preskoče na drugu lokaciju samo ako je određeni uslov ispunjen.

Skokovi na adresu registra: Ove instrukcije omogućavaju programu da preskoče na adresu koja je sadržana u određenom registru.

**63. Ukratko opisati mehanizam prekida (ISR)**

Prekid (Interrupt) je mehanizam koji omogućava uređaju ili softveru da privuče pažnju procesora. Kada se prekid dogodi, procesor obustavlja trenutno izvršavanje programa, i prelazi na izvršavanje posebne rutine poznate kao prekidna rutina ili ISR (Interrupt Service Routine). Prekidi omogućavaju efikasniju upotrebu procesora, jer on ne mora stalno da proverava da li su I/O operacije završene ili da li su se dogodile greške.

**64. Navesti glavne funkcije upravljanja memorijom računara.**

1. *Traganje za Slobodnom Memorijom:* Sistem mora pronaći dovoljno slobodne memorije za smeštanje novih procesa.
2. *Alokacija i Oslobađanje Memorije:* Sistem dodeljuje slobodnu memoriju procesima i oslobađa je kada je više ne koriste.
3. *Zaštita Memorije:* Sistem štiti procese od neovlašćenog pristupa memoriji drugih procesa.
4. *Virtualna Memorija:* Omogućava procesima da koriste više memorije nego što je fizički dostupno, premeštajući podatke između RAM-a i diska.
5. *Straničenje i Segmentacija:* Sistemi mogu koristiti straničenje, segmentaciju, ili oboje da bi efikasno upravljali memorijom i smanjili fragmentaciju.
6. *Menadžment Keš Memorije:* Keš memorija čuva kopije najčešće korišćenih podataka radi poboljšanja performansi.
7. *Garbage Collection:* U nekim sistemima, proces automatskog povraćanja memorije ("garbage collection") oslobađa memoriju koja više nije potrebna.

**65. Opisati šta je fizički a šta logički adresni prostor.**

Fizički adresni prostor predstavlja stvarna mesta u memoriji računara, to jest, prave fizičke adrese na kojima se podaci nalaze.

Logički adresni prostor je kako programi vide memoriju. Svaki program ima svoj sopstveni prostor u memoriji tj. logički adresni prostor, a operativni sistem koristi mape da bi pretvorio ove logičke adrese u fizičke adrese gde se podaci stvarno nalaze.

**66. Opisati overlay mehanizam upravljanja memorijom. Uraditi kratak primer.**

Overlay mehanizam omogućava izvršavanje programa koji je veći od raspoložive memorije. To postiže tako što deli program na manje delove, poznate kao "overlays". Samo jedan overlay je učitan u memoriju u datom trenutku.

Primer: Pretpostavimo da imamo program koji se sastoji iz tri segmenta: A, B i C. Segment A je uvek potreban, dok su segmenti B i C potrebni samo ponekad i oni se međusobno isključuju.

1. Inicijalno, učitavamo segment A u memoriju, jer je uvek potreban.
2. Kada je potrebno izvršiti segment B, overlay mehanizam učitava segment B preko segmenta C u memoriji.
3. Ako kasnije treba izvršiti segment C, overlay mehanizam učitava segment C preko segmenta B u memoriji.

**67. Opisati koncept virtuelne memorije.**

Virtuelna memorija je tehnika koja računarima omogućava da koriste više memorije nego što je fizički dostupno. Omogućava da se podaci privremeno skladište na disku kada nije dovoljno prostora u RAM-u, a zatim se vrate u RAM kada su potrebni. To poboljšava performanse i omogućava izvršavanje većih programa.

**68. Opisati proces prevođenja virtuelnih u fizičke memorijske adrese.**

Prevođenje virtuelnih u fizičke adrese je proces kojim operativni sistem mapira adrese koje program koristi (virtuelne) na stvarne lokacije u RAM-u (fizičke). Ovo se postiže kroz dva koraka: segmentaciju, gde se virtuelna adresa deli na segment i offset; i straničenje, gde se segment dalje deli na stranice, a offset unutar stranice određuje tačnu lokaciju.

**69. Koja su dva tipa memorijske alokacije?**

1. Statička alokacija: U ovom tipu alokacije, memorija se rezerviše pre izvršenja programa. Količina memorije koja se rezerviše je fiksna i ne menja se tokom izvršenja. Ovaj tip alokacije je efikasan kada je unapred poznato koliko će memorije biti potrebno.
2. Dinamička alokacija: U ovom tipu alokacije, memorija se rezerviše tokom izvršenja programa. Količina memorije može se menjati kako se program izvršava, omogućavajući fleksibilnost ali i veću kompleksnost upravljanja memorijom. Ovaj tip alokacije je koristan kada se količina potrebne memorije ne može unapred odrediti.

**70. Ukratko opisati proces straničenja.**

Straničenje je tehnika upravljanja memorijom u kojoj se fizička memorija deli na blokove fiksne veličine, poznate kao stranice. Virtuelna memorija operativnog sistema je takođe segmentirana na blokove iste veličine, nazvane stranički okviri. Kada je procesu potrebna memorija, operativni sistem mu dodeljuje odgovarajući broj straničnih okvira.

**71. Ukratko opisati proces segmentacije.**

Segmentacija je metoda upravljanja memorijom u kojoj se memorija deli na segmente različitih veličina, a svaki segment odgovara određenom delu programa ili podataka. Svaki segment ima svoju bazu (početnu adresu) i granicu (veličinu). Kada program pristupa memoriji, operativni sistem proverava da li je adresa unutar granica segmenta. Ako jeste, adresa se prevede u fizičku adresu. Ako nije, generiše se greška.

**72. Opisati virtuelni memoriju kod i7 procesora.**

Arhitektura memorije Intel i7 procesora koristi tzv. "paging" tehniku, gde je virtuelni adresni prostor podeljen na stranice fiksne veličine, tipično 4KB. Ove stranice su mapirane na fizičku memoriju kroz strukturu podataka poznatu kao "page table". Page table je zapravo niz referenci koji omogućavaju prevođenje virtuelnih adresa u fizičke.

**74. Ukratko opisati ulogu asemblera.**

Asemblerski programi su brzi i štede na memoriji. Asembler transformiše simboličke adrese u numeričke, pretvara simboličke kodove operacija u mašinski kod i rezerviše memoriju za instrukcije i podatke.

Rezultat je mašinski kod, koji se, ako je potrebno, povezuje sa bibliotekama pomoću linkera. Zatim, loader postavlja ovaj kod u memoriju, gde se izvršava. Ukratko, put koda je: asembler - linker - loader - izvršavanje.

**75. Koje su prednosti programa napisanih u asembleru, a koje mane.**

Assembler programi omogućavaju direktnu interakciju s procesorom, efikasno koristeći celokupnu računarsku arhitekturu, što rezultira manjim i bržim programima.

Sa druge strane, ovi programi su teško čitljivi, posebno u velikim projektima, a njihova složenost i specifičnost arhitekture čine prenos koda na druge platforme izazovnim.

**76. Opisati asemblersku instrukciju u opštem slučaju.**

Asemblerska instrukcija sastoji se od četiri dela:

* Labele: simbolički označavaju memorijske lokacije za skokove ili imenovanje podataka.
* Naredbe: predstavljaju mašinske instrukcije ili uputstva asembleru.
* Operandi: elementi nad kojima se instrukcija izvršava (adrese, registri, konstante).
* Komentari: tekstualne napomene koje objašnjavaju funkcionisanje programa.

**77. Šta predstavlja pseudoinstrukcije?**

Pseudoinstrukcije su specijalne instrukcije koje pružaju uputstva asembleru o tome kako da asemblira program, ali se ne prevode direktno u mašinske instrukcije i zbog toga se ne izvršavaju od strane procesora kao obične mašinske instrukcije.

**78. Objasniti funkciju makro naredbe.**

Makro naredbe u asembleru omogućavaju grupisanje instrukcija koje se često ponavljaju, pružajući efikasan način za njihovo ponovno korišćenje. Kada asembler prvi put susretne makro, smešta ga u specijalnu tabelu. Pri svakom narednom susretu sa pozivom makroa, umesto da pozove proceduru, asembler kopira ceo blok instrukcija tog makroa na mesto poziva. Ovo se zove makro proširenje, što se brže ivršava, ali može zauzeti više memorije.

**79. Može li makro instrukcija pozvati drugu makro instrukciju. Objasniti.**

Da, makro instrukcija može pozvati drugu makro instrukciju. Ovo se naziva ugnježđenje makroa. Ugnježđene makro instrukcije omogućavaju programerima da stvore kompleksne blokove koda koji se mogu lako ponavljati. Kada asembler naiđe na makro instrukciju unutar drugog makroa, ona se proširuje kao što bi se desilo i sa bilo kojom drugom makro instrukcijom.

**80. Ukratko objasniti funkciju prvog prolaza prevođenja asemblerskog koda.**

Prvi prolaz prevođenja asemblerskog koda prvenstveno služi za izgradnju tabele simbola, koja obuhvata definicije simbola, uključujući oznake instrukcija. Ovaj proces je neophodan zbog specifičnosti asemblera; one zahtevaju da prevođenje prođe više puta kroz kod programa da bi rešilo problem referenci unapred, tj. situacija u kojima se simbol koristi pre nego što je definisan.

**81. Koje interne tabele koristi asembler u prvom prolazu?**

* Tabela simbola: Sadrži sve simbole i njihove vrednosti/adrese.
* Tabela pseudo instrukcija: Čuva direktive asembleru poput definicija podataka.
* Tabela operacionih kodova: Povezuje asemblerske instrukcije sa mašinskim kodovima.
* Tabela literala: Čuva konstantne vrednosti korišćene u programu.

**82. Ukratko objasniti funkciju drugog prolaza prevođenja asemblerskog koda.**

U drugom prolazu prevođenja asemblerskog koda, asembler koristi informacije prikupljene tokom prvog prolaza, posebno tabelu simbola, da generiše konačan mašinski kod, ili objektni kod. To je proces prevođenja svake asemblerske instrukcije u odgovarajući mašinski kod.Opcionalno, asembler može generisati listing prevođenja, koji je tekstualni prikaz asemblerskog koda pored generisanog mašinskog koda.

**83. Ukratko opisati uloge prevođenja izvornih datoteka i povezivanje objektnih modula.**

* Prevođenje izvornih datoteka: Ovaj korak pretvara izvorni kod, koji je napisan u nekom programskom jeziku, u mašinski kod. Rezultat je objektni fajl.
* Povezivanje objektnih modula: Ovde se objektni fajlovi kombinuju u jedan izvršni fajl. Povezivač rešava sve nedefinisane reference iz objektnih fajlova i uključuje potrebne biblioteke. Rezultat je izvršni fajl koji može da se pokrene.

**84. Objasniti proces povezivanja.**

Povezivanje je korak u programiranju gde se objektni moduli spajaju u jedan izvršni program. Prvo se simboli (nazivi funkcija, promenljivih itd.) u programu pronalaze i povezuju. Zatim se moduli1 postavljaju na odgovarajuće mesto u memoriji. Ako jedan deo programa upućuje na nešto u drugom delu, povezivanje popravlja tu referencu tako da upućuje na pravo mesto. Na kraju, dobija se jedan izvršni program koji se može pokrenuti.

**85. Objasniti funkciju objektnog modula kod procesa prevođenja.**

Objektni modul je rezultat prevođenja izvornog koda. Sadrži mašinski kod, informacije o relokaciji za smeštanje koda u memoriji, i simboličke informacije korisne za linkovanje s drugim modulima. Njegova glavna funkcija je da posluži kao međukorak u stvaranju izvršnog programa koji se može pokrenuti.

**86. Objasniti pojam vezivanja u više-programskom okruženju.**

U višeprogramskom okruženju, više programa je učitano u memoriju i izvršava se istovremeno. Pojam "vezivanje" se odnosi na proces dodeljivanja konkretne adrese u memoriji svakoj referenci unutar programa.

Kada se program izvršava, njegov kod mora biti učitan u memoriju. Adresa u memoriji na kojoj je program učitan naziva se početna adresa. Kada program poziva funkciju ili pristupa promenljivoj, koristi se relativna adresa, koja je određena u odnosu na početnu adresu.

**87. Objasniti ulogu dinamičkog povezivanja.**

Dinamičko povezivanje omogućava da se delovi programa (moduli) povežu i učitaju u memoriju samo kada su potrebni tokom izvršenja. Ovo poboljšava efikasnost resursa i omogućava fleksibilnost u modularnosti i ponovnoj upotrebi koda. U višeprogramskom okruženju, dinamičko povezivanje optimizuje upotrebu memorije tako što učitava module samo kada su potrebni.

**88. Opisati Flinovu taksonomiju za paralelne računare.**

Flinova taksonomija je klasifikacijski sistem za paralelne računare. Ona je zasnovana na broju instrukcijskih nizova (koji se odnose na programski brojač) i nizova podataka koje mašina može obraditi. Postoje četiri klase:

* SISD (Single Instruction, Single Data) – Von Neuman mašina
* SIMD (Single Instruction, Multiple Data) – Vektorski superračunar
* MISD (Multiple Instruction, Single Data) – Nema primera
* MIMD (Multiple Instruction, Multiple Data) - Multiračunar

**89. Opisati paralelizam na nivou čipa.**

Paralelizam na nivou čipa se odnosi na istovremeno izvršavanje više instrukcija na jednom CPU. To se može postići kroz instrukcijski paralelizam, multithreading, ili korišćenje više procesora na jednom čipu, kao što su superskalarni ili VLIW procesori.

**90. Objasniti instrukcijksi nivo paralelizma.**

Instrukcijski nivo paralelizma (Instruction Level Parallelism - ILP) se odnosi na mogućnost izvršavanja više instrukcija u jednom procesorskom ciklusu. Osnovna ideja ILP-a je da dok se jedna instrukcija izvršava, procesor može započeti izvršavanje sledeće instrukcije pre nego što prva instrukcija završi.

**91. Objasniti tehniku više-niti na čipu.**

Tehnika više-niti na čipu (On-Chip Multithreading) znači da jedan procesor (ili jezgro procesora) može upravljati izvršavanjem više niti izvršavanja istovremeno. Kada jedna nit doživi kašnjenje, kao što je promašaj u kešu gde se mora čekati na podatke iz glavne memorije, procesor može preći na izvršavanje druge niti koja je spremna za rad.

**92. Opisati simultani multithreding.**

Simultani multithreading (SMT) je tehnika u arhitekturi procesora koja omogućava višestrukim niti da se istovremeno izvršavaju na jednom CPU jezgru. U SMT, više niti se može simultano izvršavati na jednom procesoru, deljenjem resursa jezgra kao što su registri i cache memorija, ali svaka nit ima sopstveni set registara.

**93. Objasniti tehniku multiprocesora na jednom čipu.**

Tehnika multiprocesora na jednom čipu, poznata kao multicore procesor, uključuje ugradnju više CPU jezgara na jedan čip. Svako jezgro može izvršavati sopstvenu nit, omogućavajući paralelno izvršavanje zadataka. Ova tehnika može povećati performanse sistema, posebno za aplikacije koje su optimizirane za korištenje više jezgara, ali takođe zahteva napredne tehnike za upravljanje izvršenjem niti i raspodelu resursa.

**94. Opisati ulogu koprocesora.**

Koprocesor je specijalizovan procesor koji radi paralelno sa glavnim procesorom (CPU) kako bi poboljšao njegovu funkcionalnost i performanse. Koprocesori su dizajnirani da obavljaju specifične zadatke brže i efikasnije nego što bi to mogao glavni procesor. Oni mogu biti ugrađeni direktno u čip CPU-a, na istoj matičnoj ploči ili u zasebnom kućištu.

**95. Opisati ulogu mrežnih procesora.**

Mrežni procesori su specijalizovani mikroprocesori namenjeni za rukovanje mrežnim paketima u realnom vremenu. Oni se nalaze u mrežnim uređajima poput rutera, switch uređaja i firewall-a. Mrežni procesori su dizajnirani da efikasno obrade veliki broj dolaznih i odlaznih paketa, omogućavajući brzi i efikasni protok podataka kroz mrežu. Oni su tipično programabilni, što znači da se njihove funkcije mogu prilagoditi za specifične potrebe u mrežnom okruženju.

**96. Objasniti razlike izmedju multiprocesorskih i multikompjuterskih sistema sa višestrukim procesorima.**

Multiprocesorski sistemi: Ovi sistemi koriste zajedničku (deljivu) memoriju. Svi procesori koji rade u ovom sistemu imaju pristup istom memorijskom prostoru, što znači da svaki procesor može čitati ili pisati reči u memoriji. Multikompjuterski sistemi: Za razliku od multiprocesorskih sistema, multikompjuterski sistemi se sastoje od samostalnih računara (sa sopstvenim CPU-om i memorijom) koji su međusobno povezani putem mreže. Ovi sistemi ne dele zajedničku memoriju; svaki procesor ima sopstvenu lokalnu memoriju.

**97. Opisati hijerarhijske slojeve kod grid računara.**

* Strukturni sloj (Fabric layer): Predstavlja osnovne hardverske resurse (CPU, diskovi, mreže, senzori) i softverske entitete (programi, podaci) koji čine grid.
* Sloj resursa (Resource layer): Upravlja pojedinačnim resursima unutar grida, obezbeđujući uniformni interfejs za upravljanje resursima, praćenje njihovog statusa i omogućavanje sigurnog korišćenja.
* Kolektivni sloj (Collective layer): Obezbeđuje funkcionalnosti koje se odnose na grupe resursa, kao što su lociranje dostupnih resursa, replikacija podataka, upravljanje pridruživanjem novih resursa i održavanje baza podataka pravila pristupa.
* Sloj aplikacija (Application layer): Ovde se nalaze korisnički programi koji koriste niže slojeve da bi dobili potrebne dozvole za korišćenje resursa, podneli zahteve za korišćenje, nadgledali izvršenje zahteva i obrađivali greške.

**98. Objasniti cloud computing, ulogu i funkciju.**

Cloud computing, ili računarstvo u oblaku, predstavlja model u kojem se resursi (poput računske snage, skladišnog prostora, aplikacija) pružaju kao usluga preko interneta. Njegova glavna karakteristika je skalabilnost - mogućnost da se lako povećava ili smanjuje količina resursa koje korisnik koristi u skladu sa potrebama. Osnovna funkcija cloud computinga je pružanje pristupa IT resursima bez potrebe za direktanim upravljanjem i održavanjem hardvera.

**99. Opisati tri osnovna modela usluga kod klaud računarstva.**

Tri osnovna modela usluga u cloud computingu su:

1. Software as a Service (SaaS): Softver se dostavlja kao usluga preko interneta, umesto da se instalira na korisnikov uređaj. Primeri uključuju Google Docs, Salesforce, Microsoft Office 365.
2. Infrastructure as a Service (IaaS): Pruža korisnicima pristup računarskoj infrastrukturi, kao što su serveri, mreža, operativni sistemi, skladišni prostor. Primeri uključuju Amazon Web Services (AWS), Microsoft Azure.
3. Platform as a Service (PaaS): Pruža platformu koja uključuje operativni sistem, bazu podataka, servere i razvojni alati, omogućavajući razvoj, testiranje i isporuku softvera. Primeri uključuju Google App Engine, Heroku.

**100. Opisati glavne karakteristike x64 arhitekture.**

* Kompatibilnost: x64 je kompatibilna sa 32-bitnom x86 arhitekturom, što znači da većina softvera dizajniranog za x86 može da se izvršava na x64 bez modifikacija.
* Prošireni registri: Postojeći 32-bitni registri su prošireni na 64 bita, a dodato je i osam novih 64-bitnih registara, što omogućava obradu većih količina podataka istovremeno.
* Podrška za 64-bitne celine brojeve: x64 arhitektura podržava 64-bitne celine brojeve kao izvorni tip podataka, što omogućava precizniju i bržu obradu.
* Rad u 32-bitnom kompatibilnom režimu: x64 procesori mogu raditi u 32-bitnom kompatibilnom režimu, omogućavajući korišćenje 32-bitnih operativnih sistema i izvršavanje 32-bitnih aplikacija. U ovom režimu, 64-bitna proširenja nisu dostupna.

**101. Objasniti ARM arhitekturu.**

ARM (Advanced RISC Machines) arhitektura je tip procesorske arhitekture zasnovan na RISC (Reduced Instruction Set Computing) principima, što znači da koristi manji i jednostavniji set instrukcija što doprinosi efikasnijem radu procesora.

Ključne karakteristike ARM arhitekture uključuju:

* Energetska efikasnost: ARM procesori su dizajnirani da koriste manje energije, što ih čini idealnim za mobilne uređaje koji rade na baterije.
* Sistem-na-čipu (SoC): ARM često koristi SoC dizajn, kombinujući različite komponente kao što su CPU, GPU, i kontroleri memorije na jednom čipu.
* Podrška za različite režime rada: ARM arhitektura definiše različite režime rada procesora, poput korisničkog, supervizorskog, i prekidnih režima.

**102. Objasniti RISC arhitekturu.**

RISC (Reduced Instruction Set Computing) je tip procesorske arhitekture koja se fokusira na brzo izvršavanje velikog broja jednostavnih instrukcija.

Ključne osobine RISC arhitekture uključuju:

* Jednostavne instrukcije: RISC koristi jednostavne instrukcije koje se mogu izvršiti za jedan mašinski ciklus. Ovo smanjuje složenost dekodiranja i izvršavanja instrukcija.
* Fiksna širina kodiranja instrukcija: Instrukcije u RISC arhitekturi imaju fiksnu dužinu, što olakšava dekodiranje instrukcija.
* Upotreba registara: RISC koristi registarsko-registarski model, gde se aritmetičko-logičke operacije izvršavaju isključivo nad registarskim operacijama. Instrukcije LOAD i STORE koriste se za prebacivanje podataka između registara i memorije.

**103. Objasniti CISC arhitekturu.**

CISC (Complex Instruction Set Computing) je tip procesorske arhitekture koja koristi manji broj složenijih instrukcija. CISC je dizajniran da izvršava složeniji skup instrukcija sa ciljem smanjenja broja instrukcija po programu.

Ključne osobine CISC arhitekture uključuju:

1. Operandi u memoriji: U CISC arhitekturi, operandi mogu biti smešteni direktno u memoriji, za razliku od RISC arhitekture gde su operandi obično smešteni u registre.
2. Manji kod: Budući da CISC koristi manje, složenije instrukcije, programi napisani za CISC procesore obično zauzimaju manje memorije.
3. Više taktova po instrukciji: Složenost CISC instrukcija znači da im je obično potrebno više taktova procesora da se izvrše.

**104. Opisati RISC-V arhitekturu.**

RISC-V (pronounced "risk-five") je otvoren standard skupa instrukcija (ISA) koji se zasniva na principima smanjenog skupa instrukcija računara (RISC). RISC-V specifikacija je besplatna za javnu upotrebu (open source), a omogućava otvorenu implementaciju hardvera.

Neki ključni aspekti RISC-V arhitekture uključuju:

1. Modularnost: RISC-V arhitektura je dizajnirana da bude modularna i proširiva. Početni skup instrukcija RISC-V, poznat kao "bazni integer ISA", je relativno malen, i drugi skupovi instrukcija, poznati kao "proširenja", mogu se dodati prema potrebi.
2. Jednostavnost: U skladu sa RISC principima, RISC-V instrukcije su dizajnirane da budu jednostavne i efikasne za dekodiranje i izvršavanje.
3. Skalabilnost: RISC-V podržava širok spektar implementacija, od malih, jednojezgarnih mikrokontrolera, do velikih, višejezgarnih procesora namenjenih za visoko-preformansno računanje.

**105. Opisati oblasti u kojima RISC-V procesori ostvaruju značajne rezultate.**

RISC-V procesori ostvaruju značajne rezultate u različitim oblastima, uključujući:

* Ugrađenim sistemima i IoT: Zahvaljujući energetskoj efikasnosti, malom broju tranzistora i prilagodljivosti, idealni su za ugrađene sisteme i IoT uređaje.
* Visoko-preformansnom računanju (HPC): Iako novi na ovom polju, RISC-V procesori se sve više koriste u visoko-preformansnom računanju zbog modularnosti i otvorenosti arhitekture.
* Data centrima i cloud infrastrukturi: Postoji rastući interes za korišćenje RISC-V procesora u data centrima i cloud infrastrukturi zbog mogućnosti prilagođavanja hardvera specifičnim potrebama.
* AI i Machine Learning: Prilagodljivost RISC-V arhitekture omogućava optimizaciju za specifične radne opterećenja kao što su veštačka inteligencija i mašinsko učenje.

**106. Objasniti osnovni set instrukcija sa posebnim osvrtom na osam sistemskih RISC-V arhitekture.**

Osnovni set instrukcija RISC-V arhitekture se sastoji od 4 tipa: integer, multiply, atomic, i floating-point, grupisane u module za različite konfiguracije procesora. Postoje tri osnovna seta instrukcija: RV32I (32-bitni), RV64I (64-bitni) i RV128I (128-bitni).

RISC-V ima 8 sistemskih registara. To su: Zero (uvek nula), RA (Return Address), SP (Stack Pointer), GP (Global Pointer), TP (Thread Pointer), i T0-T2 za privremene podatke.

**107. Opisati razlike kod Qualcomm Snapdragon 8 gen 1 i APPLE M1 procesora.**

* Procesorske Jezgre: Snapdragon 8 Gen 1 koristi 8 jezgara podeljenih u tri klastere, dok Apple M1 koristi 8 jezgara podeljenih u dva klastera.
* Proizvodni Proces: Snapdragon 8 Gen 1 je izrađen na 4nm tehnologiji, dok je Apple M1 izrađen na 5nm tehnologiji. Manji broj znači da je tehnologija novija i efikasnija.
* Performanse: Oba čipa nude visoke performanse, ali se razlikuju u odnosu na određene zadatke. Apple M1 je optimiziran za rad sa Apple-ovim operativnim sistemom i softverom, dok Snapdragon 8 Gen 1 pruža snažne performanse u širokom spektru Android uređaja.
* Memorija: Oba čipa mogu podržati do 16 GB RAM-a, ali se može razlikovati u brzini i efikasnosti memorije.