

**Aktivera Kursens mål:**

LV5 Fo12

- ▶ Konstruera en dator mha grindar och programmera denna

**Aktivera Förra veckans mål:**

- ▶ Konstruera styrenheten.... genom att....
- ▶ implementera olika maskininstruktioner i styrenheten.
- ▶ Kunna använda instruktionslistan och skriva mycket enkla assemblerprogram
- ▶ Studera olika instruktionstyper och adresseringsmoder
- ▶ Använda utvecklingsmiljön för FLISP

**Veckans mål:**

- ▶ Konstruera styrenheten.... genom att....
- ▶ .... implementera olika maskininstruktioner i styrenheten.
- ▶ Villkorliga hopp
- ▶ Subrutiner och stack
- ▶ Skriva enkla program för FLISP
- ▶ Strukturerad assemblerprogrammering

**Läs smart!  
Lär dig mer!**

Grundläggande Datorteknik LV5

1

**Programexempel för FLEX**

Addera de 16-bitars talen P och Q.

P hittas på minnesadress  $20_{16}$  (och  $21_{16}$ .)

Q hittas på minnesadress  $22_{16}$  (och  $23_{16}$ .)

Placera resultatet på adress  $24_{16}$  (och  $25_{16}$ .)

Programmets startadress är  $40_{16}$

Grundläggande Datorteknik LV5

2

**Dagens mål: Du ska kunna.....**

LV5 Fo12

- ▶ **Förstå villkorliga hopp i program**
  - ▶ Implementera BEQ-instruktionen i styrenheten.
  - ▶ Använda villkorliga hoppinstruktioner
  - ▶ Förstå begreppen stack, stackpekare och stackinstruktioner
  - ▶ Implementera PSH-instruktionen i styrenheten.
  - ▶ Förstå användningen av subrutiner
  - ▶ Skriva subrutiner
- ▶ "Programmera i FLEX-miljön"

**Fokus på...**

Grundläggande Datorteknik LV5

3

**Villkorliga hopp:**

Om PIN-koden är korrekt given  
- öppna telefonen  
Annars  
- stäng telefonen

...någon addition...  
Om  $C=1$   
- så hoppa till felrutin  
Annars  
- fortsatt beräkningen

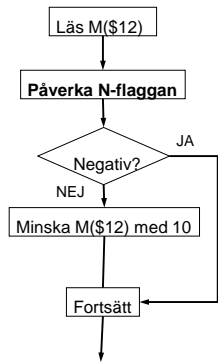
1) Om vi inte bearbetat alla dataord  
- bearbeta nästa dataord  
- börja om från 1)  
2) Annars fortsatt med annat arbete

Vid villkorliga hopp används **relativa hoppinstruktioner**

Grundläggande Datorteknik LV5

4

Villkorliga hopp -Instruktioner



Negativt när N=1

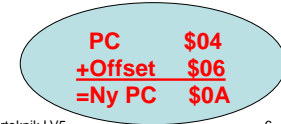
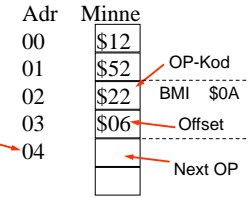
“Testa” SALDO

if SALDO ≥ 0  
minska SALDO med 10:-  
fortsätt

Lämpliga hoppinstruktioner  
BMI eller BPL

Vad gör processorn vid BMI ?

- Läser in HELA branch-instruktionen  
(PC pekar på “nästa” instruktion i minnet  
dvs. PC = \$04)
- Undersöker N-flaggan  
OM N=0  
FETCH på adress \$04  
OM N=1  
PC + offset → PC  
FETCH på adress \$0A



LV5 Fo12

Dagens mål: Du ska kunna.....

- ▶ Förstå villkorliga hopp i program
- ▶ **Implementera BEQ-instruktionen i styrenheten.**
- ▶ Använda villkorliga hoppinstruktioner
- ▶ Förstå begreppen stack, stackpekare och stackinstruktioner
- ▶ Implementera PSH-instruktionen i styrenheten.
- ▶ Förstå användningen av subrutiner
- ▶ Skriva subrutiner
- ▶ ”Programmera i FLEX-miljön”

**FOKUS PÅ...**

Relativa villkorliga hopp

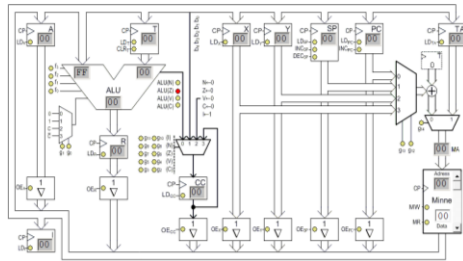
Om Z = 1  
PC+Offset → PC, FETCH

Annars

FETCH

Adress (Hex)	
20	LDA #FFF
21	
22	INCA
23	BEQ \$22
24	
25	BRA \$20
26	
27	

Testa först i FLISP-simulatorn



**Execute**  
BEQ \$22

**LV5 Fo12**

**Dagens mål: Du ska kunna....**

- ▶ Förstå villkorliga hopp i program
- ▶ Implementera BEQ-instruktionen i styrenheten.
- ▶ **Använda villkorliga hoppinstruktioner**
- ▶ Förstå begreppen stack, stackpekare och stackinstruktioner
- ▶ Implementera PSH-instruktionen i styrenheten.
- ▶ Förstå användningen av subrutiner
- ▶ Skriva subrutiner
  
- ▶ "Programmera i FLEX-miljön"

**Fokus på...**

State	Summa-term	RTN	Styrsignaler = 1	Kommentar
Grundläggande Datorteknik LV5				9

Grundläggande Datorteknik LV5

10

**Villkorliga hopp**

**Ext 9**

Instruktionsuppsättningen för FLEX-processorn har ett antal villkorliga hoppinstruktioner.

De kan indelas i följande tre grupper:

1. Enkla hoppvillkor.
2. Hoppvillkor för tal utan inbyggt tecken.
3. Hoppvillkor för tal med inbyggt tecken.  
(2-komplementrepresentation)

Grundläggande Datorteknik LV5

11

**Villkorliga hopp - forts**

Ext 9

**1. Enkla hoppvillkor.**

Vid de enkla villkorliga hoppen testas innehållet i en av flaggvipporna N, Z, V eller C och hoppet utförs om villkoret är uppfyllt, dvs den aktuella flaggvippans värde, är 0 resp 1.

Grundläggande Datorteknik LV5

12

## Villkorliga hopp - forts

Ext 9

**2. Hoppvillkor för tal utan inbyggt tecken.**

Förutsätt att flaggorna har påverkats av en subtraktion  $X - Y$  enligt:

1) LDAA	XVALUE	Läs X från minnet till A
2) CMPA	#Y	Låt skillnaden $X - Y$ påverka flaggorna
3) B(Villkor)	Hoppadress	Utför hoppet om villkoret är uppfyllt

$X$  och  $Y$  är 8-bitars tal som tillhör intervallet  $[0, 255]$ .

Grundläggande Datorteknik LV5

13

## Villkorliga hopp - forts

Ext 9

**2. Hoppvillkor för tal utan inbyggt tecken.**

$X$  och  $Y$  är 8-bitars tal som tillhör intervallet  $[0, 255]$ . **Flaggor C och Z**

$X > Y$ ,  $X \geq Y$ ,  $X = Y$ ,  $X \neq Y$ ,  $X \leq Y$  och  $X < Y$ .

Relation	C	Z	Sant om
$X > Y$	0	0	$C' \cdot Z' = 1$
$X = Y$	0	1	$Z = 1$
$X < Y$	1	0	$C = 1$

Grundläggande Datorteknik LV5

14

## Villkorliga hopp - forts

Ext 9

**3. Hoppvillkor för tal med inbyggt tecken. (2-komplementstal)**

Förutsätt att flaggorna har påverkats av en subtraktion  $X - Y$  enligt:

1) LDAA	XVALUE	Läs X från minnet till A
2) CMPA	#Y	Låt skillnaden $X - Y$ påverka flaggorna
3) B(Villkor)	Hoppadress	Utför hoppet om villkoret är uppfyllt

$X$  och  $Y$  är 8-bitars tal som tillhör intervallet  $[-128, 127]$ .

Grundläggande Datorteknik LV5

15

## Villkorliga hopp - forts

Ext 9

**3. Hoppvillkor för tal med inbyggt tecken.**

$X$  och  $Y$  är 8-bitars tal som tillhör intervallet  $[-128, 127]$ .

**Flaggor N, V och Z**

$X > Y$ ,  $X \geq Y$ ,  $X = Y$ ,  $X \neq Y$ ,  $X \leq Y$  och  $X < Y$ .

Relation	$N \oplus V$	Z	Sant om
$X > Y$	0	0	$(N \oplus V)' \cdot Z' = 1$
$X = Y$	0	1	$Z = 1$
$X < Y$	1	0	$N \oplus V = 1$

Grundläggande Datorteknik LV5

16

## Uppgift

Två variabler P och Q är lagrade i minnet.  
Jämför talen och skriv det största till variabeln R.

P är placerad på adress  $20_{16}$  i minnet.  
Q är placerad på adress  $21_{16}$  i minnet.  
R är placerad på adress  $22_{16}$  i minnet.

Programmets startadress  $40_{16}$  i minnet

## Dagens mål: Du ska kunna.....

- ▶ Förstå villkorliga hopp i program
- ▶ Implementera BEQ-instruktionen i styrenheten.
- ▶ Använda villkorliga hoppinstruktioner
- ▶ **Förstå begreppen stack, stackpekare och stackinstruktioner**
- ▶ Implementera PSH-instruktionen i styrenheten.
- ▶ Förstå användningen av subrutiner
- ▶ Skriva subrutiner

- ▶ "Programmera i FLEX-miljön"

**Fokus på...**

## STACK och STACKPEKARE <sup>Ext 17</sup>

**STACK:** Ett minnesutrymme  
Används för att lagra temporära  
data (registerinnehåll och  
återhopsadresser)

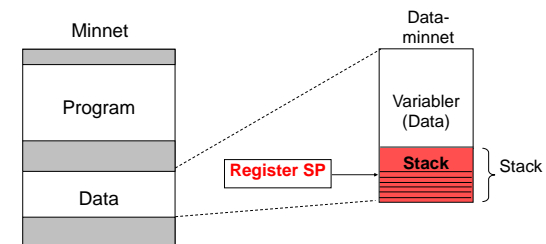
**STACKPEKARE:** Ett register (**Reg SP**)  
som pekar på det senast ditlagda

### INSTRUKTIONER:

**PSH:** Placera ett registerinnehåll **PÅ** stacken

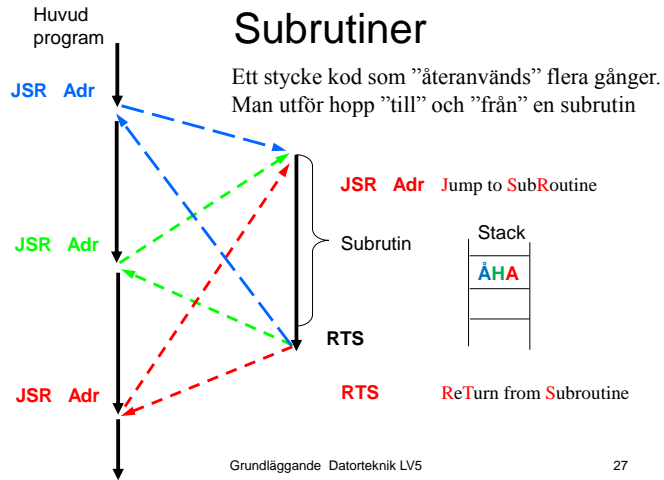
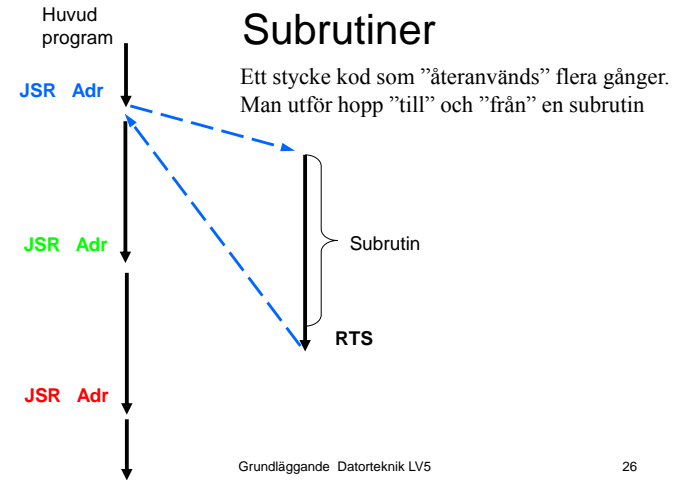
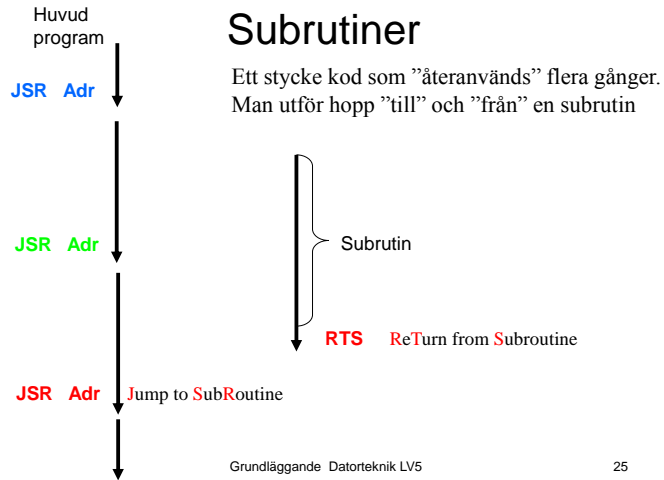
**PUL:** Hämta **FRÅN** stacken **TILL** ett register

## Stacken - forts <sup>Ext 17</sup>



**Stacken: ett minnesutrymme som vi temporärt utnyttjar**

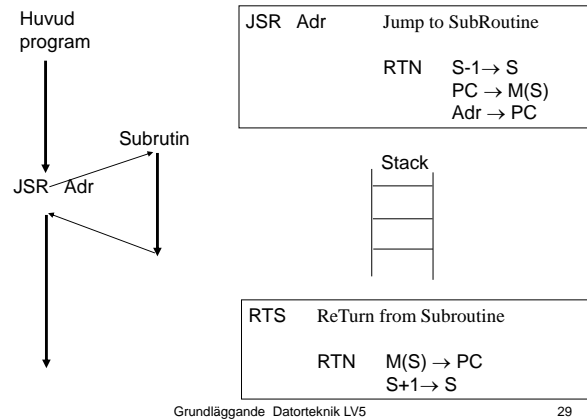




### Subrutin o stack *Aktivera ....*

- Stack:** Ett minnesutrymme
  - Stackpekare:** Register S. Pekar på översta elementet på stacken
  - PSH:** Kopiera ett reg TILL stacken
  - PUL:** Hämta FRÅN stacken TILL ett reg
  - Subrutin:** Ett stycke kod, avslutat med instruktionen RTS
  - JSR:** Hopp TILL subrutin
  - RTS:** Återhopp FRÅN subrutin
- 
- The diagram shows a box labeled 'Data-minnet' containing 'Variabler (Data)'. Below this is a 'Stack' represented by a vertical stack of horizontal lines. A box labeled 'Register S' has an arrow pointing to the top of the stack.

## Subrutiner

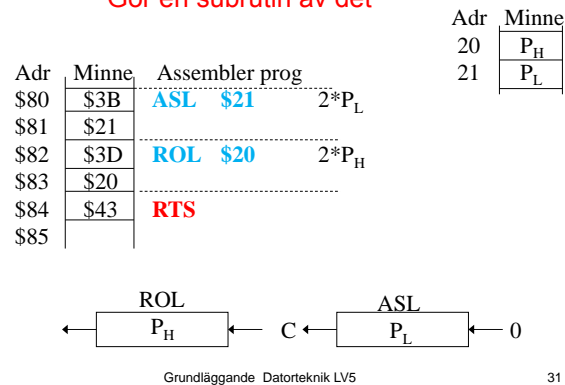


## Dagens mål: Du ska kunna.....

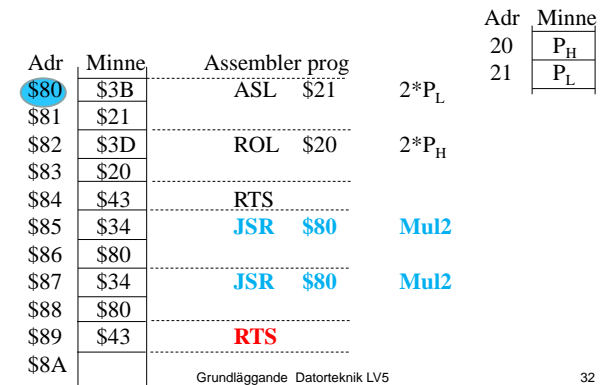
- ▶ Förstå villkorliga hopp i program
- ▶ Implementera BEQ-instruktionen i styrenheten.
- ▶ Använda villkorliga hoppinstruktioner
- ▶ Förstå begreppen stack, stackpekare och stackinstruktioner
- ▶ Implementera PSH-instruktionen i styrenheten.
- ▶ Förstå användningen av subrutiner
- ▶ **Skriva subrutiner**
- ▶ "Programmera i FLEX-miljön"

**Fokus på...**

## Multiplicera den 16-bitars variabeln P med 2 "Gör en subrutin av det"



## Multiplicera den 16-bitars variabeln P med 4 "Gör en subrutin av det"





LV5 Fo12

Dagens mål: Du ska kunna.....

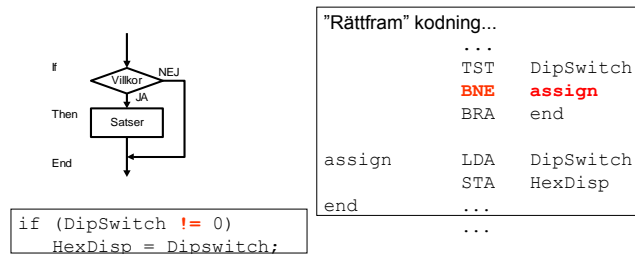
- ▶ Förstå villkorliga hopp i program
  - ▶ Implementera BEQ-instruktionen i styrenheten.
  - ▶ Använda villkorliga hoppinstruktioner
  - ▶ Förstå begreppen stack, stackpekare och stackinstruktioner
  - ▶ Implementera PSH-instruktionen i styrenheten.
  - ▶ Förstå användningen av subrutiner
  - ▶ Skriva subrutiner
- 
- ▶ "Programmera i FLEX-miljön"

**FOKUS PÅ...**

Grundläggande Datorteknik LV5

33

Kontrollstruktur if (...) {Satser}



BNE	"Hopp" om ICKE zero	Z=0
BEQ	"Hopp" om zero	Z=1

Grundläggande Datorteknik LV5

35

Veckans mål:

LV5 Fo13

- ▶ Konstruera styrenheten... genom att....
- ▶ .... implementera olika maskininstruktioner i styrenheten.
- ▶ Villkorliga hopp
- ▶ Subrutiner och stack
- ▶ "Programmera i FLISP-miljön"
- ▶ Strukturerad assemblerprogrammering

Dagens mål: Du ska kunna....

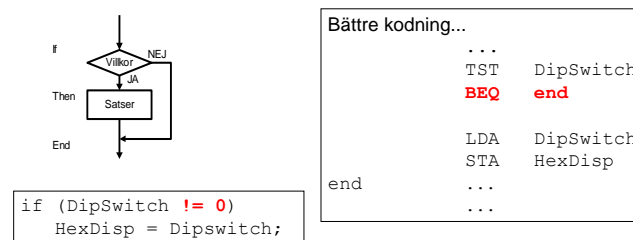
- ▶ Skriva kontrollstrukturer
  - ▶ If then else; While, Do while
  - ▶ "testa bitar"
- ▶ Använda IO-simulatorer
- ▶ Skriva strukturerade assemblerprogram för FLISP
  - ▶ Programmoduler
  - ▶ Top-down/Bottom up
  - ▶ Inre och yttre dokumentation

**Läs smart!  
Lär dig mer!**

Grundläggande Datorteknik LV5

34

Kontrollstruktur if (...) {Satser}

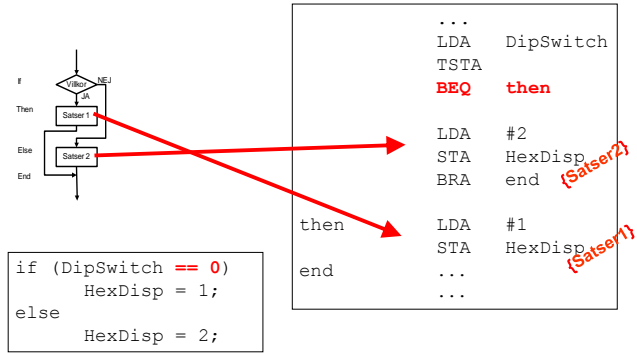


BNE	"Hopp" om ICKE zero	Z=0
BEQ	"Hopp" om zero	Z=1

Grundläggande Datorteknik LV5

36

### Kontrollstruktur **if (...) {Sats1} else {Sats2}**

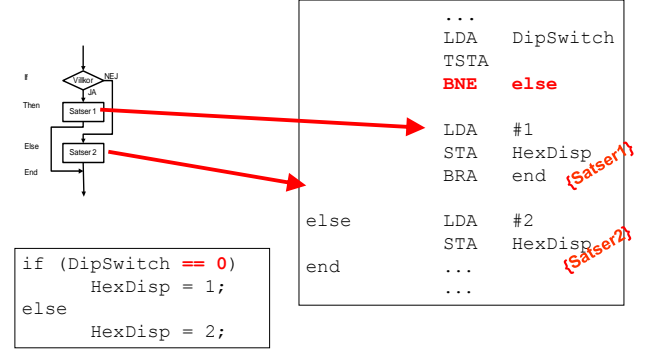


BEQ	"Hopp" om zero	Z=1
-----	----------------	-----

Grundläggande Datorteknik LV5

37

### Kontrollstruktur **if (...) {Sats1} else {Sats2}**

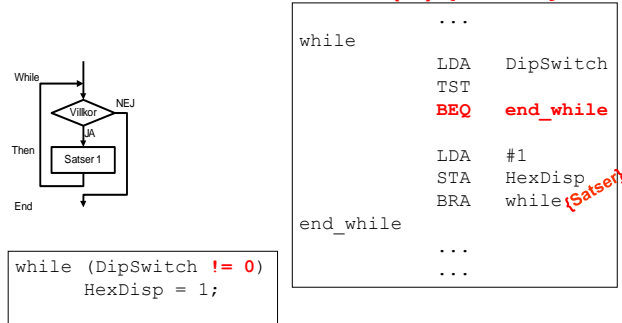


BNE	"Hopp" om ICKE zero	Z=0
-----	---------------------	-----

Grundläggande Datorteknik LV5

38

### Kontrollstruktur **while (...) {Sats1}**



BEQ	"Hopp" om zero	Z=1
-----	----------------	-----

Grundläggande Datorteknik LV5

39

### Kontrollstruktur **do {Sats1} while (...)**



Test av tal utan tecken		
BBS	Vilkor: R>M	C=0

Grundläggande Datorteknik LV5

40

Villkorlig programflödeskontroll

Mnemonic	Funktion	Villkor
<b>Enkla flaggtest</b>		
BCS	"Hopp" om <i>carry</i>	C=1
BCC	"Hopp" om ICKE <i>carry</i>	C=0
BEQ	"Hopp" om <i>zero</i>	Z=1
BNE	"Hopp" om ICKE <i>zero</i>	Z=0
BMI	"Hopp" om <i>negative</i>	N=1
BPL	"Hopp" om ICKE <i>negative</i>	N=0
BVS	"Hopp" om <i>overflow</i>	V=1
BVC	"Hopp" om ICKE <i>overflow</i>	V=0
<b>Test av tal utan tecken</b>		
BHI	Villkor: R>M	C + Z = 0
BHS	Villkor: R≥M	C=0
BLO	Villkor: R<M	C=1
BLS	Villkor: R≤M	C + Z = 1
<b>Test av tal med tecken</b>		
BGT	Villkor: R>M	Z + (N ⊕ V) = 0
BGE	Villkor: R≥M	N ⊕ V = 0
BLT	Villkor: R<M	N ⊕ V = 1
BLE	Villkor: R≤M	Z + (N ⊕ V) = 1

Grundläggande Datorteknik LV5

41

Dagens mål: Du ska kunna....

- ▶ Skriva kontrollstrukturer
  - ▶ If then else; While, Do while
  - ▶ "testa bitar"
- ▶ Använda IO-simulatorer
- ▶ Skriva strukturerade assemblerprogram för FLISP
  - ▶ Programmoduler
  - ▶ Top-down/Bottom up
  - ▶ Inre och yttre dokumentation

**Läs smart!  
Lär dig mer!**

Grundläggande Datorteknik LV5

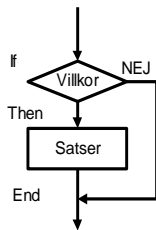
42

Att testa bitar på inporten – EN BIT

If villkor then  
Satser  
end

if b<sub>3</sub>=1 then  
Satser  
end

```
LDA Inport
ANDA #%00001000
BEQ end
--- Satser
end
---
```



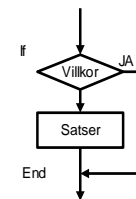
```
LDA Inport
BITA #%00001000
BEQ end
--- Satser
end
---
```

Grundläggande Datorteknik LV5

43

Att testa bitar på inporten – FLERA BITAR

if (b<sub>3</sub>=1) && (b<sub>0</sub>=1) then goto end  
Satser  
end



```
LDA Inport
ANDA #%00001001
CMPA #%00001001
BEQ end
--- Satser 1
end
---
```

Grundläggande Datorteknik LV5

44

## LV5 Fo13

**Dagens mål: Du ska kunna....**

- ▶ Skriva kontrollstrukturer
  - ▶ If then else; While, Do while
  - ▶ "testa bitar"
- ▶ **Använda IO-simulatorer**
- ▶ Skriva strukturerade assemblerprogram för FLISP
  - ▶ Programmoduler
  - ▶ Top-down/Bottom up
  - ▶ Inre och yttre dokumentation

Grundläggande Datorteknik LV5

45

**Läs smart!  
Lär dig mer!**

## Exempel IN- o UT-matning / Testa en bit

Skriv ett program som hela tiden läser inporten (DipSwitch), om  $b_6$  av inporten noll, skriv 7 till utporten (Hexdisplay) annars skriv 22 till utporten

Programmets startadress: \$10

Grundläggande Datorteknik LV5

46

## Exempel IN- o UT-matning / Testa en bit

Skriv ett program som hela tiden läser inporten (Keyboard), om tangent nummer 6 är aktiverad, skriv \$66 till utporten (HexDisplay) annars skriv \$FF till utporten

Programmets startadress: \$30

Grundläggande Datorteknik LV5

47

## LV5 Fo13

**Dagens mål: Du ska kunna....**

- ▶ Skriva kontrollstrukturer
  - ▶ If then else; While, Do while
  - ▶ "testa bitar"
- ▶ Använda IO-simulatorer
- ▶ **Skriva strukturerade assemblerprogram för FLISP**
  - ▶ Programmoduler
  - ▶ Top-down/Bottom up
  - ▶ Inre och yttre dokumentation

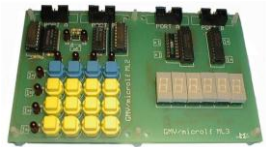
Grundläggande Datorteknik LV5

48

**Läs smart!  
Lär dig mer!**

## Programmeringsprojekt:

### Bygg ett stoppur



Du ska kunna:

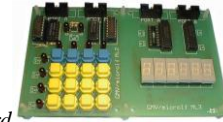
1. Starta klockan
2. Stoppa klockan
3. Nollställa klockan

## Programmeringsprojekt

### Bygg ett stoppur

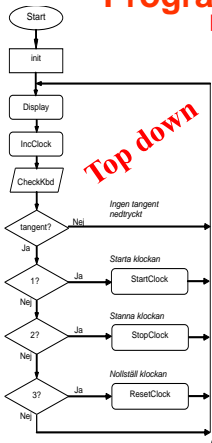
Vad behövs?

1. Räknare som håller reda på tiden
2. Huvudprogram
3. Rutin som visar tiden
4. Hur lagras tiden, format?
5. Rutin som ändrar (ökar) tiden
6. Rutin som läser av tangentbord...
7. Koda tangentnumren (Start, stop, nollställ klockan)
8. Rutin som startar klockan
9. Rutin som stannar klockan
10. Rutin som nollställer klockan
11. mm

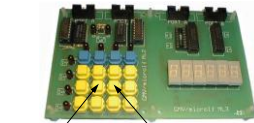


## Programmeringsprojekt

### Bygg ett stoppur



Top down



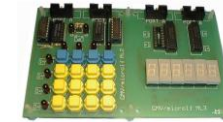
Start Stopp

Adress	Data	Kommentar
Clock	\$02	Timmar
	\$05	Tio minuter
	\$01	Minuter
	\$03	Tio sekunder
	\$07	Sekunder
	\$04	Tiondelar

Bottom up

## Programmeringsprojekt

1. Initiera
2. Main Loop
3. Subrutin som visar tiden
4. Subrutin som ändrar (ökar) tiden
5. Rutin som läser av tangentbordet och  
Startar klockan eller  
Stoppa klockan eller  
Nollställer klockan



Dokumentation  
Yttre o Innre

## YTTRE Dokumentation

```
*****
*SUBROUTIN - DELAY
* Beskrivning: Skapar en fördröjning om
* ANTAL x 500 ms.
*
* Indata: Antal intervall, om 500 ms i A
* Anrop : LDA #6      Fördröj 6*500ms = 3s
*         JSR DELAY
*
* Utdata: Inga
* Register-påverkan: Ingen
* Anropad subrutin: Ingen.
*****
```

Grundläggande Datorteknik LV5

53

## INNRE Dokumentation

```
DELAY PSHA           Spåra reg på stacken
      PSHX
      TSTA           Är fördröjningsvärde noll ?
      BEQ DelExit   Hoppa om noll

ALOOP LDX #Konst

* Snurra som tar 10 klockcykler
XLOOP LEAX-1,X      2 cykler
      NOP           1 cykel
      CMPX#0        2 cykler
      BNE XLOOP     3 cykler

DECA
BNE ALOOP           Ytterligare fördröjning ?
---
```

Grundläggande Datorteknik LV5

54

Skriv en rutin som räknar antal ettställda bitar på inporten (Dipswitch) och skriver resultatet på utporten (Hexdisplay)

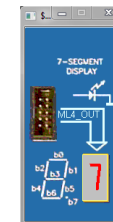
Programmets startadress:	\$20
Inport adress:	\$FB
Utport adress:	\$FC

Lös uppgiften en gång till – fast på ett strukturerat sätt

Grundläggande Datorteknik LV5

55

Skriv ut siffrorna [0..9<sub>10</sub>] på sifferindikatoren

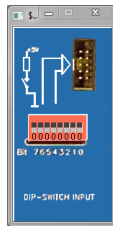


Testa programmet med "Run Slow" och "Run Fast"

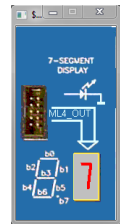
Digital o Datorteknik fo 14

56

Skriv strömbrytarnas värde  $[0..F_{16}]$  på sifferindikatoren

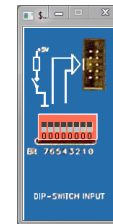


Digital o Datorteknik fo 14



57

Skriv strömbrytarnas värde  $[0..9_{10}]$  på sifferindikatoren



Digital o Datorteknik fo 14



58