Mašinske instrukcije

Karakteristike mašinskih instrukcija

Skup različitih mašinskih instrukcija koje mogu da se izvršavaju na nekom procesoru se naziva *skup instrukcija procesora*.

Informacije potrebne za uspešno izvršavanje uključuju:

1. **Operacioni kod instrukcije** koji definiše operaciju koja će biti izvršena
2. **Reference na operande instrukcije**
3. **Referencu na narednu instrukciju** koja treba da se prenese u procesor po završetku izvršavanja tekuće instrukcije.

Format instrukcija

Sve instrukcije, bez obzira o kom se procesoru radi imaju sličan format. Na početku instrukcije se nalazi operacioni kod, a za njim slede operandi instrukcije.

Operacioni kod	Operand1	Operand2

- Mnemoničke oznake za operacione kodove.

80x86	PowerPC	S/390	Značenje
ADD	add	A	Sabira vrednosti (sadržaje) operanada
SUB	subf	S	Oduzima vrednosti (sadržaje) operanada
MUL	mullw	M	Množi vrednosti (sadržaje) operanada
AND	and	N	Logičko AND nad vrednostima operanada
LDS	ld	L	Punjenje podataka iz memorije

Podelu skupa instrukcija moguće je izvesti po operaciji koja je zadata instrukcijom. Skup instrukcija može da sadrži više različitih instrukcija za jednu istu operaciju koje se primenjuju u zavisnosti od tipa operanda. Npr. deo skupa instrukcija procesora S/390 koji se koristi za operaciju sabiranja je

Mne-monic	Operands	Name	Format	Op. Code
A	R1,D2(X2,B2)	Add	RX	5A
AD	R1,D2(X2,B2)	Add Normalized (LH)	RX	6A
ADB	R1,D2(X2,B2)	Add (LB)	RXE	ED1A
ADBR	R1,R2	Add (LB)	RRE	B31A
ADR	R1,R2	Add Normalized (LH)	RR	2A
AE	R1,D2(X2,B2)	Add Normalized (SH)	RX	7A
AEB	R1,D2(X2,B2)	Add (SB)	RXE	ED0A
AEBR	R1,R2	Add (SB)	RRE	B30A
AER	R1,R2	Add Normalized (SH)	RR	3A
AH	R1,D2(X2,B2)	Add Halfword	RX	4A
AHI	R1,I2	Add Halfword Immediate	RI	A7A
AL	R1,D2(X2,B2)	Add Logical	RX	5E
ALR	R1,R2	Add Logical	RR	1E
AP	D1(L1,B1),D2(L2,B2)	Add Decimal	SS	FA
AR	R1,R2	Add	RR	1A
AU	R1,D2(X2,B2)	Add Unnormalized (SH)	RX	7E
AUR	R1,R2	Add Unnormalized (SH)	RR	3E
AW	R1,D2(X2,B2)	Add Unnormalized (LH)	RX	6E
AWR	R1,R2	Add Unnormalized (LH)	RR	2E
AXBR	R1,R2	Add (EB)	RRE	B34A
AXR	R1,R2	Add Normalized (EH)	RR	36

Tabela 1: Instrukcije S/390 koje se odnose na sabiranje

Podelu formata instrukcija moguće je napraviti i prema vrsti i broju njihovih operanada. Instrukcije u istoj grupi imaju istu dužinu, broj operanada i korišćeni način adresiranja. Na primer, instrukcije S/390 procesora iz tabele 1 se dele u grupe prema lokaciji gde se nalaze operandi na:

- RR: Registar–registar
- RX: Registar–indeksirana memorija
- RXE: Registar–indeksirana memorija sa produženim poljem za operacioni kod
- RRE: Registar–registar sa produženim poljem za operacioni kod
- RS: Registar–memorija
- RI: Registar–neposredno (operand se neposredno navodi u instrukciji)
- SS: Memorija–memorija

Sami operandi mogu, po tipu podatka, da predstavljaju brojeve, niske karaktera ili nestruktuirane logičke podatke.

Tipovi instrukcija

Instrukcije se mogu grupisati i po vrsti operacije koja se izvršava njihovim pozivanjem. Instrukcije se dele na

1. Aritmetičke
2. Logičke
3. Instrukcije za konverziju
4. Instrukcije za prenos podataka
5. Ulazno/izlazne instrukcije
6. Kontrolne instrukcije
7. Instrukcije za prenos kontrole

Broj adresa u instrukciji

Jedan od načina klasifikacije arhitektura procesora koji je dosta korišćen u prvim decenijama razvoja računara je podela prema maksimalnom broju adresa koji može da se pojavi u instrukciji.

Računari koji imaju maksimalno jednu adresu u instrukcijama se nazivaju *jednoadresni*, dve *dvoadresni* i tri *troadresni*.

Posmatrajmo izvršavanje aritmetičkog izraza: F = (A + B +C) * (D + E). Neka A, B, C, D, E i F označavaju memorijske lokacije. U tabeli 2 su korišćene sledeće mnemoničke oznake:

Za jednoadresne računare

LOAD Z	prenos podatka sa adrese Z u akumulator
STORE Z	vrednost iz akumulatora se upisuje na adresu Z
ADD Z	vrednost akumulatora i vrednost sa adrese Z se sabiraju i rezultat se upisuje u akumulator
MUL Z	vrednost akumulatora se množi sa vrednošću na adresi Z i rezultat se upisuje u akumulator

Za dvoadresne računare

LOAD R, Z	prenos podatka sa adrese Z u opšti registar R
ADD R,Z	vrednosti iz registra R i sa adrese Z se sabiraju i rezultat upisuje u registar R
MR R1,R2	vrednosti registara R1 i R2 se množe i rezultat se upisuje u registar R1
STORE R, Z	vrednost iz registra R se upisuje na adresu Z

Za troadresne računare

ADD Y,Z,W	vrednosti sa adresa Z i W se sabiraju i rezultat se upisuje na adresu Y
MUL Y,Z,W	vrednosti sa adresa Z i W se množe rezultat se upisuje na adresu Y

Jednoadresni računar	Dvoadresni računar	Troadresni računar
LOAD A	LOAD R1, A	ADD R,A,B
ADD B	ADD R1, B	ADD R,R,C
ADD C	ADD R1, C	ADD F,D,E
STORE F	LOAD R2, D	MUL F,F,R
LOAD D	ADD R2, E	
ADD E	MR R1,R2	
MUL F	STORE R1, F	
STORE F		

Tabela 2: Program koji izračunava vrednost F = (A + B + C)*(D + E)

Načini adresiranja

Za navodjenje adresa operanada koriste se različite tehnike poznate pod nazivom *načini adresiranja*.

Radi ilustracije ovih načina adresiranja uvedimo sledeće oznake:

A	–	Sadržaj adresnog polja u instrukciji
SA	–	Stvarna (efektivna) adresa lokacije koja sadrži operand
R	–	Sadržaj opšteg registra koji se javlja u instrukciji
(l)	–	Sadržaj lokacije l, pri čemu l može da predstavlja lokaciju u memoriji ili opšti registar
B	–	Bazni registar
X	–	Indeks registar
PC	–	Brojač instrukcija

Način adresiranja odredjuje algoritam po kome se izračunava stvarna adresa. Stvarna adresa može da označava registar, realnu ili virtualnu memoriju.

Skoro svi računari koriste bar jedan od načina adresiranja koji je izložen u daljem tekstu, pri čemu je vrlo čest slučaj da se navedeni načini kombinuju. U tom slučaju informacija o načinu adresiranja mora da bude uključena i instrukciju. To se postiže na dva načina:

1. Upotrebom različitih operacionih kodova za istu operaciju sa različitim načinima adresiranja (na primer, `A` i `AR` za sabiranje.

2. Upotrebom tzv. *indikatora modifikacije*. Indikatori modifikacije predstavljaju jedan ili više bitova u samoj instrukciji čije vrednosti označavaju način adresiranja.

Registarsko adresiranje

Kod registarskog adresiranja oznaka registra koji sadrži operand je uključena u instrukciju. Stvarna adresa se u ovom slučaju odredjuje kao

SA = R

Direktno adresiranje

Direktno adresiranje predstavlja način adresiranja kod koga se stvarna adresa direktno uključuje u instrukciju:

SA = A

Adrese koje se javljaju u instrukcijama sa direktnim adresiranjem se nazivaju i *apsolutne adrese*.

Indirektno adresiranje

Indirektno adresiranje predstavlja način adresiranja kod koga se adresa operanda navodi indirektno, preko adrese lokacije na kojoj se nalazi adresa operanda.

Ako adresni deo instrukcije sadrži adresu A lokacije na kojoj se nalazi adresa operanda, tada je stvarna adresa operanda

SA = (A)

Ovaj način adresiranja zahteva dva memorijska ciklusa, jedan za čitanje adrese lokacije na kojoj se nalazi adresa operanda i drugi za čitanje samog operanda.

Moguće je vršiti indirektno adresiranje po dubini:

SA =(...(A)...)

Indirektno registarsko adresiranje

Ovaj način adresiranja je sličan indirektnom sem što se u adresnom polju instrukcije nalazi oznaka registra a ne adresa lokacije u memoriji. Stvarna adresa se u ovom slučaju odredjuje kao

SA = (R)

Adresiranje sa referencom i udaljenjem

Ovaj način adresiranja predstavlja kombinaciju direktnog i registarskog indirektnog adresiranja. Instrukcija sadrži referencu na dva adresna polja od kojih je bar jedno eksplicitno. Vrednost A koja je sadržana u prvom adresnom polju se koristi kao da je u pitanju direktno adresiranje. Drugo adresno polje sadrži oznaku registra čiji sadržaj se sabira sa A radi dobijanja stvarne adrese. Ako drugo adresno polje ne postoji, tada se implicitno referiše neki konkretan registar računara. U oba slučaja vrši se referenca na registar dok vrednost A predstavlja udaljenje (eng. *displacement*) od početne adrese navedene u registru R.

Postoji više varijanti adresiranja sa referencom i udaljenjem. Najznačajnije su:

- Relativno adresiranje
- Adresiranje sa baznim registrom i udaljenjem
- Adresiranje sa indeks registrom
- Adresiranje sa baznim registrom, indeks registrom i udaljenjem

Relativno adresiranje

U slučaju da se kao registar koji se referiše implicitno koristi brojač instrukcija, takav način adresiranja se naziva relativno adresiranje. Za referentnu (početnu) adresu se uzima adresa sadržana u brojaču instrukcija, tj. adresa tekuće instrukcije. U adresni deo instrukcije se upisuje ceo broj koji predstavlja udaljenje (pozitivno ili negativno) od početne adrese. Izračunata stvarna adresa je jednaka

SA = A + (PC)

Adresiranje sa baznim registrom i udaljenjem

Ukoliko se referentna adresa nalazi u posebnom registru procesora (različitom od brojača instrukcija) tada se takav način adresiranja naziva adresiranje sa baznim registrom i udaljenjem. Udaljenje od adrese sadržane u baznom registru je specificirano u adresnom delu. Stvarna adresa kod ovog načina adresiranja je

SA = A + (B)

Adresiranje sa indeks registrom

U ovom načinu adresiranja referentna adresa se nalazi u adresnom delu instrukcije a udaljenje u indeks registru. Stvarna adresa kod ovog načina adresiranja se dobija sabiranjem ovih vrednosti:

SA = A + (X)

Naziv *indeks registar* potiče od njegove upotrebe jer se ovaj način adresiranja koristi za izvodjenje iteracije.

Adresiranje sa baznim registrom, indeks registrom i udaljenjem

Ovaj način adresiranja predstavlja proširenje adresiranja aa baznim registrom i udaljenjem. Referentna adresa se nalazi u baznom registru; adresni deo instrukcije sadrži udaljenje dok indeks registar sadrži udaljenje koje može jednostavno da se poveća ili smanji. Stvarna adresa kod ovog načina adresiranja se dobija sabiranjem ovih vrednosti:

SA = (B) + A + (X)

Dodavanjem indeks registra omogućena je primena adresiranja sa baznim registrom u iterativnim ciklusima.

Neposredno adresiranje

Neposredno adresiranje predstavlja način adresiranja kod koga se operand nalazi u adresnom delu instrukcije:

operand = A

Implicitno adresiranje

Ovaj način adresiranja se javlja kada u procesoru postoje jedinstveni objekti odredjenog tipa tako da ih nije potrebno navoditi u instrukcijama (npr. rad sa stekom)

Drugi primer implicitnog adresiranja su instrukcije bez operanada. Operandi ovih instrukcija se nalaze na unapred poznatim lokacijama.

Mašinski i asemblerski jezici

Mašinski jezik čini skup instrukcija procesora koji poseduje semantičku interpretaciju u hardveru ili mikrokodu računara.

Sve mašinske instrukcije i njihovi operandi predstavljaju nizove binarnih cifara. Na primer, neka treba programirati na mašinskom jeziku naredbu višeg programskog jezika

```
A = B + C * D;
```

Neka program počinje na adresi 51 heksadekadno, a prostor na kome su smeštene promenljive počinje na adresi 81 heksadekadno. Neka su celobrojne promenljive `A`, `B`, `C` i `D` tim redom smeštene u memoriji i neka svaka od njih zauzima po 2 bajta, pri čemu su inicijalne vrednosti A=0, B=10, C=20, D=30 dekadno. Takodje, neka je adresibilna jedinica bajt, a dužina instrukcije dva bajta pri čemu prvi polubajt zauzima operacioni kod, drugi polubajt oznaka registra, dok je u preostalom bajtu upisana adresa. Program se sastoji od 4 instrukcije:

1. Prenesi sadržaj sa lokacije 85 u opšti registar 1.
2. Pomnoži sadržaj lokacije 87 sa sadržajem opšteg registra 1.
3. Saberi sadržaj lokacije 83 sa sadržajem opšteg registra 1.
4. Dobijeni zbir upiši na lokaciju 81.

Za programe koji su pisani sa menmoničkim oznakama instrukcija uz korišćenje relativnog adresiranja se kaže da su pisani u *asemblerskom jeziku*. Program koji vrši prevodjenje programa na asemblerskom jeziku na mašinski jezik računara se naziva *asembler*.

U narednoj tabeli koja prikazuje program koji izvršava naredbu

`A = B + C * D;`

dok su 1, 2, 3, i 4 kodovi za instrukcije LOAD, ADD, MUL i STORE

Adresa	Sadržaj (binarno)			
51	0001	0001	1000	0101
53	0011	0001	1000	0111
55	0010	0001	1000	0011
57	0100	0001	1000	0001
81	0000	0000	0000	0000
83	0000	0000	0000	1010
85	0000	0000	0001	0100
87	0000	0000	0001	1110

[a] Binarni program

Adresa	Sadržaj (heks.)
51	1185
53	3187
55	2183
57	4181
81	0000
83	000A
85	0014
87	001E

[b] Heksadekadni program

Adresa	Simb. kod	Operandi
51	LOAD	1,85
53	MUL	1,87
53	ADD	1,83
55	STORE	1,81
81	INT	0
83	INT	10
85	INT	20
87	INT	30

[c] Simbolički program

Labele	Simb. kod	Operandi
START	LOAD	1,C
	MUL	1,D
	ADD	1,B
	STORE	1,A
A	INT	0
B	INT	10
C	INT	20
D	INT	30

[d] Asemblerski program

Tabela 3: Izračunavanje izraza `A = B + C * D`

Pozivi potprograma

Uprošćen redosled akcija pri pozivu potprograma obuhvata:

1. Odredjivanje adrese potprograma koga treba pozvati.
2. Odredjivanje i pripremu argumenata potprograma koji se poziva.
3. Čuvanje svih relevantnih parametara trenutnog stanja programa koji se izvršava.
4. Specificiranje adrese na koju se vrši povratak iz potprograma.
5. Izvršavanje instrukcije skoka kojom se izvršavanje prenosi na naredbu potprograma.
6. Izvršavanje potprograma
7. Izvršavanje naredbe povratka na zahtevanu adresu u pozivajućem programu.

Argumenti i rezultati potprograma se mogu preneti u opštem slučaju na tri načina:

1. Preko opštih registara.
2. Upisivanjem u odredjenu zonu memorije čija adresa se prenosi preko opštih registara.
3. Putem steka, zajedno sa adresom povrataka.

Reentrant potprogrami, su potprogrami čiji se kod ne menja pri njihovom izvršavanju.

Smeštanje podataka u memoriji

Redosled smeštanja bajtova

Pri zapisivanju višebajtnih skalarnih vrednosti u memoriji moguća su dva pristupa:

- Bajt najveće težine se upisuje u bajt sa najmanjom adresom u memoriji. Za ovaj način zapisa koristićemo naziv *sa levog kraja* (eng. *big-endian*).
- Bajt najmanje težine se upisuje u bajt sa najmanjom adresom u memoriji. Za ovaj način zapisa koristićemo naziv *sa desnog kraja* (eng. *little-endian*).

U oba slučaja se preostale vrednosti upisuju kontinuirano. Kada se posmatra zapis višebajtne skalarne vrednosti oba zapisa su medjusobno inverzni. Na primer, neka heksadekadnu vrednost 12345678 treba zapisati u 4 bajta počev od adrese 101:

	101	102	103	104
I način	12	34	56	78
II način	78	56	34	12

```
Struct{
   int    a;      \\ hex 1112_1314           rec
   short  b;      \\ hex 2122                polurec
   double c;      \\ hex 3132_3334_3536_3738  dupla rec
   char   d[5];   \\ 'A','B','C','D','E'      niz bajtova
   short  e;      \\  hex 5152               polurec
}s;
```

00	14	08	38	10	'A'
	13		37		'B'
	12		36		'C'
	11		35		'D'
04	22	0C	34	14	'E'
	21		33		
			32		52
			31		51

Zapis sa desnog kraja

00	11	08	31	10	'A'
	12		32		'B'
	13		33		'C'
	14		34		'D'
04	21	0C	35	14	'E'
	22		36		
			37		51
			38		52

Zapis sa levog kraja

Slika 1: Preslikavanje C strukture u memoriji

Prednosti i mane svakog od načina zapisa su:

- Sa desnog kraja
  - Aritmetičke operacije koje zahtevaju veću preciznost se jednostavnije obavljaju jer se lakše odredjuje bajt najmanje težine.
  - Konverzija 32-bitne u 16-bitnu celobrojnu adresu je jednostavnija jer ne zahteva dodatne operacije za izdvajanje bajtova najmanje težine.
- Sa levog kraja
  - Ovaj način zapisa omogućuje brže poredjenje karakter niski koje su zapisane tako da po poravnanju odgovaraju celobrojnim vrednostima. Nad ovakvim niskama se može primeniti paralelno poredjenje više bajtova u ALU.
  - Celobrojne vrednosti i karakter niske se zapisuju konzistentno, tj. u istom redosledu.
  - Pri pregledu sadržaja memorije u slučaju otkrivanja grešaka sve vrednosti se štampaju sleva na desno, što je uobičajen redosled na koji su ljudi navikli.

Redosled smeštaja bitova

Slično kao i u prethodnom slučaju ne postoji jedinstveni način smeštanja bitova unutar bajta. U zavisnosti od proizvodjača računara brojanje (i označavanje) bitova unutar bajta može biti:

- sa desnog kraja

7	6	5	4	3	2	1	0

- sa levog kraja

0	1	2	3	4	5	6	7

Takodje, ne postoji ni konzistentno označavanje rednog broja početnog bita: neki proizvodjači počinju od 0, dok neki počinju od 1, kao što je prikazano na narednoj slici.

7	6	5	4	3	2	1	0

8	7	6	5	4	3	2	1