Softver za osnovni računar

Računarski softver

- Računarski sistem sastoji se iz hardvera i softvera
  - Hardver su fizičke komponente i oprema
  - Softver su programi koji su napisani za računar
- Mašinski jezik, težak za korišćenje, prevodilac je program koji prevodi programe sa “višeg” jezika na mašinski

Asemblerski jezik

- Asemblerski jezik, zapisivanje naredbi pomoću skraćenica, adrese se zapisuju pomoću imena promjenljivih
- Asembler je sistemski program koji prevodi program sa asemblerskog jezika na odgovarajući program napisan u mašinskom jeziku

Kompajler ili prevodilac

- Prevodilac je sistemski program koji prevodi tekst programa na “višem” programskom jeziku u ekvivalentni program na mašinskom jeziku
- Ulazni podatak za kompajler je tekst programa P_1 a njegov izlazni podatak je tekst programa P_2 na nekom drugom jeziku, čiji je nivo niži

Primjer

a)

adresa	naredba
0	0010 0000 0000 0100
1	0001 0000 0000 0101
10	0011 0000 0000 0110
11	0111 0000 0000 0001
100	0000 0000 0101 0011
101	1111 1111 1110 1001
110	0000 0000 0000 0000

b)

	LDA A	donesi u akumulator broj sa adrese A
	ADD B	uvećaj akumulator za broj sa adrse B
	STA C	broj iz akumulatora pošalji na adresu C
	HLT	zaustavi računar
A,	DEC 83	
B,	DEC −23	
C,	DEC 0	

c)

```
program progra;
var a,b,c: integer;
begin
a:=83; b:=-23;
c:=a+b
end.
```

Sistemski i aplikativni softver

- Sistemski softver sastoji se od niza izvršnih programa koji treba da učine da korišćenje računara bude lako i efikasno
  - Operativni sistem je skup programa koji omogućavaju korišćenje hardvera, softver najbliži hardveru, posredstvom operativnog sistema ostali programi pristupaju hardveru, jednokorisnički (DOS) ili višekorisnički (Unix)
- Aplikativni softver su programi napravljeni za konkretne ciljeve korisnika računara

Primjeri programa za ulaz i izlaz

- Dva načina, programsko prenošenje ili prenošenje pomoću sistema prekida
- Programsko prenošenje, računar čeka dok se ne završi ulazno-izlazna komunikacija, dio koda koji realizuje učitavanje jednog karaktera

```
240  SKI       (prekoči jednu naredbu ako je INPR pun)
241  BUN 240   (bezuslovni skok na prethodnu adresu)
242  INP       (prepiši iz INPR u AC)
```

Primjer

- Program koji bez korišćenja prekida štampa sadržaj lokacija 298 i 299

```
→ 200   IOF        IEN ← 0 interrupt off
  201   LDA 298    AC ← c(298)
  202   OUT        OUTR ← AC(9–16) print
203 CIR 204 CIR 205 CIR 206 CIR 207 CIR 208 CIR 209 CIR 210 CIR right shift
right shift right shift right shift right shift right shift right shift right shift
  211   SKO        if FGO = 1 then PC ← PC + 1 skip on output flag
  212   BUN 211    PC ← 211 goto 211
  213   OUT        OUTR ← AC(9–16) print
  214   LDA 299    AC ← c(299)
  215   SKO        FGO = ?
  216   BUN 215    goto 215
  217   OUT        print
218 CIR 219 CIR 220 CIR 221 CIR 222 CIR 223 CIR 224 CIR 225 CIR
  226   SKO        FGO = ?
  227   BUN 226    goto 226
  228   OUT        print
  229   HLT        stop the computer
```

Ulaz/izlaz sa prekidima

- Sistem prekida temelji se na hadrveru i softveru, sporost perifernih uređaja je problem
- Karakteri sa tastature upisuju se u ulazni bafer koji je organizovan kao kružni red
- Program upisuje podatke u izlazni bafer, iz njega se podaci šalju na štampač po mjeri spremnosti štampača
- Ključna uloga je rutina za obradu prekida R

Ulaz/izlaz sa prekidima (2)

- Neka rutina zauzima adrese 100 <= *a* <= 199 za svoje naredbe i podatke, pri tome
  - Ulazni bafer adrese 100 – 109
  - Izlazni bafer adrese 110 – 119
  - Ulazna tačka prekidne rutine R je *a* = 120
  - Adresa *a* = 0 čuva povratnu adresu (tj. PC) iz rutine R
  - Adresa *a* = 1 sadrži naredbu skoka na rutinu
  - Adresa *a* = 198 čuva AC
  - Adresa *a* = 199 čuva E

Ulaz/izlaz sa prekidima (3)

- Ulazni bafer
  - Adresa *a* = 100 gdje se nalazi prvi član bafera, *ib*
  - Adresa *a* = 101 broj članova bafera, *in*
  - Sami članovi bafera od *a* = 102 do *a* = 109
  - To znači da je 0 <= *in* <= 8, ako je *in* = 0 bafer je prazan, ako je *in* = 8 bafer je pun, uvijek je 102 <= *ib* <= 109 osim kada je bafer prazan, podaci se u bafer unose po redu, nakon upisa na 105 upisuje se u 106, bafer je kružan što znači da se nakon 109 upisuje u 102

Ulazni bafer

- Prvi član koji je upisan u bafer će prvi iz njega i izaći, ako je bafer prazan neka bude *ib* = 0, prekidna rutina R puni bafer a program P ga prazni, ako je npr. *ib* = 102 i *in* = 4, u baferu imamo 4 člana i oni su redom 102, 103, 104 i 105, ili ako je *ib* = 108 i *in* = 4, u baferu imamo 4 člana na adresama 108, 109, 102 i 103, prlikom svakog upisivanja i uzimanja iz bafera ažurira se *ib* i *in*, bafer je organizovan kružno da bi se njegov raspoloživi prostor maksimalno koristio

Izlazni bafer

- Izlazni bafer sadrži 10 lokacija 110 – 119, prve dvije za evidenciju, *ob* adresa prvog člana (output begin), *on* broj članova (output number), program P stavlja u izlazni bafer karaktere za štampanje, a rutina R uzima iz bafera i šalje štampaču

Prekidna rutina

- Osnovni koraci prekidne rutine R

  a) Registre AC i E sačuvati u memoriji: STA 198, CLA, CIL, STA 199

  b) Ako je FGI = 0 pređi c, ako je *in* = 8 ulazni bafer je pun pa daj zvučni signal i pređi na c, ako je FGI = 1 i ako je *in* < 8 tada rutina R preuzima jedan karakter, dva slučaja, ako je *in* = 0 onda, uz pretpostavku da je na 197 upisana vrijednost 102: LDA 197, STA 100, CLA, INC, STA 101, INP, STA 102, ako je *in* > 0: LDA 101, INC, STA 101, ADD 100, STA 196, INP, STA* 196

Prekidna rutina (2)

c) ako je FGO = 0 onda pređi na d, ako je FGO = 1 (štampač je spreman) i *on* = 0 (izlazni bafer je prazan) pređi na d, ako je FGO = 1 i *on* > 0 izvrši naredbu OUT, ažuriraj *ob* i *on* i pređi na d

d) vratiti prethodno sačuvane vrijednosti sa 198 i 199 u AC i E

e) naredba ION, efekat IEN = 1

f) naredba BUN* 0, efekat PC = sadržaj(0) čime se obnavlja izvršavanje programa P

- Napomena: program P nema naredbu za ulaz INP, nego koristi naredbu LDA za preuzimanje iz ulaznog bafera u AC, slično nema naredbu OUT, nego koristi naredbu STA sa stavljanje u izlazni bafer

Primjer

- Program treba da odštampa nižih 8 bita sa lokacija 296 do 299 (ukupno 4 lokacije), a zatim da izvrši veliki broj aritmetičkih operacija
  - Umjesto da sam štampa i čeka završetak štampanja za svaki karakter, program će, koristeći mogućnost prekida, upisati podatke u bafer i uposliti prekidnu rutinu da štampa
  - U lokacije 294 i 295 upisati 112 i 4
  - Prekidna rutina mora sa bude učitana i zauzima 100-199
  - Program startovati sa adrese 200

Rješenje

```
     1   BUN 120    goto 120, stvoreni su preduslovi za skok na R
→  200   IOF        IEN ← 0, interrupt off
   201   CLA        AC ← 0, clear AC
   202   STA 100    c(100) ← AC, ib ← 0
   203   STA 101    c(101) ← AC, in ← 0 (ulazni bafer je prazan)
   204   LDA 296    AC ← c(296), počinje upisivanje u izlazni bafer
   205   STA 112    c(112) ← AC
   206   LDA 297    AC ← c(297)
   207   STA 113    c(113) ← AC
   208   LDA 298    AC ← c(298)
   209   STA 114    c(114) ← AC
   210   LDA 299    AC ← c(299)
   211   STA 115    c(115) ← AC, četiri vrijednosti su upisane u izlazi
   212   LDA 294    AC ← c(294) = 112
   213   STA 110    c(110) ← AC, ob ← 112, ažuriranje
   214   LDA 295    AC ← c(295) = 4
   215   STA 111    c(111) ← AC, on ← 4, ažuriranje
   216   ION        IEN ← 1, interrupt on
   217   ADD 254    počinju aritmetičke i logičke operacije
   218   ADD 253
   219   AND 252
   220   AND 251
   . . . . . .
   294   DEC 112    ova lokacija neka sadrži brojnu vrijednost
   295   DEC 4      ova lokacija neka sadrži brojnu vrijednost
```

Periferni uređaji

- Osnovni uređaj ima dva periferna uređaja
  - Svaki od njih po jedan fleg: FGI, FGO
  - Tokom prekidnog ciklusa izvršiće se skok na prekidnu rutinu R, ako je makar jedan fleg setovan i ako su flegovi dozvoljeni
  - Flegovi informišu procesor koji uređaj želi prekid, izdvojiti jedan od više zahtjeva
  - Mogućnost prekidanja prekidne rutine, čuvanje adresa i registara nije više moguće na fiksnim adresama već na sistemski stek

Primjer aritmetičkog programa

- Program računa y = 2a+b+1, program se učitava od adrese 200, podaci od adrese 210

```
200 LDA 210
201 CIL
202 ADD 211
203 INC
204 STA 212
205 HLT
210 DEC 300   (a = 300)
211 DEC 44    (b = 44)
212 DEC 0     (y = 0)
```

Zadaci za vježbu

- Zadatak 1. Računa se y = a + b + 10, za c < 0 i y = a*a + b*b + 100, inače
- Zadatak 2. Računa se y = a + (a+1) + … + b, gdje su a, b ulazni podaci
- Zadatak 3. Sastaviti program koji učitava niz od 10 brojeva na lokacije 1001-1010, a onda računa i štampa sumu niza

Rad sa potprogramom

- Sastaviti program koji računa y = (a/16 + b)/16, adrese za program 200-208, početna adresa 200, adrese za potprogram 250-255, potprogram računa dijeljenje sa 16, povratna adresa iz potprograma je upisana na 250, posljednja naredba potprograma je BUN* 250, pozivanje potprograma sa BSA 250 (čuvanje povratne adrese na 250 uz skok na lokaciju 251, razmjena parametara preko AC, moguće preko nekoliko adresa u potprogramu ili nekoliko fiksiranih apsolutnih adresa

Rad sa potprogramom (2)

```
200 LDA 206
201 BSA 250
202 ADD 207
203 BSA 250
204 STA 208
205 HLT
206 DEC 900
207 DEC 800
208 DEC 0
250 DEC 0
251 CIR
252 CIR
253 CIR
254 CIR
255 BUN*250
```

Loader

- Loader je program koji učitava druge programe u memoriju, apsolutni loaderi, loaderi sa relokacijom

Apsolutni loader

- Kada u trenutku prevođenja programa znamo apsolutne adrese, bukvalno kopiranje programa sa ulaza ili diska u memoriju, nakon učitavanja počinje izvršavanje programa
- Pretpostavimo da loader zauzima 0-99, učitavanje loadera nije programsko, tokom rada loadera je IEN=0, u trenutku startovanja računara je PC=0, ulazni podaci loadera su adrese djelova programa za kod i podatke i početna adresa

Apsolutni loader (2)

- Loader čita u grupama po 4 karaktera

00C8 20D2 7040 10D3 7020 30D4 7001 7777
00D2 012C 002C 0000 7777 FFFF 00C8

- Prva cjelina definiše mjesto gdje pošinje učitavanje, slijede naredbe, cjelina se završava sa 7777, oznaka FFFF znači da više nema cjelina za učitavanje, jedino ostaje da se naznači početna adresa, na kraju loader izvršava bezuslovni skok

Apsolutni loader (3)

- Objašnjenje, load point je tačka punjenja lp, c(adr) je sadržaj lokacije adr

1. read four; if four = FFFF goto 3; lp ← four
2. read four; if four = 7777 goto 1; c(lp) = four; lp = lp + 1; goto 2;
3. read four; PC = four;

- Posle završetka rada loadera, prostor 0-99 može se prebrisati

Loader sa relokacijom

- Znati memorijsku lokaciju na kojoj će se program nalaziti u vrijeme pisanja programa predstavlja veliko ograničenje, današnji računari dozvoljavaju da se u memoriji nalazi više programa, nemoguće je unaprijed znati dio memorije u koji se smješta program
- Program se piše kao da počinje sa adrese 0, u trenutku učitavanja svaka adresa uveća se za pomjeraj delta koji se određuje prije učitavanja, *relokacija*

Loader sa relokacijom (2)

- Za osnovni računar relokacija se realizuje uvećavanjem za delta adresnog polja 5-16 svake naredbe prve vrste, konkretno za naredbu ADD a, relokacijom se podešava da se drugi sabirak uzima sa adrese a + delta umjesto sa a, relokacija naročito važna za potprograme jer unaprijed ne znamo koliki je glavni program, koliko ima potprograma i koliki su, koja je eventualna zajednička zona u memoriji itd.

Loader sa relokacijom (3)

- Za jedan odsječak ulaz u loader sa relokacijom

lp delta four four … four 7777

- Na primjer, ulaz može da bude

00C8 0010 20D2 … 7001 7777 00D2 0000 012C 002C 0000 7777 FFFF 00C8

- Rezultat

200 LDA 226; 201 CIL; 202 ADD 227; 203 INC; 204 STA 228; 205 HLT

Bootstrap loader

- Program koji pokreće računar naziva se inicijalni ili startni loader (bootstrap loader)
- Ovaj program radi kada je memorija prazna, odnosno kada je samo on učitan u memoriju
- Čuva se u ROM memoriji, njegovo izvršavanje počinje pritiskom dugmeta za startovanje računara
- Najčešće samo učitava jedan veći program i započinje njegovo izvršavanje, u slučaju DOS-a učitava se tzv. boot block (prvi sektor na disku) i započinje se njegovo izvršavanje, on dalje provjerava periferne uređaje, inicijalizuje registre itd.

Asembler

- Zadatak asemblera je da generiše mašinski kod P_2 od programa P_1 koji je napisan na jeziku asemblera
- Primjer programa P_1, y = 2a + b + 1

```
ORG 200
LDA a
CIL
ADD B
INC
STA c
HLT
a, DEC 300
b, DEC 44
c, DEC 0
END
```

Asembler (2)

- Jedan red jedna naredba, naredba može da bude pseudo-naredbe ili obična naredba
- Pseudo-naredbe ORG, ENC, DEC, HEX
  - ORG n, znači da program P_2 zauzima adrese n, n+1, …
  - END označava kraj teksta programa P_1
  - DEC n, znači da na odgovarajućem mjestu u P2 treba da piše $(n)_{10}$
  - HEX n, slično samo što je sada $(n)_{16}$

Asembler (3)

- Obične naredbe sastoje se od tri polja

  labela opkod adresa
- Polje labela je opciono
- Polje opkod označava kod operacije, npr. AND, AND*, ADD, ADD*, …, CLA, CLE, …, INP, OUT
- Polje adresa odsustvuje za naredbe druge i treće vrste, primjeri naredbi sa adresom: AND a, AND* a, BUN MJESTO, BUN* MJESTO, simboličke adrese a, MJESTO
- Rad u dva prolaza, prvi za formiranje tabele simbola, drugi za genrisanje mašinskog koda koristeći tabelu simbola

Sistemski softver osnovnog računara

- Bootstrap loader, unošenje ručno pomoću switch registra SR, ili da se čuva u ROM-u, ili u nekom malom RAM-u, ili da se učitava sa bušenih kartica
- Apsolutni loader, zauzima 2-99, stalno prisutan u memoriji
- Loader sa relokacijom
- Rutina za obradu prekida, zauzima 100-199, stalno prisutna u memoriji
- Asembler
- Debager, zauzima 3000-4095
- Potprogram za množenje
- Potprogrami za podršku ulazu i izlazu