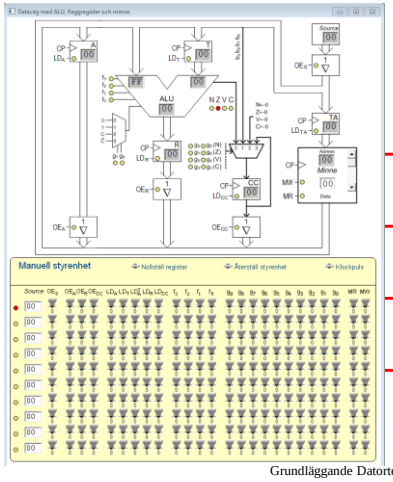
Lärandemål:

5) Datorns uppbyggnad och funktion

1. Beskriva, analysera och konstruera kombinatoriska och sekventiella nät som typiskt används för att bygga en dators centralenhet, dvs. dataväg, styrenhet, aritmetisk/logisk enhet (Arithmetic/Logical Unit).
2. Beskriva, analysera och konstruera en styrenhet baserad på fast kopplad logik och kunna implementera instruktions-exekvering i denna logik.
3. Kunna utföra elementär maskinnära programmering (maskinprogrammering och assemblerprogrammering).
4. Beskriva in-/ut- matningsenheter och minnessystem tillsammans med centralenheten.

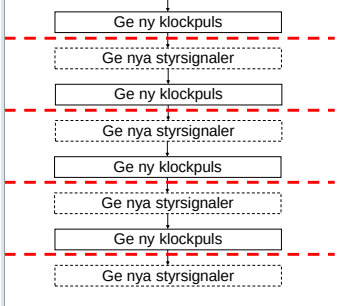
Veckans mål:

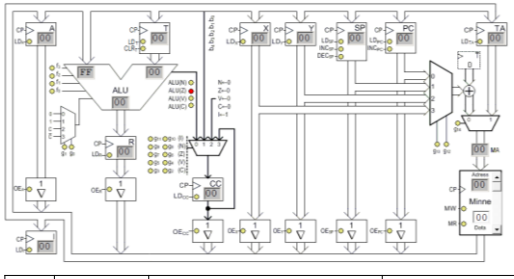
- Konstruera styrenheten…. genom att….
- …. implementera olika maskininstruktioner i styrenheten.
- Villkorliga hopp
- Subrutiner och stack
- Skriva enkla program för FLISP
- Strukturerad assemblerprogrammering



Arb s 65

Enkel dataväg

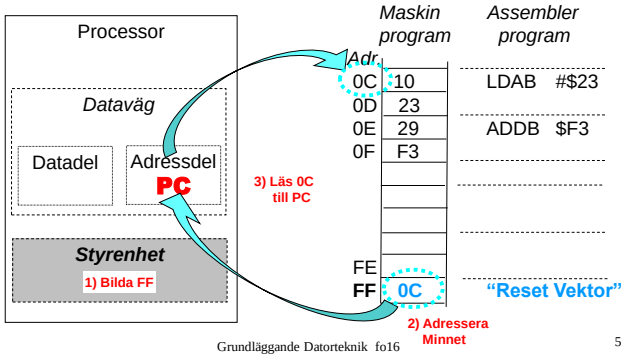




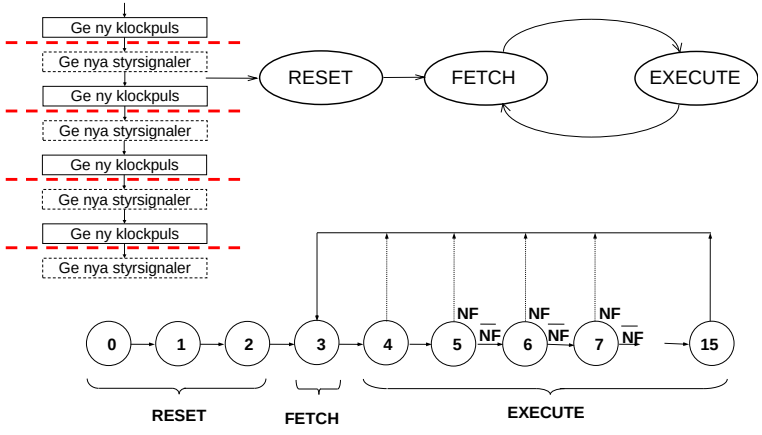
Execute

State	Summa-term	RTN	Styrsignaler = 1	Kommentar

Processorns arbetssätt - RESET



Aktivera "Processorns arbetssätt"



Aktivera "Program och minne"

Instruktionsformat

LDA Adr

INCA

Maskinprogram	*Maskinprogram*
00001111_2	$0F_{16}$
00001011_2	$0B_{16}$
00111111_2	$3F_{16}$
11111110_2	FE_{16}
00011001_2	19_{16}
01000001_2	41_{16}
01001010_2	48_{16}

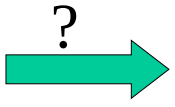
Mask. prog i minnet Adr.		*Tillhörande assemblerprog*	
0C	10	LDA	#$23
0D	23		
0E	29	ADDA	$F3
0F	F3		
10	02	TFR	X,Y
11	4F	CMPA	#$03
12	03		
13	61	BLO	$29
14	13		

Instruktionslistan för FLISP

Handassemblering

Assemblerprogram

```
LDA   #$3C
ADDA  $43
STA   4,X
JMP   $2E
```



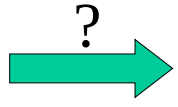
Maskinprogram

Adr	
18	F0
19	3C
1A	96
1B	43
1C	E3
1D	04
1E	33
1F	2E
20	

Disassemblering

Maskinprogram

Adr	
18	F0
19	3C
1A	96
1B	43
1C	E3
1D	04
1E	33
1F	2E
20	



Assemblerprogram

```
LDA   #$3C
ADDA  $43
STA   4,X
JMP   $2E
```


Vad gör processorn vid BMI ?

1) Läser in HELA branch-instruktionen
(*PC pekar på "nästa" instruktion i minnet dvs. PC = $04)*

2) Undersöker N-flaggan
OM N=0
FETCH på adress $04
OM N=1
PC + offset → PC
FETCH på adress $0A

Adr	Minne	
00h	$12	
01h	$52	OP-Kod
02h	$5B	BMI $0A
03h	$06	Offset
04h		Next OP

PC $04
+Offset $06
=Ny PC $0A

Subrutin o stack

Stack: Ett minnesutrymme

Stackpekare: Register SP
Pekar på översta elementet på stacken

Subrutin: Ett stycke kod, avslutat med instruktionen RTS

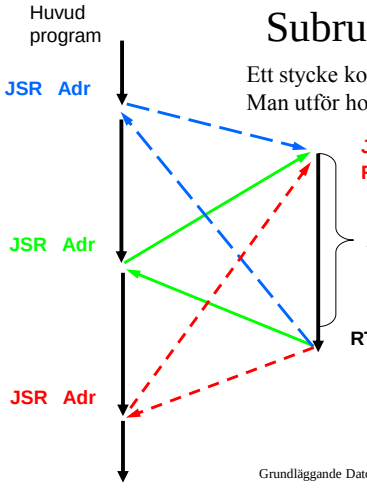
JSR: Hopp TILL subrutin

RTS: Återhopp FRÅN subrutin

Minne

Program 1
Data 1

Subrutiner



Ett stycke kod som ”återanvänds” flera gånger.
Man utför hopp ”till” och ”från” en subrutin

JSR Adr Jump to SubRoutine
RTS ReTurn from Subroutine

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!

Kontrollstruktur **if (...) {Satser1} else {Satser2}**


```
if (DipSwitch >= 5)
      HexDisp = 1;
else
      HexDisp = 2;
```

```
          ...
          LDAB  DipSwitch
          CMPB  #5
          BLO   else

          LDAB  #1
          STAB  HexDisp
          BRA   end
else
          LDAB  #2
          STAB  HexDisp
end       ...
          ...
```

{Satser1}

{Satser2}

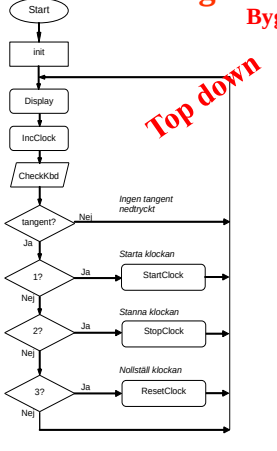
Test av tal utan tecken		
BLO	Villkor: R<M	C=1

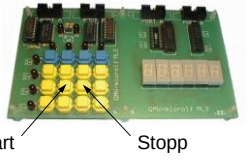
Test av tal med tecken		
BLT	Villkor: R<M	N ⊕ V = 1

Programmeringsprojekt

s40

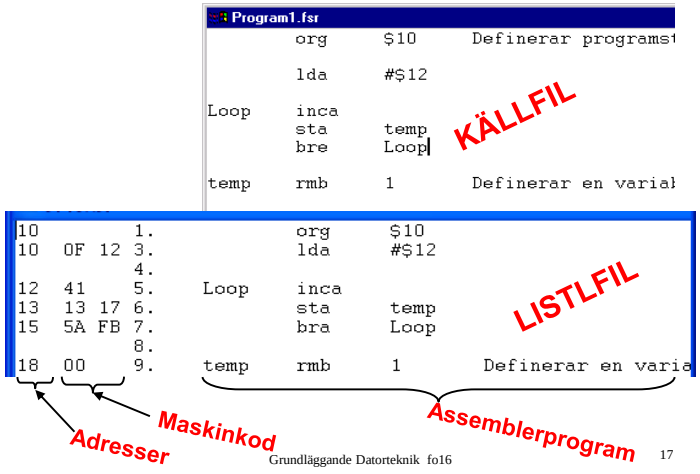
Bygg ett stoppur





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

Bottom up



Assemblerdirektiv

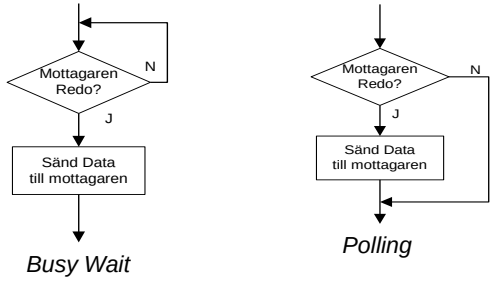
[symbol]	**FCB**	uttryck[,uttryck[,...]]	(**Form Constant Byte**)
[symbol]	**FCS**	"teckensträng"	(**Form Constant String**)
[symbol]	**RMB**	uttryck	(**Reserve Memory Bytes**)
[symbol]	**ORG**	uttryck	(**Origin**)
symbol	**EQU**	uttryck	(**Equate**)

LV6 Fo14

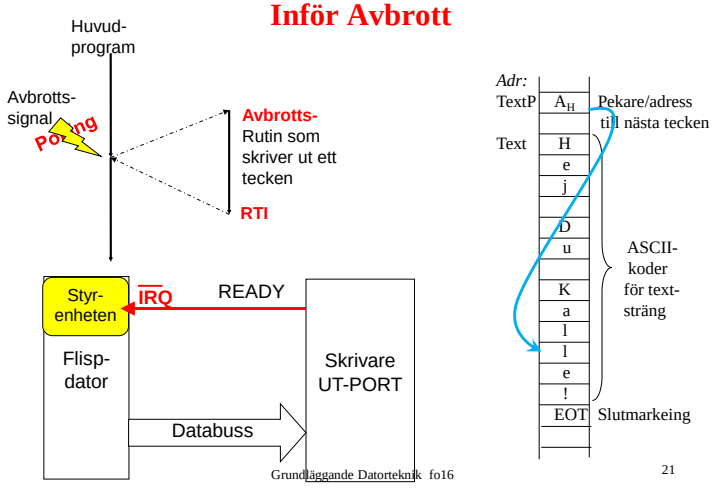
Veckans mål:

- Ansluta In- och Utportar till Flisp
- In-och utmatning
- Avbrott
- Adressavkodning
- Ansluta minnen och I/O-moduler

Villkorlig överföring

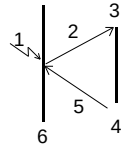


Inför Avbrott



Avbrott - Sammanfattning

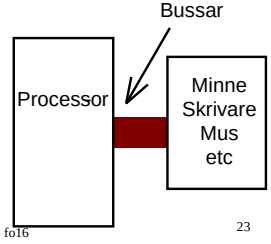
1. **OM I-flaggan=0**: Processorn känner att IRQ är aktiverad och slutför utförandet av pågående instruktion.
2. Processorn sparar huvudprogrammets återhoppsadress och övriga registerinnehåll på stacken, *save status*.
   Därefter läser processorn startadressen för avbrottsrutinen från IRQ-vektorn (från adress $FD).
   Denna startadress placeras i PC. I-flaggan ETT-ställs
3. Avbrottsrutinen startas (med I-flaggan=1).
4. Avbrottsrutinen avslutas med instruktionen RTI som får processorn att utföra *restore status*,
   dvs registerinnehållen återställs från stacken (med gamla I=0).
5. Återhopp till huvudprogram.
6. Därmed återstartas huvudprogrammet där det blev avbrutet



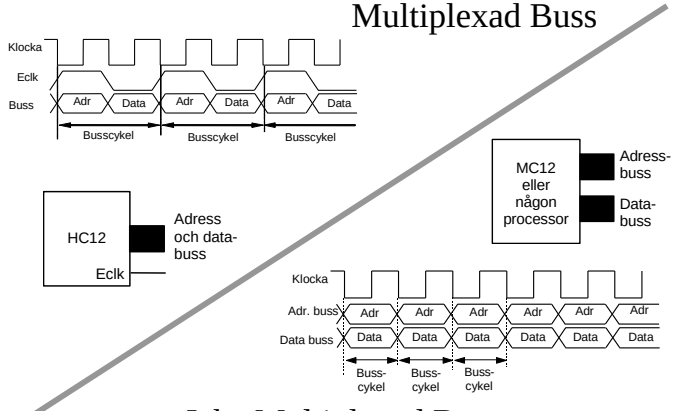
LV6 Fo15

Dagens mål: Du ska kunna….

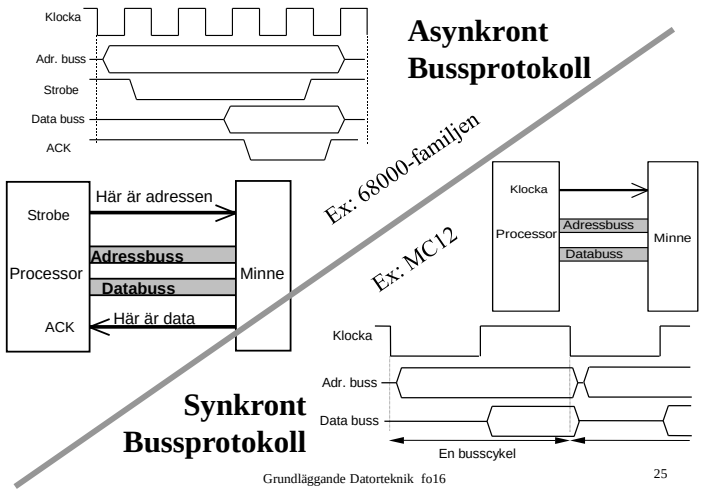
- Skilja på olika minnestyper
- Förklara principer för olika bussprotokoll
- Förstå begreppen Timing och VM (Valid Memory Adress)
- Konstruera adressavkodningslogik ….
  - 1) Fullständig Adr Avk
  - 2) Ofullständig Adr Avk
  - Processorns adressrum



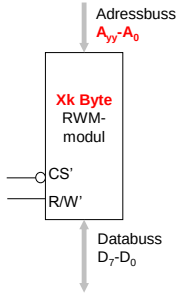
Multiplexad Buss



Icke Multiplexad Buss



Att ansluta en Xk Byte Minnes- modul med startadress $zzzz



Arbetsgång:

- ”Tolka” beskrivningen av minnesmodulen
- Rita tabell
- Ange modulens första adress
- Ange modulens sista adress
- Märk ut konstanta resp varierande adressledningar
- Rita adressavkodningslogiken

Att ansluta en 8-kbyte ROM- modul till ett befintligt system