F4: Assemblerprogrammering

- Hoppinstruktioner
  - Branch
    - Med vilkor
  - Jump
- IF satser
- Loopar
  - while-loopar
  - do-while- loopar
  - for-loopar
- Stackhantering
- Underprogram

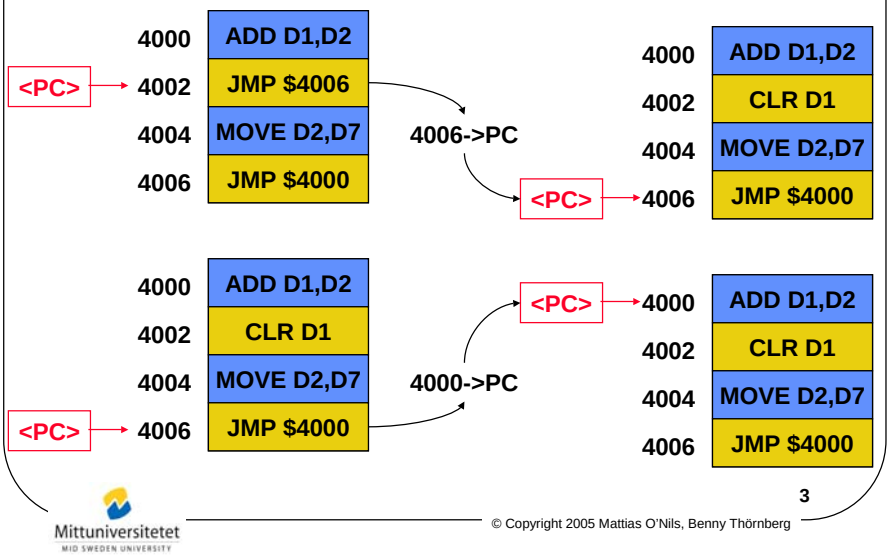
Mittuniversitetet
MID SWEDEN UNIVERSITY

Hoppinstruktioner

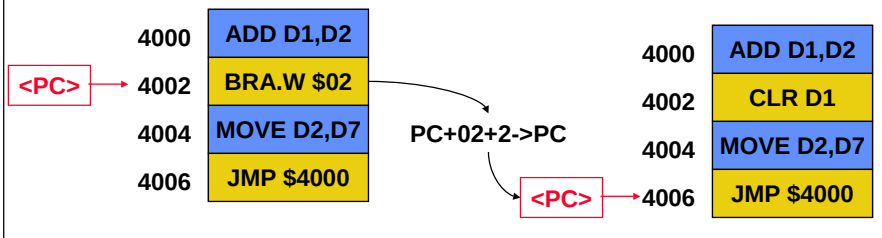
- BRA rel_adr
  - Branch Always
  - Relativadressering
    - rel_adr + <PC> -> <PC>
- Bcc rel_adr
  - Branch on condition
  - Relativadressering
    - rel_adr + <PC> -> <PC>
- JMP abs_adr
  - Jump
  - Absolutadressering
    - abs_adr -> <PC>

Mittuniversitetet
MID SWEDEN UNIVERSITY

Absolutadress



Relativadress



BRA finns med 8-bits eller 16 bits offset
dvs **BRA.B** eller **BRA.W**

Relativa hopp ger kod som enkelt kan flyttas till annan plats i minnet

Relokerbar kod

Compare-instruktionen

CMP.B <ea>, Dn
W
L

Subtrahera källoperanden från destinationsoperanden. Resultatet av subtraktionen sparas **EJ** till destinationen. Alla flaggor i CCR utom Extend-flaggan sätts enligt resultatet för subtraktionen.

Compareinstruktionen används i kombination med villkorliga hopp.

Mittuniversitetet
MID SWEDEN UNIVERSITY

Villkorliga hopp

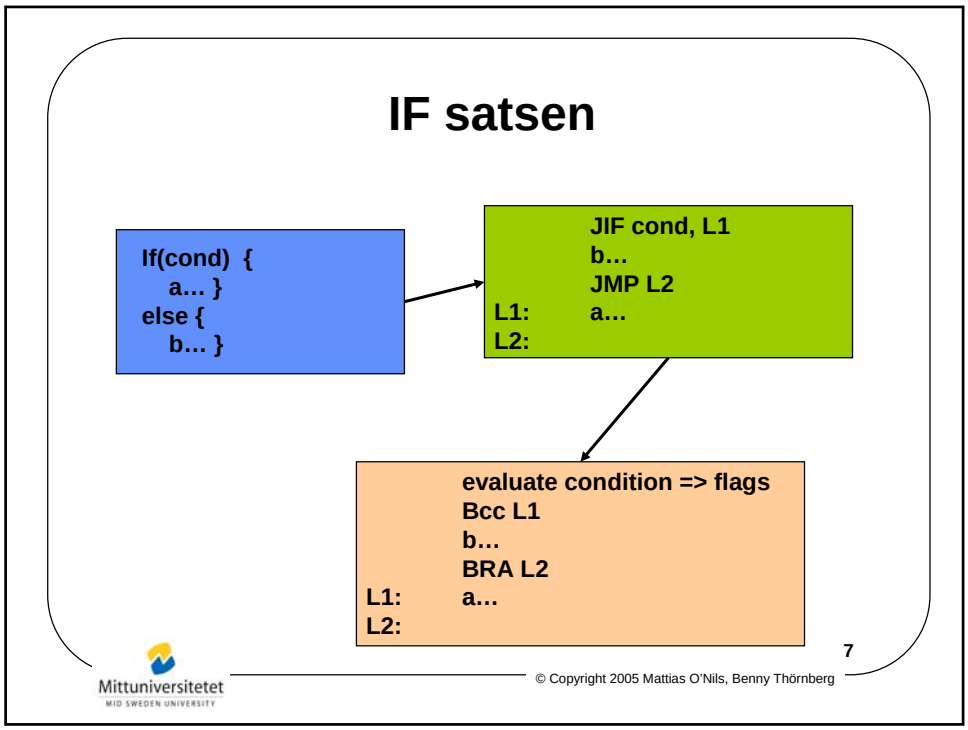
Bcc <label>

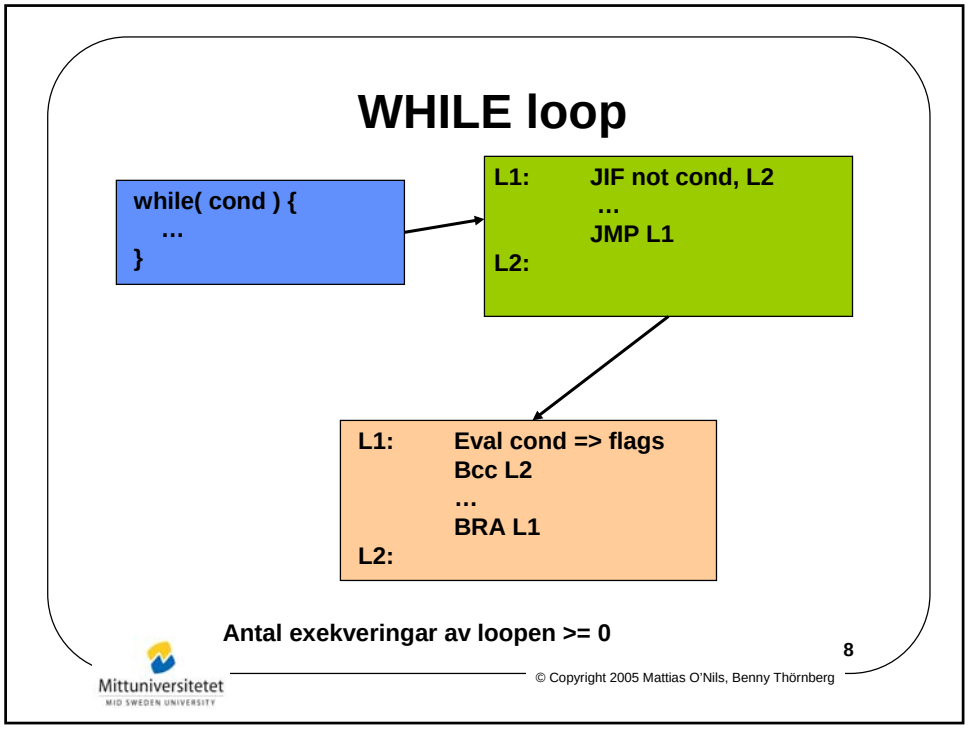
Exempel på villkor:

	Instruktion	Semantik	Logiskt uttryck
	BEQ	branch on equal	Z
	BCC	branch on carry clear	$\overline{C}$
Signed format	BGE	Branch on greater than or equal	$N.V+\overline{N}.\overline{V}$
Unsigned	BHS	Branch on higher than or same	$\overline{C}$

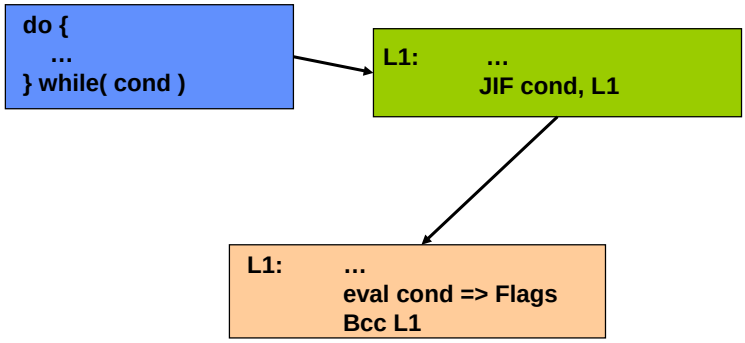
I kombination med compare så syftar semantiken alltid på destinationen. Ex. destinationen är större eller lika med källoperanden

Mittuniversitetet
MID SWEDEN UNIVERSITY


IF satsen
If(cond) {
a… }
else {
b… }
JIF cond, L1
b…
JMP L2
L1: a…
L2:
evaluate condition => flags
Bcc L1
b…
BRA L2
L1: a…
L2:
Mittuniversitetet
MID SWEDEN UNIVERSITY
7
© Copyright 2005 Mattias O'Nils, Benny Thörnberg


WHILE loop
while(cond) {
…
}
L1: JIF not cond, L2
…
JMP L1
L2:
L1: Eval cond => flags
Bcc L2
…
BRA L1
L2:
Antal exekveringar av loopen >= 0
Mittuniversitetet
MID SWEDEN UNIVERSITY
8
© Copyright 2005 Mattias O'Nils, Benny Thörnberg

While-do



Antal exekveringar av loopen >= 1

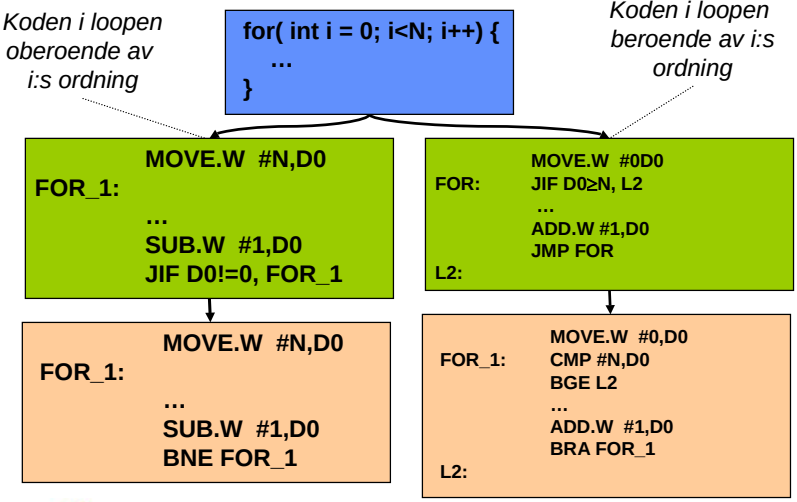
Mittuniversitetet
MID SWEDEN UNIVERSITY

FOR loop

Ersätt i med ex. D0

Koden i loopen oberoende av i:s ordning

Koden i loopen beroende av i:s ordning

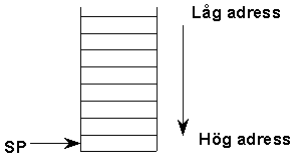


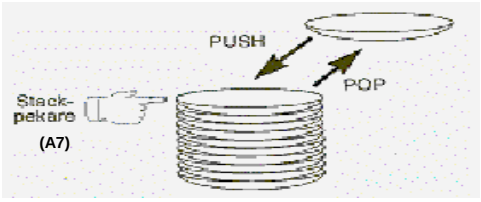
Antal exekveringar av loopen = N

Mittuniversitetet
MID SWEDEN UNIVERSITY

Minnesstack, 1(3)

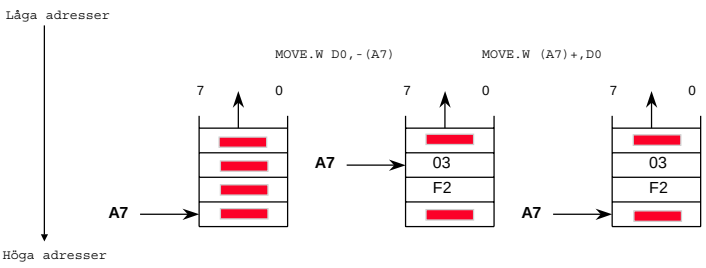
- **Dynamiskt temporärt minne**
  - **Växer mot lägre adresser**
  - **SP är adressregister 7 (A7)**





Stack, 2(3)

- **Lägg till data på stacken (PUSH)**
  - **Ex.** `MOVE D0,-(A7)`
- **Hämta tillbaka data från stack (POP)**
  - **Ex.** `MOVE (A7)+,D0`

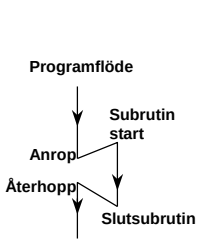


Stack, 3(3)

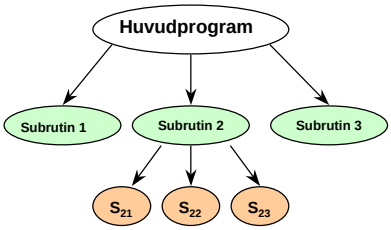
- Stacken kan benämnas på flera olika sätt
- Specifik benämning
  - USP = User Stack Pointer
    - Stackpekare som används i användarmod
  - SSP = Supervisor Stack Pointer
    - Stackpekare som används i systemmod
- Generell
  - A7 ⇔ SP
    - Systemmod ⇒ A7/SP = SSP
    - Användarmod ⇒ A7/SP = USP

Subrutin?

- Vad är en subrutin ?
  - Ett underprogram som kan anropas i från ett annat program.
  - Dvs. fungerar som en funktion i C/C++, men utan parameteröverföring.
- Återanvändning av kod...

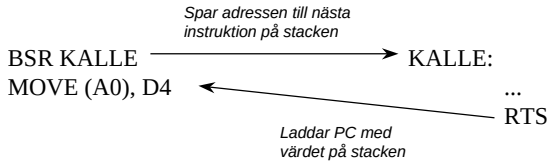


Ger strukturerade program



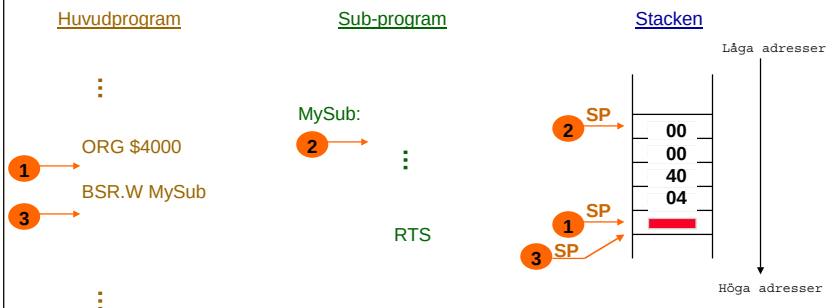
Subrutin, hur?

- **BSR/JSR**
  - **Hoppar till den adress som finns i argumentet och sparar adressen för nästa instruktion på stacken.**
- **RTS**
  - **Hämtar adressen från stacken och hoppar till den adressen.**



Mittuniversitetet
MID SWEDEN UNIVERSITY

Subrutinen och Stacken – Ett exempel



BSR.W

PC → -(SP)

PC+d → PC

RTS

(SP)+ → PC

Mittuniversitetet
MID SWEDEN UNIVERSITY