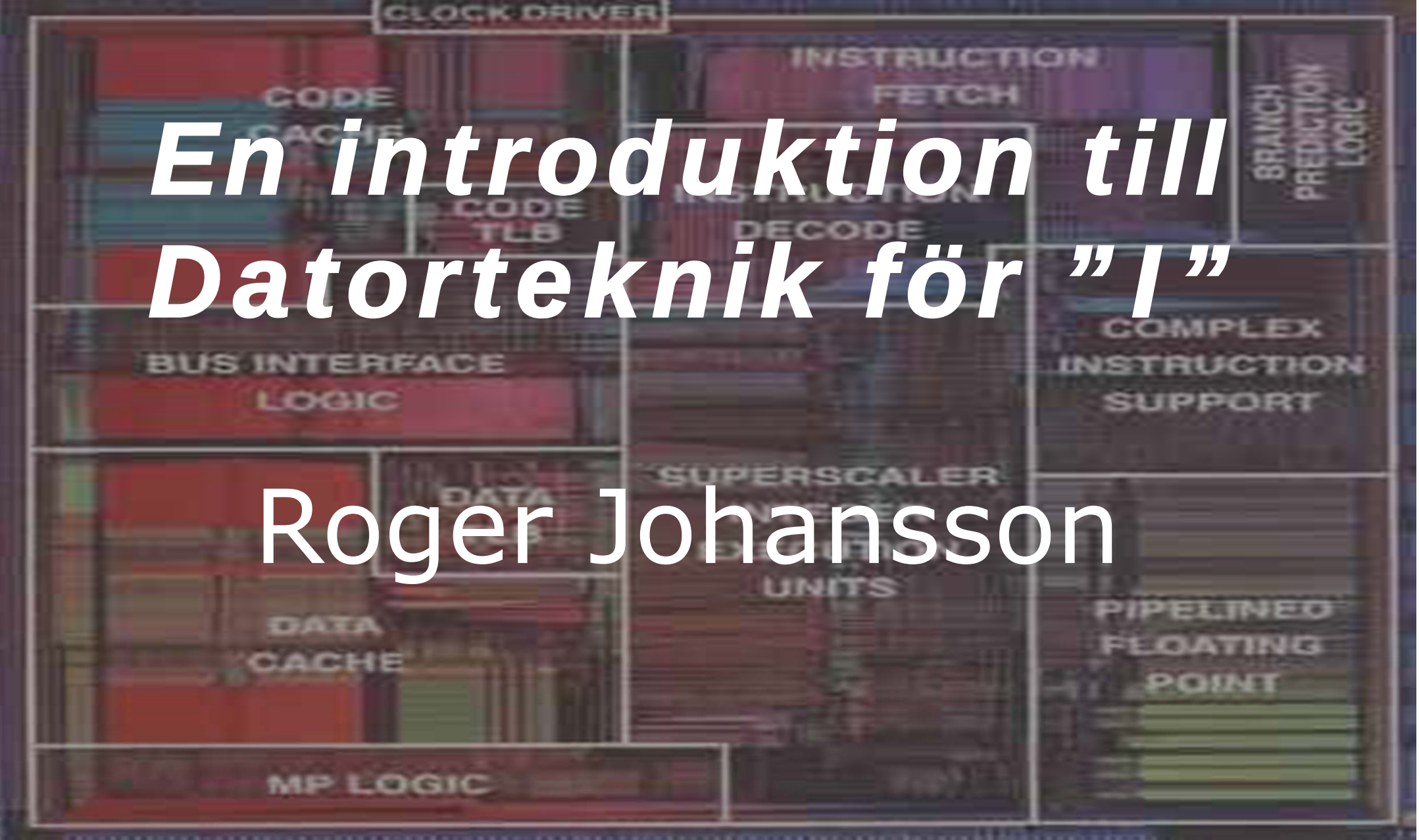




En introduktion till Datorteknik för "I"

Roger Johansson

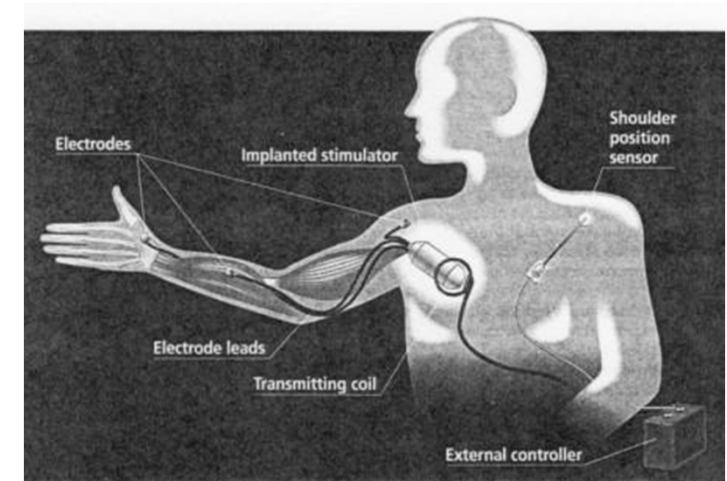
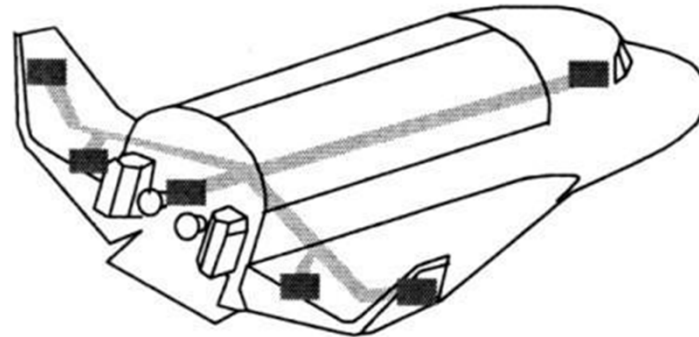
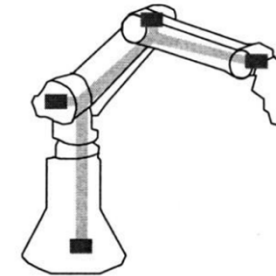
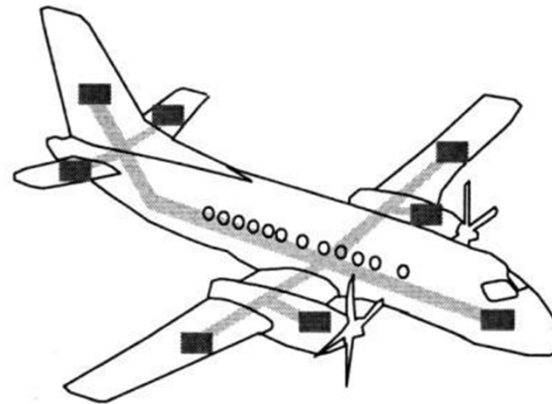
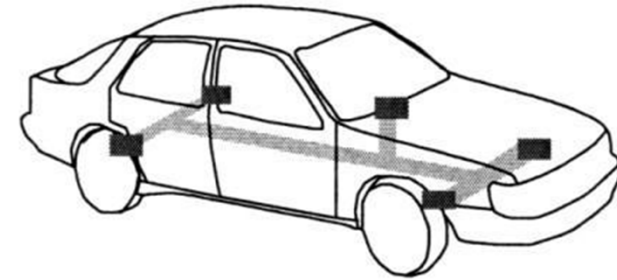


Datortekniken ligger till grund för en lång rad välbekanta vardagsprylar

- Mobiltelefoner,
- mediaspelare; mp3, IPOD
- digitalboxar, "laptops", hemma-bio
- spelkonsoler
- mikrovågsugnar
- huslarm, "smartcards" etc.

Samma teknik ligger också till grund för ett stort antal av samhällets tekniska system

- arbetsstationer
- bilar
- flygplan
- trafiksystem
- larmsystem
- robotar
- processmaskiner
- medicinsk apparatur
- rymdfarkoster m fl



Centraliserade tjänster

..för att inte nämna
datorkluster för
lagring/bearbetning av
mycket stora datamängder

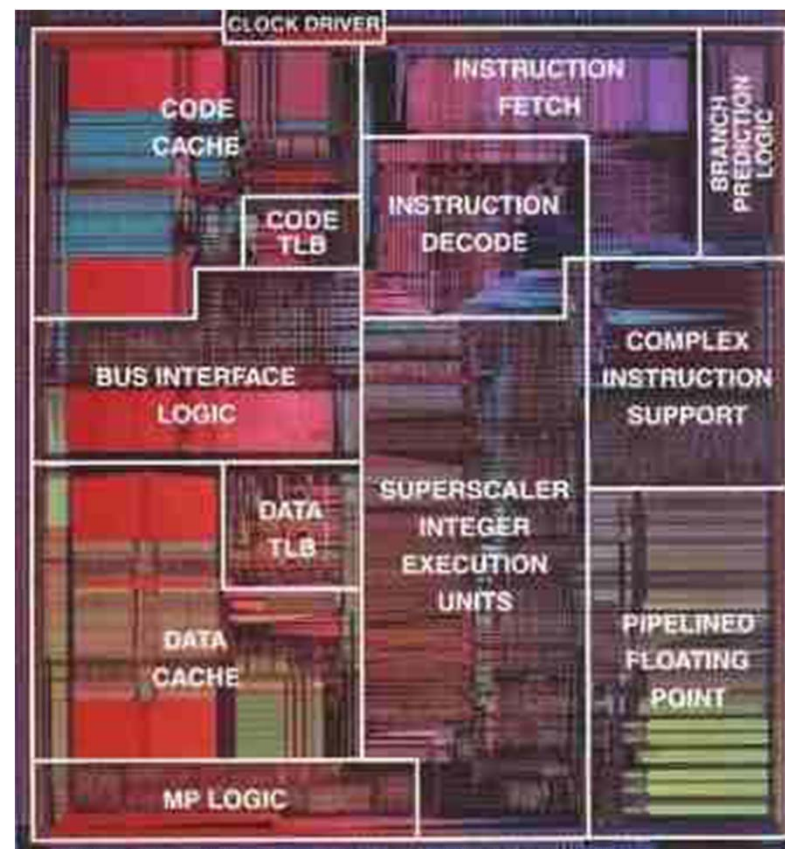
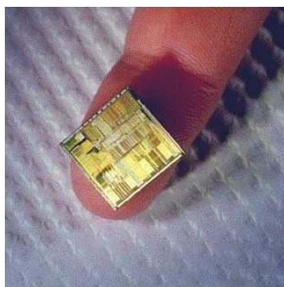


Alla dessa tekniska "prylar" och system innehåller **digitala system**.

Datorsystem är programmerbara digitala system.

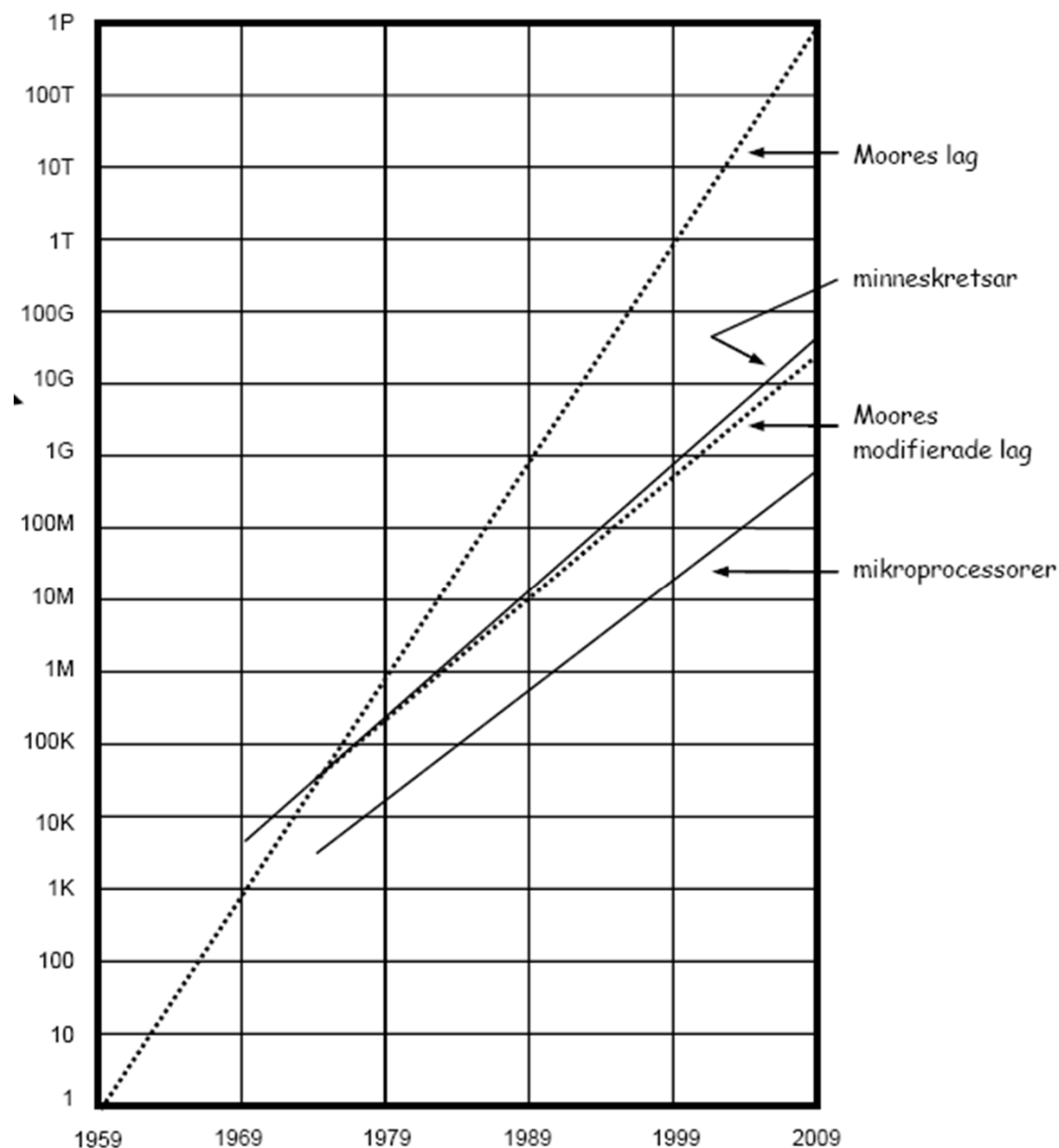
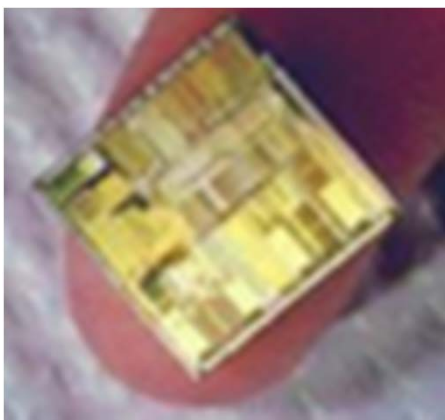
Digitala system baseras på **digital teknik** som alltså är fundamental för skapandet av datorsystem.

Kärnan i ett datorsystem är en **mikroprocessor**.

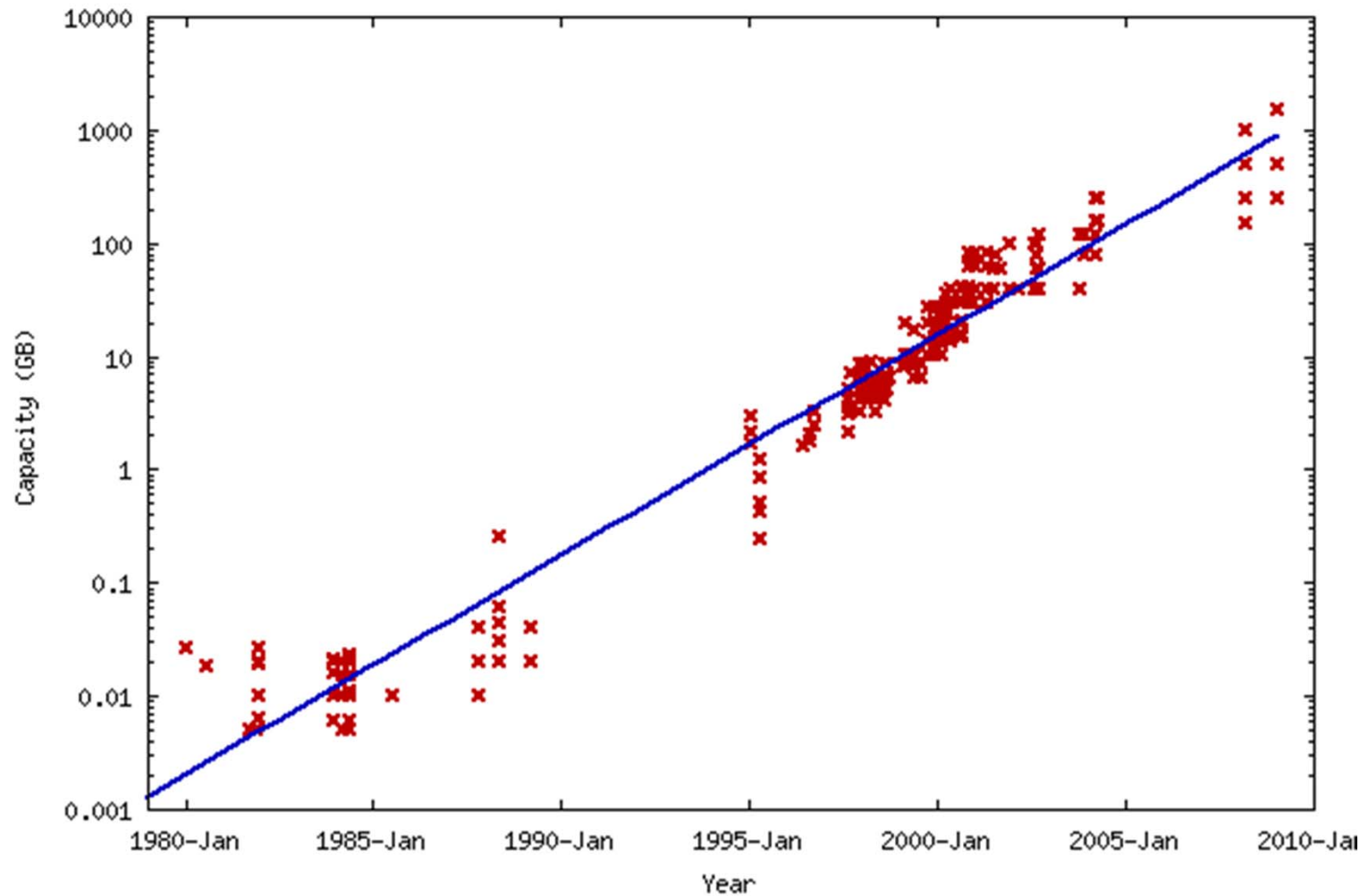


Mikroelektronikens utveckling

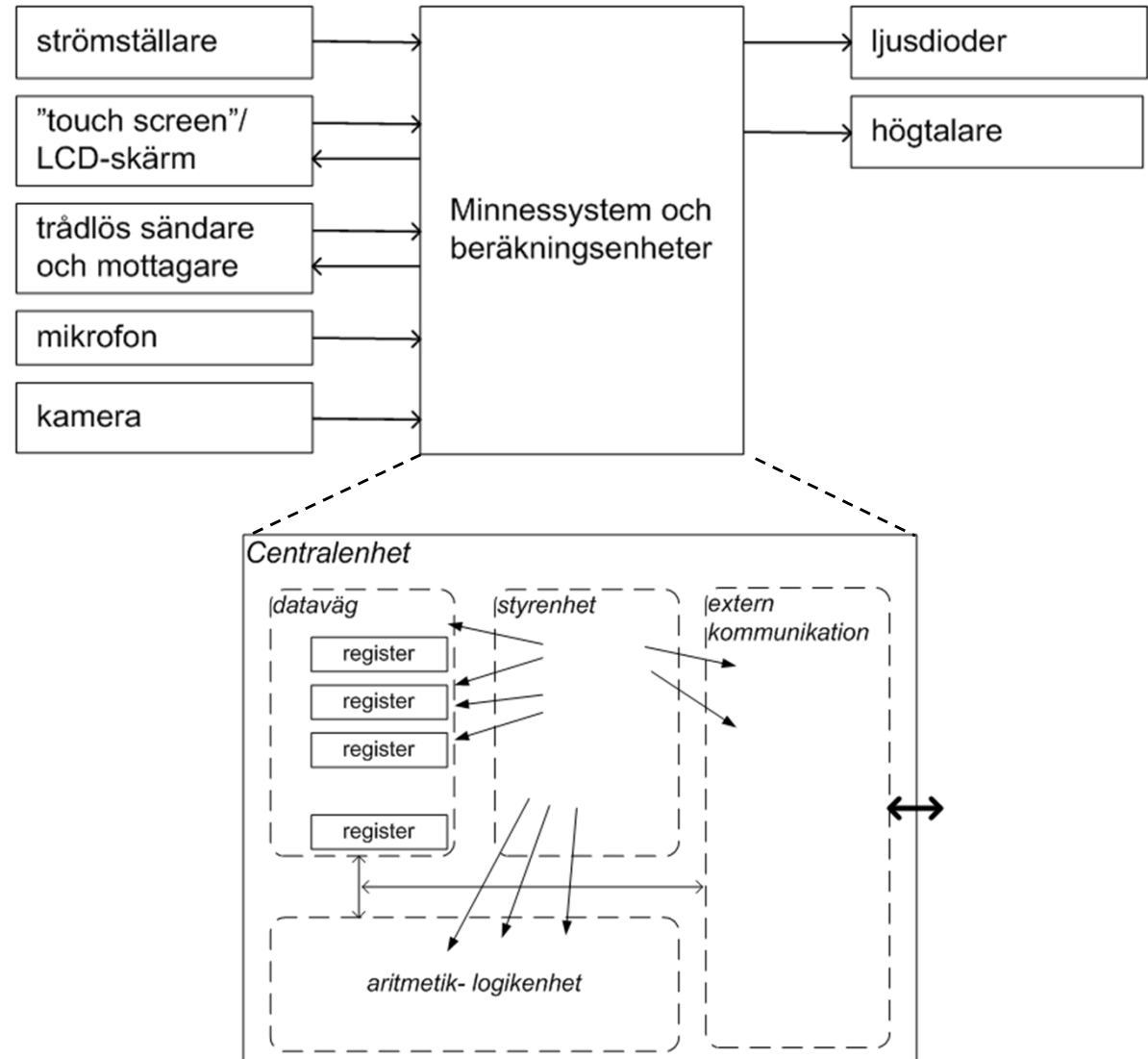
Antalet transistorer som ryms på en kiselbricka fördubblas var 18-e månad



Sekundärlagringskapacitetens utveckling



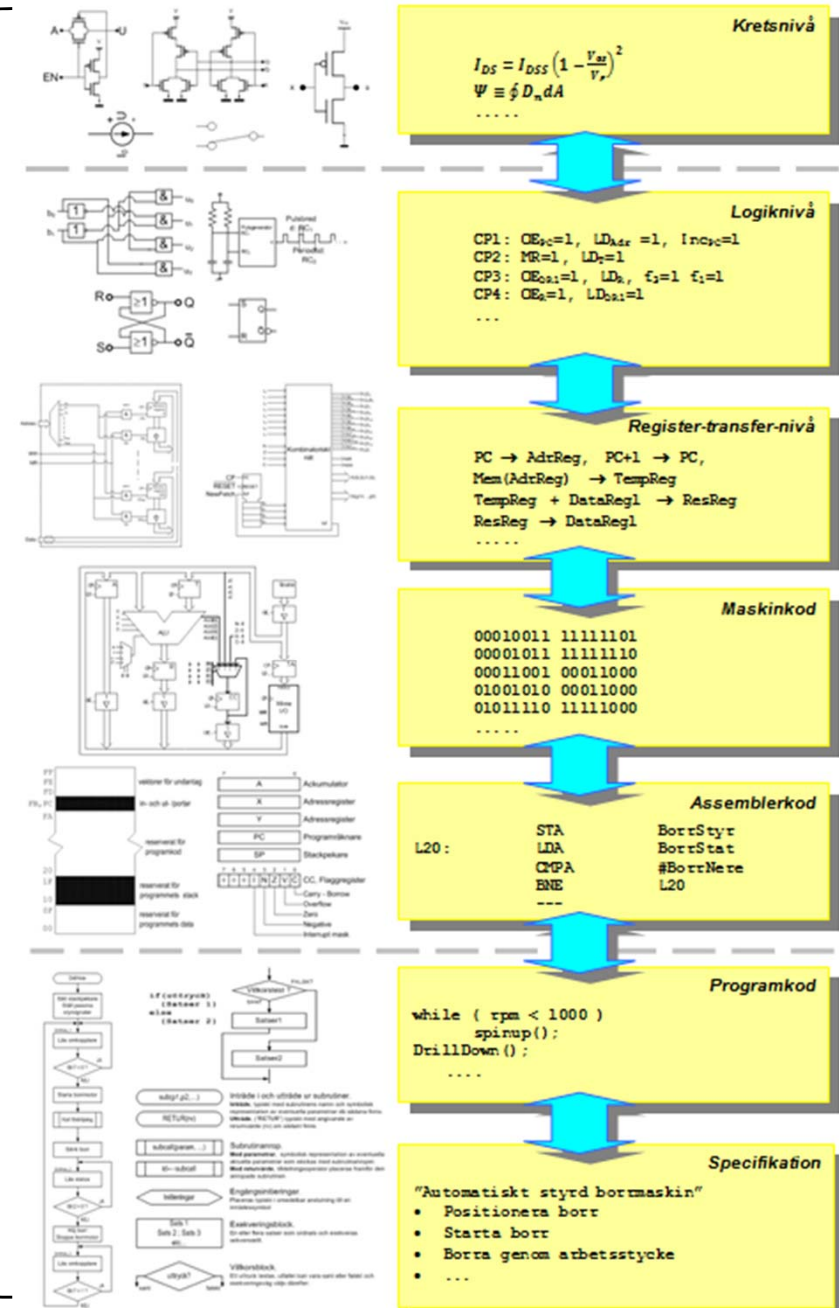
Datorsystemets uppbyggnad



Beskrivningsnivåer

Kan beskrivas ur olika aspekter

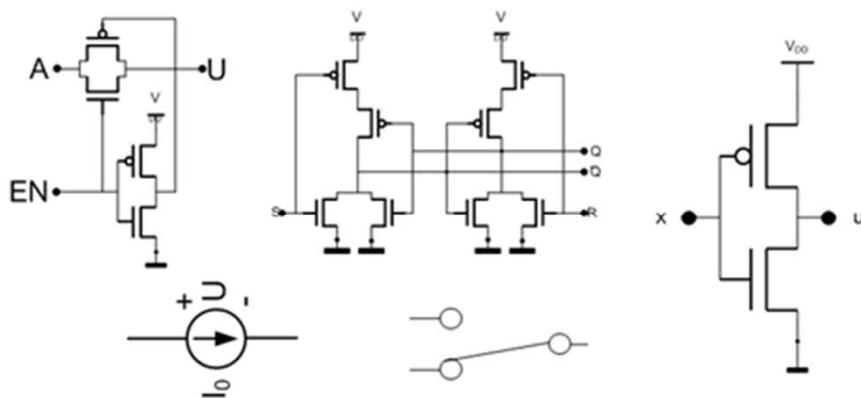
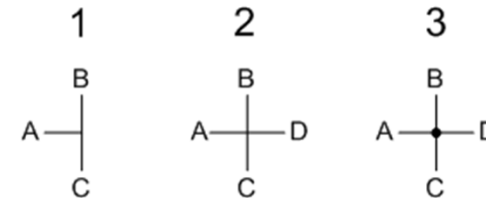
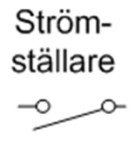
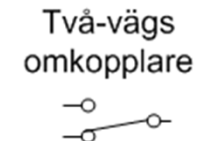
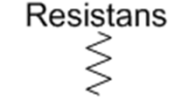
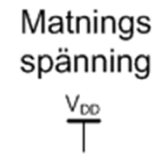
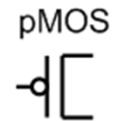
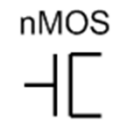
- Kretsnivå
- Logiknivå
- Register-transfervnivå
- Maskinkod
- Assemblerkod
- Programkod
- Specifikation



Kretsnivå

Komponenter

- ☐ Resistorer
- ☐ Kondensatorer
- ☐ Induktorer
- ☐ Dioder
- ☐ Transistorer



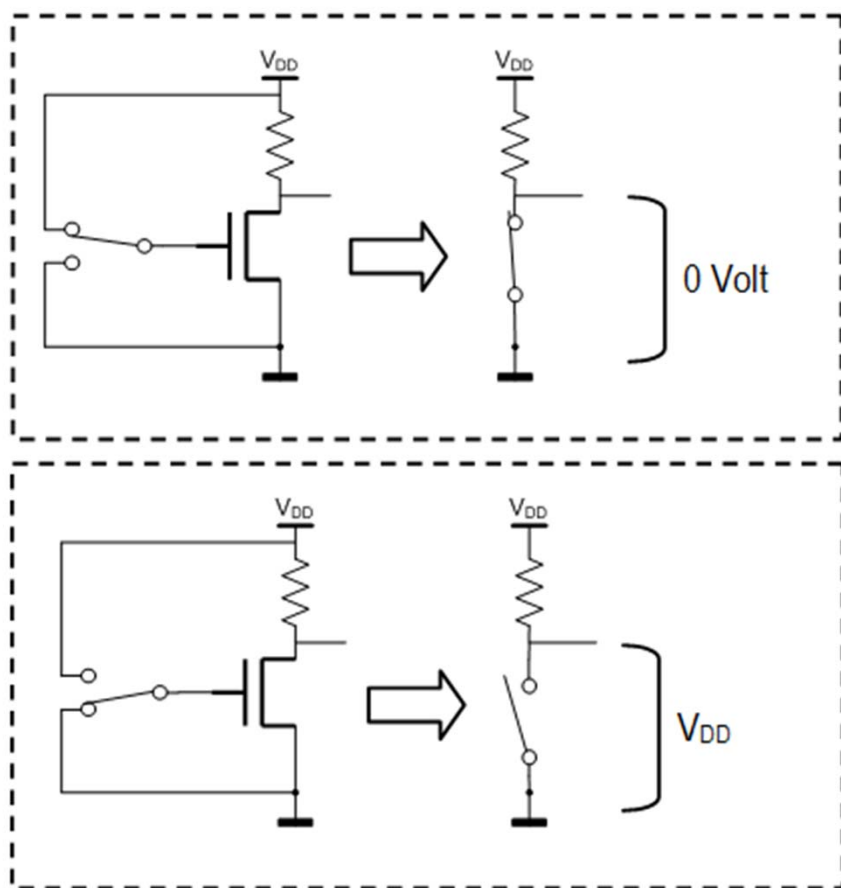
Kretsnivå

$$I_{DS} = I_{DSS} \left(1 - \frac{V_{GS}}{V_P}\right)^2$$

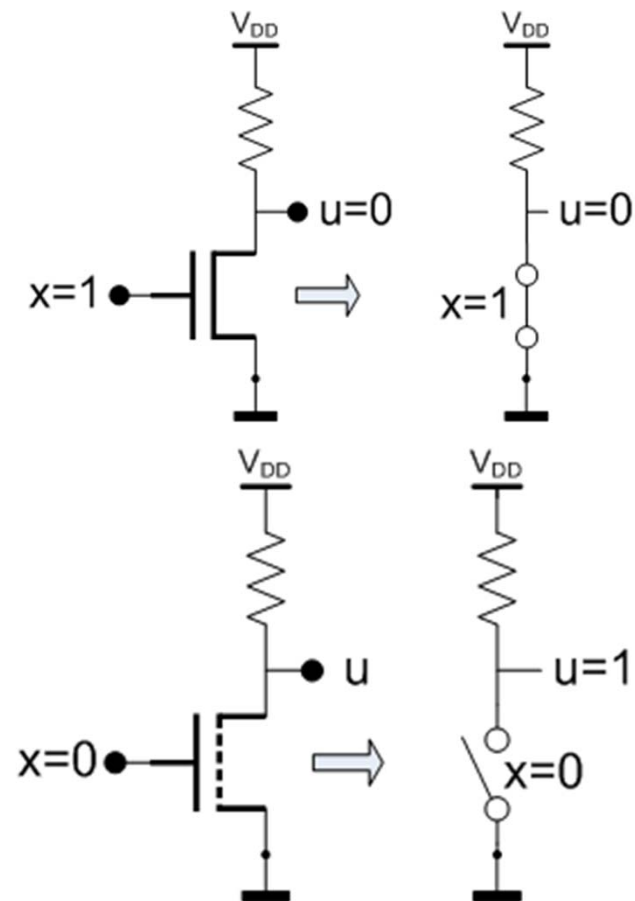
$$\Psi \equiv \oint D_n dA$$

.....

Transistorn som omkopplare

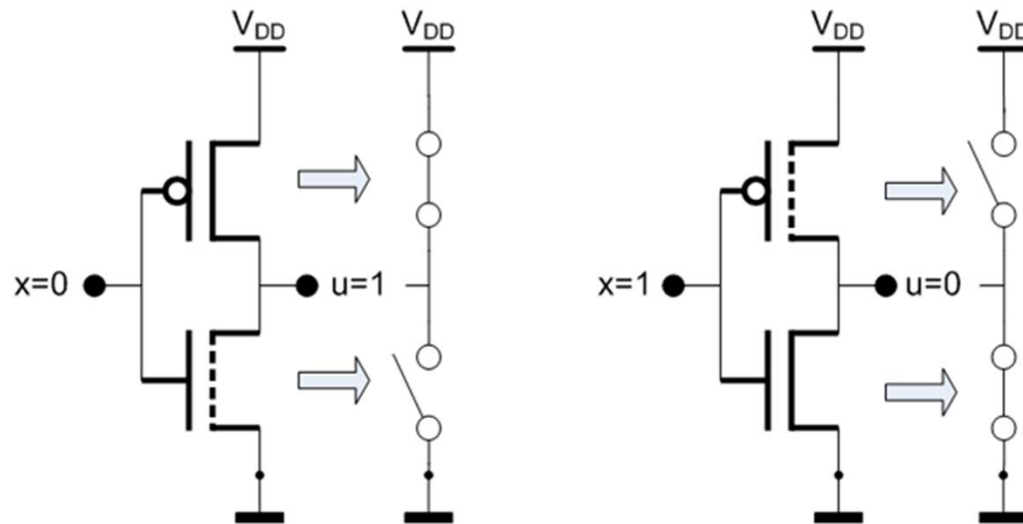


Signalnivåer (V_{DD} eller 0 Volt)

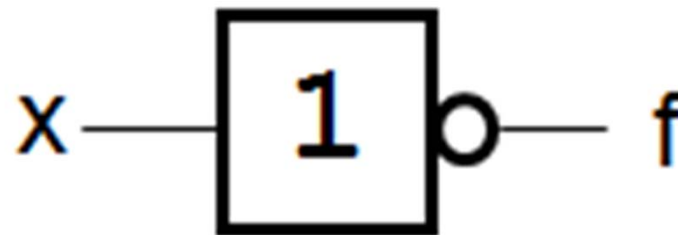


Logiknivåer (1 eller 0)

CMOS – en logikfunktion



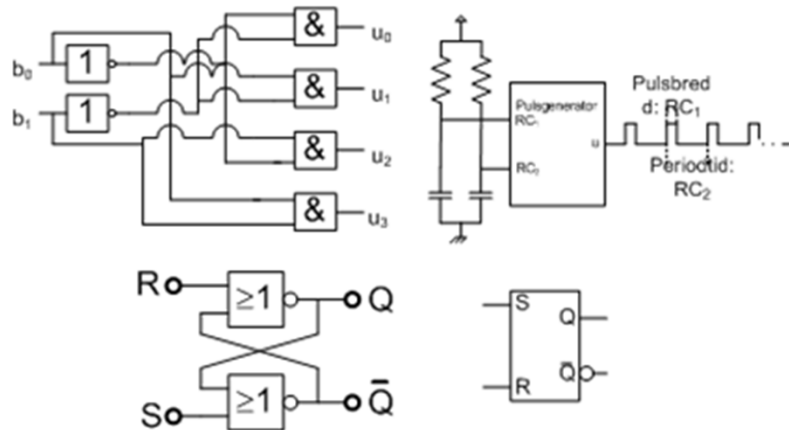
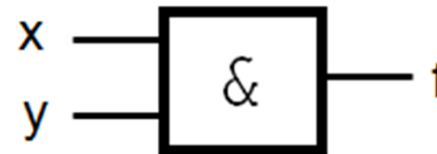
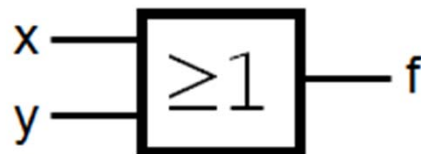
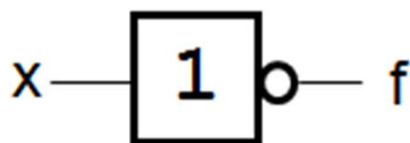
Funktionstabell	
x	$u = f(x)$
0	1
1	0



”Inverterare”

Logiknivå – de elementära byggblocken

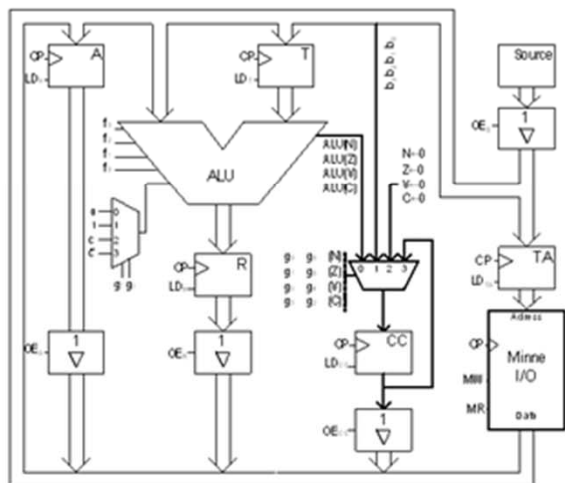
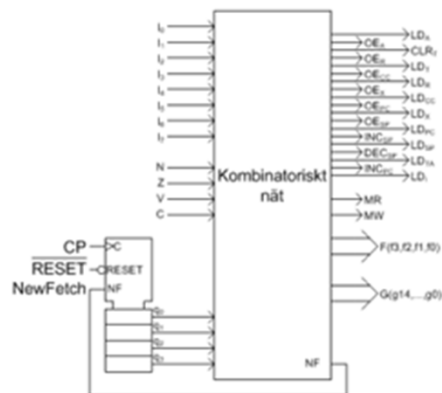
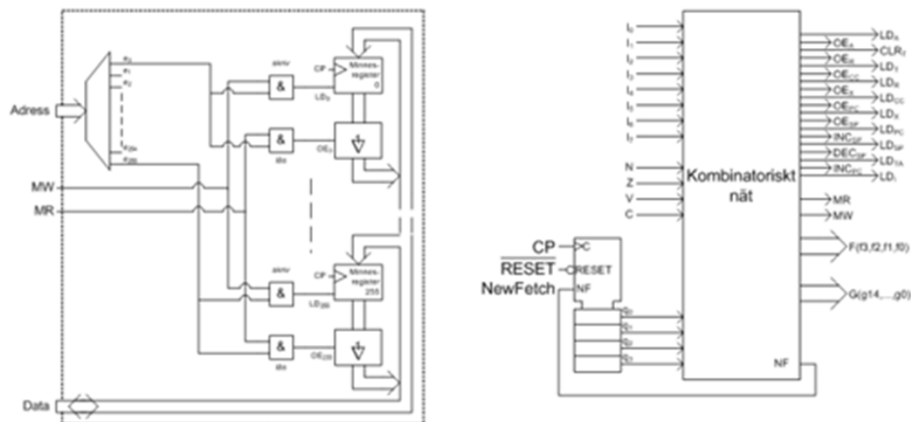
Logikkretsar byggs med hjälp av tre grundläggande funktioner



Logiknivå

CP1: $OE_{PC}=1, LD_{Adr}=1, Inc_{PC}=1$
 CP2: $MR=1, LD_T=1$
 CP3: $OE_{DR1}=1, LD_R, f_3=1, f_1=1$
 CP4: $OE_R=1, LD_{DR1}=1$
 ...

Register-transfervnivå - styrsignalsekvenser



Register-transfer-nivå

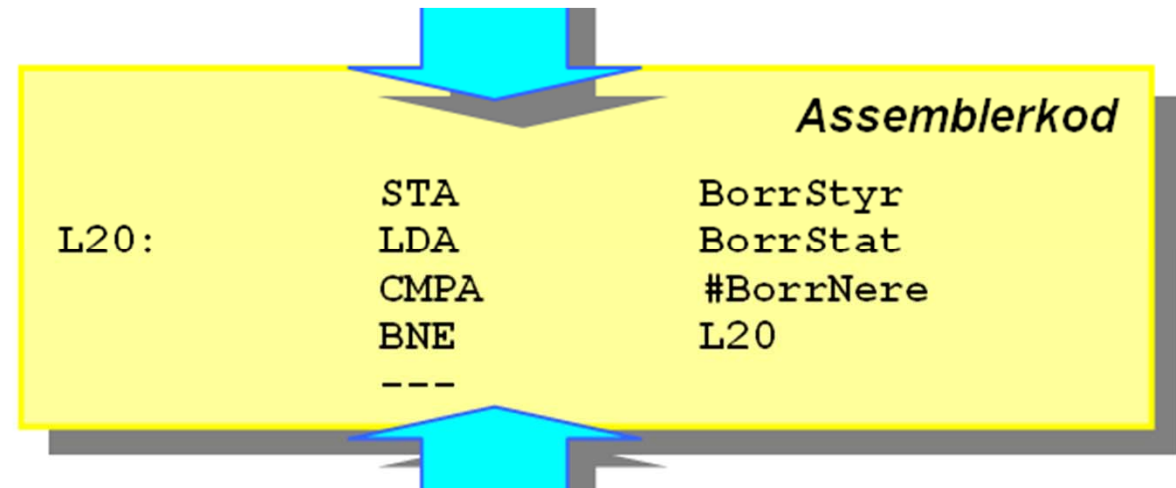
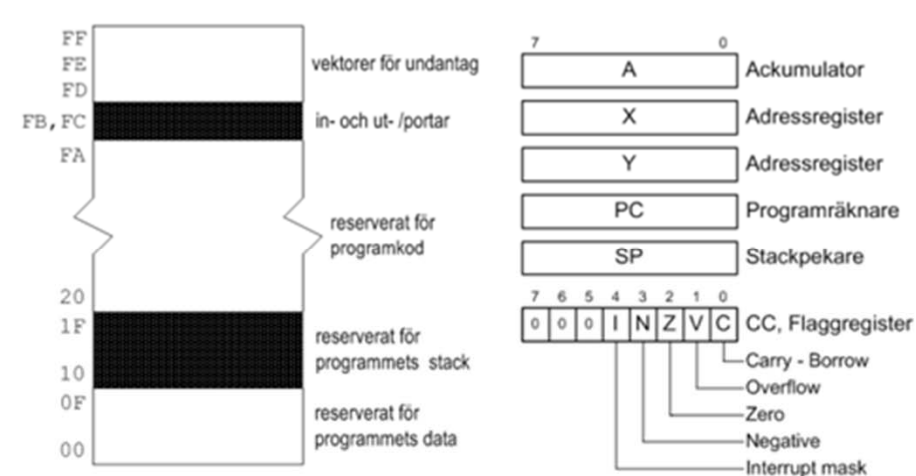
PC $\rightarrow$ AdrReg, PC+1 $\rightarrow$ PC,
 Mem(AdrReg) $\rightarrow$ TempReg
 TempReg + DataReg1 $\rightarrow$ ResReg
 ResReg $\rightarrow$ DataReg1

Maskinkod

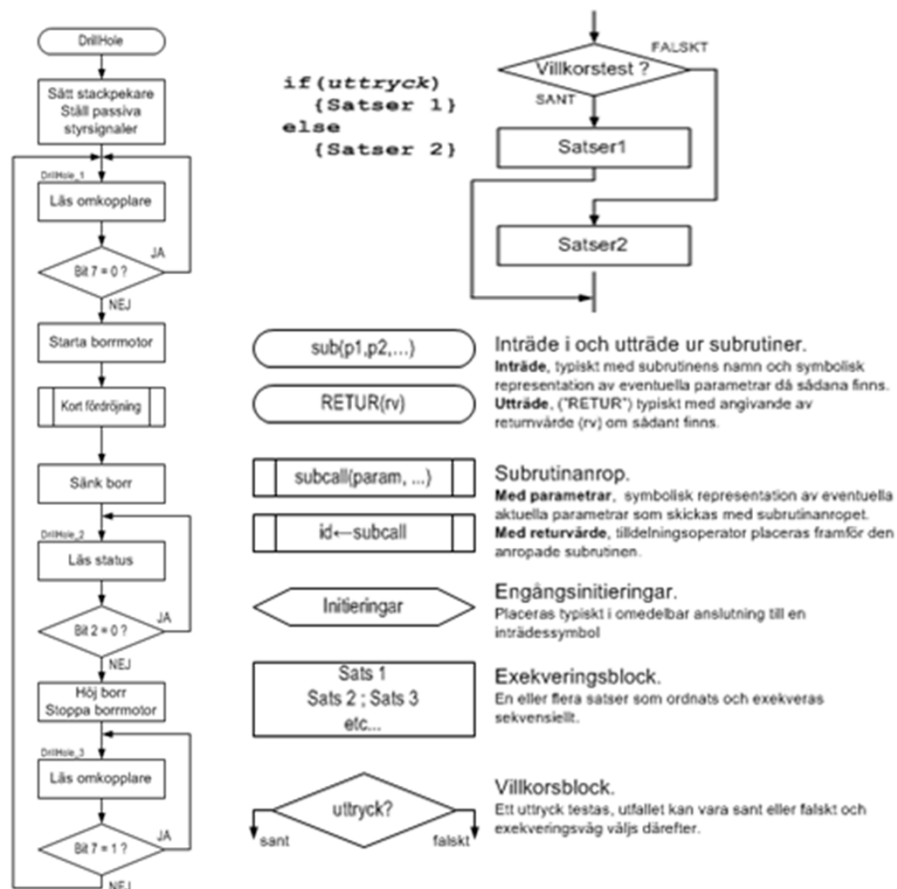
00010011 11111101
 00001011 11111110
 00011001 00011000
 01001010 00011000
 01011110 11111000

Assemblerprogrammering

Symbolfält	Mnemoniceller direktiv	Operand	Kommentarsfält
	ORG	\$20	; Program börjar
InPort	EQU	\$FB	
UtPort	EQU	\$FC	
start:	LDA	InPort	; Läs data
	STA	UtPort	; Skriv data
	JMP	start	; Repetera...



Specifikation och program



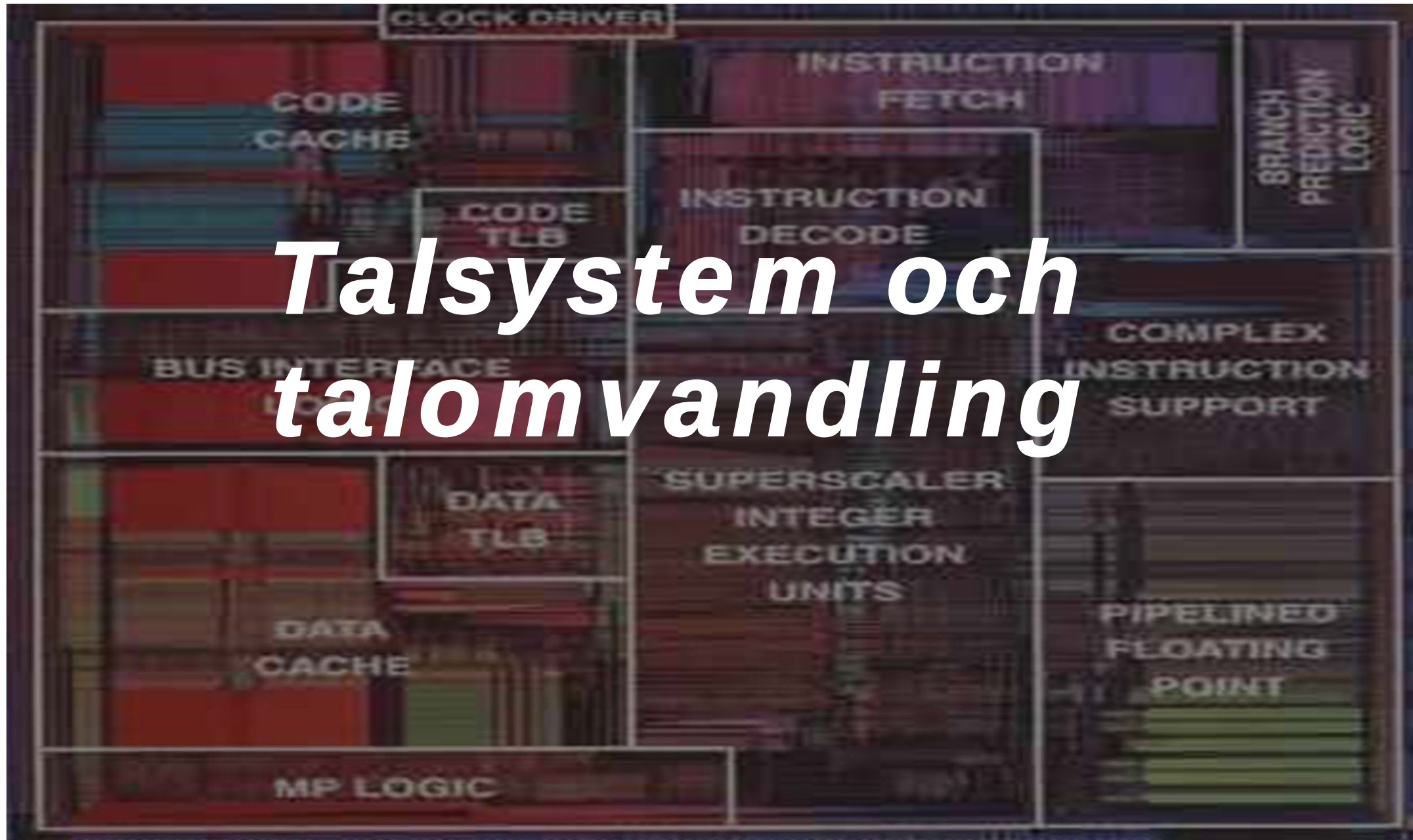
Programkod

```
while ( rpm < 1000 )
    spinup () ;
DrillDown () ;
....
```

Specifikation

"Automatiskt styrd bormaskin"

- Positionera borr
- Starta borr
- Borra genom arbetsstycke
- ...



Talsystem och talomvandling

Binär kodning

- Begrepp
- Tal och talsystem
- ASCII-kod
- NBCD

*Vad betyder
ettorna och nollorna?*

100110100011110101010111

Begrepp vid binär kodning

<i>begrepp</i>	<i>betydelse</i>	<i>exempel...</i>
bit/bitar	minsta informationsenhet, kan anta två värden	0 eller 1
bitsträng binärt ord	sekvens av bitar	101100100001...
kodord	$K_7 K_6 K_5 K_4 K_3 K_2 K_1 K_0$ också ett binärt ord men med en fastställd kodning (betydelse)	1000001 = "A" (ASCII) 1000001 = 65 (naturligt tal) 1000001 = -63 (heltal)
ordlängd	antal bitar i ordet	
nibble	ordlängden 4 bitar	0101
byte	ordlängden 8 bitar	01011100

Positionssystem, 10-decimalt

Ett N -bitars tal. $N = n+m$ där n är antalet siffror i heltalsdelen och m är antalet siffror i bråkdelen skriver vi allmänt:

$$\begin{array}{ccccccc} & d_{n-1} & d_{n-2} & \dots & d_1 & d_0 & . & d_{-1} & d_{-2} & \dots & d_{-(m-1)} & d_{-m} \\ \text{Mest signifikanta} & \swarrow & & & & & \searrow & & & & & \swarrow & \\ \text{siffror (MSD)} & & & & & & & \text{Decimalpunkt} & & & & & \text{Minst signifikanta} \\ & & & & & & & & & & & & \text{siffror (LSD)} \end{array}$$

Exempelvis, talet:

$$\begin{aligned} 123,456 &= 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2} + 6 \times 10^{-3} \\ &= 100 + 20 + 3 + 0,4 + 0,05 + 0,006 \end{aligned}$$

Där $N=6$, $n=m=3$, varje siffras vikt avgörs av dess position i talet...

Positionssystem, generellt

Talbasen β kan dock vara praktiskt taget vad som helst...

$$d_{n-1} \times \beta^{n-1} + d_{n-2} \times \beta^{n-2} + \dots + d_1 \times \beta^1 + d_0 \times \beta^0 + d_{-1} \times \beta^{-1} + d_{-2} \times \beta^{-2} + \dots d_{-(m-1)} \times \beta^{-(m-1)} + d_{-m} \times \beta^{-m}$$

Exempel: $\beta=10$

$$d_{n-1} \times 10^{n-1} + d_{n-2} \times 10^{n-2} + \dots + d_1 \times 10^1 + d_0 \times 10^0 + d_{-1} \times 10^{-1} + d_{-2} \times 10^{-2} + \dots d_{-(m-1)} \times 10^{-(m-1)} + d_{-m} \times 10^{-m}$$

Exempel: $\beta=2$

$$d_{n-1} \times 2^{n-1} + d_{n-2} \times 2^{n-2} + \dots + d_1 \times 2^1 + d_0 \times 2^0 + d_{-1} \times 2^{-1} + d_{-2} \times 2^{-2} + \dots d_{-(m-1)} \times 2^{-(m-1)} + d_{-m} \times 2^{-m}$$

Vi använder vanligen det enklare skrivsättet

$$N = (d_{n-1}d_{n-2}\dots d_1d_0 . d_{-1}d_{-2}\dots d_{-(m-1)}d_{-m})_{\beta}$$

Talbaser

Vi använder huvudsakligen tre olika talbaser:

Decimalt, för att vi är vana vid det.

Binärt, för att det motsvarar informationselementen i det digitala systemet.

Hexadecimalt, därför att det är ett bekvämt sätt att skriva grupper av binära siffror

Exempel:

$$(13)_{10} = (1101)_2 = (D)_{16}$$

bas 10 decimalt	bas 2 binärt	bas 16 hexadecimalt
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Talomvandlingar

För talomvandling *till* basen 10 använder vi definitionen direkt...

Exempel:

Omvandla till decimal form:

a) $(110.111)_2$

b) $(1A.8F)_{16}$

Lösning:

a) $(110.111)_2 = \{N=6, n=m=3, \beta=2\} =$
 $1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} =$
 $4 + 2 + 0 + 1/2 + 1/4 + 1/8 = 6 + 7/8 = (6,875)_{10}$

b) $(1A.8F)_{16} = \{N=4, n=m=2, \beta=16\} =$
 $1 \times 16^1 + 10 \times 16^0 + 8 \times 16^{-1} + 15 \times 16^{-2} =$
 $16 + 10 + 8/16 + 15/256 = 26 + 143/256 = (26,55859375)_{10}$

Omvandling från N_{10} till N_{β}

1. Dela upp N_{10} i heltalsdel och bråktalsdel.
2. Heltalsdelen omvandlas via succesiva divisioner med β .
3. Bråkdelen omvandlas via succesiva multiplikationer med β .

Exempel:

Omvandla $(122,18)_{10}$ till binär form. Bråkdelen avkortas vid behov till 7 korrekta bråksiffror.

1. Omvandla $(122)_{10}$ till binär form

$$122/2 = 61 + 0/2 \rightarrow d_0 = 0$$

$$61/2 = 30 + 1/2 \rightarrow d_1 = 1$$

$$30/2 = 15 + 0/2 \rightarrow d_2 = 0$$

$$15/2 = 7 + 1/2 \rightarrow d_3 = 1$$

$$7/2 = 3 + 1/2 \rightarrow d_4 = 1$$

$$3/2 = 1 + 1/2 \rightarrow d_5 = 1$$

$$1/2 = 0 + 1/2 \rightarrow d_6 = 1$$

Terminerings-
villkor

Heltalsdelen således:

$$(1111010)_2$$

2. Omvandla $(0,18)_{10}$ till binär form

$0,18 \times 2 =$	0,36	$\rightarrow d_{-1} =$	0	Termineringsvillkor enligt uppgiftstexten 7 st. korrekta bråksiffror
$0,36 \times 2 =$	0,72	$\rightarrow d_{-2} =$	0	
$0,72 \times 2 =$	1,44	$\rightarrow d_{-3} =$	1	
$0,44 \times 2 =$	0,88	$\rightarrow d_{-4} =$	0	
$0,88 \times 2 =$	1,76	$\rightarrow d_{-5} =$	1	
$0,76 \times 2 =$	1,52	$\rightarrow d_{-6} =$	1	
$0,52 \times 2 =$	1,04	$\rightarrow d_{-7} =$	1	

Bråkdelen således:

$$(0.0010111)_2$$

Omvandla till hexadecimal form

Exempel:

Omvandla $(122,18)_{10}$ till hexadecimal form. Bråkdelen avkortas vid behov till 2 korrekta bråksiffror.

Heltalsdelen:

$$122/16 = 7 + 10/16 \rightarrow d_0 = (10)_{10} = (A)_{16}$$

$$7/16 = 0 + 7/16 \rightarrow d_1 = (7)_{10} = (7)_{16}$$

Bråkdelen:

$$0,18 \times 16 = 2,88 \rightarrow d_{-1} = (2)_{10} = (2)_{16}$$

$$0,88 \times 16 = 14,08 \rightarrow d_{-2} = (14)_{10} = (E)_{16}$$

Svar:

$$(122,18)_{10} \approx 7A.2E$$

Alfanumeriska tecken → ASCII

American Standard Code for Information Interchange

Typiskt användningsområde: Tangentbord



7-bitars ASCII kodning

0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1	K ₆ K ₅ K ₄ K ₃ K ₂ K ₁ K ₀
NUL	DLE	SP	0	@	P	`	p	0 0 0 0
SOH	DC1	!	1	A	Q	a	q	0 0 0 1
STX	DC2	“	2	B	R	b	r	0 0 1 0
ETX	DC3	#	3	C	S	c	s	0 0 1 1
EOT	DC4	\$	4	D	T	d	t	0 1 0 0
ENQ	NAK	%	5	E	U	e	u	0 1 0 1
ACK	SYN	&	6	F	V	f	v	0 1 1 0
BEL	ETB	‘	7	G	W	g	w	0 1 1 1
BS	CAN	(	8	H	X	h	x	1 0 0 0
HT	EM	)	9	I	Y	i	y	1 0 0 1
LF	SUB	*	:	J	Z	j	z	1 0 1 0
VT	ESC	+	;	K	[Ä	k	{ä	1 0 1 1
FF	FS	,	<	L	\Ö	l	ö	1 1 0 0
CR	GS	-	=	M	]Å	m	}å	1 1 0 1
S0	RS	.	>	N	^	n	~	1 1 1 0
S1	US	/	?	O	_	o	RUBOUT (DEL)	1 1 1 1

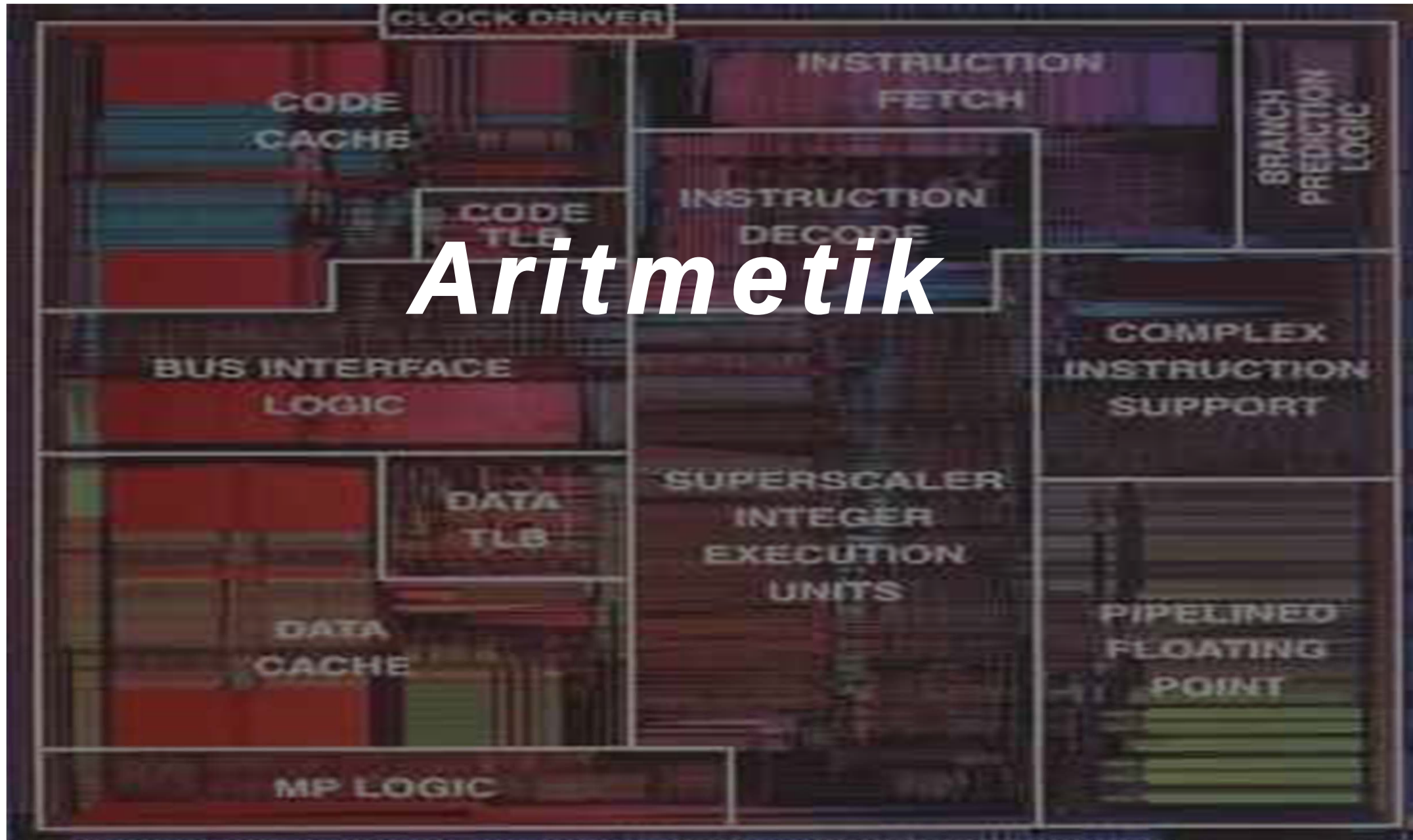
ASCII– Exempel

Textsträngen "Hej" representeras som:

1001000	1100101	1101010
'H'	'e'	'j'

Textsträngen "9756" representeras som:

0111001	0110111	0110101	0110110
'9'	'7'	'5'	'6'



Binär addition – ”papper och penna metod”

Exempel: $(5)_{10} + (23)_{10} = ?$

$(5)_{10}$		$\begin{array}{r} 1\ 1\ 1 \\ \hline 101 \end{array}$	minnessiffror
$(23)_{10}$	+	10111	augend addend
$(28)_{10}$		11100	summa

Binär multiplikation – ”papper och penna metod”

Exempel: $(25)_{10} \times (11)_{10} = ?$

$(25)_{10}$		11001	multiplikator
$(11)_{10}$	$\times$	1011	multiplikand
		11001	
		11001	
		00000	
	+	11001	
$(275)_{10}$		100010011	produkt

Binär subtraktion – ”papper och penna metod”

Exempel: $(110)_{10} - (43)_{10} = ?$

		10 10	minnessiffror
(110) ₁₀		1101110	minuend
(43) ₁₀	-	101011	subtrahend
(67) ₁₀		1000011	skillnad

Binär division – ”papper och penna metod”

Exempel: $(33)_{10} : (6)_{10} = ?$

		0101	kvot
divisor	110	100001	dividend
	-	000	
		1000	
	-	110	
		0100	
	-	000	
		1001	
	-	110	
		011	rest

$$(100001)_2 : (110)_2 = (0101)_2 + (011)_2 : (110)_2$$

8-bitars addition

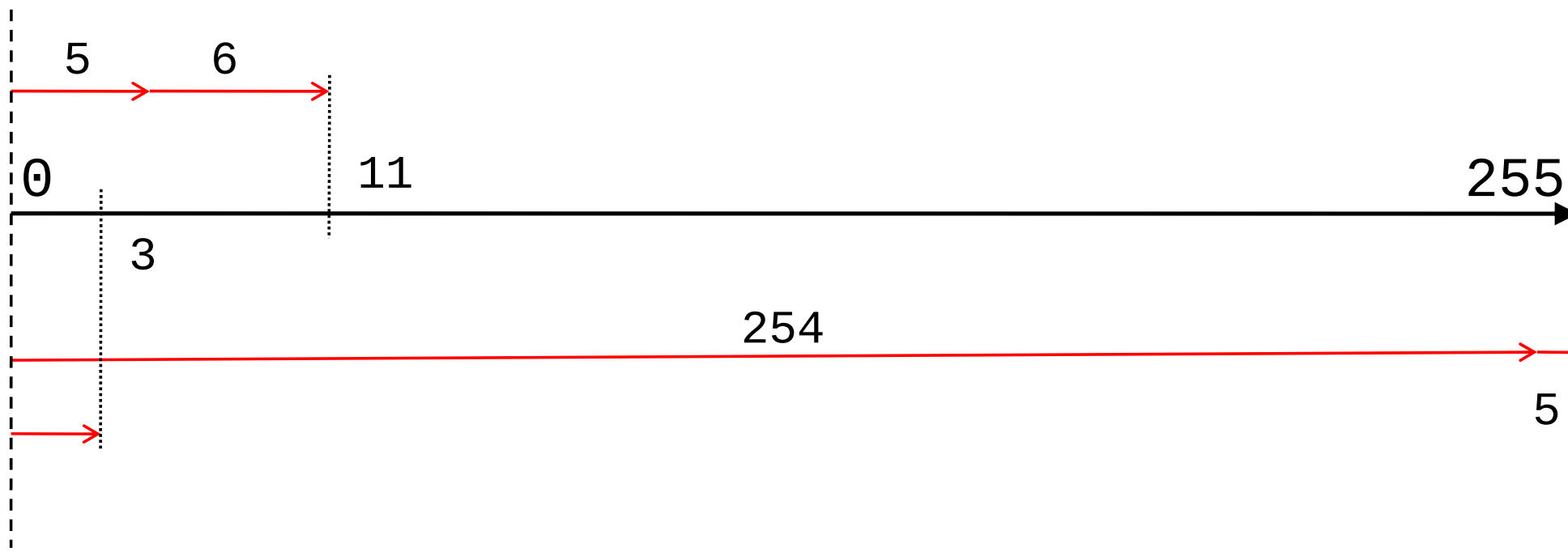
$$\begin{array}{r}
 5 \\
 + 6 \\
 \hline
 11
 \end{array}
 \qquad
 \begin{array}{r}
 1 \\
 \hline
 00000101 \\
 + 00000110 \\
 \hline
 00001011
 \end{array}$$

$$\begin{array}{r}
 254 \\
 + 5 \\
 \hline
 259
 \end{array}
 \qquad
 \begin{array}{r}
 111111 \\
 \hline
 11111110 \\
 + 00000101 \\
 \hline
 100000011 \\
 \boxed{00000011}
 \end{array}$$

8 bitar ger
talområdet
 $0..2^8-1 = 0..255$

Spill ! ("Overflow")

Geometrisk tolkning - tallinje



Vi säger att C-flaggan, genererad från addition av de mest signifikanta bitarna är en ***spillindikator***.

8-bitars subtraktion

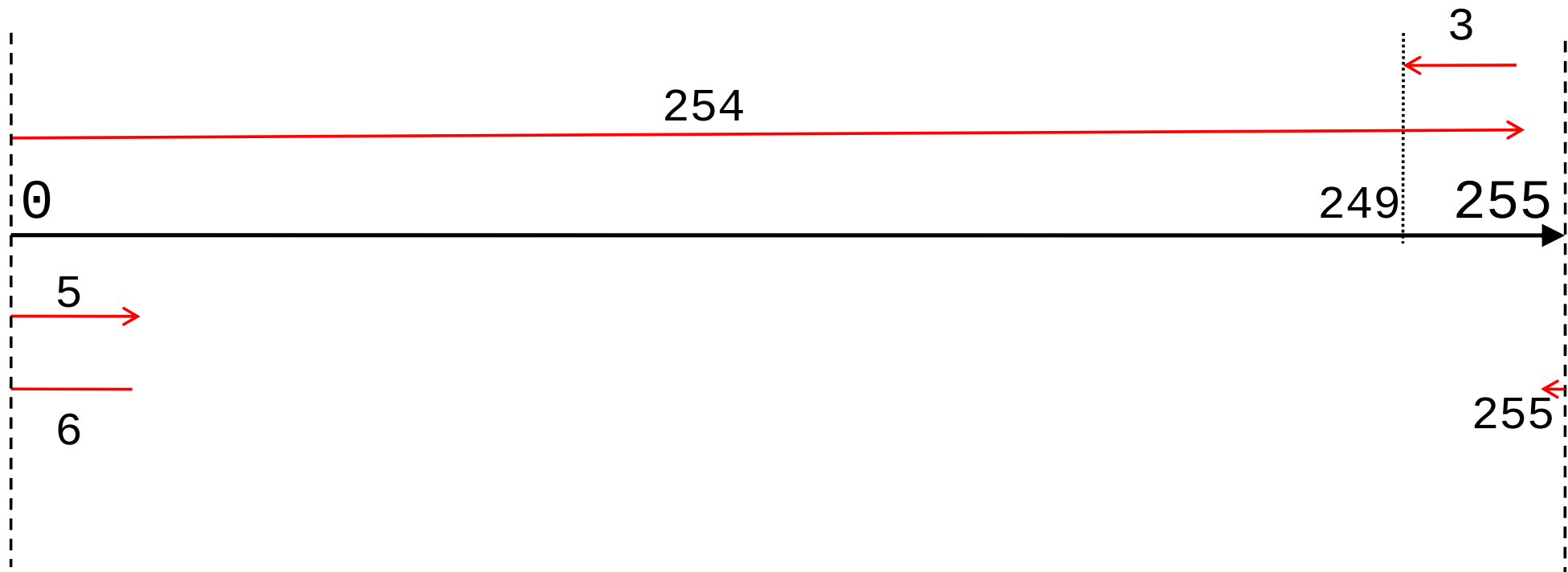
$$\begin{array}{r}
 254 \\
 - 5 \\
 \hline
 249
 \end{array}
 \qquad
 \begin{array}{r}
 1 \\
 \hline
 11111110 \\
 - 00000101 \\
 \hline
 11111001
 \end{array}$$

$$\begin{array}{r}
 5 \\
 - 6 \\
 \hline
 255
 \end{array}
 \qquad
 \begin{array}{r}
 1 \\
 1010101010 \\
 \hline
 00000101 \\
 - 00000110 \\
 \hline
 11111111
 \end{array}$$

För att kunna utföra subtraktionen tvingas vi låna av en tänkt siffra med vikt 2^8 .

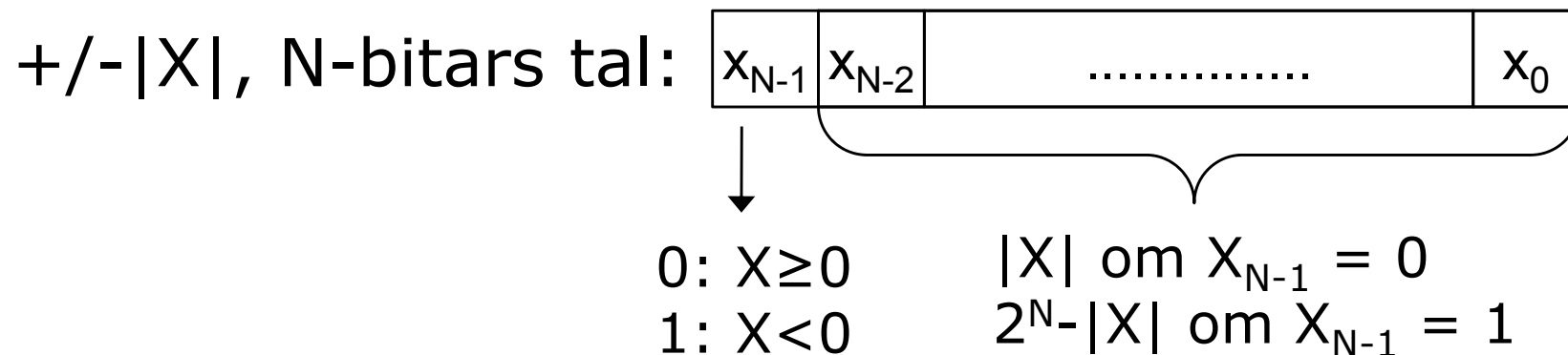
Spill ! ("Underflow")

Geometrisk tolkning - tallinje

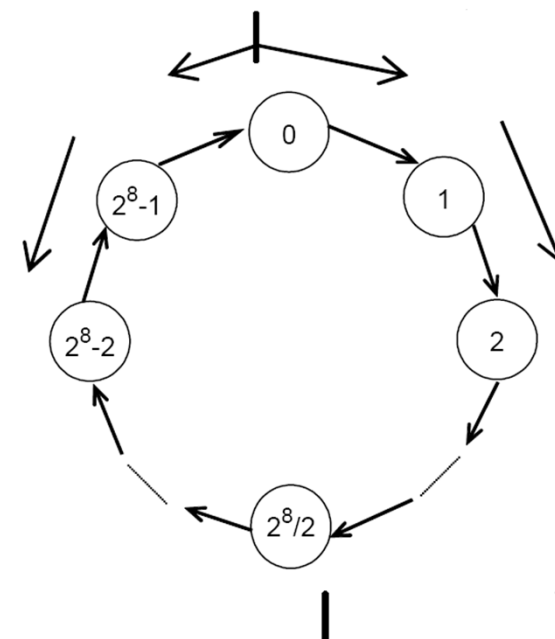
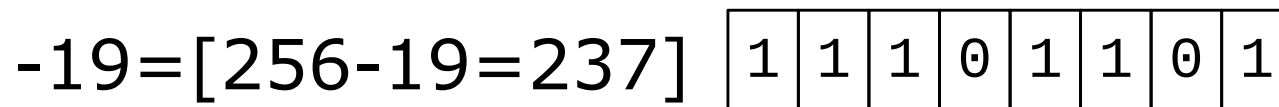
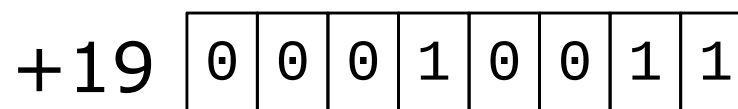


Den tänkta "lånebiten" kallar vi "Borrow",
en ***spillindikator***.

Tal med tecken - Tvåkomplementsform



Exempel: 8-bitars tal $+/- 19$:



Tvåkomplementsform - Metod för teckenbyte

$X+Y = 2^n \Rightarrow Y$ är 2-komplementet till X (n-bitars tal)

För 8-bitars tal således:

$$\begin{aligned} Y = X_{2k} &= 2^8 - X = \\ &= (2^8 - 1) - X + 1 \\ &= \underbrace{11111111 - X_7X_6X_5X_4X_3X_2X_1X_0}_{\text{1-komplement}} + 1 \end{aligned}$$

Detta kallas 1-komplement (X_{1k}).
Bitvis invertering

Exempel: Bestäm maskintalet på 8 bitars tvåkomplementsform för decimala talet -50

Vi utgår enklast från $X=50$ (och söker X_{2k})

$$(50)_{10} = X = 00110010$$

$$\begin{array}{r} X_{1k} = 11001101 \\ + 1 \end{array}$$

$$- (50)_{10} = X_{2k} = 11001110$$

Tvåkomplementsform - addition

Relation A och B, om:	Utförs A+B som:
$A, B \geq 0$	$A+B$
$A \geq 0, B < 0,$	$A+B_{2k} = [A+(2^N-B)] \pmod{2^N} = A-B = A + (-B)$
$A < 0, B \geq 0,$	$A_{2k}+B = [(2^N-A)+B] \pmod{2^N} = -A+B = B + (-A)$
$A, B < 0$	$A_{2k}+B_{2k} = [(2^N-A)+(2^N-B)] \pmod{2^N} = -A-B = -(A+B)$

Dvs. Oavsett vilka tecken de ingående talen har så fungerar rättfram binär addition.

Tvåkomplementsform - subtraktion

Vi inser också att en subtraktion kan utföras med hjälp av en adderare ty

$$A - B = A + (-B) \text{ och } -B = B_{2k} = B_{1k} + 1$$

Exempel: $6 - 5 = 6 + (-5) = 00000110 + 11111010 + 1$

$$(5)_{10} =$$

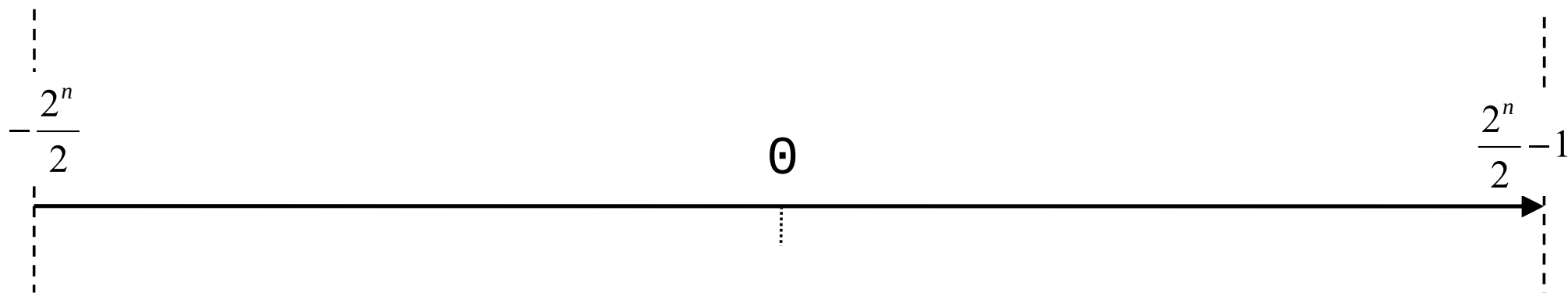
$$(00000101)_2$$

Dvs 1-komplement:

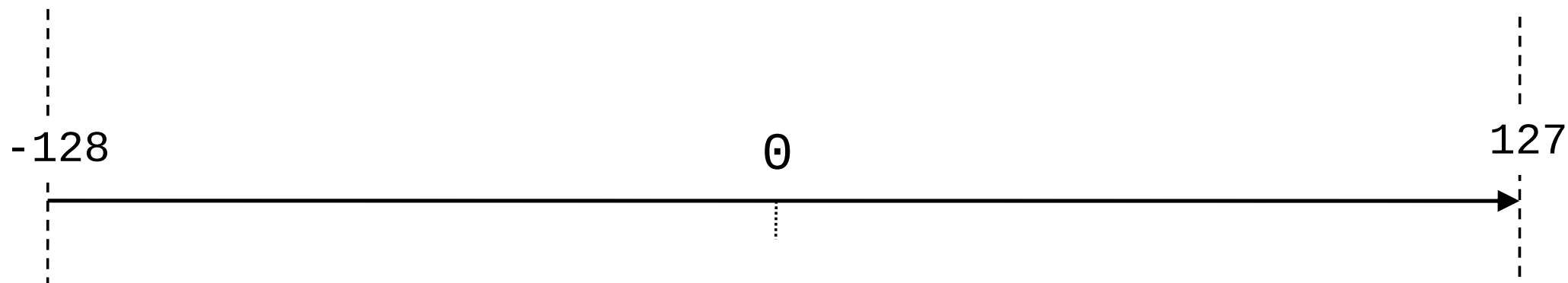
$$(11111010)_{1k}$$

$$\begin{array}{r} 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\ \hline 00000110 \\ +\ 11111010 \\ \hline 00000001 \end{array}$$

Tvåkomplementsform - talområde

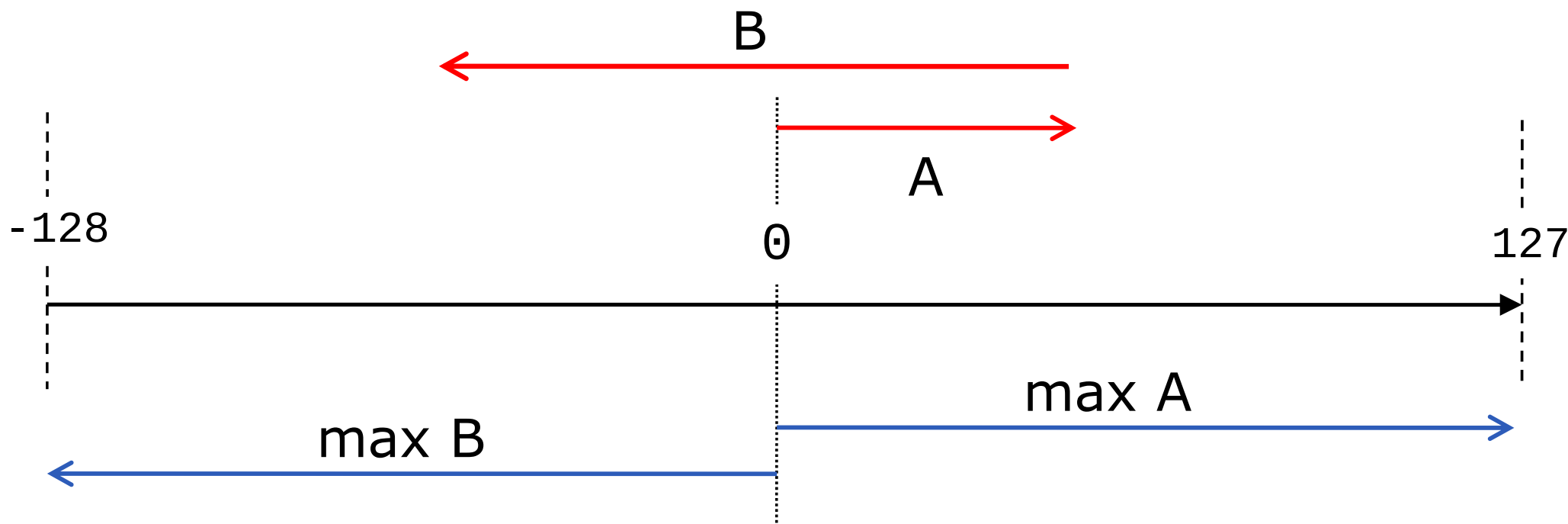


Exempel: 8-bitars tal



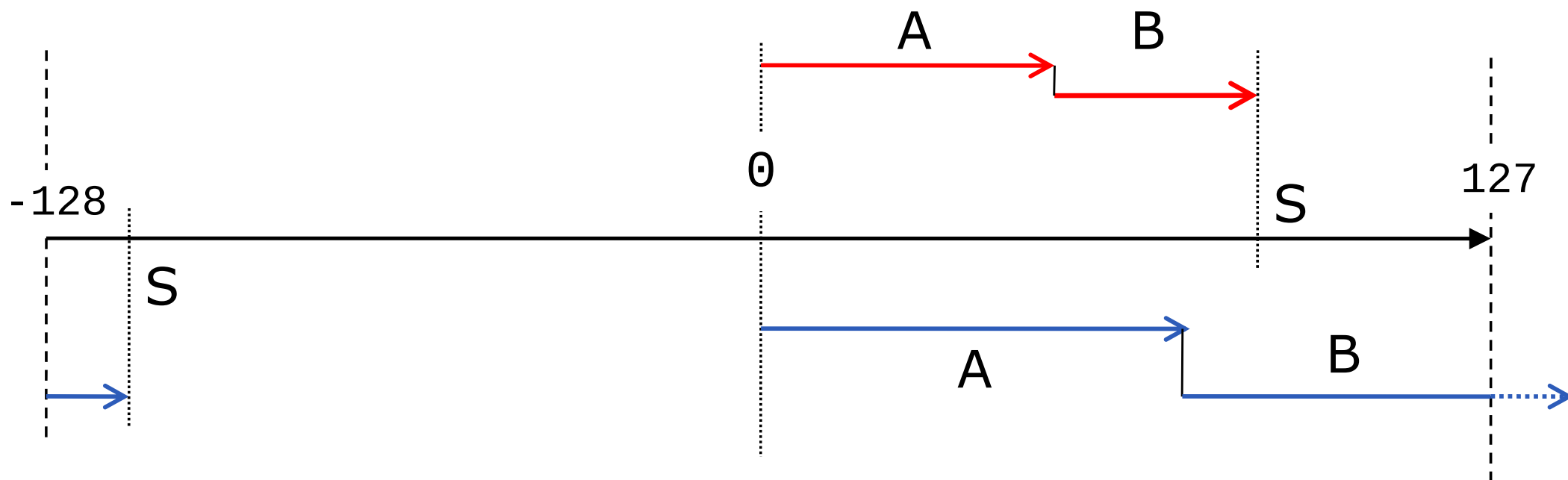
Tvåkomplementsform - spillindikatorer

$A + B$ där $A \geq 0$ och $B < 0$



Slutsats: Om A och B har olika tecken vid addition kan 2-komplementspill inte uppträda

$$A + B = S, \text{ där } A \geq 0 \text{ och } B \geq 0$$

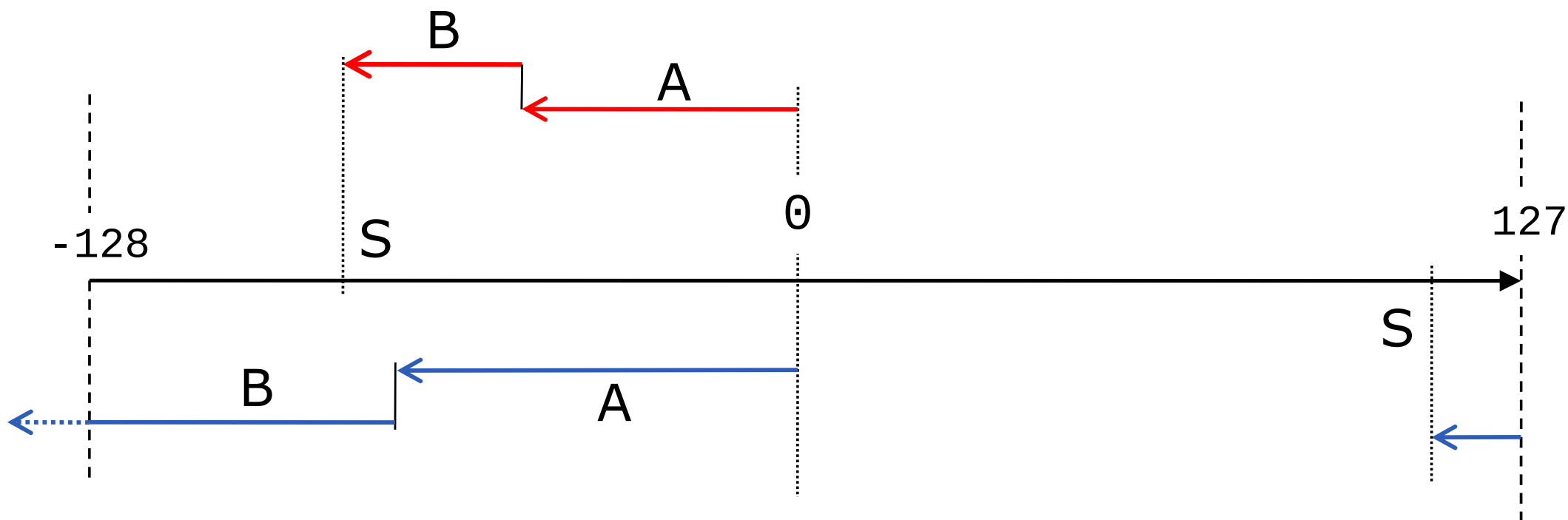


Slutsats:

Om A och B har samma tecken vid addition **kan** 2-komplementspill uppträda.
Vi kan konstatera spill genom en teckenöverläggning, dvs:

$$spill = (A \geq 0) (B \geq 0) (S < 0)$$

$A + B = S$, där $A < 0$ och $B < 0$



I detta fall kan vi skriva spillvillkoret:

$$spill = (A < 0) (B < 0) (S \geq 0)$$



Grindar och vippor...

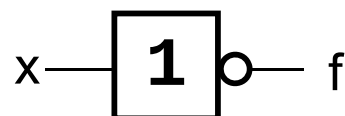
Negation, "ICKE" → NOT-grind (Inverterare)

Observera de alternativa skrivsätten inom Boolesk algebra $x' = \bar{x}$

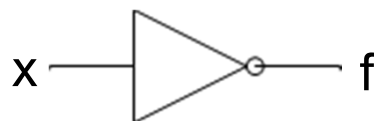
satslogik

Boolesk algebra

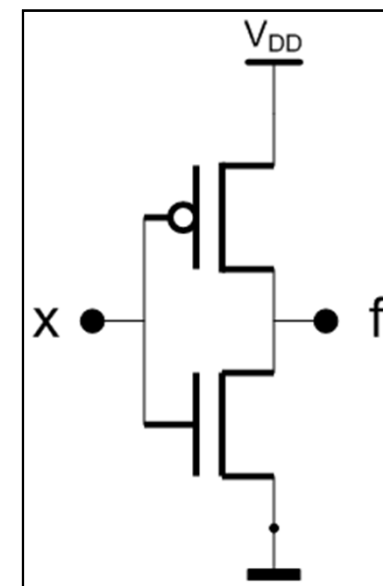
IEC-symbol



Amerikansk symbol



CMOS
(Complementary MOS)



sannings-
tabell

p	¬p
F	S
S	F

funktions-
tabell

x	f=x'
0	1
1	0

logiknivå

kretsnivå

Disjunktion, "ELLER" → OR-grind

satslogik

Boolesk algebra

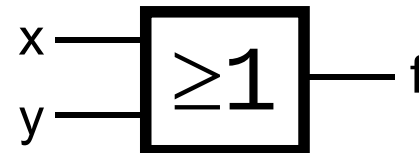
sannings-
tabell

p	q	$p \vee q$
F	F	F
F	S	S
S	F	S
S	S	S

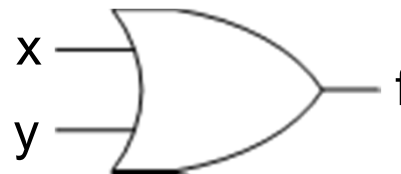
funktions-
tabell

x	y	$f = x + y$
0	0	0
0	1	1
1	0	1
1	1	1

IEC-symbol

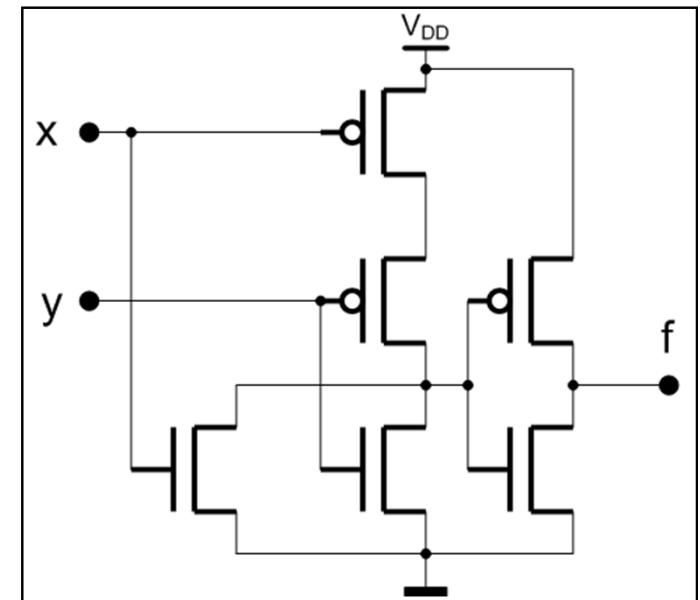


Amerikansk symbol



logiknivå

*CMOS
(Complementary
MOS)*



kretsnivå

Konjunktion, "OCH" → AND-grind

satslogik

Boolesk algebra

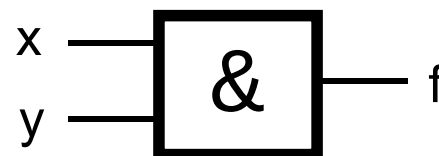
sannings-
tabell

p	q	$p \wedge q$
F	F	F
F	S	F
S	F	F
S	S	S

funktions-
tabell

x	y	$f = x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

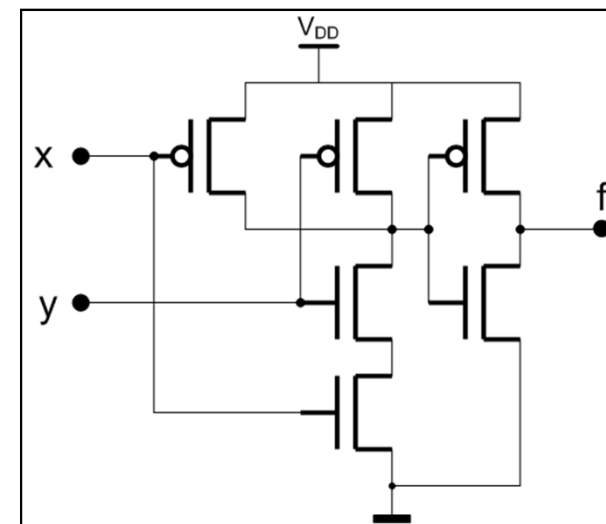
IEC-symbol



Amerikansk symbol



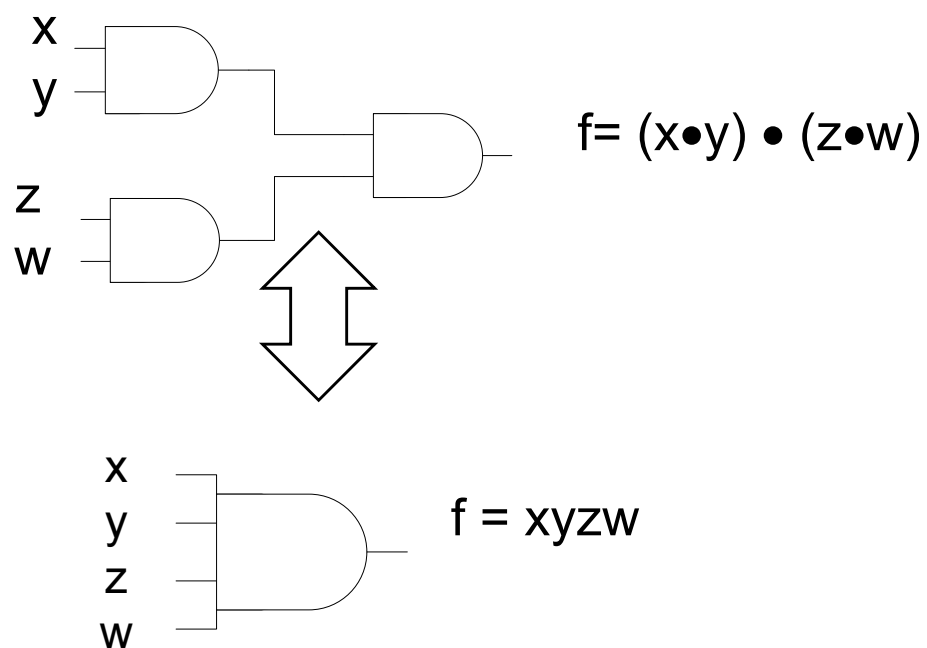
CMOS
(Complementary MOS)



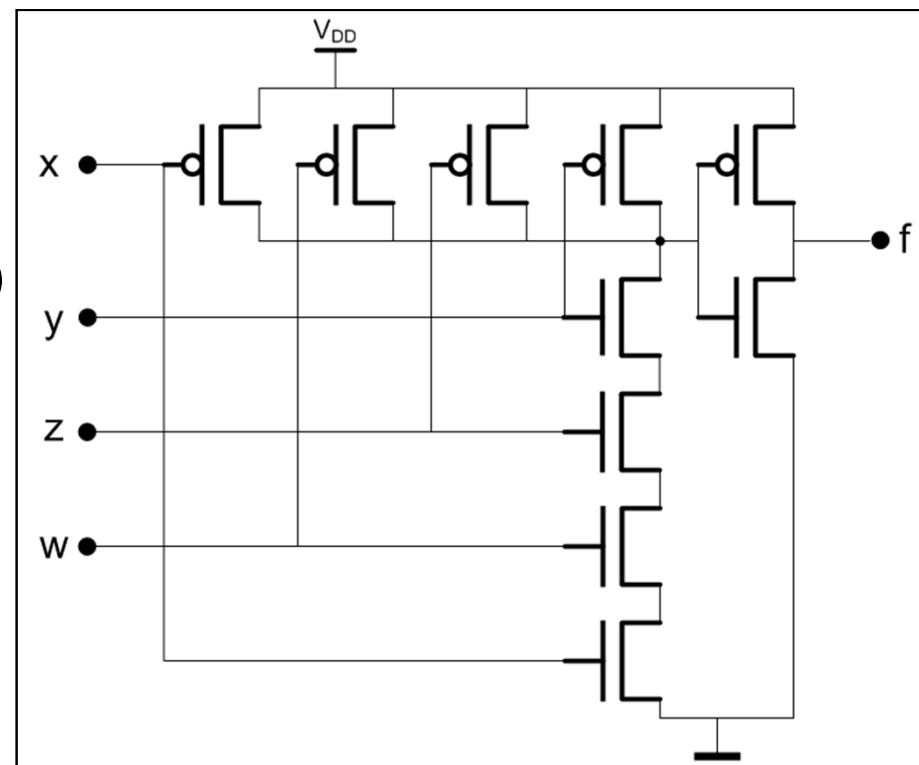
kretsnivå

logiknivå

Antalet ingångar kan utökas

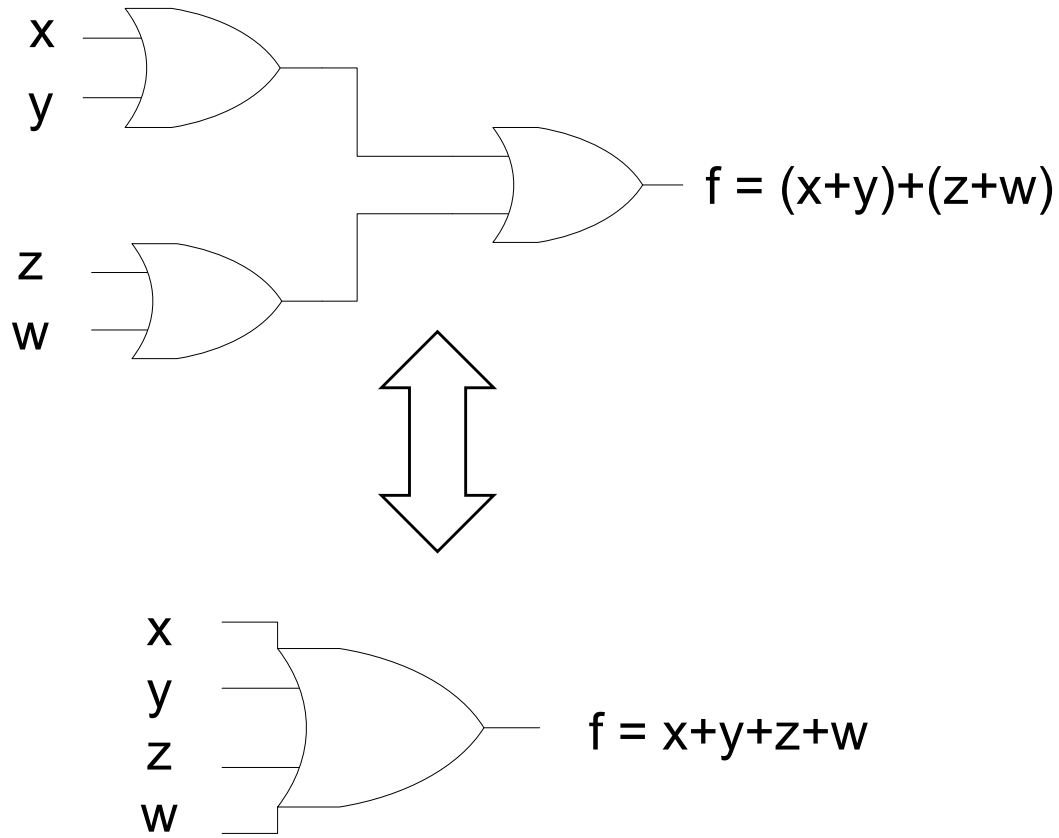


logiknivå

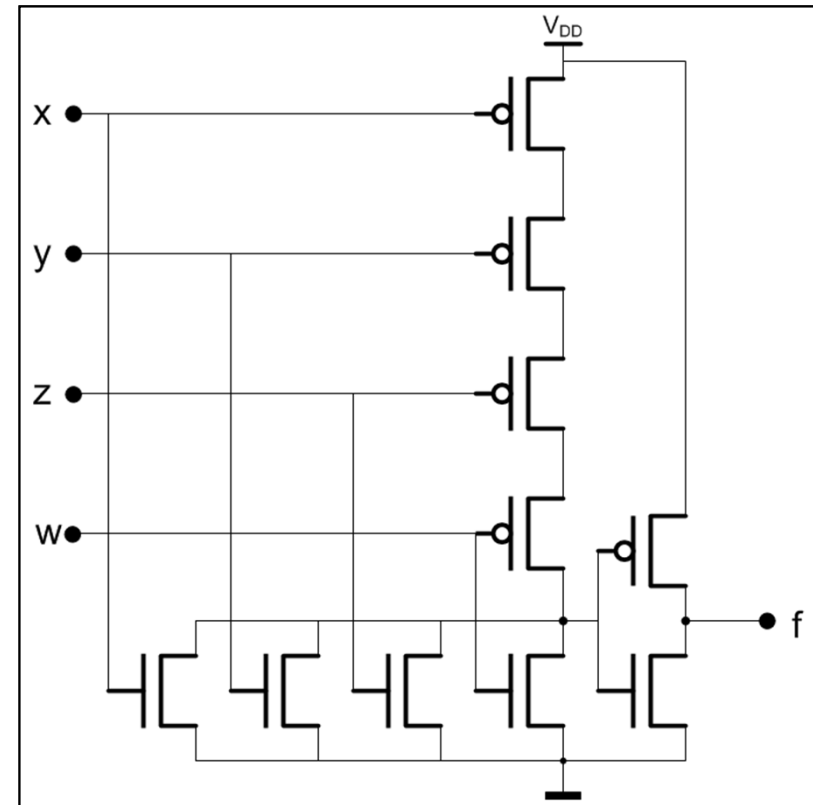


kretsnivå

Antal ingångar (fan-in), begränsas av använd kretsteknologi.



logiknivå



kretsnivå

Negerad konjunktion, "ICKE-OCH" → NAND-grind

satslogik

Boolesk algebra

sannings-
tabell

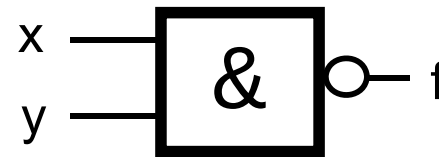
funktions-
tabell

p	q	$\neg(p \wedge q)$
F	F	S
F	S	S
S	F	S
S	S	F

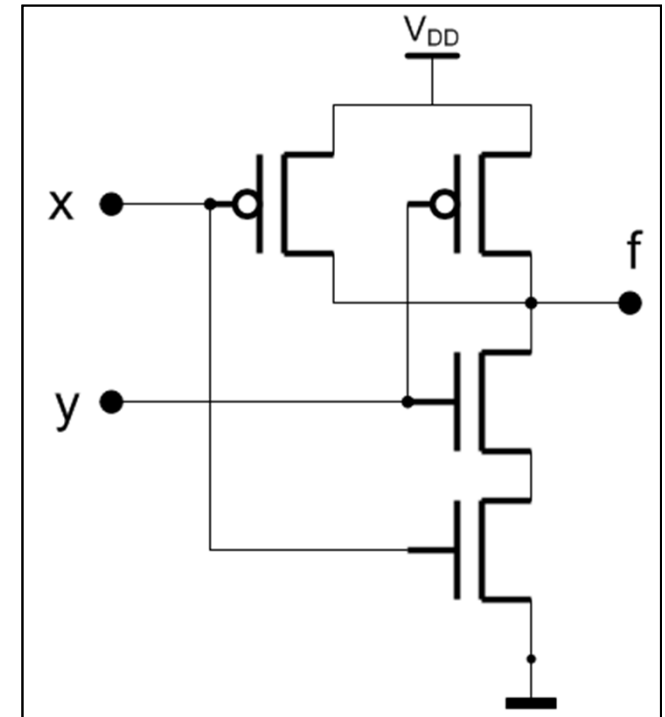
x	y	$f = (x \bullet y)'$
0	0	1
0	1	1
1	0	1
1	1	0

logiknivå

IEC-symbol



Amerikansk symbol



kretsnivå

Negerad disjunktion, "ICKE-ELLER" → NOR-grind

satslogik

Boolesk algebra

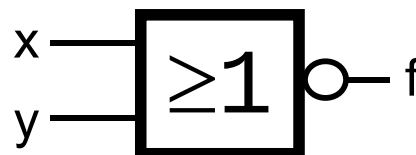
sannings-
tabell

p	q	$\neg(p \vee q)$
F	F	S
F	S	F
S	F	F
S	S	F

funktions-
tabell

x	y	$f = (x + y)'$
0	0	1
0	1	0
1	0	0
1	1	0

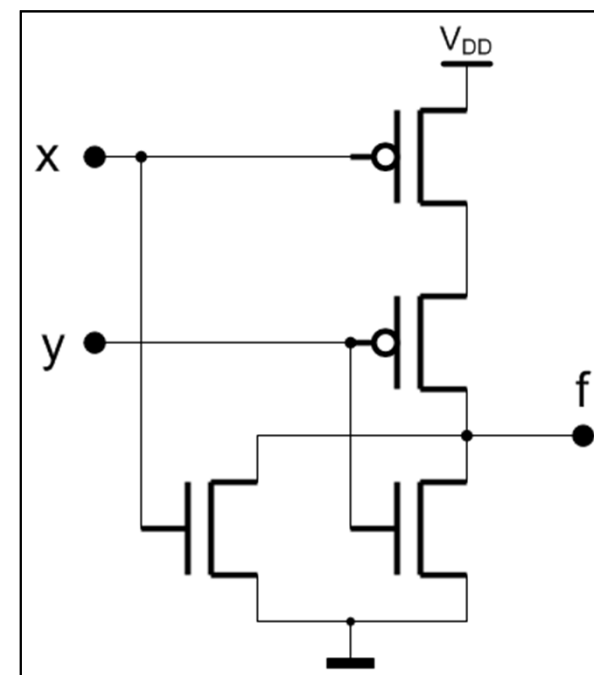
IEC-symbol



Amerikansk symbol



logiknivå



kretsnivå

(NOT) Exkluderande ELLER, (ICKE) XOR-grind

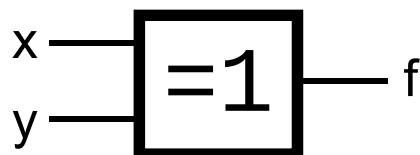
Definition:

$$x \oplus y = x'y + xy'$$

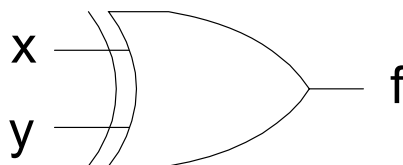
funktionstabell

x	y	f=x⊕y
0	0	0
0	1	1
1	0	1
1	1	0

IEC-symbol



Amerikansk symbol

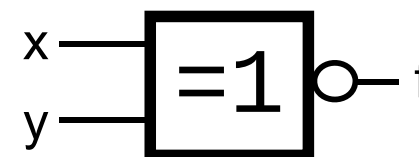


$$(x \oplus y)' = x'y' + xy$$

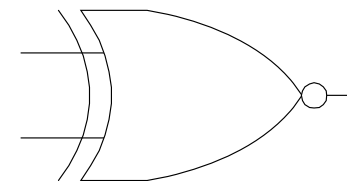
funktionstabell

x	y	f=(x⊕y)'
0	0	1
0	1	0
1	0	0
1	1	1

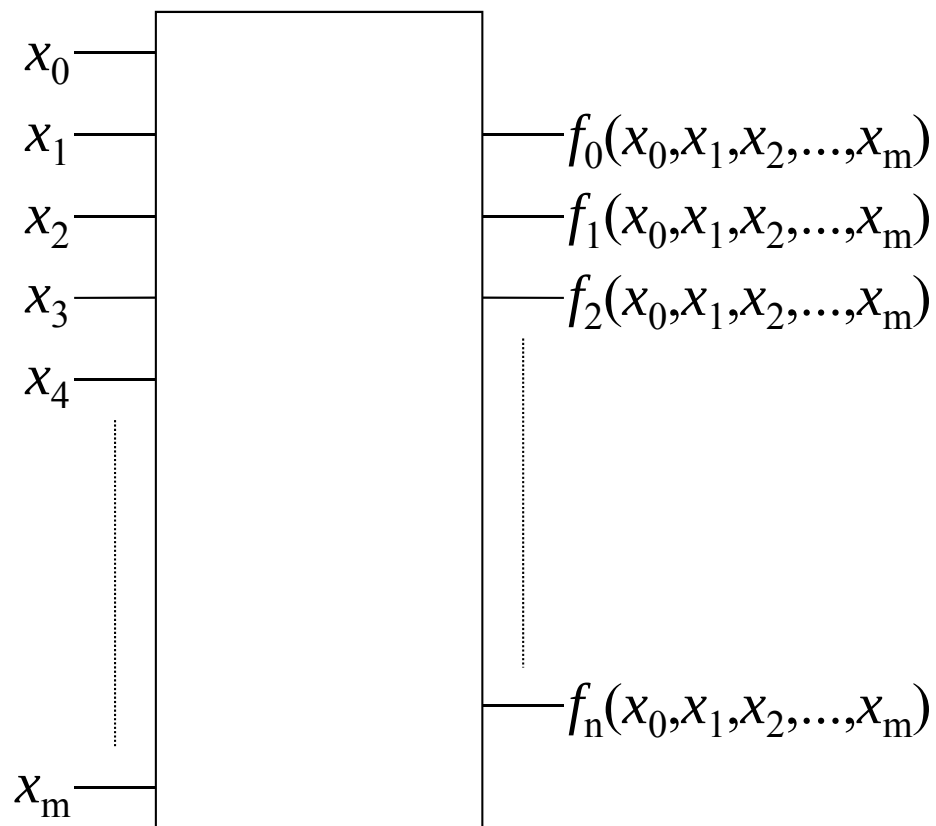
IEC-symbol



Amerikansk symbol



Kombinatoriska nät



Ett kombinatoriskt nät har fixt antal ingångar ($m-1$) och fixt antal utgångar ($n-1$).

Varje utgång har i varje ögonblick det värde som entydigt bestäms av insignalerna.

- Vi introducerar komponenter som byggblock i datorn.

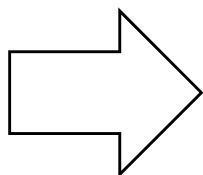
Binär addition

Uppställning för addition av två st. n -bitars tal:

$$\begin{array}{r}
 C_n \quad C_{n-1} \quad C_{n-2} \quad \dots \quad C_i \quad \dots \quad C_1 \\
 + \quad a_{n-1} \quad a_{n-2} \quad \dots \quad a_i \quad \dots \quad a_1 \quad a_0 \\
 \quad b_{n-1} \quad b_{n-2} \quad \dots \quad b_i \quad \dots \quad b_1 \quad b_0 \\
 \hline
 S_n \quad S_{n-1} \quad S_{n-2} \quad \dots \quad S_i \quad \dots \quad S_1 \quad S_0
 \end{array}$$

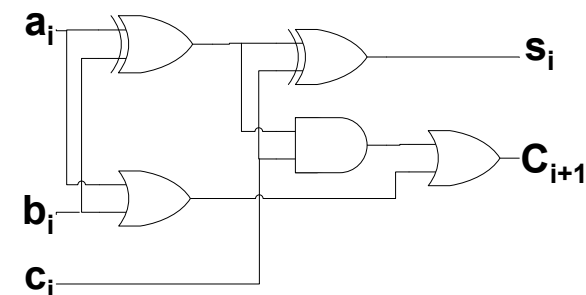
vi koncentrerar oss på
addition
av *en* bit, säg bit i .

$$\begin{array}{r}
 C_{i+1} \quad C_i \\
 + \quad X_i \quad Y_i \\
 \hline
 = \quad S_i
 \end{array}$$

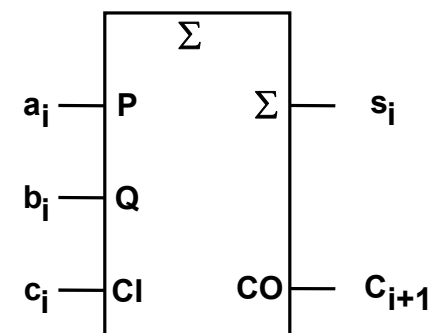


a_i	b_i	c_i	s_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

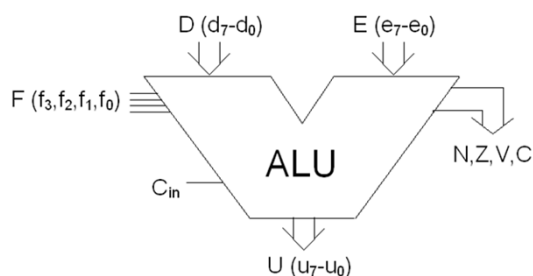
Heladderaren,
ett kombinatoriskt nät
med 3 insignaler och
2 utsignaler



Symbol för heladderare



ALU-funktioner



funktion				operation	utsignaler											
f_3	f_2	f_1	f_0		u_7	u_6	u_5	u_4	u_3	u_2	u_1	u_0	N	Z	V	C
0	0	0	0	RTN	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	1	$U = FD_{16}$	1	1	1	1	1	1	0	1	1	0	0	0
0	0	1	0	$U = FE_{16}$	1	1	1	1	1	1	1	0	1	0	0	0
0	0	1	1	$U = FF_{16}$	1	1	1	1	1	1	1	1	1	0	0	0
0	1	0	0	$U = E$	e_7	e_6	e_5	e_4	e_3	e_2	e_1	e_0	u_7	(1)	0	0
0	1	0	1	$U = D_{1k} + C_{in}$									u_7	(1)	(8)	(7)
0	1	1	0	$U = D \vee E$	$d_7 \vee e_7$	$d_6 \vee e_6$	$d_5 \vee e_5$	$d_4 \vee e_4$	$d_3 \vee e_3$	$d_2 \vee e_2$	$d_1 \vee e_1$	$d_0 \vee e_0$	u_7	(1)	0	0
0	1	1	1	$U = D \wedge E$	$d_7 \wedge e_7$	$d_6 \wedge e_6$	$d_5 \wedge e_5$	$d_4 \wedge e_4$	$d_3 \wedge e_3$	$d_2 \wedge e_2$	$d_1 \wedge e_1$	$d_0 \wedge e_0$	u_7	(1)	0	0
1	0	0	0	$U = D \oplus E$	$d_7 \oplus e_7$	$d_6 \oplus e_6$	$d_5 \oplus e_5$	$d_4 \oplus e_4$	$d_3 \oplus e_3$	$d_2 \oplus e_2$	$d_1 \oplus e_1$	$d_0 \oplus e_0$	u_7	(1)	0	0
1	0	0	1	$U = D + C_{in}$									u_7	(1)	(2)	(3)
1	0	1	0	$U = D + FF_{16}$									u_7	(1)	(2)	(3)
1	0	1	1	$U = D + E + C_{in}$									u_7	(1)	(2)	(3)
1	1	0	0	$U = D + E_{1k} + C_{in}$									u_7	(1)	(2)	(3)
1	1	0	1	$U = D \ll 1 (C_{in})$	d_6	d_5	d_4	d_3	d_2	d_1	d_0	C_{in}	u_7	(1)	0	(4)
1	1	1	0	$U = (C_{in}) 1 \gg D$	C_{in}	d_7	d_6	d_5	d_4	d_3	d_2	d_1	u_7	(1)	(6)	(5)
1	1	1	1	$U = (d_7) 1 \gg D$	d_7	d_7	d_6	d_5	d_4	d_3	d_2	d_1	u_7	(1)	0	(5)

(1) $Z = \overline{u_7} \wedge \overline{u_6} \wedge \overline{u_5} \wedge \overline{u_4} \wedge \overline{u_3} \wedge \overline{u_2} \wedge \overline{u_1} \wedge \overline{u_0}$, dvs. $Z=1$ då samtliga bitar i register U är 0, $Z=0$ annars.

(2) $V = (\overline{u_7} \wedge d_7 \wedge e_7) \vee (u_7 \wedge \overline{d_7} \wedge \overline{e_7})$, dvs. V -flaggan sätts enligt reglerna för tvåkomplementsaritmetik.

(3) $C = c_8$, dvs. carry ut från additionen av de mest signifikanta siffrorna.

(4) $C =$ utskiftad bit, dvs. bit d_7 före vänsterskiftet.

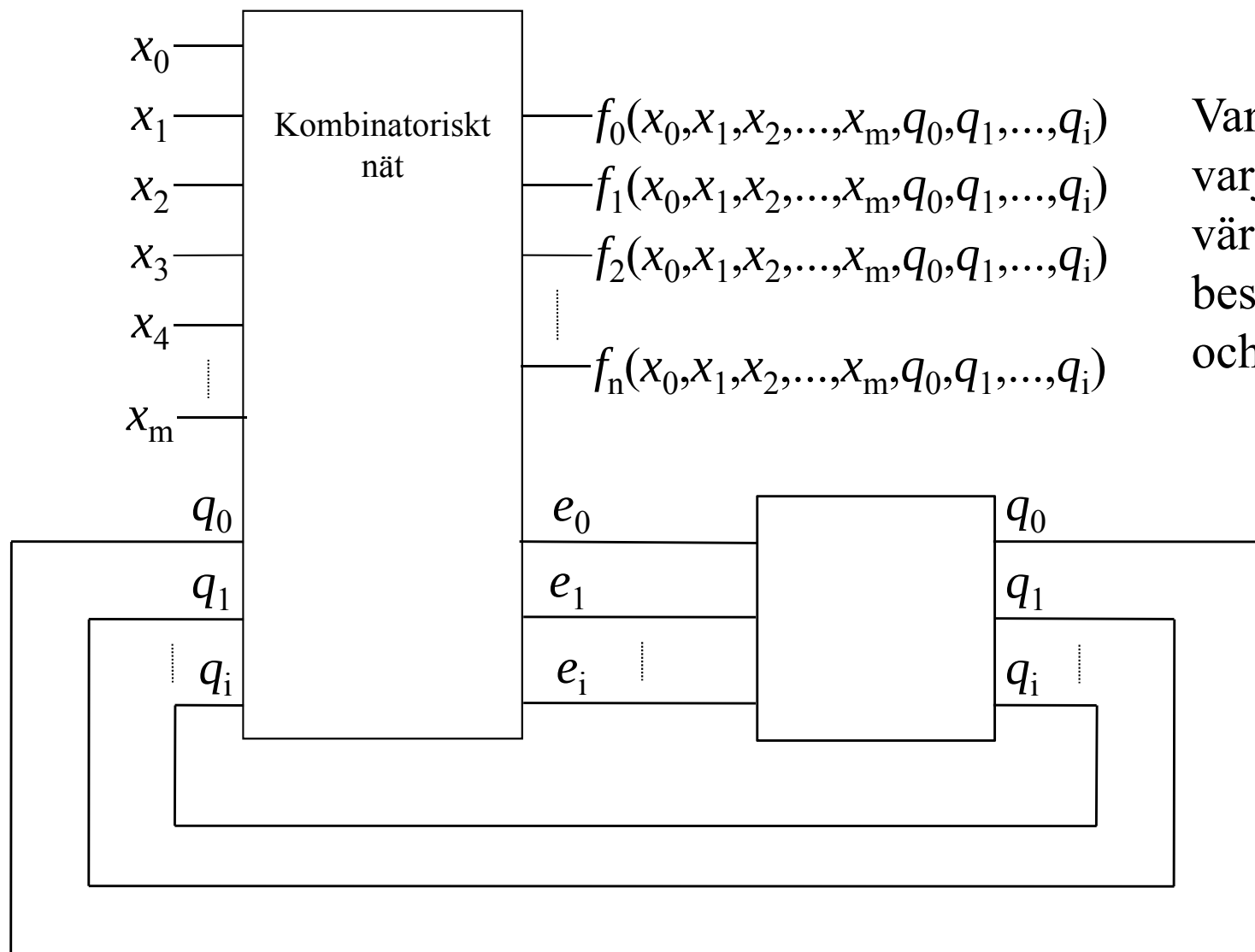
(5) $C =$ utskiftad bit, dvs. bit d_0 före högerskiftet.

(6) $V = C_{in} \oplus d_7$ före skift, dvs. sätts till 1 om skiftet föranleder teckenbyte.

(7) C ettställs om $D=0$, C nollställs annars.

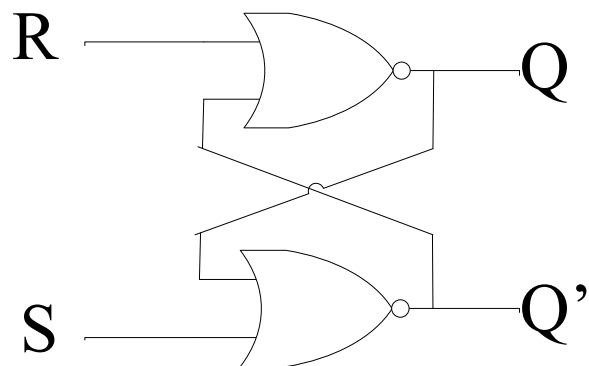
(8) V ettställs om $D=(10000000)_2$, V nollställs annars.

Sekvensnät



Varje utgång har i varje ögonblick det värde som entydigt bestäms av insignaler och tillstånd.

Latch (låskrets)



Q kan ettställas eller nollställas för att därefter behålla värdet.

Kopplingen är ett "minneselement" och kallas SR-latch.

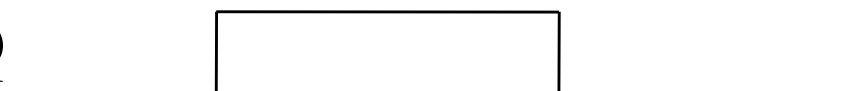
R=Reset



S=Set



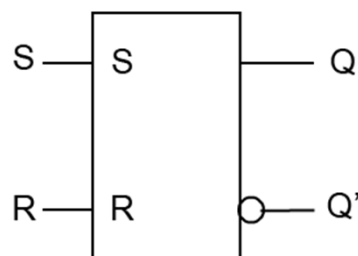
Q



Funktionstabell

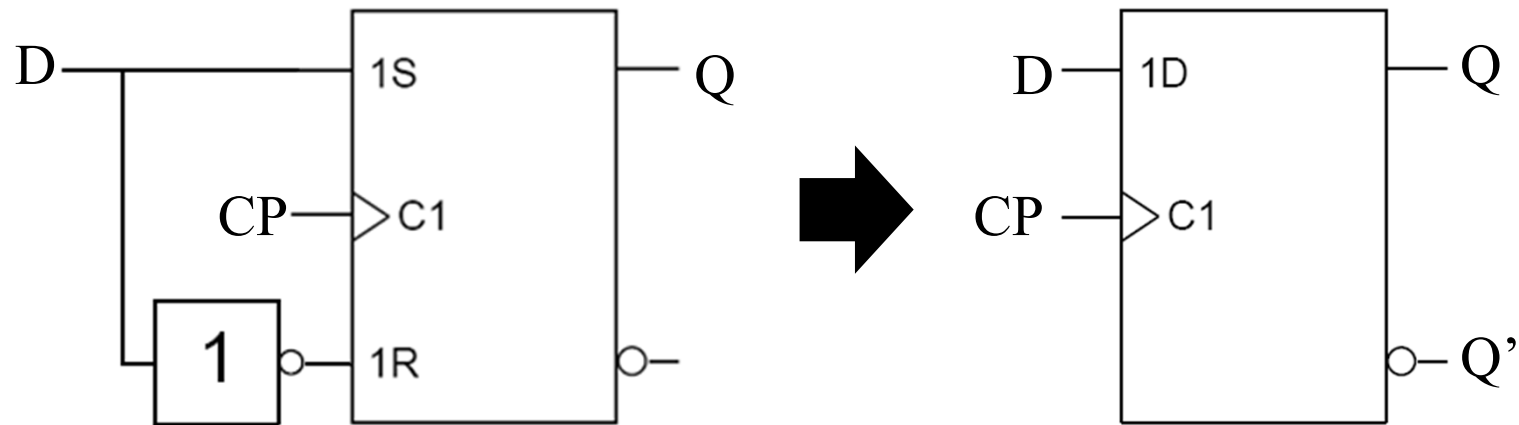
S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	*

Symbol



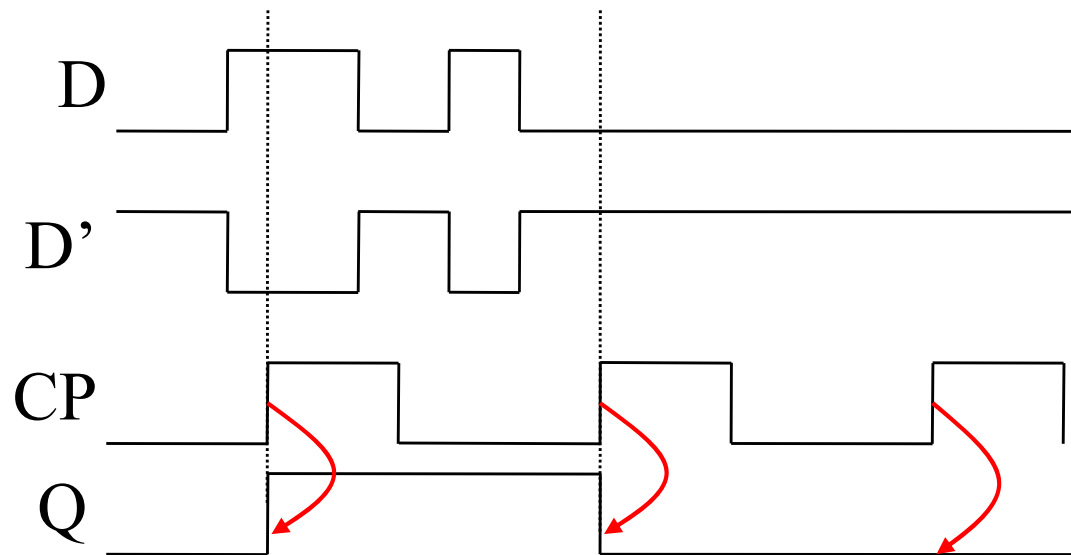
Asynkront minneselement

Flanktriggad D-vippa



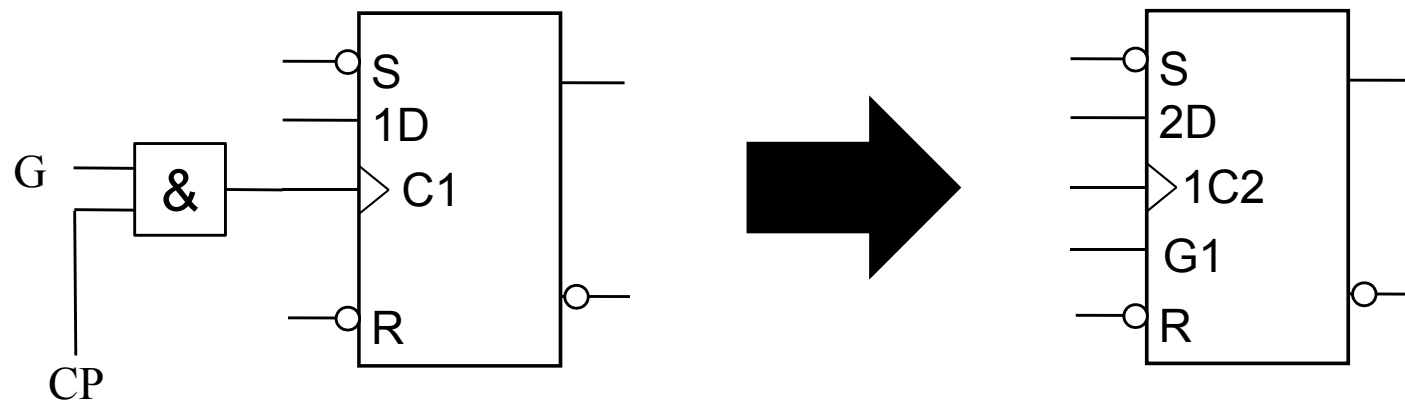
Funktions-
tabell

D	Q ⁺
0	0
1	1



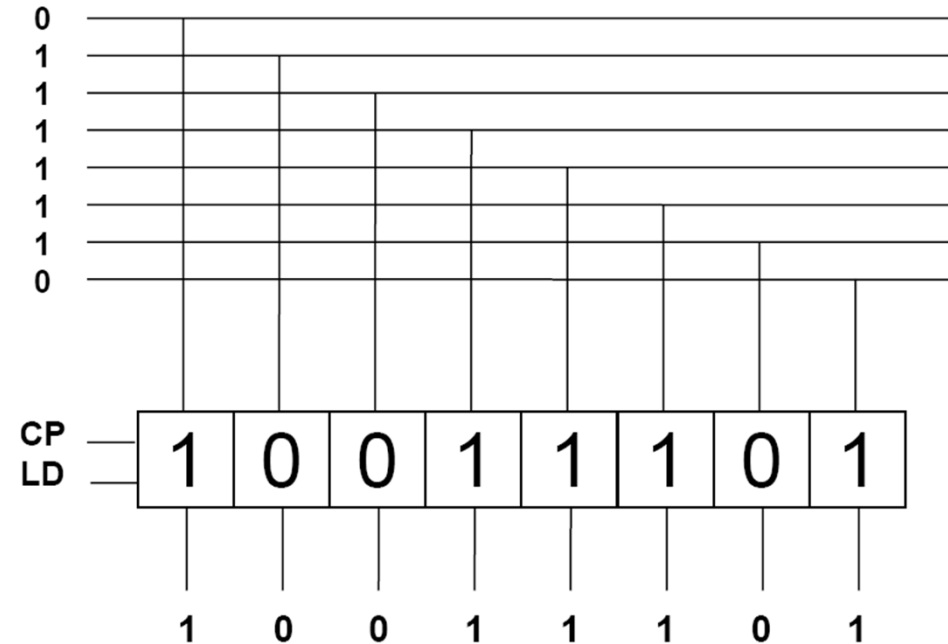
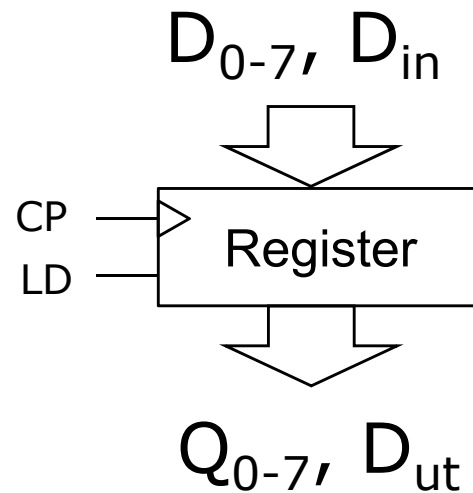
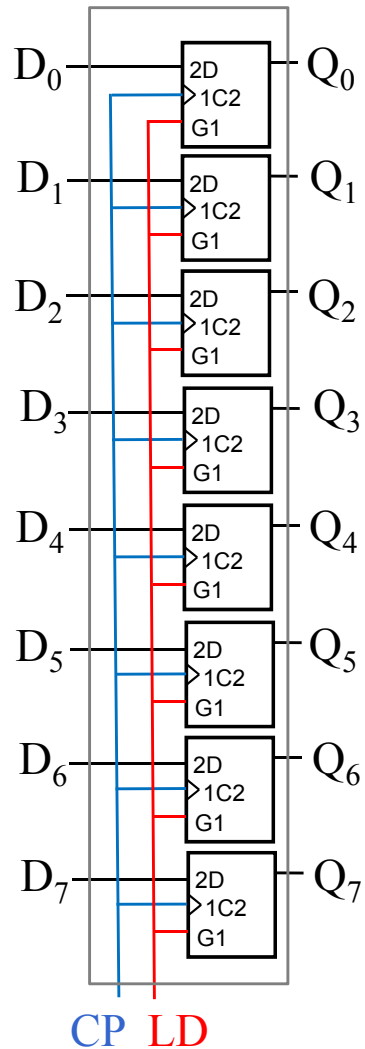
Synkront minneselement

D-vippa med "Load Enable"



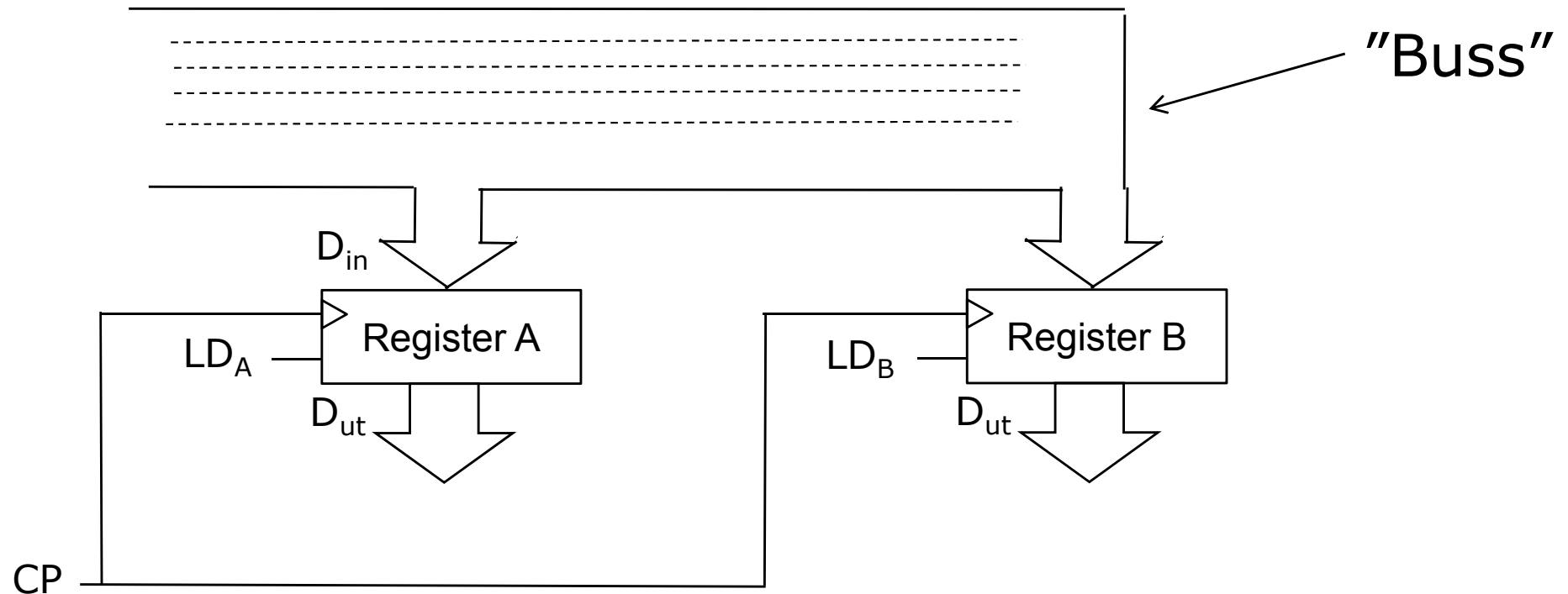
Styrsignalen G ("Gate") kan användas för att "strypa" klockpulsen.
 Då $G=0$ behåller vippan sitt värde oavsett vad som finns på D-ingången

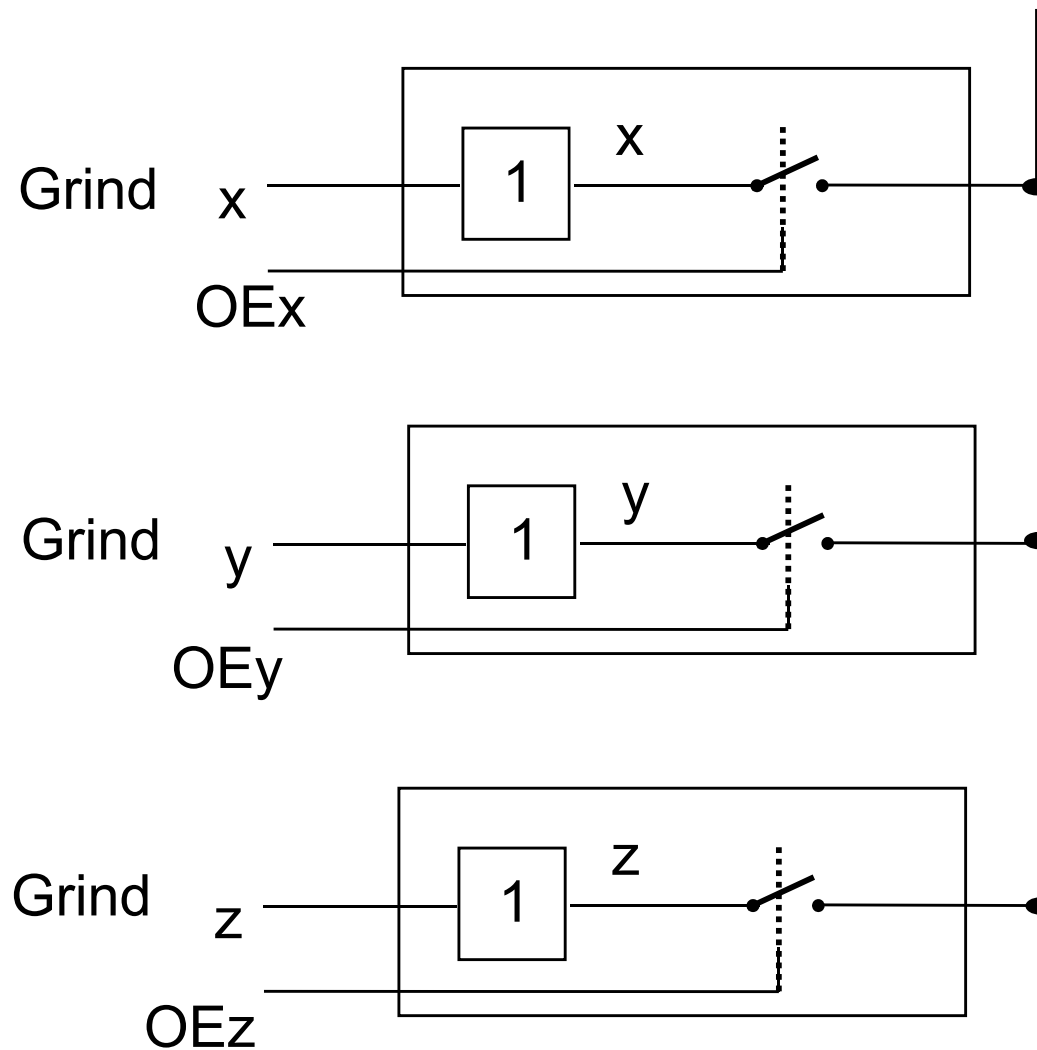
Laddbart register



Registrets innehåll påverkas av ingångarnas nivåer först vid en klockpuls OCH om LD-signalen är aktiv.

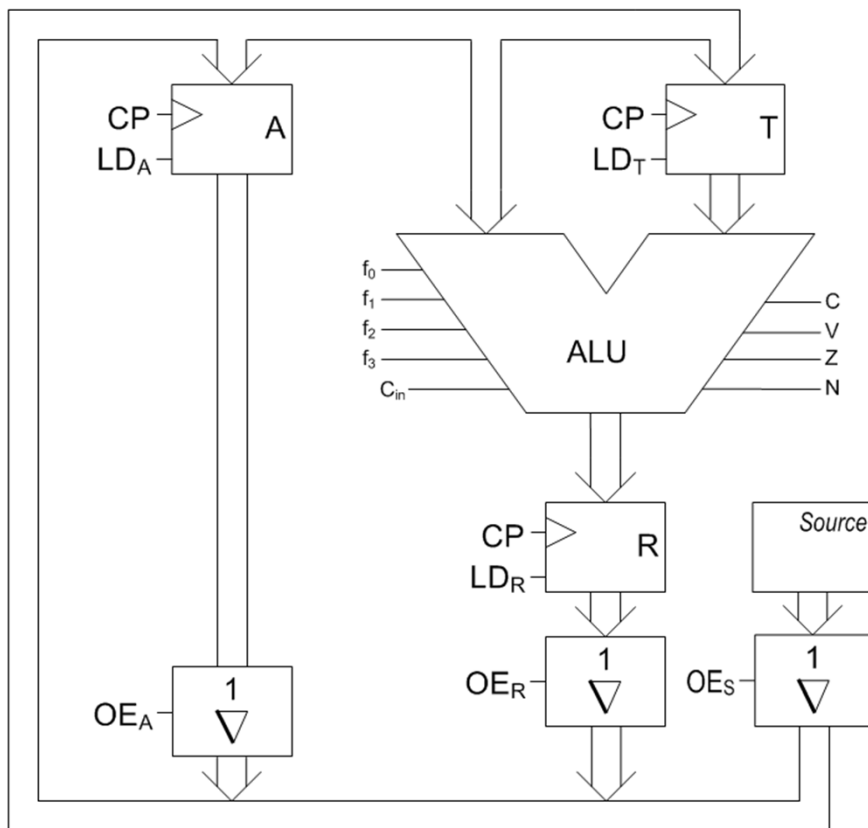
"Dataväg" – sammankopplade register



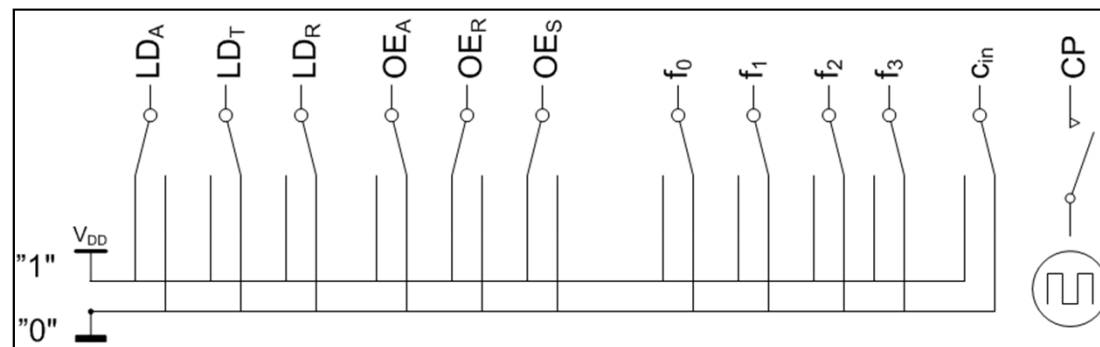
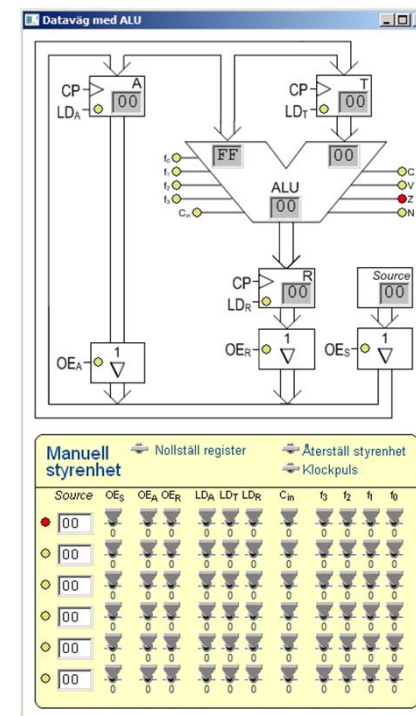


Högst **en** OE-signal aktiv åt gången...

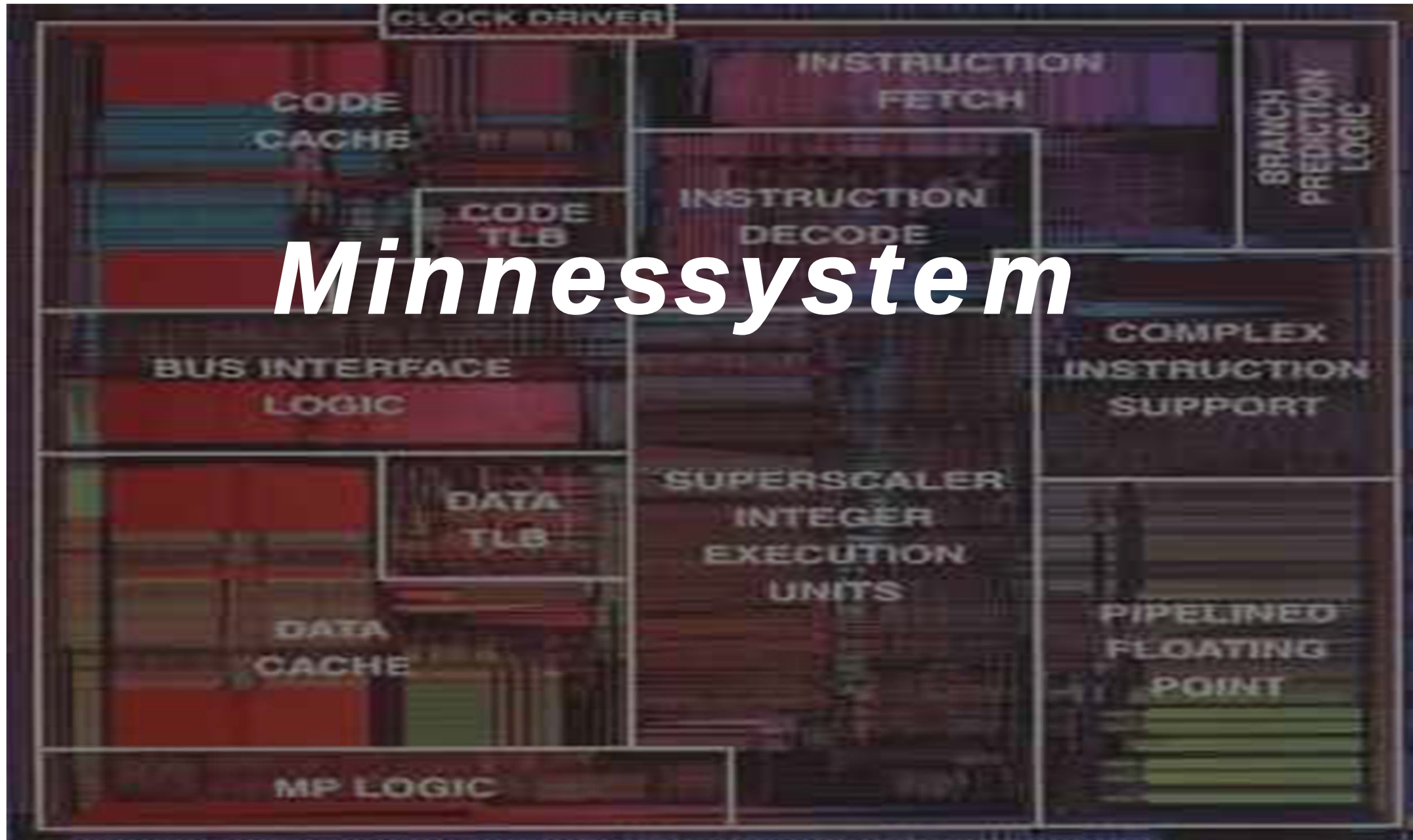
Dataväg med ALU



funktion				operation
f_3	f_2	f_1	f_0	
0	0	0	0	RTN
0	0	0	1	$U = 0$
0	0	1	0	$U = FD_{16}$
0	0	1	1	$U = FE_{16}$
0	1	0	0	$U = E$
0	1	0	1	$U = D_{1k} + C_{in}$
0	1	1	0	$U = D \vee E$
0	1	1	1	$U = D \wedge E$
1	0	0	0	$U = D \oplus E$
1	0	0	1	$U = D + C_{in}$
1	0	1	0	$U = D + FF_{16}$
1	0	1	1	$U = D + E + C_{in}$
1	1	0	0	$U = D + E_{1k} + C_{in}$
1	1	0	1	$U = D \ll 1 (C_{in})$
1	1	1	0	$U = (C_{in}) 1 \gg D$
1	1	1	1	$U = (d_7) 1 \gg D$

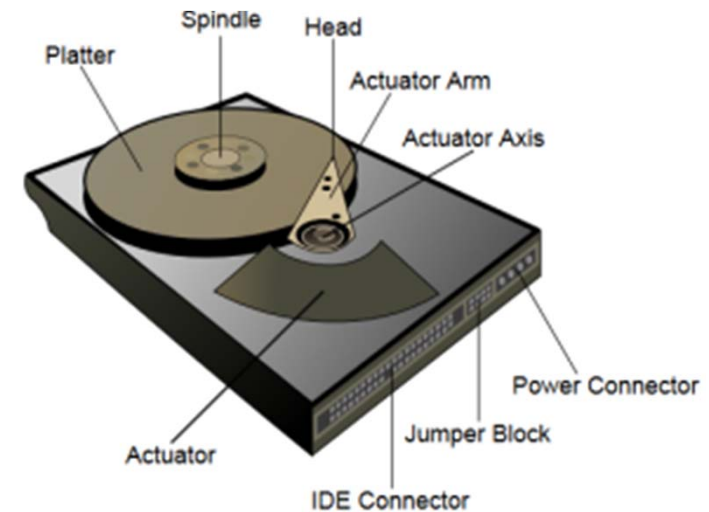
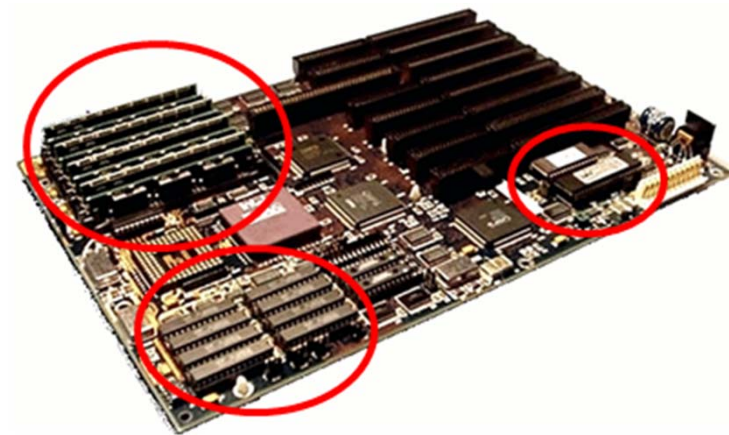


Manöverpanel – strömställare för styrsignaler.



Minnessystem

Minneshierarki,
minnestyper och
teknologier



Huvudtyper – Random, block och sequential access

- Primärminnen

S-RAM, D-RAM, ROM, EPROM

"Random Access"

- Sekundärminnen

SSD (Solid State Drive)

Hårddisk

Flexskiveenheten

"Block Access"

- Tertiärminnen

Bandstationer

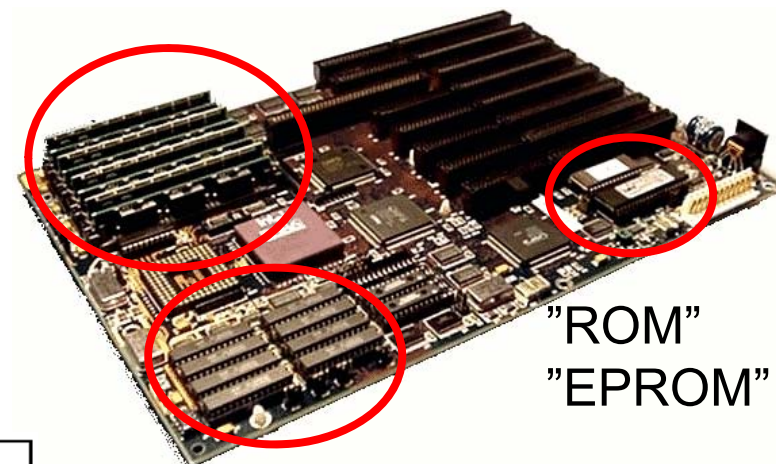
Optiska lagringsmedia

"Sequential Access"

Random Access Memory (RAM)

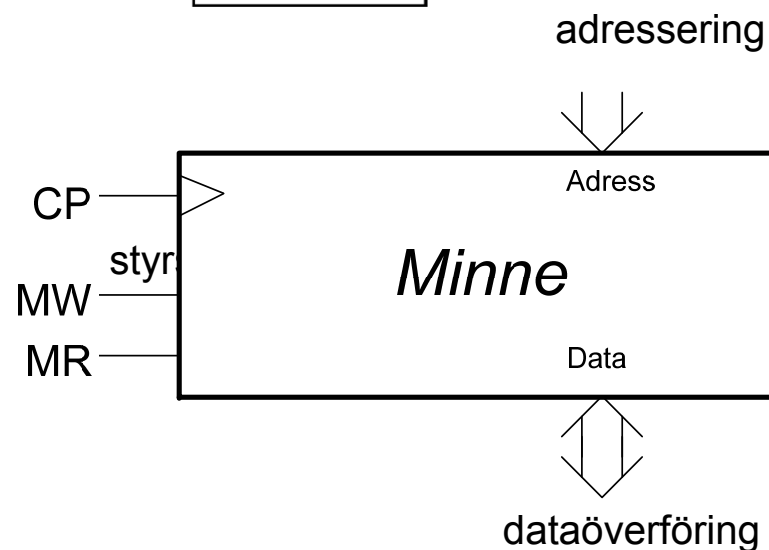
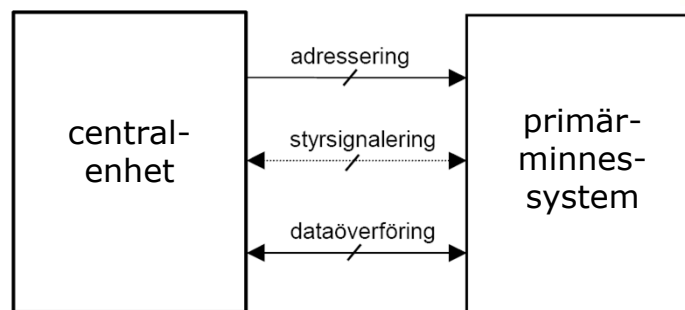
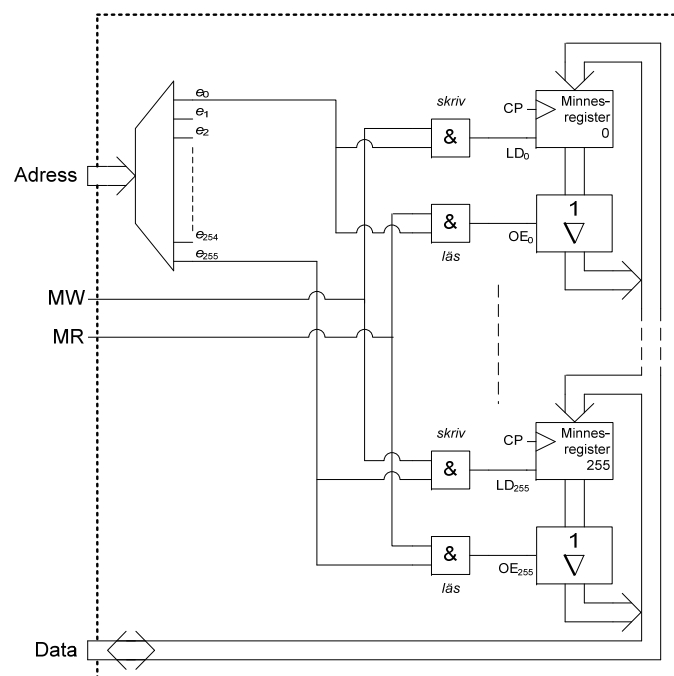
Alla minnets delar har samma åtkomsttid.

"DRAM"



"ROM"
"EPROM"

"SRAM"

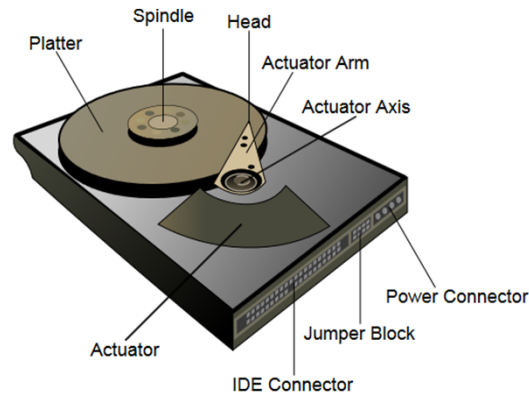
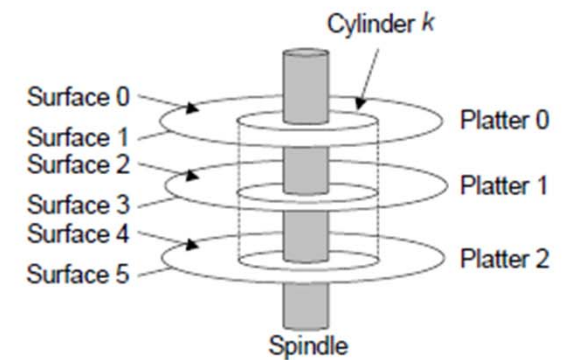
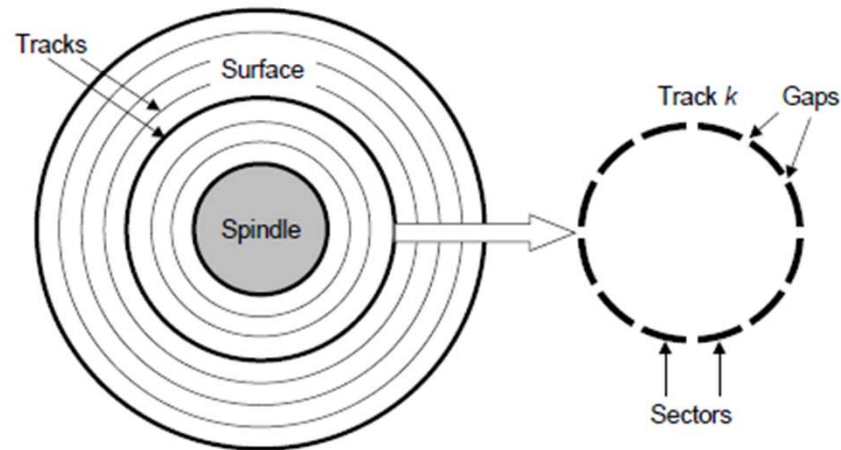


Block Access Memory

Roterande skivor

Åtkomsttid:

"spin up" + söktid(spår) + rotera fram rätt sektor + överför block till primärminne.

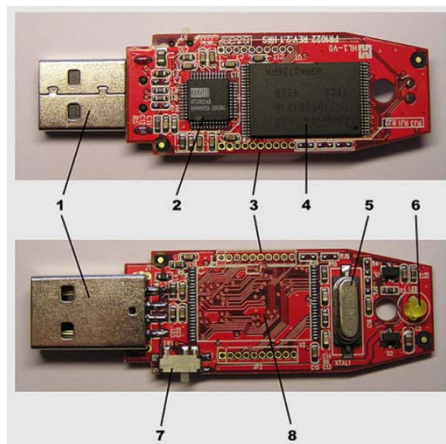


Block Access Memory

FLASH-minnen

”USB-minne”

standardiserat
seriellt gränssnitt



Åtkomsttid:

Beror på – gränssnitt.

Begränsat eftersom data nås efter ett begränsat antal operationer



”Minneskort”
olika enkla
parallella
gränssnitt

”Solid State Drive”

standardiserat gränssnitt
som kan ersätta
”roterande skivor”



Sequential Access Memory

Åtkomsttid ej förutsägbar:

Man måste söka upp rätt data sekventiellt.
Det finns ingen direkt "block-sök" algoritm för dessa media.

Åtkomsttiden är inte begränsad eftersom man potentiellt måste följa många anvisningar innan data har hittats.

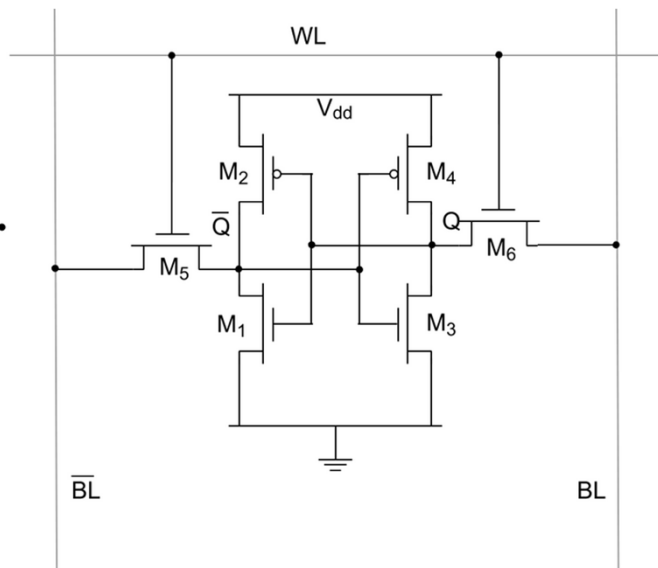


Teknologier

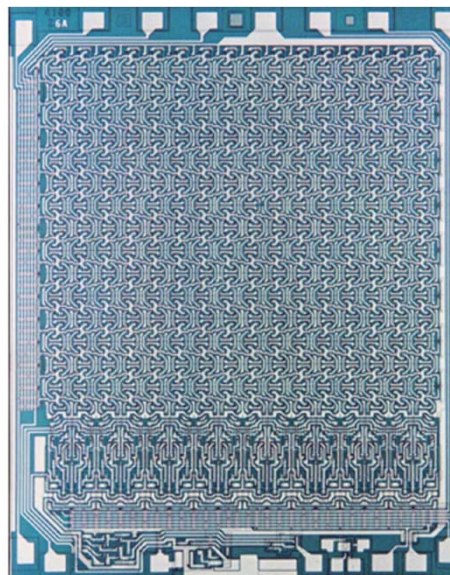
- Skriv/läs-minnen, kallas ibland "dataminne"
 - Halvledarminnen, SRAM/DRAM
- Läs-minnen, kallas ibland "Programminne"
 - Halvledarminne, ROM
 - Halvledarminne, FLASH

Halvledarminnen 1966

Statiskt RAM (SRAM) 6 transistorer/-bit

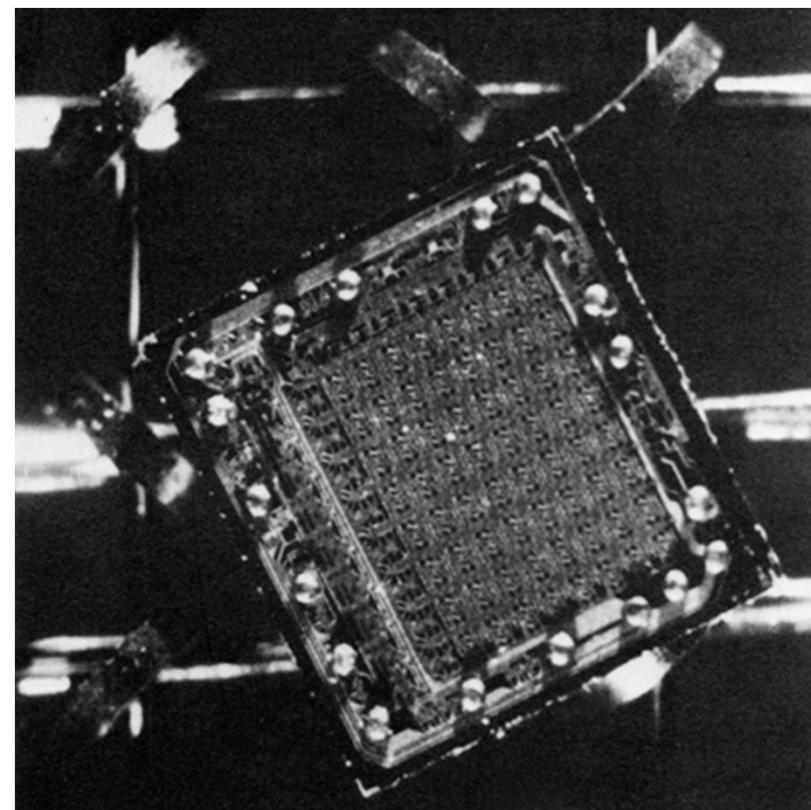


WL = 1, väljer denna cell.
(M₅ och M₆ leder).
Minneselementet har två stabila tillstånd:
BL=1 och BL' = 0 → "1"
BL=0 och BL' = 1 → "0"
(Jämför med SR-latch)

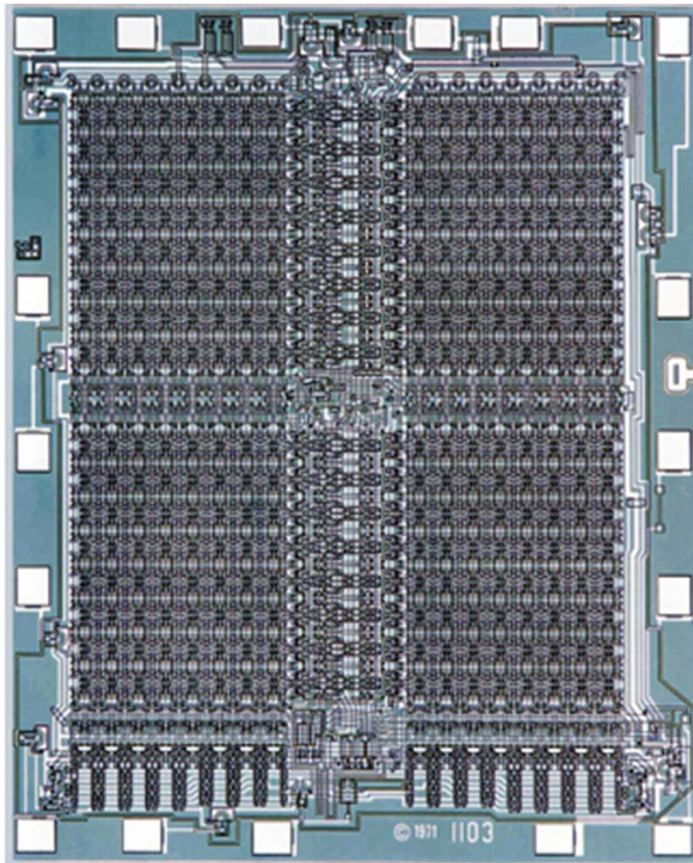


256-bit TTL
RAM (Fairchild)

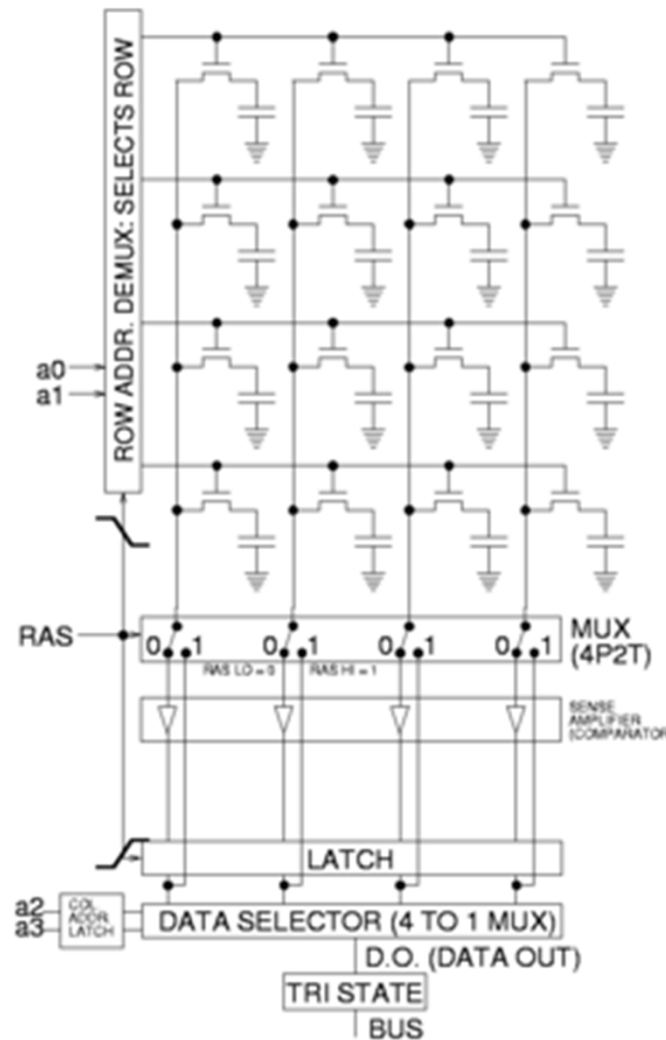
128-bit RAM
(IBM System/360 Model 145 primärminne)
jämförd med kärnminne.



Dynamiskt RAM (DRAM) 1970

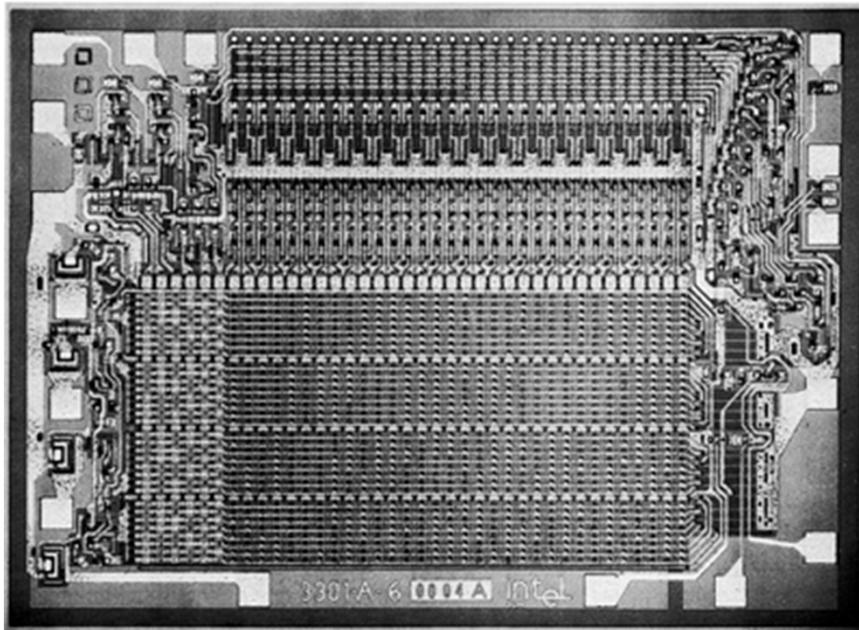


Intel i1103 1024-bit
Dynamiskt RAM

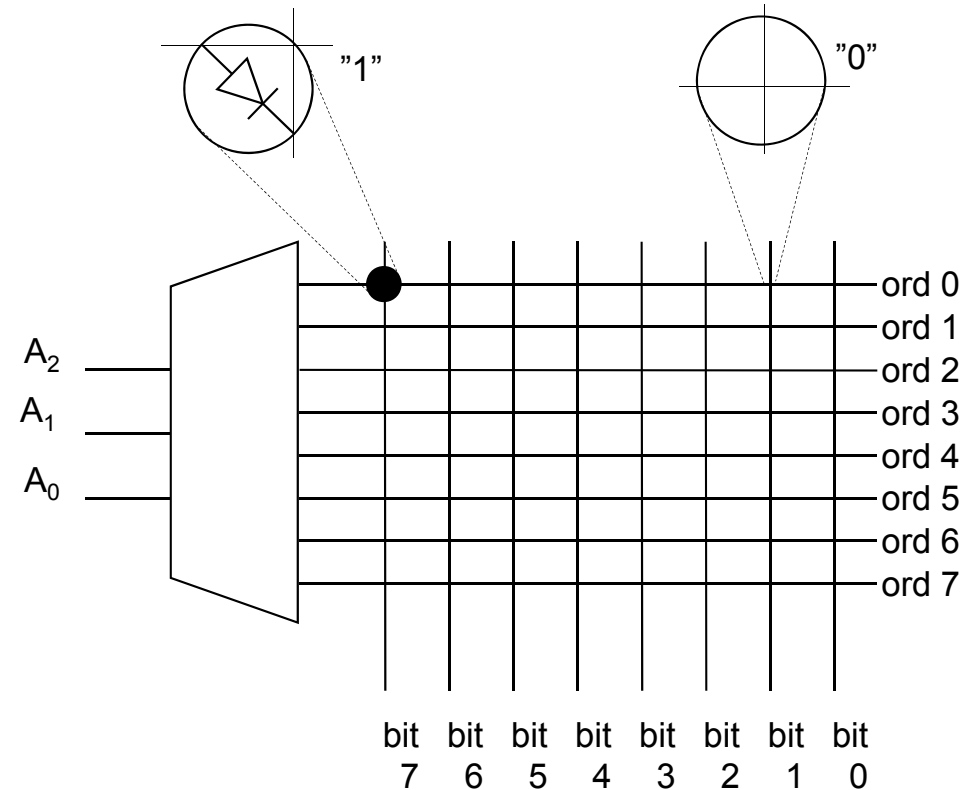


Endast en transistor krävs för att lagra en bit.
Kräver "refresh" eftersom kondensatorn laddas ur efter hand.
(Jämför med Williamstuben)

ROM (1965)



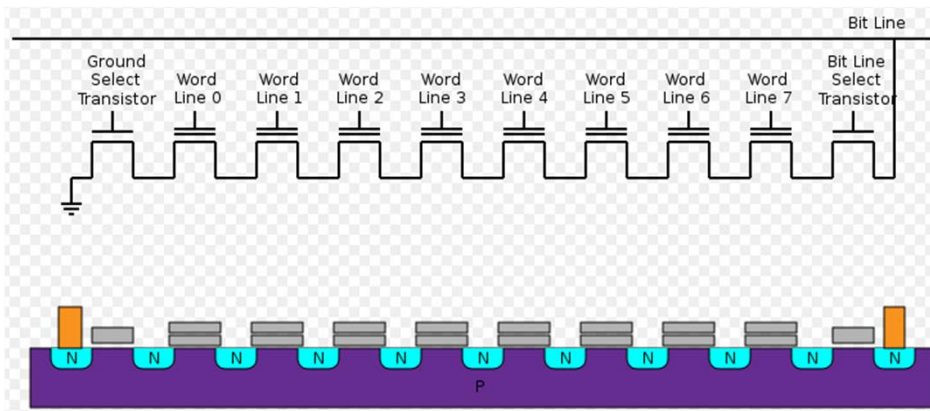
Intel 3301, 1024-bit ROM



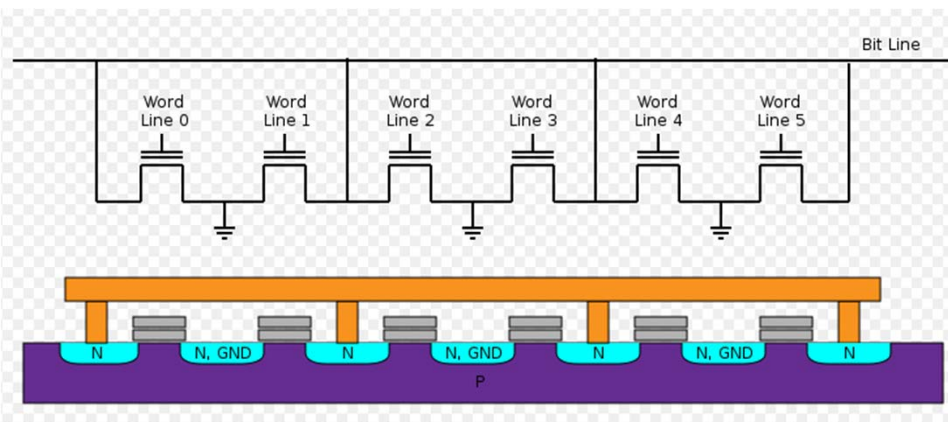
Dioder (ettor) placeras i diodmatrisens skärningar vid tillverkningsprocessen, "mask-programmerade"

FLASH 1988

”block access”



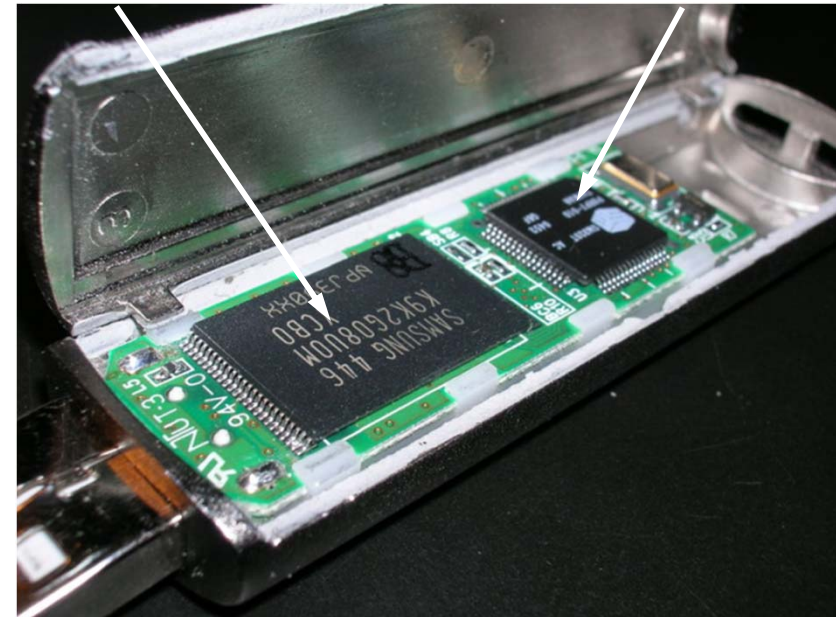
NAND-FLASH



NOR-FLASH

MINNE

STYRKRETS



”random access”

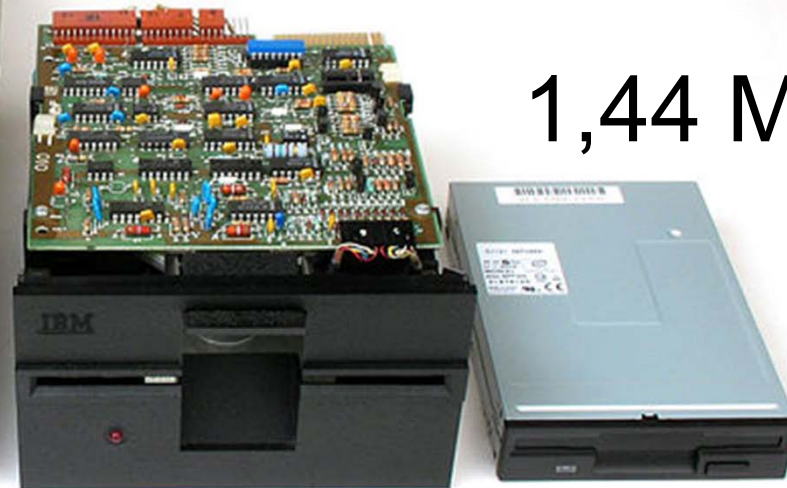
Flexskivan (1967)



90 kByte



1,44 MByte



360 kByte

Optisk disk(1965)

Laser Disc (1971)



Digital Versatile Disc(1995)

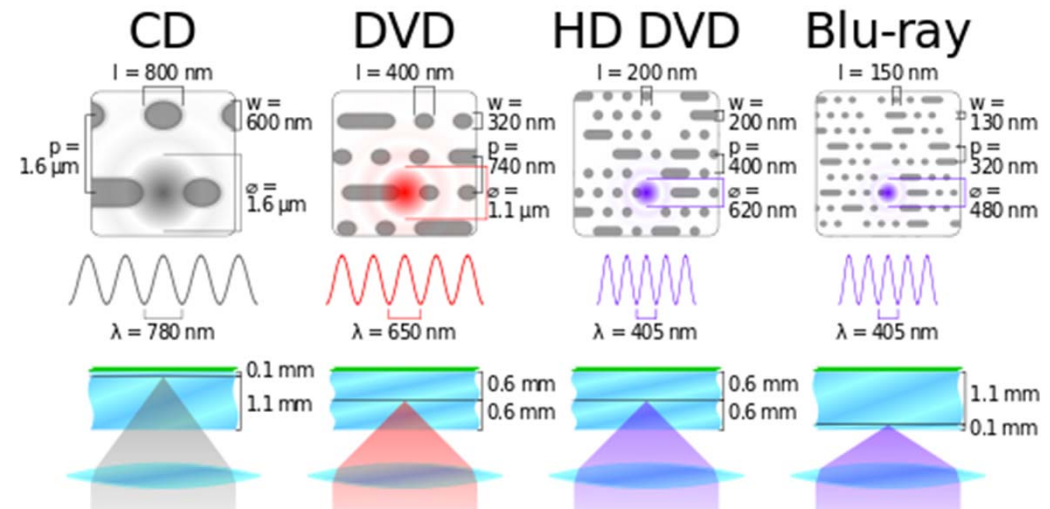
4,7-17 GByte



Compact Disc(1981)



650-900 MByte



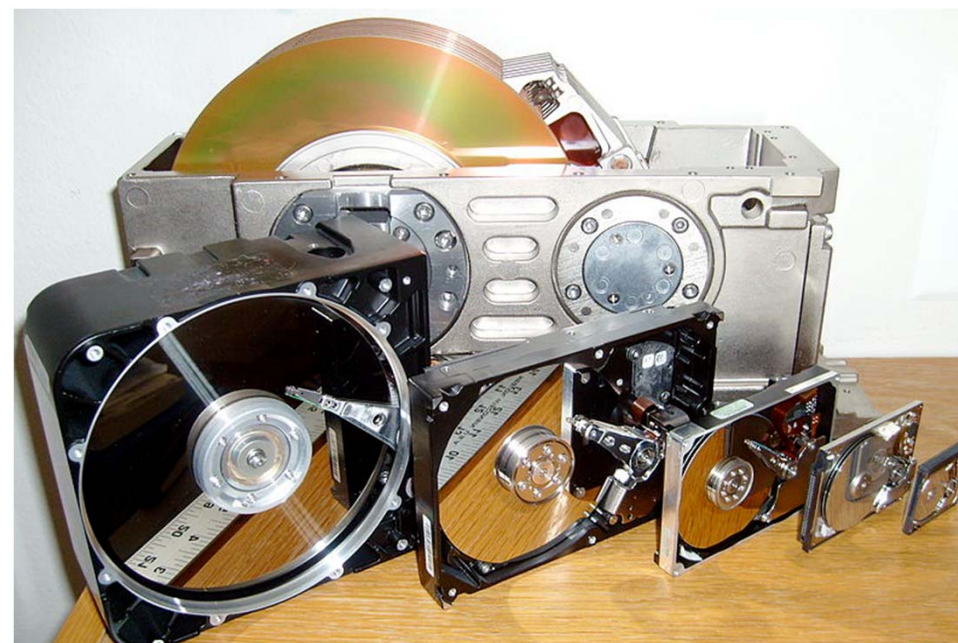
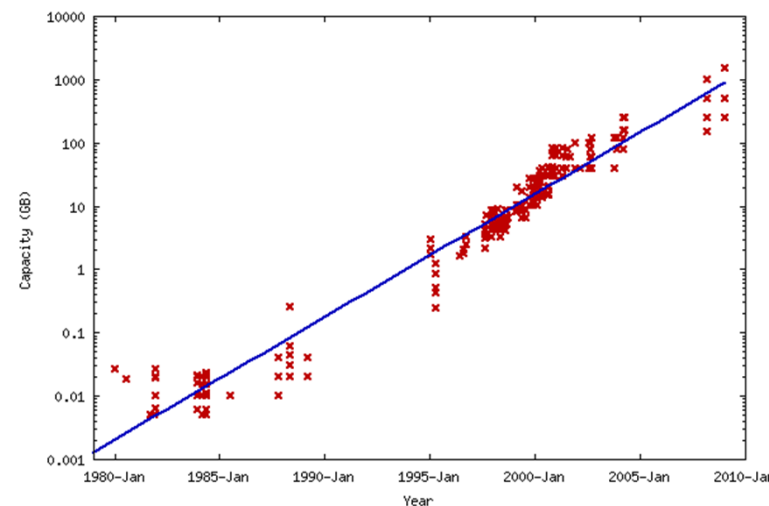
Hårddisken



IBM 350 (1956)

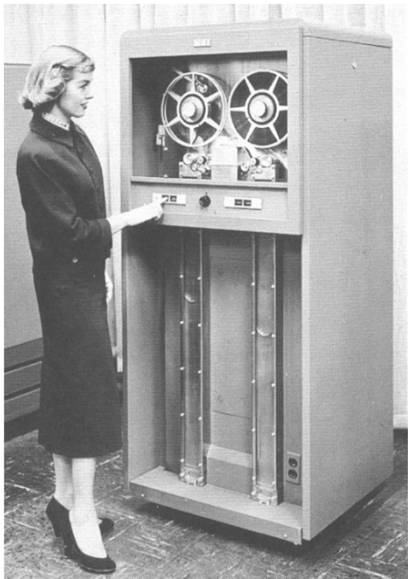


Kapacitet 5-20 MByte



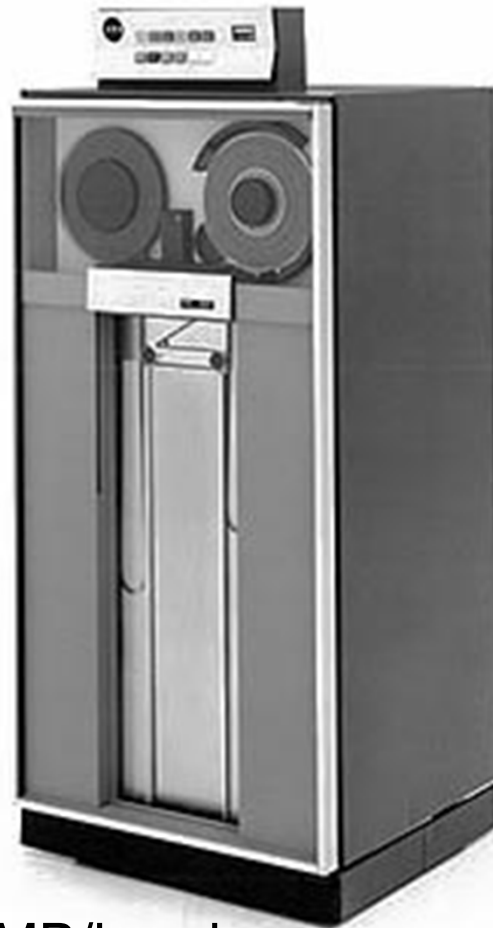
Bandstationer

IBM 727 (1953)



5,6 MB/band

IBM 3420 (1970)



20 MB/band

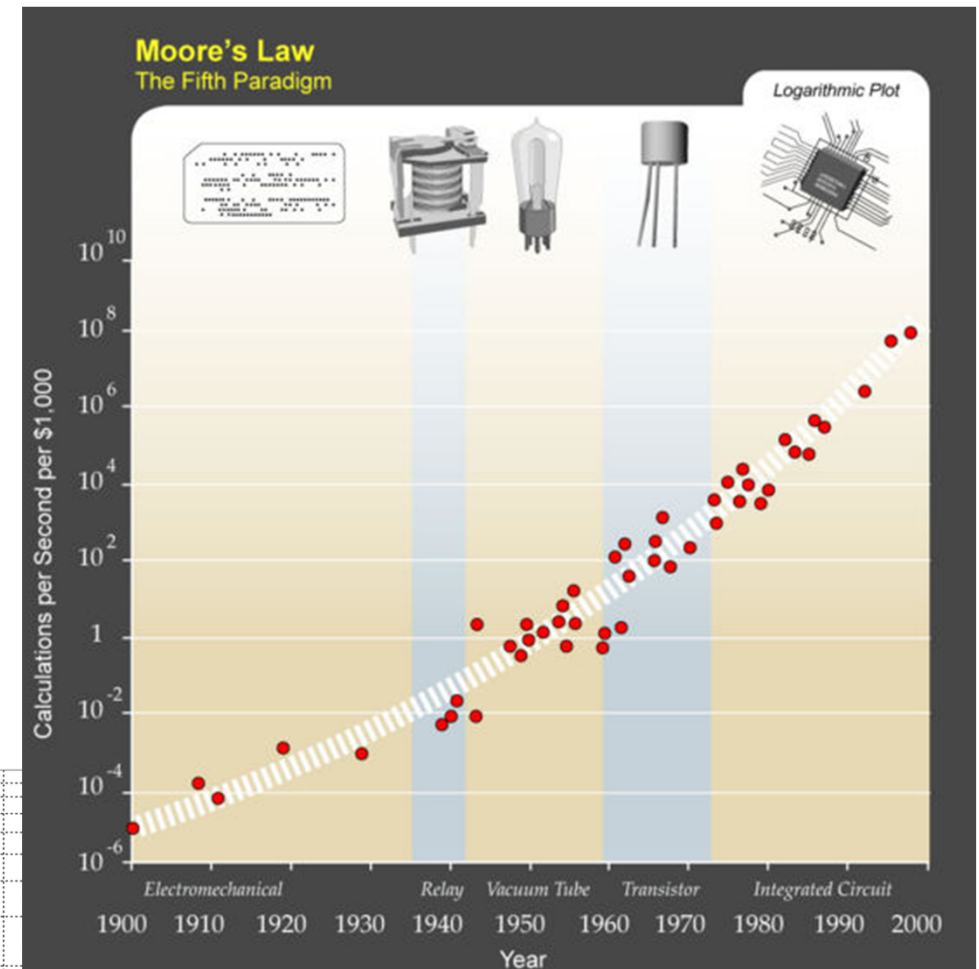
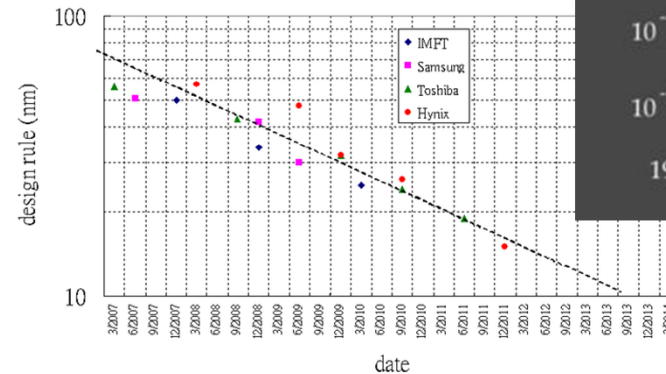
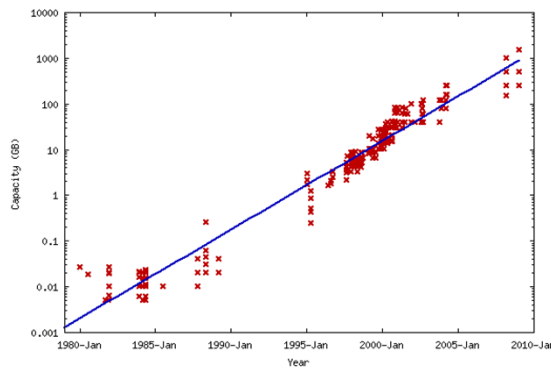
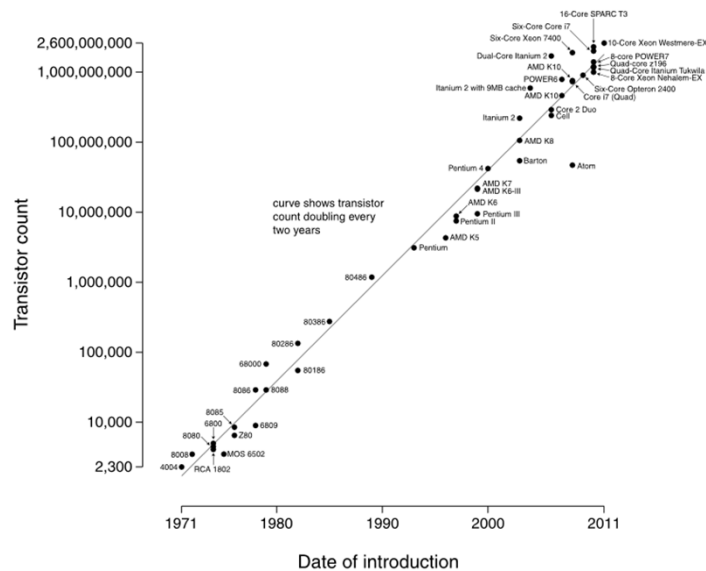
IBM MP350 (1996)
300GB/kasett



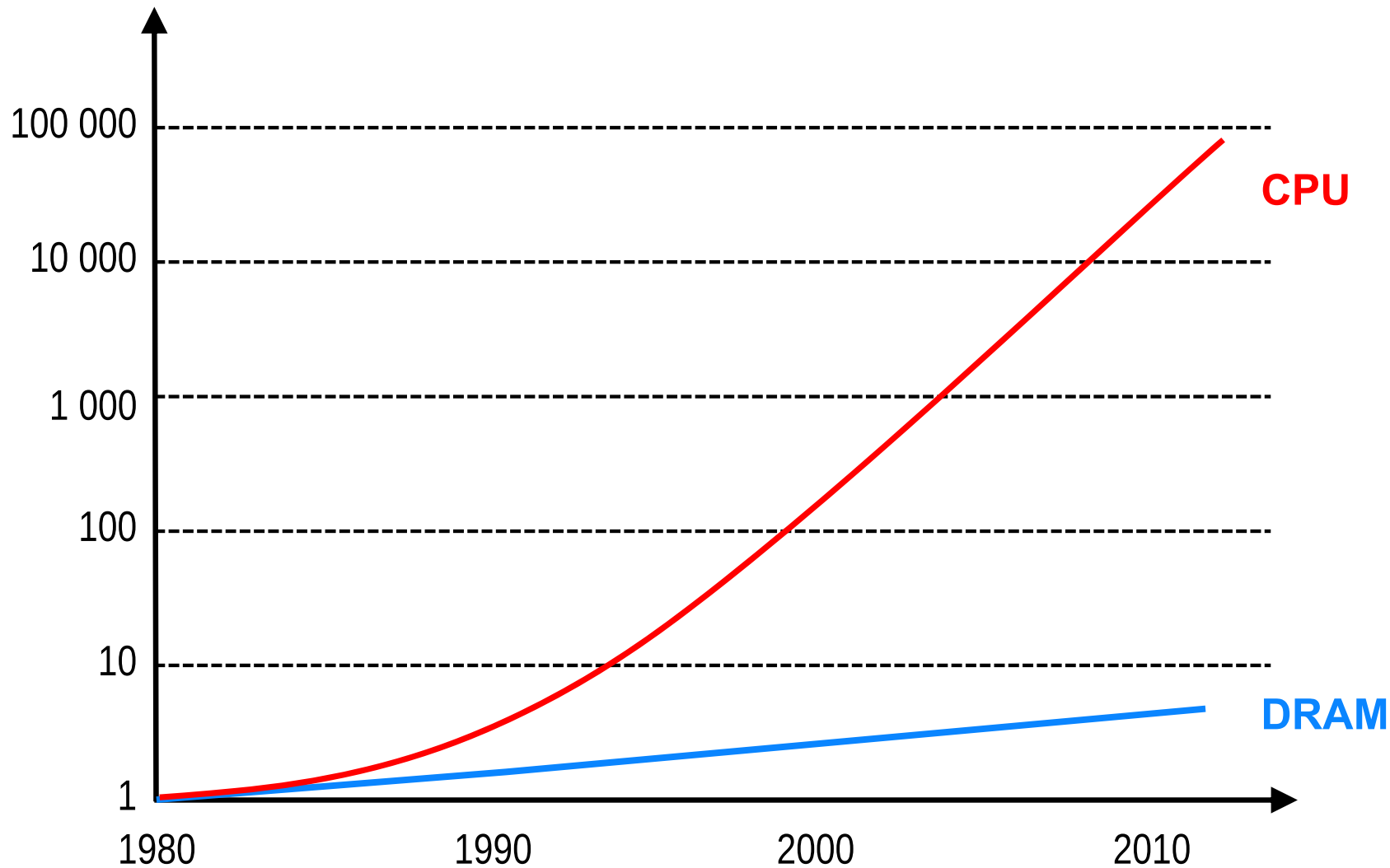
Oracle (2013)
5TB/kasett

Teknologiutveckling – Moores lag med konsekvenser

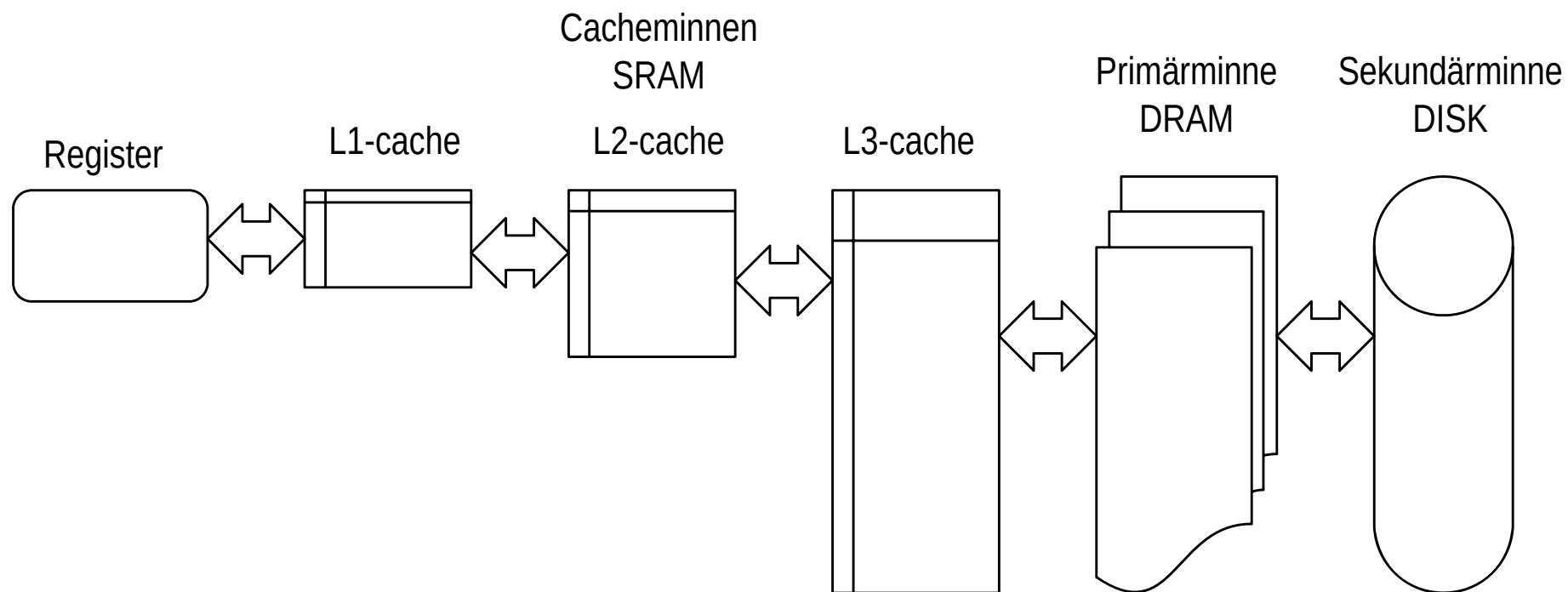
Microprocessor Transistor Counts 1971-2011 & Moore's Law



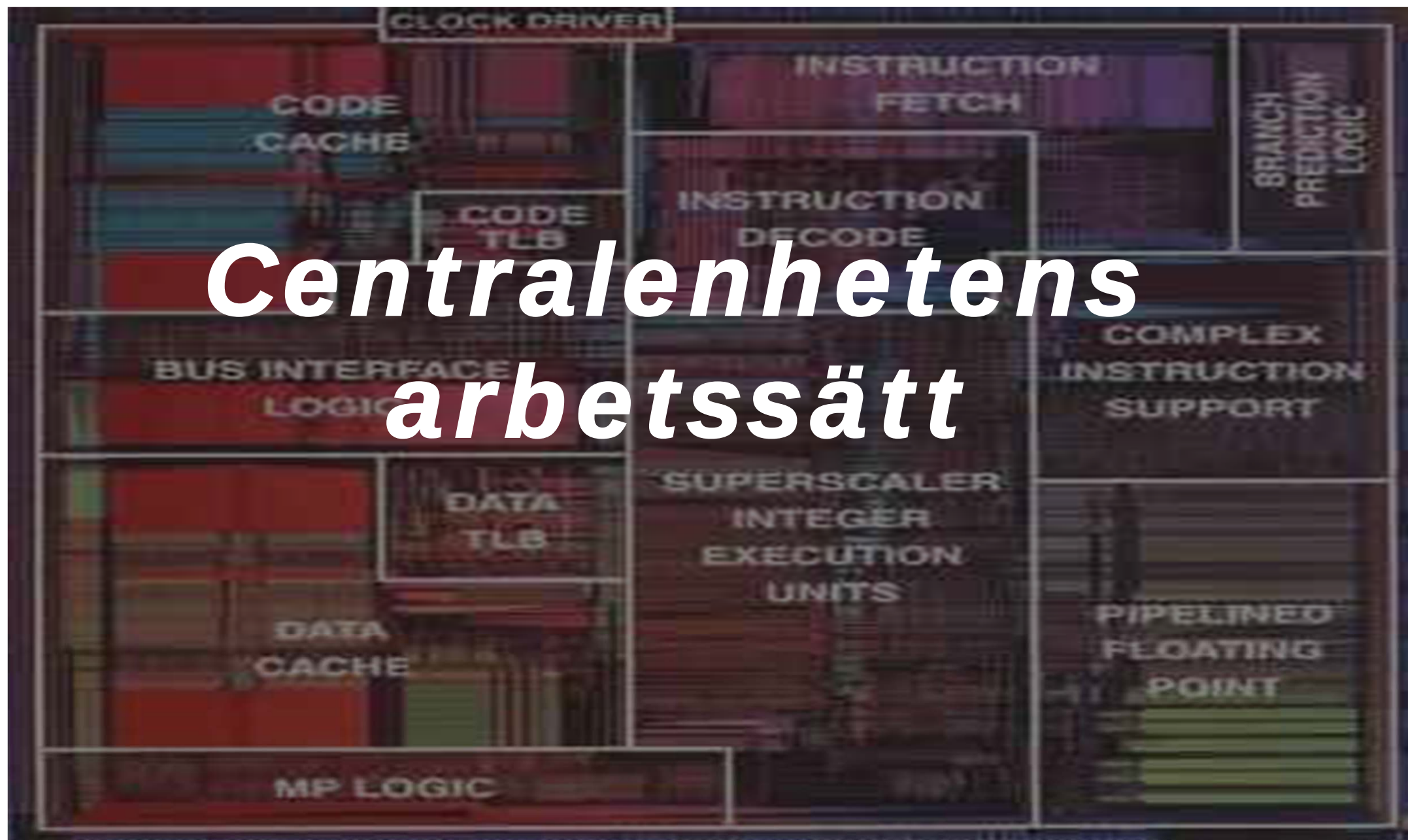
Skillnader i prestanda CPU/DRAM



Åtkomsttider i minneshierarkin (2013)



Storlek	1 KB	64 KB	256 KB	2-4 MB	4-16GB	4-16 TB
Åtkomsttid	300 ps	1 ns	3-10 ns	10-20 ns	50-100 ns	5-10 ms



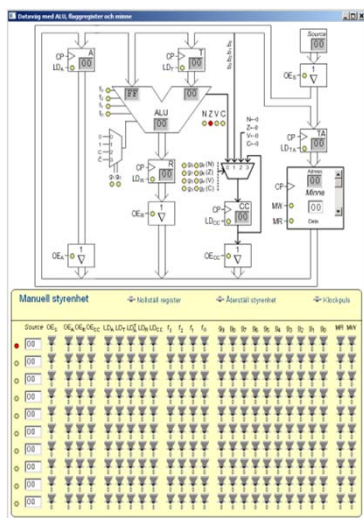
Centralenhetens arbetssätt

Det lagrade programmets princip

John Louis Von Neumann (1903-1957)

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

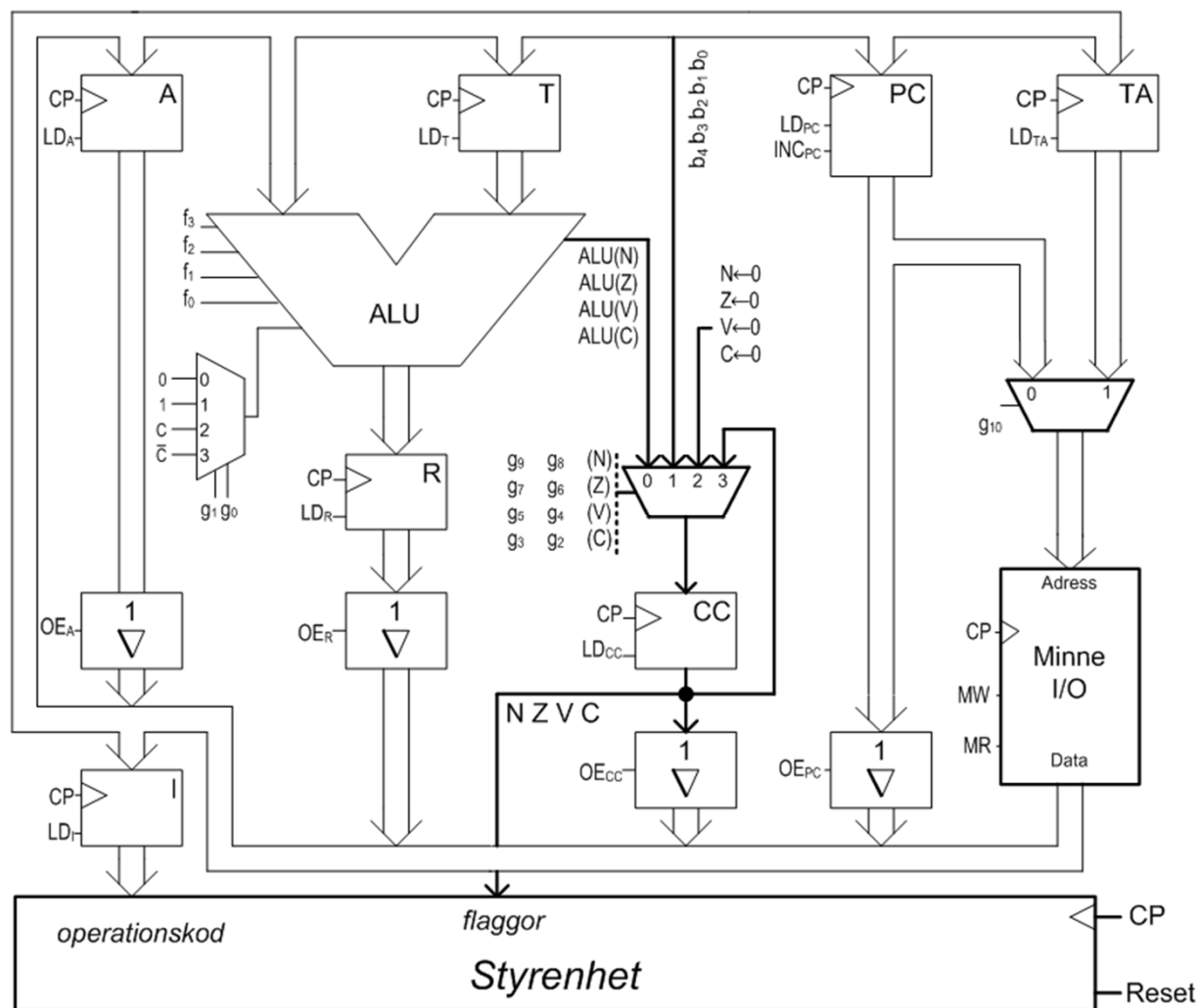
Sådana datorer kallas också ”von Neumann-dator”



Omkopplare som tidigare ställts in för att styra registeröverföring, ALU-operation etc. ersätts av en ”instruktion” som placerats i minnet.

(Adress) ₁₆	Innehåll	
00	00000000	instruktion 1
01	11110000	
02	10101100	instruktion 2
03	00110100	
04	11001110	instruktion 3
05	01011100	
FE	XXXXXXXX	data
FF	00000000	

Centralenhet och minne

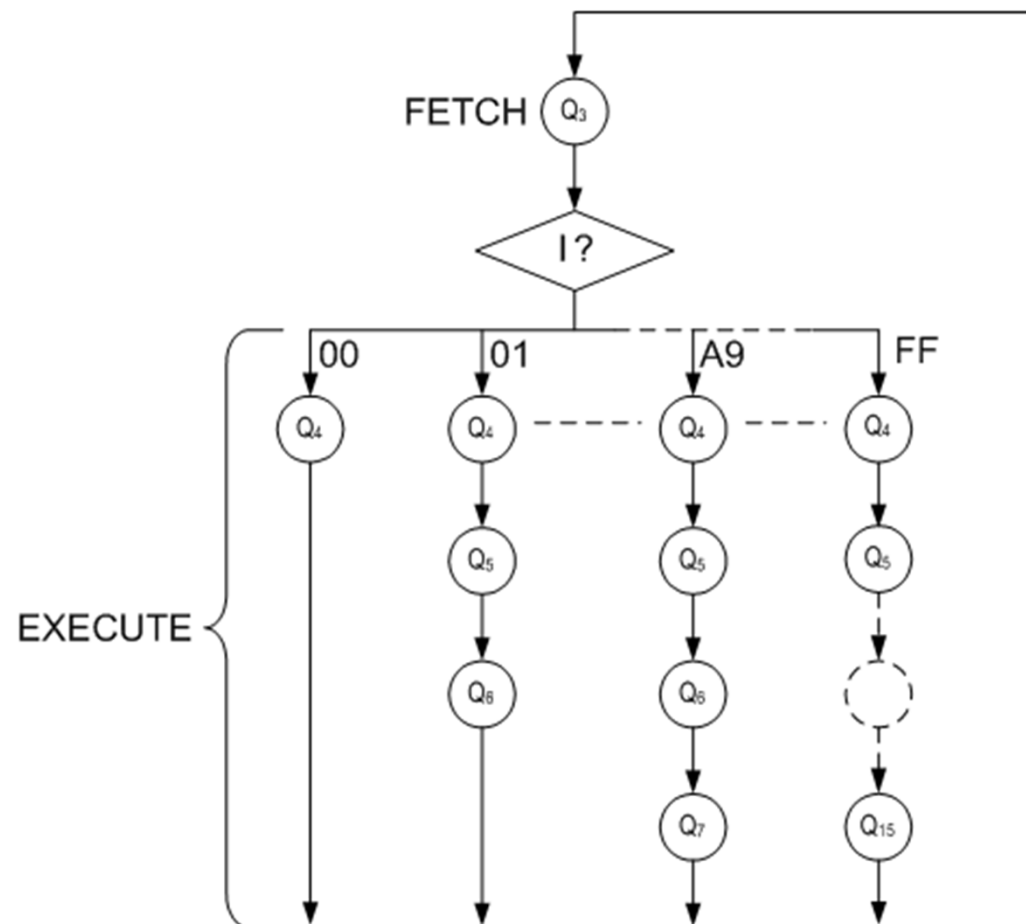


(Adress) ₁₆	Innehåll	
00	00000000	instruktion 1
01	11110000	
02	10101100	instruktion 2
03	00110100	instruktion 3
04	11001110	
05	01011100	
...		
FE	XXXXXXXX	data
FF	00000000	

Instruktionens faser

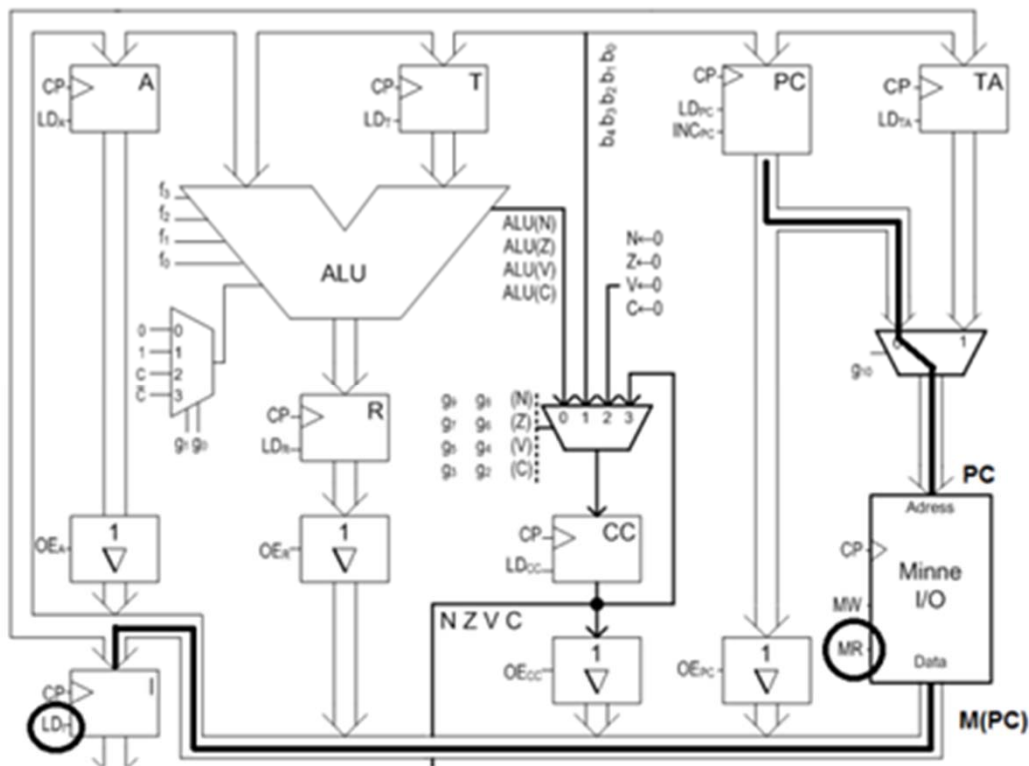
I sin enklaste form kan en instruktion delas in i två faser:

- I den första fasen, som kallas *hämtfasen*, läses operationskoden för instruktionen i minnet och lagras i instruktionsregistret I,
- I den andra fasen, som kallas *utförandefasen*, avkodar styrenheten innehållet i instruktionsregistret I, dvs operationskoden, och utför den styrsignalsekvens som operationskoden anger.



Hämta instruktion (FETCH)

RTN- beskrivning	Styrsignaler	Kommentarer
M(PC)→I;	MR=1; LD _I =1; INC _{PC} =1;	Adressen för nästa instruktions operationskod dvs. PC kopplas till minnets adressbuss. Läs operationskoden från minnet och placera i instruktionsregistret I. Adressen som finns i PC ökas med ett.

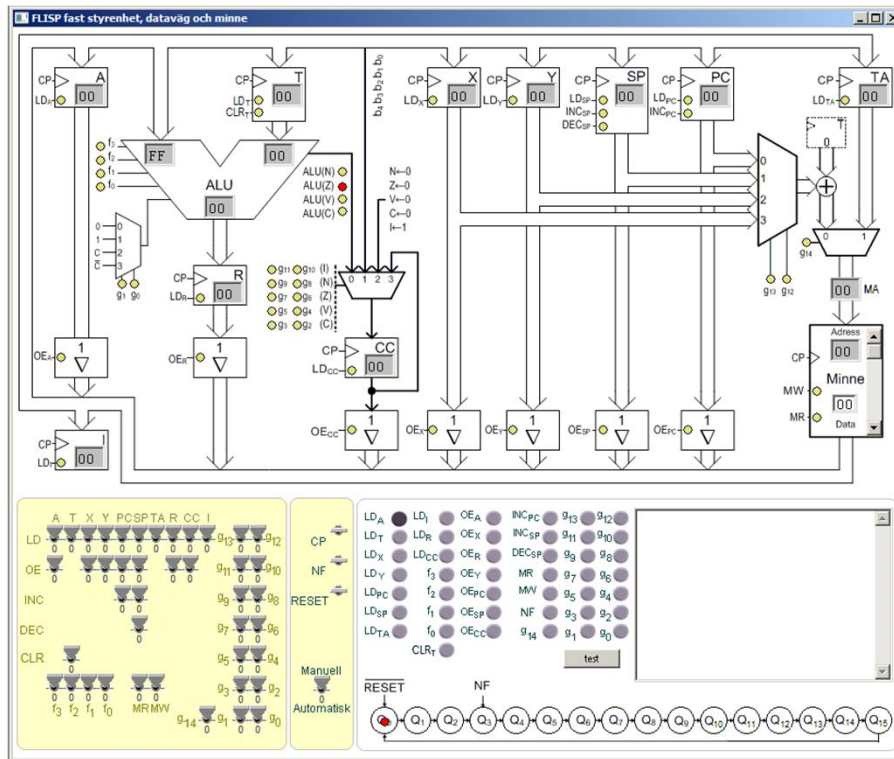


- FETCH-fasens uppgift är att överföra nästa instruktions operationskod till instruktionsregistret
- Detta förutsätter då att PC verkligen innehåller adressen till en instruktion (snarare än data...)
- Efter instruktionshämtning är det nödvändigt att öka PC med 1, och förbereda för en exekveringsfas eller en ny instruktionshämtning

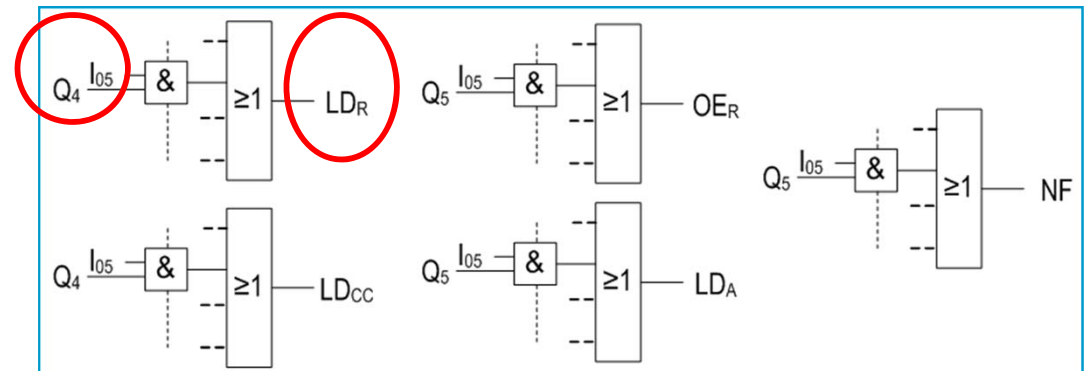
Automatisk styrenhet

Den automatiska styrenheten genererar sekvenser med styrsignalinformation.
En specifik instruktion implementeras med en unik styrsignalsekvens.

EXEMPEL; **CLRA** (Clear register A)



Tillstånd	Summa- term	RTN- beskrivning	Styrsignaler (=1)	Kommentarer
Q4	$(Q_4 \cdot I_{05})$	$R \rightarrow R;$	$LD_R;$	Alla ALU'ns f-signaler är 0 vilket ger ALU-funktion "nollställning", värdet 0 placeras därför i R Samtliga flaggor från ALU'n överförs till CC
Q5	$(Q_5 \cdot I_{05})$	$R \rightarrow A$	$OE_R; LD_A;$ NF	Resultat (0) överförs till register A Instruktion klar, hämta nästa...



I RTN-beskrivningen anger vi, klockpuls för klockpuls, hur datavägen används.

Tillstånd	Summa-term	RTN-beskrivning	Styrsignaler (=1)	Kommentarer
Q4	(Q4●I05)	0→R; ALU(N,V,Z,C) 0→CC;	LDR; LDCC;	Alla ALU'ns f-signaler är 0 vilket ger ALU-funktion "nollställning", värdet 0 placeras därför i R Samtliga flaggor från ALU'n överförs till CC
Q5	(Q5●I05)	R→A	OER; LDA; NF	Resultat (0) överförs till register A Instruktion klar, hämta nästa...

Program och minne

Tillhörande assemblerprogram

Maskinprogram i minnet

(Adress) ₁₆	Innehåll	
00	00000000	instruktion 1
01	11110000	
02	10101100	instruktion 2
03	00110100	
04	11001110	instruktion 3
05	01011100	
FE	XXXXXXXX	data
FF	00000000	

Symbolfält	Mnemoniceller direktiv	Operand	Kommentarsfält
	ORG	\$20	; Program börjar
start:	LDA	\$FB	; Läs data
	STA	\$FC	; Skriv data
	JMP	start	; Repetera...
Då symbolfältet används måste symbolnamnet börja i radens första position. Kolon (:) kan, men behöver inte anges. I detta fält ska instruktioner eller assemblerdirektiv anges Om instruktionen/direktivet har någon operand, ska denna anges här. Det sista fältet kan användas för kommentarstext. Det är lämpligt att markera detta med ett inledande semikolon, men det är inte nödvändigt.			

Hemuppgifter...

Dessa dokument behövs: (se kursens hemsida)

CHALMERS
Institutionen för data- och informationsteknik



Hemuppgifter

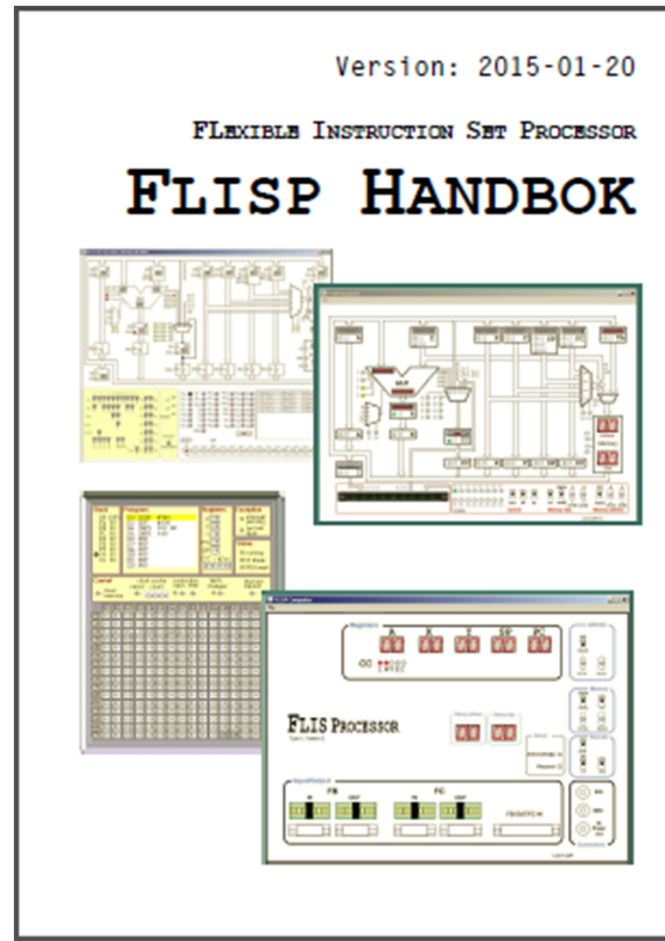
Introduktion till digital- och datorteknik för I

1. Detta häfte innehåller tre inlämningsuppgifter. Börja med att skriva ut en papperskopia av detta dokument. Följ därefter följande anvisningar som gäller för pappersdokumentet. Detta häfte (i pappersform) ska lämnas in för bedömning.
2. Skriv namn och personnummer nedan.
3. Lös uppgifterna och fyll i dina lösningar i detta häfte, var noggrann, elialliga eller kladdiga lösningar kommer inte att bedömas utan underkännas omedelbart. Lämnas in din ifyllda häfte för granskning.
4. Signaturerna nedan är din kvittens på att du är godkänd på respektive uppgift. För godkännande på detta kursmoment krävs att alla tre lösningar är godkända.

Godkännande av inlämningsuppgifter		
Uppgift	Datum	Granskat av (underskrift)
1		
2		
3		

Namn: _____
 Personnummer: _____

Hemuppgift.pdf



FlisProcessorHandbok - A4.pdf

I-Datorintroduktion.pdf



Samt kurslitteratur...

Inlämningsuppgift 1

Uppgift 1

Visa hur räkneoperationen $A=B-C$ utförs med en additionsenhet i ett digitalt system (binära talsystemet) där all aritmetik utförs på 2-komplements form och med 8 bitars noggrannhet.

$$B = (xxx)_{10}$$

$$C = (yyy)_{10}$$

$$\text{Ledning: } B - C = B + (-C)$$

Minnes-siffror									
	$(B)_2 =$								
+	-								
	$(C)_2 =$								
=	$(A)_2 =$								

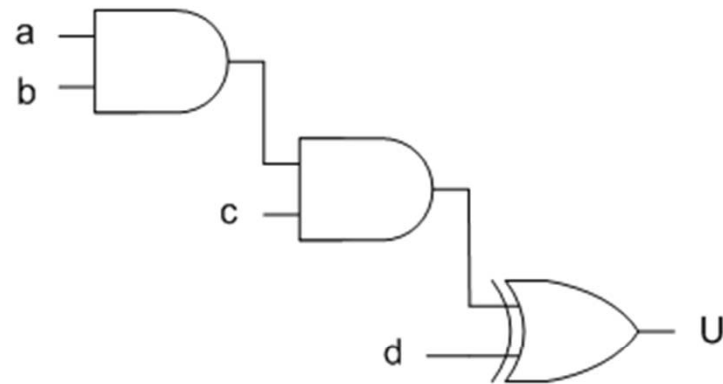
!

Datoraritmetik, se speciellt bilder 40-44

Bedrägningsuppgift 2

Uppgift 2

Betrakta följande kombinatoriska nät ("switchnät"),



Förutsätt insignaler (a,b,c,d) enligt följande tabell och ange den resulterande utsignalen (U) i respektive fall.

a	b	c	d	U
0	0	0	0	
0	0	0	1	
0	1	0	1	
1	1	1	1	

***Logiknivåer och grindar,
se speciellt bilder 51-58***

Lite jobbigare uppgift, se speciellt ”Flis processor Handbok”

Inlämningsuppgift 3

Uppgift 3

I en dators minne avläser vi följande innehåll, angivet på HEXADECIMAL form. Använd instruktionslistan för FLISP och bestäm den instruktionssekvens som representeras under förutsättning att innehållet på adress 20₁₆, är en instruktion för FLIS-processor'n.

a) Presentera din lösning i form av MNEMONICS för FLISP. (Kompletera följande tabell)

Adress	Minnesinnehåll	MNEMONIC / OPERAND
20	xx(opkod)	
21	xx	
22	xx(opkod)	
23	xx(opkod)	
24	xx(opkod)	
25	xx	

b) Beskriv med egna ord vad instruktionssekvensen utför.