Alati za arhitekturu računara

Mladen Nikolić

1 SIM - simulator za mali mašinski jezik

1.1 Opis računara

Jednostavan, fiktivni računar za koji je mašinski jezik sasatavljen, raspolaže jednostavnim procesorom i unutrašnjom memorijom.

1.1.1 Memorija

Memorija sadrži 256 8-bitnih memorijskih lokacija.

1.1.2 Registri

Procesor sadrži sledeće registre:

- 8-bitni akumulator koji čuva rezultate operacija.
- Brojač instrukcija koji sadrži adresu instrukcije koja je na redu da bude izvršena.
- 4-bitni flags registar koji sadrži flag-ove koji signaliziraju razna značajna stanja.

1.1.3 Flag-ovi

Flag-ovi imaju sledeće skracenice i značenja:

- V - Overflow - Biva postavljen kad dodje do prekoračenja pri sabiranju ako se brojevi smatraju označenim, ukoliko se komplementira najmanji negativan broj i ako pomeranje ulevo promeni znak vrednosti akumulatora.
- S - Sign - Postavlja se ako je sadržaj akumulatora, shvaćen kao označen broj, negativan. U suprotnom se poništava.
- Z - Zero - Postavlja se ako je sadržaj akumulatora 0. U suprotnom se poništava.

- C - Carry - Postavlja se kad dodje do prekoračenja pri sabriranju ako se brojevi shvate kao neoznačeni. Takodje, u njega se upisuje vrednost poslednjeg istisnutog bita pri pomeranju vrednosti akumulatora.

1.1.4 Skup instrukcija

Za svaku instrukciju sledi skraćenica, numerička vrednost i značenje. Bitna informacija je i da li instrukcija zahteva argument. Njegovo značenje je odredjeno načinom adresiranja. Instrukcije koje zahtevaju argument su dvobajtne, dok su instrukcije koje ne zahtevaju argument jednobajtne.

- MUA 0x0 - Upisuje broj u akumulator. Broj može biti dat neposredno ili se može nalaziti u memoriji. Zahteva argument.
- AUM 0x1 - Upisuje sadržaj akumulatora na odredjenu memorijsku lokaciju. Zahteva argument. Ne dozvoljava neposredno adresiranje.
- ADD 0x2 - Sabira sadržaj akumulatora sa nekim brojem koji se, ponovo, može nalaziti u memoriji ili može biti dat neposredno. Rezultat ostaje u akumulatoru. Zahteva argument.
- CPL 0x3 - Komplementira sadržaj akumulatora u smislu potpunog komplementa. Ne uzima argument. Način adresiranja nije bitan.
- AND 0x4 - Izračunava bitsku konjunkciju sadržaja akumulatora i nekog broja koji može biti zadat slično instrukcijama MUA, ADD... Rezultat, naravno, ostaje u akumulatoru. Zahteva argument.
- XOR 0x5 - Izračunava bitsku ekskluzivnu disjunkciju sadržaja akumulatora i nekog broja. Zahteva argument.
- SHL 0x6 - Pomeranje sadržaja akumulatora za odredjeni broj mesta ulevo. Zahteva argument.
- SHR 0x7 - Aritmetičko pomeranje sadržaja akumulatora udesno. Zahteva argument.
- HALT 0x8 - Zaustavlja izvršavanje programa. Ne uzima argument. Način adresiranja nije bitan.
- JZ 0x9 - Prenosi izvršavanje programa na zadatu adresu ukoliko je postavljen zero flag. Argument je upravo adresa na kojoj se nastavlja izvršavanje.
- ADDC 0xA - Sabira sadržaj akumulatora sa datim brojem i sadržajem carry flag-a. Zahteva argument.
- FUA 0xB - Upisuje flags registar u akumulator u redosledu bitova: overflow, sign, zero, carry.

1.1.5 Načini adresiranja

Na raspolaganju su četiri načina adresiranja:

- Neposredno 0x1 - Argument je sadržan u drugom bajtu insturkcije.
- Direktno 0x0 - Argument se nalazi u memoriji na adresi koja je data u drugom bajtu instrukcije.
- Indirektno 0x2 - Drugi bajt instrukcije sadrži memorijsku adresu na kojoj se nalazi adresa na kojoj se nalazi argument.
- Relativno 0x3 - Drugi bajt sadrži udaljenje (pozitivno ili negativno) u odnosu na adresu tekuće instukcije čijim se sabiranjem dobija adresa na kojoj se nalazi argument.

1.1.6 Format instrukcije

Instrukcije se sastoje od jednog ili dva bajta. Drugi bajt predstavlja podatak koji se tumači u zavisnosti od načina adresiranja i vezan je za argument instrukcije. Prvi bajt ima binarni oblik: IIII00AA, gde su sa I označeni bitovi pomoću kojih se zapisuje operacioni kod instrukcije, a oni označeni sa A odredjuju način adresiranja.

1.2 Opis korišćenja simulatora

Simulator se pokreće iz komandne linije bez argumenata. Raspolaže skupom interaktivnih komandi za punjenje memorije, pokretanje programa i jednostavno debagovanje.

1.2.1 Komande

Komande se simulatoru jednostavno zadaju. Komanda se sastoji od jednog slova i može biti praćena argumentom. Sledi spisak komandi uz kratke opise. Sve adrese koje se navode moraju biti zadate kao dvocifreni heksadekadni brojevi.

- **q** (quit) - Prekida rad simulatora.
- **l** (load) - Puni memodiju sadržajem fajla. Ime fajla je obavezan argument ove komande.
- **r** (run) - Pokreće izvršavanje programa sa odredjene adrese koja je data kao obavezan argument.
- **p** (print) - Ispisuje sadržaj memorije na zadatoj adresi ili u zadatom opsegu adresa (npr. A1-FA).
- **b** (breakpoint) - Služi za manipulaciju breakpoint-ima koji omogućavaju prekid izvršavanja programa na adresama na kojima su postavljeni. Ukoliko se ne navede argument, štampaju se sve adrese na kojima se nalaze

breakpoint-i. Ukoliko je data adresa kao argument, breakpoint se postavlja na toj adresi, ako na njoj već ne postoji, a uklanja se ako već postoji.

- **c** (continue) - Nastavlja prekinuto izvršavanje programa od adrese koja se nalazi u instruction counter-u.
- **n** (next) - Izvršava instrukciju čiju adresu sadrži instruction counter koji se uz to povećava. Omogućava korak po korak izvršavanje i efikasno debagovanje.
- **a** (accumulator) - Ispisuje sadržaj akumulatora.
- **i** (instruction counter) - Ispisuje sadržaj instruction counter-a.
- **f** (flags) - Ispisuje sadržaj flags registra u poretku VSZC.

1.2.2 Format ulaznog fajla

Ulazni fajl se sastoji od niza linija. Svaka linija se sastoji od niza dvocifrenih heksadekadnih brojeva koji mogu biti razdvojeni radi čitljivosti, ali to nije neophodno. Prvi broj u liniji je uvek adresa memorijske lokacije od koje počinje punjenje sadržajem linije. Ostali brojevi se smeštaju na uzastopne memorijske lokacije počevši od one kojoj odgovara adresa koja je data ne početku linije. Na kraju svake linije može se pojaviti komentar koji počinje znakom @ i važi do kraja linije.

2 ASM1 - jednoprolazni asembler

Asembler ASM1 prevodi program napisan u jednostavnom asemblerskom jeziku na mašinski jezik i koji se može izvršavati na SIM-u. Karakteristično je da se uvode skraćenice za instrukcije, ali je neophodno koristiti apsolutne adrese.

2.1 Instrukcije

Skup instrukcija odgovara skupu instrukcija koje su dostupne u mašinskom jeziku, sa razlikom što se umesto operacionih kodova pišu odgovarajuće skraćenice za instrukcije. Ove skraćenice su navedene u odeljku koji se bavi instrukcijama mašinskog jezika.

2.2 Načini adresiranja

Za različite načine adresiranja se uvode oznake koje stoje uz argument instrukcije. Oznake su sledece.

- Direktno adresiranje - Bez oznake
- Neposredno adresiranje - #
- Indirektno adresiranje - [argument]

- Relativno adresiranje - (argument)

2.3 Način korišćenja

Asembler se poziva iz komandne linije navodjenjem imena izvršnog fajla - asm1. Moguća su dva parametra komandne linije. Prvi je obavezan i predstavlja ime ulazne datoteke sa programom napisanim u asemblerskom jeziku. Drugi je opcion i predstavlja ime izlazne datoteke. Ukoliko nije naveden, izlazna datoteka se naziva a.out.

2.4 Format ulaznog fajla

Ulazni fajl se sastoji od više sekcija. Sekcije mogu biti za podatke ili za program. Prvo se navode sve sekcije za podatke, a onda sve sekcije za program. Sekcija bilo kog tipa može biti više od jedne.
Sekcija za prodatke se navodi direktivnom .data iza koje sledi adresa počevsi od koje se podaci pišu u memoriju, a posle nje se navode jednobajtni brojevi koji predstavljaju podatke. Za podatke je rezervisan memorijski prostor sa adresama od 160 do 255, uključujući.
Sekcija za program se navodi direktivom .prg iza koje sledi adresa počevši od koje se program upisuje u memoriju, a posle nje se navode instrukcije koje čine program. Za program je rezervisan memorijski prostor sa adresama od 0 do 159, uključujući.
Svi brojevi koji se pojavljuju u programu moraju biti jednobajtni. Brojevi se zapisuju dekadno. Moguć je zapis označenih i neoznačenih brojeva. Neoznačeni se pišu bez znaka i imaju raspon od 0 do 255. Označeni su svi brojevi napisani sa znakom. Imaju raspon od -128 do +127. Naravno, ovaj način pisanja je samo pogodnost korisniku prilikom zapisivanja brojeva i ni na koji način se ne garantuje doslednost u tretiranju brojeva kao označenih ili neoznačenih. Na to mora da pazi korisnik.

3 ASM2 - dvoprolazni asembler

Osnovna razlika izmedju ASM1 i ASM2 je što se u drugom asembleru dozvoljava upotreba promenljivih i oznaka, odnosno uvode se simbolička imena za memorijske adrese. Imena instrukcija su ista, a takodje i oznake načina adresiranja. Takodje, ASM2 se poziva na isti način kao ASM1.

3.1 Format ulaznog fajla

Program se sastoji od tri vrste sekcija. To su sekcija za promenljive, sekcije za podatke i sekcija za program. Dozvoljena je samo jedna sekcija za promenljive i samo jedna sekcija za program, dok može postojati više sekcija za podatke.
Sekcija za promenljive se navodi direktivom .var bez argumenata. Dalje se navode promenljive koje se koriste u programu. Svaka promenljiva mora biti

praćena vrednošću kojim se inicijalizuje. Za promenljive je odvojen prostor sa adresama od 200 do 255. Smeštanjem promenljivih u memoriju upravlja asembler prilikom prevodjenja.
Za sekciju podataka važi sve što je rečeno za ASM1, osim ograničenja da adrese moraju biti od 160 do 199.
Sekcija za program počinje direktivom .prg bez argumenata. Program se smešta u memoriju počevši od adrese 0x00.
Kao argumenti instrukcija sada se mogu pojavljivati i promenljive i oznake. Oznake se navode u programu tako što se na odredjenom mestu navede ime oznake koje se završava karakterom ':'. Prilikom korišćenja oznake ovaj karakter se ne navodi.