

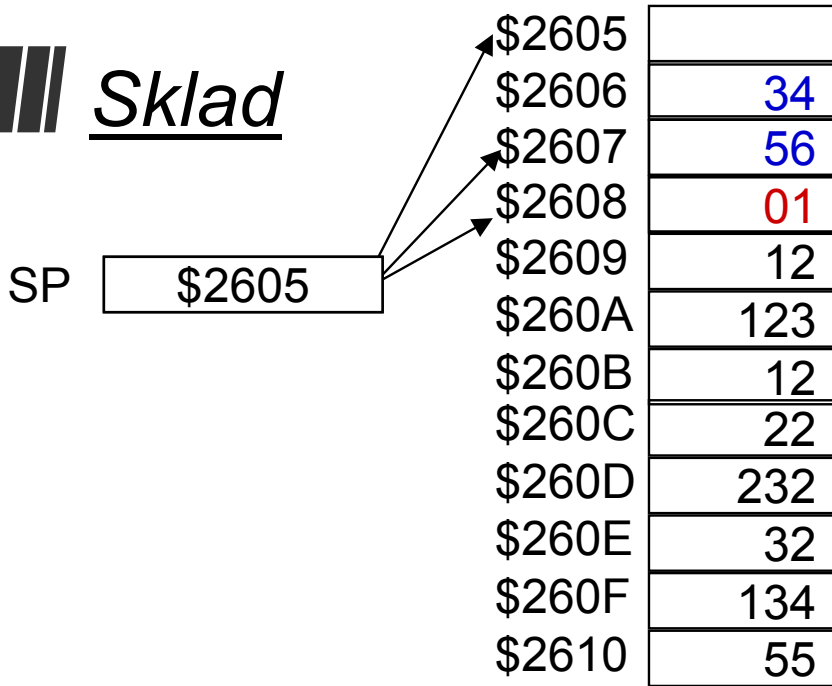


Podatkovne Strukture

- **Programski jeziki visokega nivoja omogočajo uporabo uporabnih podatkovnih struktur**
 - Skladi, nizi, vektorji, vrste, drevesa, sezname...
- **Uporaba the struktur je možna tudi v zbirniku, vendar mora za podrobnosti poskrbeti programer**
 - Celo dostop do preprostih nizov in vektorjev zahteva več ukazov
 - Pomagamo si tako, da napišemo knjižnice za delo s podatkovnimi strukturami



Sklad



Dodajanje (*push*) elementa na sklad:

- Piši na vrh sklada
- Zmanjšaj SP za *velikost elementa*

```

LDAA #1
PSHA      ;Pom[SP] <- A, SP <- SP-1
*         ;SP == $2607

LDX  #$3456
PSHX      ;Pom[SP-1,SP] <- X, SP <- SP-2
*         ;SP == $2605

```



Sklad

	\$2605	
	\$2606	34
	\$2607	56
	\$2608	01
	\$2609	12
	\$260A	123
	\$260B	12
	\$260C	22
	\$260D	232
	\$260E	32
	\$260F	134
	\$2610	55

} novi Y
 novi B

Višji pomnilniški naslovi ↓

SP \$2608

Jemanje (*pull, pop*) elementa s sklada:
 •Prepiši element z vrha sklada *plus 1*
 •Zmanjšaj SP za *velikost elementa*

```

PULY      ;Y <- Pom[SP+1,SP+2], SP <- SP+2
*         ;SP == $2607, Y == $3456

PULB      ;B <- Pom[SP+1], SP <- SP+1
*         ;SP == $2608, B == $01
  
```

Podprogrami

- **Podprogrami (procedure) so deli programske kode, ki jih kličemo iz ostalih delov programa**
 - Če večkrat uporabimo isto zaporedje ukazov
 - Program, razdeljen na podprograme je lažje razumljiv.
- **Splošne podprograme najdemo v knjižnicah**
 - Knjižnico lahko napiše kdo drug!
 - Preprosto lahko prekopiramo podprograme, ki jih potrebujemo.
 - Knjižnice podprogramov k programu običajno doda povezovalnik (*linker*) - izvorne kode nam ni potrebno poznati...
- **Ukazi za klic podprogramov so podobni vejitvenim ukazom**
 - [`<oznaka>`] BSR `<rel>`
 [`<oznaka>`] JSR `<opr>`
 - Oba ukaza predstavljata skok na začetek podprograma
 - Pred tem na vrh sklada shranita *povratni naslov*

Klici podprogramov

- Ukazi za klic podprogramov so podobni vejitvenim ukazom

- [*<oznaka>*] BSR *<rel>*
- [*<oznaka>*] JSR *<opr>*

Naslov naslednjega ukaza.

- Oba ukaza predstavljata skok na začetek podprograma
- Pred tem na vrh sklada shranita *povratni naslov*

\$2607	24
\$2608	2F
\$2609	12
\$260A	123
\$260B	12
\$260C	22

→	242d 8d 12	bsr	test
→	242f c6 0e	ldab	#14
		...	
→	2441 86 0c	test	ldaa #12
→	2443 39	rts	

SP \$2608

PC \$242F

RTS - return from subroutine

S sklada vzame povratni naslov in skoči tja

Delo s podprogrami

- **Težave z akumulatorji/registri**
 - Podprogram običajno spremeni vrednosti registrov
- **Prenos parametrov**
 - Kako v podprogram prenesemo parametre?
- **Vračanje rezultatov**
 - Kako prenesti vrednost iz podprograma v glavni program?
- **Lokalne spremenljivke**
 - Spremenljivke, ki so vidne le v podprogramu?

Zaključek: Vse to zahteva začasno uporabo pomnilnika

Shranjevanje akumulatorjev/registrov

- **V podprogramih je potrebna uporaba akumulatorjev in drugih registrov**
 - Ne vemo, katere registre uporablja glavni program...
 - Če spremenimo vrednosti teh registrov, je delovanje napačno
- **Posredovati mora programer**
 - Shraniti registre, preden jih uporabimo v podprogramu
 - Na koncu podprograma v registre vrniti shranjene vrednosti

Shrani klicoči:

Glavni program: Shrani vrednosti vseh registrov **v uporabi** na sklad

<klic podprograma - vračanje>

Glavni program: Registrom vrne prejšnje vrednosti - vrednosti prebere s sklada

Shrani-klicani:

Glavni program: <klic podprogr.>

Podprogram: Shrani vrednosti vseh registrov, ki jih **bo uporabil** na sklad.

<telo podprograma>

Podprogram: Registrom vrne prejšnje vrednosti (prebere s sklada)

Uporaba sklada – prenos parametrov

Predpostavimo, da želimo napisati podprogram, ki ga kličemo iz drugih delov programa. Ob vstopu v podprogram so spremenljivke L (1 bajt), W (1 bajt) in H (2 bajta) shranjene na skladu...

```
L  FCB  $30
W  FCB  $26
H  FDB  $128A
LDS  #$3FFF ;začetna nastavitev SP
*
*          ;(ponavadi prvi ukaz
*          ;po vključitvi napajanja)
LDAA L    ← ;Odloži L na sklad
PSHA     ← ; po operaciji, SP==$3FFE
LDAA W    ← ;Odloži W na sklad
PSHA     ← ; po operaciji, SP==$3FFD
LDX H    ← ;Odloži H na sklad
PSHX     ← ; po operaciji, SP==$3FFB
JSR  PODPR ;Klic podprograma
        ← ; po operaciji, SP=$3FF9
```

SP	\$3FF9
\$3FF9	?
\$3FFA	PC _H
\$3FFB	PC _L
\$3FFC	12
\$3FFD	8A
\$3FFE	26
\$3FFF	30

} H
W
L

Uporaba sklada (boljši način)

Podprogram računa $L \cdot W + H$, torej moramo dostopati do teh spremenljivk. Rezultat želimo shraniti na sklad na mesto H, na skladu ne želimo spremeniti nič drugega.

