Aktivera Kursens mål:

LV6 LV7

- Konstruera en dator mha grindar och programmera denna

Aktivera Förra veckans mål:

- Konstruktruera olika kombinatoriska nät som ingår i en dator.
- Studera hur addition/subtraktion utförs (+flaggor)
- Konstruera register som används för att lagra data i datorn.
- Koppla samman register med bussar till en enkel dataväg

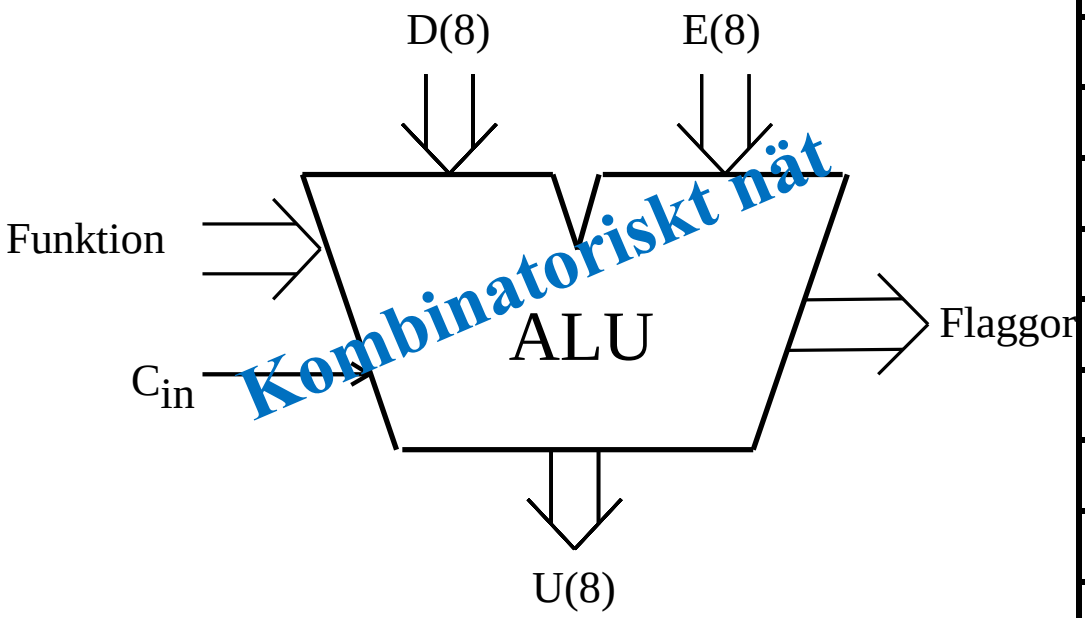
Veckans mål:

- Koppla samman register och ALU till en dataväg
- Förstå hur ett minne är uppbyggd och ansluta detta till datavägen
- Program och hur detta lagras i minne
- Fatta hur datorn startar och arbetar
- Räknare och mera vippor

Dagens mål, Du ska kunna:

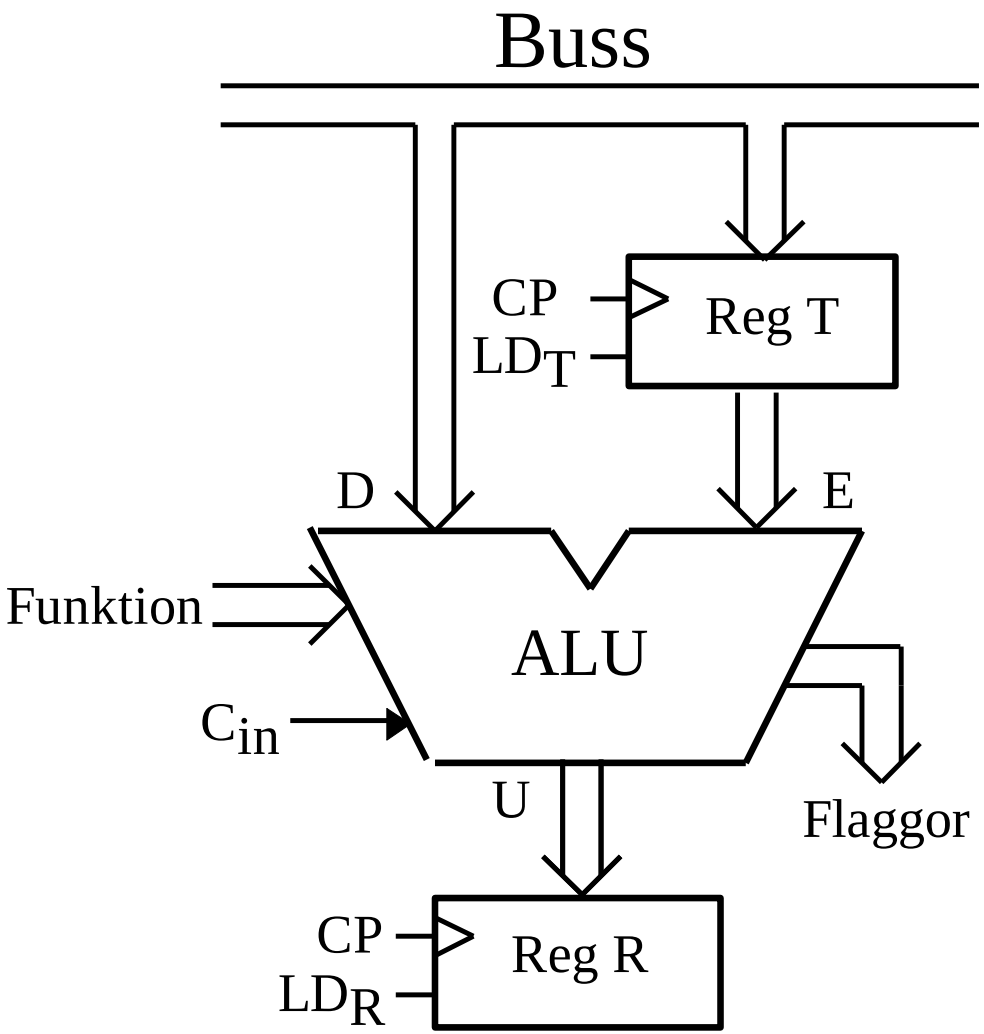
- Konstruera och använda en enkel dataväg
- Kunna programmera en enkel dataväg (RTN)
- Förstå uppbyggnaden av ett minne
- Använda en enkel dataväg med minne
- Förstå von Neumans princip med program och minne
- Ansluta CC-register till datavägen

ALU:n ska anslutas – hur då?

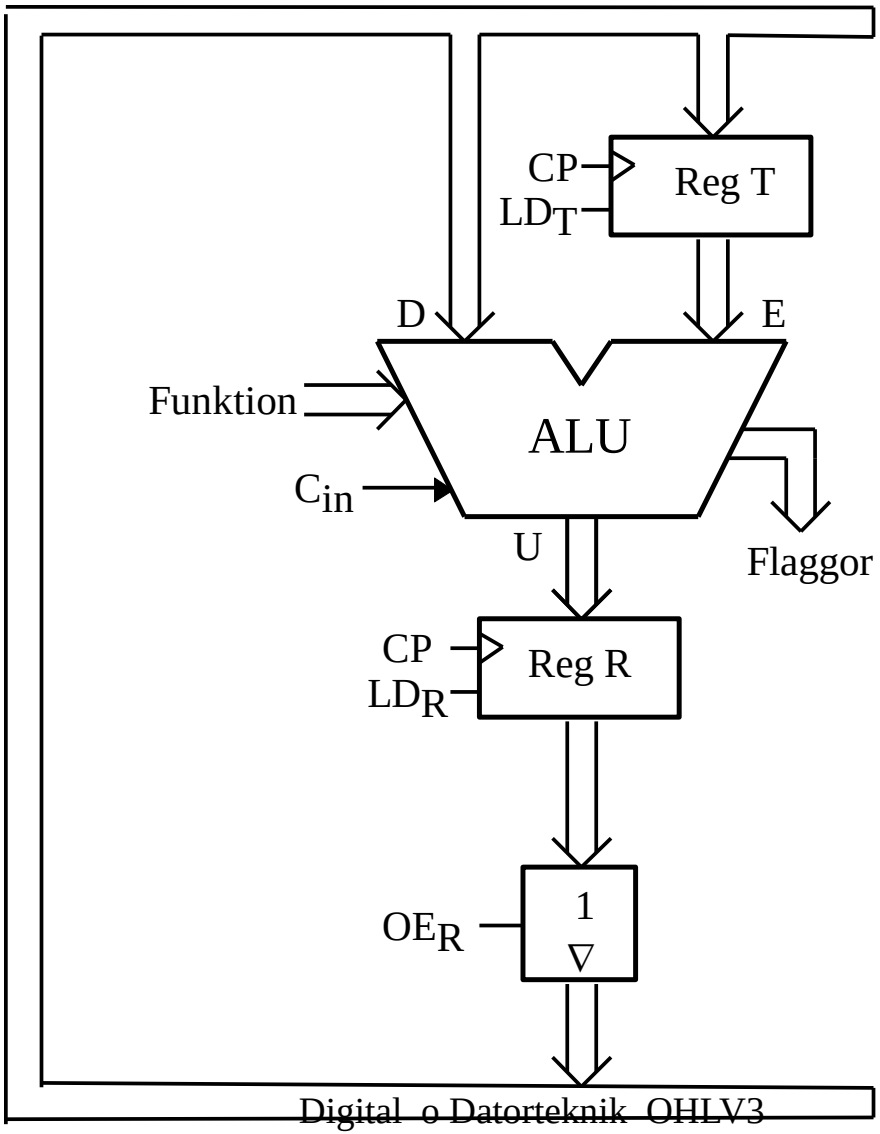


$f_3\ f_2\ f_1\ f_0$	U = f(D,E,F,C_{in})	
	Operation	Resultat
0 0 0 0	Bitvis nollställning	0
0 0 0 1		D
0 0 1 0		E
0 0 1 1	Bitvis invertering	D_{1k}
0 1 0 0	Bitvis invertering	E_{1k}
0 1 0 1	Bitvis OR	D OR E
0 1 1 0	Bitvis AND	D AND E
0 1 1 1	Bitvis XOR	D XOR E
1 0 0 0	D + 0 + C_{in}	D + C_{in}
1 0 0 1	D + FFH + C_{in}	D – 1 + C_{in}
1 0 1 0		D + E + C_{in}
1 0 1 1	D + D + C_{in}	2D + C_{in}
1 1 0 0	D +E_{1k} + C_{in}	D – E – 1 + C_{in}
1 1 0 1	Bitvis nollställning	0
1 1 1 0	Bitvis nollställning	0
1 1 1 1	Bitvis ettställning	FFH

Användning av resultatregister (R) för lagring av utdata från ALU.

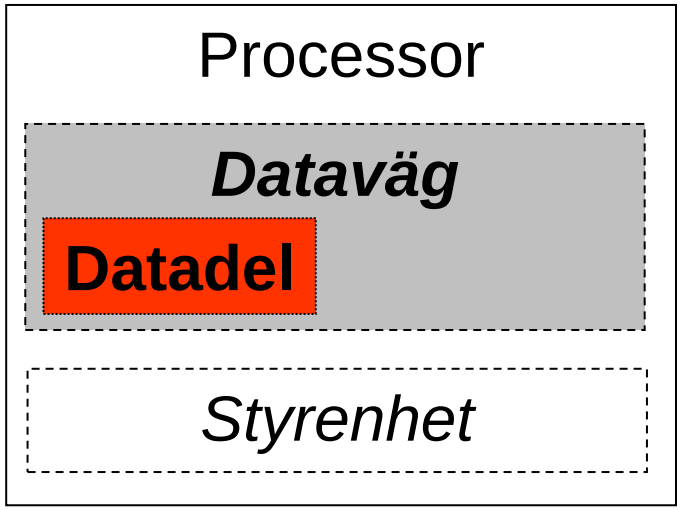


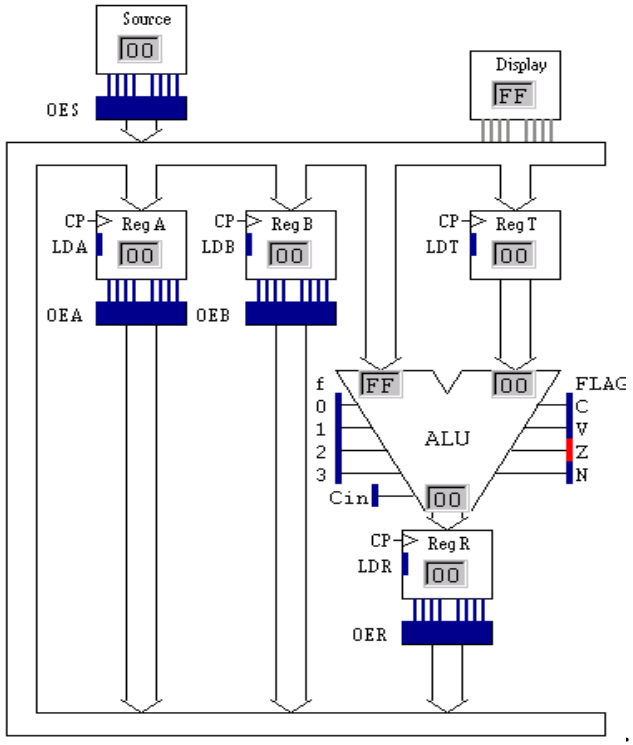
Anslutning av resultatregister (R) till buss.

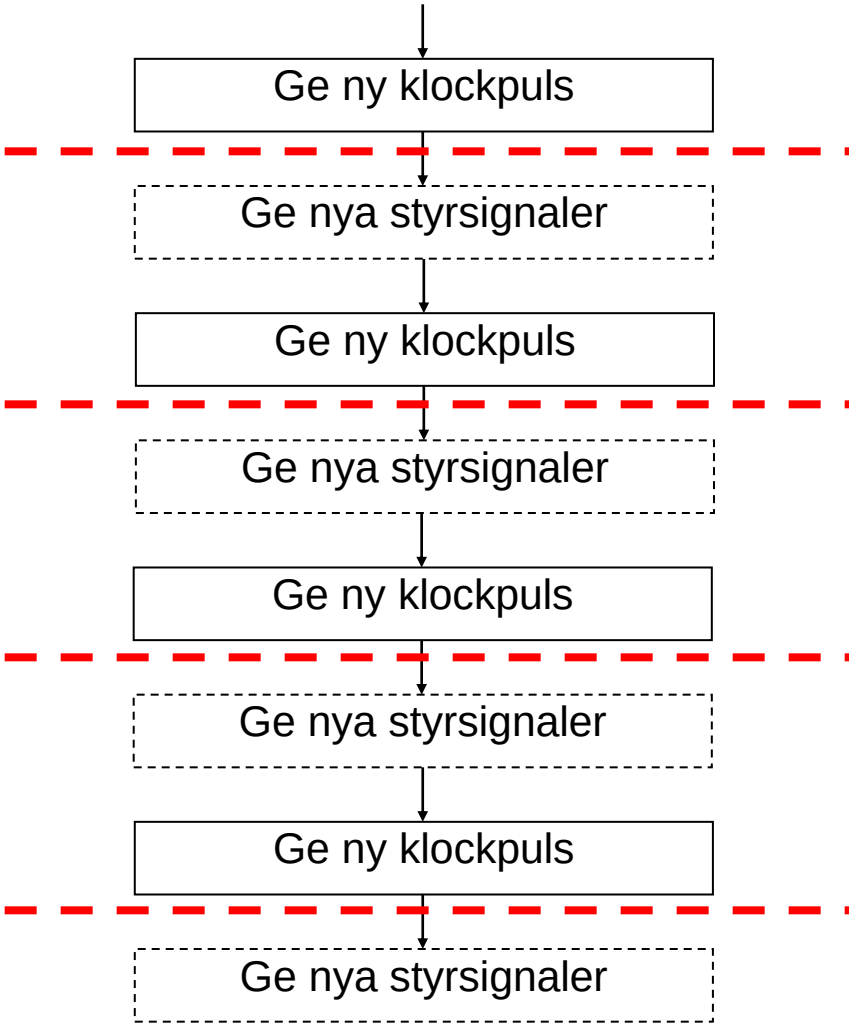


Enkel dataväg

Arb s 74



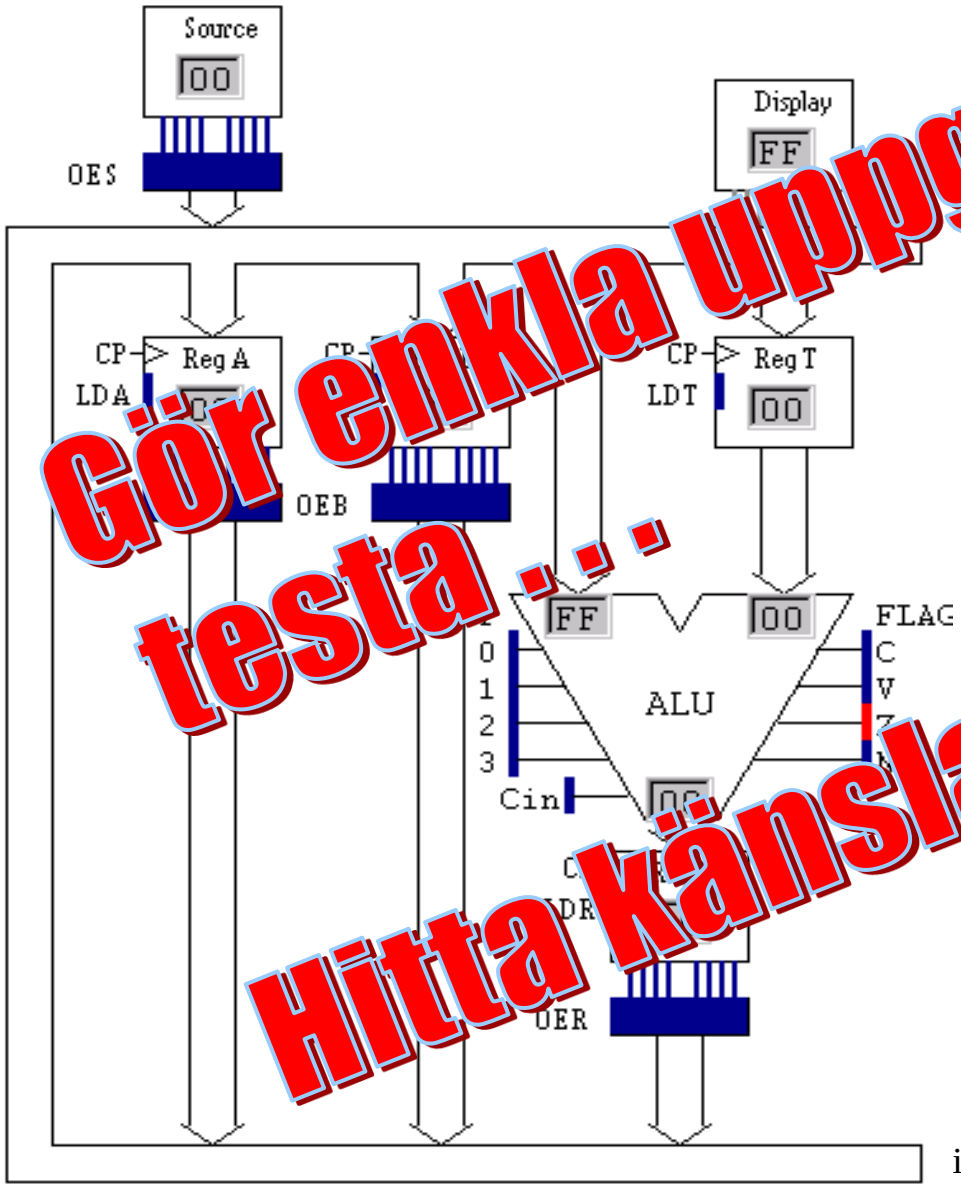




Dagens mål:

- Konstruera och använda en enkel dataväg
- Kunna programmera en enkel dataväg (RTN)
- Förstå uppbyggnaden av ett minne
- Använda en enkel dataväg med minne
- Förstå von Neumans princip med program och minne
- Ansluta CC-register till datavägen

Ge en sekvens av styrsignaler som utför:



Gör enkla uppgifter . . .

testa . . .

Hitta känslan . . .

rivning:

CP1: _____

_____

CP2: _____

_____

CP3: _____

_____

CP4: _____

_____

CP5: _____

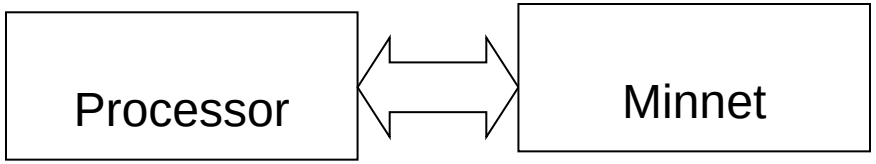
_____

ik OH

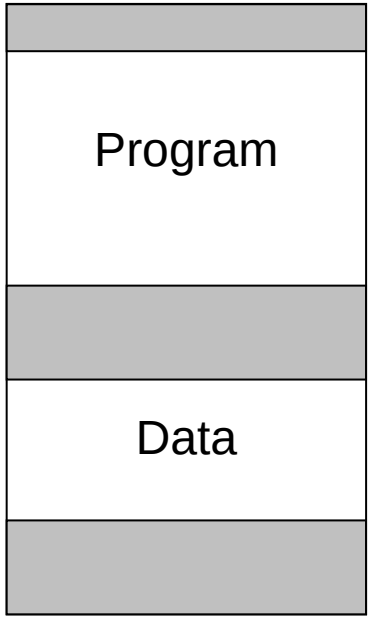
Dagens mål:

- Konstruera och använda en enkel dataväg
- Kunna programmera en enkel dataväg (RTN)
- Förstå uppbyggnaden av ett minne
- Använda en enkel dataväg med minne
- Förstå von Neumans princip med program och minne
- Ansluta CC-register till datavägen

Minnet



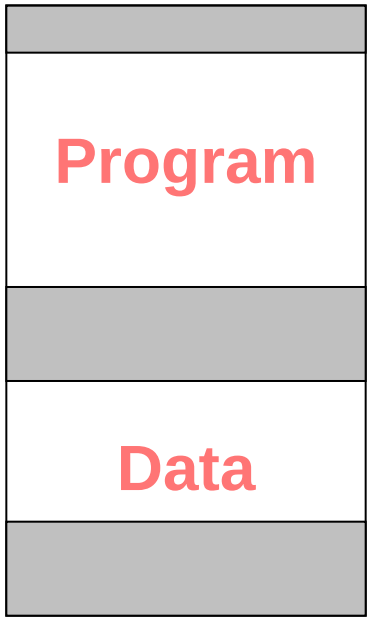
Minnet

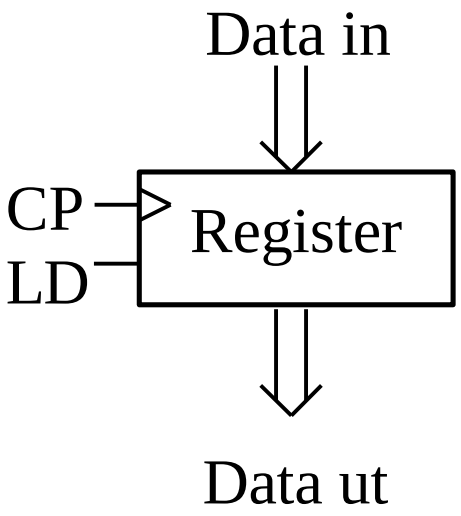


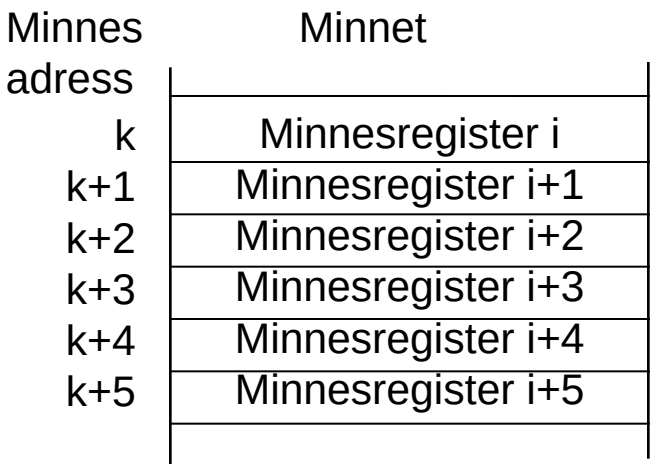
Primärminnen
Sekundärminne
Blockminne

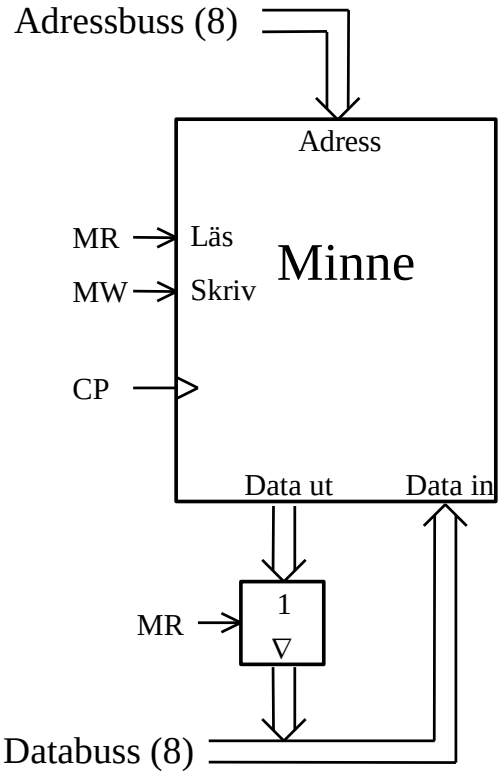
RAM-minnen:
- ROM
- PROM
- FLASH
- RWM

Minne


Program
Data

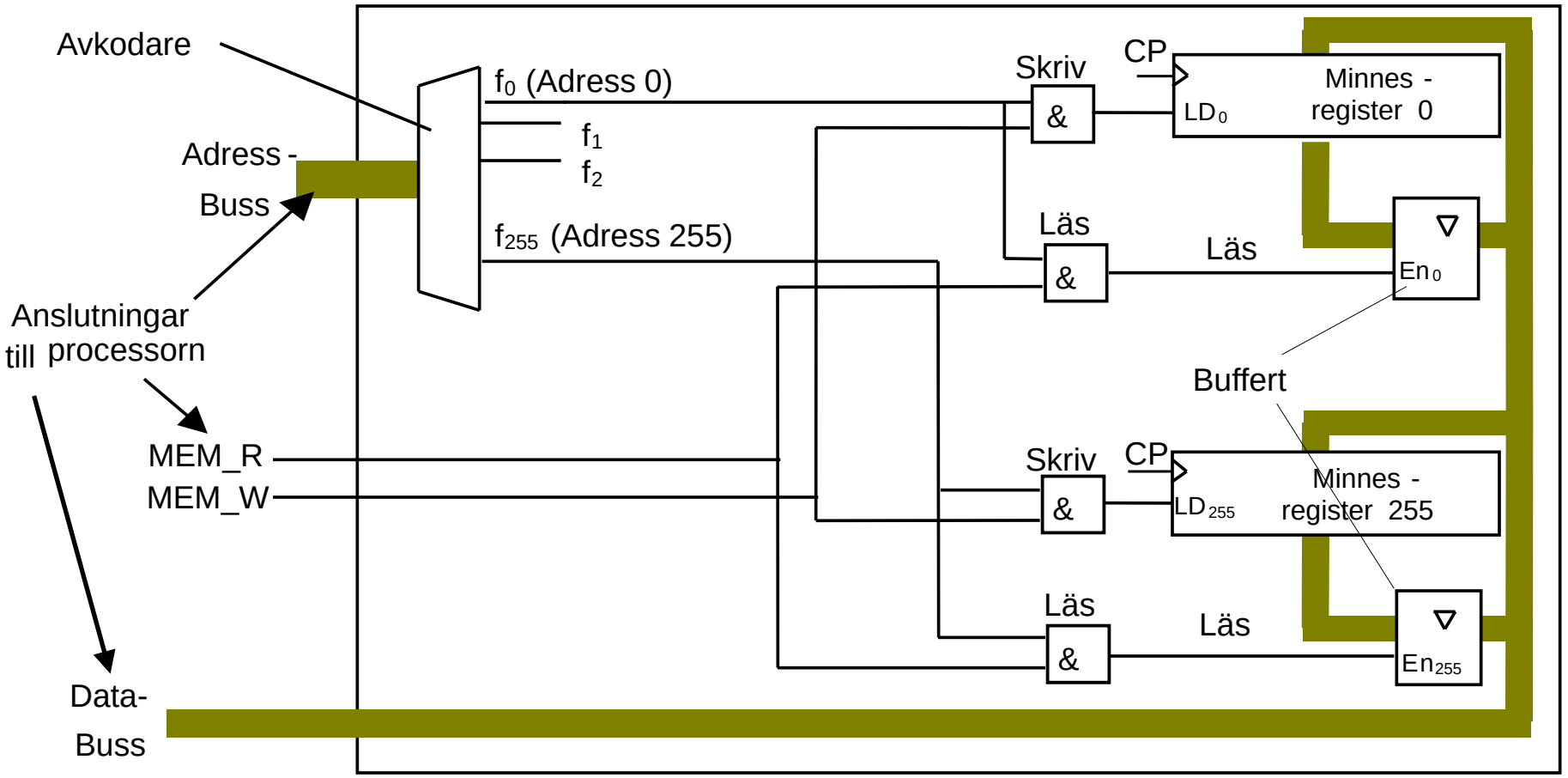

Data in
CP
LD
Register
Data ut


Minnes adress
Minnet
k
k+1
k+2
k+3
k+4
k+5
Minnesregister i
Minnesregister i+1
Minnesregister i+2
Minnesregister i+3
Minnesregister i+4
Minnesregister i+5


Adressbuss (8)
Adress
MR
Läs
MW
Skriv
Minne
CP
Data ut
Data in
MR
1
Databuss (8)

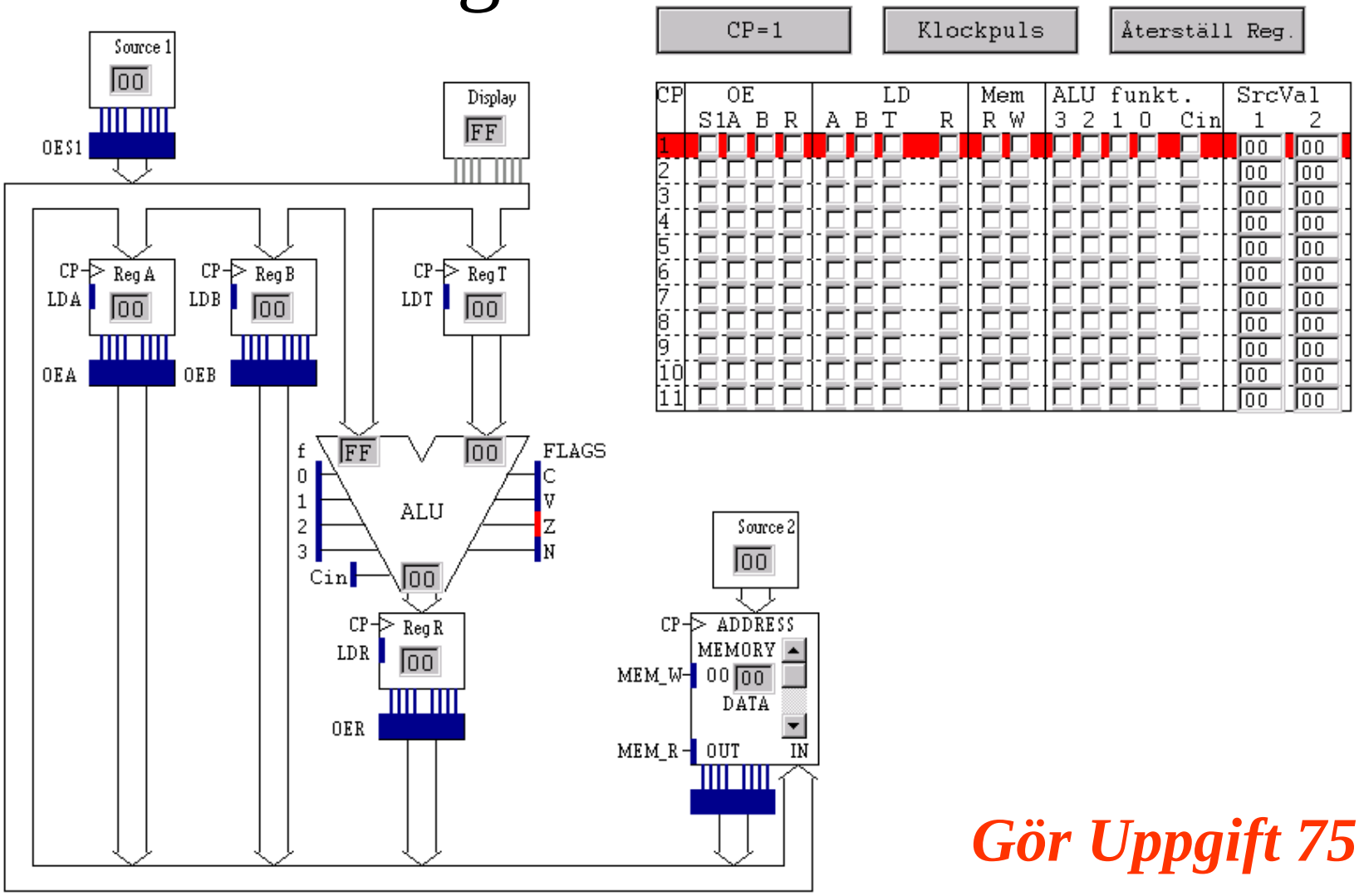
Minns Man Minnet?

Arb s 83



Enkel dataväg med RWM

Arb s 86

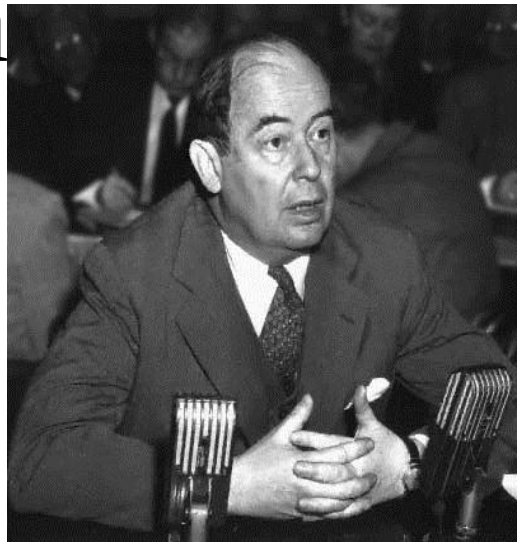


Gör Uppgift 75

Dagens mål:

- Konstruera och använda en enkel dataväg
- Kunna programmera en enkel dataväg (RTN)
- Förstå uppbyggnaden av ett minne
- Använda en enkel dataväg med minne
- Förstå von Neumans princip med program och minne
- Ansluta CC-register till datavägen

Program och min

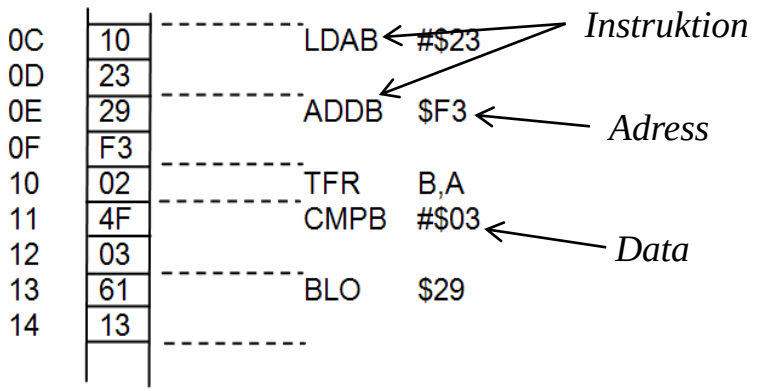


- **John Louis Von Neumann** (1903-1957)

”Det lagrade programmets princip”, dvs program och data i samma minne.

adress hex	innehåll hex	operation
07	01	
08 09	0B 91	- load the A-register with the contents of this address (91H)
0A 0B	28 92	- add to the A-register the contents of this address (92H)
0C 0D	61 07	- branch, if carry=1, to the address = this number (07H) + address of next OPcode (0EH) (= 15H)
0E 0F	13 94	- store the contents of the A-register on this address (94H)
10	47	- clear the A-register
11 12	13 93	- store the contents of acc A (00H) on this address (93H)
13 14	59 1B	- jump to this address (1BH)
15 16	13 94	- store the contents of the A-register on this address (94H)
17 18	0F 01	- load the A-register with this number (01H)
19 1A	13 93	- store contents of the A-register (01H) on this address (93H)
1B	0B	

Maskinprogram i minnet *Tillhörande assemblerprogram*



Program o Minne - forts

Instruktionsformat

INCA

LDAA Adr

Maskinprogram

00001111_2
00001011_2
00111111_2
11111110_2
00011001_2
01000001_2
01001010_2

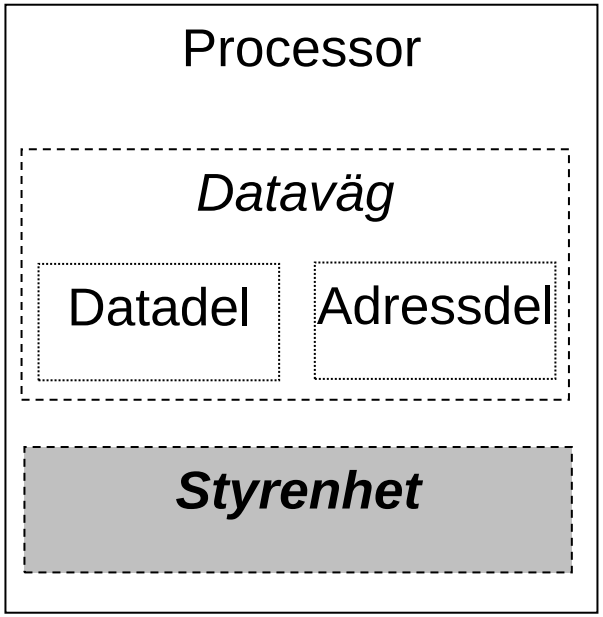
Maskinprogram

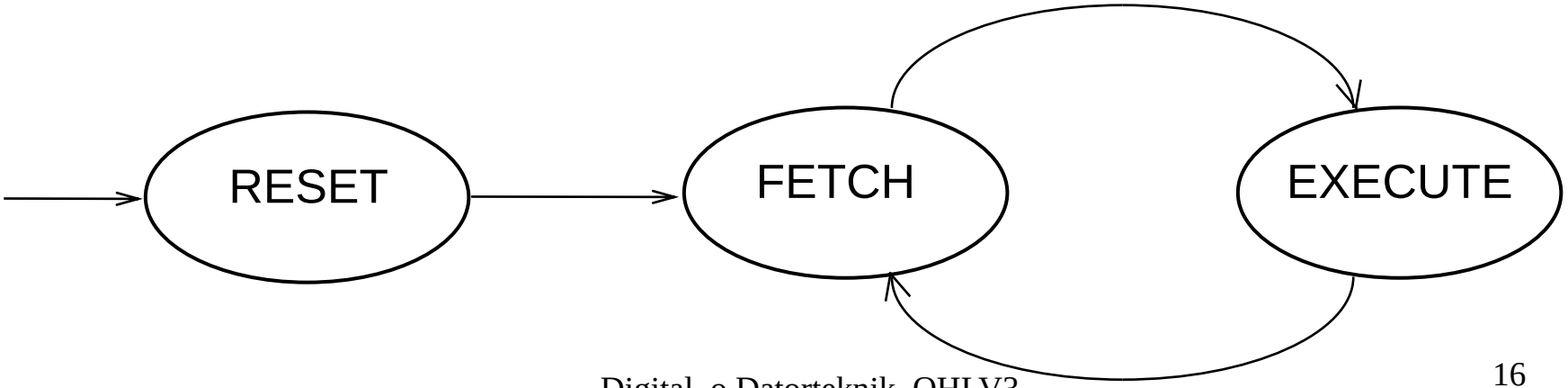
$0F_{16}$
$0B_{16}$
$3F_{16}$
FE_{16}
19_{16}
41_{16}
$4A_{16}$

Processorns arbetssätt

Arb s 95

Adr.	*Maskinprogram i minnet*	*Tillhörande assemblerprogram*	
0C	10	LDAB	#$23
0D	23		
0E	29	ADDB	$F3
0F	F3		
10	02	TFR	B,A
11	4F	CMPB	#$03
12	03		
13	61	BLO	$29
14	13		

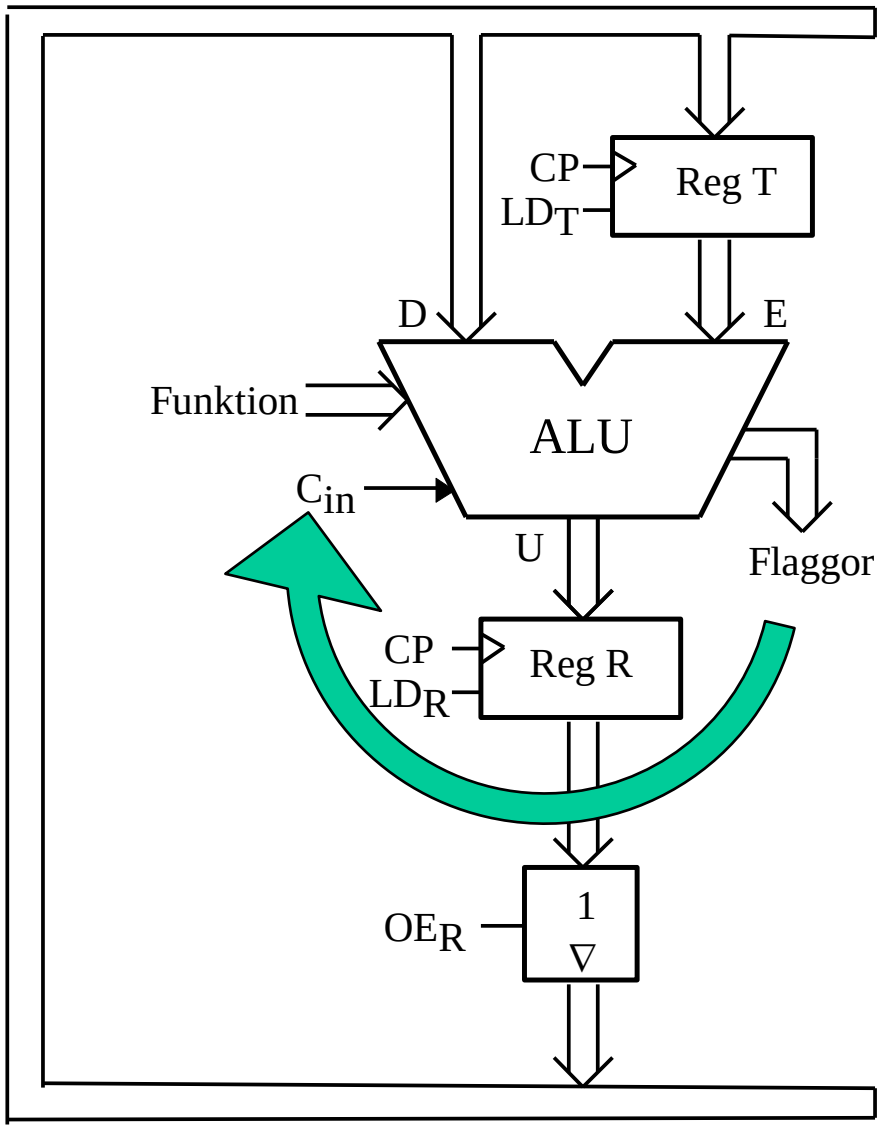




Dagens mål:

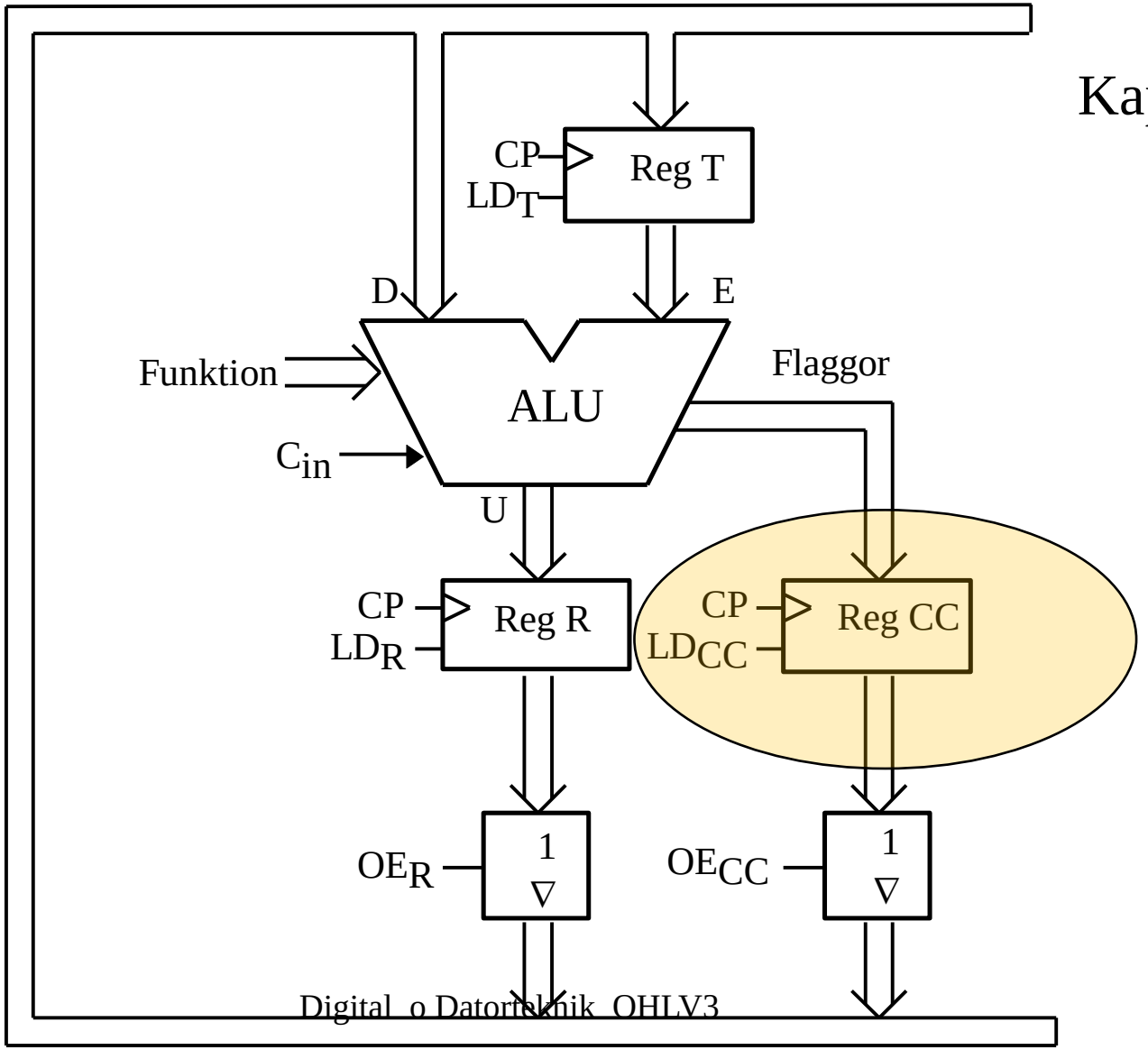
- Konstruera och använda en enkel dataväg
- Kunna programmera en enkel dataväg (RTN)
- Förstå uppbyggnaden av ett minne
- Använda en enkel dataväg med minne
- Förstå von Neumans princip med program och minne
- Ansluta CC-register till datavägen

Fånga C-flaggan.

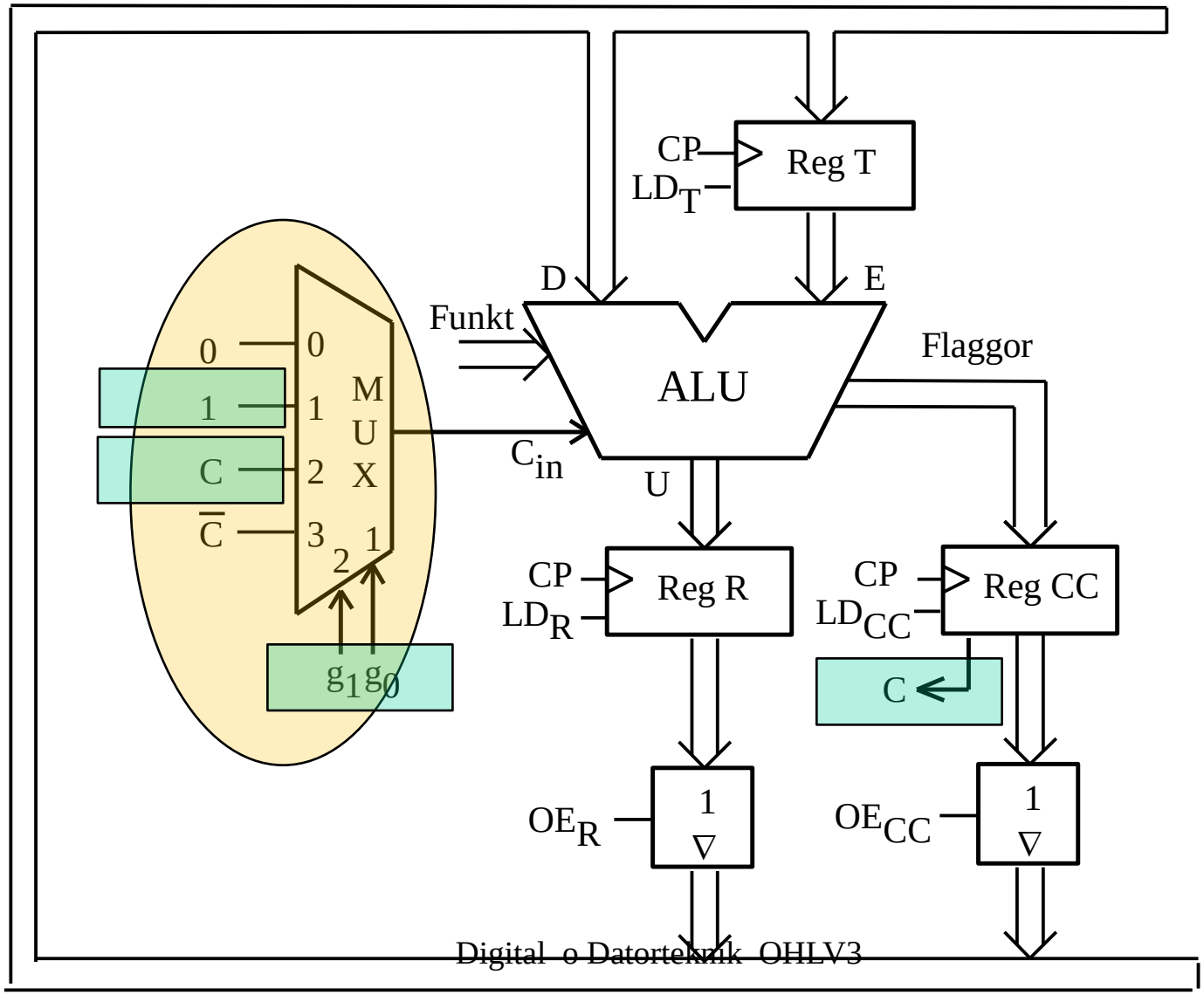


Inkoppling av flaggregister mellan ALU och buss.

Kap 7 Blå

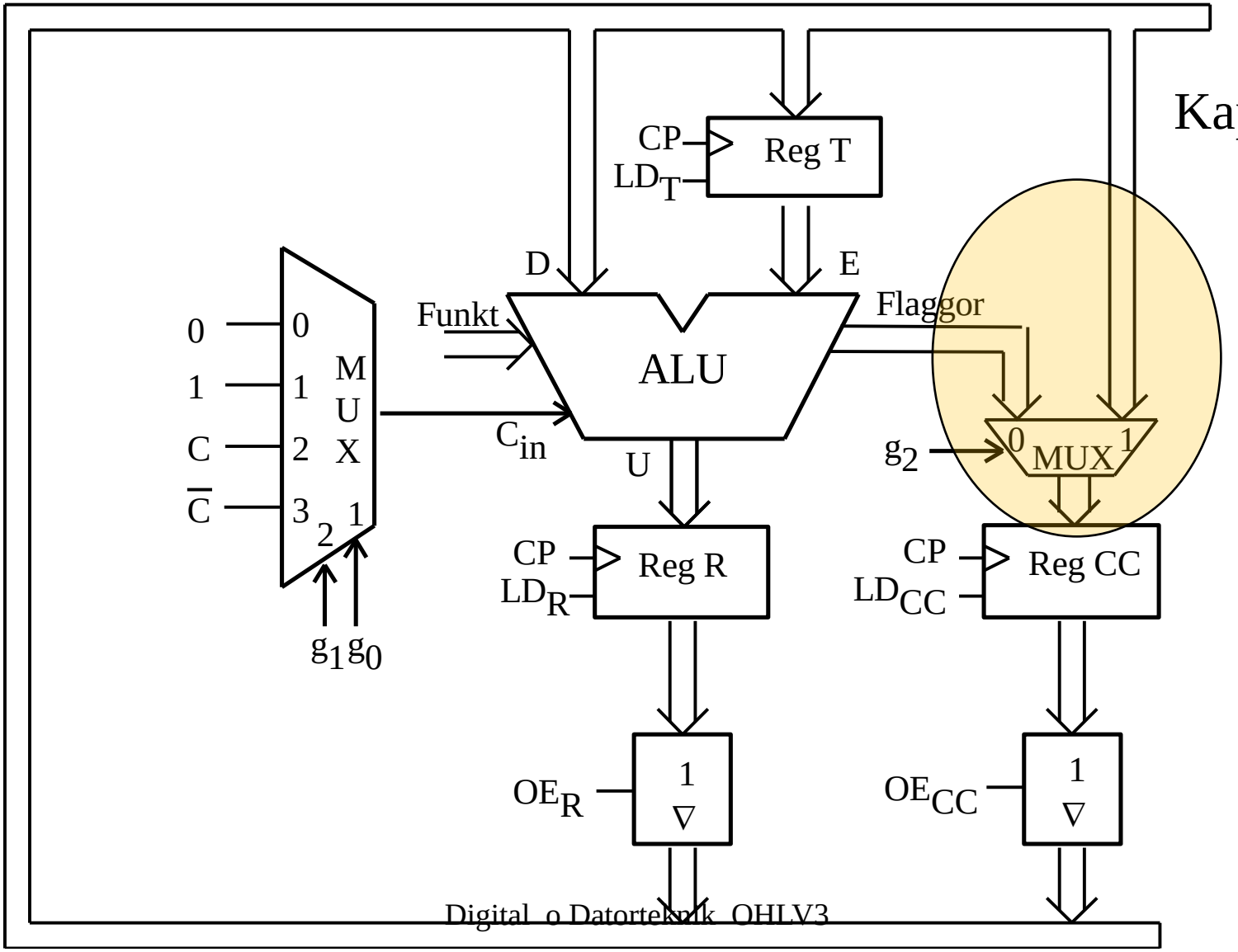


Inkoppling av väljare (multiplexer) för val av C_{in}.

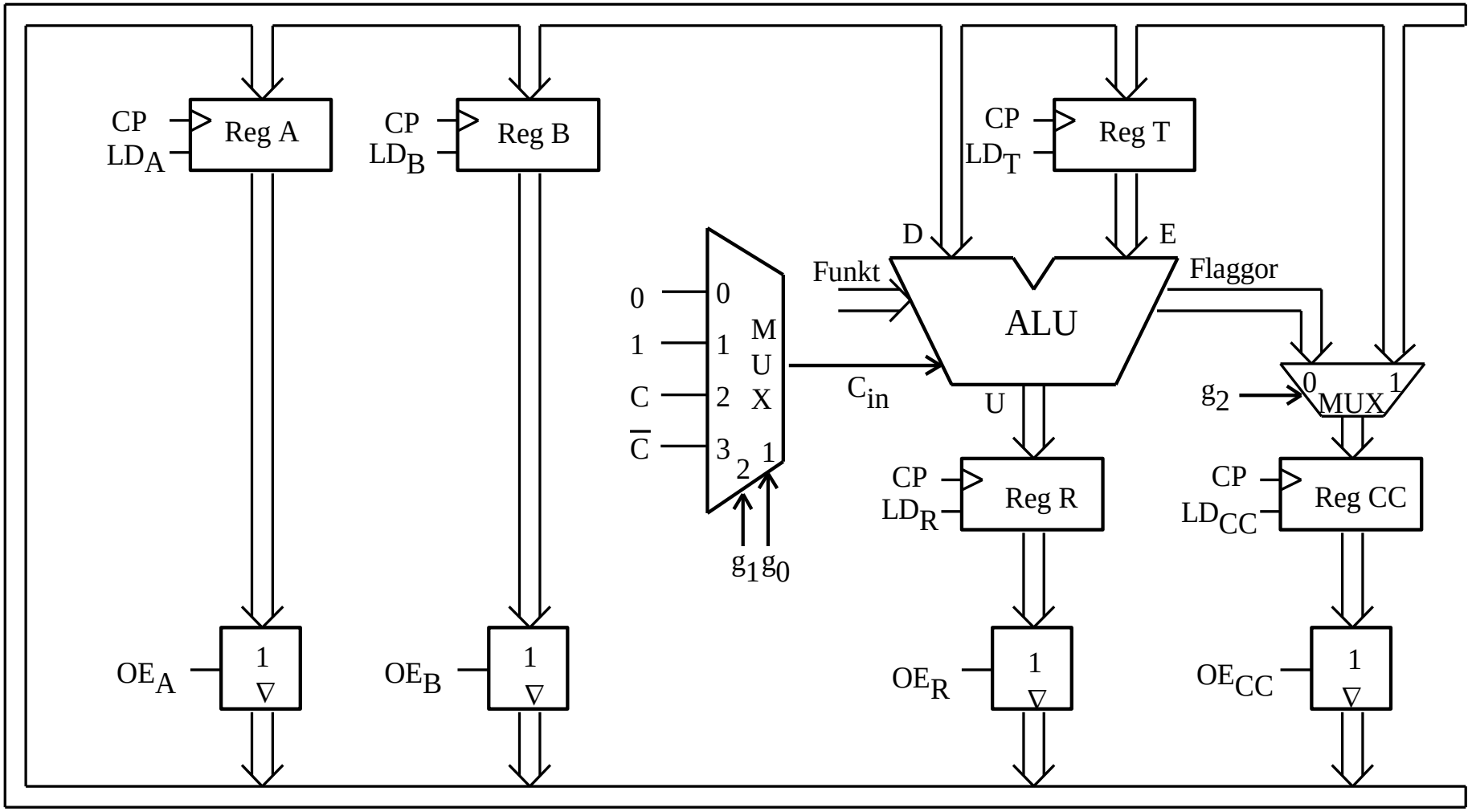


Kap 7 Blå

Inkoppling av väljare (mux) för val av indata till CC-registret.



Logiknät för databehandling med aritmetik/logikenhet (ALU).



Veckans mål:

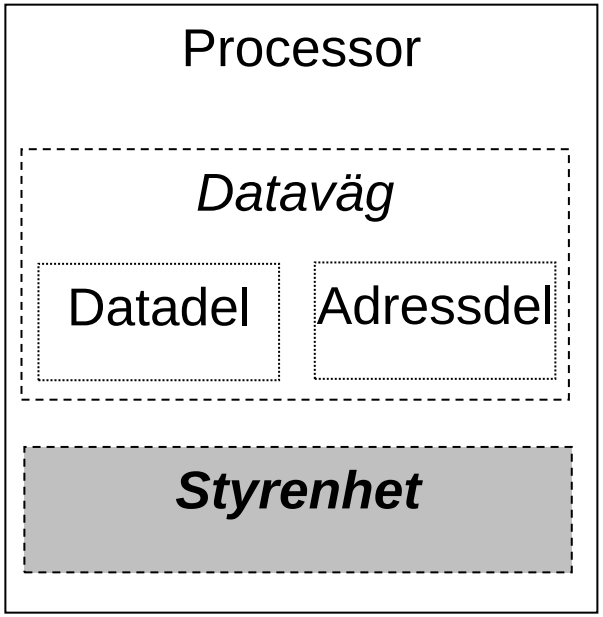
- Koppla samman register och ALU till en dataväg
- Förstå hur ett minne är uppbyggd, ansluta detta till datavägen
- Program och hur detta lagras i minne
- Fatta hur datorn startar och arbetar
- Räknare och mera vippor

Dagens mål. Du ska kunna:

- Beskriva Processorns Arbetssätt
  - Ange styrsignalsekvens för RESET
  - Ange styrsignalsekvens för FETCH
  - Ange styrsignalsekvenser för olika EXECUTE-[illegible]er
- Konstruera och använda JK- och T-vippor
- Kunna analysera räknare
- Ta fram Excitationstabeller

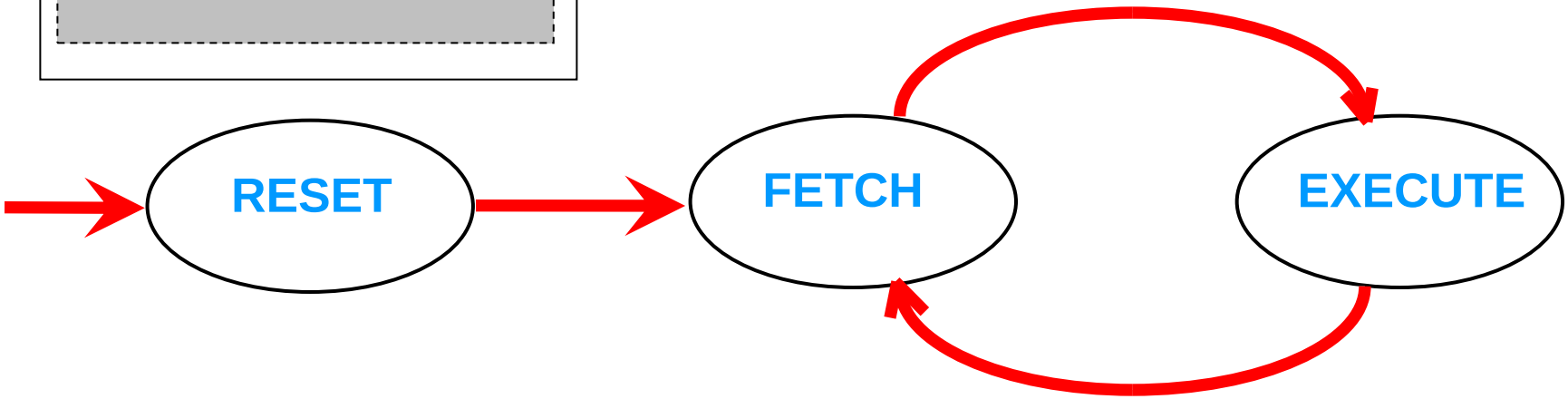
Processorns arbetssätt

Arb s 95



Maskinprogram i minnet / *Tillhörande assemblerprogram*

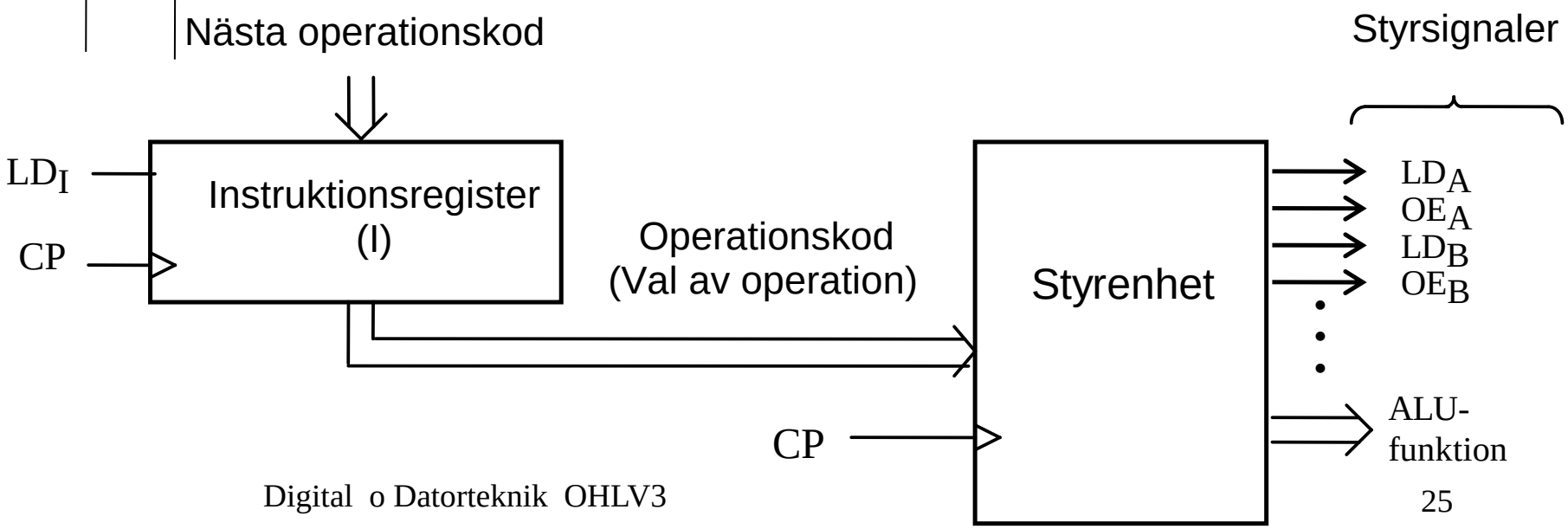
Adr.			
0C	10	LDB	#$23
0D	23		
0E	29	ADDB	$F3
0F	F3		
10	02	TFR	B,A
11	4F	CMPB	#$03
12	03		
13	61	BLO	$29
14	13		



Maskinprogram i minnet

Adr.			
0C	10	LDB	#$23
0D	23		
0E	29	ADDB	$F3
0F	F3		
10	02	TFR	B,A
11	4F	CMPB	#$03
12	03		
13	61	BLO	$29
14	13		

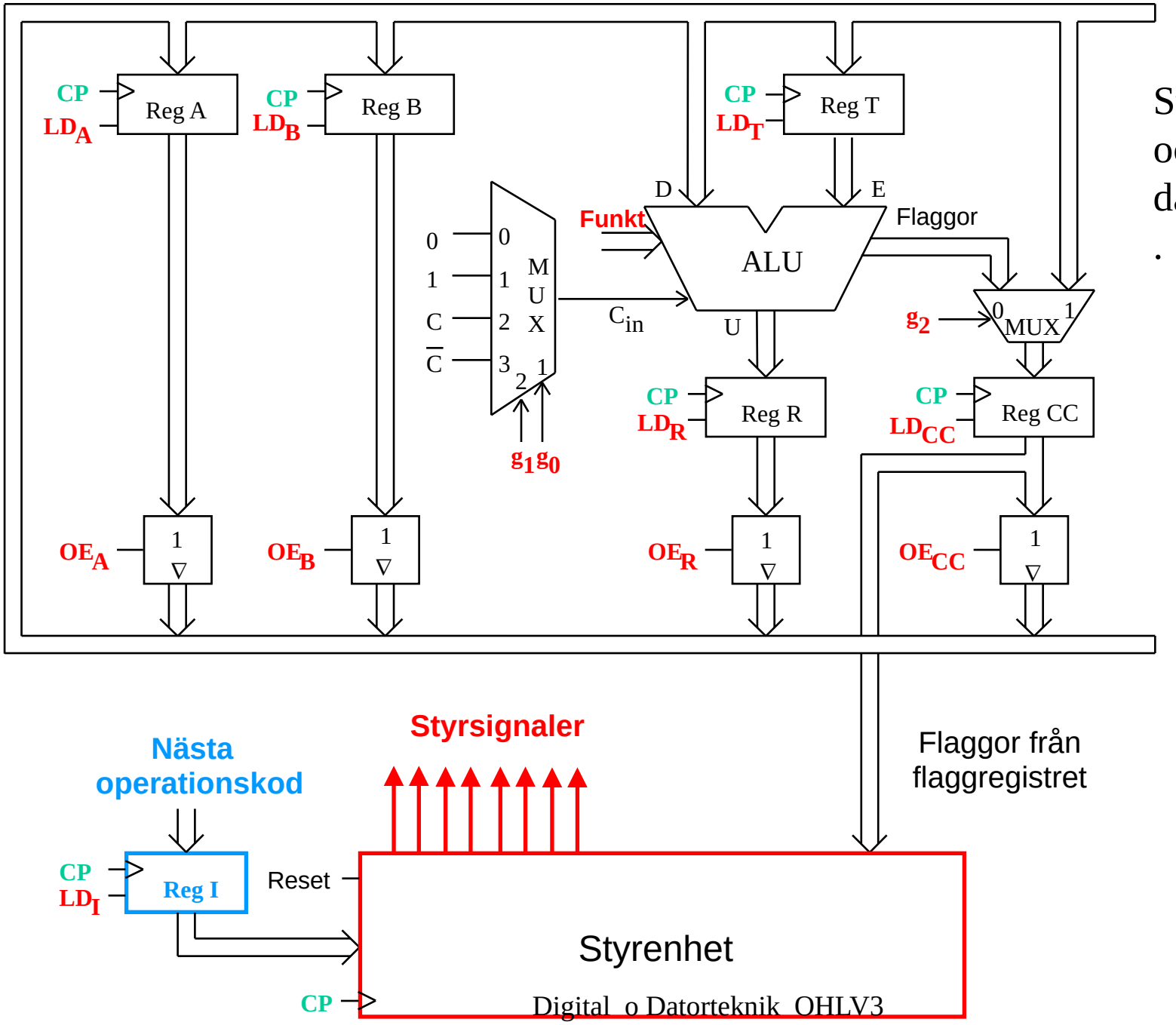
Styrsignalgenerering för den databehandlande enheten.



Kap 7 Blå

Styrenhet och del av dataväg

.



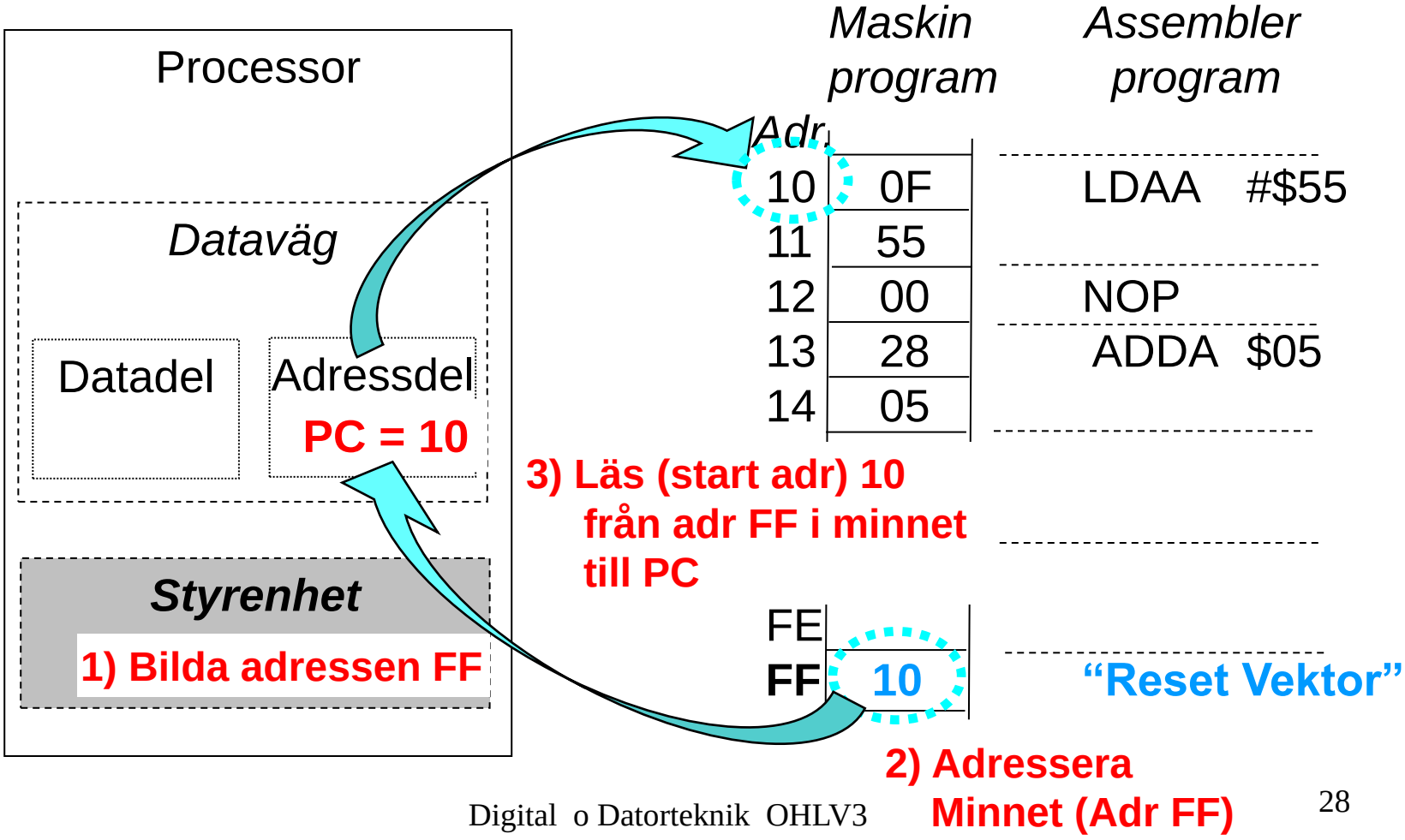
Dagens mål. Du ska kunna:

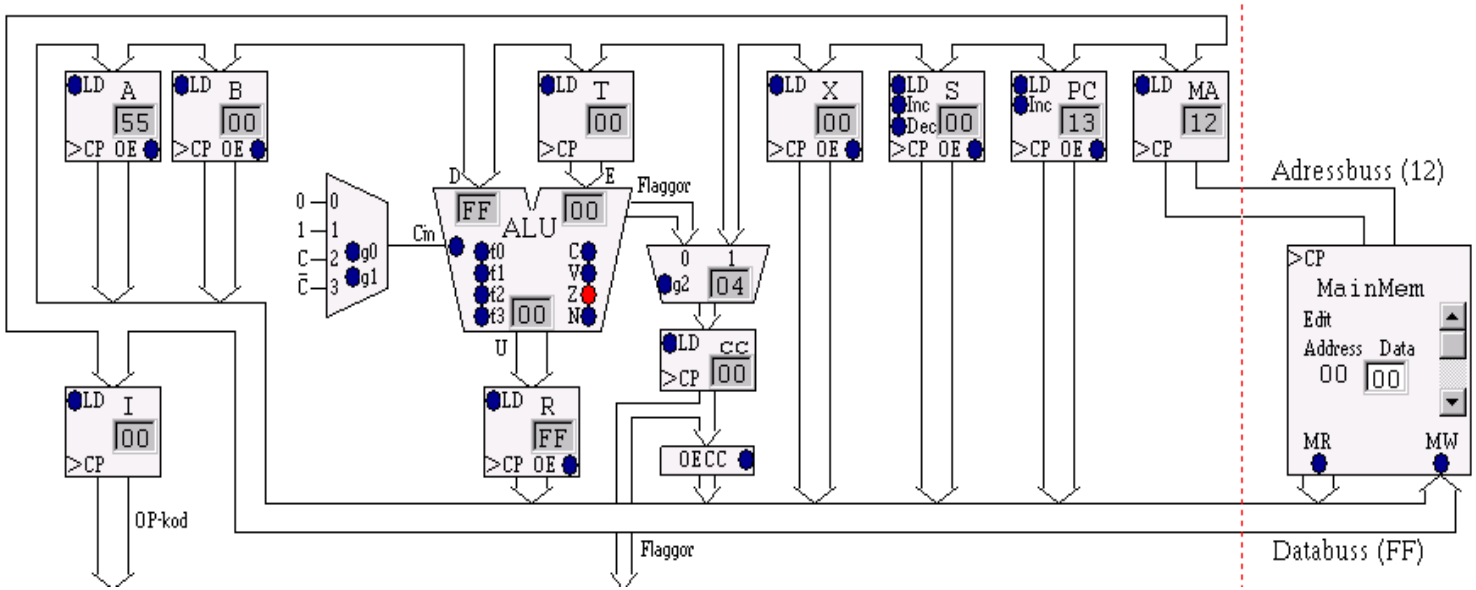
- Beskriva Processorns Arbetssätt
  - **Ange styrsignalsekvens för RESET**
  - Ange styrsignalsekvens för FETCH
  - Ange styrsignalsekvenser för olika EXECUTE-faser
- Konstruera och använda JK- och T-vippa
- Kunna analysera räknare
- Ta fram Excitationstabeller

Processorns arbetssätt – RESET

Arb s 95

PC: Programräknare
(Pekar på nästa instruktion)

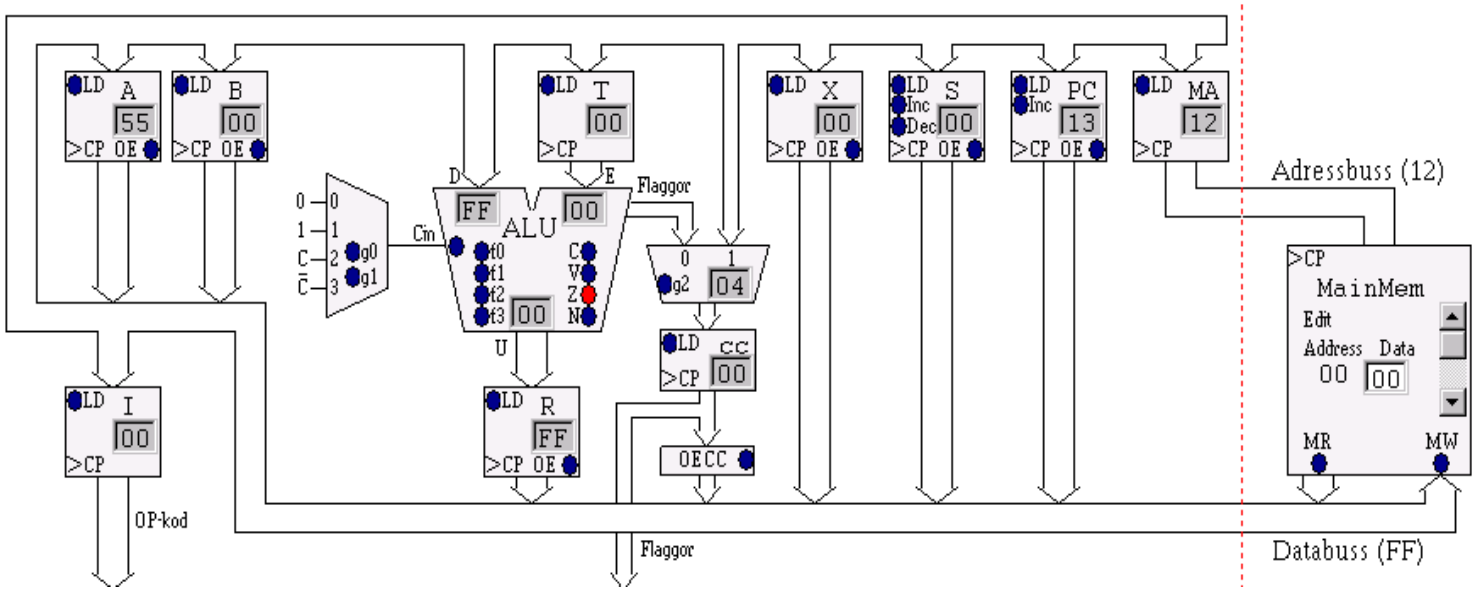




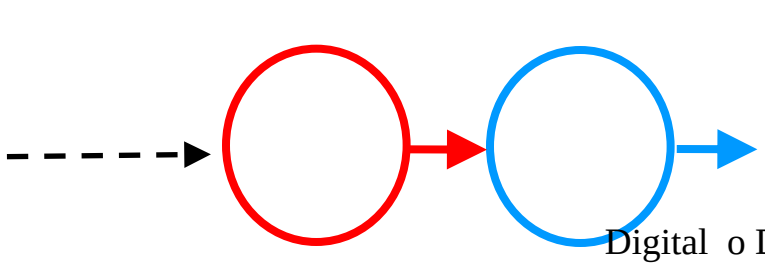
State	RTN-beskr	Styrsignaler	Kommentar
0	$FF_{16} \rightarrow R$	ALU-fkn = F_{16}, LD_R=1.	ALU-funktionen väljs så att talet FF_{16} finns på ALU:ns utgång. Laddingången på R-registret ettställs så att utvärdet från ALU'n (FF_{16}) laddas i R-registret vid nästa klockpuls.
1	R→MA	OE_R=1, LD_{MA}=1.	Talet FF_{16} i R-registret kopplas ut på bussen. Bussinnehållet laddas i minnesadressregistret vid nästa klockpuls.
2	M→PC	MR=1, LD_{PC}=1.	Minnesinnehållet på adressen FF_{16} läses. Det dataord som läses placeras i PC vid nästa klockpuls. Nästa klockcykel skall vara den första i FETCH-sekvensen.

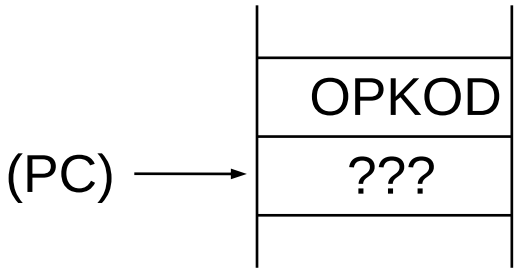
Dagens mål. Du ska kunna:

- Beskriva Processorns Arbetssätt
  - Ange styrsignalsekvens för RESET
  - **Ange styrsignalsekvens för FETCH**
  - Ange styrsignalsekvenser för olika EXECUTE-faser
- Konstruera och använda JK- och T-vippa
- Kunna analysera räknare
- Ta fram Excitationstabeller



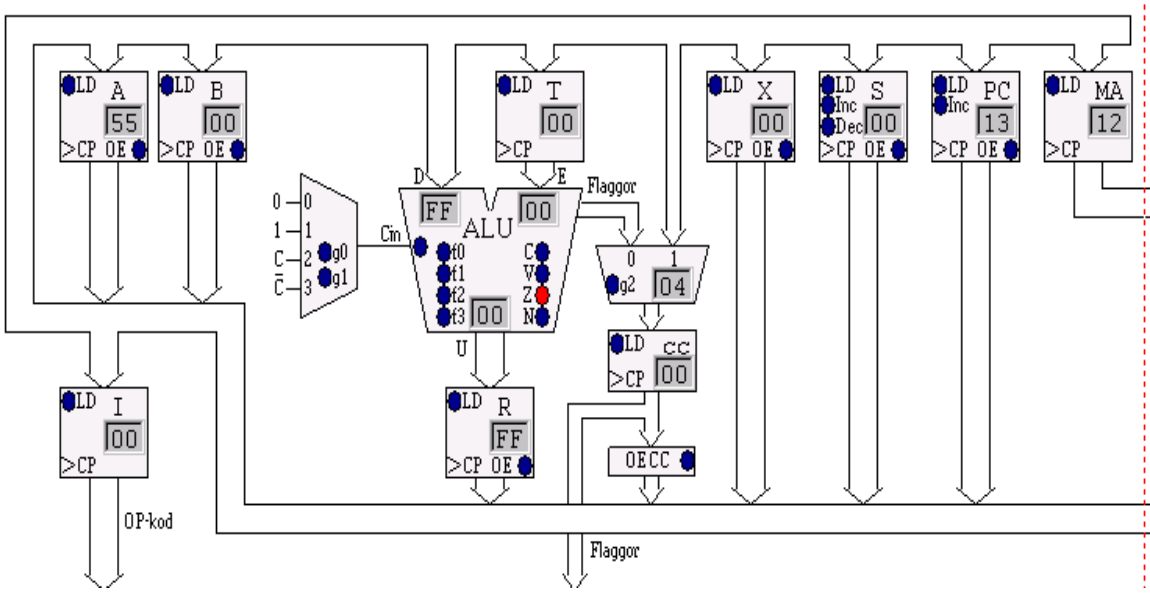
State	RTN	Styrsignaler	Kommentar
0	**PC→MA,** **PC+1→PC**	**OE_{PC}=1, LD_{MA}=1,** **IncPC=1.**	Adressen till instruktionens operationskod kopieras från PC till minnesadressregistret MA. Adressen som finns i PC ökas med ett.
1	**M→I**	**MR=1,** **LD_I=1.**	Läs operationskoden från minnet. Placera den i instruktionsregistret I. Nästa state skall vara det första i EXECUTE-sekvensen.

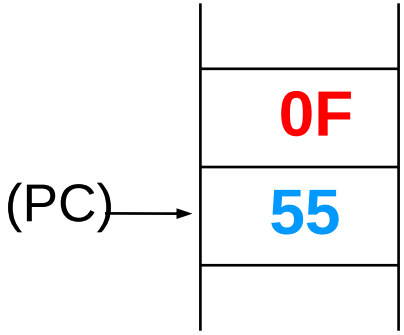




Dagens mål. Du ska kunna:

- Beskriva Processorns Arbetssätt
  - Ange styrsignalsekvens för RESET
  - Ange styrsignalsekvens för FETCH
  - **Ange styrsignalsekvenser för olika EXECUTE-faser**
- Konstruera och använda JK- och T-vippa
- Kunna analysera räknare
- Ta fram Excitationstabeller





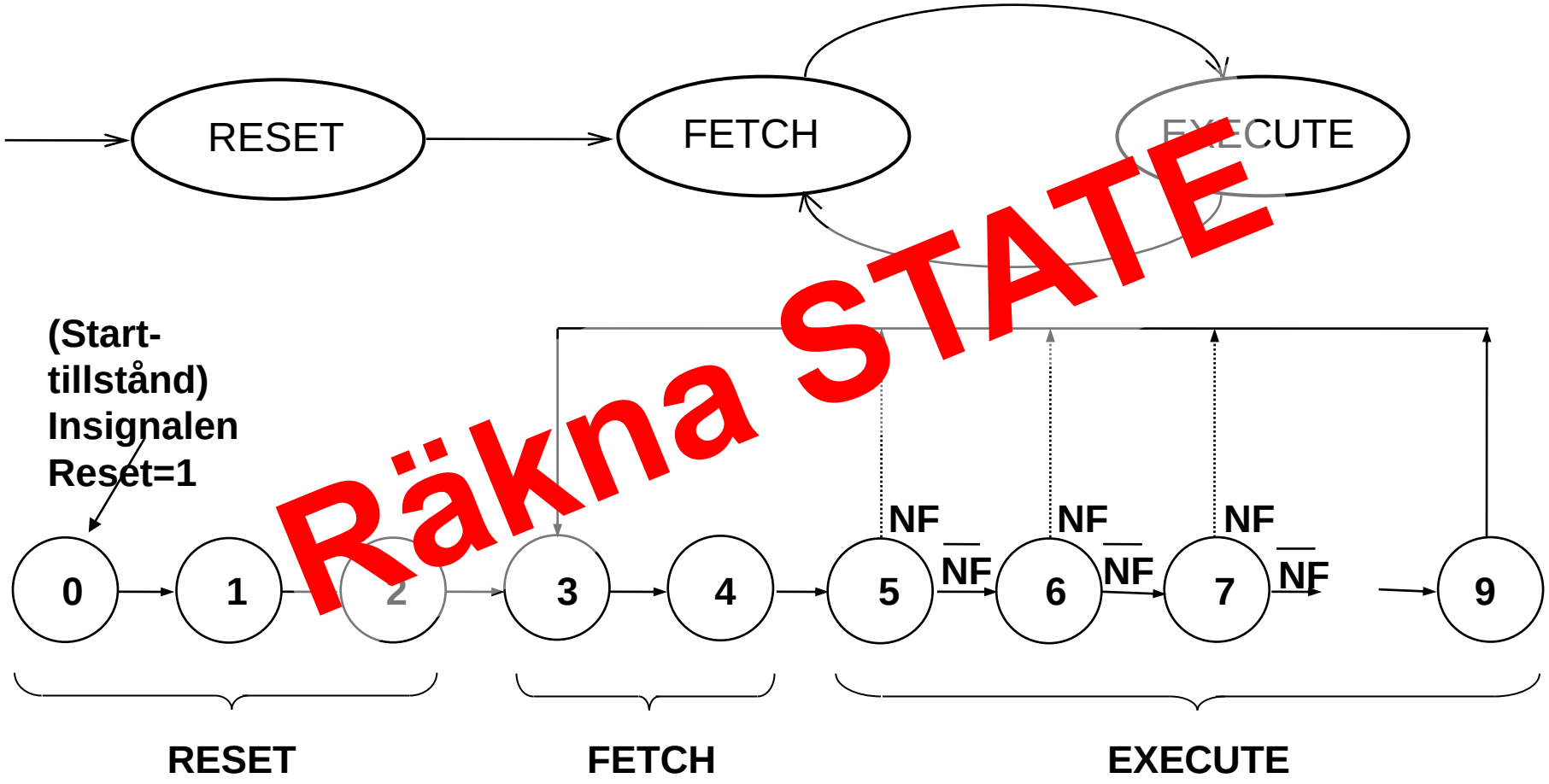
Instruktion: *Format:*

LDAA #Data Ord1: OP-kod

(**LDAA #$55**) Ord2: Data

State	RTN	Styrsignaler	Kommentar
0	PC→MA, PC+1→PC	OE_{PC}=1, LD_{MA}=1, IncPC=1.	Instruktionens datadel finns i minnesordet efter OP-koden. När EXECUTE-sekvensen inleds pekar PC på instruktionens datadel. PC kopieras därför över till minnesadressregistret MA så att datadelen kan läsas från minnet under nästa klockcykel. Innehållet i PC ökas med ett, så att PC pekar på nästa adress i minnet där OP-koden för nästa instruktion skall finnas.
1	M→A	MR=1, LD_A=1	Läs instruktionens datadel "Data" från minnet och placera den i A-registret. Nästa klockcykel skall vara den första i FETCH-sekvensen.

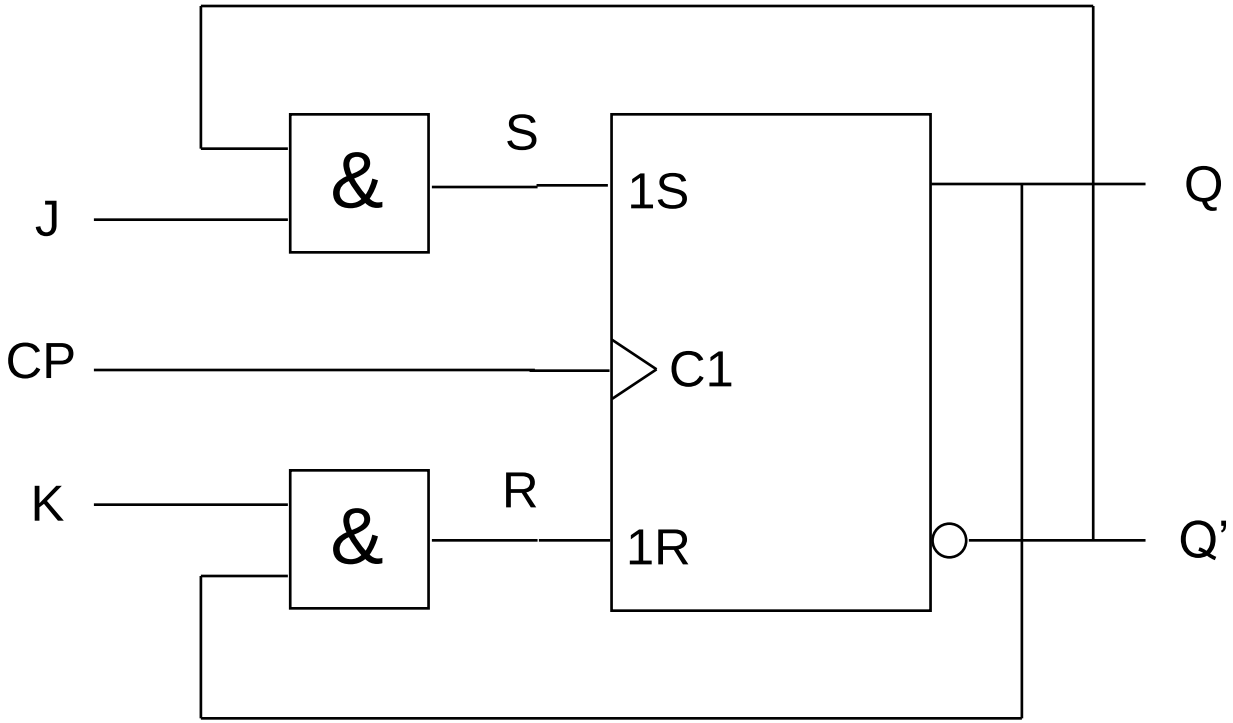
Processorns arbetssätt



Dagens mål. Du ska kunna:

- Beskriva Processorns Arbetssätt
  - Ange styrsignalsekvens för RESET
  - Ange styrsignalsekvens för FETCH
  - Ange styrsignalsekvenser för olika EXECUTE-faser
- **Konstruera och använda JK- och T-vippa**
- Kunna analysera räknare
- Ta fram Excitationstabeller

Principen för JK-vippa:

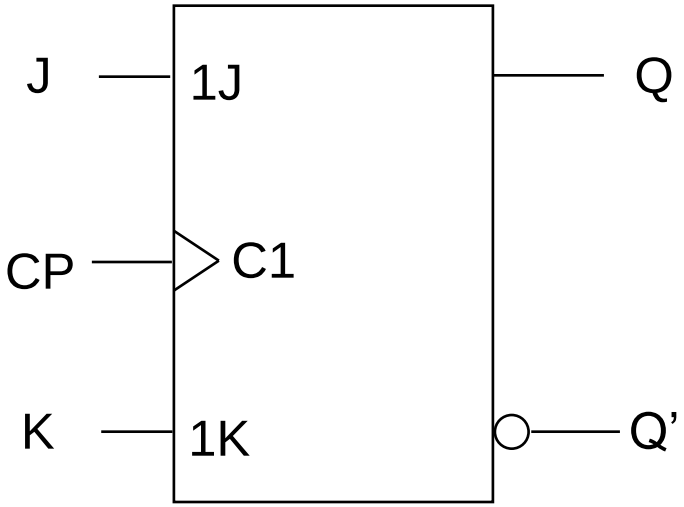


S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	**Otillåtet**

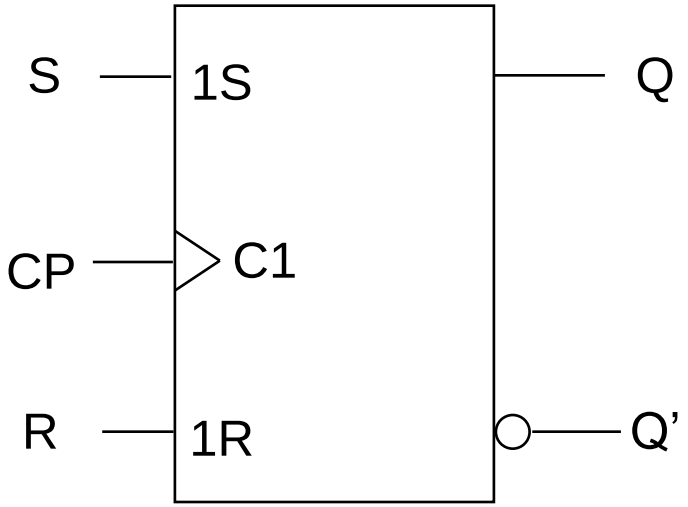
J	K	S	R	Q^+
0	0	0	0	
0	1	0	Q	
1	0	Q'	0	
1	1	Q'	Q	

JK - Vippan

Jfr med SR-vippan

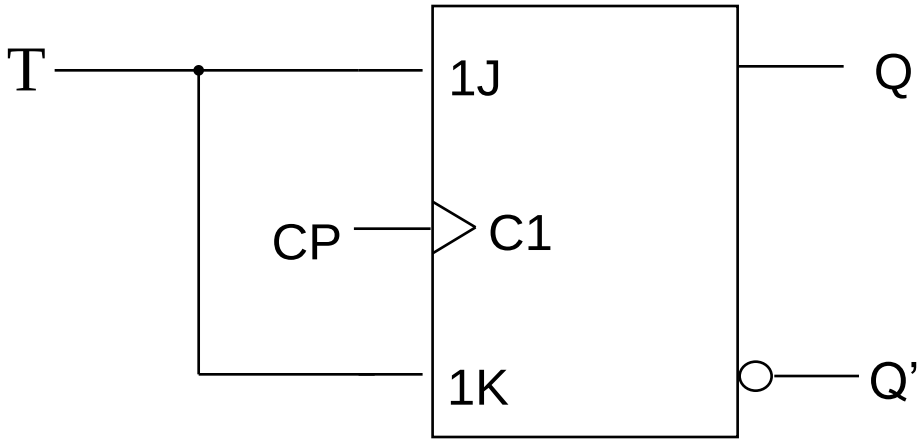


J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	**Q'**



S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	**Otillåtet**

T - Vippan



Funktionstabell

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Q'

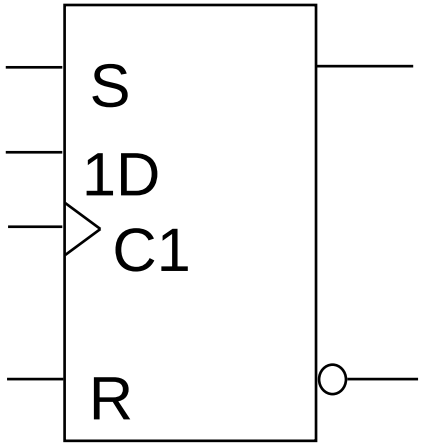
Funktionstabell

T	Q^+
0	Q
1	Q'

Asynkrona ingångar för ettställning och nollställning

Vippor förses ofta med extra ingångar som påverka utsignalen **oberoende** av klockpulsen och övriga insignaler.
Dessa ingångar kallas därför för **asynkrona ingångar**.
För **ett-** (eng **Preset**) och **nollställning** (eng **Clear**) av utsignalen.

Benämns S - R på samma sätt som S- och R-signalerna hos en SR-latch.

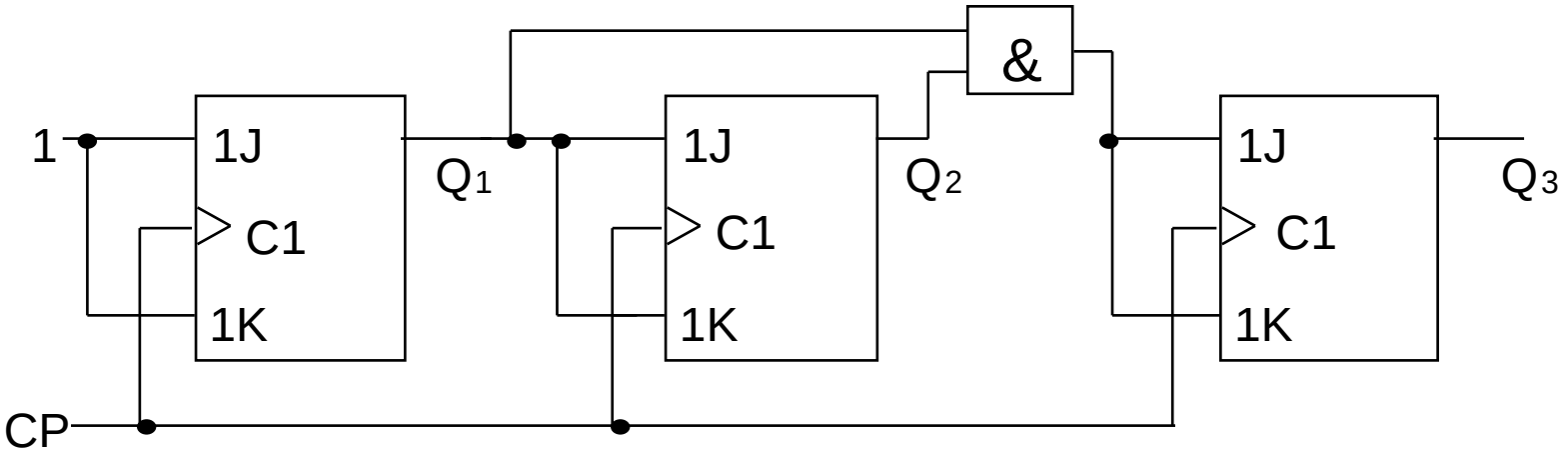


D-vippan har asynkrona S- och R-ingångar.

Dagens mål. Du ska kunna:

- Beskriva Processorns Arbetssätt
  - Ange styrsignalsekvens för RESET
  - Ange styrsignalsekvens för FETCH
  - Ange styrsignalsekvenser för olika EXECUTE-faser
- Konstruera och använda JK- och T-vippa
- **Kunna analysera räknare**
- Ta fram Excitationstabeller

Arbetsgång - analys räknare s5.24 ->



1 Studera kopplingen och **bestäm vippornas insignaler** (T_1=, T_2=, T_3=)

2 Sätt upp en tabell med

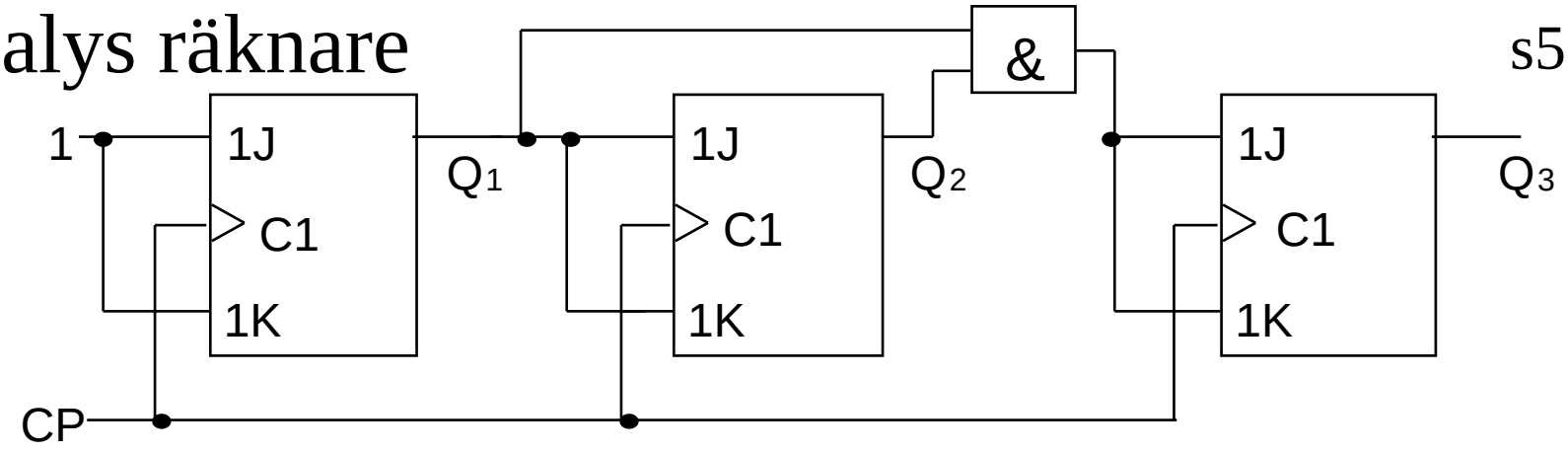
- ”**Detta tillstånd**” (Alla kombinationer av Q_1, Q_2, Q_3)
- **Insignaler** (T_1, T_2, T_3)
- ”**Nästa tillstånd**” (Q_1^+, Q_2^+, Q_3^+)

3 Ange insignalernas värden i tabellen och----

4 ange vad ”nästa tillstånd” blir

5 Rita slutligen en **tillståndsgraf**

Analys räknare



1)

$T_1 = 1$

$T_2 = Q_1$

$T_3 = Q_1 Q_2$

Funktionstabell

T	Q^+
0	Q
1	Q'

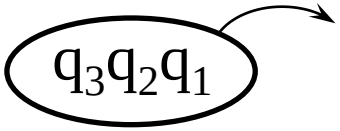
2)

Detta Tillstånd			Insignaler			Nästa Tillstånd		
Q_3	Q_2	Q_1	T_3	T_2	T_1	Q_3^+	Q_2^+	Q_1^+
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	0
0	1	0	0	0	1	0	1	1
0	1	1	1	1	1	1	0	0
1	0	0	0	0	1	1	0	1
1	0	1	0	1	1	1	1	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

3) **4)**

Analys räknare

Tillståndsdiagram:

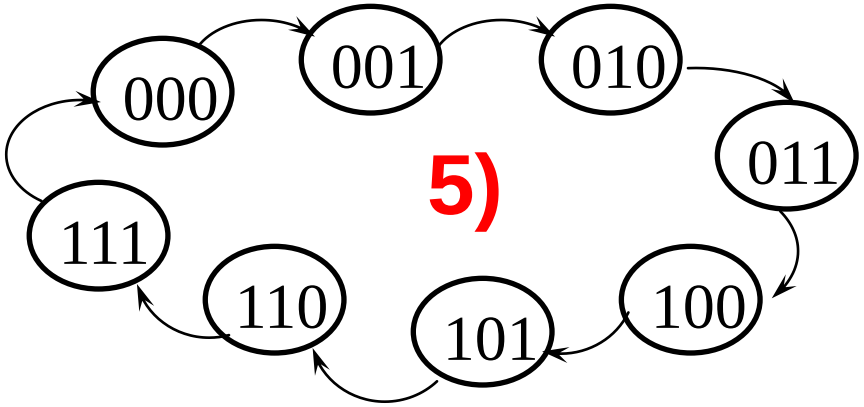


Fördelar:

Vi ser funktionen.

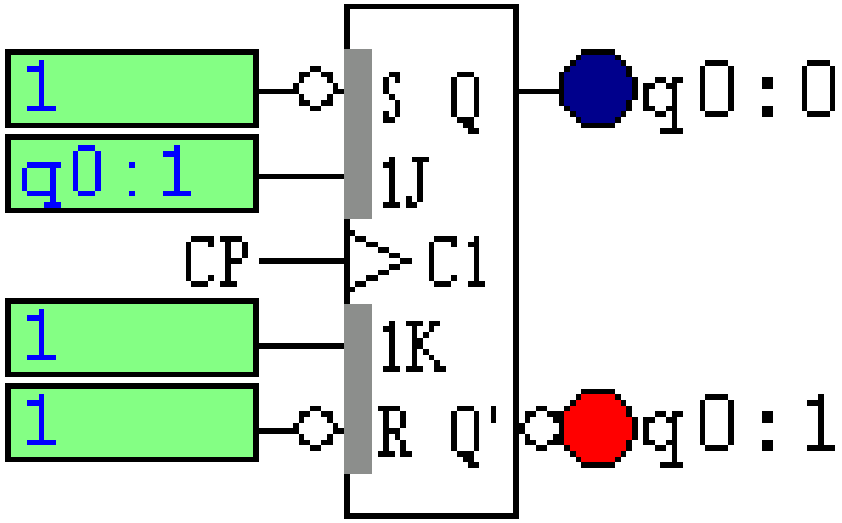
Vi upptäcker om vi har gjort fel!

Q_3	Q_2	Q_1	Q_3^+	Q_2^+	Q_1^+
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0



Uppgift 88

Välj en JK-vippa och anslut Q'-utgången till J-ingången.
Ettställ de övriga ingångarna.
Ge ett antal klockpulser och observera att utgångarna



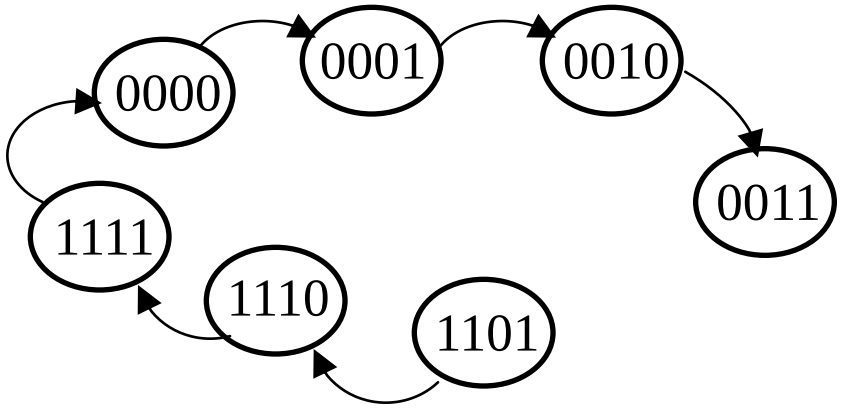
Gör Uppgift 88

Dagens mål. Du ska kunna:

- Beskriva Processorns Arbetssätt
  - Ange styrsignalsekvens för RESET
  - Ange styrsignalsekvens för FETCH
  - Ange styrsignalsekvenser för olika EXECUTE-faser
- Konstruera och använda JK- och T-vippa
- Kunna analysera räknare
- **Ta fram Excitationstabeller**

Syntes Räknare

Konstruera en räknare som räknar sekvensen
0-1-2-3-…-14-15-0-1 osv.



Vi vet ”Detta tillstånd”

Vi vet ”Nästa tillstånd”

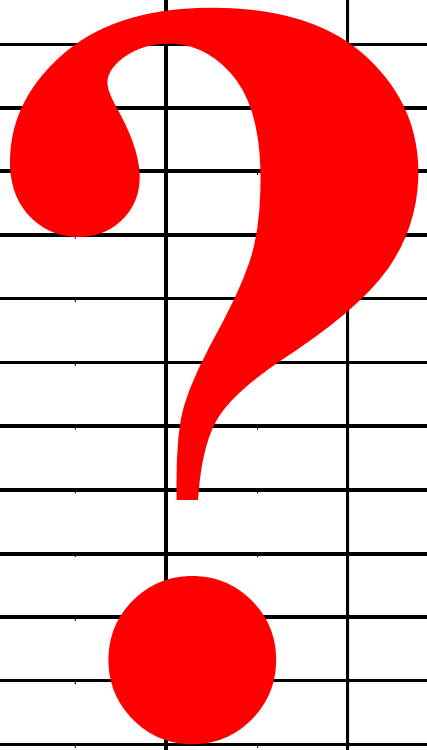
VAD SKA VIPPORNAS INSIGNALER VARA
FÖR ATT HAMNA I ”NÄSTA TILLSTÅND”????

⇒ NY TABELL

Utsignaler								Insignaler							
Detta tillstånd Q				Nästa tillstånd Q^+				J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
q3	q2	q1	q0	$q3^+$	$q2^+$	$q1^+$	$q0^+$								
0	0	0	0												
0	0	0	1												
0	0	1	0												
0	0	1	1												
0	1	0													
0	1														
0	1														
0			1												
			0												
		0	1												
	0	1	0												
1	0	1	1												
1	1	0	0												
1	1	0	1												
1	1	1	0												
1	1	1	1												

Alla kombinationer Av ”Detta Tillstånd”

Fyll i ”Nästa Tillstånd”



Excitationstabeller - *forts*

Kmp s 5.18

Vad blir utgången Q^+ om insignalen är…..

Funktionstabell

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Q′

Vad skall ingången vara om tillståndsändringen i $Q \rightarrow Q^+$ är…..

Excitationstabell

Q	Q^+	J	K
0	0		
0	1		
1	0		
1	1		

Veckans mål:

- Koppla samman register och ALU till en dataväg
- Förstå hur minne är uppbyggd, ansluta detta till datavägen
- Program och hur detta lagras i minne
- Fatta hur datorn startar och arbetar
- Räknare och mera vippor

Dagens mål. Du ska kunna:

- Konstruera räknare
- Använda räknaren 74HC163
- Konstruera en Fast kopplat styrenhet till FLEX
  - Implementera RESET-fasen i FLEX
  - Implementera FETCH-fasen i FLEX
  - Implementera olika EXECUTE-faser i FLEX

Utsignaler								Insignaler							
Detta tillstånd Q				Nästa tillstånd Q^+				J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
q3	q2	q1	q0	$q3^+$	$q2^+$	$q1^+$	$q0^+$								
0	0	0	0												
0	0	0	1												
0	0	1	0												
0	0	1	1												
0	1	0	0												
0	1	0	1												
0	1	1	0												
0	1	1	1												
1	0	0	0												
1	0	0	1												
1	0	1	0												
1	0	1	1												
1	1	0	0												
1	1	0	1												
1	1	1	0												
1	1	1	1												

Alla kombinationer Av ”Detta Tillstånd”

Vi vet alla ”Nästa Tillstånd”

Fyll i Vipparnas Insignaler

Arbetsgång - syntes räknare

Konstruera en räknare som räknar sekvensen ????

1 Rita en tillståndsgraf

2 Sätt upp en tabell med:

- ”Detta tillstånd” (Alla kombinationer av Q_1, Q_2, Q_3)
- ”Nästa tillstånd” (Q_1^+, Q_2^+, Q_3^+)
- Vippornas Insignaler

3 Ange ”Nästa tillstånd” i tabellen

4 Använd vippornas excitationstabell och ange vippornas insignaler

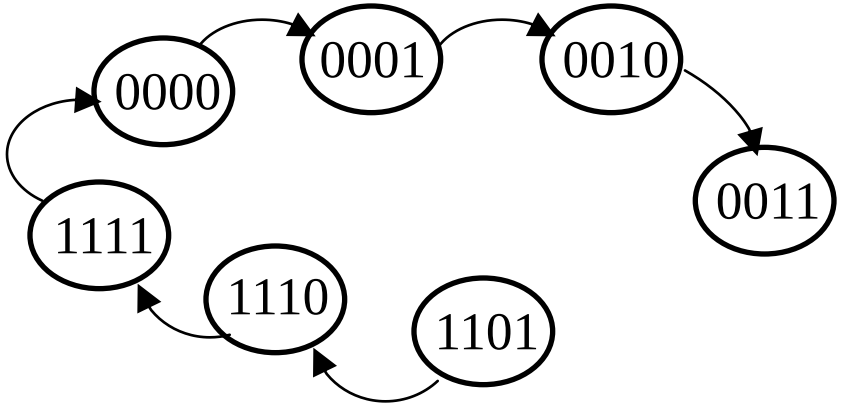
5 Minimera uttrycken för insignalerna

6 Realisera räknaren

Uppgift 93

Konstruera en räknare som räknar sekvensen
0-1-2-3-…-14-15-0-1 osv.

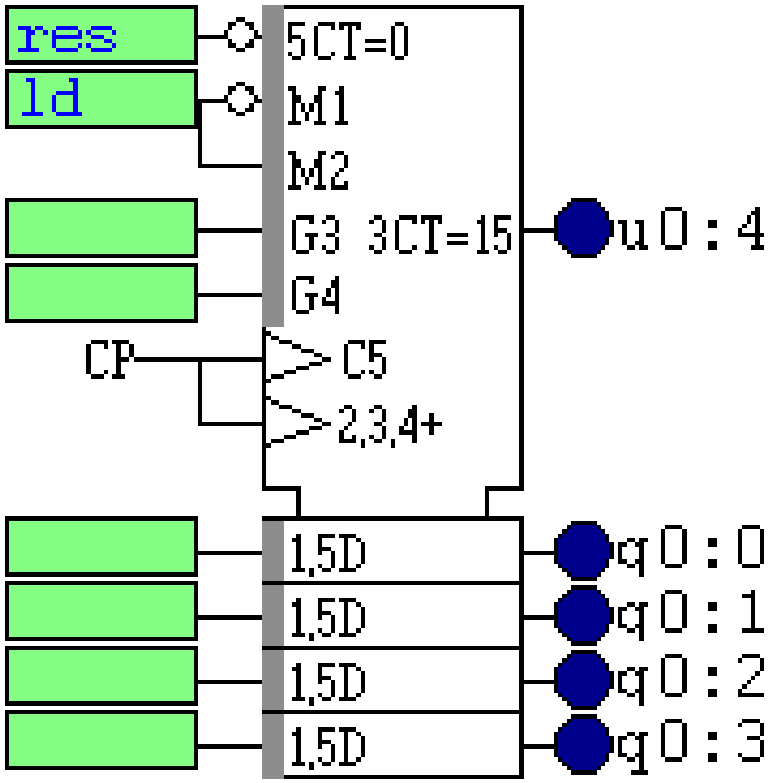
Räknaren skall realiseras med JK-vippor grindar tillgängliga i Kopplingsboxen



Dagens mål. Du ska kunna:

- Konstruera räknare
- **Använda räknaren 74HC163**
- Konstruera en Fast kopplat styrenhet till FLEX
  - Implementera RESET-fasen i FLEX
  - Implementera FETCH-fasen i FLEX
  - Implementera olika EXECUTE-faser i FLEX

Uppgift 96 -Räknaren 74HC163



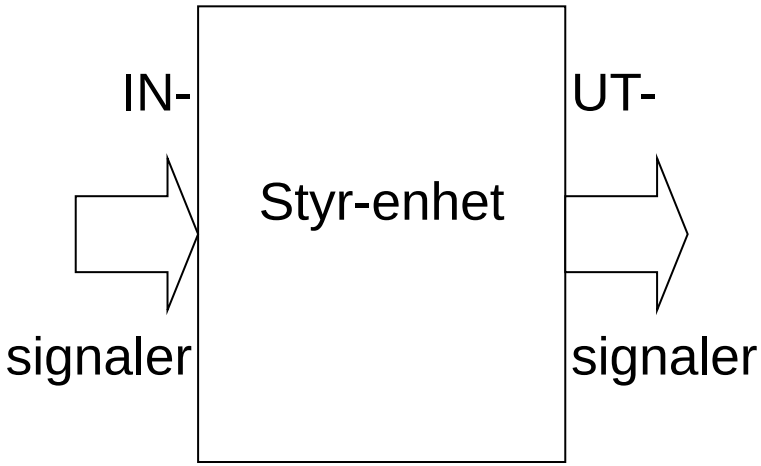
Grönt = etta

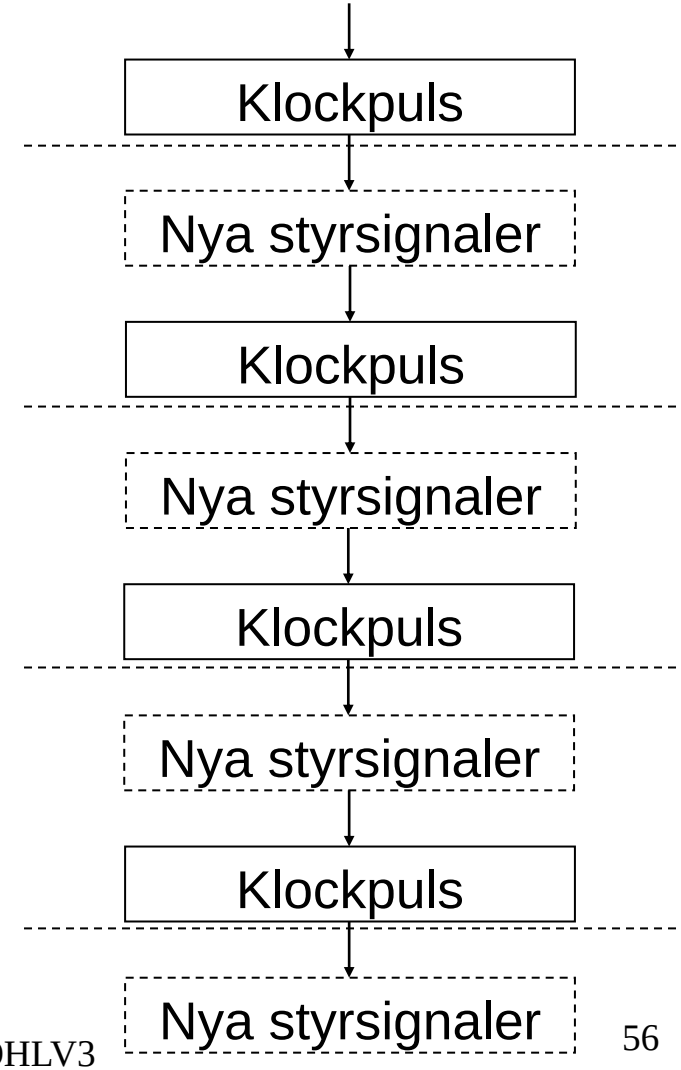
Gör uppgift 96

Dagens mål. Du ska kunna:

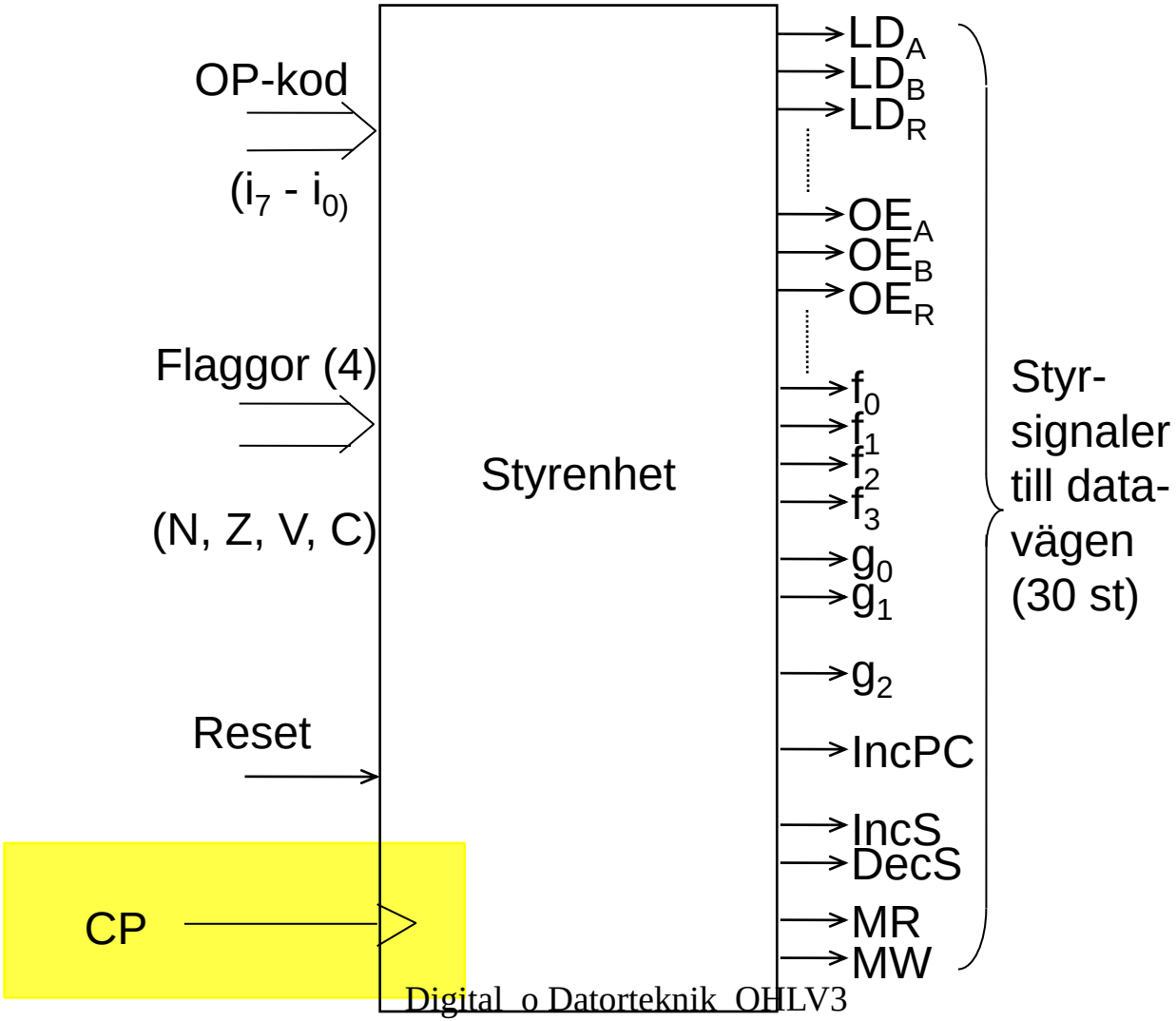
- Konstruera räknare
- Använda räknaren 74HC163
- **Konstruera en Fast kopplat styrenhet till FLEX**
  - Implementera RESET-fasen i FLEX
  - Implementera FETCH-fasen i FLEX
  - Implementera olika EXECUTE-faser i FLEX

Styrenheten



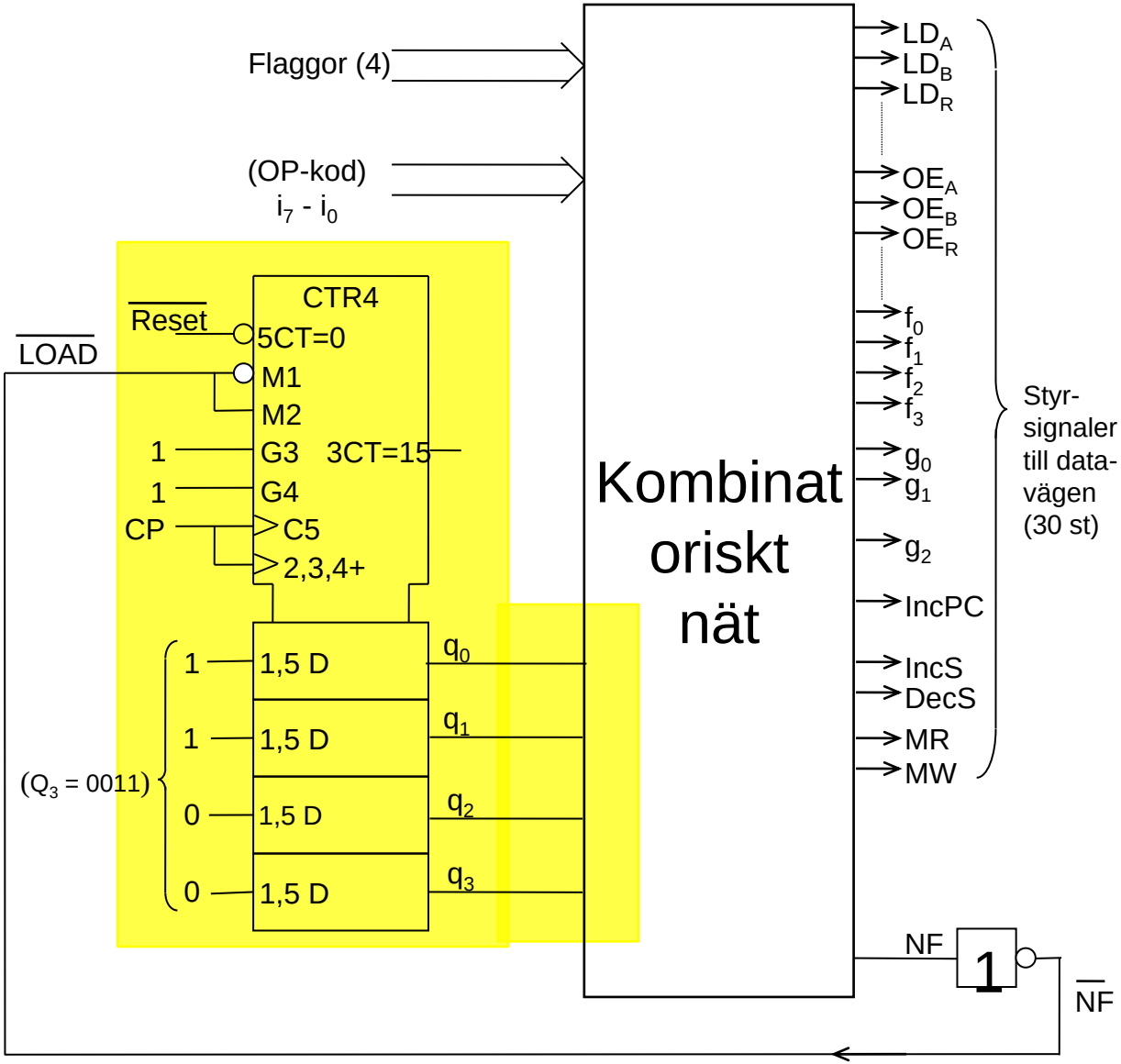


Styrenheten - forts



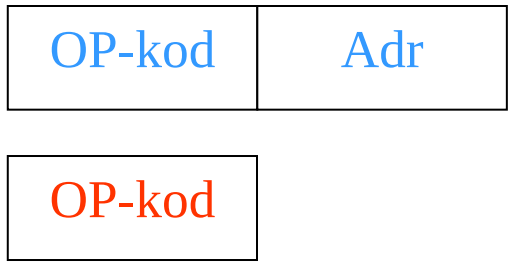
Styrenheten - forts

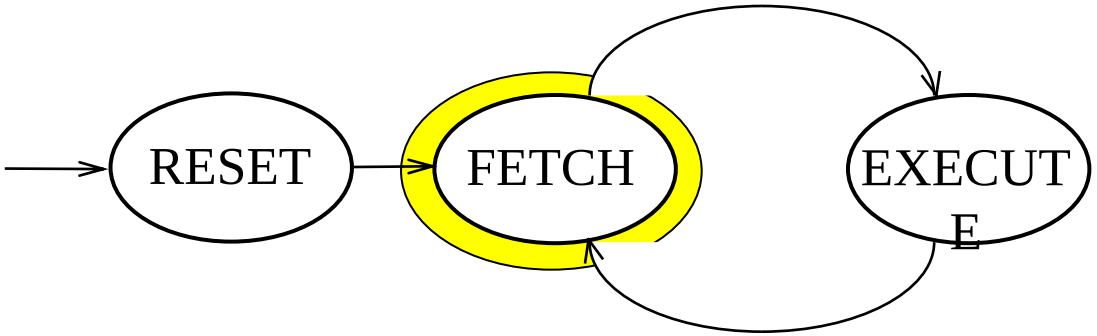
Arb s 123

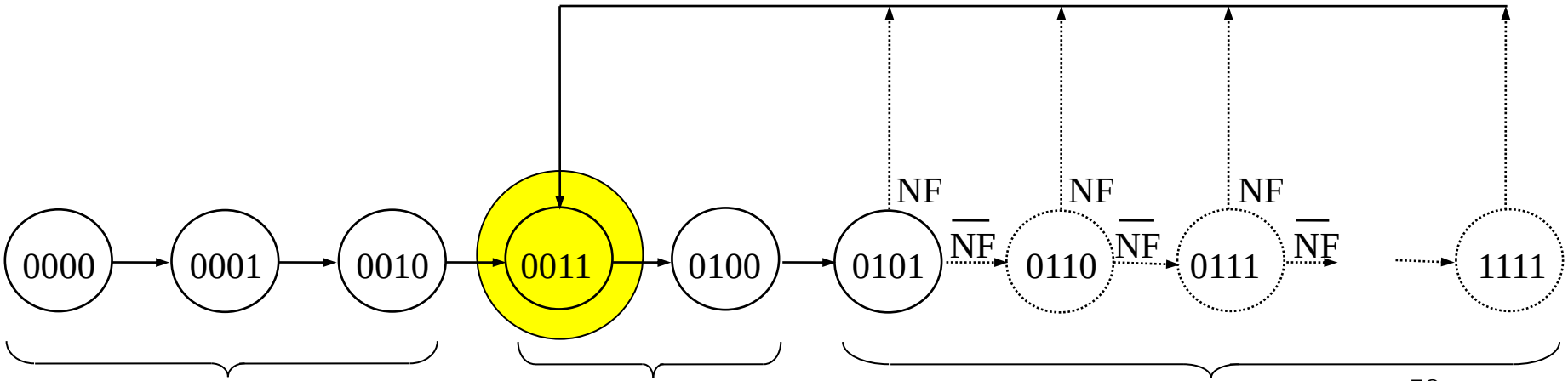


Mask. prog i minnet Adr.		*Tillhörande assemblerprog*	
0C	10	LDAB	#$23
0D	23		
0E	29	ADDB	$F3
0F	F3		
10	02	TFR	B,A
11	4F	CMPB	#$03
12	03		
13	61	BLO	$29
14	13		

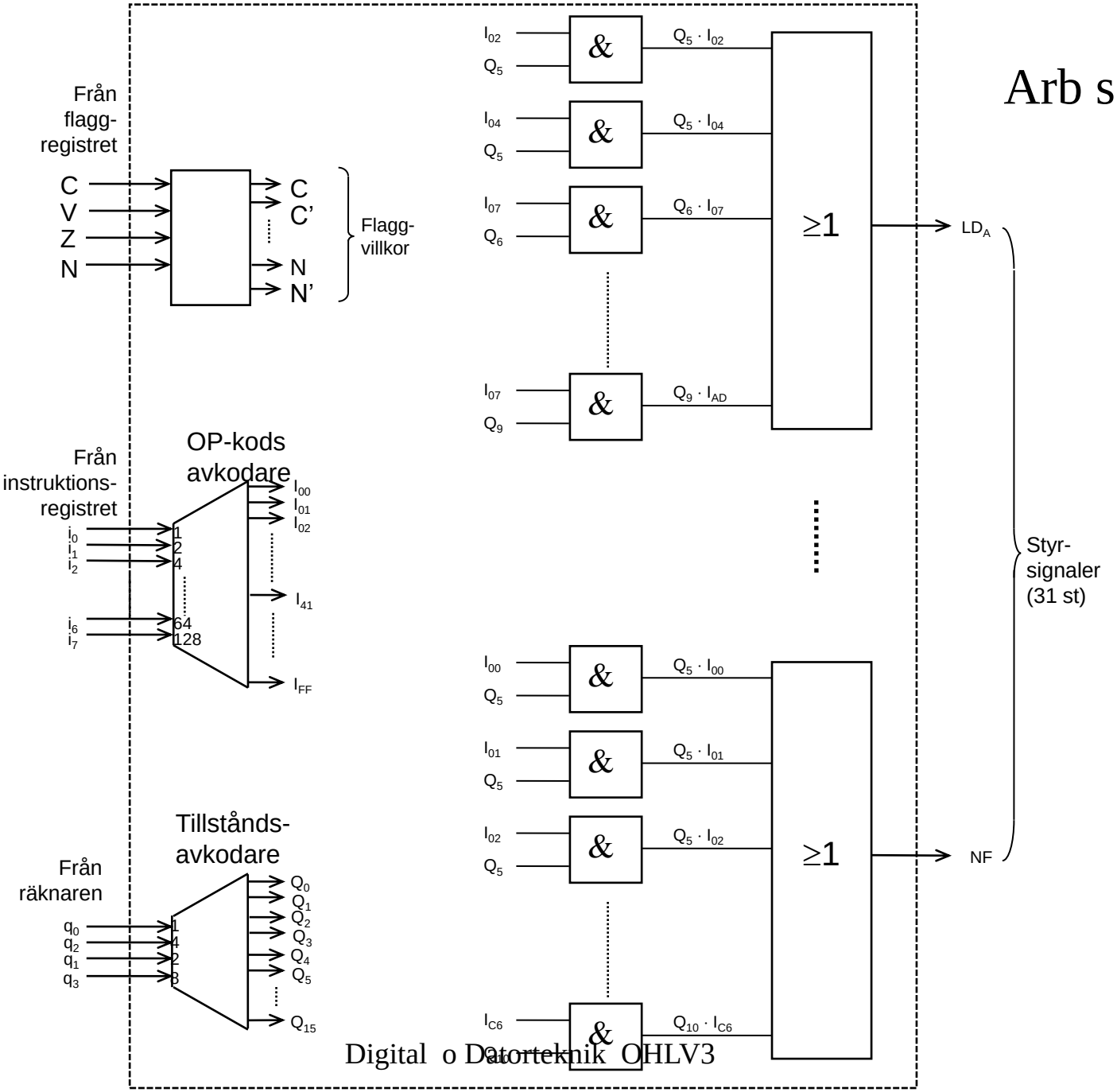
Aktivera





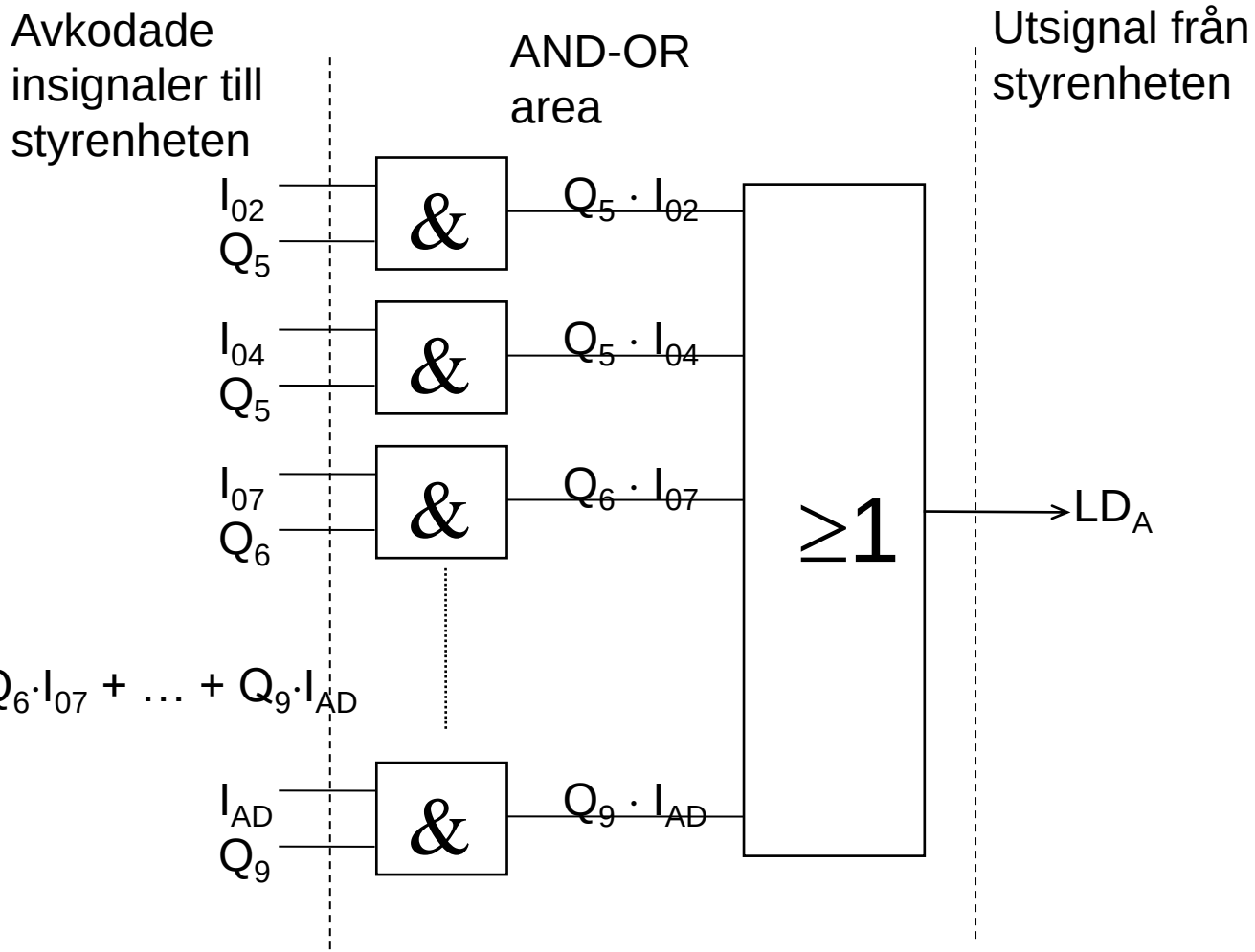


Arb s 124


Från flagg-registret
C
V
Z
N
C
C'
N
N'
Flagg-villkor
OP-kods avkodare
Från instruktions-registret
i_0
i_1
i_2
i_6
i_7
1
2
4
64
128
I_{00}
I_{01}
I_{02}
I_{41}
I_{FF}
Tillstånds-avkodare
Från räknaren
q_0
q_2
q_1
q_3
1
4
2
8
Q_0
Q_1
Q_2
Q_3
Q_4
Q_5
Q_{15}
I_{02} Q_5 & $Q_5 \cdot I_{02}$
I_{04} Q_5 & $Q_5 \cdot I_{04}$
I_{07} Q_6 & $Q_6 \cdot I_{07}$
I_{07} Q_9 & $Q_9 \cdot I_{AD}$
≥1
LD_A
I_{00} Q_5 & $Q_5 \cdot I_{00}$
I_{01} Q_5 & $Q_5 \cdot I_{01}$
I_{02} Q_5 & $Q_5 \cdot I_{02}$
I_{C6} Q_{10} & $Q_{10} \cdot I_{C6}$
≥1
NF
Styr-signaler (31 st)

Styrenheten - forts

Arb s 125



Booleskt uttryck:

$LD_A = Q_5 \cdot I_{02} + Q_5 \cdot I_{04} + Q_6 \cdot I_{07} + \ldots + Q_9 \cdot I_{AD}$

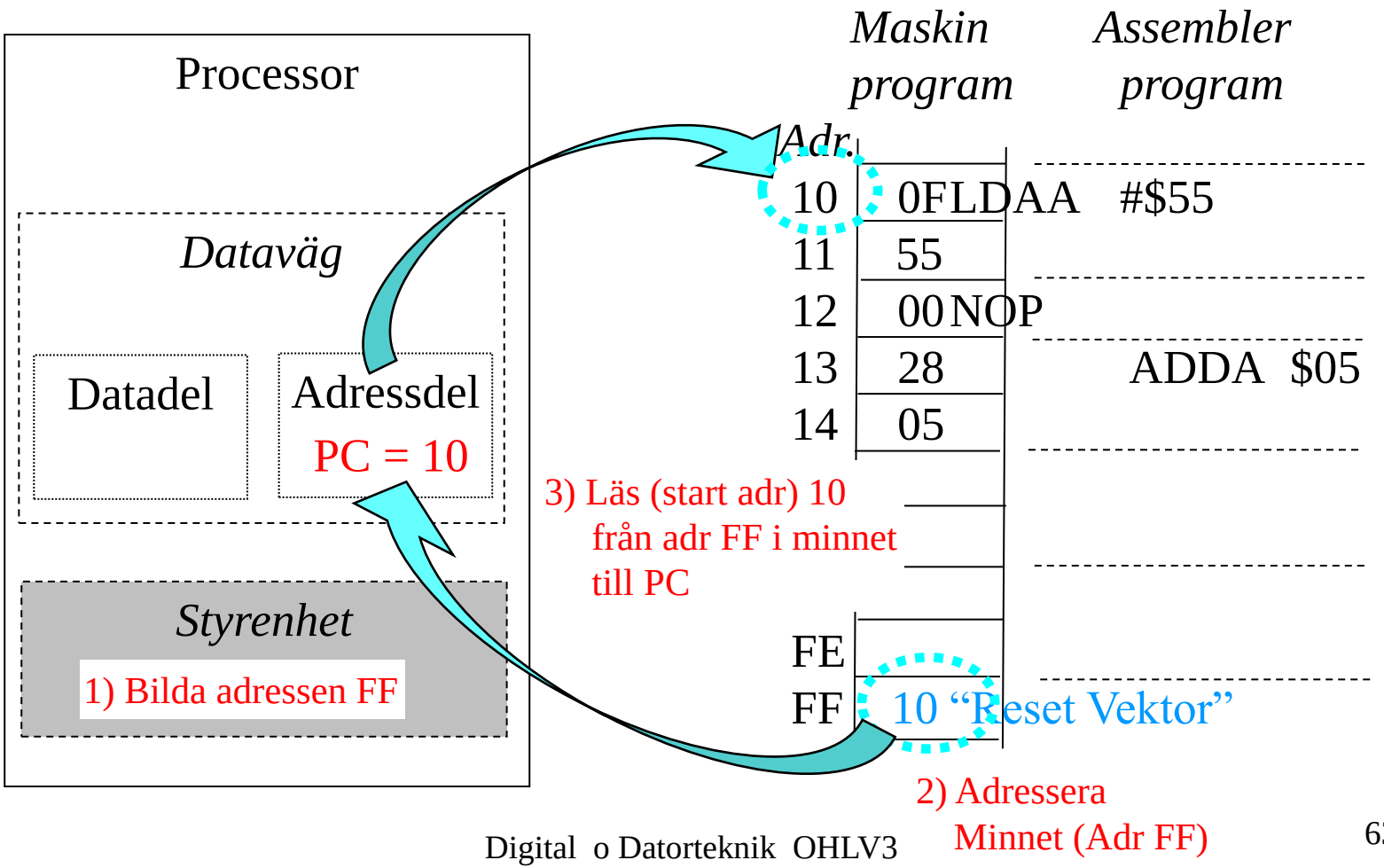
Dagens mål. Du ska kunna:

- Konstruera räknare
- Använda räknaren 74HC163
- Konstruera en Fast kopplat styrenhet till FLEX
  - **Implementera RESET-fasen i FLEX**
  - Implementera FETCH-fasen i FLEX
  - Implementera olika EXECUTE-faser i FLEX

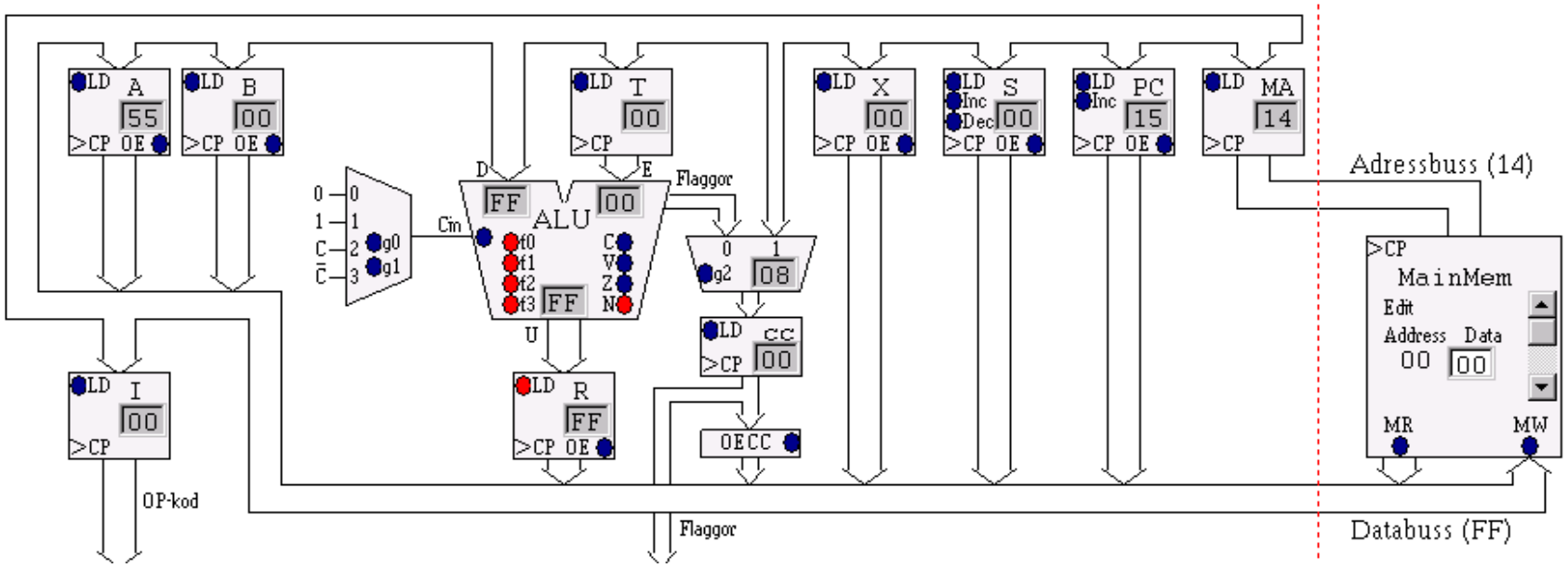
Processorns arbetssätt – RESET

Arb s 95

PC: Programräknare
(Pekar på nästa instruktion)

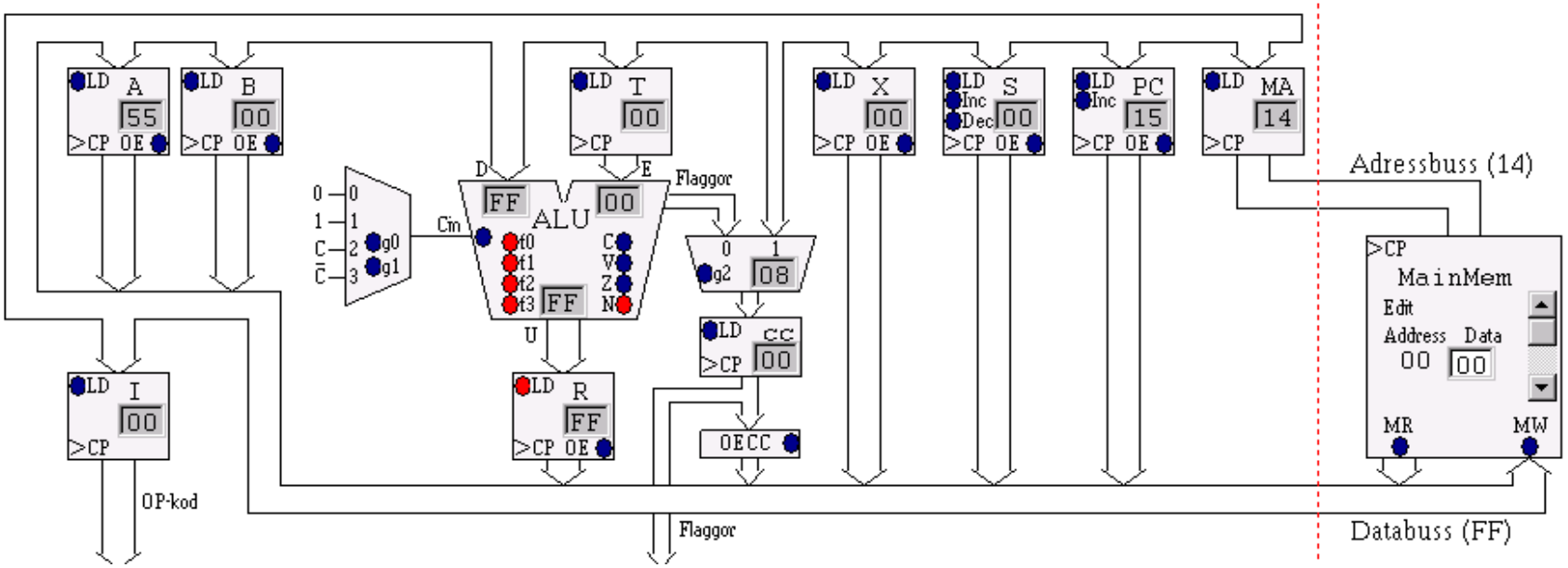


Processorns arbetssätt - RESET

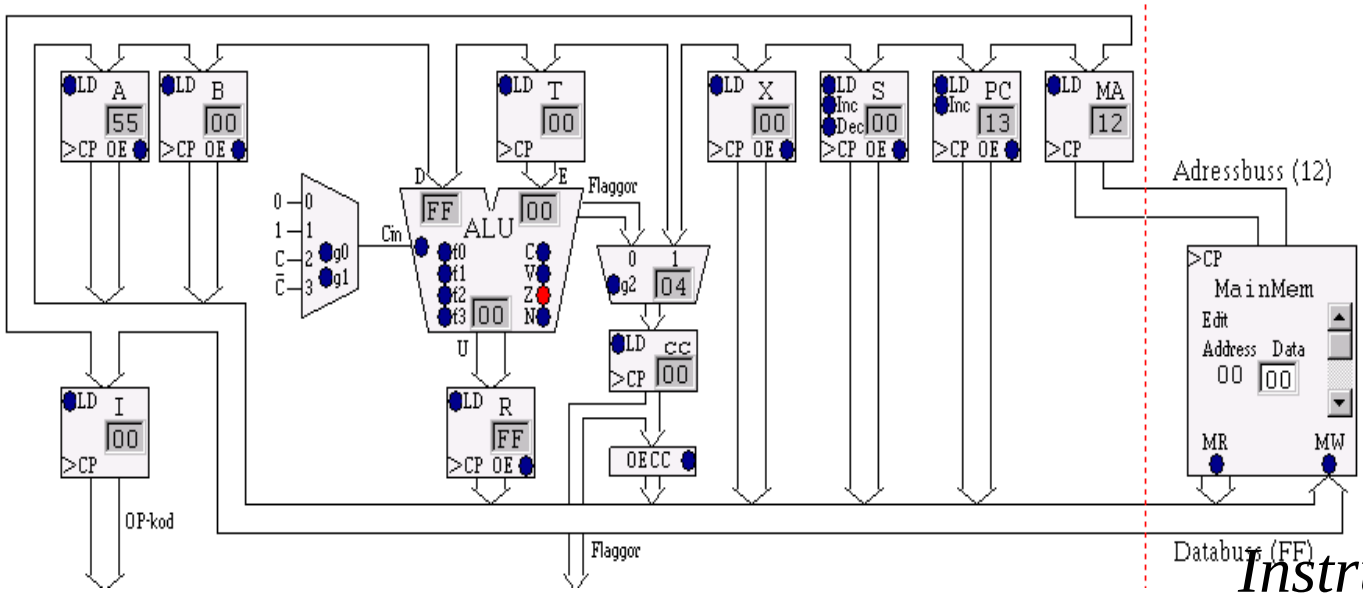


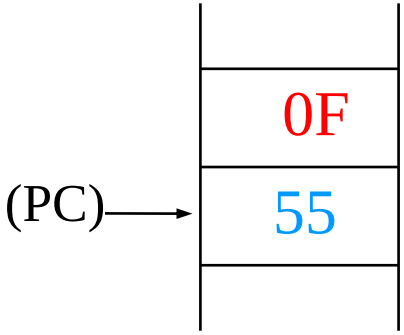
Tillstånd	Summaterm	RTN-beskrivning	Styrsignaler (=1)
Q_0	Q_0	$FF_{16} \rightarrow R$	f_3, f_2, f_1, f_0, LD_R
Q_1	Q_1	$R \rightarrow MA$	OE_R, LD_{MA}
Q_2	Q_2	$M \rightarrow PC$	MR, LD_{PC}

Processorns arbetssätt - FETCH



Tillstånd	Summaterm	RTN-beskrivning	Styrsignaler (=1)
Q_3	Q_3	PC→MA PC+1→PC	OE_{PC}, LD_{MA} IncPC
Q_4	Q_4	M→I	MR, LD_I



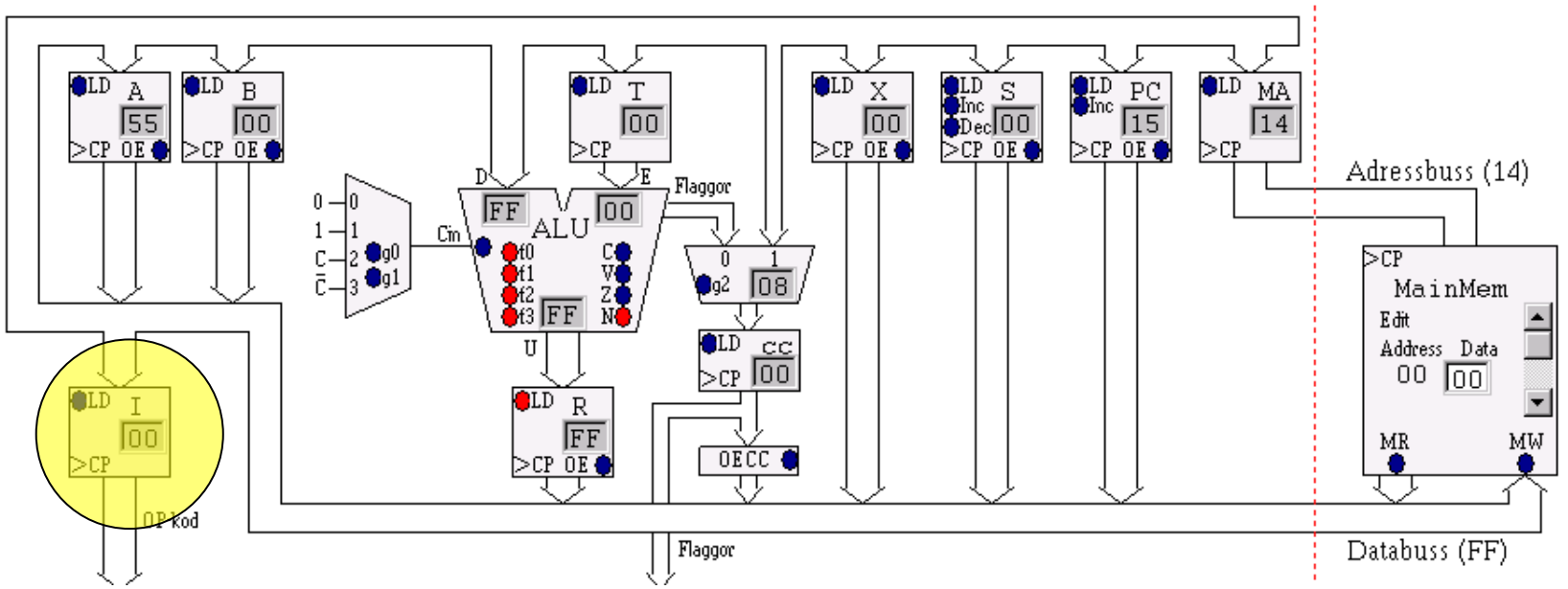


Instruktion: *Format*

LDAA #Data Ord1: OP

(LDAA #$55) Ord2: D

Tillstånd	Summaterm	RTN-beskrivning	Styrsignaler (=1)
Q_5	$Q_5 * I_{0F}$	PC→MA PC+1→PC	OE_{PC}, LD_{MA} IncPC
Q_6	$Q_6 * I_{0F}$	M→A	MR, LD_A, NF



Tillstå	Summater	RTN-beskrivning	Styrsignaler (=1)
	I91 * Q5	Y → Z	
	I91 * Q6	Q → P	
	I91 * Q7	W → U	NF

struktionsnr (OP-kod) AND State xx

LDAB #$23
ADDB $F3

Aktivera ”Processorns arbetssätt”