Građa računala

Funkcije

Stog na M68k

- Stog – dio memorije na koji kaže “stogovni pokazivač” (stack pointer)
- Dva stoga, na različitim adresama
  - Korisnički
  - Sistemski
- Dva stack pointera
  - Registar A7 (po jedan A7 registar za sistem i korisnika)
  - Koji se koristi ovisi o supervisor bitu

Stog na M68k

- Stog raste prema manjim memorijama
- A7 uvijek kaže na zadnji element na stogu
- Brisanje elemenata u stogu
  - Samo pomičemo pokazivač na višu adresu
  - Nije potrebno eksplicitno mijenjati memoriju
- Brojne instrukcije koriste stog
  - `JSR`, `RTS` – za poziv podrutina
- Na stog možemo i sami stavljati podatke

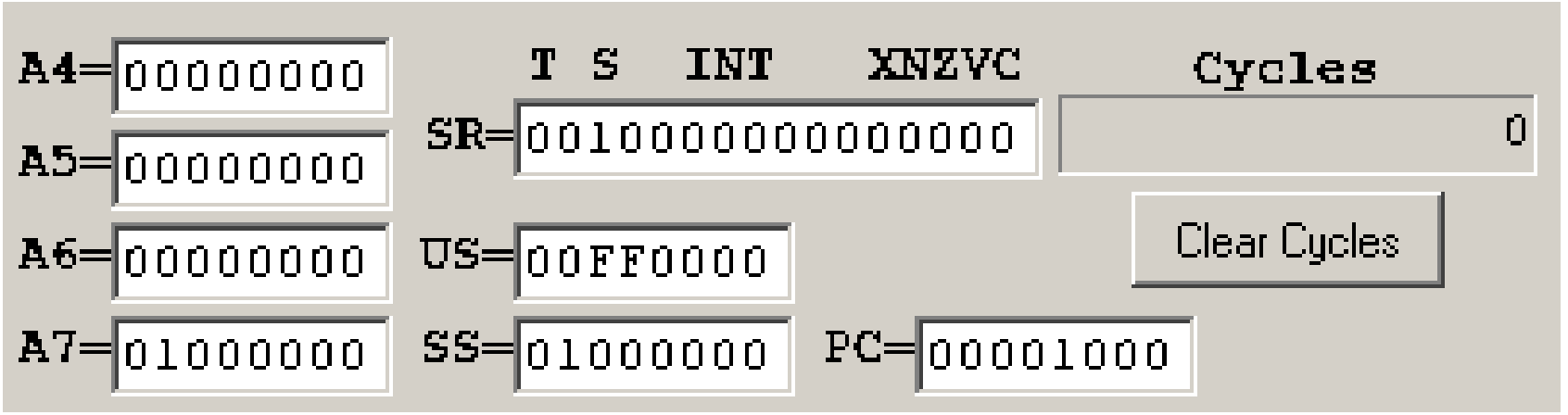
```
MOVE.B PODACI, -(A7)
```

Instrukcije `JSR` i `RTS`

- Grananje
  - Skok na neki label
  - Uvjetno izvršavanje nekog koda
- Instrukcija `JSR` (Jump to SubRoutine)
  - Bezuvjetni skok na dani label
  - Bezuvjetno izvršavanje nekog koda
  - Organizacija koda u blokove
  - Korištenje stoga za lokalne varijable (memorija koja pripada samo funkciji i neće biti “pokvarena” izvana)

Instrukcije JSR i RTS

- Na stog se stavlja adresa instrukcije NAKON JSR

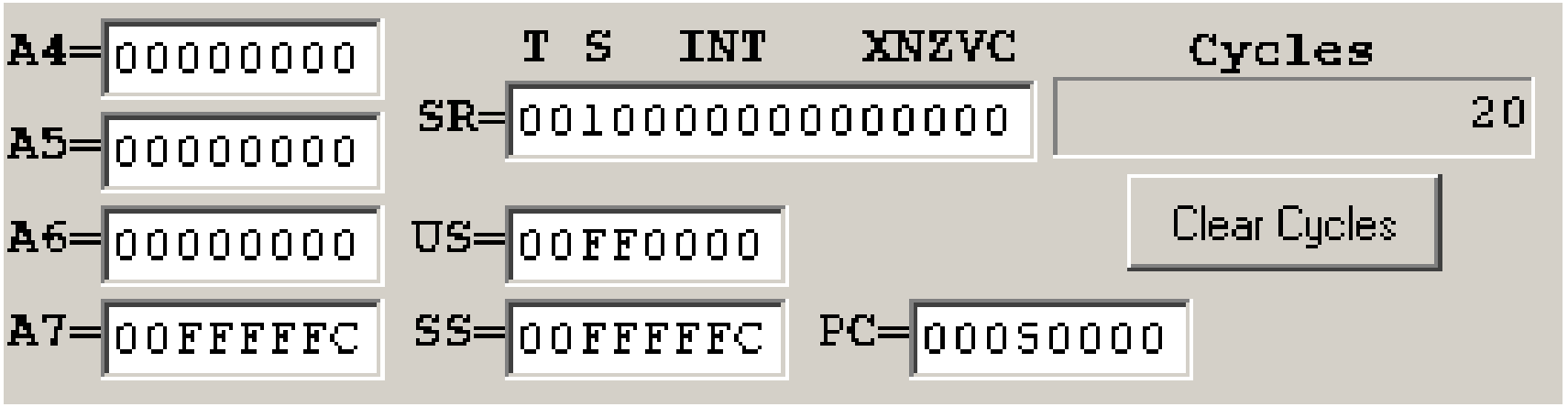



```
    ORG $50000
FJA: MOVE.L #0, D0
    RTS

    ORG    $1000
START:
    JSR FJA
    SIMHALT
    END    START
```

```
00FFFFE4: FF FF FF FF
00FFFFE8: FF FF FF FF
00FFFFEC: FF FF FF FF
00FFFFF0: FF FF FF FF
00FFFFF4: FF FF FF FF
00FFFFF8: FF FF FF FF
00FFFFFC: FF FF FF FF
00000000: FF FF FF FF
```

Instrukcije `JSR` i `RTS`

- Na stog se stavlja adresa instrukcije NAKON JSR



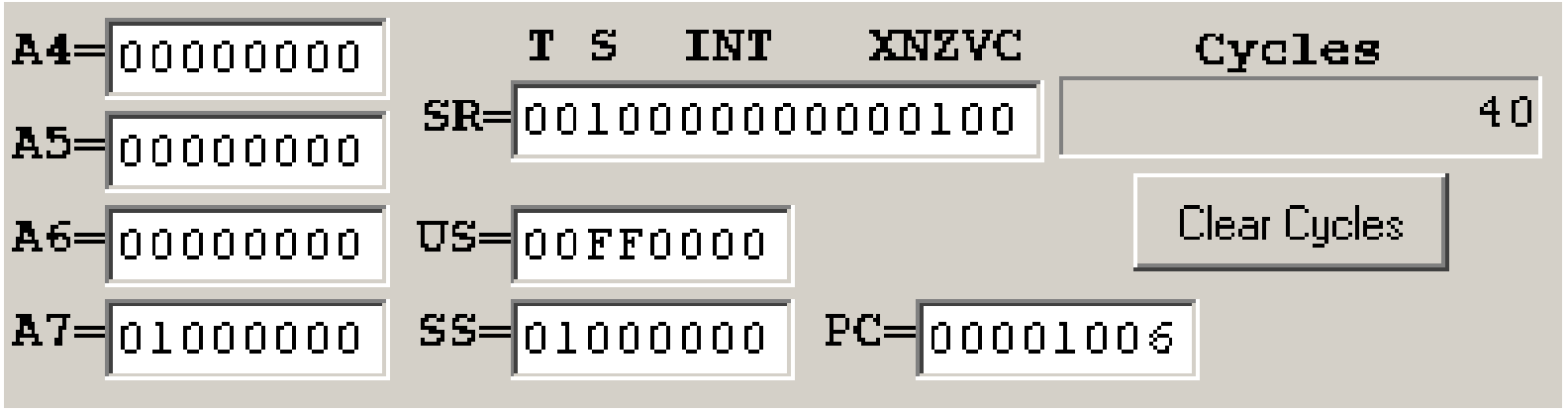

```
        ORG $50000
FJA:  MOVE.L #0, D0
        RTS

        ORG     $1000
START:
        JSR FJA
        SIMHALT
        END     START
```

```
00FFFFE4: FF FF FF FF
00FFFFE8: FF FF FF FF
00FFFFEC: FF FF FF FF
00FFFFF0: FF FF FF FF
00FFFFF4: FF FF FF FF
00FFFFF8: FF FF FF FF
00FFFFFC: 00 00 10 06
00000000: FF FF FF FF
```

Instrukcija `RTS`

- Nakon što funkcija završi, još treba
  - Vratiti se na instrukciju nakon poziva
  - Počistiti stog
- `RTS` (ReTurn from Subroutine)
  - U PC stavlja vrh stoga
  - Pomiče (čisti) stog
  - Pažnja – RTS ne zna i ne provjerava što je na vrhu stoga!

Instrukcija `RTS`




```
     ORG $50000
FJA: MOVE.L #0, D0
     RTS

     ORG     $1000
START:
     JSR FJA
     SIMHALT
     END     START
```

```
00FFFFE4: FF FF FF FF
00FFFFE8: FF FF FF FF
00FFFFEC: FF FF FF FF
00FFFFF0: FF FF FF FF
00FFFFF4: FF FF FF FF
00FFFFF8: FF FF FF FF
00FFFFFC: 00 00 10 06
00000000: FF FF FF FF
```

Poziv podrutina

- Funkcije u C-u parametre primaju preko stoga
- *Calling convention*
  - Dogovor oko načina prenošenja parametara u funkcije
  - Ovisi o procesoru/assemblyu/dogovoru
- Obradit ćemo tri načina prijenosa podataka u podrutinu:
  - Preko registara
  - Preko memorije
  - Preko stoga
- Sva tri načina su dobra, odabir ovisi o potrebi/dogovoru

Prijenos preko registara

- *Kontekst* – stanje registara u nekom trenu
- Neki registri se unaprijed odrede kao registri za prijenos parametara
  - Kontekst se sprema
  - Možemo spremiti sve ili samo dijelove koje koristimo
  - Poanta – nakon izvršavanja stanje registara koji se NE koriste za prijenos treba biti jednak kao prije poziva
  - Brzo, ali nefleksibilno
- Povrat parametara
  - Registri
  - Unaprijed određene adrese

Prijenos preko memorije

- Za svaki poziv funkcije, pripremimo blok memorije
- Obično slijedi odmah nakon poziva

```
; po potrebi napunimo listu parametara
MOVE.L   #1, BROJ1
MOVE.L   #2, BROJ2

; poziv
JSR  ZBROJI
DC.L BROJ1
DC.L BROJ2
DC.L REZULTAT
```

- Povratni parametri
  - Memorija (neka unaprijed određena adresa) ili registri

Prijenos preko memorije

- Dohvaćanje parametara
  - Znamo da je prilikom ulaska u funkciju na stogu spremljen PC koji kaže na adresu nakon `JSR`
  - Ta adresa je adresa prvog parametra funkcije
- Prije povratka u pozivajući program
  - Potrebno je korigirati ranije spremljenu adresu (PC) tako da kaže na sljedeću instrukciju (preskačemo parametre)
- Kod ovog načina poziva, registri trebaju biti nepromijenjeni nakon povratka iz rutine

Prijenos preko stoga

- Parametre stavljamo na stog prije poziva funkcije
- Povratni parametar će biti u nekom *registru* (scratch registri, obično D0 i A0) ili na *stogu*
- Više načina implementiranja
  - Ovisi o tome tko je zadužen za “čišćenje” stoga
- Čišćenje stoga
  - Pomicanje vrha na mjesto gdje je bio prije poziva
  - Ako pozivatelj čisti – treba maknuti parametre (i eventualni rezultat) sa stoga
  - Ako funkcija čisti – nakon izlaska stog izgleda kao prije poziva, rezultat mora biti spremljen *negdje*