Upute za prvi ciklus laboratorijskih vježbi iz Arhitekture računala 2

1. Cilj vježbe

Cilj vježbe je upoznavanje s osnovama programiranja mikroprocesora Motorola MC68000 u asembleru.

2. Zadaci

1. Upoznati se s radnim okruženjem simulatora na kojemu se obavljaju vježbe i s arhitekturom procesora MC680000 (pročitati 3., 4. i 5. poglavlje ovih materijala)

2. (a) Pretipkati sljedeći jednostavni asemblerski program kojim se ostvaruje *sortiranje znakovnog niza (uzlazno)* [komentare nije nužno pretipkavati]:

```
        MOVE.L #NIZ,A1    * inicijalizacija; adresa niza postavlja se u reg. A1

LOOP1   CMP.B #0,(A1)     * jesmo li dosli na kraj niza?
        BEQ KRAJ
        MOVE.L A1,A2
LOOP2   ADD.L #1,A2
        CMP.B #0,(A2)
        BEQ DALJE

        * usporedba elementa na adresi A1 i elementa na adresi A2
        MOVE.B (A2),D1
        CMP.B (A1),D1
        BGE LOOP2

        * ako je element na adresi A1 veći od onoga na adresi A2, zamjenjujemo ih
        MOVE.B (A1),D0
        MOVE.B (A2),(A1)
        MOVE.B D0,(A2)

        BRA LOOP2

DALJE   ADD.L #1,A1
        BRA LOOP1

        * ispis sortiranog niza
KRAJ    MOVE.L #13,D0
        MOVE.L #NIZ,A1
        TRAP #15

        * definicija izvornog niza
NIZ     DC.B 'ARHITEKTURA RACUNALA 2',0
```

   (b) Prevesti program i isprobati njegovo izvođenje *odjednom* kao i *instrukciju po instrukciju*.
   (c) Analizirati način funkcioniranja programa.
   (d) Modificirati program tako da sortira niz *silazno* umjesto *uzlazno*.
   (e) Izbaciti definiciju znakovnog niza (zadnji redak: `"NIZ DC.B ..."`) iz izvornog koda programa i umjesto toga definirati niz izravnim upisivanjem u memoriju. Ponoviti izvođenje programa.

3. Riješiti zadatke koji će biti zadani u laboratoriju.

3. Opis radnog okruženja

Vježbe se obavljaju pod razvojnim okruženjem *EASy68K* za operacijski sustav MS Windows. *EASy68K* se sastoji od dva dijela: (i) editor/prevoditelj [izvršna datoteka EDIT68K.EXE] i (ii) simulator [izvršna datoteka SIM68K.EXE]. Zahvaljujući intuitivnom i preglednom grafičkom sučelju, rad s razvojnim okruženjem je vrlo jednostavan. Zato će ovdje biti dostatan vrlo kratak opis.

Rad započinjemo pokretanjem editora [EDIT68K.EXE]. Neposredno po pokretanju, editor već sadrži "kostur" programa sljedećeg oblika:

```
*-----------------------------------------------------------
* Program    :
* Written by :
* Date       :
* Description:
*-----------------------------------------------------------
START   ORG     $1000



        MOVE.B  #9,D0
        TRAP    #15             Halt Simulator

        END     START
```

Naredbe programa unose se neposredno ispod teksta "`START  ORG  $1000`", uz poštivanje sljedećih pravila:

(1) Komentari započinju zvjezdicom:
`*OVO JE KOMENTAR.`
(2) Naredbe moraju biti odmaknute od početka retka (tj. lijevog ruba) barem jednim razmakom ili TABom.
(3) Sve što počinje uz sam lijevi rub tumači se kao labela.

Kad je program upisan, potrebno ga je prevesti i pokrenuti, što se postiže pritiskom na tipku "Assebmle source" (krajnje desna tipka s crnim trokutićem u traci s alatima) ili tipkom F9. Prevoditelj će prijaviti greške ako one postoje, a ako ih nema, omogućit će izvođenje programa (gumb "Execute").

Pritiskom na "Execute", *automatski* se pokreće simulator [SIM68K]. Na vrhu prozora simulatora nalaze se meniji i traka s alatima. Ispod njih, još uvijek u gornjem dijelu, prikazan je sadržaj svih registara procesora (u registre je moguće i izravno upisivati nove vrijednosti). U donjem dijelu ekrana nalazi se listing programa.

Program možemo pokrenuti *odjednom*, tipkom "Run" (tipka s crnim trokutićem, druga s lijeva u traci s alatima), odnosno tipkom F9, ili ga možemo izvoditi *instrukciju po instrukciju* tipkama "Step over" ili "Trace into" (peta i šesta tipka u traci s alatima; odgovaraju im funkcijske tipke F8 i F7 na tipkovnici), što može biti posebno korisno kod "debugiranja".

Izbornik "View" omogućava pregled i izmjenu sadržaja drugih resursa računala, kao što su *memorija*, *stog*, itd.

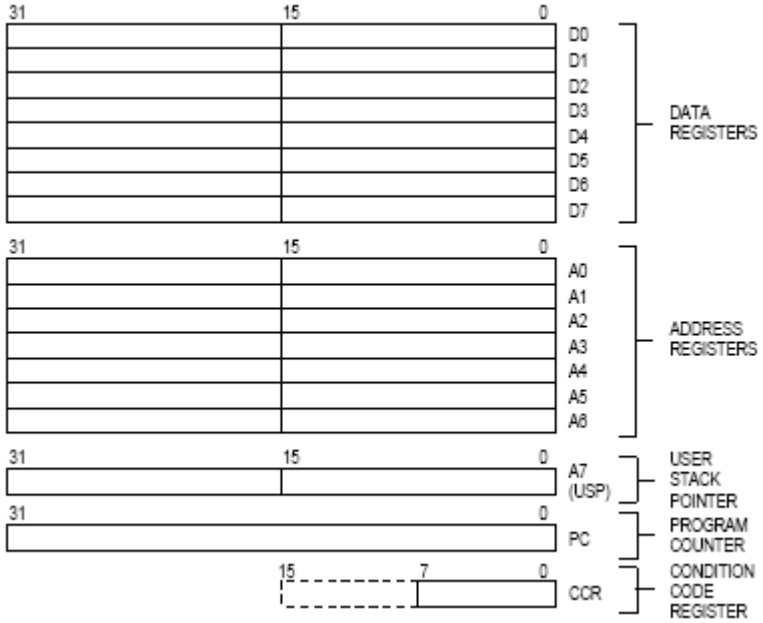
4. Arhitektura mikroprocesora Motorola MC68000

U okviru ovih materijala dan je vrlo kratak osvrt na arhitekturu mikroprocesora MC68000, priložen je kratak opis procesora te sažeta tablica instrukcija. Detaljan opis procesora i skupa instrukcija može se naći u knjizi "Motorola M68000 Family – Programmer's Reference Manual", koju možete naći u .PDF formatu na našem webu:

Da bi mogao uspješno pristupiti programiranju u asembleru ili strojnom jeziku, programer mora poznavati sljedeće tri značajke arhitekture procesora: (1) "programski model", tj. skup registara; (2) načine adresiranja; te (3) skup instrukcija. Ove tri značajke biti će zasebno opisane u nastavku.

4.1. Programski model mikroprocesora MC68000

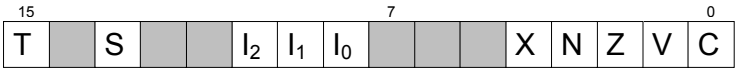
Pod pojmom *programskog modela mikroprocesora* razumijeva se skup registara dostupnih programeru u strojnom jeziku ili asembleru. Slika prikazuje programski model mikroprocesora MC68000.



Mikroprocesor MC68000 ima osam 32-bitovnih *regisatra podataka*, koji se označavaju s D0 do D7. Svi podatkovni registri se mogu koristiti na jednak način, a upotrebljavaju se za rukovanje 8-, 16- i 32-bitovnim podacima. MC 68000 ima i osam 32-bitovnih *adresnih registara* (A0-A7) koji su namijenjeni pohranjivanju i baratanju adresama. Registar A7 je specifičan po tome što se koristi kao *kazalo stoga*. Mikroprocesor MC 68000 ima dva stoga – *korisnički* i *nadgledni*, koji se koriste u skladu s načinom rada u kojem se procesor nalazi (korisnički/nadgledni). Zato je adresni registar A7 izveden kao dvostruki registar, čiji je samo jedan (32-bitovni) dio vidljiv u jednom trenutku. Za stavljanje na stog nije predviđena posebna naredba, već se to obavlja naredbom **`MOVE`**.

Programsko brojilo je širine 32-bita, ali se efektivno koristi samo 24-bitovna adresa, pa je omogućeno izravno adresiranje 16 MB memorije.

Važan dio programskog modela je i *registar stanja* ili *statusni registar* SR. To je 16-bitovni registar, koji se sastoji od korisničkog (bitovi SR_7 do SR_0) i sistemskog dijela (bitovi SR_{15} do SR_8). Korisnički dio se naziva i CCR (*condition code register*) i u njemu se nalaze pohranjene zastavice. Sistemskom dijelu SR može se pristupiti samo u nadglednom načinu rada procesora. Raspored zastavica prikazan je na slici.



U korisničkom dijelu registra stanja su zastavice X (*extend*) koja u većini slučajeva ima istu vrijednost kao i zastavica C, te zastavice N (*negative*), Z (*zero*), V (*overflow*) i C (*carry*).

U sistemskom dijelu registra stanja nalazi se također pet zastavica. Zastavica T (*trace mode*) označava da je omogućeno praćenje, dok S (*supervisor state*) označava da je procesor u nadglednom načina (odnosno u stanju obrade iznimke). Zastavice I_2 do I_0 predstavljaju *prekidnu masku*, tj. označavaju koje razine prekida će biti prihvaćene na obradu.

4.2. Načini adresiranja

Općenito, strojne instrukcije se izvode nad nekim operandima. Stoga svaka (ili gotovo svaka) instrukcija specificira i adrese operanada nad kojima se ona izvodi. Načini specifikacije adresa tih operanada nazivaj se *načinima adresiranja*. Za ilustraciju različitih načina adresiranja podržanih od procesora MC68000 upotrijebit ćemo naredbu **`MOVE`**, koja premješta (kopira) podatak s jednog odredišta na drugo. Sintaksa naredbe **`MOVE`** je:

```
MOVE <ea1>, <ea2>
```

gdje su **`<ea1>`** i **`<ea2>`** "efektivne adrese", tj. adrese specificirane nekim od načina adresiranja. Ovom naredbom, podatak specificiran efektivnom adresom <ea1> prebacuje se na efektivnu adresu <ea2>.

Mikroprocesor MC68000 podržava 14 načina adresiranja, navedenih u tablici.

Način adresiranja	Oblikovanje efektivne adrese
izravno adresiranje registara	Operand je u nekom od registara. Kao efektivna adresa zadaje se ime registra.
(1) adresiranje podatkovnog registra	*<ea> = Dn* (npr. **`MOVE D1,D2`**)
(2) adresiranje adresnog registra	*<ea> = An* (npr. **`MOVE A1,A2`**)
apsolutno adresiranje	Operand je u memoriji. Kao efektivna adresa zadaje se adresa memorijske lokacije.
(3) kratko apsolutno adresiranje	*<ea> = (sljedeća riječ)*
(4) dugo apsolutno adresiranje	*<ea> = (sljedeća duga riječ)* (npr. **`MOVE $1000,D2`**) Napomena: oznaka $ označava da se radi o heksadekadskoj vrijednosti.
posredno adresiranje registrom	Operand se nalazi u memoriji, a njegova adresa je u nekom od adresnih registara. Kao efektivna adresa zadaje se naziv adresnog registra u zagradama.
(5) registarsko posredno	*<ea> = (An)* (npr. **`MOVE (A1),(A2)`**)

(6) registarsko posredno s postinkrementiranjem	*<ea> = (An); An An+N* (npr. **`MOVE (A1)+,(A2)+`**)
(7) registarsko posredno s predekrementiranjem	*An An-N; <ea> = (An)* (npr. **`MOVE -(A1),-(A2)`**)
(8) registarsko posredno s pomakom	*<ea> = (An)+d_{16}* (npr. **`MOVE (5,A1),D1`**)
(9) indeksno registarsko posredno s pomakom	*<ea> = (An)+(Xn)+d_8* (npr. **`MOVE (5,A1,D1),D0`**)
relativno adresiranje u odnosu na programsko brojilo	Koristi se kod naredbi grananja. Ostvaruje se skok na adresu koja je pomaknuta za određeni iznos u odnosu na trenutnu sadržaj PC.
(10) odnosno s pomakom	<ea> = (PC)+d_{16}
(11) odnosno s indeksom i pomakom	<ea> = (PC)+(Xn)+d_8
usputno adresiranje (*immediate*)	Jedan od operanada je konstanta.
(12) usputno	podaci = jedna ili više sljedećih riječi
(13) brzo usputno (*quick immediate*)	inherentni podatak (npr. **`MOVE #5,D2`**)
implicitno adresiranje	Efektivna adresa je podrazumijevani registar.
(14) podrazumijevani registar	<ea> = SR, USP, SP, PC

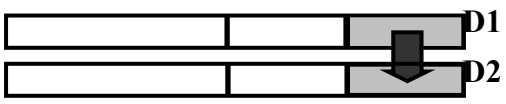
Oznake: <ea> = efektivna adresa; An = adresni registar; Dn = registar podataka; Xn = adresni ili podatkovni registar kao indeksni registar; SR = statusni registar; PC = programsko brojilo; d_8 = 8-bitovni pomak; d_{16} = 16-bitovni pomak; N = 1 za bajt, 2 za riječ i 4 za dugu riječ; () = sadržaj; = zamjena

MC 68000 podržava 5 osnovnih tipova podataka:

bit;
BCD znamenka (4 bita);
bajt (8 bita);
riječ (16-bita);
duga riječ (32-bita).

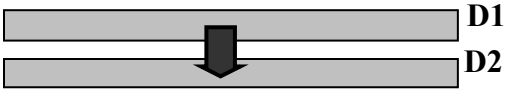
Za rukovanje bitovima i BCD znamenkama predviđeno je nekoliko posebnih instrukcija u skupu instrukcija (vidjeti tablicu s potpunim popisom instrukcija), dok se za rukovanje preostalim trima tipovima podataka (bajtom, riječju i dugom riječju) mogu koristiti sve preostale instrukcije, i to tako da se nakon naziva instrukcije navede ekstenzija "**.B**", "**.W**" ili "**.L**" Prikažimo to na primjeru naredbe MOVE, uz napomenu da isti princip vrijedi i za većinu ostalih instrukcija (za detalje vidjeti tablicu):

`MOVE.B D1,D2` Instrukcija premiješta najnižih 8 bita registra D1 u najnižih 8 bita registra D2. Preostala gornja 24 bita ostaju nepromijenjeni:



MOVE.W D1,D2 Instrukcija premiješta nižih 16 bita registra D1 u nižih 16 bita registra D2. Preostalih gornjih 16 bita ostaju nepromijenjeni:

MOVE.L D1,D2 Instrukcija premiješta svih 32 bita registra D1 u registar D2.



Važno je napomenuti da je podrazumijevana (default) veličina operanda **riječ**, tj. ako se prilikom zadavanja neke instrukcije ne navede nikakva ekstenzija (niti ".B", niti ".W", niti ".L", instrukcija će se obaviti nad 16-bitnim operandima. Dakle vrijedi jednakost:

MOVE D1,D2 = MOVE.W D1,D2

4.3. Skup instrukcija

MC68000 je procesor tipa CISC, što između ostaloga znači da ima velik i bogat skup instrukcija. Među njima ističe se nekoliko važnijih skupina, čije ćemo samo najvažnije predstavnike navesti i ukratko opisati. Cjelovit popis instrukcija i detalji vezani uz načine adresiranja koje one podržavaju nalazi se u tablici koja je dana kasnije. Bitno je napomenuti da ne podržava svaka instrukcija svaki tip adresiranja (detalje vidjeti u tablici s popisom instrukcija).

Instrukcije za prijenos podataka

MOVE <ea1>, <ea2>	prenosi podatak iz <ea1> u <ea2>

Aritmetičke instrukcije

ADD <ea1>, <ea2>	zbraja <ea1> sa <ea2>, rezultat pohranjuje u <ea2>
SUB <ea1>, <ea2>	oduzima <ea1> od <ea2> (dakle računa <ea2>-<ea1>,rezultat pohranjuje u <ea2>
CMP <ea1>, <ea2>	uspoređuje <ea1> sa <ea2> tako što oduzima <ea2> - <ea1> ali rezultat ne pohranjuje nego samo postavlja zastavice u SR
MULS <ea1>, <ea2>	množi cjelobrojno predznačno <ea1> sa <ea2>, rezultat pohranjuje u <ea2>
MULU <ea1>, <ea2>	množi cjelobrojno nepredznačno <ea1> sa <ea2>, rezultat pohranjuje u <ea2>
DIVS <ea1>, <ea2>	dijeli cjelobrojno predznačno <ea2> sa <ea1>, rezultat i ostatak pohranjuje u <ea2> (rezultat je u nižih 16 bita, a ostatak u viših 16 bita)
DIVU <ea1>, <ea2>	dijeli cjelobrojno nepredznačno <ea2> sa <ea1>, rezultat i ostatak pohranjuje u <ea2> (rezultat je u nižih 16 bita, a ostatak u viših 16 bita)

Instrukcije za posmak

LSL <e11>, <ea2>	logički posmak u lijevo operanda <ea2> za <ea1> mjesta
LSR <e11>, <ea2>	logički posmak u desno operanda <ea2> za <ea1> mjesta
ASL <e11>, <ea2>	aritmetički posmak u lijevo operanda <ea2> za <ea1> mjesta
ASR <e11>, <ea2>	aritmetički posmak u desno operanda <ea2> za <ea1> mjesta
ROL <e11>, <ea2>	kružni posmak (rotacija) u lijevo operanda <ea2> za <ea1> mjesta
ROR <e11>, <ea2>	kružni posmak (rotacija) u desno operanda <ea2> za <ea1> mjesta

Logičke instrukcije

AND <ea1>, <ea2>	logičko I <ea1> i <ea2>, rezultat pohranjuje u <ea2>
OR <ea1>, <ea2>	logičko ILI <ea1> i <ea2>, rezultat pohranjuje u <ea2>
EOR <ea1>, <ea2>	logičko ISKLJUČIVO ILI <ea1> i <ea2>, rezultat pohranjuje u <ea2>
NOT <ea>	logičko NE <ea>, rezultat pohranjuje u <ea>

Upravljačke instrukcije

JMP <ea>	bezuvjetni skok (grananje) na adresu <ea> koja se izravno zadaje
BRA <ea>	isto što i JMP <ea>
B*cc* <ea>	uvjetni skok (grananje) na adresu <ea> , pri čemu "*cc*" označava jedan od mogućih uvjeta (npr "EQ", "NE", i sl., vidjeti popis u prilogu iza tablice)
JSR <ea>	poziv potprograma na adresi <ea>
RTS	povratak iz potprograma

Instrukcije za rukovanje bitovima

BCLR <ea1>,<ea2>	očisti (postavlja u "0") bit na poziciji <ea1> u podatku <ea2>
BSET <ea1>,<ea2>	postavlja (u "1") bit na poziciji <ea1> u podatku <ea2>
BTST <ea1>,.<ea2>	ispituje bit na poziciji <ea1> u podatku <ea2> i postavlja Z=1 ako je bit=0

Instrukcije za rukovanje BCD brojevima

ABCD <ea1>,<ea2>	zbraja dva dvoznamenkasta BCD broja, rezultat sprema u <ea2>
SBCD <ea1>,<ea2>	oduzima dva dvoznamenkasta BCD broja, rezultat sprema u <ea2>

5. Ulazno-izlazne funkcije

Za ostvarivanje komunikacije s korisnikom (npr. ispis na zaslon ili učitavanje podatka s tipkovnice) koriste se ulazno-izlazne funkcije sustava. Ove funkcije pridružene su naredbi "programske zamke" TRAP #n. Naredba TRAP #n rezultirat će pozivanjem sistemskog potprograma (obično smještenog u ROM memoriji kod fizičkih sustava) koji ostvaruje ulazno-izlaznu operaciju.

U simulatoru, za sve ulazno-izlaze operacije koristi se TRAP #15, a konkretna ulazna ili izlazna operacija specificira se prethodnim smještanjem tzv. "funkcijskog broja" (koji određuje o kojoj se funkciji radi) u registar D0, i smještanjem eventualnih dodatnih parametara u druge registre. Npr, za ispis znakovnog niza na zaslon, u D0 se smješta funkcijski broj 13, a u A1 smješta se memorijska adresa tog znakovnog niza. Sljedeći primjer ispisuje na zaslon poruku "Dobar dan!":

```
        MOVE.L #13,D0
        MOVE.L #PORUKA, A1
        TRP #15

PORUKA  DC.B 'Dobar dan!",0
```

Popis svih podržanih ulazno-izlaznih funkcija, njima pripadajućih funkcijskih brojeva, kao i njihovih dodatnih parametara dostupan je kroz sustav pomoći (HELP) simulatora.

Dodatak: Popis instrukcija procesora MC 68000

INSTRUKCIJA	OPIS INSTRUKCIJE	OBLIK	VELIČINA	ZASTAVICE **X N Z V C**
ABCD	BCD_Broj1 + BCD_Broj2 + X BCD_Broj2, X(ako je došlo do preljeva). Zbraja 2 BCD broja i sadržaj X zastavice te rezultat stavlja u 2. BCD broj	**Dx,Dy** **-(Ax),-(Ay)**	B--	* U * U *
ADD	Bin_Broj1 + Bin_Broj2 Bin_Broj2 Zbraja 2 broja i rezultat stavlja u 2. broj	**Dn,<ea>** **<ea>,Dn**	BWL	* * * * *
ADDA	Broj + An An Zbraja Broj sa sadržajem Adresnog registra	**<ea>,An**	-WL	- - - - -
ADDI	#Broj + (ea) (ea) Zbraja neposredni podatak (#Broj) sa sadržajem (ea)	**#x,<ea>**	BWL	* * * * *
ADDQ	Brzo zbrajanje s brojevima od 1 do 8	**#<1-8>,<ea>**	BWL	* * * * *
ADDX	Bin_Broj1 + Bin_Broj2 + X Bin_Broj2 Zbraja 2 broja i sadržaj X zastavice te rezultat stavlja u 2. broj	**Dy,Dx** **-(Ay),-(Ax)**	BWL	* * * * *
AND	Binarno 'I' između dva broja i rezultat u drugi broj	**<ea>,Dn** **Dn,<ea>**	BWL	- * * 0 0
ANDI	Binarno 'I' između neposrednog podatka (#Broj) i broja te rezultat u drugi broj	**#<data>,<ea>**	BWL	- * * 0 0
ASL	Aritmetički posmak ulijevo. Najviši bit ostaje nepromijenjen i kopira se u C i X zastavice, u najniži bit (LSB) se zapisuje 0.	**#<1-8>,Dy** **Dx,Dy** **<ea>**	BWL	* * * * *
ASR	Aritmetički posmak udesno. Najviši bit ostaje nepromijenjen. Najniži bit kopira se u C i X zastavice.	**#<1-8>,Dy** **Dx,Dy** **<ea>**	BWL	* * * * *
Bcc	Skok ako je ispunjen uvjet 'cc'	**Bcc.S <label>** **Bcc.W <label>**	BW-	- - - - -
BCHG	Ispituje bit i mijenja njegovo stanje. Z = 1 ako je bit bio 0	**Dn,<ea>** **#<data>,<ea>**	B-L	- - * - -
BCLR	Ispituje bit i postavlja ga na 0. Z = 1 ako je bit bio 0	**Dn,<ea>** **#<data>,<ea>**	B-L	- - * - -
BSET	Ispituje bit i postavlja ga na 1. Z = 1 ako je bit bio 0	**Dn,<ea>** **#<data>,<ea>**	B-L	- - * - -
BSR	Skok na potprogram. Povratak iz potprograma instrukcijom RTS	**BSR.S <label>** **BSR.W <label>**	BW-	- - - - -
BTST	Ispituje bit. Z = 1 ako je bit bio 0	**Dn,<ea>** **#<data>,<ea>**	B-L	- - * - -
CHK	Ispituje da li se sadržaj (ea) nalazi između 0 i vrijednosti Dn. Ako se nalazi dodgađa se izuzetak (TRAP)	**<ea>,Dn**	-W-	- * U U U
CLR	Sadržaj (ea) postavlja na 0	**<ea>**	BWL	- 0 1 0 0
CMP	Oduzima (ea) od Dn i postavlja zastavice. Sadržaj (ea) i Dn se ne mijenja. Koristi se za uspoređivanje dvaju brojeva.	**<ea>,Dn**	BWL	- * * * *
CMPA	Oduzima (ea) od An i postavlja zastavice. Sadržaj (ea) i An se ne mijenja. Koristi se za uspoređivanje dvaju adresa.	**<ea>,An**	-WL	- * * * *
CMPI	Isto kao i CMP, ali sa neposrednim podatkom.	**#<data>,<ea>**	BWL	- * * * *
CMPM	Uspoređuje dva broja čije su memorijske lokacije u Ax i Ay	**(Ay)+,(Ax)+**	BWL	- * * * *
DBcc	Ako uvjet 'cc' nije ispunjen smanjuje Dn za 1, i ako je Dn različit od '-1', skače na <label>	**DBcc** **Dn,<label>**	-W-	- - - - -
DIVS	Dijeli cjelobrojno predznačno Bin_Broj2 sa Bin_Broj1. Ostatak.w i Rezultat.w sprema u Bin_Broj2.l	**<ea>,Dn**	-W-	- * * * 0
DIVU	Dijeli cjelobrojno nepredznačno Bin_Broj2 sa Bin_Broj1. Ostatak.w i Rezultat.w sprema u Bin_Broj2.l	**<ea>,Dn**	-W-	- * * * 0
EOR	Binarno 'isključivo ILI' između dva broja i rezultat u drugi broj	**Dn,<ea>**	BWL	- * * 0 0
EORI	Binarno 'isključivo ILI' između neposrednog podatka (#Broj) i broja te rezultat u drugi broj	**#<data>,<ea>**	BWL	- * * 0 0
EXG	Zamjenjuje sadržaje dvaju registara	**Rx,Ry**	--L	- - - - -
EXT	Proširuje (Dn.b na Dn.w) ili (Dn.w na Dn.l) tako da kopira najviši bit (Dn.b ili Dn.w) preko dijela registra na koji se vrijednost registra proširuje	**Dn**	-WL	- * * 0 0
ILLEGAL	Instrukcija izaziva iznimku (Exception)	**ILLEGAL**		- - - - -
JMP	Bezuvjetni skok na (ea)	**<ea>**		- - - - -
JSR	Bezuvjetni skok na potprogram (ea)	**<ea>**		- - - - -
LEA	Stavlja (ea) u An	**<ea>,An**	--L	- - - - -
LINK	Dodaje prostor na stog tako da stari pokazivač stoga spremi na stari stog i zatim napuni pokazivač stoga sa sadržajem An +	**An,#<displac>**		- - - - -

	#pomak			
LSL	Posmak ulijevo. Najniži bit postaje 0, najviši bit se kopira u C,X.	**Dx,Dy** **#<1-8>,Dy** **<ea>**	BWL	* * * 0 *
LSR	Posmak udesno. Najviši bit postaje 0, najniži bit se kopira u C,X.	**Dx,Dy** **#<1-8>,Dy** **<ea>**	BWL	* * * 0 *
MOVE	Kopira vrijednost iz (ea1) u (ea2)	**<ea>,<ea>**	BWL	- * * 0 0
MOVE	Kopira vrijednost (ea).b u CCR	**<ea>,CCR**	–W–	I I I I I
MOVE	Kopira vrijednost (ea).w u SR	**<ea>,SR**	–W–	I I I I I
MOVE	Kopira vrijednost SR u (ea).w	**SR,<ea>**	–W–	- - - - -
MOVE	Kopira vrijednost USP u An registar, ili vrijednost An u USP	**USP,An** **An,USP**	––L	- - - - -
MOVEA	Kopira vrijednost (ea) u An	**<ea>,An**	–WL	- - - - -
MOVEM	Kopira navedene registre na (ea) ili sa (ea) u navedene registre	**<rglst>,<ea>** **<ea>,<rglst>**	–WL	- - - - -
MOVEP	Stavlja pojedinačne bajtove iz Dn registra na na gornje bajtove riječi na koje pokazuje x(An) ili obrnuto (pri čitanju iz memorije)	**Dn,x(An)** **x(An),Dn**	–WL	- - - - -
MOVEQ	Stavlja 8-bitnu predznačenu vrijednost u Dn kao dugu riječ	**#<-128,** **+127>,Dn**	––L	- * * 0 0
MULS	Množi cjelobrojno predznačeno Bin_Broj1i Bin_Broj2.	**<ea>,Dn**	–W–	- * * 0 0
MULU	Množi cjelobrojno nepredznačeno Bin_Broj1i Bin_Broj2.	**<ea>,Dn**	–W–	- * * 0 0
NBCD	Negira BCD brojeve i zbraja ih sa X zastavicom. (0 - BCD_broj - X BCD_broj, X)	**<ea>**	B––	* U * U *
NEG	Mijenja predznak broju u (ea) (0 - broj broj)	**<ea>**	BWL	* * * * *
NEGX	Mijenja predznak broju (ea)+X (0 - broj -X broj, X)	**<ea>**	BWL	* * * * *
NOP	Ne radi baš ništa	**NOP**		- - - - -
NOT	Komplementira broj. Binarni 'NOT' broja	**<ea>**	BWL	- * * 0 0
OR	Binarni 'OR' broja	**<ea>,Dn** **Dn,<ea>**	BWL	- * * 0 0
ORI	Binarni 'OR' broja sa neposrednim podatkom	**#<data>,<ea>**	BWL	- * * 0 0
PEA	Stavlja (ea) na stog	**<ea>**	––L	- - - - -
RESET	Resetira sve vanjske uređaje	**RESET**		- - - - -
ROL	Rotacija ulijevo bitova unutar registra. Najviši bit se kopira u C zastavicu, svi ostali bitovi se posmiču u lijevo, C zastavica se kopira u najniži bit.	**#<1-8>,Dy** **Dx,Dy** **<ea>**	BWL	- * * 0 *
ROR	Rotacija udesno bitova unutar registra. Najniži bit se kopira u C zastavicu, svi ostali bitovi se posmiču u desno, C zastavica se kopira u najviši bit.	**#<1-8>,Dy** **Dx,Dy** **<ea>**	BWL	- * * 0 *
ROXL	Rotacija ulijevo bitova unutar registra preko X zastavice. Najviši bit se kopira u C i X zastavicu, svi ostali bitovi se posmiču u lijevo, X zastavica se kopira u najniži bit.	**#<1-8>,Dy** **Dx,Dy** **<ea>**	BWL	* * * 0 *
ROXR	Rotacija udesno bitova unutar registra preko X zastavice. Najniži bit se kopira u C i X zastavicu, svi ostali bitovi se posmiču u desno, X zastavica se kopira u najviši bit.	**#<1-8>,Dy** **Dx,Dy** **<ea>**	BWL	* * * 0 *
RTE	Povratak iz obrade iznimke	**RTE**		I I I I I
RTR	Povratak iz potprograma sa povratkom CCR registra	**RTR**		I I I I I
RTS	Povratak iz potprograma	**RTS**		- - - - -
SBCD	BCD_Broj2 - BCD_Broj1 - X BCD_Broj2, X (ako je došlo do podljeva). Oduzima 2 BCD broja i sadržaj X zastavice te rezultat stavlja u 2. BCD broj	**Dx,Dy** **-(Ax),-(Ay)**	B––	* U * U *
Scc	U bajt na (ea) upisuje $FF (-1) ako je cc istinito inače upisuje 0	**<ea>**	B––	- - - - -
STOP	Stavlja neposredni podatak u SR i zaustavlja izvršavanje naredbe dok se ne dogodi TRACE prekid ili RESET iznimka	**#<data>**		I I I I I
SUB	Oduzima dva broja. (BIN_broj2 - BIN_broj1 BIN_broj2)	**Dn,<ea>** **<ea>,Dn**	BWL	* * * * *
SUBA	Oduzima broj od An. (An - BIN_broj1 An)	**<ea>,An**	–WL	- - - - -
SUBI	Oduzima neposredni podatak od broja. (BIN_broj2 - #broj BIN_broj2)	**#x,<ea>**	BWL	* * * * *
SUBQ	Brzo oduzimanje neposrednih podataka iz intervala (1, 8) od broja.	**#<data>,<ea>**	BWL	* * * * *
SUBX	Oduzima dva broja sa X zastavicom.	**Dy,Dx**	BWL	* * * * *

	(BIN_broj2 - BIN_broj1 - X BIN_broj2, X)	**-(Ay),-(Ax)**		
SWAP	Mijenja položaj riječi unutar duge riječi Dn.	**Dn**	-W-	- * * 0 0
TAS	Ispituje sadržaj (ea), postavlja najviši bit u (ea) i postavlja N ili Z zastavicu	**<ea>**	B--	- * * 0 0
TRAP	Započinje izvršavanje TRAP iznimke.	**#<vector>**		- - - - -
TRAPV	Započinje izvršavanje TRAP iznimke ako je V zastavica postavljena	**TRAPV**		- - - - -
TST	Ispituje sadržaj (ea) i postavlja N ili Z zastavicu	**<ea>**	BWL	- * * 0 0
UNLK	Oslobađa stog koji je rezervirala LINK instrukcija. An registar se ne smije promjeniti od izvođenja LINK instrukcije.	**An**		- - - - -

OPIS OZNAKA ZA ZASTAVICE

*****	Postavlja se u ovisnosti o rezultatu operacije
-	Ne mijenja se
0	Postavljena na 0
1	Postavljena na 0
U	Stanje nakon operacije nije definirano
I	Postavljeno s neposrednim podatkom

OPIS OZNAKA ZA OBLIKE INSTRUKCIJA

<ea>	Efektivna adresa
<data>	Neposredni podatak
<label>	Labela
<vector>	Vektor TRAP iznimke
<rg.lst>	Lista registara koje prebacuje MOVEM instrukcija
<displ.>	Veličina novog stoga kojeg kreira LINK instrukcija

Adresni načini

```
Data Register Direct                              Dn
Address Register Direct                           An
Address Register Indirect                        (An)
Address Register Indirect with Post-Increment    (An)+
Address Register Indirect with Pre-Decrement     -(An)
Address Register Indirect with Displacement      w(An)
Address Register Indirect with Index            b(An,Rx)
Absolute Short                                     w
Absolute Long                                      l
Program Counter with Displacement               w(PC)
Program Counter with Index                     b(PC,Rx)
Immediate                                        #x
Status Register                                  SR
Condition Code Register                          CCR
```

Uvjeti koji se koriste za Bcc , DBcc i Scc instrukcije

Uvjeti koji se postave nakon izvođenja CMP D0,D1 instrukcije.

Odnos	**Nepredznačni**	**Predznačni**
D1 < D0	CS - Carry Bit Set	LT - Less Than
D1 <= D0	LS - Lower or Same	LE - Less than or Equal
D1 = D0	EQ - Equal (Z-bit Set)	EQ - Equal (Z-bit Set)
D1 != D0	NE - Not Equal (Z-bit Clear)	NE - Not Equal (Z-bit Clear)
D1 > D0	HI - HIgher than	GT - Greater Than
D1 >= D0	CC - Carry Bit Clear	GE - Greater than or Equal
	PL - PLus (N-bit Clear)	MI - Minus (N-bit Set)
	VC - V-bit Clear (No Overflow)	VS - V-bit Set (Overflow)
	RA - BRanch Always	

SAMO DBcc - F - Never Terminate (DBRA is an alternate to DBF)
T - Always Terminate

SAMO Scc - SF - Never Set
ST - Always Set