

# Skupovi instrukcija za Motorolu



Vladimir Filipović  
vladaf@matf.bg.ac.yu

## Registri, memorija i prenos podataka kod Motorole 68000

Ovde će biti opisana osnovna arhitektura procesora u Motorolinoj porodici 68000, time što će biti opisan 68000 ISA.

Familija sadrži veći broj vrsta procesora, koji obezbeđuju različite performanse. Svi članovi ove porodice imaju istu osnovnu arhitekturu, ali noviji članovi imaju dodatne karakteristike koje poboljšavaju njihove performanse.

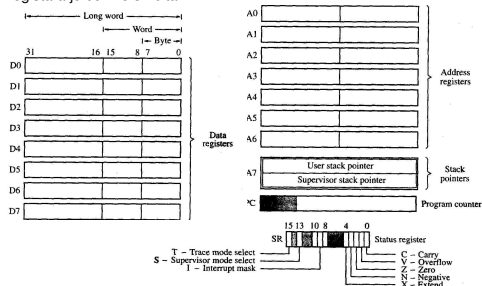
Ovde se opisuje 68000, zato što je on jednostavniji za opis, a odskakava najvažnije osobine cele porodice procesora.

U razmatranju se koncentrišemo na najvažnije aspekte ovog procesora, sa dovoljno detalja da se može napraviti, asemblirati i izvršiti neki prostiji program.

Procesor 68000 karakteriše 16-bitna spoljašnja reč, zato što čip ima 16 data pinova za vezu sa memorijom. Ipak, unutar procesora se manipuliše sa podacima u registrima dužine 32 bita. Napredniji članovi ove porodice procesora 68020, 68030 i 68040 imaju 32 bita pina. Stoga oni i interno i eksterno mogu raditi sa 32-bitnim binarnim niskama.

## Registri, memorija i prenos podataka kod Motorole 68000

Procesor 68000 sadrži 8 registara za podatke i 8 adresnih registara. Svaki od tih registara je dužine 32 bita.



## Registri, memorija i prenos podataka kod Motorole 68000

Registri za podatke služe kao akumulatori opšte namene i kao brojači.

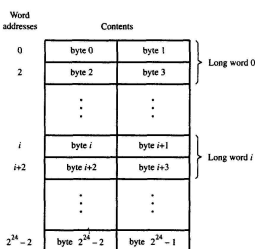
Instrukcije za 68000 rade sa operandima veličine 8, 16 i 32 – bajtovima, rečima i dugim rečima. Kada instrukcija koristi bajt-operand ili reč-operand registra, tada operand zauzima bitove najniže težine registra. U najvećem broju slučajeva, takva instrukcija ne menja bitove veće težine registra, mada neke instrukcije prošire znak tog kraćeg operanda na sve bitove veće težine.

Adresni registri sadrže informacije koje služe za određivanje adrese memorijskog operanda. Te informacije su ili reči ili duge reči. I adresni registri i registri za podatke mogu da se koriste kao indeks-registri. Dreni registar A7 ima specijalnu funkciju – on predstavlja pokazivač na vrh steka procesora.

Adresni registri i izračunavanje adrese uključuju 32 bita. Ipak, kod 68000 samo 24 bita najmanje težine se koriste eksterno za pristup memoriji. Procesori 68020, 68030 i 68040 imaju spoljašnje 32-bitne adresne linije i 32-bitne linije za podatke.

Status registar procesora SR sadrži, pored ostalih i pet bitova za uslovni kod, tri bita za prekid i dva bita za izbor moda rada procesora.

## Registri, memorija i prenos podataka kod Motorole 68000



Memorija za 68000 je organizovana u 16-bitne reči i bajt-adresibilna je.

Dve uzastopne reči se mogu interpretirati kao jedna duga reč.

Reč mora biti poravnata na pamu granicu, tj. njena adresa mora početi od parnog broja.

Ka što se vidi sa dijagrama, pri radu memorije se koristi big endian adretna shema.

Kako 68000 generiše 24-bitne adrese, njegov adresibilni prostor je  $2^{24}$  bajtova, odnosno 16 M.

Taj prostor se može posmatrati kao 512 ( $2^9$ ) strana veličine 32 K ( $2^{15}$ ). Tako, adrese 0 do 7FFF oformljuju stranu 0, adrese 8000 do FFFF stranu 1, itd.

## Registri, memorija i prenos podataka kod Motorole 68000

Procesor 68000 ima veći broj adresnih modova, uključujući one koji su popisani kada se govorilo o generičkom procesoru. Veliki broj instrukcija procesora staje u jednu 16-bitnu reč, ali postoje instrukcije koje zahtevaju dodatne reči za informacije o adresi. Prva reč instrukcije je OP kod, koji specifikira koja će se operacija izvršiti i koji daje neke adresne informacije. Ostale informacije o adresi, ukoliko postoje, se nalaze u rečima koje slede.

Dostupni načini adresiranja su:

- **Neposredni mod** – operand se sadrži u okviru instrukcije. Mogu se specificirati četiri veličine operanda – bajtovi, reči, duge reči i vrlo mali brojevi koji se uključuju direktno u OP kod nekih instrukcija.
- **Apsolutni mod** – absolutna adresa operanda je data u instrukciji, neposredno iza OP koda. Postoje dve verzije ovog moda – kratki i dugački. U dugom modu, 24-bitna adresa se eksplicitno specificira. U kratkom modu, 16-bitna vrednost koja je prosledjena instrukciji se koristi kao niži 16 bitova adrese, a preostali bitovi se dobijaju proširivanjem znaka te 16-bitne vrednosti. Lako je pokazati da se kratkim absolutnim modom mogu adresirati samo dve strane: strana 0 i strana FF8.
- **Registarski mod** – operand je registar procesora koji je specificiran u instrukciji.

Matematički fakultet Mikoročunari vladan@matf.bg.ac.yu 1943

## Registri, memorija i prenos podataka kod Motorole 68000

- **Registarski indirektni mod** – efektivna adresa operanda je adresni registar procesora koji je specificiran u instrukciji.
- **Autoinkrement mod** - efektivna adresa operanda je adresni registar An koji je specificiran u instrukciji. Po pristupu operandu, sadržaj registra An se uvećava za 1, 2 ili 4 –zavisno od toga da li je operand bio bajt, reč ili duga reč respektivno.
- **Autodekrement mod** - sadržaj registra An se umanjuje za 1, 2 ili 4 – zavisno od toga da li je operand kome se pristupa bajt, reč ili duga reč respektivno. Po izvršetku umanjnja, efektivna adresa operanda je ta umanjena vrednost adresnog registra An.
- **Osnovni indeksni mod** – u instrukciji su specificirani 16-bitni označeni ofset i adresni registar An. Efektivna adresa operanda se dobija sabiranjem ofseta i sadržaja registra An.
- **Puni indeksni mod** - u instrukciji su specificirani 8-bitni označeni ofset, adresni registar An i indeksni registar Rk (on može biti i adresni registar i registar sa podacima). Efektivna adresa operanda se računa kao suma ofseta i sadržaja registra An i Rk. Za adresu se koriste bilo 32 bita, bilo znakom proširenih donjih 16 bitova. Asembleru se ukazuje o kojoj se varijanti radi tako što se na naziv registra doda sufiks L za dužu reč, odnosno W za reč. Tako, na primer, imamo D1.L i D1.W, pri čemu je ovo drugo podrazumevano.

Matematički fakultet Mikoročunari vladan@matf.bg.ac.yu 643

## Registri, memorija i prenos podataka kod Motorole 68000

| Name              | Assembler syntax                 | Addressing function                                         |
|-------------------|----------------------------------|-------------------------------------------------------------|
| Immediate         | #Value                           | Operand = Value                                             |
| Absolute Short    | Value                            | EA = Sign Extended WValue                                   |
| Absolute Long     | Value                            | EA = Value                                                  |
| Register          | Rn                               | EA = R <sub>n</sub><br>that is, Operand = (R <sub>n</sub> ) |
| Register Indirect | (An)                             | EA = (A <sub>n</sub> )                                      |
| Autoincrement     | (An)+                            | EA = (A <sub>n</sub> );<br>Increment A <sub>n</sub>         |
| Autodecrement     | -(An)                            | Decrement A <sub>n</sub> ;<br>EA = (A <sub>n</sub> )        |
| Indexed basic     | WValue(An)                       | EA = WValue + (A <sub>n</sub> )                             |
| Indexed full      | BValue(An,Rk,S)                  | EA = BValue + (A <sub>n</sub> ) + (R <sub>k</sub> )         |
| Relative basic    | WValue(PC)<br>or Label           | EA = WValue + (PC)                                          |
| Relative full     | BValue(PC,Rk,S)<br>or Label (Rk) | EA = BValue + (PC) + (R <sub>k</sub> )                      |

EA = effective address  
Value = a number given either explicitly or represented by a label  
BValue = an 8-bit Value  
WValue = a 16-bit Value  
A<sub>n</sub> = an address register  
R<sub>k</sub> = an address or a data register  
S = a size indicator: W for signextended 16-bit word and L for 32-bit long word

Matematički fakultet Mikoročunari vladan@matf.bg.ac.yu 1943

## Registri, memorija i prenos podataka kod Motorole 68000

U ova dva poslednja slučaja, umesto adresnog registra se koristi PC. Razmotrimo instrukciju sa punim relativnim modom adresiranja kod prvog (izvornog) operanda i registarskim kod drugog (određiškog) operanda:

ADD 100(PC,A1), D0

Mašinski kod ove instrukcije se sastoji od dve reči. OP kod instrukcije ukazuje da se vrši sabiranje, da je određiškni operand registar D0 i da se za izvor koristi puni relativni mod. Druga reč instrukcije, koja se još zove i reš proširenja, specificira da se A1 koristi kao ofset registar i sadrži broj 100 kao ofset koji je enkodiran u osam bitova.

Kada se dohvati OP kod reč ove instrukcije, i dok se ona dekodira, PC već pokazuje na reč proširenja instrukcije, što znači da PC sadrži 1002.

EA = (PC) + (A1) + 100  
= 1002 + 6 + 100  
= 1008

Matematički fakultet Mikoročunari vladan@matf.bg.ac.yu 1043

## Registri, memorija i prenos podataka kod Motorole 68000

Instrukciju sabiranja u prethodnom primeru smo pisali u eksplicitnom formatu. Asembler dopušta da se relativni mod specificira na jednostavniji način. Tada se asembler informiše (korišćenjem odgovarajuće direktive) da će u datoj sekciji programa biti korišćeno relativno adresiranje. Nadalje, vrednosti 1102 se dodeli simboličko ime, npr ARRAY, pa se onda instrukcija može zapisati sa

ADD ARRAY(A1), D0

Asembler ovu specifikaciju izvornog operanda interpretira kao puni relativni mod i računa ofset na način koji je objašnjen prethodnim dijagramom. Ofset kod punog relativnog adresiranja je označen broj koji se smešta u jednom bajtu u formatu potpunog komplementa i on se kreće između -128 i 127.

Matematički fakultet Mikoročunari vladan@matf.bg.ac.yu 1143

## Instrukcije kod Motorole 68000

Procesor 68000 ISA obezbeđuje veoma opsežan skup instrukcija, od kojih najveći broj može raditi sa sve tri moguće vrednosti operanda. Svi adresni modovi se mogu koristiti na uniformni način sa najvećim brojem instrukcija. Za skupove instrukcija koji imaju ovakvo svojstvo kažemo da su ortogonalni.

Procesor sadrži instrukcije sa jednim i sa dva operanda. Instrukcija sa dva operanda ima oblik

OP src, dst

gde se operacija vrši nad izvorom i određiškem, a rezultat se smešta u određiškni operand.

Uočimo instrukciju

ADD #9,D3

Uočena instrukcija izvršava sledeću akciju

D3 ← D3 + 9

Kod ADD instrukcije, bilo izvor bilo određište moraju biti registri sa podacima, a drugi operator može biti u registru ili u memoriji.

Matematički fakultet Mikoročunari vladan@matf.bg.ac.yu 1243

## Instrukcije kod Motorole 68000

Sledeći dijagram jasnije pokazuje strukturu instrukcije ADD #9,D3 i efekat njenog izvršavanja. Pre dohvaćanja instrukcije, PC ukazuje na OP-kod na adresi i. Kako se koja reč dohvata iz memorije, sadržaj PC se uvećava za 2, pa je po završetku instrukcije u PC smeštena vrednost i+4.

(a) Format of the OP-code word for an ADD instruction

(b) Encoding of the OP-code word

(c) Consequences of the execution

(d) Encoding of the OP-code word

(e) Consequences of the execution

Matematički fakultet
Mikroračunari
vlasar@matf.bg.ac.rs
1343

## Instrukcije kod Motorola 68000

Razmotrimo sada dijagram rutine za realizaciju izračunavanja  $C = |A| + |B|$

Neka su ove instrukcije smeštene u memoriju računara 68000, kao na slici. Podrazumeva se da su operandi 16-bitni i da su njihove adrese registrovane u apsolutnom modu.

Kako se adrese ne mogu predstaviti unutar 16 bitova, to se mora iskoristiti dugačak mod instrukcije.

Viših 16 bitova 32-bitne adrese se smeštaju nižu reč adrese, a nižih 16 bitova u višu reč adrese, shodno big endian rasporedu.

```

A = 201150
B = 201152
C = 202200

201200  OP-code word
201202  20
201204  1150
201206  OP-code word
201208  20
20120A  1152
20120C  OP-code word
20120E  20
201210  2200

```

MOVE A,D0  
ADD B,D0  
MOVE D0,C

ADD B,D0  
MOVE D0,C

After execution, [202200] = 424<sub>16</sub>

Matematički fakultet
Mikroračunari
vlasar@matf.bg.ac.rs
1443

## Asemblerski jezik Motorola 68000

Razmatranje o "generičkom" asemblerskom jeziku iz prošlog poglavlja može da se, skoro bez ikakvih izmena, primeni na Motorolu 68000.

S obzirom da instrukcije za 68000 mogu da rade sa tri vrste operanda, asemblerska instrukcija mora ukazati koja će se vrsta koristiti. Ovo se postiže dodavanjem indikatora veličine (L za dugu reč, W za reč i B za bajt) na mnemonik instrukcije.

Tako, ako instrukcija sabiranja sabira duge reči, tada se njen mnemonik zapisuje kao

ADD.L

Kada nema indikatora veličine, tada će se podrazumevati da se radi o reč operandima. Stoga su ekvivalentni ADD.W #20, D1 i ADD #20, D1

Sve asemblerske direktive "generičkog" asemblerskog jezika se, uz minimalne izmene, javljaju kod procesora 68000:

- Početna adresa bloka instrukcija ili podataka data je sa direktivom ORG.
- Direktiva EQU izjednačava imena sa numeričkim vrednostima
- Konstante – podaci se unose u objektni program korišćenjem direktive DC, pri čemu se dodaje indikator za veličinu koji određuje veličinu podatka. Jedna DC direktiva može da se iskoristi za definisanje većeg broja konstanti.

Matematički fakultet
Mikroračunari
vlasar@matf.bg.ac.rs
1543

## Asemblerski jezik Motorola 68000

Na primer, direktive

```

ORG 100
PLACE DC.B 23,$4F,%10110101

```

dovode do smeštanja heksadekadnih vrednosti 17, 4F i B5 u memorijske lokacije 100, 101 i 102 respektivno. Labeli PLACE se dodeljuje vrednost 100.

- Memorijski blok se može rezervisati za podatke korišćenjem direktive DS (define storage).

Na primer, direktiva

```

ARRAY DS.L 200

```

Rezervise 200 dugih reči i imenu ARRAY pridružuje adresu prve rezervisane duge reči.

Matematički fakultet
Mikroračunari
vlasar@matf.bg.ac.rs
1643

## Asemblerski jezik Motorola 68000

Sljedeći kod predstavlja primer asemblerskog programa za 68000 kojim se realizuje sabiranje dva broja:

|                                               | Memory address label | Operation | Addressing or data information |
|-----------------------------------------------|----------------------|-----------|--------------------------------|
| Assembler directives                          | C                    | EQU       | \$202200                       |
|                                               |                      | ORG       | \$201150                       |
|                                               | A                    | DC.W      | 639                            |
|                                               | B                    | DC.W      | -215                           |
|                                               |                      | ORG       | \$201200                       |
| Statements that generate machine instructions |                      | MOVE      | A,D0                           |
|                                               |                      | ADD       | B,D0                           |
| Assembler directive                           |                      | MOVE      | D0,C                           |
|                                               |                      | END       |                                |

Matematički fakultet
Mikroračunari
vlasar@matf.bg.ac.rs
1743

## Kontrola toka izvršavanja kod Motorola 68000

Kao što je već istaknuto, instrukcije uslovnog skoka su potrebne radi realizacije programskih struktura kao što su grananja i petlje. Instrukcije uslovnog skoka proveravaju da li važi uslov i zaviso od dobijenog rezultata, dovode do nastavka izvršavanja programa jednim od dva moguća puta. Uslov koji se proverava se odnosi na rezultate prethodno izvršene operacije.

68000 ima pet flegova uslovnog koda sadržanih u statusnoj reči.

Pored prethodno opisanih N,Z,V i C flegova, tu je i X (extend) fleg. X fleg biva postavljen u istim slučajevima kada i C fleg, ali na njega ne utiče tako mnogo instrukcija kao što je to slučaj sa "klasičnim" flegom prenosa. Korist od ovakvog dupliranja funkcionalnosti biće očigledna kada se budu proučavale operacije velike preciznosti. Dakle, i C i X bivaju postavljeni ako dođe do prenosa sa mesta najveće težine tokom sabiranja ili ako bude pozajmice sa tog mesta tokom oduzimanja. Kako operandi mogu biti dužine bajt, reč i duga reč, to ovi flagovi mogu ukazivati na prenos sa bitovne pozicije 7, 15 i 31 respektivno, zaviso od toka koja se varijanta instrukcije izvršavala.

Instrukcija MOVE podešava N i Z fleg da odslikavaju rezultat, ali briše C i V fleg. Međutim, instrukcija MOVE ne utiče na X fleg.

Matematički fakultet
Mikroračunari
vlasar@matf.bg.ac.rs
1843

## Kontrola toka izvršavanja kod Motorola 68000

Instrukcija uslovnog skoka dovodi do nastavka izvršavanja programa od instrukcije koja je ciljna adresa skoka, pod pretpostavkom da su ispunjeni uslovi za skok. Ciljna adresa, tj. adresa doskoka, se određuje na osnovu ofseta skoka koji se nalazi u polju za operand instrukcije. Kod procesora 68000 ofset skoka u instrukciji se može odrediti na dva načina:

- Kratki ofset (dužine 8 bitova) uključen u reč OP-koda. On se može koristiti kada je određite skoka u opsegu -128 do +127 bajtova od vrednosti PC u trenutku kada se računa adresa na koju se "skače".
- Dugi ofset (dužine 16 bitova) koji se nalazi u reči koja sledi iza reči OP-koda. Tu je opseg adresa koje mogu biti određite doskoka mnogo veći (±32K). U tom slučaju, ofset je rastojanje od reči-proširenja do određite skoka.

Procesor 68000 poseduje 16 instrukcija uslovnog skoka, svaka sa 8-bitnim i 16-bitnim ofsetom. On takođe sadrži i instrukciju bezuslovnog skoka BRA, i tu se skok uvek realizuje.

Matematički fakultet Mikoročunari via@matf.bg.ac.rs 1943

## Kontrola toka izvršavanja kod Motorole 68000

Sledeći dijagram ilustruje korišćenje instrukcije uslovnog skoka sa kratkim ofsetom. On pokazuje kako se petlja za sabiranje može realizovati na procesoru 68000.

Appearance of loop in memory

|           |              |                      |
|-----------|--------------|----------------------|
| LOOP 1000 | OP-code word | ← LOOP ADD.W (A2)+D0 |
| 1002      | OP-code word | ← SUBQ.W #1,D1       |
| 1004      | OP code -6   | ← BGT LOOP           |
| 1006      |              |                      |

Assembly language version of loop

```

LOOP ADD.W (A2)+D0
SUBQ.W #1,D1
BGT LOOP

```

[PC] = 1006 when branch address is computed  
Branch address = 1006 - 6 = 1000

Matematički fakultet Mikoročunari via@matf.bg.ac.rs 2043

## Kontrola toka izvršavanja kod Motorole 68000

Uočavamo u prethodnim primeru da je instrukcija dekrementiranja, realizovana korišćenjem Motorola 68000 instrukcije SUBQ (subtract quick). Konkretno, ovde se radilo o instrukciji

SUBQ #1, D1

koja neposredni operand 1 oduzima od vrednosti registra D1. Neposredni operand ove instrukcije je specifikiran kao trobitna vrednost unutar OP-koda, pa je za reprezentaciju ove instrukcije dovoljna samo jedna reč.

Kod koji sledi realizuje sabiranje niza brojeva:

```

MOVE.L N,D1
MOVEAL #NUM1,A2
LOOP CLR.L D0
      ADD.W (A2)+,D0
      SUBQ.L #1,D1
      BGT LOOP
      MOVE.L D0,SUM

```

N contains n, the number of entries to be added, and D1 is used as a counter that determines how many times to execute the loop. A2 is used as a pointer to the list. It is initialized to NUM1, the address of the first entry. D0 is used to accumulate the sum. Successive numbers are added in it. Decrease the counter. If [D1] ≠ 0, execute the loop again. Store the sum in SUM.

Matematički fakultet Mikoročunari via@matf.bg.ac.rs 2143

## Kontrola toka izvršavanja kod Motorole 68000

Procesor 68000, pored uobičajenih instrukcija skokova, sadrži i skup kompleksnijih instrukcija skokova, koje uključuju i brojače. Takve instrukcije omogućuje efikasnije realizovanje petlji u programu.

Te instrukcije se pišu u formatu

DBcc Dn, LABEL

gde cc označava uslov skoka. Na primer, ako se postavi GT umesto cc, dobije se instrukcija DBGT – umanj i skoči dok ne bude veći od.

Način izvršenja DBcc instrukcija se razlikuje od načina izvršenja Bcc instrukcija:

- Ako je uslov određen sa cc zadovoljen, tada se nastavlja sa izvršavanjem od instrukcije koja neposredno sledi iza DBcc.
- Ako uslov nije zadovoljen, tada se dekrementira 16 nižih bitova registra Dn. Ako je rezultat -1, nastavlja se sa izvršavanjem od instrukcije koja neposredno sledi iza DBcc. Ako rezultat nije -1, vrši se skok na adresu specifikiranu labelom LABEL.

Matematički fakultet Mikoročunari via@matf.bg.ac.rs 2243

## Kontrola toka izvršavanja kod Motorole 68000

Tako se na primer, instrukcije

DBcc D3, PETLJA

Sledeća instrukcija

može realizovati na sledeći način:

Bcc SLEDECA  
SUBQ #1, D3  
BGE PETLJA

SLEDECA      Sledeća instrukcija

Uočava se da DBcc naredba predstavlja pogodan način za realizaciju brojačke petlje iz koje se može ranije izaći ako je ispunjen neki uslov.

Jedna od DBcc instrukcija DBF (decrement and branch if false) koristi test uslov koji je uvek netačan i kod nje odluka o skoku zavisi isključivo od vrednosti brojačkog registra. Ova instrukcija ima još jedan sinonim DBRA (decrement and branch always).

Matematički fakultet Mikoročunari via@matf.bg.ac.rs 2343

## Kontrola toka izvršavanja kod Motorole 68000

Da bi demonstrirali korisnost instrukcije DBcc, ponovo ćemo napisati program za sabiranje niza brojeva:

```

MOVE.L N,D1
SUBQ.L #1,D1
MOVEAL #NUM1,A2
CLR.L D0
LOOP ADD.W (A2)+,D0
      DBRA D1,LOOP
      MOVE.L D0,SUM

```

Put n - 1 into the counter register D1.

Loop back until [D1] = -1.

Iako je broj instrukcija isti ako i u prethodnoj varijanti, ovaj program se brže izvršava, zato što se manji broj instrukcija nalazi u telu petlje.

Matematički fakultet Mikoročunari via@matf.bg.ac.rs 2443

## I/O operacije kod Motorole 68000

Procesor 68000 zahteva da svi statusni registri i baferi za podatke interfejsa I/O uređaja budu adresibilni kao da se radi o memorijskim lokacijama. Dakle, zahteva se memorijsko mapiranje I/O uređaja.

Pretpostavimo da bit b<sub>3</sub> statusnog registra tastature (na lokaciji INSTATUS) sadrži kontrolni fleg ulaza SIN. Tada se ulaz sa tastature može realizovati sledećim kodom:

```

READWAIT BTST.W #3,INSTATUS
BEQ READWAIT
MOVE.B DATAIN,D1

```

Instrukcija BTST testira stanje jednog bita određeniog operanda i postavlja uslovni fleg Z na komplement vrednosti testiranog bita. Prvi operand instrukcije specifikira poziciju bita koji se testira

Matematički fakultet
Mikroračunari
vlasat@mat.tg.ac.yu
2043

## I/O operacije kod Motorola 68000

Pretpostavimo da bit  $b_3$  statusnog registra monitora (na lokaciji OUTSTATUS) sadrži kontrolni fleg ulaza SOUT. Tada se slanje znaka koji se nalazi u registru D1 može realizovati sledećim sekvencom instrukcija:

```

WRITEWAIT BTST.W #3,OUTSTATUS
BEQ WRITEWAIT
MOVE.B D1,DATAOUT

```

Matematički fakultet
Mikroračunari
vlasat@mat.tg.ac.yu
2043

## I/O operacije kod Motorola 68000

Sledeći program procesora Motorola 68000 čita jednu liniju znakova sa ulaza, smešta ih u memoriju i prikazuje ih na izlazu:

| MOVEA.L | #LOC,A1              | Initialize pointer register A1 to contain the address of the first location in memory where the characters are to be stored. |
|---------|----------------------|------------------------------------------------------------------------------------------------------------------------------|
| READ    | BTST.W #3,INSTSTATUS | Wait for a character to be entered in the keyboard buffer DATAIN.                                                            |
| BEQ     | READ                 | Transfer the character from DATAIN into the memory (this clears SIN to 0).                                                   |
| MOVE.B  | DATAIN,(A1)          | Wait for the display to become ready.                                                                                        |
| ECHO    | BTST.W #3,OUTSTATUS  | Move the character just read to the output buffer register (this clears SOUT to 0).                                          |
| BEQ     | ECHO                 | Check if the character just read is CR (carriage return). If it is not CR, then branch back and read another character.      |
| MOVE.B  | (A1),DATAOUT         | Also, increment the pointer to store the next character.                                                                     |
| CMPI.B  | #CR,(A1)+            |                                                                                                                              |
| BNE     | READ                 |                                                                                                                              |

Matematički fakultet
Mikroračunari
vlasat@mat.tg.ac.yu
2043

## Stekovi i potprogrami kod Motorola 68000

U ranijim sekcijama je već opisano da se stek implementira tako što se neki adresni registar koristi kao pokazivač steka. Postojanje autodekrement i autodekrement adresnog moda kod procesora 68000 pojednostavljuje i olakšava realizaciju steka.

Kod 68000 jedan konkretan registar, registar A7, je dizajniran kao pokazivač steka procesora, pa se stek na koji registar A7 pokazuje i naziva stek procesora.

Na dijagramu koji prikazuje registre procesora 68000 uočavaju se dva registra A7, naime, 68000 obezbeđuje dva moda rada – korisnički mod i supervizorski mod. Svaki od modova ima svoju verziju pokazivača na stek procesora. Kada procesor radi u supervizorskom modu, on može da izvršava sve instrukcije. Kada procesor radi u korisničkom modu, neke instrukcije, tzv. privilegovane instrukcije, ne mogu da se izvršavaju. Obično aplikativni programi rade u korisničkom modu, a programi operativnog sistema u supervizorskom modu. Bit S u statusnom registru procesora određuje koji od ova dva moda je aktivan, a samim tim i koji od ova dva A7 registra se koristi za tekuće izvršavanje.

Matematički fakultet
Mikroračunari
vlasat@mat.tg.ac.yu
2043

## Stekovi i potprogrami kod Motorola 68000

Kod 68000 za poziv potprograma se koristi instrukcija BSR (branch to subroutine). Ona je implementirana kao i ostale instrukcije skoka, ali istovremeno dovodi i do toga da sadržaj registra PC bude gurnut na stek. Ciljna adresa kod BSR instrukcije predstavlja adresu prve instrukcije potprograma.

Kada potprogram završi sa radom, treba sa se izvrši instrukcija RTS (return from subroutine) kako bi se vratili u pozivajući program. Instrukcija RTS skine adresu sa vrha steka i smesti je u registar PC.

Matematički fakultet
Mikroračunari
vlasat@mat.tg.ac.yu
2043

## Stekovi i potprogrami kod Motorola 68000

Sada ćemo prethodni program za sabiranje niza brojeva organizovati tako da poziva potprogram u kome se realizuje to sabiranje. Adresa niza i broj članova su prosledeni potprogramu pomoću registra A2 i D1. Po izvršenju sumiranja, potprogram je vratio rezultat preko registra D0.

**Calling program**

```

MOVEA.L #NUM1,A2 Put the address NUM1 in A2
MOVE.L N,D1 Put the number of entries n in D1
BSR LISTADD Call subroutine LISTADD
MOVE.L D0,SUM Store the sum in SUM.
next instruction

```

**Subroutine**

```

LISTADD SUBQ.L #1,D1 Adjust count to n - 1.
CLR.L D0
LOOP ADD.W (A2)+,D0 Accumulate sum in D0.
DBRA D1,LOOP
RTS

```

Matematički fakultet
Mikroračunari
vlasat@mat.tg.ac.yu
2043

## Stekovi i potprogrami kod Motorola 68000

U sledećem primeru se argumenti potprograma prenose preko steka, na koji pokazuje registar A7:

(Assume top of stack is at level 1 below.)

**Calling program**

```

MOVE.L #NUM1-(A7) Push parameters onto stack.
MOVE.L N-(A7)
BSR LISTADD
MOVE.L 4(A7),SUM Save result.
ADD.L #8,A7 Restore top of stack.

```

**Subroutine**

```

LISTADD MOVE.L D0-D1/A2, -(A7) Save registers D0, D1, and A2.
MOVE.L 16(A7),D1 Initialize counter to n.
SUBQ.L #1,D1 Adjust count to use DBRA.
MOVEA.L 20(A7),A2 Initialize pointer to the list.
CLR.L D0 Initialize sum to 0.
LOOP ADD.W (A2)+,D0 Add entry from list.
DBRA D1,LOOP
MOVE.L D0,20(A7) Put result on the stack.
MOVE.L (A7)+,D0-D1/A2 Restore registers.
RTS

```

(a) Stack contents at different times

(b) Calling program and subroutine

## Stekovi i potprogrami kod Motorole 68000

Instrukcija MOVEM (move multiple registers) se koristi za čuvanje i restauraciju registara A1, D2 i D0. Na dijagramu koji prikazuje sadržaj steka vidimo redosled kojim su ti registri smešteni na stek.

Prva MOVEM instrukcija, kod koje se koristi autodekrement adresni mod, gura specificirane registre na stek.

Druge MOVEM instrukcija, kod koje je specificiran autoinkrement adresni mod, skida vrednosti sa steka i smešta ih u registre (u obrnutom redosledu od redosleda navođenja).

## Stekovi i potprogrami kod Motorole 68000

Sledeći primer predstavlja ilustraciju realizovanja ugnježenih potprograma (glavni program).

| Memory location        | Instructions     | Comments                   |
|------------------------|------------------|----------------------------|
| <b>Calling program</b> |                  |                            |
| 2000                   | MOVEM.L          | PARAM2, -(A7)              |
| 2006                   | MOVEM.L          | PARAM1, -(A7)              |
| 2012                   | BSR              | SUB1                       |
| 2014                   | MOVEM.L          | (A7), RESULT               |
| 2020                   | ADD.L            | #8, A7                     |
| 2024                   | next instruction |                            |
|                        |                  | Place parameters on stack. |
|                        |                  | Load parameter.            |
|                        |                  | Store result.              |
|                        |                  | Restore stack level.       |

## Stekovi i potprogrami kod Motorole 68000

Sledeći primer predstavlja ilustraciju realizovanja ugnježenih potprograma (prvi potprogram).

### First subroutine

|      |      |         |                 |                               |
|------|------|---------|-----------------|-------------------------------|
| 2100 | SUB1 | LINK    | A6, #0          | Set frame pointer.            |
| 2104 |      | MOVEM.L | D0-D2/A0, -(A7) | Save registers.               |
|      |      | MOVEA.L | 8(A6), A0       | Load parameters.              |
|      |      | MOVE.L  | 12(A6), D0      |                               |
|      |      |         |                 |                               |
|      |      | MOVE.L  | PARAM3, -(A7)   | Place a parameter on stack.   |
| 2160 |      | BSR     | SUB2            |                               |
| 2164 |      | MOVE.L  | (A7)+, D1       | Pop result from SUB2 into D1. |
|      |      |         |                 |                               |
|      |      | MOVE.L  | D2, 8(A6)       | Place result on stack.        |
|      |      | MOVEM.L | (A7)+, D0-D2/A0 | Restore registers.            |
|      |      | UNLK    | A6              | Restore frame pointer.        |
|      |      | RTS     |                 | Return.                       |

## Stekovi i potprogrami kod Motorole 68000

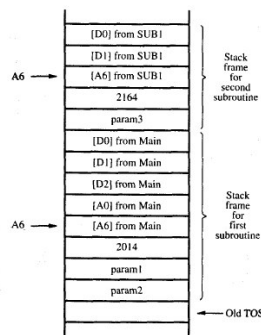
Sledeći primer predstavlja ilustraciju realizovanja ugnježenih potprograma (drugi potprogram).

### Second subroutine

|      |      |         |              |                        |
|------|------|---------|--------------|------------------------|
| 3000 | SUB2 | LINK    | A6, #0       | Set frame pointer.     |
|      |      | MOVEM.L | D0-D1, -(A7) | Save registers.        |
|      |      | MOVE.L  | 8(A6), D0    | Load parameter.        |
|      |      |         |              |                        |
|      |      | MOVE.L  | D1, 8(A6)    | Place result on stack. |
|      |      | MOVEM.L | (A7)+, D0-D1 | Restore registers.     |
|      |      | UNLK    | A6           | Restore frame pointer. |
|      |      | RTS     |              | Return.                |

## Stekovi i potprogrami kod Motorole 68000

Sadržaj steka prilikom realizovanja ugnježenih potprograma.



## Stekovi i potprogrami kod Motorole 68000

Uočavamo da se, pre poziva potprograma SUB1, na stek guraju parametri za taj potprogram.

Potprogram počinje tako što kreira svoj sopstveni stek-okvir. Specijalna instrukcija LINK Ai, #disp

- podešava Ai kao pokazivač na stek-okvir, tako što izvrši sledeće operacije:
1. Gurne sadržaj registra Ai na stek procesora.
  2. Prepise sadržaj pokazivača na stek procesora, tj. registra A7, u registar Ai.
  3. Doda specificiranu vrednost pomeraja na registar A7

Ako je vrednost pomeraja negativna, to će dovesti do podizanja vrha steka (prema nižim memorijskim adresama) i time će biti kreiran prazan prostor na steku, koji potprogram može koristiti za smeštaj lokalnih promenljivih. Vrednostima lokalnih promenljivih koje se tu smešte može da se pristupi preko pokazivača na stek okvir, tj. registra Ai, korišćenjem indeksnog adresiranja.

Na kraju potprograma treba da se izvrši instrukcija UNLK (unlink) koja će poništiti efekat instrukcije LINK: prepisati će Ai u A7 (čime će se vrh steka spustiti na poziciju pre dodavanja pomeraja) i restaurirati vrednost registra Ai sa steka (skinuti adresu sa steka, pa je smestiti u registar Ai).



## Stekovi i potprogrami kod Motorole 68000

U prethodnom primeru nije bilo potrebe za lokalnim promenljivim potprograma, pa su potprogrami počinjali instrukcijom LINK A6,#0.

Potom su, korišćenjem MOVEM instrukcije, na stek čuvane vrednosti registara koji će biti menjani tokom rada potprograma.

Uočavamo da će instrukcija BSR na lokaciji 2012 da bude smeštena u reč (tj. biće korišćena kratka forma ofseta), zato što potprogram SUB1 nije veliki. Sa druge strane, instrukcija BSR na adresi 3160 zauzima dve reči, jer je ofset do ciljane adrese suviše veliki da bi mogao biti predstavljen pomoću 8 bitova.



## Logičke instrukcije kod Motorole 68000

Procesor Motorola 68000 poseduje instrukcije kojima se realizuju logičke operacije AND, OR i XOR nad bitovima, kao i raznovrsne instrukcije pomeranja i rotacije.

Razmotrimo sledeći primer: Neka registar D1 sadrži neki 32-bitni binarni obrazac, i neka nam je zadatak da utvrdimo da li se u tom bitovnom obrascu na pozicijama  $b_{18}$  do  $b_{14}$  nalazi 11001. To može biti urađeno na sledeći način:

```
AND.L  #57C000,D1
CMPL.L #564000,D1
BEQ    YES
```

Program za pakovanje cifara.

|         |           |                                  |
|---------|-----------|----------------------------------|
| MOVEA.L | #LOC,A0   | A0 points to data.               |
| MOVE.B  | (A0)+,D0  | Load first byte into D0.         |
| LSL.B   | #4,D0     | Shift left by 4 bit positions.   |
| MOVE.B  | (A0),D1   | Load second byte into D1.        |
| ANDL.B  | #5F,D1    | Clear high-order 4 bits to zero. |
| OR.B    | D0,D1     | Concatenate the digits.          |
| MOVE.B  | D1,PACKED | Store the result.                |



## Primeri programa kod Motorole 68000

$$\text{Dot Product} = \sum_{i=1}^{n-1} A(i) \times B(i)$$

|      |         |            |                                 |
|------|---------|------------|---------------------------------|
|      | MOVEA.L | #AVEC,A1   | Address of first vector.        |
|      | MOVEA.L | #BVEC,A2   | Address of second vector.       |
|      | MOVE    | N,D0       | Number of elements.             |
|      | SUBQ    | #1,D0      | Adjust count to use DBRA.       |
|      | CLR     | D1         | Use D1 as accumulator.          |
| LOOP | MOVE    | (A1)+,D2   | Get element from vector A.      |
|      | MULS    | (A2)+,D2   | Multiply element from vector B. |
|      | ADD     | D2,D1      | Accumulate product.             |
|      | DBRA    | D0,LOOP    |                                 |
|      | MOVE    | D1,DOTPROD |                                 |



## Primeri programa kod Motorole 68000

```
for (j = n-1; j > 0; j = j-1)
{
  for (k = j-1; k >= 0; k = k-1)
  {
    if (LIST[k] > LIST[j])
    {
      TEMP = LIST[k];
      LIST[k] = LIST[j];
      LIST[j] = TEMP;
    }
  }
}
```

Sortiranje niza

|       |         |                 |                                   |
|-------|---------|-----------------|-----------------------------------|
|       | MOVEA.L | #LIST,A1        | Pointer to the start of the list. |
|       | MOVE    | N,D1            | Initialize outer loop             |
|       | SUBQ    | #1,D1           | index j in D1.                    |
| OUTER | MOVE    | D1,D2           | Initialize inner loop             |
|       | SUBQ    | #1,D2           | index k in D2.                    |
|       | MOVE.B  | (A1,D1),D3      | Current maximum value in D3.      |
| INNER | CMP.B   | D3,(A1,D2)      | If LIST(k) <= [D3],               |
|       | BLE     | NEXT            | do not exchange.                  |
|       | MOVE.B  | (A1,D2),(A1,D1) | Interchange LIST(k)               |
|       | MOVE.B  | D3,(A1,D2)      | and LIST(j) and load              |
|       | MOVE.B  | (A1,D1),D3      | new maximum into D3.              |
| NEXT  | DBRA    | D2,INNER        | Decrement counters k and j        |
|       | SUBQ    | #1,D1           | and branch back                   |
|       | BGT     | OUTER           | if not finished.                  |



## Primeri programa kod Motorole 68000

### Subroutine

|           |         |           |                                   |
|-----------|---------|-----------|-----------------------------------|
| INSERTION | CMPA.L  | #0,A0     | A0 is RHEAD.                      |
|           | BGT     | HEAD      |                                   |
|           | MOVEA.L | A1,A0     | A1 is RNEWREC.                    |
| HEAD      | RTS     |           |                                   |
|           | CMP.L   | (A0),(A1) | Compare ID of new record to head. |
|           | BGT     | SEARCH    |                                   |
|           | MOVEA.L | A0,A1     | New record becomes head.          |
|           | MOVEA.L | A1,A0     |                                   |
| SEARCH    | RTS     |           |                                   |
| LOOP      | MOVEA.L | A0,A2     | A2 is RCURRENT.                   |
|           | MOVEA.L | 4(A2),A3  | A3 is RNEXT.                      |
|           | CMPA.L  | #0,A3     |                                   |
|           | BEQ     | TAIL      |                                   |
|           | CMP.L   | (A3),(A1) |                                   |
|           | BLT     | INSERT    |                                   |
|           | MOVEA.L | A3,A2     | Go to next record.                |
|           | BRA     | LOOP      |                                   |
| INSERT    | MOVEA.L | A2,4(A1)  |                                   |
| TAIL      | MOVEA.L | A1,4(A2)  |                                   |
|           | RTS     |           |                                   |



## Primeri programa kod Motorole 68000

### Subroutine

|          |         |          |                     |
|----------|---------|----------|---------------------|
| DELETION | CMP.L   | (A0),D1  | D1 is RIDNUM.       |
|          | BGT     | SEARCH   |                     |
|          | MOVEA.L | 4(A0),A0 | Delete head record. |
|          | RTS     |          |                     |
| SEARCH   | MOVEA.L | A0,A2    | A2 is RCURRENT.     |
| LOOP     | MOVEA.L | 4(A2),A3 | A3 is RNEXT.        |
|          | CMP.L   | (A3),D1  |                     |
|          | BEQ     | DELETE   |                     |
|          | MOVEA.L | A3,A2    |                     |
|          | BRA     | LOOP     |                     |
| DELETE   | MOVEA.L | 4(A3),D2 | D2 is RTEMP.        |
|          | MOVEA.L | D2,4(A2) |                     |
|          | RTS     |          |                     |



## Zadaci

- 
- 
-