Građa računala

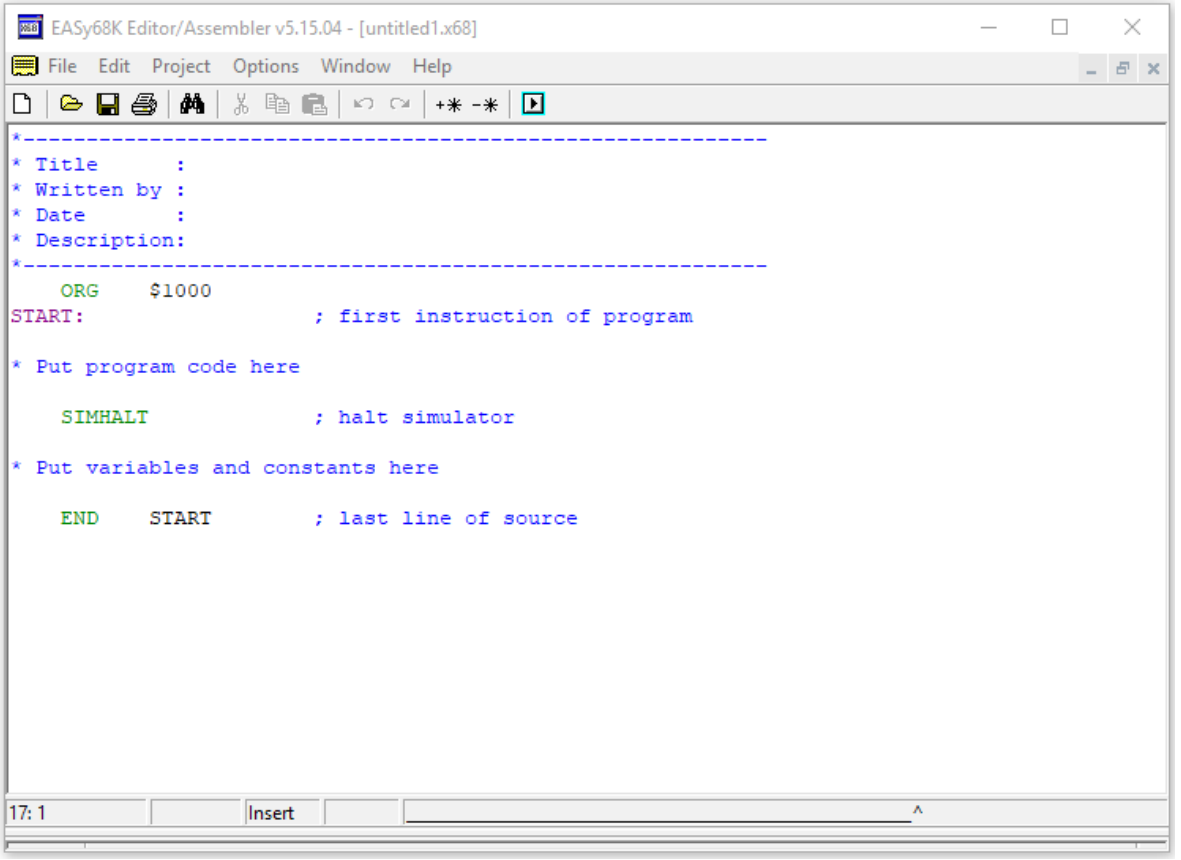
Osnovni programi u Motorola 68k asembleru

Easy68k simulator

- Omogućava pristup svim dijelovima stroja
  - Registri procesora
  - Memorija
  - Vanjski uređaji (kasnije!)
- Mogući uvid u izvršavanje programa na razini pojedinih instrukcija
  - Simulator može stati s izvršavanjem nakon svake instrukcije
  - *Break points*

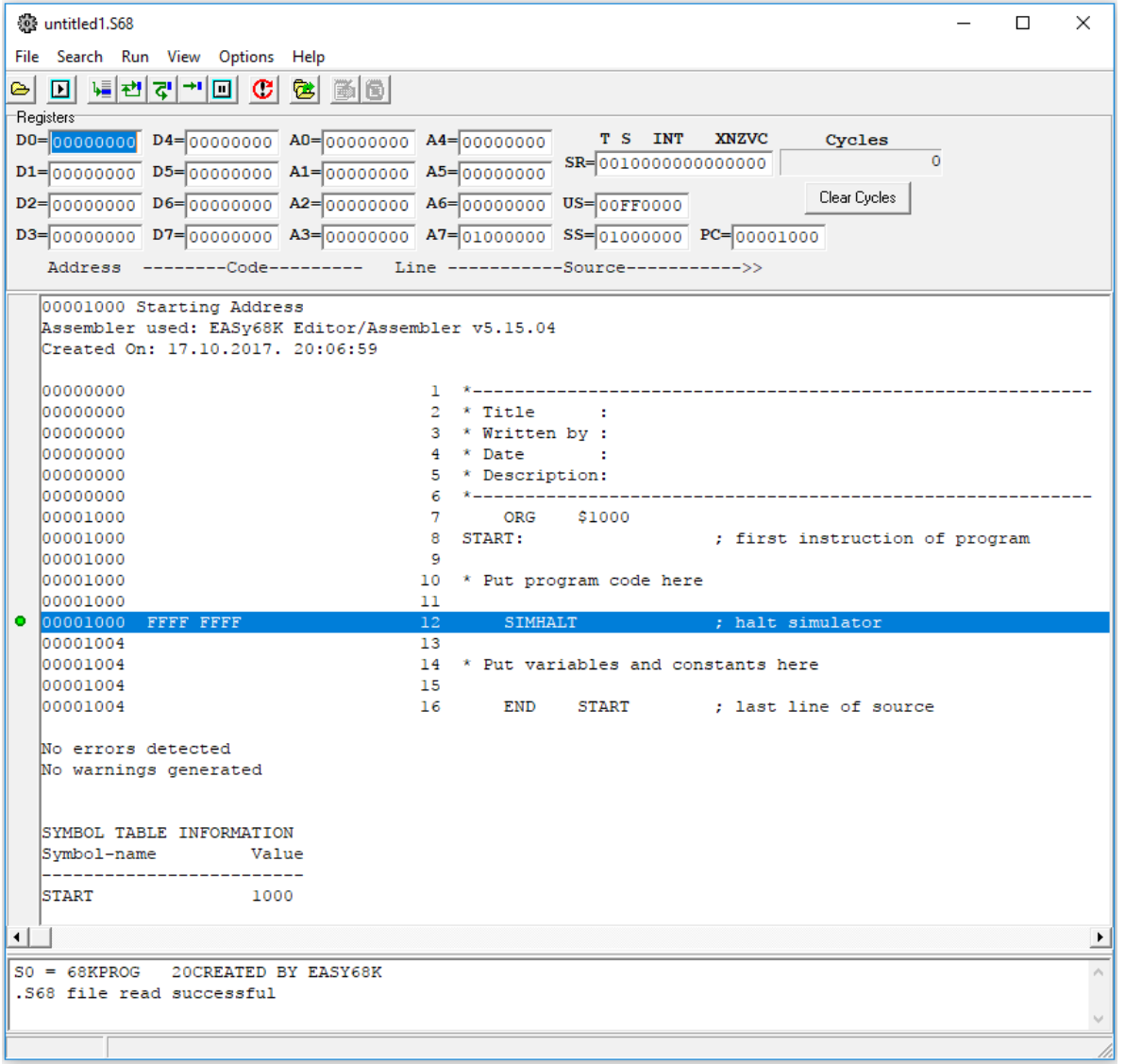
Easy68k simulator

- Editor



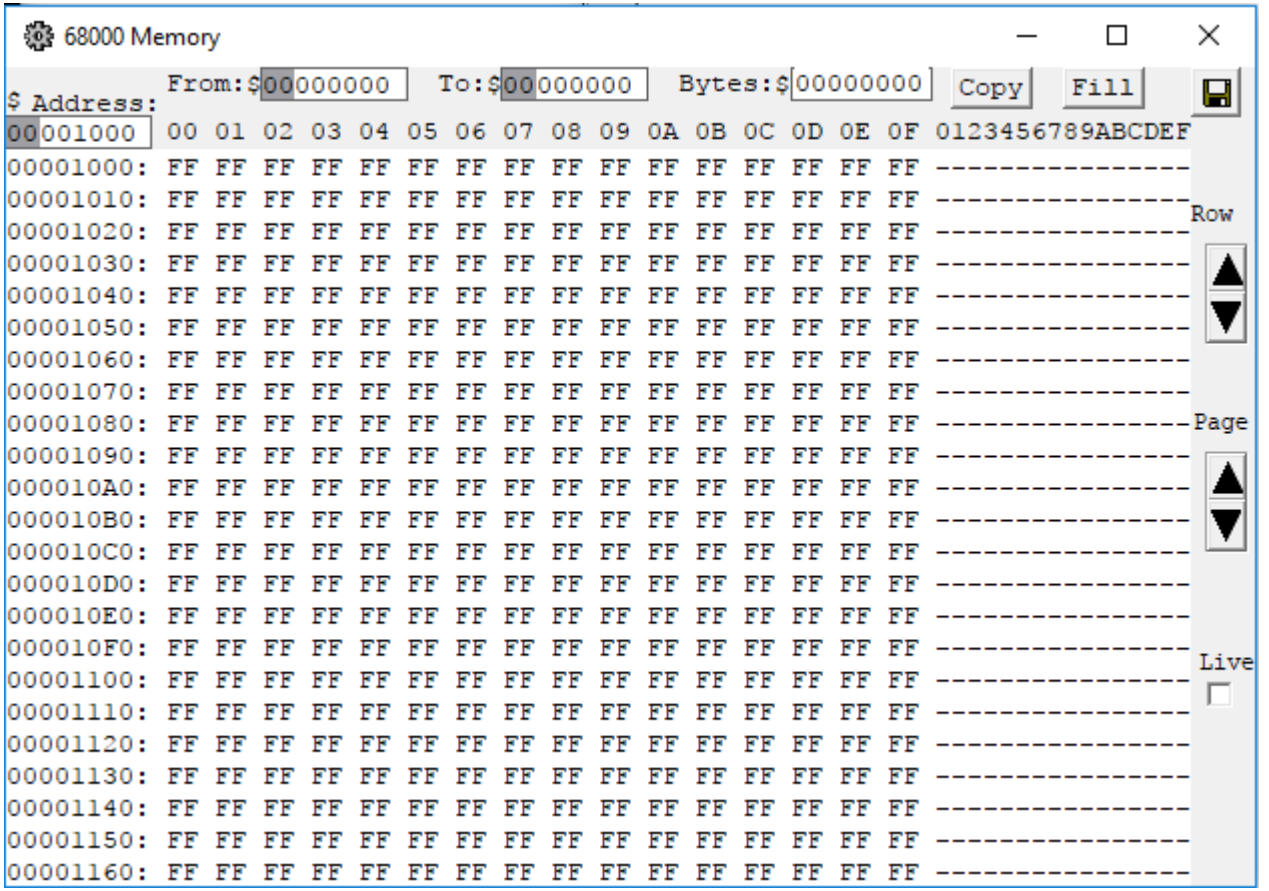
Easy68k simulator

- Editor
- Simulator
  - Registri



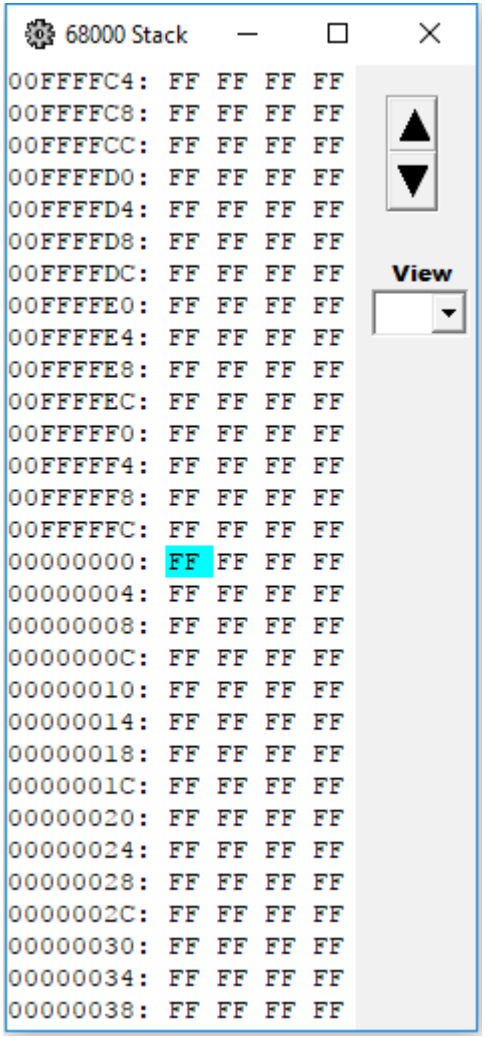
Easy68k simulator

- Editor
- Simulator
  - Registri
  - Memorija



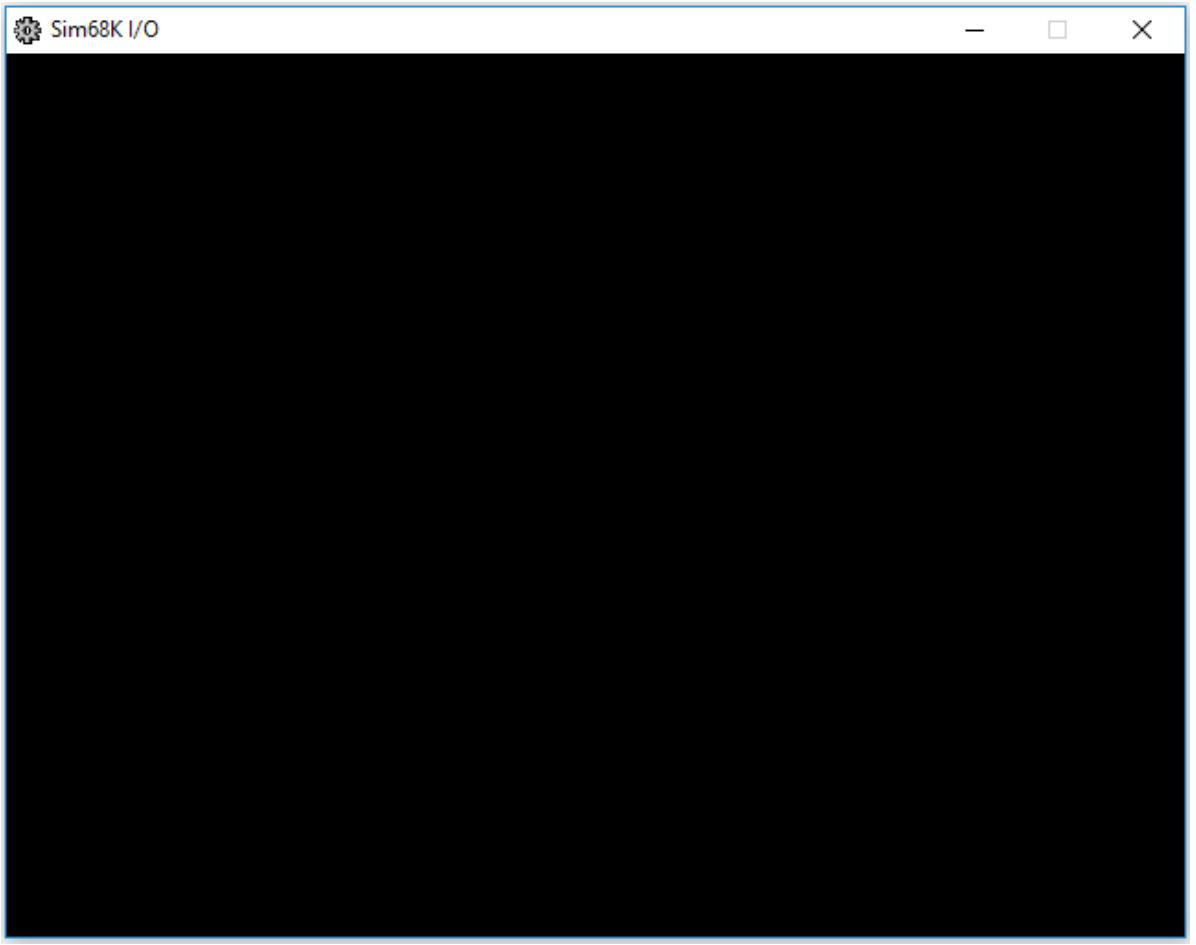
Easy68k simulator

- Editor
- Simulator
  - Registri
  - Memorija
  - Stack



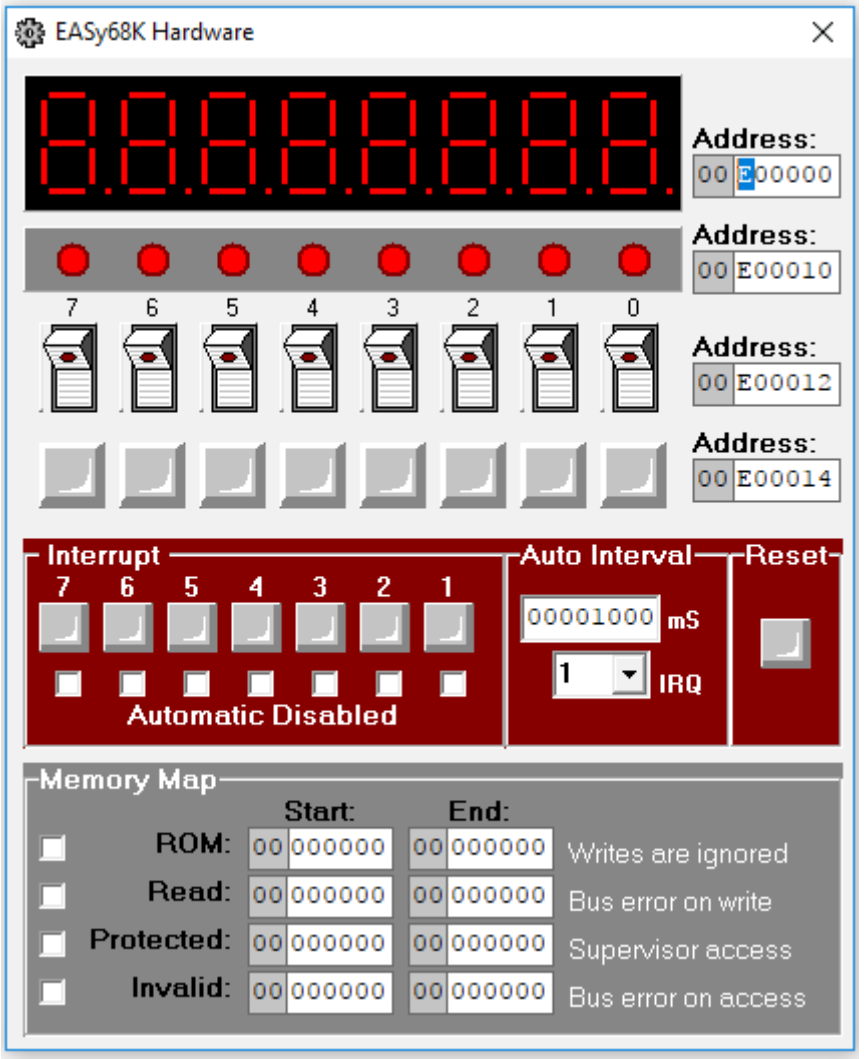
Easy68k simulator

- Editor
- Simulator
  - Registri
  - Memorija
  - Stack
  - Komandna linija



Easy68k simulator

- Editor
- Simulator
  - Registri
  - Memorija
  - Stack
  - Komandna linija
  - Hardver



Direktive, labele i instrukcije

- Direktive
  - Ne generiraju strojni kod (neće se vidjeti u memoriji)
  - Tiču se interpretiranja dijelova koda
  - “izvršavaju” se prilikom prevođenja programa
- Oznake (labele)
  - Nazivi za memorijske lokacije
- Instrukcije
  - Eksplicitno pišu u memoriji
  - Izvršavaju se tokom pokretanja programa
  - Mijenjaju stanje procesora, memorije i sabirnica

Direktiva `END`

- `END` – direktiva koja određuje gdje je smještena prva instrukcija programa.
- Određuje početni sadržaj programskog brojila (`PC`).
- Obavezan dio svakog programa. Minimalni program sadrži samo `END` direktivu:

```
END     0
```

```
END     $1000
```

- Ovakvi programi će se prevesti ali neće uvijek raditi korektno (zašto?).

Direktiva `ORG`

- `ORG` – direktiva koja definira na koju adresu će se spremiti instrukcija ili podatak koji neposredno slijedi direktivu.

- Ostale instrukcije će biti smještene sljedno u memoriji, jedna iza druge.

- Direktivom `ORG` možemo instrukcije smjestiti na druge lokacije.

Labele

- Labele – nazivi memorijskih lokacija
- Implicitno definiranje
  - natpis na početku retka ili korištenje `DC` i `DS` direktiva (kasnije)
  - adresa ovisi o poziciji u kodu
- Eksplicitno definiranje
  - asemblerska direktiva `EQU`

```
        ORG $1000
START:
        END START
```

```
START   EQU $1000

        END START
```

U oba primjera, labela `START` je sinonim za broj `$1000`. Dvotočka je opcionalna kod definiranja.

Instrukcije

- Zadaju se pomoću mnemonika – kratkih riječi koje ugrubo govore što naredba radi

```
ADD #13, D0
```

- Svaka ima svoj jedinstveni binarni kod

```
0640 000D
```

- Kod je smješten u memoriji

```
00002FF0: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00003000: 06 40 00 0D FF FF FF FF FF FF FF FF FF FF FF FF
00003010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

- Pazite: možete “pogaziti” instrukcije programa!

Instrukcije

- Većina ih ima tri varijante, ovisno o sufiksu
  - 8(`.B`), 16(`.W`) ili 32(`.L`) bita

  Dozvoljeni parametri tj. vrste *adresiranja*:
  - Implicitno – adresa operanda je sadržana u instrukciji (`BRA` koristi `PC`, `RTS` koristi pokazivač na stog)
  - Direktno – adresa operanda daje se kao parametar
  - Indirektno – adresa operanda računa se na temelju danih parametara
- Pojedine instrukcije dozvoljavaju samo neke varijante adresiranja
- Detalji SVIH instrukcija i mogućnosti simulatora:
  - Help->Index

TRAP #15 / SIMHALT

- Posebne naredbe čije djelovanje je definirano simulatorom (*specifične za Easy68K*)
- Služe za zaustavljanje simulatora (`TRAP` može puno više)
- `SIMHALT`
  - Nije dio instrukcijskog seta motorole
  - Instrukcija se sastoji od svih jedinica: `$FFFFFFFF`
- `TRAP`
  - Različite funkcije ovisno o stanju registara i danom parametru
  - Da bi se zaustavio simulator u registru `D0` treba pisati dekadsko `9`
  - Puno više o `TRAP` instrukciji kasnije

Naredba `MOVE`

- Služi za kopiranje podataka
  - Dozvoljava većinu varijanti direktnog i indirektnog adresiranja
- Veličina podataka
  - Memorija i podatkovni registri: 8(`.B`), 16(`.W`) ili 32(`.L`) bita
    - Parne adrese za `W` i `L`
  - Adresni registri: 16 ili 32 bita
    - *Sign extension*

Direktno adresiranje

- Podatkovni registri

```
                    MOVE.L D0,D3
D0: 10204FFF                        D0: 10204FFF
D3: 1034F88A                        D3: 10204FFF
```

- Bitno je koju veličinu mičemo
  - `.B` će utjecati samo na najdonji bajt
  - `.W` će utjecati na donju riječ
  - `.L` će promijeniti sadržaj cijelog registra

Direktno adresiranje

- Adresni registri

```
                MOVEA.L A0,A3
A0: 00200000                    A0: 00200000
A3: 0004F88A                    A3: 00200000
```

- Dozvoljene veličine su .W i .L
  - Neovisno o veličini, promijenjen je cijeli registar
  - Sign extension
- Možete koristiti `MOVE` umjesto `MOVEA` (ista je instrukcija!)

Direktno adresiranje

- Trenutno adresiranje

```
MOVE.L #$1FFFF, D0
```

D0: 012309FF D0: 0001FFFF

- Koristimo znak ‘#’ da bismo naglasili da se radi o broju (inače se radi o adresi)
- Znak ‘$’ označava heksadecimalni broj, ‘%’ binarni
- Obratite pozornost na vodeće nule!

Direktno adresiranje

- Apsolutno adresiranje – zadana je adresa operanda

```
            MOVE.L #$10030, $1200
00001200 00                  00001200 00
00001201 00                  00001201 01
00001202 00                  00001202 00
00001203 00                  00001203 30
```

- *Big endian* – najznačajniji bajt je na najmanjoj adresi

Indirektno adresiranje

- Indirektno adresiranje (IA) dijelimo na:
  1) IA pomoću adresnog registra (IAPAR)
  2) IA pomoću adresnog registra uz preddekrementiranje
  3) IA pomoću adresnog registra uz postinkrementiranje
  4) IA pomoću adresnog registra uz odmak
  5) IA pomoću adresnog registra uz indeks i odmak
  6) IA pomoću programskog brojila (IAPPB) uz odmak
  7) IA pomoću programskog brojila uz indeks i odmak

IA pomoću adresnog registra (IAPAR)

- Adresni registar sadrži adresu operanda (pointeri!)
- Dereferenciranje – zagrade oko registra

```
MOVE.L D0, (A0)
```

```
D0: 10434567
A0: 00001200
```

```
00001200    10
00001201    43
00001202    45
00001203    67
```

IAPAR uz preddekrementiranje

- U ovom se tipu adresiranja prvo dekrementira adresni registar te se potom koristi operand s nove adrese
- U assemblyju to označavamo s `-(An)`
- Analogno u `C`-u

```
int a[] = {1,2,3,4,5,6};
int *b = a+4;
int c = *(--b);
printf("Operand je %d", c);
```

IAPAR uz postinkrementiranje

- U ovom se tipu adresiranja prvo koristi operand na kojeg pokazuje adresni registar te se potom inkrementira vrijednost u adresnom registru
- U assemblyju to označavamo s `(An)+`
- Analogno u `C`-u

```
int a[] = {1,2,3,4,5,6};
int *b = a+4;
int c = *(b++);
printf("Operand je %d", c);
```

IAPAR uz odmak

- U ovom tipu adresiranja efektivnu adresu operanda dobivamo kao zbroj neke 16 bitne vrijednosti x (odmak) i vrijednosti koja je sadržana u adresnom registru
- Za x vrijedi *sign extension*
- U assemblyju to označavamo s `x(An)`

```
int a[] = {1,2,3,4,5,6};
short x = 2; // moze biti negativan
int *b = a+2;
int c = *(b+x);
printf("Operand je %d", c);
```

IAPAR uz indeks i odmak

- U ovom tipu adresiranja efektivna aresa operanda dobivena je zbrajanjem adrese u danom adresnom registru, vrijednosti (indeksa) u nekom drugom (indeksnom) registru `IR` te odmaka x.
- Indeksni registar može biti bilo koji od podatkovnih i adresnih registara.
- Indeks i odmak mogu biti 16 i 32 bitne vrijednosti. U slučaju 16 bitnih vrijednosti koristi se *sign extension*
- U assemblyju to označavamo s `x(An,IR)`

Naredba `MOVE`

- **Primjeri**: `gr-v2_01.x68`, `gr-v2_02.x68`

  Zadaci i pitanja:

  1) Napišite program od dvije instrukcije, tako da prvom instrukcijom "uništite" drugu.

  2) Napišite program koji tokom izvršavanja napiše kod neke valjane instrukcije u memoriju tako da se instrukcija pokrene tokom istog izvršavanja.

  3) Hoće li se naredba `MOVE.W $1001,D0` prevesti? Hoće li se izvršiti?

  4) Hoće li se naredba `MOVE.B #$123456,D0` prevesti? Hoće li se izvršiti? Što ako maknemo znak '#'?

Naredba MOVE

5) Što će pisati u registru D0 nakon pokretanja sljedećeg programa:

```
        ORG $1000
START:
        MOVE.L #$123456, D0

        MOVE.W #$DDAA, D0

        MOVE.B #64, D0

        SIMHALT
        END START
```

Naredba MOVE

6) Što će pisati u registru A0 nakon pokretanja sljedećeg programa:

```
        ORG $1000
START:
        MOVE.L #$12345678,A0

        MOVE.W #$8000, A0

        MOVE.W #$0, A0

        SIMHALT
        END START
```

Naredba MOVE

7) Što će pisati u uvjetnom registru nakon pokretanja sljedećeg programa:

```
        ORG $1000
START:  MOVE.L #$0, D0

        MOVEA.W #$11, A0

        MOVE.W #$8022, D0

        MOVE.W #$22, D0

        SIMHALT
        END START
```

Dodatno

8) Pomoću `ORG` direktive napravite program koji sadrži “mrtvi kod”, tj. instrukcije koje se nikad neće izvršiti