



## Adressierung

- Adressierung von Daten  
bei LOAD/STORE – Befehlen
- Angabe der Zieladresse  
bei Sprungbefehlen

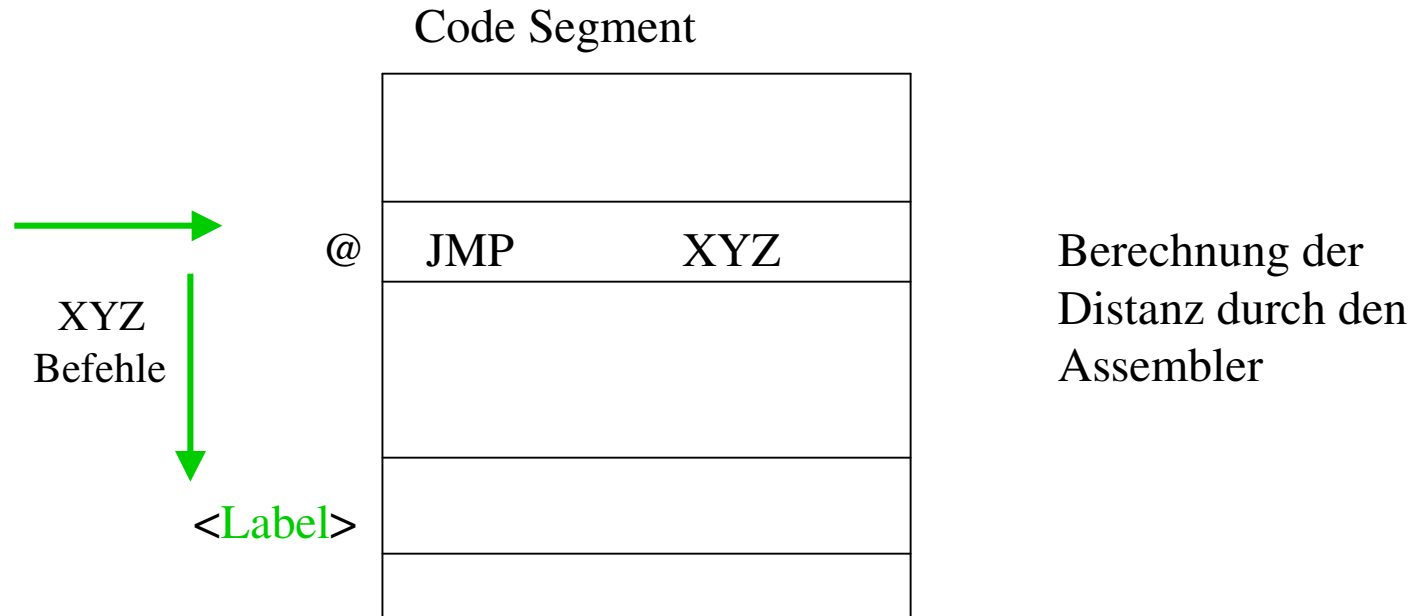






# Adressierung

JMP <Label> unbedingter **relativer** Sprung



$$\text{Zieladresse} = @ \pm \underbrace{\text{XYZ} * 4}$$

Maximale Distanz =  $2^{24} - 1$  Befehle



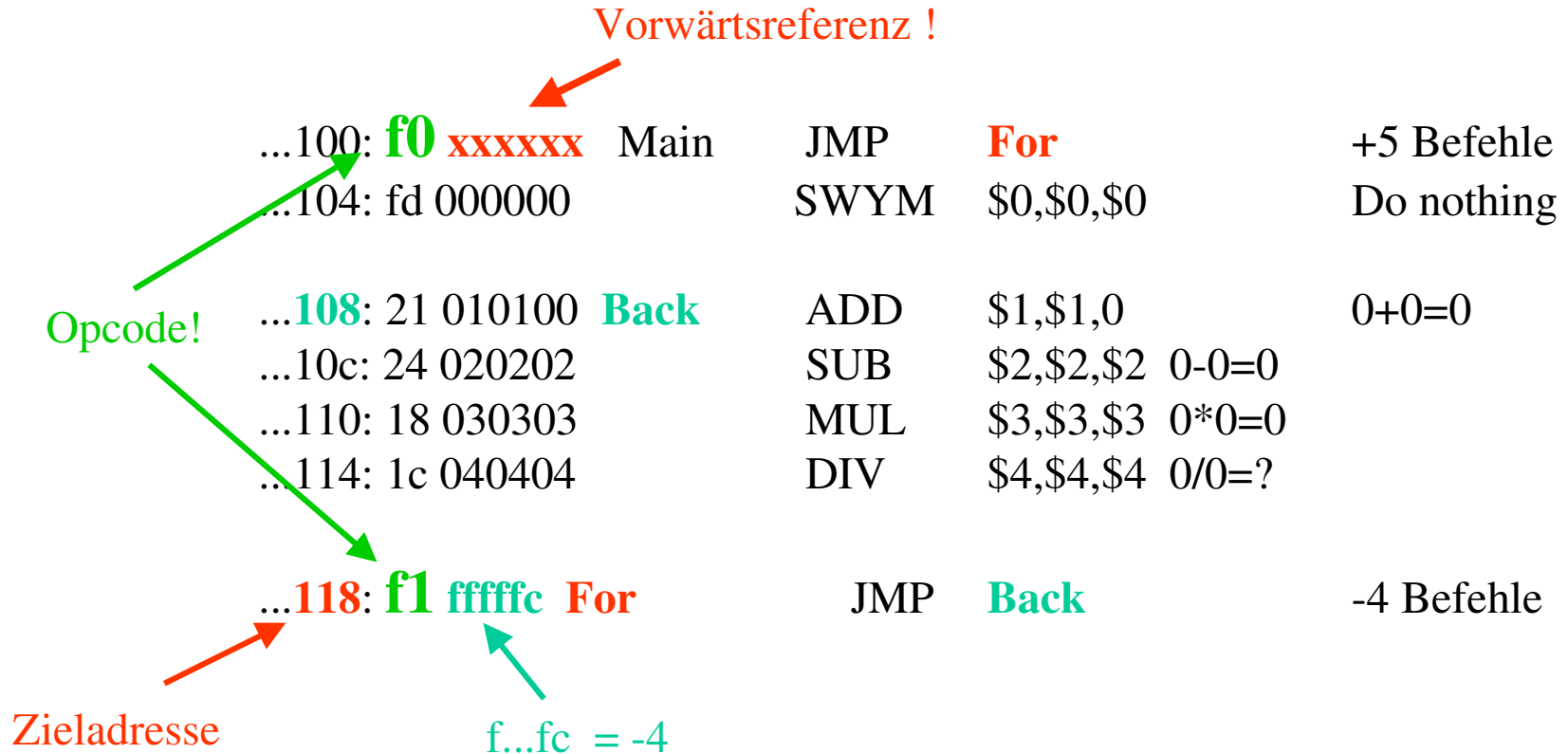
**JMP** <Label> unbedingter **relativer** Sprung

<b>Main</b>	<b>JMP</b>	<b>For</b>	+5 Befehle
	SWYM	\$0,\$0,\$0	Do nothing
<b>Back</b>	ADD	\$1,\$1,0	0+0=0
	SUB	\$2,\$2,\$2	0-0=0
	MUL	\$3,\$3,\$3	0*0=0
	DIV	\$4,\$4,\$4	0/0=?
<b>For</b>	<b>JMP</b>	<b>Back</b>	-4 Befehle



# Adressierung

JMP <Label> unbedingter **relativer** Sprung





## Adressierung

JMP <Label> unbedingter **relativer** Sprung

mmix> 1. 00000000000000**100**: f00000**06** (JMP) -> **#118**

$$\#118 = \#100 + \#(4*6)$$

(now at location #00000000000000118)

mmix> 1. 00000000000000**118**: f1ffffc (JMPB) -> **#108**

(now at location #00000000000000108)

$$\#108 = \#118 - \#(4*4)$$

...



**B\*/BN\*** **\$X**, <Label>      bedingter **relativer** Sprung

Testbedingung in \$X

Distanz in YZ  
maximal  $2^{16} - 1$  Befehle

Zieladresse = @ + YZ \* 4

→ analog zu JMP

* BZ/BNZ	zero
BP/BNP	positive
BN/BNN	negative
BE	even
BO	odd

Unterscheidung  
der Richtung  
durch Opcode, z.B.  
**BZ** : 42 forward  
43 backward



## Unterprogramm-Sprung mit Stack mit **relativer** Adressierung

**PUSHJ** \$X, <Label>

$$\text{Zieladresse} = @ \pm YZ * 4$$

Berechnung der Distanz durch den Assembler

Unterscheidung der Richtung durch Opcode

<b>PUSHJ</b>	F2	forward
<b>PUSHJB</b>	F3	backward

\$X : Stackgrenze



Unterprogramme mit PUSH



## Sprung mit **Registeradresse**

**GO**    \$X,\$Y,Z  
          \$X,\$Y,\$Z

Zieladresse = \$Y + Z (immediate)  
              \$Y + \$Z (indiziert)

\$X = @ + 4 Returnadresse        UP mit GO



## Sprung mit **Registeradresse**

**GO**    \$X,\$Y,Z

Zieladresse = \$Y + Z (immediate)

Z = 0            **registerindirekte** Adressierung  
absolute Adresse des Ziels in \$Y

Z <> 0            fester Offset im Befehl



## Sprung mit Registeradresse

**GO** \$X,<Label>



LOC #100  
GREG @

Main

GO \$0,Label  
SWYM  
SWYM  
SWYM

Label



...  
GO \$2,Label  
SWYM  
TRAP 0,Halt,0

Keine Vorwärtsreferenzen!



GETA \$Y,<Label>  
GO \$X,\$Y,0

Auflösung von  
Rückwärtsreferenz  
durch den Assembler



Sprung mit **Registeradresse**      **GO**    **\$X,<Label>**

```
LOC    #100
($254=#00000000    GREG @
00000100)
***** Main GO $0,Label
***** error: YZ field is undefined!
...100: fd000000    SWYM
...104: fd000000    SWYM
...108: fd000000 Label SWYM
...10c: 9f 02 fe 08    GO $2,Label
...110: fd000000    SWYM
...114: 00000000    TRAP 0,Halt,0
```



Symbol table:  
**Label = #0000000000000108 (2)**  
Main = #0000000000000100 (1)



Sprung mit **Registeradresse**

**GO**    \$X,\$Y,\$Z

$$\text{Zieladresse} = \$Y + \$Z$$



Basis

Offset

**Indizierte** Adressierung



“Sprungteppich“ mit GO  
(Switch)



Unterprogramm-Sprung mit Stack mit **Registeradresse**

**PUSHGO** \$X,\$Y,\$Z

Zieladresse = \$Y + Z

\$Y + \$Z



Basis

Offset

Bestimmung der Zieladresse wie bei GO

\$X : Stackgrenze



Unterprogramme mit PUSH



## Adressierung bei Sprüngen

Angabe der Zieladresse



Relativ zur Adresse @ des Sprungbefehls  
mit Distanz als immediate Operand in

- XYZ      JMP
- YZ      B\*,BN\*,PUSHJ



Indiziert mit Basis in \$Y und Offset

- als immediate Operand in Z      GO, PUSHGO
- variabel in \$Z      GO, PUSHGO