SISTEMSKI SOFTVER SI3SS,
SISTEMSKO PROGRAMIRANJE IR3SP

JUN-JUL 2016.

Opis okruženja

Projekat se radi pod operativnim sistemom Linux na x86 arhitekturi u programskim jezicima C, C++ ili asembleru (moguća je i kombinacija). Potrebi alati su opisani u materijalima za predavanja i vežbe.

Napisati dvoprolazni asembler za procesor opisan u prilogu. Ulaz asemblera je tekstualni fajl u skladu sa sintaksom opisanom u nastavku. Izlaz asemblera treba da bude predmetni program zapisan u tekstualnom fajlu. Format predmetnog programa bazirati na školskoj varijanti elf formata (tekstualni fajl kakav je korišćen u zadatku 9 u prezentaciji V3_Konstrukcija asemblera.ppt) i predložiti izmene u formatu u skladu sa potrebama ciljne arhitekture (nove sekcije, novi tipovi zapisa o relokacijama, dodatna polja u postojećim tipovima zapisa i sl.). Prilikom generisanja izlaznog fajla voditi se principima koje koristi GNU asembler.

Sintaksa asemblera i ostali zahtjevi:

- u jednoj liniji može biti najviše jedna komanda,
- na početku svake linije može da stoji labela koja se završava dvotačkom,
- labela može da stoji i u praznoj liniji i tada je njena vrednost jednaka adresi prve sledeće instrukcije,
- simboli mogu da se izvoze tako što se na početku fajla navede direktiva:

.public <ime globalnog simbola>,...

- simboli mogu da se uvoze tako što se na početku fajla navede direktiva:

.extern <ime uveženog simbola>,...

- u jednoj direktivi može da se navede i više globalnih naziva koji su odvojeni zapetama,
- izvorni kod je podeljen u sekcije:
- .text - sekcije koje sadrže mašinski kod,
- .data - sekcije koje sadrže inicijalizovane podatke,
- .bss - sekcije koje sadrže neinicijalizovane podatke,
- svaka od sekcija iza naziva može da ima i podnaziv odvojen tačkom. Osnovni naziv sekcije određuje tip sekcije, dok podnaziv omogućava pravljenje više različitih sekcija istog tipa (npr. .text.prvideo, .text.drugideo, .text su 3 različite sekcije istog tipa)
- fajl sa izvornim kodom se završava direktivom .end. Ostatak fajla se odbacuje (ne prevodi se),
- pored mnemonika koji su definisani treba obezbediti direktive .char, .word, .long, .align i .skip sa istim funkcionalnostima kao u GNU asembleru,
- ostatak sintakse definisati po sopstvenom nahođenju.

Opis dela 32-bitnog procesora potrebnog za izradu domaćeg zadatka

U nastavku je opisan deo 32-bitnog dvoadresnog procesora sa Von-Neuman arhitekturom. Adresibilna jedinica je bajt, a raspored bajtova u reči je little-endian. Veličina adresnog prostora je 2^{32}B. Procesor poseduje 16 opštenamenskih 32-bitnih registara označenih sa r0-r15. Pored opštenamenskih registara postoje registari: PC (programski brojač), SP (pokazivač na vrh steka), LR (registar u kojem se čuva povratna adresa prilikom poziva potprograma i skoka na prekidnu rutinu) i PSW (statusna reč procesora). Stek raste ka nižim adresama, a SP pokazuje na zauzetu lokaciju na vrhu steka. Statusna reč procesora. PSW u najniža 4 bita sadrži flegove: Z – rezultat prethodne operacije je 0, O – prekoračenje, C – prenos i N – rezultat je negativan. Najviši bit u PSW služi za globalno maskiranje prekida.

Počev od adrese 0 nalazi se IVT sa 16 ulaza. Svaki ulaz zauzima 4 bajta i sadrži adresu odgovarajuće prekidne rutine. Na ulazu 0 se nalazi adresa prekidne rutine koja se izvršava prilikom pokretanja, odnosno prilikom resetovanja procesora. Na ulazu 1 se nalazi adresa prekidne rutine koja se izvršava periodično sa periodom 1 sekund. Prekidna rutina na ulazu 1 se izvršava periodično samo ako je bit 30 u PSW registru postavljen na 1. U suprotnom se ne izvršava. Na ulazu 2 se nalazi adresa prekidne rutine koja se izvršava ako se pokuša izvršavanje nekorektne instrukcije. Na ulazu 3 se nalazi adresa prekidne rutine koja se izvršava kada se pritisne neki taster i tada je potrebno očitati kod pritisnutog tastera (potrebne adrese su prikazane u nastavku). Ostali ulazi su slobodni za korišćenje od strane programera.

Sve instrukcije su veličine reči (4 bajta) i u svakoj se može navesti uslov pod kojim se instrukcija izvršava. Uslov se navodi u prva 3 bita (tabela sa kodovima data na kraju), naredni bit govori da li treba menjati utiče na promenu flegova u skladu sa instrukcijom (vrednost 0 znači da flegovi u svakom slučaju ostaju neizmenjeni, dok za vrednost 1 flegovi se menjaju ukoliko ih posmatrana instrukcija menja), dok je narednih 4 bita rezervisano za operacioni kod instrukcije. Ostatak instrukcije zavisi od tipa i opisan je u narednoj tabeli.

<table>
<tr><th>mnemonik</th><th>oc</th><th>operandi</th><th>Efekat</th><th>Flegovi koji se menjaju</th></tr>
<tr><td>int</td><td>0</td><td>src:4,nu:20</td><td>src - broj prekida koji se pokreće
push PSW
LR := PC</td><td>-</td></tr>
<tr><td>add</td><td>1</td><td rowspan="5">dst:5*1, 0:1, src:5*1, nu:13
dst:5*1, 1:1, imm(18)
PSW nije dozvoljen
Za PC, LR i SP samo add i sub</td><td rowspan="4">reg[dst] := reg[dst] OP reg[src]
ili
reg[dst] := reg[dst] OP signExt (imm)</td><td rowspan="2">zocn</td></tr>
<tr><td>sub</td><td>2</td></tr>
<tr><td>mul</td><td>3</td><td rowspan="2">zn</td></tr>
<tr><td>div</td><td>4</td></tr>
<tr><td>cmp</td><td>5</td><td>kao oduzimanje ali se ne menja reg[dst]</td><td>zocn</td></tr>
<tr><td>and</td><td>6</td><td rowspan="2">dst:5*1, src:5*1, nu:14</td><td>reg[dst] := reg[dst] & reg[src]</td><td rowspan="2">zn</td></tr>
<tr><td>or</td><td>7</td><td>reg[dst] := reg[dst] | reg[src]</td></tr>
</table>

not	8	PC, LR i PSW nisu dozvoljeni	reg[dst] := ~reg[src]	
test	9		reg[dst] & reg[src]	

ldr/str	10	a:5[*1], r:5[*1], f:3, l/s:1, imm:10	if (f==4) a += 4; elsif (f==5) a -= 4;
			reg[r] <-> mem[reg[a]+signExt(imm)]
			if (f==2) a += 4; elsif (f==3) a -= 4;
			Ako je adresni registar PC, f mora biti 0.
			PSW ne može biti adresni registar.
			l/s == 1 -> ldr
			l/s == 0 -> str
				-

call	12	dst:5[*1], imm:19	LR := PC (adresa sledeće instrukcije) goto reg[dst]+signExt(imm)	
in/out	13	dst:4, src:4, i/o:1, nu:15	reg[dst] <-> io[reg[src]] i/o == 1 -> in	
mov/shr/shl	14	dst:5[*1],src:5[*1],imm:5, l/r:1, nu:8	reg[dst] := reg[src] (<<|>>) imm (unsigned) Ako je odrediste PC, i setovan bit za računanje flegova, sa steka se skida PSW (implementacija IRET instrukcije). l/r == 1 -> left	zcn
ldch/l	15	dst:4,h/l:1, nu:3,c:16	reg[dst] upisuje c u viših 16 bitova za h/l=1, u suprotnom c upisuje u nižih 16 bitova (u viših 16 se upisuje 0). U asembleru postoje ldch i ldcl instrukcije, kao i pseudo instrukcija ldc koja učitava čitavu reč u zadati registar (generiču se dve mašinske instrukcije, jedna za nižih 16 bitova i druga za viših 16 bitova)	-

*1 - vrednosti dst manje od 16 znače da se kao reg[dst] koristi neki od 16 opštenamenskih registara. Vrednosti preko 15 imaju sledeće značenje: PC - 16, LR - 17, SP - 18, PSW - 19.

Objašnjenja i dodatne napomene:

- Operandi su navedeni tako što je prvo naveden naziv polja, a zatim iza dvotačke i broj bita koje polje zauzima. Tako npr. za mov instrukciju postoji 8 polja. Prva tri su kod uslova, bit za upravljanje postavljanjem flegova i operacioni kod. Potom redom slijede dst (redni broj destinacionog registra zapisan u 5 bitova), src (redni broj izvorišnog registra zapisan u 5 bitova), pomjeraj koji zauzima 5 bitova, smjer pomjeranja 1 bit i poslednje polje veličine 8 bitova se ne koristi (nu – not used).

- Oznaka imm(4..0) predstavlja najnižih 5 bitova polja imm.

- Funkcija signExt() prosiruje operand znakom do veličine od 32 bita.

- Sve aritmetičke operacije se izvode tako da odgovaraju označenim celim brojevima.

- Instrukcije cmp i tst ne čuvaju direktni rezultat odgovrajuće operacije, već samo u skladu sa rezultatom postavljaju nove vrednosti flegova u PSW.

- Čitanje sa adrese 0x1000 u ulazno-izlaznom adresnom prostoru vraća sledeći pritisnuti taster na tastaturi, na adresi 0x1010 je statusni registar tastature u kojem bit 10 označava da li trenutno postoji pritisnut taster, dok upis na adresu 0x2000 prikazuje znak na tekućoj poziciji na ekranu i pomera tekuću poziciju za jedno mesto.

- uslovni kodovi su navedeni u narednoj tabeli:

eq	0	==
ne	1	!=
gt	2	> (označeno)
ge	3	>= (označeno)
lt	4	< (označeno)
le	5	<= (označeno)
	6	ne koristi se
al	7	bezuslovno