Skripta za prvi deo(Teorija+Urađeni zadaci)

1.1.Osnovna podela ili klasifikacija softvera ja na aplikativni i sistemski.

1.2.Aplikativni softver se projektuje za rešavanje konkretnih problema,kao sto su na primer:placanje preko Interneta, rezervacija avionskih karata,upravljanje bankomatima,učenje na daljinu ili rad na berzi.Sistemski softver predstavlja programsko okruzenje koje omogućava programerima da brze i jenostavnije kreiraju aplikacije prema zahtevima korisnika.Na primer:operativni sistemi,programski prevodioci,linkeri i intepreteri pripadaju sistemskom softveru.

1.3.Arhitektura računara pokazuje racunarski sistem iz ugla programera,bitan koncept u arhitekturi računara je koriscenje razlicitih nivoa apstakcije.Svaki nivo apstarkije sadrzi interfejs.Dobro poznavanje arhitekture računara je važno za razumevanje programskih prevodioca,operativnih sistema i pisanje programa.Na primer arihtektura računara pokazuje veličine tipova podataka i tipove podrzanih operacija dok ne pokazuje vrstu tehnologije čipa koji se koristi za implementaciju memorije.

1.4.Organizacija računara pokazuje strukturne veze unutar računara koje nisu vidljive od strane programera,kao sto su na primer interfejsi ka periferijskim uređajima,učestanost takta i tehnologija koja se koristi za memoriju,tok informacija izmedju komponenata,mogućnosti i preformanse funkcijskih jedinica.Dobro poznavanje organizacije računara je važno za logičko projektovanje računara i za optimizaciju preformansi aplikacija.

1.5.Osnovne komponente originalne von Neumann-ove mašine su:
Memorija,
upravlječka jedinica,
aritmetičko-logička jedinica,
ulazna jedinica
i izlazna jedinica.(Slika 1.1)

1.6.Bitna karakteristika von Neumann-ove arihtekture je sekvencijalno izvršavanje instrukcija,tj. Odsustvo bilo kakvog paralelizma ili preklapanja instrukcije.Druga važna karakteristika je postojanje samo jedne putanje izmedju operativne memorije i uoravljačke jedinice procesora.To direktno utiče na performanse sistema i naziva se von Neumann-ovo usko grlo.

1.7.Harvard arihitektura predstavlja varijaciju originalne von Neumann-ove arhitekture.Harvard arhitektura je imala posebnu memoriju za instrukcije i posebnu memoriju za podatke.Ova arhitektura takodje sadrzi posebne putanje za instrukcije i podatke nezavisno od toga da li se kosriste posebne memorije.(Slika 1.3)

1.8.Slika 1.5

1.9.Princip ekvivalencije hardvera i softvera:sve sto se moze uraditi pomocu softvera moze se takođe uraditi i pomocu hardvera i sve sto se moze uraditi pomocu hardvera moze se takođe uraditi i pomocu softvera.Ovaj princip ne govori o brzini implementacije.Hardverske implementacije su skoro ucek brze od softverskih.Izbor izmedju hardverskog i softverskog resenja zavisi od faktora kao sto su cena,brzina,pouzdanost i frekvencija očekivanih promena.

1.10.Michael Flynn polazeći od toga da računari koriste podatke i instrukcije,klasifikovao sve računarske sisteme u četri grupe.Osnovna klasifikacije su tok insturkcija i tok podataka.Definisao je tok instrukcija kao niz instukcija koje izvršava data mašina i tok podataka kao niz podataka koje koristi tok instukcija(uključujuci ulazne podatke i parcijalne rezultate).Flynn-ova klasifikacija računara:
1.SISD(˝Single Instruction stream,Single Data stream˝),
2.SIMD(˝Single Instruction stream,Multiple Data streams˝),
3.MIMD(˝Multiple Instruction streams,Multiple Data streams˝),
4.MISD(˝Multipe Instruction streams,Single Data stream˝).

1.11.Slika 1.6

1.12.Slika 1.7

1.13.Slika 1.8

Cvija

Skripta za prvi deo(Teorija+Urađeni zadaci)
1.14.Slika 1.9

1.15.Implementacione tehnologije:
Tehnologija integracionih kola,
Tehnologija magnetnih diskova,
Tehnologija DRAM(˝Dynamic Random Access Memory˝)memorija i
Tehnologija mreža.
1.16.Jedinica mere MIPS(˝Milion of Instruction Per Secound˝)označava izvršenje milion mašinskih instrukcija u sekundi.
1.17.Jedinica mere megaflops-MFLOPS(˝Milion of Floathin point Operations Per Secound˝)označava milion operacija u pokretnom zarezu izvšenih u jednoj sekundi.
Gigaflops-GFLOPS označava milijardu operacija u pokretnom zarezu izvšenih u jednoj sekundi.
Teraflops-TFLOPS označava bilion operacija u pokretnom zarezu izvršenih u jednoj sekudi.
1.18.Za najsporije računare milisekunda(0.001 s),a za najbrže pikosekunda(10 na -12 s)
1.19.GHz-Brzina takta-milijardu ciklusa u sekundi.
1.20.Kompatibilan ˝naniže˝(˝backward compatible˝ili ˝downward compatible˝),tj. Da bude u mogucnosti da izvršava prevedeni kod za bilo koji predhodni model procesora.Kompatabilnost ˝naniže˝ima značenje da viši model u istoj seriji procesora može da izvršava sve sistemske i korisničke programe koji se mogu izvršiti na predhodnim modelima.
1.21.Murov zakon(˝Moor's Law˝)preformanse računarskih sistema se poboljšavaju geometriski,a ne linearno.

2.1.Računar opšte namene pored procesora i memorije ima više različitih U/I uređaja.(Slika 2.2)
2.2.Procesor je sa ostalim komponentama povezan pomoću sistemske magistrale.Sistemska magistrala je skup paralelnih zica za prenos podataka,adresa i upravljačkih signala.
2.3.Sistemska magistrala se sastoji od:magistrale podataka,magistrale upravljačkih signala i adresne magistrale.Preko magistrale podataka se prenose podaci,preko adrene magistrale se vrši adresiranje gde se prenose podaci ili odakle.Upravljačka magistrala se koristi za upravljanje funkcijama magistrale i ona omogućava korisnicima magistrale da signaliziraju kada su podaci raspoloživi.
2.4.Memoriska mastrala se koristi za komunikaciju između procesora i primarne memorije.
2.5.U/I magistrala se koristi za komunikaciju između procesora i U/I uređaja.
2.6.Procesor se sastoji od:upravljačke jedinice,aritmetičko-logičke jedinice(ALU-˝Arithmetic Logical Unit˝) i registra.
2.7.Upravljačka jedinica upravlja izvršenjem pojedinačnih insrukcija i pravlja izvršenjem programa u celini.Ona treba da obezbedi da se po izvršenju jedne instrukcije,izvrši sledeća insrukcija programa.
2.8.Aritmetičko-logička jedinica na osnovu signala uravljačke jedinice izvršava potrebne operacije nad operandima instrukcije i generiše rezultat.
2.10.Registar brojač naredbi(BN)-njegov zadatak je da u svakom trenutku izvršenja programa sadrzi adresu sledece instrukcije programa.
2.11.Registar naredbi sadrzi kopiju masinske instrukcije koja je učitana iz glavne memorije i koja se trenutno izvršava.
2.12.U procesu izimanja instrukcije(˝fetch˝)još dva regustra imaju posebnu ulogu.Brojač naredbi sadrži adresu sledeće instrukcije koju treba učitati MAR(memorijski adresni registar),a zatim na adresnu magistralu.Nakon toga upravljačka jedinica zahteva čitanje iz memorije.Početna vrednost se upisuje na magistralu podataka

kopira u PRM(prihvatni registar memorije)i prenosi u registar naredbi RN.Brojač naredbi se povećava za jedan(tačnije za dužinu instrukcije)i na taj način priprem za čitanje sledeće instrukcije.(Slika2.5)
2.13.Arhitektura procesora je određena sa:
Skupom registara,
Skupom tipova podataka koje procesor podržava,
Formatom instrukcija,
Skupom instrukcija,
Načinima adresiranja i
Mehanizmom prekida.
2.14.Nad memoriskim lokacijama se obavljaju dve operacije:operacija čitanja i operacije pisanja(upisivanja u lokaciju).
2.15. Brisanje sadrzaja 1 memorijske reci bez koriscenja CLEAR se moze uraditi na vise nacina:
1. primenom AND operacije i drugog operanda 0 (dobija se 0)
npr. 0101 and 0000 = 0000
2. primenom XOR operacije sa istom reci (dobija se 0)
npr. 0101 xor 0101 = 0000
3. primenom masinske instrukcije ADD A1 0 0(2. i 3. operand su 0, rezultat se upisuje na adresu A1)
npr. a1 = 0 + 0; (moze i da se oduzme sam od sebe, SUB A1, A1, A1)
2.17.Najvažnije karakteristike memorije računara su brzina,veličina(kapacitet),cena i postojanost.Za brzinu i veličinu memorije poželjno je da bzde što je veće moguće,a za cenu što je moguće niža.Bolje je da memorija bude postojana,ali kod brzih memorija to nije slučaj.Sve četri navedene osobine su određene tehnologijom implementacije memorije i vremenom postaju sve bolje.
2.18.Asocijativna memorija je specijalna,mala keš memorija sa brzim pretraživanjem koja se koristi za memorisanje tabela stranica(˝page tables˝).Korišćenje asocijativne memorije može da ubrza pristup,tj. Da smanji efektivno vreme pristupa.
2.19.Za svaki procesor je karakterističan˝uzmi-analiziraj-izvrši˝ciklus(˝fetch-decode-execute cycle˝)koje se stalno ponavlja od trenutka uključenja napajanja da trenutka isključenja napajanja.Najčesće se ovaj ciklus realizuje u sledećih šest koraka ili faza:
1)procesor iz primarne memorije uzima instrukciju koju treba da izvrši na osnovu adrese u BN,
2)dekodira se operacioni kod instrukcije,
3)procesor uzima iz memorije operand ili operande ako je to potrebno,
4)izvršava se instrukcija,
5)rezultat se upisuje ili u registar ili u operativnu memoriju,
6)prekazak na korak 1. (Slika 2.9)
2.20.Podaci ne magnetnom disku se zapisuju u ovbliku koncentričnih krugova koji se zovu staze(˝tracks˝).
2.21.Ovakav način skladištenja podataka predstavlja kompomis između kapaciteta memorije i slozenosti hardvera diska.
2.22.Slika 2.10
2.23.Slika 2.11
2.24.Vreme pristupa disku ima tri komponente:vreme traženja(˝seek time˝),vreme usred rotacije diska(˝rotatiomal latency˝)i vreme prenosa podataka sa diska u operativnu memoriju.Vreme traženja je vreme potrebno za pokretanje glave diska od adresiranog cilindra.Vreme usred rotacije diska je dodatno vreme potrebno da disk rotira do adresnog sektora,tj. Da glava diska bude iznad adresiranog sektora.Vreme prenosa podataka sa diska u operativnu memoriju direktno zavisi od količine podataka koja se prenosi.

2.25.Korišćenjem RAID tehnologije podaci se,umesto samo na jedan,upisuje na više magnetnih diskova.Na taj način može se povećati brzina,pouzdanost i kapacitet memorisanja podataka.
2.26. 2.27. 2.28. 2.29. 2.30. 2.31. 2.32. (str. 47-48)
2.33.Procesor moze da periodično proverava status uređaja(˝pollin˝).Moguća stanja uređaja su na primer:KOMANDA_SPREMNA,ZAUZETA I GREŠKA.Na taj način ostvaruje se ciklus čekanja(˝busy-wait cycle˝)U/I aktivnosti na datom uređaju.Drugi način komunikacije sa U/I uređajima je korišćenje sistema prekida.Signal prekida prihvata funkciju za upravljanje prekidima.Prekidi se mogu maskirati kada je potrebno da se prekid odloži ili ignoriše.Treći način je da se koristi direktan pristup memoriji(DMA-˝Direct Memory Access˝).Kod ovog načina nije potreban programiran U/I,ali je neophodan DMA kontroler.U ovom slučaju podaci se direktno prenose između U/I uređaja i memorije,bez potrebe da procesor učestvuje u procesu,Na kraju prenosa podataka DMA kontroler prekidom obaveštava procesor da je prenos podataka završen.
2.34.Prekidi se koriste za komunikaciju periferijskih uređaja sa centralnim procesorom.Omogućavaju korisćenje servisa operativnog sistema,koriste se za merenje vremena i vremensku kontrolu poslova koju izvršavaju mikroprocesori.
2.35.Postoje tri vrste prekida:
Softverski prekidi,
Hardverski prekidi i
Izuzeci,
2.36.Sistemski pozivi omogućavaju korisničkim procesima da zahtevaju servise operativnog sistema.
2.37.Za maskirajuće prekide dozvola rada se vrši softverski postavljanjem odgovarajućeg bita u statusnoj reči.
2.38.Nastanak takve situacije se može rešiti ili zabranom više prekida u isto vreme ili uvođenjem prioriteta prekida.
2.39.Vreme reakcije predstavlja vreme koje protekne od trenustka pojave signala prekida do trenutka prelaska na prekidni program.
2.40.Zadatak paralelizma ne nivou instrukcije je da na datom računarskom sistemu izvrši sto više instrukcija. Jedan pristup je uzimanje instrukcija unapred.Drugi pristup je korišćenje tehnike protočne obrade(˝pipeline processing˝).Treći pristup je korišćenje superskalarnih arhitektura.
2.47.Može se desiti da više procesora konkurentno pristupa globalnoj memoriji.Tada se za rešavanje mogućeg nastanka kolozije kod pristupa istoj memoriskoj lokaciji korist sledeća tri modela:
EREW(˝Exclusive Read Exclusive Write˝),
CREW(˝Concurrent Read Exclusive Write˝) i
CRCW(˝Concurrent Read Concurrent Write˝).
Prvi navedeni model EREW-eksluzivno čitanje,ekskluzivno pisanje ne dozvoljava nastajanje konflikta niti za čitanje niti za pisanje.Ako prilikom izvršavanja programa do konflikta ipak dodje,stanje programa je tada nedefinisno.
Drugi navedeni mode CREW-konkurentno čitanje,eksluzivno pisanje dozvoljava konkurentno čitanje za više procesora,tj. Za više procesora može istovremeno da čita podatke sa iste memoriske lokacije.Međutim,ako dođe do konkurentnog pisanje stanje programa je nedefinisano.
Treći navedeni model CRCW-konkurentno čitanje,konkurentno pisanje dozvoljava istovremeni pristup više procesora i za operaciju pisanja. U slučaju da više procesora pokušava da istovremeno upiše neku vrednost u istu memorisku lokaciju potrebno je definisati koji će od tih procesora izvršiti upis.
2.48.Moguća su sledeća tri slučaja:
a)svi procesori koji pokušavaju da izvrše operaciju pisanja moraju da upišu istu vrednost,tako da će rezultat upisa biti isti bez obzira koji je procesor to uradio(˝COMMON CRCW˝),

Skripta za prvi deo(Teorija+Urađeni zadaci)
b)bilo koji procesor može da izvrši operaciju pisanja i nema ograničenja za vrednost koje se upisuje(˝ARBITARY CRCW˝)Ovde je moguće da svaki procesor promeni vrednost koju je neki drugi procesor pre toga upisao.
c)svaki procesor ima unapred definisan prioritet i u datom trenutku operaciju pisanja izvršava procesor sa najvišim prioritetom(˝PRIORITY CRCW˝).

3.1.Format instrukcije je definisan ukupnom dužinom instrukcije izraženom u bitovima,brojem,dužinom i značenjem svakog njenog polja.
3.2. Tabela 3.1
3.9. Tipovi instrukcija na konvekcionalnom mašinskom nivou:
Prva grupa – operacije nad podacima:
-instrukcije za prenos podataka,
-unarne operacije i
-binarne operacije
Druga grupa – upravljanje redosledom izvršavanja instrukcija programa:
- poredjenje i uslovni skokovi,
- upravljanje iterativnom programskom strukturom,
- poziv podprograma,
- aktiviranje korutina
Treća grupa – razmena podataka sa ulazno/izlaznim urenajima:
- ulazno/izlazne instrukcije
3.12.Postoje sledeći načini adresiranja:trenutno,direktno memorijsko,direktno registarsko,indirektno memorisko,indirektno registarsko,bazno registarsko i inedksno adresiranje.
3.24. U prvoj reci pozvanog modula, ili na steku.
3.26. *Korutine se koriste u projektovanju i implementaciji programa prevodioca (kompajlera) i u simulaciji uporednih procesa.*

4.1.Skup mikroinstrukcija na osnovu koga je napravljen potpun skup instrukcija se zove mikroprogram ili mikrokod.
4.3.Mikroinstrukcija je skup mikrooperacija koje se izvršavaju u isto vreme,mikroinstrukcije se mogu koristiti za implementaciju stvarnog skupa date mašine.
Mikroprogramiranje je tehnika koja se koristi za implementaciju uoravljačke jedinice.Osnovna ideja je da se upravljačka jedinica implementira kao mašina za izvršavanje mikroprograma(računar unutar računara).
4.4.Dati skup instrukcija na kovnvencionalnom mašinskom nivou može biti implementiran pomoću rezličitih mikroarhitektura.Na primer,skup instrukcija na konvenkcionalnom mašinskom nivou za Intel-ov Pentium procesor je implementiran na više različitih načina.Pored Intel-a ovaj skup instrukcija su implementirali i konkurentni proizvođači procesora:AMD i Cyrix.U okviru svake mikroarhitekture se može ostvariti zadati cilj,kao što su sledeći ciljevi:obezbediti što veću brzinu izvršavanja instrukcije,obezbediti malu potrošnju napajanja,odnsono što manju disipaciju snage ili obezbediti nisku cenu procesora.Proizvođači procesora su u mogućnosti da primenom novijih tehnologija u izradi integrisanih kola ostvare značajna poboljšanja po pitanju preformansi hardvera,a da istovremeno zadrže kompatibilnost softvera u koji su mnoge kompanije pre toga investirale.
Programi se mogu izvršiti nepromenjeni na različitim procesorima sve dotle dok oni podržavaju isti skup instrukcija na konvencionalnom mašinskom nivou nezavisno od pojedinačnih mikroarhitektura.
4.7.Mikroprogram se često zove i firmver(˝firmware˝),jer povezuje hardver i softver,odnosno smanjuje razliku izmedju hardvera i softvera.Osnovna namena firmvera je interpretiranje skupa instrukcija koji je vidljiv krajnjem korisniku.

Cvija

Skripta za prvi deo(Teorija+Urađeni zadaci)
4.8. Firmver se skladišti se u upravljacku memoriju.
4.9.Mikroarhitektura se može podeliti u dve sekcije:sekciju podataka(˝datapath˝) i upravljačku sekciju,Sekciju podataka čine registri i ALU,a upravljačku sekciju upravljačka jedinica.Mikroprogram koji nije vidljiv krajnjem korisniku sistem vrši mikroprogramsko upravljanje,tj. Izvršava operacije nad registrima i drugim komponentima date mašine.
4.10.Za programiranje firmvera koristi se mikro-asambler koji nije vidljiv krajnjem korisniku sistema.
4.11.Bilo koja promena u skupu instrukcija na konvencionalnom mašinskom nivou prouzrokuje promenu i u firmveru.
4.12.Promena softvera na aplikativnom nivou(na nivou korisnika)ne utiče na firmver,
4.13.HDL(˝Hardware Description Language˝)poseban jezik za opis hardvera,Jedan primer HDL jezika je VHDL,koji je akronim za VHSIC(˝Very High Speed Integrated Circuit˝)HDL.VHDL koristi se za opis arhitekture na veoma visokom nivou i može se prevesti u projekat hardvera pomoću procesa koji se zove ˝silikonsko prevođenje˝(silicon compilation˝).Za projektovanje hardverskog rešenja za upravljačku jedinicu podesniji je HDL jezik nižeg nivoa koji se zove RTL(˝Register Transfer Language˝)
4.14.Struktura podataka tipa LIFO(˝Last In First Out˝).Sastoji se od skupa memoriskih lokacija u koje se upisuju vrednosti podataka.Kada se vrednost podataka upisuje na stek to se radi unutar lokacije koja je na vrhu steka i tada se sve do tada upisanje vrednosti pomeraju za jednu lokaciju naniže.Podaci se mogu čitati samo sa vrha steka.Tada se svi do tada upisani podaci pomeraju za jednu lokaciju naviše.
4.15.Hipotetička stek mašina je mašina sa apstraktnom strukturom podataka tipa LIFO,pri čemu program ne pristupa registarskoj dadoteci,već isključivo steku.
4.17.Radi ubtzanja pristupa steku korsti se registarska datoteka unutar procesora gde se memoriše prvih N vrednosti sa steka,gde je N veličina registarske datoteke.
4.18. Upravljacka memorija je ROM memorija (64x27 bita)

Slike:

Zadaci:

1. 32-bitna vrednost 0x 30a79847 se nalazi na lokaciji 0x 1000. Koja je vrednost bajta na adresi 0x 1002 ako je sistem big endian, a koja ako je little endian?

Odg:Ako je big endian ond je 98, a ako je little endian onda je a7.

Cvija

Rešenje:

Kao jednu lokaciju uzimaš dva znaka. (30 a7 98 47)

Za big endian(prvo ide par bajtova sa najvećom težinom)ideš ovako:
lokacija 1000 - 30
lokacija 1001 - a7
lokacija 1002 - 98
lokacija 1003 - 47
pošto ti treba lokacija 1002 pogledaš koja je njena vrednost i to upišeš.

Za little endian(prvo ide bajt sa najmanjom težinom) ide obrnuto:
lokacija 1000 - 47
lokacija 1001 - 98
lokacija 1002 - a7
lokacija 1003 - 30
vrednost lokacije 1002 sad je a7 i gotov zadatak.

2. Koja ce decimalna vrednost biti u 16-bitnom registru posle izvesavanja sledeceg niza instrukcija:
1. sub r1 r1 r1 (oduzimanje)
2. add r1 r1 7 (sabiranje)
3. and r1 r1 5 (1 i 1 = 1 ostalo je 0)
4. shr r1 r1 1 (pomeranje u desno)

Rešenje:

0000 0000 (uvek se stavi na pocetku 8 nula)
add 0000 0111 (7 binarno,add je sabiranje)

0000 0111
and 0000 0101 (and radi isto kao logicki i, odnosno: samo 1 i 1 je 1, sve ostalo je 0)

0000 0101
shr 0000 0010 (pomeranje u desno za 1, dodaje se nula ispred, a brise zadnja cifra)

Pošto piše da treba da se predstavi decimalno verovatno treba 0000 0010=2

3. Navesti sve mogucnosti za broj razlicitih instrukcija i broj razlicitih memorijsih lokacija koje se mogu adresirati ako imate na raspolaganju 2 adresni tip instrukcija duzine 3 bajta i ako je operacioni kod 6 bita ?

Rešenje:

3B=3*8=24b (pretvaranje bajtova u bitove)
24-6=18 (oduzima se duzina operacionog koda)
Znaci treba rasporetiti 2^18

2^17 memoriskih lokacija 2^1 instrukcija
2^16 memoriskih lokacija 2^2 instrukcija
2^15 memoriskih lokacija 2^3 instrukcija
.
.
.
2^1 memoriskih lokacija 2^17 instrukcija (sve kombinacije)

4.(3.11.)Šta je pakovanje(ubacivanje)bitova?Objasnite to na primeru kad je data reč oblika:
1100 1100 1111 0000 0101 0101 1100 0011 (A) a treba da se dobije:
1100 1100 1111 0000 1101 1111 1100 0011 ((A and B) or C)

Rešenje:

1100 1100 1111 0000 0101 0101 1100 0011 (A) (Dat je na početku ubacivanje bitova)
1111 1111 1111 1111 0000 0000 1111 1111 (B) (B je razlika imeđu početnog i onog što treba da se dobije.Petom i šestom redu od četri bita sa leve strane,znači samo tu idu nule,ostalo 1)

1100 1100 1111 0000 0000 0000 1100 0011 (A and B) (and,gde su:0i0=0,0i1=0,1i0=0, 1i1=1)
0000 0000 0000 0000 1101 1111 0000 0000 (C) (C dobijamo tako sto vidimo gde treba da su jedinice u onome sto se dobija a na ostalim mestima ide 0)

1100 1100 1111 0000 1101 1111 1100 0011 ((A and B) or C)(or,gde su:0i0=0,0i1=1,1i0=1,1i1=1)

5.Dati sistem ima 64-bitne virtuelne adrese,36-bitne fizičke adrese i 2 GB glavne memorije. Ako sistem koristi stranice veličine 4096 bajtova (4 KB), koliko virtuelnih i fizičkih stranica sistem može da podrži? Koliki je mogući broj okvira stranica u glavnoj memoriji?

Rešenje:
Broj okvira stranica u glavnoj memoriji = kapacitet glavne memorije : velicina stranice, tj. 2*1024*1024KB : 4KB = 524 288 okvira stranica
Broj virtuelnih stranica = 2^64 : 4096 = 2^64 : 2^12 = 2^52
Broj fizickih stranica = 2^36 : 2^12 = 2^24

6.Šta je izvalcenje a sta je pakovanje bitova??

Rešenje:
To su unarne operacije premestanja bitova. Kod izvlacenja bitova, daju ti rec koju trebas da ispomeras tj. da izvuces odredjene bite da bi dobio vrednost koju ti takodje daju.Na primer: reč A: 1100 1010 1010 1010 0111 1101 0110 0110 , a treba da dobijes:
0000 0000 0000 0000 0000 0000 0111 1101 .
Ti sad uzmes početnu reč A i pogledas sta trebas da izvuces (drugi bit sa desne strane) i na osnovu toga pravis masku B tako sto svuda stavljas nule sem na deo koji je promenjen u krajnjoj vrednosti gde stavljas jedinice:
B maska : 0000 0000 0000 0000 1111 1111 0000 0000 , sada koristis operaciju A and B i dobijes:
0000 0000 0000 0000 0111 1101 0000 0000 , poslednji deo zadatka je da bitove pomeris za onoliko mesto koliko je potrebno da bi vrednost bila ista sa trazenom, u ovom slucaju: SHR8(pomera se udesno za 8 mesta):
0000 0000 0000 0000 0000 0000 0111 1101 . I to je to sto treba da se uradi...
Kod pakovanja bitova se umesto jedinica na promenjeni deo stavljaju nule a na ostatak jednice i posle operacije "and" se dodaje rec C koja sadrzi nule i bite koji su dati kao trazena vrednost, a nakon toga ide (A and B) OR C...

EAX = akumulator, EDX = i/o pokazivac, a EIP = brojac naredbi.

7.Neka je zadat sadrzaj akumulator datog procesora:EAX = 12345678(16)
Prikazati sadzaj AX, AH, AL....Kako se ovo radi? Objasnjenje?

Resenje: AX = 5678(16), AH = 56(16), AL = 78(16)

AX=AH AL
AH je bajt najvece tezine u okviru
AL bajt najmanje tezine

8.Navesti sve mogućnosti za broj različitih instrukcija i broj različitih memorijskih lokacija koje se mogu adresirati ako imate na raspolaganju dvoadresni tip instrukcija dužine tri bajta i ako je opkod dužine 5 bita? Broj instrukcija = 2na5 = 32.

Rešenje:
Pa imas operacioni kod i adrese, ako kazu da su dali dvoadresni tip instrukcija 3 bajta to prebacis u bite i to je 3*8=24 bita, pa kazu da je OPK duzine 5 bita, to je onda 24-5=19, i od tih 19 pravis kombinacije.

32 instrukcije, broj adresa 2^18 i 2^1
32 instrukcije, broj adresa 2^17 i 2^2
32 instrukcije, broj adresa 2^16 i 2^3
.....
32 instrukcije, broj adresa 2^1 i 2^18

9. Navesti sve mogućnosti za broj različitih instrukcija i broj različitih memorijskih lokacija koje se mogu adresirati ako imate na raspolaganju jednoadresni tip instrukcija dužine 4 bajta.

Rešenje:
Kod jednoadresnog ova 4 bajta su ti 32 bita, a to posto je samo jedna adresa onda pravis kombinacije za OPK i tu jednu adresu.
broj instrukcija 2^31 i broj adresa 2^1
broj instrukcija 2^30 i broj adresa 2^2
.....
broj instrukcija 2^1 i broj adresa 2^31

10. Navesti sve mogucnosti za broj razlicitih mem. lokacija koje se mogu adresirati ako imate na raspolaganju dvoadresni tip instrukcije duzine 8 bajtova i ako je operacioni kod duzine 5 bita

Rešenje:

dvoadresni 8bajta=64 bita -5 za OPK = 59 i od ta dva pravis kombinacije za te dve adrese.
broj razlicitih instrukcija je 2^5

adresa 1	adresa 2
2^1	2^58
2^2	2^57
. .	
. .	
. .	
2^58	2^1

11. Neka programski prevodilac koristi memorijske lokacije počev od lokacije 0 za čuvanje vrednosti promenljive x i neka koristi big endian predstavljanje podataka. Prikazati sadržaj ovih memorijskih lokacija u binarnom obliku ako program ima sledeću naredbu int x = 17
Dužina memorijske lokacije je 1 bajt i x je 32-bitni broj

Rešenje:

1bajt=8bita
32-bitni broj/8bita=4bajta(4memoriske lokacije)

lokacija:	sadrzaj:
3	10001
2	00000
1	00000
0	00000

Ovo 10001 je broj 17 napisan binarno...mada bih ja napisala ovako 0001 0001 jer je bajt u pitanju, da su 2 bajta u pitanju bilo bi 0000 0000 0001 0001 a posto je 32-bitni broj ide se do 3, tj. nula, jedan, dva, tri...da je 16-bitni broj i jedan bajt u pitanju bilo bi samo nula i jedan i kod jedan bi stajao 17 binarno...

12.32-bitni broj 12345678 (heksadecimalno) se nalazi u memoriji cija je duzina jedne lokacije 8 bita, koristeci "big endian" metod pocev od lokacije sa adresom 100.Koji deo datog broja se nalazi na lokaciji sa adresom 102 ?

Rešenje:

big endian metod znaci da se cita s leva na desno (jer su na levoj strani bitovi koji nose vecu vrednost a sa desne manje - zato je big endian). Ako je 12345678 32-bitni broj a duzina lokacije 8 bita, znaci da jednu lokaciju zauzimaju 2 cifre (posto ce 32-bitni broj 12345678 u memoriji sa 8-bitnom celijom zauzeti 4 mem. lokacije (celije) => 8 cifara/4 lokacije = 2 cifre po lokaciji). Posto je prva adresa 100 ona ce nositi 12, 101 ce nositi 34, a 102 ce 56, 103 verovatno 78...(posto je big endian). Da je zadatak bio isti, samo little endian, onda bi bilo 100:78, 101:56, 102:34...)

13. (3.5)

Rešenje:

Moguce je kad je 2^(duzina adresnog registra memorije)>=broj lokacija u memoriji, a onda, ako je to ispunjeno, ima smisla kad je veci broj lokacija u memoriji(uglavnom preko hiljadu), a relativno mala duzina lokacija(8-64 uglavnom),
Recimo zadatak 3.5 iz knjige: pretposlednja nema smisla zato sto ima malo lokacija u memoriji, a prevelika je njihova duzina; a poslednja ima jako malo lokacija u memoriji, a postoji mogucnost za mnogo vise, cak 2^32.

14. Racunar ima instrukcije duzine 16 bita .Koristi jednoadresne, dvoadresne i troadresne instrukcije. Svaka adresa 4 bita, pa sad ako se korsite i jednoadresne, i dvoadresne i troadresne treba odrediti maksimalan broj troadresnih instrukcija i obrazloziti odgovor.

Rešenje:

Duzina instrukcije je 16bita (dato u zadatku).
3 adrese po 4 bita, to je ukupno 12 bita.
Ostaju 4 bita za op. kod
To je ukupno 2^4 mogucnosti,ali se oduzimaju 2, koje idu na oznake za jednoadresnu i dvoadresnu instrukciju, tako da je max broj troadresnih instrukcija 14.

15. Pomocu instrukcija PUSH, POP, ADD napisati C=A+B.

Rešenje:

push A

push B

ADD

pop C

16. Ako je za citanje instrukcije iz memorije potrebno 5ns, za dekodiranje 2ns, za citanje registarske datoteke 3ns, za izvrsavanje instrukcije 3ns i za upisivanje rezultata u registarsku datoteku 2ns, kolika je maksimalna brzina takta procesora?

Rešenje:

Ovo ti je lako podelis 1 sa zbirom ovih brojeva ovako: 1/(5+2+3+3+2)= 1/15 ns pa pretvoris MHz a to ti je 1/15ns=66,6Mhz

Tipovi izvršnih jedinica u IA-64 ?

I-jedinica (za celobrojnu aritmetiku,logičke instrukcije, instrukcije poređenja, ...)

M-jedinica (učitava i skladišti podatke između registra i memorije)

B-jedinica (instrukcije grananja)

F-jedinica (instrukcije za rad u pokretnom zarezu)

Spekulativno upravljanje kod IA-64 ?

Spekulativno upravljanje kod IA-64 omogućava procesoru da podatke iz memorije učita pre nego što programu budu potrebne da bi se izbeglo kašnjenje spore memorije.

Instrukcije u arhitekturi IA-64:

SHR (shift right,pomeranje u desno),

SHL (shift left,pomeranje u levo),

PADD (paralelno sabiranje),

PSUB (paralelno oduzimanje),

ST (store,podatak prenosi iz procesora u memoriju),

LD (load,podatak prenosi iz memorije u procesor),

CMP (compare,porede dve ili više zadate vrednosti),

JP ili JMP-˝jump˝= skoči (bezuslovno grananje),

BEQ - ˝Branch if Equal˝ (uslovno granjanje na osnovu vrednosti postavljenih flegova),

HALT (zaustavljanje rada programa).

XCHG - ˝Exchange ax,bx˝ (u ax se upiše ono što je u bx,a u bx ono sto je u ax)

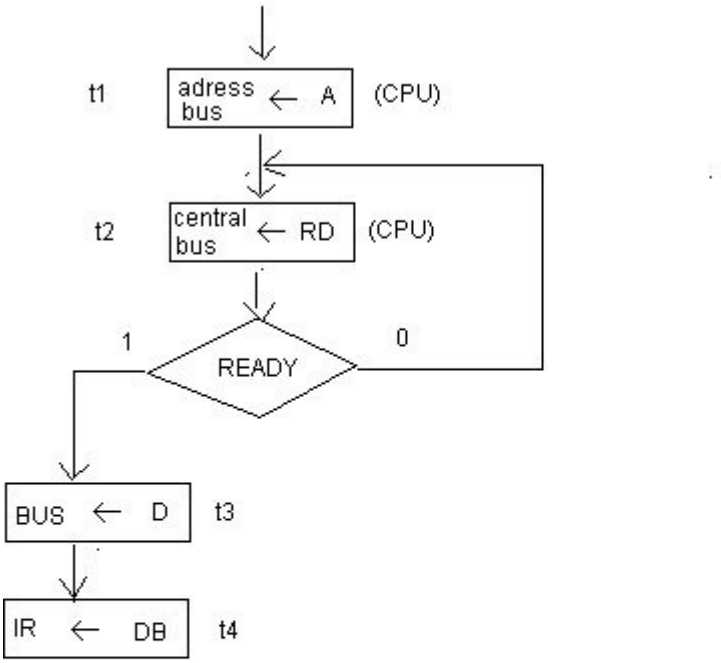
BR - ˝Branch˝ (skok)

MOV - ˝Move˝ (transfer podataka među registrima)

Skripta za prvi deo(Teorija+Urađeni zadaci)

IA -64 INSTRUKCIJE:

- Aritmetičke instrukcije (add, sub...)
- Logičke instrukcije (and, or, xor...)
- Instrukcije pomeranja (shl, shr,)
- Instrukcije poređenja (cmp)
- Instrukcije za pristup memoriji (ld, st, xchg)
- Instrukcije za transfer podataka među registrima (mov)
- Instrukcije skoka (br)
- SIMD Instrukcije (padd, psub)
- Instrukcije za rad sa realnim brojevima



Cvija