Upute za drugi ciklus laboratorijskih vježbi iz Arhitekture računala 2

1. Tema vježbe

Tema vježbe je rad na fizičkom sustavu temeljenom na procesoru Motorola MC68000/MC68020.

2. Zadaci za pripremu

Za pripremu je potrebno u simulatoru Easy68k (koji je korišten u prvom ciklusu laboratorijskih vježbi i dostupan na webu) riješiti *jedan* od sljedećih zadataka *(ovisno o terminu u kojemu student obavlja vježbu)*:

A. *(Grupe ponedjeljkom)* Napisati potprogram za množenje dvaju 32-bitnih cijelih brojeva, pri čemu rezultat može biti 64-bitni (Napomena: naredbe MULS i MULU procesora MC68000 podržavaju samo 16-bitne operande). Operandi se u potprogram prenose pomoću podatkovnih registara D0 i D1, a rezultat se vraća u registrima D2 i D3 (promatranima kao registarski par koji sadrži ukupnu vrijednost rezultata). Glavni program mora s tipkovnice učitati operande u heksadekadskom obliku, pozvati potprogram, te po povratku iz potprograma ispisati rezultat u heksadekadskom obliku na zaslon.

B. *(Grupe utorkom)* Napisati potprogram za dijeljenje dvaju 32-bitnih cijelih brojeva, pri čemu rezultat u općem slučaju može biti 32-bitni (Napomena: naredbe DIVS i DIVU procesora MC68000 podržavaju samo 16-bitne operande). Operandi se u potprogram prenose pomoću podatkovnih registara D0 i D1, a rezultat i ostatak vraćaju u registrima D2 odnosno D3. Glavni program mora s tipkovnice učitati operande u heksadekadskom obliku, pozvati potprogram, te po povratku iz potprograma ispisati rezultat u heksadekadskom obliku na zaslon.

C. *(Grupe srijedom)* Napisati potprograme koji će obavljati aritmetičke operacije cjelobrojnog množenja i dijeljenja (* i /) nad kompleksnim brojevima s 16-bitnim cjelobrojnim komponentama. Realne i imaginarne dijelove zadanih kompleksnih brojeva prenositi u potprograme pomoću podatkovnih registara D0-D3 a rezultat vratiti u registrima D4 i D5. Glavni program mora s tipkovnice učitati realne i imaginarne dijelove operanada u heksadekadskom obliku te oznaku operacije, pozvati odgovarajući potprogram, te po povratku iz potprograma na zaslonu ispisati rezultat. Rezultat operacije treba ispisati heksadekadski u obliku "Re+Im*i".

Popis instrukcija procesora Motorola MC68000 dan je u Dodatku C. Tablica ASCII kodova dana je u Dodatku D.

3. Rad u laboratoriju

U laboratoriju postoje dvije vrste sustava. Zbog ograničene količine opreme u laboratoriju, neće svi studenti raditi na istim sustavima, nego jedan dio na jednom tipu, a drugi dio na drugom tipu:

- Jedan dio studenata će raditi na sustavima temeljenima na mikroprocesoru Motorola MC68000, pod nadzornim programom TUTOR 1.2. Način rada s ovim sustavom opisan je u **Dodatku A**.

- Drugi će dio studenata obavljati vježbe na sustavu EVM 68020, temeljenom na mikroprocesoru MC68020 (snažniji nasljednik MC68000, koji je s njom u potpunosti kompatibilan). Način rada s ovim sustavom opisan je u **Dodatku B**.

Zadaci za rad u laboratoriju:

(1) Uz pomoć asistenta, utvrditi **tip sustava** na kojemu obavljate vježbu, te ovisno o tome proučiti **način rada sa sustavom** u **Dodatku A** ili **Dodatku B** (unos programa, ispis i modifikacija sadržaja registara, pokretanje programa, postavljanje prekidnih točaka, i slično).

(2) Pretipkati najprije samo **potprogram(e)** iz pripreme, te ispitati njihovu funkcionalnost izravnim upisom operanada u registre, postavljanjem prekidne točke na kraj potprograma, pokretanjem izvođenja, te pregledom sadržaja registara koji sadrže rezultat.

(3) **Glavni program**, koji ostvaruje komunikaciju s korisnikom (učitavanje operanada s tipkovnice i ispis rezultata na zaslon) biti će potrebno **modificirati** u odnosu na rješenje iz pripreme. Naime, premda je i u simulatoru i u laboratoriju riječ o istom (ili potpuno kompatibilnom) procesoru, s jednakim skupom instrukcija, valja razumjeti da naredba TRAP #n, kojom se ostvaruju ulazno-izlazne operacije, zapravo poziva jedan *sistemski potprogram* koji se obično nalazi u ROM memoriji sustava. Taj sistemski potprogram, dakle, nema veze sa samim procesorom, te može varirati (biti različit) od sustava do sustava. Tako se, primjerice, u simulatoru *Easy68K* koristi TRAP#15, u laboratorijskom sustavu pod nadzornim programom TUTOR koristi se TRAP #14, a u sustavu EVM 68020 TRAP #11. Detaljniji opis načina korištenja ulazno-izlaznih funkcija na pojedinom laboratorijskom sustavu, te popis raspoloživih ulazno-izlaznih funkcija nalaze se na kraju Dodatka A, odnosno Dodatka B.

Napomena: laboratorijski sustavi ne podržavaju *labele*. Umjesto labela potrebno je koristiti numeričke vrijednosti za adrese.

Dodatak A

Sustav temeljen na procesoru MC68000 pod nadzornim programom TUTOR

Vježbe se obavljaju na školskom sustavu MC 68000 pod upravljanjem nadglednog programa TUTOR. Valja voditi računa o **memorijskom prostoru koji je predviđen za korisničke programe**, a on se nalazi u memorijskom području **$0900 do $7FFF.**

Nadgledni program TUTOR

TUTOR omogućava pregled i izmjenu sadržaja registara, pregled i izmjenu sadržaja memorijskih lokacija, unos asemblerskih programa, pokretanje izvođenja programa, postavljanje prekidnih točaka i slično. Pregled naredbi dan je u tablici, a njihov detaljniji opis može se pronaći u uputama na radnom mjestu u laboratoriju. Bitno je nagasiti da TUTOR razlikuje velika i mala slova te je **sve naredbe potrebno zadavati velikim slovima**.

Naredba	Opis
MD <adresa> [<broj>][;<opcije>]	Prikaz sadržaja navedenog broja memorijskih lokacija (bajtova) počevši od zadane adrese. Opcijama se može zadati format podatka (B - bajt, W - riječ, L – duga riječ) ili ispis disasemblirane liste naredbi (opcija DI).
MM <adresa> [;<opcije>]	Prikaz i omogućavanje izmjene sadržaja navedene memorijske lokacije. Opcije su jednake kao kod naredbe **MD**. Posebno je bitna opcija "**;DI**" koja omogućuje unos asemblerskog programa u memoriju računala.
MS <adresa> <podatak>	Blok memorije počevši od zadane adrese puni se zadanim podacima. Npr. **MS 2000, ABCD** postavlja riječ $ABCD na memorijsku adresu $2000. **MS 2000, 'ABCD'** upisuje zadani niz znakova od adrese $2000.
.A0 do **.A7** **.D0** do **D7** **.PC, .SR** **.SS, .US**	Prikaz sadržaja navedenog registra (adresnog, podatkovnog, statusnog, programskog brojila, te nadglednog ili korisničkog kazala stoga). Ukoliko se iza registra navede novi sadržaj, on se upisuje u registar. Npr. **.A3 12AB** će u registar A3 upisati broj $000012AB.
DF	Prikazuje sadržaje svih registara procesora MC 68000.
BF <poč_adr><završ_adr><podatak>	Punjenje bloka memorije zadanog početnom i završnom adresom zadanim podatkom (zadaje se riječ). Npr. **BF 2000 2006 FFFF** upisat će riječ $FFFF na memorijske lokacije ad adrese $2000 do $2006.
BM <poč_adr><završ_adr><odred_adr>	Premještanje bloka memorije zadanog početnom i završnom adresom na zadanu odredišnu adresu.
BT <poč_adr><završ_adr>	Ispitivanje zadanog bloka memorije upisivanjem i čitanjem podataka (*destruktivno* ispitivanje).
BS <poč_adr><završ_adr>'niz' **BS** **poč_adr><završ_adr><podatak>** **[<maska>][;<opcije>]**	Pretraživanje zadanog bloka memorije. Npr. znakovni niz bismo tražili naredbom **BS 1000 2000 'ABCD'**. Drugi oblik naredbe služi za pronalaženje numeričkog podatka, koji se može maskirati (logičkom funkcijom I), a kao opcija navodi se format podatka (B, W ili L).
DC <izraz>	Pretvaranje izraza u heksadekadski i dekadski oblik. Podrazumijevani oblik podataka je heksadekadski (eksplicitno se označava znakom $, dok se dekadski oblik specificira znakom &). Primjeri: **DC &120, DC $100, DC $15 + &12 - $A.**
BR <adresa>[;<broj_ponavljanja>]	Postavlja prekidnu točku na zadanu adresu. Ako se navede broj ponavljanja, zaustavljanje izvođenja programa događa se tek nakon zadanog broja prolazaka kroz prekidnu točku.
NOBR [<adresa1><adresa2>...]	Uklanjaju se prekidne točke s navedenih adresa. Ako se ne navede lista adresa, brišu se sve prekidne točke.

GO [<adresa>]	Pokreće se program od zadane adrese. Ako se adresa ne navede, izvođenje započinje od adrese u programskom brojilu. Izvođenje se zaustavlja ako se naiđe na prekidnu točku ili ako dođe do iznimke.
GT <završna_adresa>	Pokreće se program od adrese u programskom brojilu. Izvođenje se zaustavlja na zadanoj završnoj adresi (ili ako se naiđe na prekidnu točku ili ako dođe do iznimke). Završna adresa je zapravo privremeno postavljena prekidna točka.
GD [<adresa>]	"Izravno" se pokreće program od zadane adrese. Ako se adresa ne navede, izvođenje započinje od adrese u programskom brojilu. U ovom načinu izvođenja ne provjeravaju se prekidne točke.
TR [<broj>]	Započinje se izvođenje jedne po jedne instrukcije programa od adrese u programskom brojilu. Nakon izvršenja instrukcije, prikazuju se registri procesora i disasemblira sljedeća instrukcija. U trace-modu rada TUTOR-a pritisak na tipku <CR> ("Enter") izaziva izvršavanje sljedeće instrukcije. Za izlaz iz tog načina rada potrebno je unijeti neku naredbu i <CR>. Može se navesti broj instrukcija koje će se izvesti u slijedu.
TT <završna_adresa>	Započinje se izvođenje jedne po jedne instrukcije programa od adrese u programskom brojilu (kao TR). Instrukcije se izvode u slijedu sve do navedene završne adrese. Nakon toga se sljedeća instrukcija može izvesti sa <CR>
HE	Prikaz liste raspoloživih naredbi TUTOR-a.

Primjer: želimo unijeti kratki asemblerski program u memorju, počevši od adrese 1000 i zatim ga pokrenuti. U tu svrhu zadajemo naredbu:

```
TUTOR 1.2> MM 1000;DI
```

Nadzorni program sada očekuje unos asemblerskog programa. On ispisuje stari sadržaj dotične memorijske lokacije, nakon čega slijedi znak "?" i očekuje se korisnikov unos asemblerske naredbe. Svaka naredba unosi se tako da se obavezno stavi jedan razmak na početku (što je u donjem primjeru naglašeno znakom "□"), nakon čega slijedi sama naredba. Kad je unos cijelog programa završen, povratak u prompt ostvaruje se unosom najobičnije točke:

```
? □ MOVE.L #1,D1
? □ MOVE.L #3,D2
? □ ADD.L D1,D2
?.

TUTOR 1.2>
```

Pokretanje programa sada možemo ostvariti zadavanjem naredbe GO uz navođenje početne adrese programa (postoje i drugi načini - proučiti dokumentaciju na radnom mjestu!)

```
TUTOR 1.2> GO 1000
```

Funkcije TRAP #14

U sustavu s nadglednim programom TUTOR predviđen je niz funkcija koje omogućavaju interakciju vaših asemblerskih programa s korisnikom (npr. ispis na ekran ili učitavanje podatka s tipkovnice). Te su funkcije u ovom sustavu pridijeljene naredbi programske zamke TRAP #14. Funkcije se odabiru na temelju *funkcijskog broja*, koji se prije poziva TRAP #14 upisuje *u registar **D7***. Neke funkcije zahtijevaju i neke parametre, koji se upisuju u određene podatkovne i/ili adresne registre, te rezultat izvršavanja

vraćaju vrijednosti u nekim registrima. *Napomena: funkcije ne jamče da neće promijeniti vrijednosti ostalih registara, pa je međurezultate potrebno pohraniti prije poziva neke od funkcija !*

Raspoložive funkcije ukratko su opisane u tablici. Funkcije se mogu podijeliti u 3 grupe:

1. programi za ulaz/izlaz jednog znaka ili niza znakova sa U/I pristupa;
2. programi za pretvorbu:
3. programi za prijenos upravljanja na TUTOR.

Funkc. broj	Simbolički naziv	Opis
247	INCHE	Unos jednog znaka s tipkovnice. Znak se pohranjuje u D0 (najmanje značajan oktet). Funkcija koristi i uništava sadržaj registara D1 i A0 !
241	PORTIN1	Unos znakovnog niza s tipkovnice. Pri pozivu funkcije A5 pokazuje na blok memorije gdje se pohranjuje niz, A6 pokazuje sljedeću slobodnu lokaciju. Nakon povratka iz funkcije, A6 pokazuje prvi znak **iza** unesenog niza. Unos se završava nailaskom na <CR>.
224	PORTIN1N	Kao PORTIN1, ali se ne prelazi automatski u novi red.
248	OUTCH	Ispis jednog ASCII kodiranog znaka iz registra D0 na ekran. Funkcija koristi i uništava sadržaj registara D0, D1 i A0 !
243	OUTPUT	Ispis znakovnog niza. A5 pokazuje na blok memorije gdje je niz pohranjen, a A6 pokazuje prvi znak iza niza. Poslije izvršavanja A5 također pokazuje prvi znak iza niza.
227	OUT1CR	Ispis znakovnog niza i prelazak u novi red. A5 pokazuje na blok memorije gdje je niz pohranjen, a A6 pokazuje prvi znak iza niza. Poslije izvršavanja A5 također pokazuje prvi znak iza niza. Funkcija koristi i uništava sadržaj registara D0, D1 i A0 !
236	HEX2DEC	Pretvara (heksadekadski) broj smješten u D0 u ASCII kodiran dekadski broj. A6 pokazuje na početak memorijskog bloka u koji će se znakovi pohraniti. Završna vrijednost A6 je uvećana za broj dekadskih znamenaka. Funkcija uništava sadržaj registra D0 !
235	GETHEX	Pretvara ASCII predstavljenu heksa znamenku (zapisanu u najmanje značajnom bajtu D0) u odgovarajući broj (pohranjuje se u najmanje značajni bajt D0).
234	PUTHEX	Pretvara 1 heksa znamenku iz registra D0 u ASCII oblik. A6 pokazuje na početak memorijskog bloka u koji će se znak pohraniti. Završna vrijednost A6 je uvećana za 1. Funkcija uništava sadržaj registra D0 !
233	PUT2HX	Pretvara 2 heksa znamenke iz registra D0 u ASCII znakove. A6 pokazuje na početak memorijskog bloka u koji će se znakovi pohraniti. Završna vrijednost A6 je uvećana za 2. Funkcija uništava sadržaj registara D0 i D2 !
232	PUT4HX	Pretvara 4 heksa znamenke iz registra D0 u ASCII znakove. A6 pokazuje na početak memorijskog bloka u koji će se znakovi pohraniti. Završna vrijednost A6 je uvećana za 4. Funkcija uništava sadržaj registara D0, D1 i D2 !
231	PUT6HX	Pretvara 6 heksa znamenke iz registra D0 u ASCII znakove. A6 pokazuje na početak memorijskog bloka u koji će se znakovi pohraniti. Završna vrijednost A6 je uvećana za 6. Funkcija uništava sadržaj registara D0, D1 i D2 !
230	PUT8HX	Pretvara 8 heksa znamenke iz registra D0 u ASCII znakove. A6 pokazuje na početak memorijskog bloka u koji će se znakovi pohraniti. Završna vrijednost A6 je uvećana za 8. Funkcija uništava sadržaj registara D0, D1 i D2 !

226	GETNUMA	Pretvara ASCII kodirani heksadekadski broj (niz znakova na čiji početak pokazuje A5) u (heksadekadski) broj koji se zapisuje u D0. A6 mora pokazivati znak iza zadnje znamenke ASCII kodiranog broja.
225	GETNUMD	Pretvara ASCII kodirani dekadski broj (niz znakova na čiji početak pokazuje A5) u (heksadekadski) broj koji se zapisuje u D0. A6 mora pokazivati znak iza zadnje znamenke ASCII kodiranog broja.
229	START	Prijenos upravljanja s korisničkog programa na TUTOR. Pritom se obavlja reinicijalizacija sustava. Kazalo stoga (A7) inicijalizira se na 'SYSSTACK'. Vektori iznimki, smješteni u nižem dijelu sistemskog RAM-a također se inicijaliziraju. Gornji dio RAM-a puni se nulama. Registar statusa postavlja se u $2700 – prekidna maska razine 7 i nadgledni način rada.
228	TUTOR	Prijenos upravljanja s korisničkog programa na TUTOR, ali bez reinicijalizacije sustava. A7 i SR se inicijaliziraju kao i kod funkcije START.

Dodatak B

Sustav MC68020 EVM

Vježbe se obavljaju na sustavu EVM 68020, baziranom na mikroprocesoru MC68020. Sustav sadrži vlastitu razvojnu okolinu (potprograme za pregled memorije u heksadekadskom ili mnemoničkom obliku, za pregled sadržaja registara, jednostavan asembler i sl.). Međutim, moguće je i pisanje programa na osobnom računalu, prevođenje u objektni kod (cross-assembler) i prijenos koda na modul EVM.

Razvojna okolina

Za pristup razvojnoj pločici EVM 68020 koristi se osobno računalo s operacijskim sustavom Linux. Osobno računalo se koristi kao *terminal*, a EVM modul kao centralno računalo pod upravljanjem nadzornog programa. Komunikacija s pločicom obavlja se pomoću programa za emulaciju terminala **Minicom**.

Početak rada

Na početku rada potrebno je logirati se na računalo. Korisničko ime i lozinku dobit ćete na vježbama. Nakon toga (iz komandne linije već otvorenog terminala) možete pokrenuti emulator terminala:

```
minicom –o –M
```

Resetirajte modul EVM 68020 pritiskom na malu crnu okruglu tipku, a zatim nekoliko puta pritisnite tipku *Enter* na tipkovnici računala. Kao rezultat trebate u prozoru Minicoma dobiti poruku modula EVM 68020, koja pokazuje da je veza uspostavljena i da radite u nadglednom programu (prethodno poglavlje).

Raspodjela memorijskog prostora

Sustav sadrži radnu memoriju (RAM) veličine 32 Kb, koja je smještena u adresnom prostoru od lokacije $400000. Prvi 1 Kb koristi sustav, pa nije dostupan korisniku. **Za korisničke programe predviđene su memorijske lokacije $400600 - $407FFF.**

U memoriji sustava (EPROM) pohranjen je operativni sustav modula. Memorija je veličine 64 Kb, a zauzima memorijske lokacije $000000 - $00FFFF.

Korištenje naredbi nadglednog programa

Komunikacija s korisnikom obavlja se pod upravljanjem jednostavnog nadglednog programa, uz upotrebu niza ugrađenih naredbi. Naredbe se zadaju skraćenicama koje se sastoje od po dva slova, bez pritiska na tipku <Return>. Osnovne naredbe s objašnjenjima navedene su u nastavku.

Pregled osnovnih naredbi nadglednog programa

An - ispis sadržaja adresnog registra An, i eventualna izmjena

Novi sadržaj se unosi u heksa kodu, nakon čega treba pritisnuti tipku <ENTER>. Pritisak na <ENTER> bez prethodnog unosa ostavlja stari sadržaj u registru.

AE - upis ASCII stringa u memoriju

Nakon unosa početne adrese, zahtijeva se unos niza znakova.
<BACKSPACE> omogućava brisanje znakova unazad.

<CTRL C> sprema $8D (CR sa postavljenim bitom 7).
<CTRL L> sprema $8A (LF sa postavljenim bitom 7)
<CTRL N> sprema $0 ('\0')
<CTRL T> završetak unosa (može se podesiti naredbom ST)

AS - linijski asembler

Asembler prihvaća standardne Motoroline mnemonike, i 6 pseudo-naredbi. Pretpostavljena početna adresa je $400600, a to je ujedno i najniža iskoristiva adresa u RAM-u. Izmjena početne adrese je moguća korištenjem pseudo-naredbe "$nnnnnn". Komentari se prihvaćaju, uz uvjet da postoji bar jedna praznina iza naredbe. Linija koja počinje sa "*" također se shvaća kao komentar.

Pseudo-naredbe:
$nnnnnn unosi se nova adresa za smještanje koda

ASCII unos niza znakova - nakon naredbe slijedi razmak, a zatim niz znakova, zaključen s <ENTER>

ASCIZ unesenom nizu znakova dodaje se zaključni nul-karakter ("\0")

DATA unos okteta
pr. DATA 0,1,$10,32
- podaci se poravnavaju na granicu riječi !

END završetak unosa asemblerskog koda

*Napomena: ne postoji mogućnost korištenja labela - odredište grananja se mora odrediti kao apsolutna adresa, ili kao pomak u odnosu na programsko brojilo (BRA *+28)*

Bn - prikaz/podešavanje prekidnih točaka

Moguće je koristiti do 4 prekidne točke. Unos nove adrese postavlja novu vrijednost prekidne točke. Unos adrese 0 uklanja prekidnu točku. Pritisak na <ENTER> ostavlja prekidnu točku na starom mjestu.

CM - usporedba dvaju blokova memorije

Zadaje se početna i zvršna adresa prvog bloka, te početna adresa drugog bloka memorije. Ispisuju se lokacije na kojima je ustanovljena razlika. Ako nema tog ispisa, nije ustanovljena ni jedna razlika.

CN - nastavak izvršavanja nakon prekida

Uklanja se prekidna točka na kojoj je zaustavljeno izvršavanje programa, postavlja se prava instrukcija i od nje započinje izvršavanje.

CV - pretvorba između heksadekadskog i dekadskog broja

Pretvara dekadski broj (do 8 znamenaka) u heksadekadski, ili heksadekadski (započinje sa 'X') u dekadski. Najveći broj koji se može pretvoriti je 99999999, odnosno X5F5E0FE

Dn - ispis sadržaja registra podataka Dn, i eventualna izmjena

Novi sadržaj se unosi u heksa kodu, nakon čega treba pritisnuti tipku <ENTER>. Pritisak na <ENTER> bez prethodnog unosa ostavlja stari sadržaj u registru.

DS - disasembler
zadaje se početna i završna adresa bloka koji treba disasemblirati

FI - popunjavanje memorijskog bloka sadržajem jedne riječi
Zadaje se početna i završna adresa bloka memorije i riječ koja se treba upisati.

GO - pokretanje programa
Pokreće se izvođenje programa počevši od zadane adrese. Izvođenje se zaustavlja nailaskom na prekidnu točku, ili pritiskom na tipku RESET na modulu MC68020.

HE - pomoć - ispisuje se popis narebi nadzornog programa

IR - Inicializacija registara D0-7 & A0-6.
U sve registre se upisuje zadana vrijednost. Pritisak na tipku <ENTER> upisuje u sve registre vrijednost 0.

LW - Zadana riječ traži se u specificiranom bloku memorije.

MO - pregled i izmjena sadržaja memorijskih lokacija
Omogućava pregledavanje niza memorijskih lokacija, i izmjenjivanje njihovog sadržaja. Format dohvata je riječ (16-bita). Unose se 4 heksadekadske znamenke.
Upravljanje:
<space> spremanje unesene vrijednosti (odnosno stare ukoliko nova nije unesena), i otvaranje nove lokacije.
<strelica prema gore>
nova upisana riječ se pohranjuje na prethodnu otvorenu lokaciju (otvara se prethodna lokacija.
<ENTER> upisana riječ se pohranjuje, a zatim se izlazi iz upisa.

MV - specificirani memorijski blok se premješta na zadanu adresu.

RD - prikaz sadržaja registara

SR - pregled ili upis novog sadržaja statusnog registra

SS - korak-po-korak
Izvođenje programa instrukciju po instrukciju, počevši od zadane adrese. Pritisak na <ENTER> nastavlja izvođenje.

TR - praćenje izvođenja programa
Izvođenje programa instrukciju po instrukciju, ali bez zaustavljanja nakon svake instrukcije (stanje se ispisuje).
<CTRL S> zaustavlja ispis
<DEL> prekida izvođenje
<bilo koja tipka> nastavak

Pisanje i izvođenje programa

Programi se upisuju izravno u memoriju korištenjem naredbe AS. Pokretanje programa postiže se naredbom GO. Ukoliko se program nađe u beskonačnoj petlji, njegovo zaustavljanje može se postići pritiskom na tipku RESET (program ostaje u memoriji).

Pozivi funkcija nadzornog programa

Nadzorni program stavlja korisniku na raspolaganje niz korisnih funkcija (uglavnom povezanih s ispisom na ekran i čitanje s tastature). Te funkcije se pozivaju naredbom programske zamke (TRAP #11), iza koje slijedi podatak o broju funkcije koja se poziva. Tipičan poziv funkcije broj 5 je:

```
TRAP #11
DC.W  5
```

U tablici su navedene neke važnije funkcije.

Broj	Naziv	Opis
00	renter	Povratak iz korisničkog programa u nadgledni.
01	outch	Ispis znaka iz najmanje značajnog bajta registra D0.
05	pdata1	Ispis znakovnog niza na koji pokazuje A6, koji završava znakom '\0'. Funkcija uništava sadržaj registra D0. A6 nakin izvršavanja funkcije pokazuje iza znakovnog niza.
06	pdatam	Kao "pdata1", ali vodi računa o duljini linije (40 ili 80 znakova).
07	query	Ispisuje "?" i ASCII znak '07' (bell).
08	outhex	Ispis jedne heksa znamenke iz D0 (ASCII). Funkcija uništava sadržaj registra D0.
09	out2hx	Ispis dvije heksa znamenke iz D0.B. Funkcija uništava sadržaj registra D0.
10	out4hx	Ispis 4 heksa znamenke iz D0.W. Funkcija uništava sadržaj registra D0.
11	out8hx	Ispis 8 heksa znamenke iz D0.L. Funkcija uništava sadržaj registra D0.
12	clrf	Pomak u novi red (<CR>, <LF>).
13	getch	Čita znak s tipkovnice i stavlja ga u D0.B (bit-7 postavlja u 0).
15	gethex	Uzima heksa znamenku s tipkovnice (ASCII) i ekvivalent se stavlja u D0.B. Ako znamenka nije heksa, postavlja se C-zastavica u CCR.
16	conhex	Uzima heksa znamenku (ASCII) iz D0.B i ekvivalent pohranjuje u D0.B. Ako znamenka nije heksa, postavlja se C-zastavica u CCR.
17	gpch	Uzima znak s tipkovnice i ispisuje ga na zaslonu. Znak (ASCII) se pohranjuje u D0.B. Ako je učitani znak malo slovo, pretvara se u veliko.
19	hexin	Uzima niz heksa znamenaka do prvog znaka koji nije heksa, zadnjih 8 znamenaka stavlja u D1.L, ako ih je manje od 8 puni prazna mjesta vodećim nulama, u D0.B vraća završni znak, D2.L sadrži broj učitanih znamenaka
27	inkey	Ispituje da li je pritisnuta tipka na tipkovnici. Ako jest, briše Z zastavicu u statusnom registru (Z=0). Inače je postavlja (Z=1).

Dodatak C: Popis instrukcija procesora MC68000

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 + #pomak	**An,#<displac>**		- - - - -
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. (BIN_broj2 - BIN_broj1 - X → BIN_broj2, X)	**Dy,Dx** **-(Ay),-(Ax)**	BWL	* * * * *
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

Dodatak D

Tablica ASCII kodova

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f

ASCII Name	Description	C Escape Sequence
nul	null byte	\0
bel	bell character	\a
bs	backspace	\b
ht	horizontal tab	\t
np	formfeed	\f
nl	newline	\n
cr	carriage return	\r
vt	vertical tab	
esc	escape	
sp	space	