Lösningsförslag

1 Talsystem och binära koder

1.1

a) $(8)_{10} = (1000)_2$
b) $(8)_{10} = (8)_{16}$
c) $(8)_{10} = (1000)_{NBCD}$
d) $(8)_{10} = (1011)_{EXCESS-3}$
e) $(8)_{10} = (1100)_{GRAY(4)}$

1.2

a) $(93)_{10} = (0101\ 1101)_2$
b) $(93)_{10} = (1001\ 0011)_{NBCD}$
c) $(93)_{10} = (5D)_{16}$
d) $(93)_{10} = (0110\ 0000)_{EXCESS-3}$

1.3

d)

1.4

a) ’?’
b) ’1’
c) ’+’

1.5

a) 43,48,41,4C,4D,52,53
b) 74,65,6B,6E,69,73,6B,61
c) 48,4F,47,53,4B,4F,4C,41

1.6

a) $(0101\ 0010\ 1000\ .\ 0011\ 0001)_{NBCD}$
b) $(0100\ 0000\ 0111\ .\ 1001\ 0110)_{NBCD}$

1.7

a) $(1100)_2 = (0001100)_2$
b) $(32)_{10} = (1100000)_2$
c) $(16)_{16} = (1010110)_2$

1.8

a) $(1100)_2 = (1001100)_2$
b) $(32)_{10} = (0100000)_2$
c) $(16)_{16} = (0010110)_2$

1.9

a) $(101011)_2 = (43)_{10}$
b) $(1100111)_2 = (103)_{10}$
c) $(111100)_2 = (60)_{10}$
d) $(101.011)_2 = (5{,}375)_{10}$
e) $(.111)_2 = (0{,}875)_{10}$
f) $(1000.01)_2 = (8{,}25)_{10}$

1.10

a) $(AB)_{16} = (171)_{10}$
b) $(11)_{16} = (17)_{10}$
c) $(80)_{16} = (128)_{10}$
d) $(C.4)_{16} = (12,25)_{10}$
e) $(0.68)_{16} = (0,40625)_{10}$
f) $(B3.D4)_{16} = (179,828125)_{10}$

1.11

a) $(45)_{10} = (101101)_2$
b) $(33)_{10} = (100001)_2$
c) $(87,65)_{10} = (1010111.101001)_2$
d) $(122,18)_{10} = (1111010.001011)_2$
e) $(45)_{16} = (1001010)_2$
f) $(33)_{16} = (110011)_2$
g) $(4.8)_{16} = (100.1)_2$
h) $(10.6)_{16} = (10000.011)_2$

1.12

a) $(255)_{10} = (FF)_{16}$
b) $(330)_{10} = (4A)_{16}$
c) $(19,37)_{10} = (13.5F)_{16}$
d) $(132,43)_{10} = (84.6E)_{16}$
e) $(1111110)_2 = (7E)_{16}$
f) $(100100001010)_2 = (90A)_{16}$
g) $(1111.1111)_2 = (F.F)_{16}$
h) $(1010.0011)_2 = (A3)_{16}$

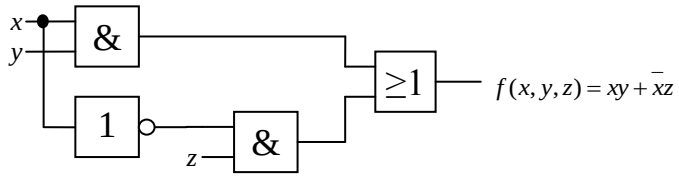
1.13

a) $512=2^9 < 360 < 256=2^8$ ger minst 9 bitar
b) $256=2^8 < 180 < 128=2^7$ ger minst 8 bitar

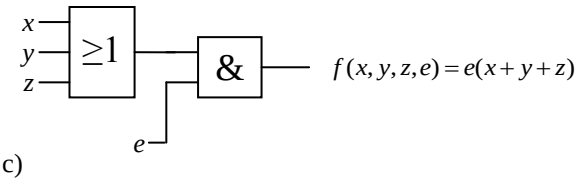
2 Switchnätalgebra

2.1

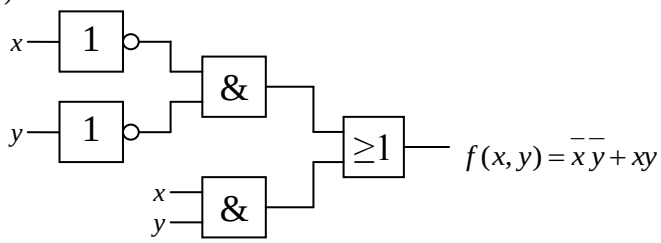
a)



b)



c)



2.2

a) $f(x,y,z)=xy+\bar{y}z$

b) $f(x,y,z)=x+y+\bar{z}$

c) $f(x,y,z)=\overline{\overline{xy}+z}$

d) $f(x,y,z)=\overline{(\overline{x+y})z}$

2.3

a) och c).

2.4

2.5

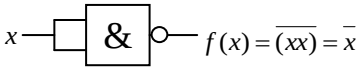
a),c) och d)

2.6

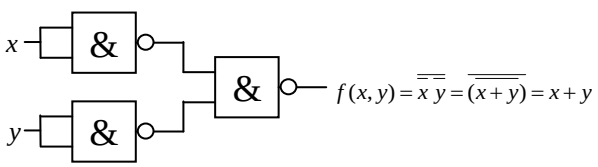
$A=\bar{x}, B=\bar{y}$

2.7

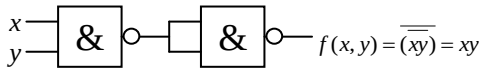
a)



b)

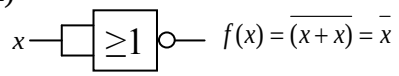


c)

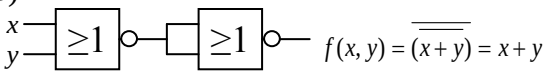


2.8

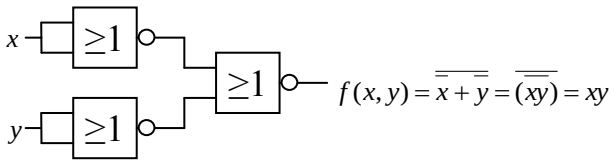
a)



b)

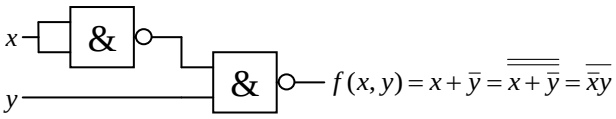


c)

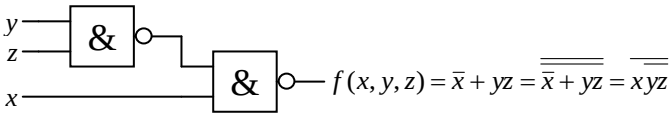


2.9

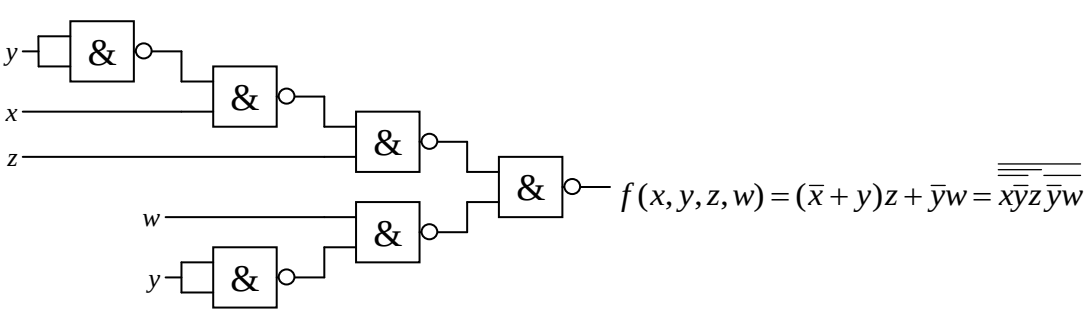
a)



b)

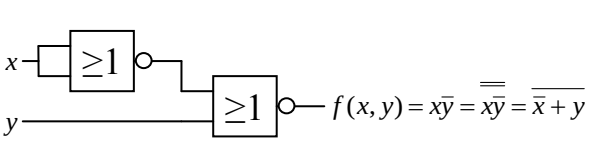


c)

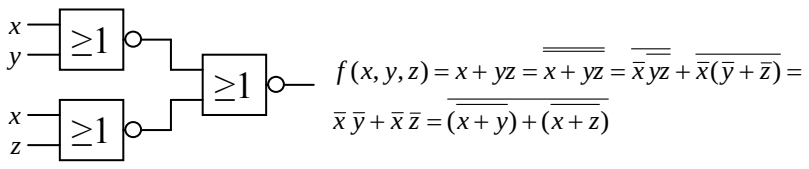


2.10

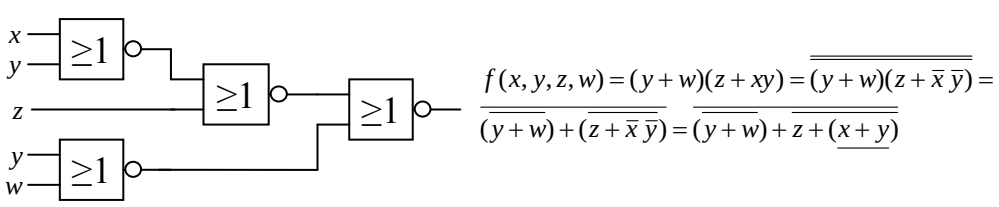
a)



b)



c)



2.11

a)

x	y	z	xy	$\bar{x}z$	$f(x,y,z)=xy+\bar{x}z$	m
0	0	0				
0	0	1		1	1	1
0	1	0				
0	1	1		1	1	3
1	0	0				
1	0	1				
1	1	0	1		1	6
1	1	1	1		1	7

Mintermer: $\sum m(1,3,6,7)= \bar{x}\bar{y}z+\bar{x}yz+xy\bar{z}+xyz$

b)

x	y	z	$\bar{y}z$	$x+\bar{y}z$	$y+z$	$f(x,y,z)=(x+\bar{y}z)(y+z)$	m
0	0	0					
0	0	1	1	1	1	1	1
0	1	0			1		
0	1	1			1		
1	0	0		1			
1	0	1	1	1	1	1	5
1	1	0		1	1	1	6
1	1	1		1	1	1	7

Mintermer: $\sum m(1,5,6,7)= \bar{x}\bar{y}z+x\bar{y}z+xy\bar{z}+xyz$

c)

x	y	z	w	xw	yz	$f(x,y,z,w)=xw+yz$	m
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		1	1	6
0	1	1	1		1	1	7
1	0	0	0				
1	0	0	1	1		1	9
1	0	1	0				
1	0	1	1	1		1	11
1	1	0	0				
1	1	0	1	1		1	13
1	1	1	0		1	1	14
1	1	1	1	1	1	1	15

Mintermer: $\sum m(6,7,9,11,13,14,15)= \bar{x}yz\bar{w}+\bar{x}yzw+x\bar{y}\bar{z}w+x\bar{y}zw+xy\bar{z}w+xyz\bar{w}+xyzw$

d)

x	y	z	w	xy	$\bar{z}\bar{w}$	$\bar{x}\bar{w}$	$\bar{x}\bar{z}$	$xy+\bar{z}\bar{w}$	$\bar{x}\bar{w}+\bar{x}\bar{z}$	$f(x,y,z,w)=(xy+\bar{z}\bar{w})(\bar{x}\bar{w}+\bar{x}\bar{z})$	m
0	0	0	0		1	1	1	1	1	1	0
0	0	0	1				1		1		
0	0	1	0			1			1		
0	0	1	1								
0	1	0	0		1	1	1	1	1	1	4
0	1	0	1				1		1		
0	1	1	0			1			1		
0	1	1	1								
1	0	0	0		1			1			
1	0	0	1								
1	0	1	0								
1	0	1	1								
1	1	0	0	1	1			1			
1	1	0	1	1				1			
1	1	1	0	1				1			
1	1	1	1	1				1			

Mintermer: $\sum m(0,4)= \bar{x}\bar{y}\bar{z}\bar{w}+\bar{x}y\bar{z}\bar{w}$

2.12

d)

2.13

a)

2.14

a)

x	y	z	$x\bar{y}$	$\bar{x}z$	$y\bar{z}$	$f(x,y,z)=x\bar{y}+\bar{x}z+y\bar{z}$	M
0	0	0				0	0
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				0	7

Maxtermer: $\prod M(0,7) = (x+y+z)(\bar{x}+\bar{y}+\bar{z})$

b)

x	y	z	xy	xz	$f(x,y,z)=(xy+xz)$	M
0	0	0			0	0
0	0	1			0	1
0	1	0			0	2
0	1	1			0	3
1	0	0			0	4
1	0	1		1		
1	1	0	1			
1	1	1	1	1		

Maxtermer: $\prod M(0,1,2,3,4) = (x+y+z)(x+y+\bar{z})(x+\bar{y}+z)(x+\bar{y}+\bar{z})(\bar{x}+y+z)$

c)

x	y	z	w	$\overline{x}\,\overline{y}$	$\overline{z}\,\overline{w}$	yz	$f(x,y,z,w)=\overline{x}\,\overline{y}+\overline{z}\,\overline{w}+yz$	M
0	0	0	0	1				
0	0	0	1	1				
0	0	1	0	1				
0	0	1	1	1				
0	1	0	0		1			
0	1	0	1				0	5
0	1	1	0			1		
0	1	1	1			1		
1	0	0	0		1			
1	0	0	1				0	9
1	0	1	0				0	10
1	0	1	1				0	11
1	1	0	0		1			
1	1	0	1				0	12
1	1	1	0			1		
1	1	1	1			1		

Maxtermer: $\prod$ M (5,9,10,11,12) = $(x+\overline{y}+z+\overline{w})(\overline{x}+y+z+\overline{w})(\overline{x}+y+\overline{z}+w)(\overline{x}+y+\overline{z}+\overline{w})(\overline{x}+\overline{y}+z+\overline{w})$

d)

x	y	z	w	$\overline{x}\,\overline{y}z$	w	$\overline{z}\,\overline{w}$	$f(x,y,z,w)=\overline{x}\,\overline{y}z+w+\overline{z}\,\overline{w}$	M
0	0	0	0			1		
0	0	0	1		1			
0	0	1	0	1				
0	0	1	1	1	1			
0	1	0	0			1		
0	1	0	1		1			
0	1	1	0				0	6
0	1	1	1		1			
1	0	0	0			1		
1	0	0	1		1			
1	0	1	0				0	10
1	0	1	1		1			
1	1	0	0			1		
1	1	0	1		1			
1	1	1	0				0	14
1	1	1	1		1			

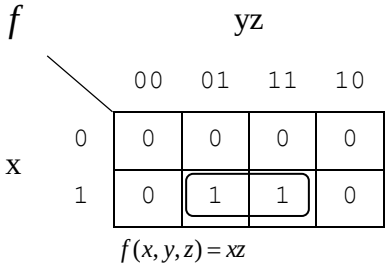
Maxtermer: $\prod$ M (6,10,14) = $(x+\overline{y}+\overline{z}+w)(\overline{x}+y+\overline{z}+w)(\overline{x}+\overline{y}+\overline{z}+w)$

2.15

e)

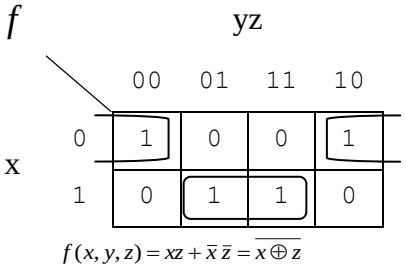
2.16

a)



$f(x,y,z)=xz$

b)



$f(x,y,z)=xz+\overline{x}\,\overline{z}=\overline{x\oplus z}$

c) f

xy \ zw	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	0	1	0
10	0	0	0	0

$f(x,y,z,w) = \bar{x}yw + yzw$

d) f

xy \ zw	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

$f(x,y,z,w) = \bar{y}\,\bar{w} + yw = \overline{y \oplus w}$

e) f

xy \ zw	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	–	1	1	–

$f(x,y,z,w) = x$

f) f

xy \ zw	00	01	11	10
00	0	0	0	0
01	–	1	1	1
11	0	0	1	0
10	0	0	0	0

$f(x,y,z,w) = \bar{x}y + yzw$

g) f

xy \ zw	00	01	11	10
00	1	1	0	0
01	0	1	1	0
11	1	0	1	1
10	0	0	0	0

$f(x,y,z,w) = \bar{x}\,\bar{y}\,\bar{z} + \bar{x}yw + xyz + xy\bar{w}$

h) f

xy \ zw	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	–	0	–	0
10	1	0	1	0

$f(x,y,z,w) =$
$\bar{x}\,\bar{y}\,\bar{z}\,\bar{w} + \bar{x}y\bar{z}w + \bar{x}\,\bar{y}zw + \bar{x}yz\bar{w} + x\bar{z}\,\bar{w} + xzw =$
$x(\overline{z \oplus w}) + \bar{x}(\overline{z \oplus y \oplus w})$

i) f

xy \ zw	00	01	11	10
00	1	0	0	1
01	–	0	1	0
11	–	0	0	1
10	1	0	1	0

$f(x,y,z,w) =$
$\bar{z}\,\bar{w} + \bar{x}\,\bar{y}z\bar{w} + \bar{x}yzw + xyz\bar{w} + x\bar{y}zw + =$
$\bar{z}\,\bar{w} + z(x \oplus \overline{y \oplus w})$

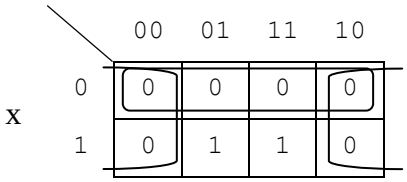
j) f

xy \ zw	00	01	11	10
00	0	0	0	1
01	0	0	1	0
11	0	1	0	0
10	1	0	0	0

$f(x,y,z,w) = \bar{x}\,\bar{y}z\bar{w} + \bar{x}yzw + xy\bar{z}w + x\bar{y}\,\bar{z}\,\bar{w} =$
$\bar{x}z(\overline{y \oplus w}) + x\bar{z}(y \oplus w)$

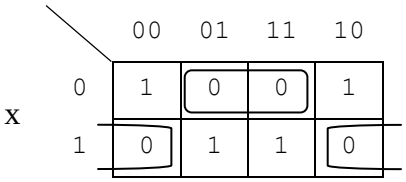
2.17

a) f



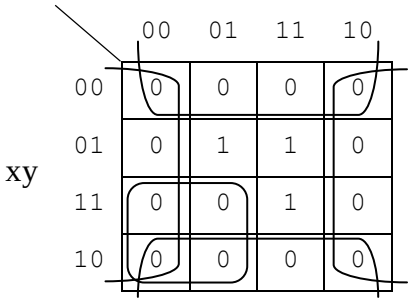
$f(x,y,z) = xz$

b) f



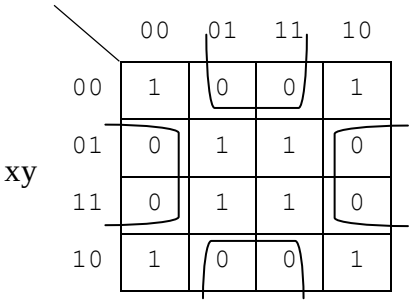
$f(x,y,z) = (x+\bar{z})(\bar{x}+z)$

c) f



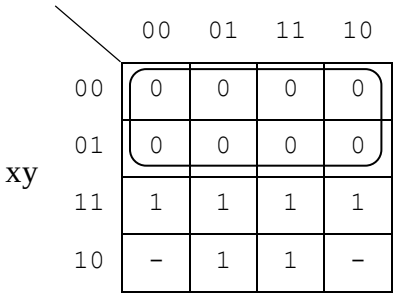
$f(x,y,z,w) = yw(\bar{x}+z)$

d) f



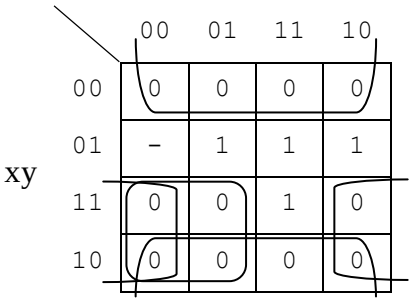
$f(x,y,z,w) = (\bar{y}+w)(\bar{x}+z)$

e) f



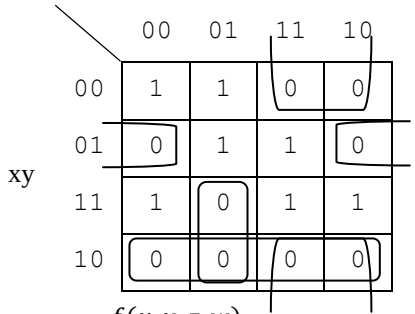
$f(x,y,z,w) = x$

f) f



$f(x,y,z,w) = y(\bar{x}+w)(\bar{x}+z)$

g) f

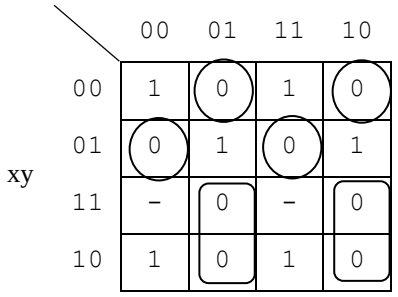


$f(x,y,z,w) =$

$(y+\bar{z})(x+\bar{y}+w)$

$(\bar{x}+z+\bar{w})(\bar{x}+y)$

h) f



$f(x,y,z,w) = (x+\bar{y}+z+w)$

$(x+y+z+\bar{w})(x+\bar{y}+\bar{z}+w)$

$(x+y+\bar{z}+w)(\bar{x}+z+\bar{w})$

$(\bar{x}+\bar{z}+w)$

3 Aritmetik

3.1

a) %01010101 = 85
b) %00111100 = 60
c) %10101010 = 170
d) %11000011 = 195

3.2

a) %01010101 = 85
b) %00111100 = 60
c) %10101010 = -86
d) %11000011 = -61

3.3

a) -10 = % -(0000 1010) = 1111 0110
b) -38 = % -(0010 0110) = 1101 1010
c) -42 = % -(0010 1010) = 1101 0110
d) -56 = % -(0011 1000) = 1100 1000

3.4 a)

Operation		Decimalt								
				0				1		
	X		36		0	0	1	0	0	1	0	0
+	Y	+	68	+	0	1	0	0	0	1	0	0
=	R		104		0	1	1	0	1	0	0	0

$R \neq 0 \rightarrow Z=0$
$c_8=0 \rightarrow C=0$

Operationen innebär X=36, Y= 68, R= 104, (36+68=104) Eftersom talen betraktas på *binärform* är talområdet (8 bitar) 0-255 och således resultatet korrekt. Detta indikeras också av att C-flaggan är 0.

b)

Operation		Decimalt								
				1	1	1	1	1	1		
	X		188		1	0	1	1	1	1	0	0
+	Y	+	68	+	0	1	0	0	0	1	0	0
=	R		0		0	0	0	0	0	0	0	0

$R=0 \rightarrow Z=1$
$c_8=1 \rightarrow C=1$

Operationen innebär X=188, Y= 68, R= 0, men 188+68=256 ! Eftersom talen betraktas på *binärform* är talområdet (8 bitar) 0-255 och således resultatet utanför talområdet. Carryflaggan sätts därför till 1, dessutom blir i detta fall innehållet i registret 0 varför Z-flaggan också sätts till 1.

c)

Operation		Decimalt								
				1						1	
	X		129		1	0	0	0	0	0	0	1
+	Y	+	129	+	1	0	0	0	0	0	0	1
=	R		2		0	0	0	0	0	0	1	0

$R \neq 0 \rightarrow Z=0$
$c_8=1 \rightarrow C=1$

Operationen innebär X=129, Y=129, R=2, men 129+129=258! Eftersom talen betraktas på *binärform* är talområdet (8 bitar) 0-255 och således resultatet utanför talområdet. Carryflaggan sätts därför till 1, resultatet i registret är dock skilt från 0 varför Z-flaggan sätts till 0.

d)

Operation		Decimalt									
					0						
	X		74		0	1	0	0	1	0	1	0
+	Y	+	53	+	0	0	1	1	0	1	0	1
=	R		127		0	1	1	1	1	1	1	1

$R \neq 0 \rightarrow Z=0$
$c_8=0 \rightarrow C=0$

Operationen innebär X=74, Y=53, R=127. Eftersom talen betraktas på *binärform* är talområdet (8 bitar) 0-255 och således resultatet korrekt. Detta indikeras också av att C-flaggan är 0.

e)

Operation		Decimalt									
					0	1			1		1	
	X		74		0	1	0	0	1	0	1	0
+	Y	+	74	+	0	1	0	0	1	0	1	0
=	R		148		1	0	0	1	0	1	0	0

$R \neq 0 \rightarrow Z=0$
$c_8=0 \rightarrow C=0$

Operationen innebär X=74, Y=74, R=148. Eftersom talen betraktas på *binärform* är talområdet (8 bitar) 0-255 och således resultatet korrekt. Detta indikeras också av att C-flaggan är 0.

3.5 a)

Operation		Decimalt									
					0				1			
	X		36		0	0	1	0	0	1	0	0
+	Y	+	68	+	0	1	0	0	0	1	0	0
=	R		104		0	1	1	0	1	0	0	0

$R \neq 0 \rightarrow Z=0 \quad N=r_7=0$
$r_7'x_7y_7 + r_7x_7'y_7'=0 \rightarrow V=0$

Operationen innebär X=36, Y= 68, R= 104, (36+68=104). Eftersom talen betraktas på *tvåkomplementsform* är talområdet (8 bitar) -128- +127 och således resultatet korrekt. Inget *spill* alltså och av teckenöverläggningen ser vi hur V-flaggan sätts till 0. Resultatet är skilt från 0 och därmed sätts Z-flaggan till 0.

b)

Operation		Decimalt									
					1	1	1	1	1	1		
	X		-68		1	0	1	1	1	1	0	0
+	Y	+	68	+	0	1	0	0	0	1	0	0
=	R		0		0	0	0	0	0	0	0	0

$R=0 \rightarrow Z=1 \quad N=r_7=0$
$r_7'x_7y_7 + r_7x_7'y_7'=0 \rightarrow V=0$

Operationen innebär X=-68, Y= 68, R= 0,(-68+68=0). Eftersom talen betraktas på *tvåkomplementsform* är talområdet (8 bitar) -128- +127 och således resultatet inom talområdet. Inget *spill* alltså och av teckenöverläggningen ser vi hur V-flaggan sätts till 0, dessutom blir i detta fall innehållet i registret 0 varför Z-flaggan sätts till 1.

c)

Operation		Decimalt									
				1						1	
	X		-127		1	0	0	0	0	0	1
+	Y	+	-127	+	1	0	0	0	0	0	1
=	R		2		0	0	0	0	0	1	0

$R \neq 0 \rightarrow Z=0 \quad N=r_7=0$

$r_7'x_7y_7 + r_7x_7'y_7' = 1 \rightarrow V=1$

Operationen innebär X=-127, Y=-127, R= 2 men -127-127 = -254. Eftersom talen betraktas på *tvåkomplementsform* är talområdet (8 bitar) -128- +127 och således resultatet utanför talområdet. Av teckenöverläggningen (två negativa tal adderas men resultatet blir positivt) ser vi att V-flaggan sätts till 1 som indikator på spillet. Resultatet är skilt från 0 varför Z-flaggan sätts till 0.

d)

Operation		Decimalt								
				0						
	X		74		0	1	0	0	1	0	1	0
+	Y	+	53	+	0	0	1	1	0	1	0	1
=	R		127		0	1	1	1	1	1	1	1

$R \neq 0 \rightarrow Z=0 \quad N=r_7=0$

$r_7'x_7y_7 + r_7x_7'y_7' = 0 \rightarrow V=0$

Operationen innebär X=74, Y= 53, R= 127, (74+53=127). Eftersom talen betraktas på *tvåkomplementsform* är talområdet (8 bitar) -128- +127 och således resultatet korrekt. Inget *spill* alltså och av teckenöverläggningen ser vi hur V-flaggan sätts till 0. Resultatet är skilt från 0 och därmed sätts Z-flaggan till 0.

e)

Operation		Decimalt									
				0	1			1		1	
	X		74		0	1	0	0	1	0	1	0
+	Y	+	74	+	0	1	0	0	1	0	1	0
=	R		-108		1	0	0	1	0	1	0	0

$R \neq 0 \rightarrow Z=0 \quad N=r_7=1$

$r_7'x_7y_7 + r_7x_7'y_7' = 1 \rightarrow V=1$

Operationen innebär X=74, Y=74, R= -108, resultatet är alltså fel. Talen betraktas på *tvåkomplementsform* och talområdet (8 bitar) är -128- +127 och således resultatet inom talområdet. Av teckenöverläggningen (två positiva tal adderas men resultatet blir negativt) ser vi dock att V-flaggan sätts till 1 som indikator på spillet. Resultatet är skilt från 0 varför Z-flaggan sätts till 0.

3.6 a)

Operation		Decimalt								
				00						
	X		126		0	1	1	1	1	1	1	0
-	Y	-	80	-	0	1	0	1	0	0	0	0
=	R		46		0	0	1	0	1	1	1	0

$C=B=c_8=0 \quad Z=0$

Operationen innebär X=126, Y=80, R=46. Eftersom talen betraktas på *binärform* är talområdet (8 bitar) 0-255 och således resultatet korrekt. Detta indikeras också av att C-flaggan är 0, dvs. ingen lånesiffra genereras från position c_8.

b)

Operation		Decimalt	
	X	80	0 1 0 1 0 0 0 0
−	Y	− 126	− 0 1 1 1 1 1 1 0
=	R	210	1 1 0 1 0 0 1 0

$C=B=c_8=1$ $Z=0$

Operationen innebär X=80, Y=126, R=210. Resultatet är uppenbarligen fel eftersom negativa tal inte kan representeras av talområdet. (Eftersom talen betraktas på *binärform* är talområdet 0-255) . Detta indikeras också av att C-flaggan är 1, dvs. lånesiffra genereras från position c_8.

c)

Operation		Decimalt	
	X	222	1 1 0 1 1 1 1 0
−	Y	− 34	− 0 0 1 0 0 0 1 0
=	R	188	1 0 1 1 1 1 0 0

$C=B=c_8=0$ $Z=0$

Operationen innebär X=222, Y=34, R=188. Eftersom talen betraktas på *binärform* är talområdet (8 bitar) 0-255 och således resultatet korrekt. Detta indikeras också av att C-flaggan är 0, dvs. ingen lånesiffra genereras från position c_8.

d)

Operation		Decimalt	
	X	34	0 0 1 0 0 0 1 0
−	Y	− 222	− 1 1 0 1 1 1 1 0
=	R	68	1 0 1 1 1 1 0 0

$C=B=c_8=1$ $Z=0$

Operationen innebär X=34, Y=222, R=68. Resultatet är fel eftersom negativa tal inte kan representeras av talområdet. (Eftersom talen betraktas på *binärform* är talområdet 0-255) . Detta indikeras också av att C-flaggan är 1, dvs. lånesiffra genereras från position c_8.

3.7 a)

Operation		Decimalt	
	X	126	0 1 1 1 1 1 1 0
+	2~Y	+ 176	+ 1 0 1 1 0 0 0 0
=	R	46	0 0 1 0 1 1 1 0

$c_8=1 \rightarrow C=B=0$ $Z=0$

Det gäller att X=126, Y=80, korrekt resultat R=46. Operationen innebär X=126, 2~Y=176, R=46 eftersom talen betraktas på *binärform* med talområdet 0-255 och alltså är resultatet korrekt. Detta indikeras också av att C-flaggan är 0, dvs. minnessiffran i position c_8 är 1.

b)

Operation		Decimalt	
	X	80	0 1 0 1 0 0 0 0
+	2~Y	+ 130	+ 1 0 0 0 0 0 1 0
=	R	210	1 1 0 1 0 0 1 0

$c_8=0 \rightarrow C=B=1$ $Z=0$

Det gäller att X=80, Y=126, korrekt resultat R=-46. Operationen innebär X=80, 2~Y=130, R=210. Resultatet är uppenbarligen fel eftersom negativa tal inte kan representeras av talområdet. (Eftersom talen betraktas på *binärform* är talområdet 0-255) . Detta indikeras också av att C-flaggan är 1, dvs. minnessiffran i position c_8 är 0.

c)

Operation		Decimalt									
				1	1		1	1	1	1	
	X	222			1	1	0	1	1	1	1	0
+	2~Y	+ 222	+	1	1	0	1	1	1	1	0
=	R	188		1	0	1	1	1	1	0	0

c_8=1→ C=B=0 Z=0

Det gäller att X=222, Y=34, korrekt resultat R=188. Operationen innebär X=222, 2~Y=222, R=188. Eftersom talen betraktas på *binärform* är talområdet (8 bitar) 0-255 och således resultatet korrekt. Detta indikeras också av att C-flaggan är 0, dvs. minnessiffran i position c_8 är 1.

d)

Operation		Decimalt									
				0		1				1	
	X	34			0	0	1	0	0	0	1	0
+	2~Y	+ 34	+	0	0	1	0	0	0	1	0
=	R	68		0	1	0	0	0	1	0	0

C=B=c_8=1 Z=0

Det gäller att X=34, Y=222, korrekt resultat R=-188. Operationen innebär X=34, 2~Y=34, R=68. Resultatet är fel eftersom negativa tal inte kan representeras av talområdet. (Eftersom talen betraktas på *binärform* är talområdet 0-255) . Detta indikeras också av att C-flaggan är 1, dvs. minnessiffran i position c_8 är 0.

3.8 a)

Operation		Decimalt									
				1	1	1	1				
	X	126			0	1	1	1	1	1	1	0
+	2~Y	+ -80	+	1	0	1	1	0	0	0	0
=	R	46		0	0	1	0	1	1	1	0

R≠0 → Z=0 N=r_7=0

r_7'x_7y_7 + r_7x_7'y_7'=0 → V=0

Det gäller att X=126, Y=80, korrekt resultat R=46. Operationen innebär X=126, 2~Y=-80, R=46 . Eftersom talen betraktas på *tvåkomplementsform* är talområdet (8 bitar) -128- +127 och operationens resultat korrekt. Inget *spill* alltså och av teckenöverläggningen ser vi hur V-flaggan sätts till 0. Resultatet är skilt från 0 och därmed sätts Z-flaggan till 0.

b)

Operation		Decimalt									
				0							
	X	80			0	1	0	1	0	0	0	0
+	2~Y	+ -126	+	1	0	0	0	0	0	1	0
=	R	-46		1	1	0	1	0	0	1	0

R≠0 → Z=0 N=r_7=1

r_7'x_7y_7 + r_7x_7'y_7'=0 → V=0

Det gäller att X=80, Y=126, korrekt resultat R=-46. Operationen innebär X=80, 2~Y=-126, R=-46. Operationens resultat är korrekt. Inget *spill* alltså och av teckenöverläggningen ser vi hur V-flaggan sätts till 0. Resultatet är skilt från 0 och därmed sätts Z-flaggan till 0.

c)

Operation		Decimalt	
	X	-34	1 1 1 1 1 1 1 0
+	2~Y	+ -34	+ 1 1 0 1 1 1 1 0
=	R	-68	1 0 1 1 1 1 0 0

$R \neq 0 \rightarrow Z=0$ $N=r_7=1$

$r_7'x_7y_7 + r_7x_7'y_7'=0 \rightarrow V=0$

Det gäller att X=-34, Y=34, korrekt resultat R=-68. Operationen innebär X=-34, 2~Y=-34, R=-68. Operationens resultat är korrekt. Inget *spill* alltså och av teckenöverläggningen ser vi hur V-flaggan sätts till 0. Resultatet är skilt från 0 och därmed sätts Z-flaggan till 0.

d)

Operation		Decimalt	
	X	34	0 1 1 0 0 0 1 0
+	2~Y	+ 34	+ 0 0 1 0 0 0 1 0
=	R	68	0 1 0 0 0 1 0 0

$R \neq 0 \rightarrow Z=0$ $N=r_7=1$

$r_7'x_7y_7 + r_7x_7'y_7'=0 \rightarrow V=0$

Det gäller att X=34, Y=-34, korrekt resultat R=68. Operationen innebär X=34, 2~Y=34, R=68. Operationens resultat är korrekt. Inget *spill* alltså och av teckenöverläggningen ser vi hur V-flaggan sätts till 0. Resultatet är skilt från 0 och därmed sätts Z-flaggan till 0.

e)

Operation		Decimalt	
	X	64	0 1 0 0 0 0 0 0
+	2~Y	+ 65	+ 0 1 0 0 0 0 0 1
=	R	-127	1 0 0 0 0 0 0 1

$R \neq 0 \rightarrow Z=0$ $N=r_7=1$

$r_7'x_7y_7 + r_7x_7'y_7'=1 \rightarrow V=1$

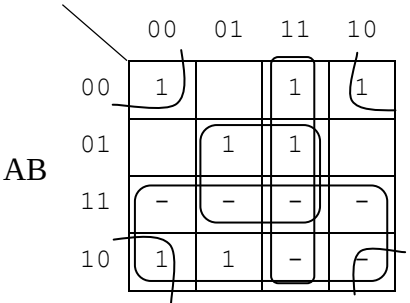
Det gäller att X=64, Y=-65, korrekt resultat R=129. Operationen innebär X=64, 2~Y=65, R=-127. Operationens resultat är fel. Av teckenöverläggningen (två positiva tal adderas men resultatet blir negativt) ser vi att V-flaggan sätts till 1 som indikator på spillet. Resultatet är skilt från 0 varför Z-flaggan sätts till 0.

4 Kombinatoriska nät

4.1

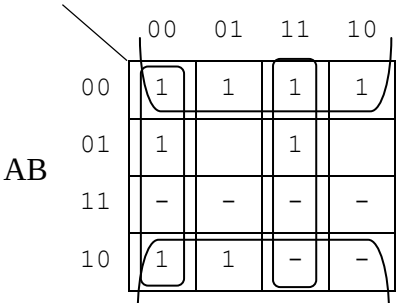
Skapa funktionerna: a,b,c,d,e,f,g och minimera med Karnaughdiagram:

a



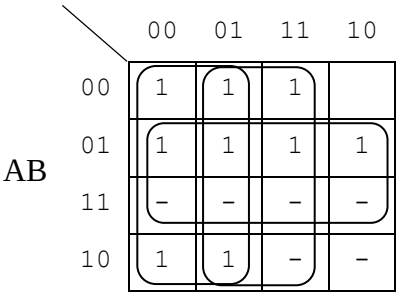
$a = \bar{B}\bar{D} + CD + BD + A$

b



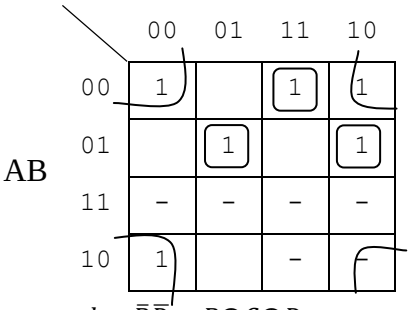
$b = \bar{C}\bar{D} + \bar{B} + CD$

c



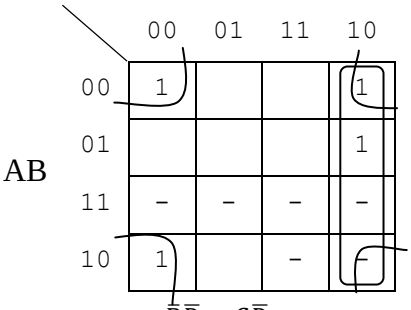
$c = \bar{C} + D + B$

d



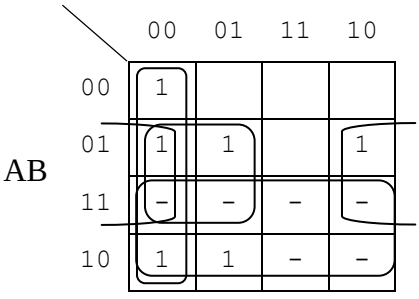
$d = \bar{B}\bar{D} + B \oplus C \oplus D$

e



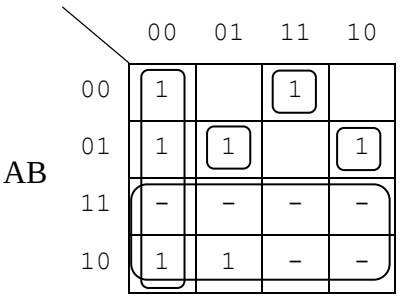
$e = \bar{B}\bar{D} + C\bar{D}$

f



$f = \bar{C}\bar{D} + B\bar{D} + A\bar{C} + A$

g



$g = \bar{C}\bar{D} + A + B \oplus C \oplus D$

Sammanfattningsvis:

$a = \bar{B}\bar{D} + CD + BD + A$
$b = \bar{C}\bar{D} + \bar{B} + CD$
$c = \bar{C} + D + B$
$d = \bar{B}\bar{D} + B \oplus C \oplus D$
$e = \bar{B}\bar{D} + C\bar{D}$
$f = \bar{C}\bar{D} + B\bar{D} + A\bar{C} + A$
$g = \bar{C}\bar{D} + A + B \oplus C \oplus D$

4.2

4.3

Funktionstabell för "4-bitars INCREMENT":

x_3	x_2	x_1	x_0	u_3	u_2	u_1	u_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	1	0	1	0
1	0	1	0	1	0	1	1
1	0	1	1	1	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	0	0	0

u_3

x_3x_2 \ x_1x_0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	1	1	0	1
10	1	1	1	1

$$u_3 = \overline{x_3}\, x_2 x_1 x_0 + x_3 \overline{x_2} + x_3 \overline{x_1} + x_3 \overline{x_0}$$

u_2

x_3x_2 \ x_1x_0	00	01	11	10
00	0	0	1	0
01	1	1	0	1
11	1	1	0	1
10	0	0	1	0

$$u_2 = \overline{x_2}\, x_1 x_0 + x_2 \overline{x_0} + x_2 \overline{x_1}$$

u_1

x_3x_2 \ x_1x_0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$$u_1 = \overline{x_1}\, x_0 + x_1 \overline{x_0}$$

u_0

x_3x_2 \ x_1x_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

$$u_0 = \overline{x_0}$$

4.4

Funktionstabell för "4-bitars DECREMENT":

x_3	x_2	x_1	x_0	u_3	u_2	u_1	u_0
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	0
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	1
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	1
1	0	1	1	1	0	1	0
1	1	0	0	1	0	1	1
1	1	0	1	1	1	0	0
1	1	1	0	1	1	0	1
1	1	1	1	1	1	1	0

u_3

x_3x_2 \ x_1x_0	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	1	1	1

$$u_3 = \overline{x_3}\,\overline{x_2}\,\overline{x_1}\,\overline{x_0} + x_3x_2 + x_3x_0 + x_3x_1$$

u_2

x_3x_2 \ x_1x_0	00	01	11	10
00	1	0	0	0
01	0	1	1	1
11	0	1	1	1
10	1	0	0	0

$$u_2 = \overline{x_2}\,\overline{x_1}\,\overline{x_0} + x_2x_0 + x_2x_1$$

u_1

x_3x_2 \ x_1x_0	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	1	0	1	0
10	1	0	1	0

$$u_1 = \overline{x_1}\,\overline{x_0} + x_1\overline{x_0}$$

u_0

x_3x_2 \ x_1x_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

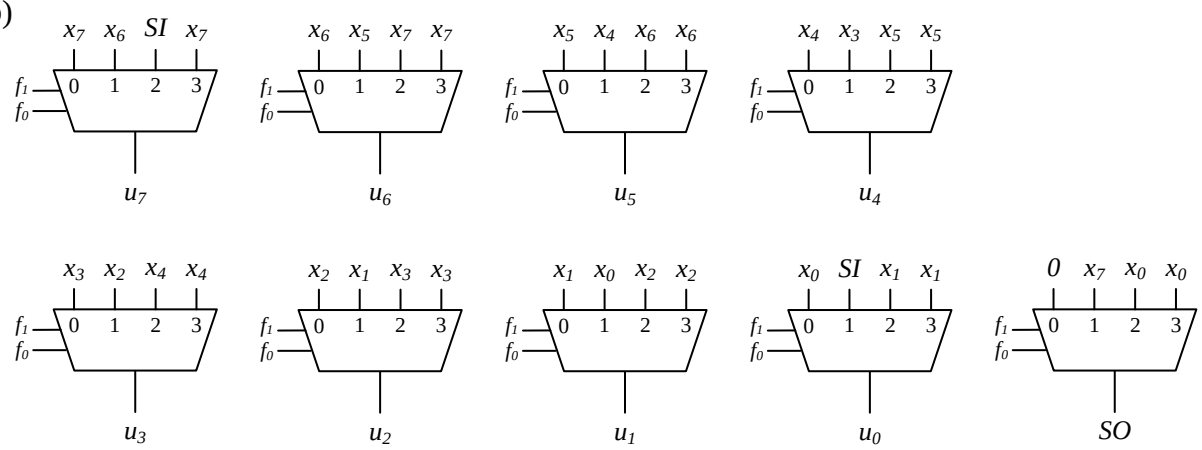
$$u_0 = \overline{x_0}$$

4.5

a)

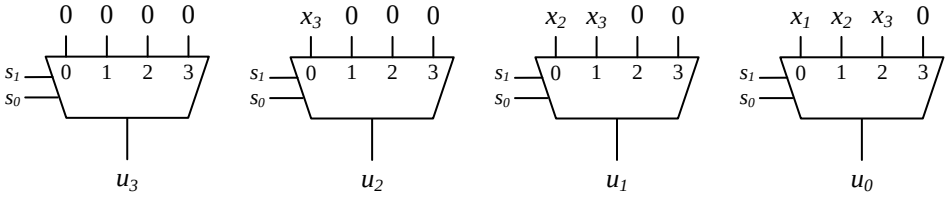
f_1	f_0	u_7	u_6	u_5	u_4	u_3	u_2	u_1	u_0	SO
0	0	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	0
0	1	x_6	x_5	x_4	x_3	x_2	x_1	x_0	SI	x_7
1	0	SI	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
1	1	x_7	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0

b)

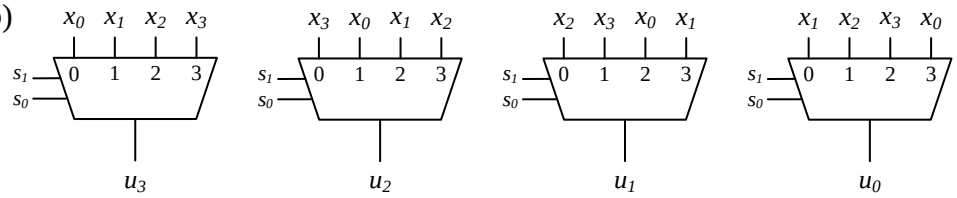


4.6

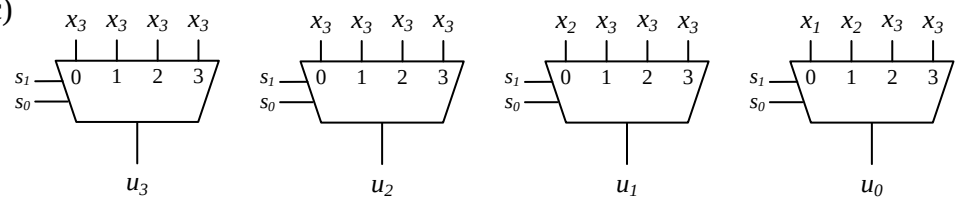
a)



b)



c)

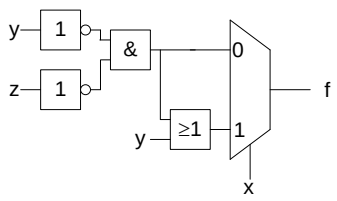


4.7

$$f = \bar{y}\bar{z} + xy$$

x är selektorvariabel:

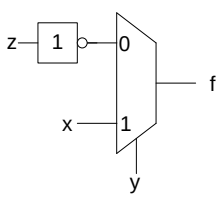
$f = (\bar{x} + x)(\bar{y}\bar{z}) + xy =$
$\bar{x}(\bar{y}\bar{z}) + x(\bar{y}\bar{z}) + xy =$
$= \bar{x}(\bar{y}\bar{z}) + x(\bar{y}\bar{z} + y)$



Totalt 5 grindar

y är selektorvariabel:

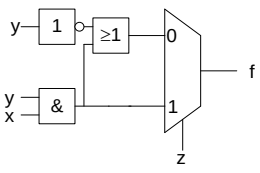
$f = \bar{y}(\bar{z}) + y(x)$



Totalt 1 grind

z är selektorvariabel:

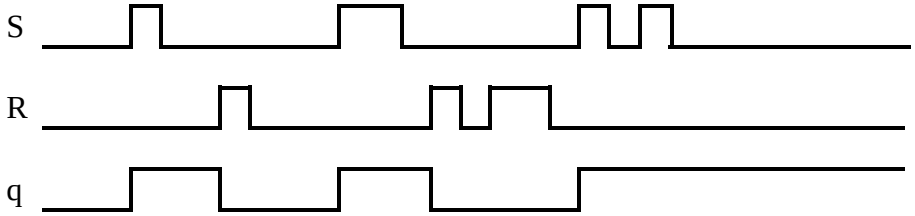
$f = \bar{z}\bar{y} + (z + \bar{z})(xy) =$
$= \bar{z}(\bar{y} + xy) + z(xy)$



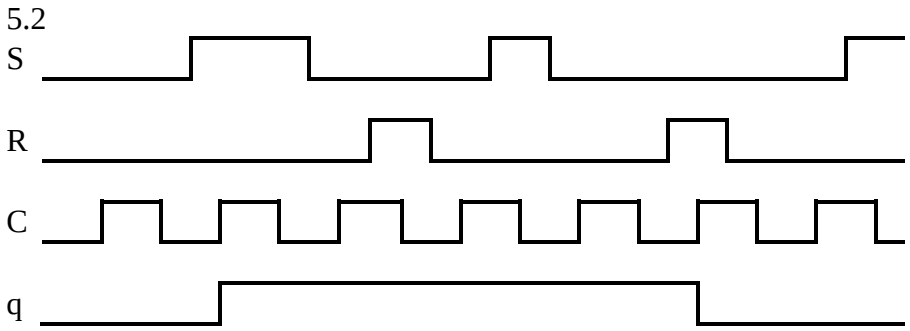
Totalt 3 grindar

5 Sekvensnät

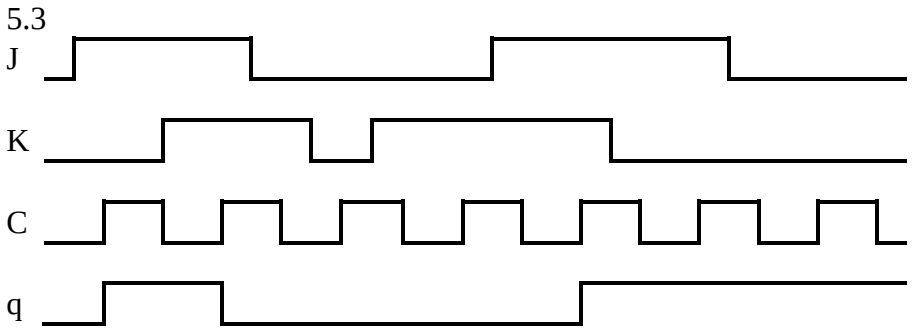
5.1



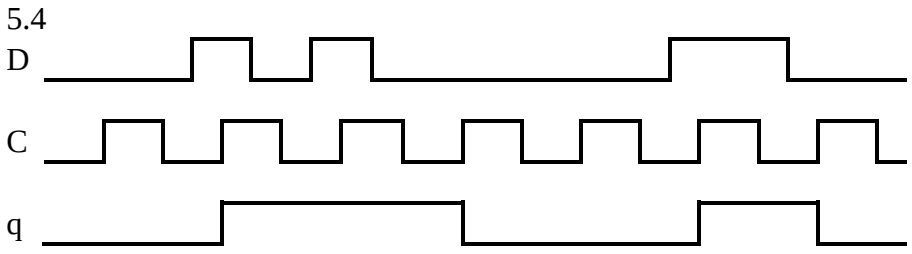
5.2



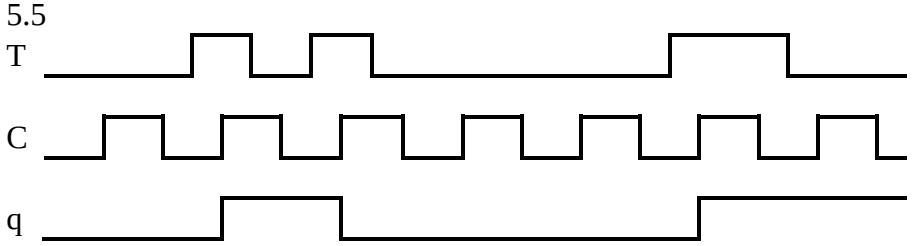
5.3



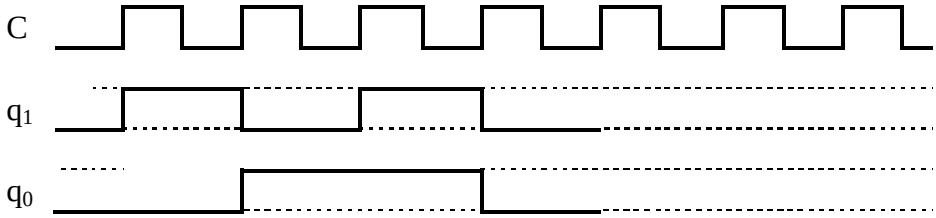
5.4

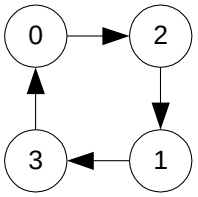


5.5



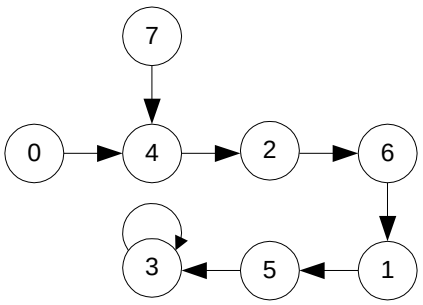
5.6

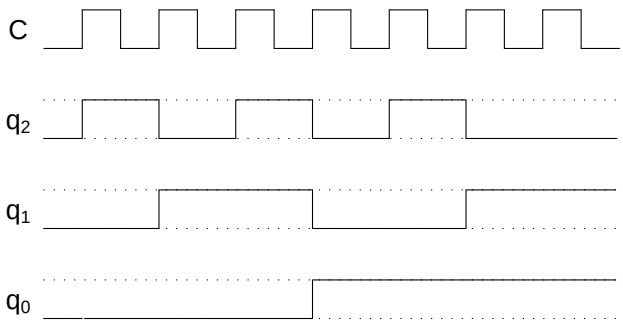




5.7

$T_2 = q_0' + q_1'$
$T_1 = q_2$
$T_0 = q_1\, q_2$

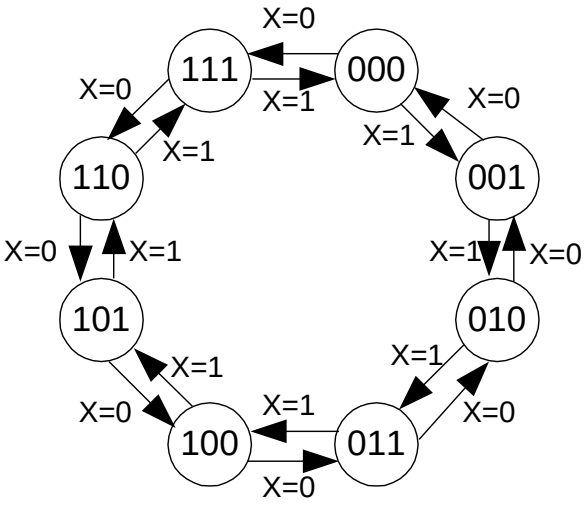




5.8

$T_2 = xq_0q_1 + x'q_0'q_1'$
$T_1 = xq_0 + x'q_0'$
$T_0 = 1$

x	q_2	q_1	q_0	T_2	T_1	T_0	q_2^+	q_1^+	q_0^+
0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	1	0	0	0
0	0	1	0	0	1	1	0	0	1
0	0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	0	1	1
0	1	0	1	0	0	1	1	0	0
0	1	1	0	0	1	1	1	0	1
0	1	1	1	0	0	1	1	1	0
1	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	0
1	0	1	0	0	0	1	0	1	1
1	0	1	1	1	1	1	1	0	0
1	1	0	0	0	0	1	1	0	1
1	1	0	1	0	1	1	1	1	0
1	1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	1	0	0	0

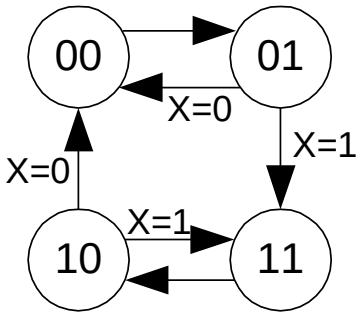


5.9

$q_0^+ = q_1'q_0' + xq_1' + xq_0'$

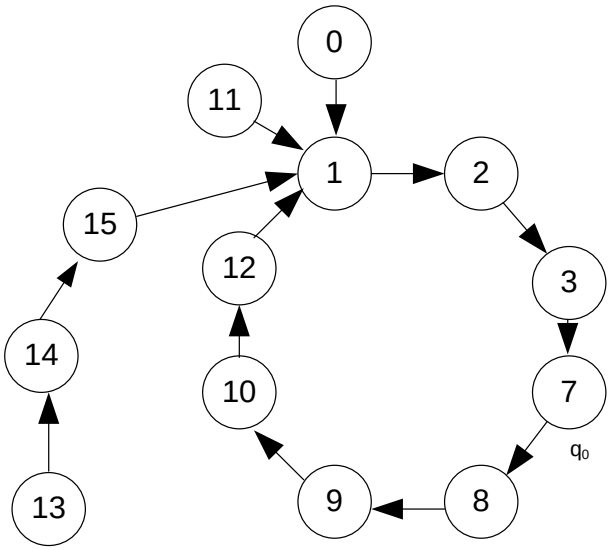
$q_1^+ = q_1q_0 + xq_1 + xq_0$

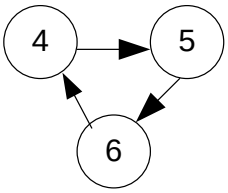
x	q_1	q_0	q_1^+	q_0^+
0	0	0	0	1
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0



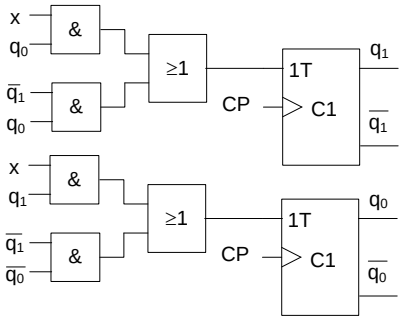
5.10 Alternativ b).

5.11





5.12



x	q_1	q_0	T_1	T_0	q_1^+	q_0^+
0	0	0	0	1	0	1
0	0	1	1	0	0	0
0	1	0	-	-	0	0
0	1	1	0	0	1	0
1	0	0	0	1	0	1
1	0	1	1	0	1	1
1	1	0	-	-	1	1
1	1	1	1	1	1	0

T_1 q_1q_0

x \	00	01	11	10
0	0	1	0	-
1	0	1	1	-

$T_1 = \overline{q_1}q_0 + xq_0$

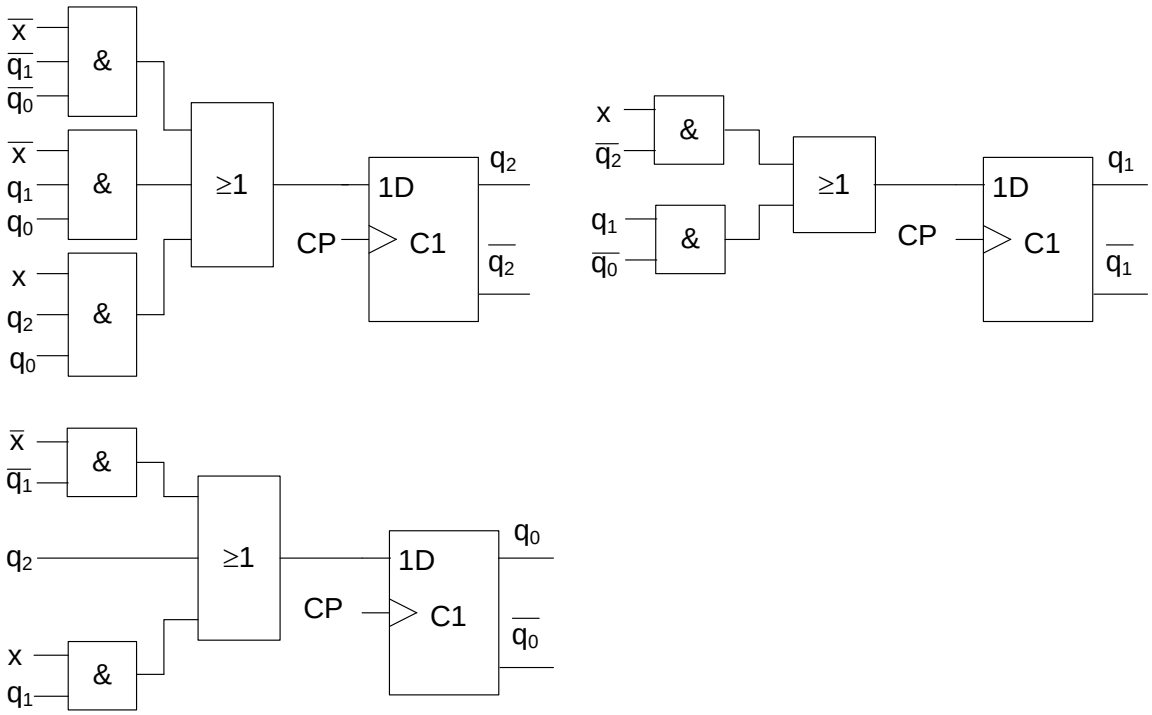
T_0 q_1q_0

x \	00	01	11	10
0	1	0	0	-
1	1	0	1	-

$T_0 = \overline{q_1}\overline{q_0} + xq_1$

5.13

$q_2^+ = \bar{x}\bar{q}_1\bar{q}_0 + \bar{x}q_1q_0 + xq_2q_0$
$q_1^+ = x\bar{q}_2 + q_1\bar{q}_0$
$q_0^+ = \bar{x}\bar{q}_1 + q_2 + xq_1$



5.14

a)

För D-vippor gäller att nästa tillstånd (q^+) är lika med D-värdet före klockningen. Därför skall D-värdet för varje D-vippa väljas till motsvarande q^+-värde, vilket också framgår av tillståndstabellen.

q_1	q_2	q_3	q_1^+	q_2^+	q_3^+	D_1	D_2	D_3
0	0	0	0	0	1	0	0	1
0	0	1	1	0	1	1	0	1
0	1	0	0	0	0	0	0	0
0	1	1	0	1	0	0	1	0
1	0	0	–	–	–	–	–	–
1	0	1	1	1	1	1	1	1
1	1	0	–	–	–	–	–	–
1	1	1	0	1	1	0	1	1

D_1	q_2q_3 00	01	11	10
q_1 0	0	1	0	0
q_1 1	–	1	0	–

$D_1 = \overline{q_2}q_3$

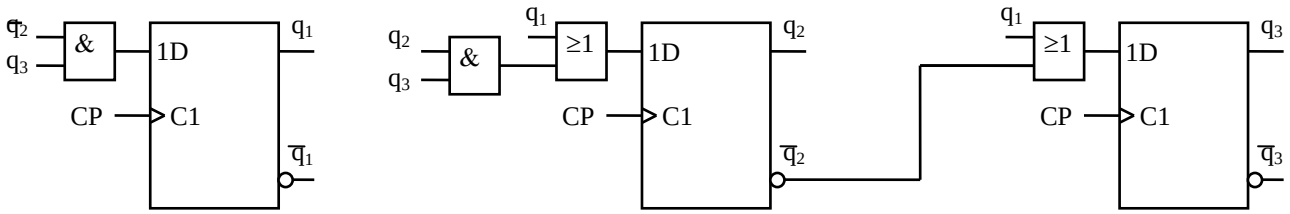
D_2	q_2q_3 00	01	11	10
q_1 0	0	0	1	0
q_1 1	–	1	1	–

$D_2 = q_1 + q_2q_3$

D_3	q_2q_3 00	01	11	10
q_1 0	1	1	0	0
q_1 1	–	1	1	–

$D_3 = q_1 + \overline{q_2}$

Realisering med D-vippor:



b)

q_1	q_2	q_3	q_1^+	q_2^+	q_3^+	T_1	T_2	T_3
0	0	0	0	0	1	0	0	1
0	0	1	1	0	1	1	0	0
0	1	0	0	0	0	0	1	0
0	1	1	0	1	0	0	0	1
1	0	0	–	–	–	–	–	–
1	0	1	1	1	1	0	1	0
1	1	0	–	–	–	–	–	–
1	1	1	0	1	1	1	0	0

T_1	q_2q_3 00	01	11	10
q_1 0	0	1	0	0
q_1 1	–	0	1	–

$$T_1 = \overline{q}_1\overline{q}_2q_3 + q_1q_2$$

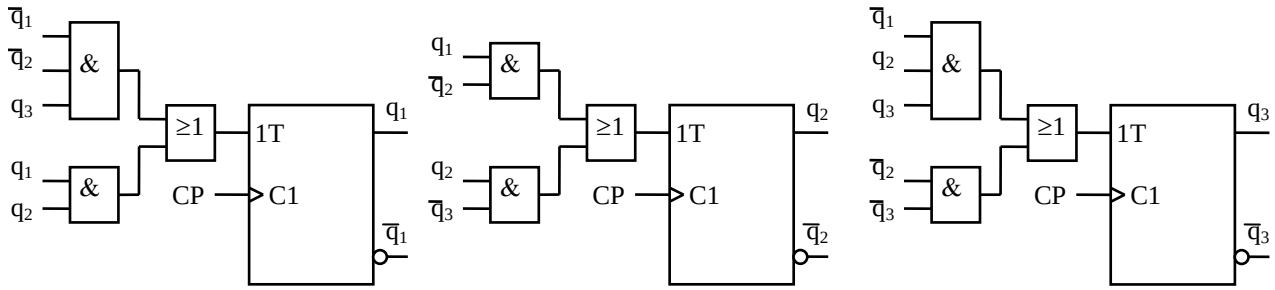
T_2	q_2q_3 00	01	11	10
q_1 0	0	0	0	1
q_1 1	–	1	0	–

$$T_2 = q_1\overline{q}_2 + q_2\overline{q}_3$$

T_3	q_2q_3 00	01	11	10
q_1 0	1	0	1	0
q_1 1	–	0	0	–

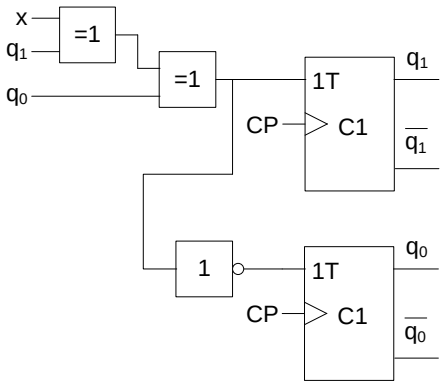
$$T_3 = \overline{q}_2\overline{q}_3 + q_1q_2q_3$$

Realisering med T-vippor



5.15

x	q_1	q_0	T_1	T_0	q_1^+	q_0^+
0	0	0	0	1	0	1
0	0	1	1	0	1	1
0	1	0	1	0	0	0
0	1	1	0	1	1	0
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	1	1
1	1	1	1	0	0	1



T_1	q_1q_0 00	01	11	10
x 0	0	1	0	1
x 1	1	0	1	0

$T_1 = x \oplus q_1 \oplus q_0$

T_0	q_1q_0 00	01	11	10
x 0	1	0	1	0
x 1	0	1	0	1

$T_0 = \overline{(x \oplus q_1 \oplus q_0)}$

5.16

x	q_2	q_1	q_0	T_2	T_1	T_0	q_2^+	q_1^+	q_0^+
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	1	0	0	1	1	0
0	0	1	1	0	0	1	0	1	0
0	1	0	0	1	0	0	0	0	0
0	1	0	1	0	0	1	1	0	0
0	1	1	0	0	0	1	1	1	1
0	1	1	1	0	1	0	1	0	1
1	0	0	0	1	0	0	1	0	0
1	0	0	1	0	0	1	0	0	0
1	0	1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	0	1	1	0	1
1	1	0	1	0	1	0	1	1	1
1	1	1	0	1	0	0	0	1	0
1	1	1	1	0	0	1	1	1	0

T_2

xq_2 \ q_1q_0	00	01	11	10
00				1
01	1			
11				1
10	1			

$T_2 = \overline{q_0}(x \oplus q_1 \oplus q_2)$

T_1

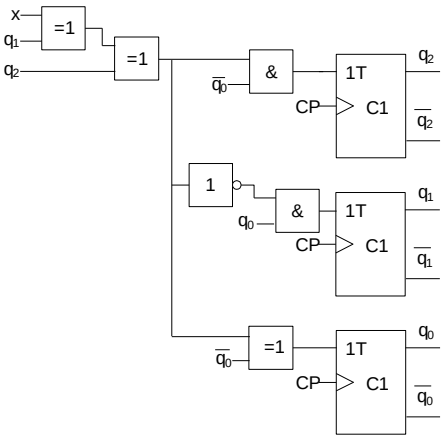
xq_2 \ q_1q_0	00	01	11	10
00		1		
01			1	
11		1		
10			1	

$T_1 = q_0(\overline{x \oplus q_1 \oplus q_2})$

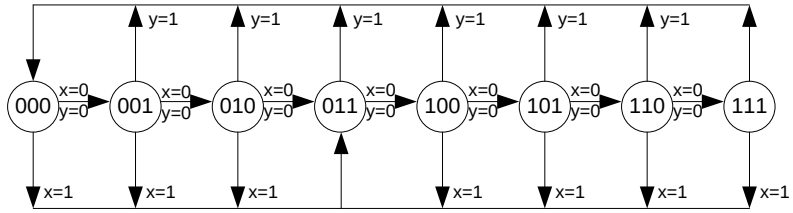
T_0

xq_2 \ q_1q_0	00	01	11	10
00	1		1	
01		1		1
11	1		1	
10		1		1

$T_0 = \overline{(x \oplus q_2 \oplus q_1 \oplus q_0)}$



5.17



x	y	q_2	q_1	q_0	q_2^+	q_1^+	q_0^+
1	0	x	x	x	0	1	1
0	1	x	x	x	0	0	0
1	1	x	x	x	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	1	0	1	0
1	0	0	1	0	0	1	1
1	1	0	1	1	1	0	0
0	0	1	0	0	1	0	1
0	1	1	0	1	1	1	0
1	0	1	1	0	1	1	1
1	1	1	1	1	0	0	0

D_2	00	01	11	10
0	0	0	1	0
1	1	1	0	1

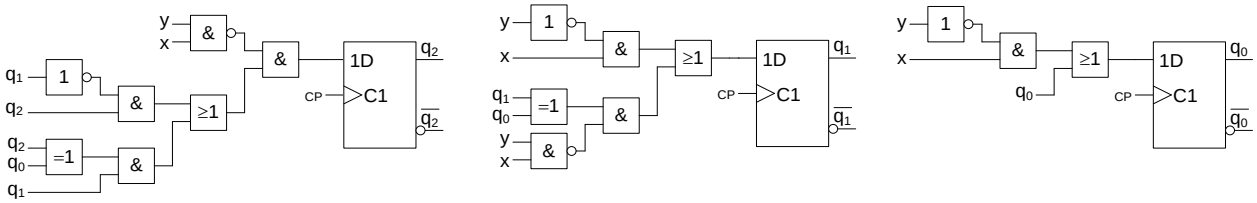
$$D_2 = \overline{xy}[q_2\overline{q_1} + (q_2 \oplus q_0)q_1]$$

D_1	00	01	11	10
0	0	1	0	1
1	0	1	0	1

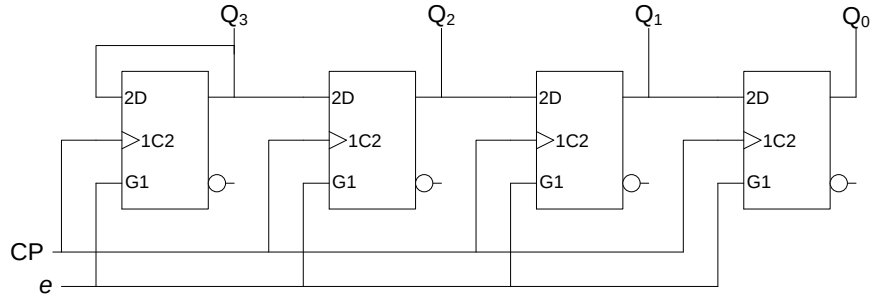
$$D_1 = x\bar{y} + \overline{xy}(q_1 \oplus q_0)$$

D_0	00	01	11	10
0	1	0	0	1
1	1	0	0	1

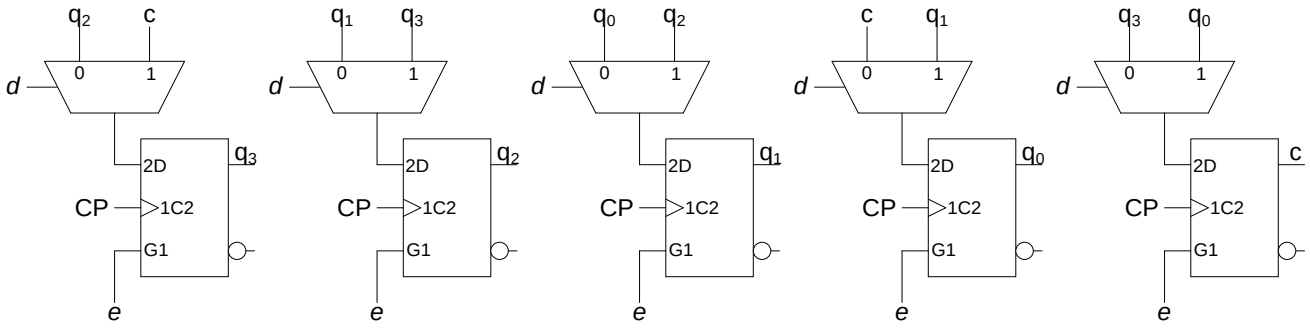
$$D_0 = x\bar{y} + \overline{q_0}$$



5.18



5.19



6 Dataväg ALU och minne

6.1

	Operation (RTN)	Aktiva styrsignaler
a)	$T \rightarrow A$	LD_A,OE_T
b)	$T \rightarrow A$,R	LD_A, LD_R, OE_T
c)	$R \rightarrow T$, $A \rightarrow R$ $T \rightarrow A$ $= A \leftrightarrow R$	1) OE_R, LD_T 2) OE_A, LD_R 3) OE_T, LD_A

6.2 a)

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$0 \rightarrow R$	LD_R
2	$R \rightarrow A$	OE_R, LD_A
eller...		
1	$0 \rightarrow A$	Source = 0; OE_S, LD_A

b)

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$A+1 \rightarrow R$	OE_A, F(1,0,0,1), C_{in}, LD_R
2	$R \rightarrow A$	OE_R, LD_A

c)

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$\sim A \rightarrow R$	OE_A, F(0,1,0,1), LD_R
2	$R \rightarrow A$	OE_R, LD_A

d)

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$2\sim A \rightarrow R$	OE_A, F(0,1,0,1), C_{in}, LD_R
2	$R \rightarrow A$	OE_R, LD_A

6.3

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$20_{16} \rightarrow A$	Source=20; LD_A
2	$A-1 \rightarrow R$	F(1,0,1,0); OE_A; LD_R
3	$R \rightarrow A$	OE_R; LD_A
4	$F0_{16} \rightarrow T$	Source=F0; LD_T
5	$A \oplus T \rightarrow R$	F(1,0,0,0); OE_A; LD_R
6	$R \rightarrow A$	OE_R; LD_A

6.4 a)

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$10_{16} \rightarrow TA$	Source = 10_{16}; OE_S; LD_{TA}
2	$(TA) \rightarrow A$	MR; LD_A

b)

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$10_{16} \rightarrow TA$	Source = $(10)_{16}$; OE_S; LD_{TA}
2	$(TA) \rightarrow TA$	MR; LD_{TA}
3	$(TA) \rightarrow A$	MR; LD_A

c)

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$A \rightarrow TA$	OE_A; LD_{TA}
2	$(TA) \rightarrow A$	MR; LD_A

d)

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$10_{16} \rightarrow$ **T**A	Source = 10_{16}; OE_S; LD_{TA}
2	$A \rightarrow (TA)$	MW; OE_A

6.5

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$14_{16} \rightarrow$ TA	Src=14_{16}; OE_S; LD_{TA}
2	M(TA)$\rightarrow$ T	MR; LD_T
3	A+T$\rightarrow$ R	OE_A; f_3; f_1; f_0; LD_R
4	R$\rightarrow$ A	OE_R; LD_A

6.6

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$13_{16} \rightarrow$ TA	Src=13_{16}; OE_S; LD_{TA}
2	M(TA)$\rightarrow$ T	MR; LD_T
3	$12_{16} \rightarrow$ TA	Src=12_{16}; OE_S; LD_{TA}
4	M(TA)+T$\rightarrow$ R	MR; f_3; f_1; f_0; LD_R
5	$14_{16} \rightarrow$ TA	Src=14_{16}; OE_S; LD_{TA}
6	R$\rightarrow$ M(TA)	OE_R; MW

6.7

Cykel	Operation (RTN)	Aktiva styrsignaler
1	$13_{16} \rightarrow$ TA	Src=13_{16}; OE_S; LD_{TA}
2	M(TA)$\rightarrow$ T	MR; LD_T
3	A+T$\rightarrow$ R ALU(N,V,Z,C) $\rightarrow$ CC	OE_A; f_3; f_1; f_0; LD_R; LD_{CC}
4	$21_{16} \rightarrow$ TA	Src=21_{16}; OE_S; LD_{TA}
5	R$\rightarrow$ M(TA)	OE_R; MW
6	$12_{16} \rightarrow$ TA	Src=12_{16}; OE_S; LD_{TA}
7	M(TA)+C$\rightarrow$ R	MR; f_3; f_0; g_1; LD_R
8	$20_{16} \rightarrow$ TA	Src=20_{16}; OE_S; LD_{TA}
9	R$\rightarrow$ M(TA)	OE_R; MW

7 Styrenheten

7.1 a)

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	M(SP)→Y;	LD_Y; g_{12};MR
Q_5	SP+1→SP;	INC_{SP}; NF

b) Instruktionen är `PULY`

7.2 a)

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	SP-1→SP;	DEC_{SP}
Q_5	Y→M(SP);	OE_Y; g_{12}; MW; NF

b) Instruktionen är `PSHY`

7.3 a)

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	n→T; PC+1→PC	MR; LD_T; INC_{PC}
Q_5	Y→M(T+X);	OE_Y; MW; g_{13}; g_{12}; NF

b) Instruktionen är `STY n,X`

7.4 a)

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	n→T;	MR; LD_T;
Q_5	M(T+X)→R;	OE_X; f_3;f_1; f_0;LD_R;
Q_6	R→PC;	OE_R; LD_{PC}; NF

b) Instruktionen är `JMP n,X`

7.5 a)

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	A'→R;	OE_A; f_2;f_0;LD_R;
	ALU(N,Z) →CC(n,Z); 0→V;	g_5; g_3; g_2; LD_{CC}
Q_5	R→A;	OE_R; LD_A; NF

b) Instruktionen är `COMA`

7.6 a)

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	(d7)A>>1→R;	OE_A; f_3;f_2;f_1; f_0;LD_R; LD_{CC}
Q_5	R→A;	OE_R; LD_A; NF

b) Instruktionen är `ASRA`

7.7 a)

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	A→T	OE_A; LD_T
Q_5	(C)(M(A+Y))>>1→R;	MR; g_{13}; f_3;f_2;f_1; g_1;LD_R; LD_{CC}
Q_6	R→ M(A+Y);	OE_R; MW; g_{13}; NF

b) Instruktionen är `ROR A,Y`

7.8 a)

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	PC→TA,T;	OE_{PC}; LD_{TA}; LD_T
Q_5	M(TA)+1→R; PC+1→PC;	MR; f_3;f_1;f_0; g_0;LD_R; INC_{PC}
Q_6	if(N=1) R→ PC;	OE_R; LD_{PC}=N; NF

b) Instruktionen är `BMI Adr`

7.9 a)

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	PC→TA,T;	OE_{PC}; LD_{TA}; LD_T
Q_5	M(TA)+1→R; PC+1→PC;	MR; f_3;f_1;f_0; g_0;LD_R; INC_{PC}
Q_6	if(C+Z=1) R→ PC;	OE_R; LD_{PC}=C+Z; NF

b) Instruktionen är `BLS Adr`

7.10

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q4	M(PC)→TA; PC+1→PC;	MR; LD_{TA}; INC_{PC};
Q5	M(TA)'→R;	MR; g_{14}; f_2;f_0;LD_R;
	ALU(N,Z) →CC(n,Z); 0→V;	g_5; g_3; g_2; LD_{CC}
Q6	R→M(TA);	OE_R; MW; g_{14}; NF

7.11

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
0	A→T;	OE_A; LD_T;
1	M(T+X)→R;	OE_X; f_3;f_1; f_0;LD_R;
2	R→PC;	OE_R; LD_{PC}; NF

7.12

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	n→T; PC+1→PC	MR; LDT; INC_{PC}
Q_5	M(T+SP)>>1(C)→R;	MR; g_{12}; f_3;f_2;f_0; g_1;LD_R;
	ALU(N,V,Z,C)→CC(N,V,Z,C)	LD_{CC};
Q_6	R→ M(T+SP);	OE_R; MW; g_{12}; NF

7.13

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	PC→TA,T;	OE_{PC}; LD_{TA};LD_T
Q_5	M(TA)+1→R; PC+1→PC;	MR; f_3;f_1;f_0; g_0;LD_R; INC_{PC}
Q_6	if(N⊕V=1) R→ PC;	OE_R; LD_{PC}=N⊕Z; NF

8 Assemblerprogrammering

8.1 OPERAND = Adress – (PC+2) = 81 – (59+2) = 26
(Alla adressvärden på hexadecimal form)

8.2 OPERAND = Adress – (PC+2) = 10 – (62+2) = AC
(Alla adressvärden på hexadecimal form)

8.3

```
LDAA          #$85          ; Talet (85)16 till A
CMPA          #U            ; Bilda (85)16 - U
B(Villkor)    Hopp
```

Storleksjämförelse görs alltså mellan $(85)_{16} = (133)_{10}$ och U. Alla hoppvillkoren i uppgifter a,b,c och d avser tal *utan* inbyggt tecken. För 8-bitars tal utan inbyggt tecken gäller: $0 \le U \le 255$

Om det villkorliga hoppet är:

a) `BHI` (*Higher*) utförs hoppet om 133 > U, dvs U < 133 Svar: **0 ≤ U < 133**
b) `BHS` (*Higher or Same*) utförs hoppet om 133 ≥ U, dvs U ≤ 133 Svar: **0 ≤ U ≤ 133**
c) `BLS` (*Lower or Same*) utförs hoppet om 133 ≤ U, dvs U ≥ 133 Svar: **133 ≤ U ≤ 255**
d) `BLO` (*Lower*) utförs hoppet om 133 < U, dvs U > 133 Svar: **133 < U ≤ 255**

Alla hoppvillkoren i uppgifter e,f,g och h avser tal *med* inbyggt tecken (2-komplementrepresentation).
För 8-bitars tal med inbyggt tecken (2-komplementrepresentation) gäller: $-128 \le U \le +127$
Talet $(85)_{16}$ tolkas som det negativa talet $-(100 - 85)_{16} = -(7B)_{16} = -(123)_{10}$

Om det villkorliga hoppet är:

e) `BGT` (*Greater Than*) utförs hoppet om -123 > U, dvs U < -123
U skall alltså tillhöra intervallet [-128, -123) Svar: **128 ≤ U < 133**
f) `BGE` (*Greater or Equal*) utförs hoppet om -123 ≥ U, dvs U ≤ -123
U skall alltså tillhöra intervallet [-128, -123] Svar: **128 ≤ U ≤ 133**
g) `BLE` (*Less or Equal*) utförs hoppet om -123 ≤ U, dvs U ≥ -123
U skall alltså tillhöra intervallet [-123, +127] Svar: **0 ≤ U ≤ 127 eller 133 ≤ U ≤ 255**
h) `BLT` (*Less Than*) utförs hoppet om -123 < Y, dvs U > -123
U skall alltså tillhöra intervallet (-123, +127] Svar: **0 ≤ U ≤ 127 eller 133 < U ≤ 255**

8.4

Adress	Innehåll	Assemblerkod (disassemblering)	
		ORG	$E0
E0	87	FCB	%10000111
E1	??	RMB	2
E2	??		
E3	7A	FCB	$7A
E4	61	FCB	´a´
E5	41	FCS	”ABCD”
E6	42		
E7	43		
E8	44		

8.5

Adress	Innehåll	Assemblerkod (disassemblering)		
			ORG	$30
30	90	Start:	LDX	#$0C
31	0C			
32	F5	Loop:	LDA	,X+
33	E1		STA	$FC
34	FC			
35	09		TSTA	
36	23		BPL	Loop
37	FA			
38	33	Stop:	JMP	Stop
39	38			

8.6

Adress	Innehåll	Assemblerkod (disassemblering)		
			ORG	$20
		Length:	EQU	4
20	90	Start:	LDX	#Table
21	32			
22	F1		LDA	Length
23	04			
24	E1		STA	Count
25	31			
26	F5	Loop:	LDA	,X+
27	E1		STA	$FC
28	FC			
29	38		DEC	Count
2A	31			
2B	F1		LDA	Count
2C	31			
2D	25		BNE	Loop
2E	F7			
2F	33	Stop	Jmp	Stop
30	2F			
31	??	Count	RMB	1
32	10	Table:	FCB	$10,%10,10,0
33	02			
34	0A			
35	00			

8.7

```
20:       LDX  #start
22:       LDY  #dest
24: L1:   LDA  ,X+
25:       BEQ  L2
27:       STA  ,Y+
28:       BRA  L1
2A:  L2:  BRA  L2
```

8.8

```
20:       LDA  $7A
22:       NEGA
23:       RORA
24:       ADDA $C4
26:       ADCA $C5
28:       BEQ  L2
2A:       LDA  #1
2C:  L2:  BRA  L2
```

8.9

PC	**22**	**25**	**45**	**60**	**49**	**27**
SP	F0	EF	EB	EA	EB	EF
(SP)	-	50	10	49	10	50
(SP+1)	-	-	50	10	50	-
(SP+2)	-	-	27	50	27	-
(SP+3)	-	-	50	27	50	-
(SP+4)	-	-	-	50	-	-

8.10

```
; Symboliska adresser
DipSwitch: EQU       $FB
HexDisp:   EQU       $FC

; Subrutin DipHex
DipHex:    LDA       DipSwitch
           LDX       #0

DipHex10:  TSTA
           BEQ       DipHex20
           LEAX      1,X
           LSRA
           BCC       DipHex10

DipHex20:  STX       HexDisp
           RTS
```

8.11

```
DipSwitch1:     EQU       $FB
DipSwitch2:     EQU       $FC
LedDisplay:     EQU       $FB

main:      JSR       DipSwitchOr
           BRA       main

DipSwitchOr:
           LDA       DipSwitch1
           ORA       DipSwitch2
           STA       LedDisplay
           RTS
```

8.12

```
;     Huvudprogram
BOS:       EQU       $20
INPORT:    EQU       $FB
UTPORT:    EQU       $FC
Error:     EQU       $5D       ; Felkod (E)

           ORG       $20
START:     LDX       #SegCode  ; Segmenttabell för 7-segmentdisplay

LOOP:      LDA       INPORT    ; Läs inport $FD

           STA       TEMP      ; Skapa kopia
           ANDA      #$0F      ; Maska bort bit 4-7
           STA       LNIB      ; Lagra låg nibble i minnet

           LDA       TEMP      ; Hämta tillbaka kopian
           LSRA                ; Placera bit7-4 till höger i A-reg
           LSRA
           LSRA
           LSRA

           ADDA      LNIB      ; Addera nibblar

           CMPA      #$09      ; >9?
           BHI       ERROR     ; Ja! Visa feltecken på display

           LDA       A,X       ; Nej, visa summasiffra på display
           STA       UTPORT
           BRA       LOOP      ; Nästa varv

ERROR:     LDA       #Error    ; Felkod
           STA       UTPORT
           BRA       LOOP      ; Nästa varv

TEMP:      RMB       1         ; Plats för kopia av A
```

```
LNIB:      RMB       1          ; Plats för nibblekopia

SegCode:   FCB       $77,$22,$5B,$6B,$2E,$6D,$7D,$23,$7F,$6F

           ORG       $FF
           FCB       START      ; RESET-vektor
```

8.13

```
;     Subrutin CNTONE
CNTONE:    CLR       COUNT      ; Nolla 1-räknare

CTLOOP:    TSTA                 ; Reg A tomt?
           BEQ       CNTOUT     ; Ja, återvänd

           LSLA                 ; Nej, skifta ut bit i carry
           BCC       CTLOOP     ; Carry=0? Ja, fortsätt

           INC       COUNT      ; Carry= 1, öka 1-räknare
           BRA       CTLOOP     ; Fortsätt

CNTOUT:    LDA       COUNT      ; Läs av räknare
           RTS

COUNT:     RMB       1          ; Plats för 1-räknare
```

8.14

```
;     Subrutin ASCBIN
ASCBIN:    ANDA       #$7F       ;Maska bort bit 7

           SUBA       #$30       ;Justera för siffror 0-9
           BLO        ASCERR     ;Ej siffra. Fel

           CMPA       #9         ;Siffra 0-9?
           BLS        ASCOK      ;Ja, återhopp

           SUBA       #7         ;Nej, justera för siffra A-F
           CMPA       #10        ;Siffra A-F?
           BLO        ASCERR     ;Nej. Fel

           CMPA       #15        ;Siffra A-F?
           BLS        ASCOK      ;Ja, återhopp

ASCERR:    LDA        #$FF       ;Ej siffra, sätt felflagga
ASCOK:     RTS
```

8.15

```
;     Subrutin CCOUNT

CCOUNT:    PSHX                  ;Spara register på stack

           CLR        COUNT      ;Nollställ "C"-räknare

CCLOOP:    LDA        ,X+        ;Hämta data från textsträng
           BEQ        CCEXIT     ;Strängslut

           ANDA       #$7F       ;Nej, maska bort bit 7

           CMPA       #'C'       ;Testa om "C"
           BNE        CCLOOP     ;Nej, fortsätt med nästa

           INC        COUNT      ;Ja, öka "C"-räknare
           BRA        CCLOOP     ;Fortsätt med nästa

CCEXIT:    LDA        COUNT      ;Hämta resultat
           PULX                  ;Återställ register
           RTS

COUNT:     RMB        1          ;Räknare för "C"
```

8.16

```
;     Subrutin AaCNT
AaCNT:     PSHX                  ;Spara register på stack
           CLR        COUNT      ;Nollställ "Aa"-räknare

AaLOOP:    LDA        ,X+        ;Hämta data från textsträng
           BEQ        AaEXIT     ;Strängslut

           ANDA       #$7F       ;Nej, maska bort bit 7 (paritetsbit)
           ORA        #$20       ;Gör om till liten bokstav (ettställ bit 5)
           CMPA       #'a'       ;Testa om "a"
           BNE        AaLOOP     ;Nej, fortsätt med nästa

           INC        COUNT      ;Ja, öka "Aa"-räknare
           BRA        AaLOOP     ;Fortsätt med nästa

AaEXIT:    LDA        COUNT      ;Hämta resultat
           PULX                  ;Återställ register
           RTS

COUNT:     RMB        1          ;Plats för räknare
```

8.17

```
;     Subrutin SMALLCNT
SMALLCNT:
          PSHX                ;Spara register på stack

          CLR       COUNT     ;Nollställ bokstavsräknare

SCLOOP:   LDA       ,X+       ;Hämta data från textsträng
          BEQ       SCEXIT    ;Strängslut

          ANDA      #$7F      ;Nej, maska bort bit 7

          CMPA      #'a'      ;Testa om "a"
          BLO       SCLOOP    ;Ej liten bokstav, fortsätt med nästa
          CMPA      #'z'      ;Testa om "z"
          BHI       SCLOOP    ;Ej liten bokstav, fortsätt med nästa

          INC       COUNT     ;Ja, öka bokstavsräknare
          BRA       SCLOOP    ;Fortsätt med nästa

SCEXIT:   LDA       COUNT     ;Hämta resultat
          PULX                ;Återställ register
          RTS

COUNT:    RMB       1         ;Bokstavsräknare
```

8.18

```
;     Subrutin LCOUNT
LCOUNT:   PSHX                ;Spara register på stack

          CLR       COUNT     ;Nollställ bokstavsräknare

LLOOP:    LDA       ,X+       ;Hämta data från textsträng
          BEQ       LEXIT     ;Strängslut

          ANDA      #$7F      ;Nej, maska bort bit 7
          ORA       #$20      ;Ettställ bit 5. (Stora bokstäver -> små)
          CMPA      #'a'      ;Testa om "a"
          BLO       LLOOP     ;Ej liten bokstav, fortsätt med nästa
          CMPA      #'z'      ;Testa om "z"
          BHI       LLOOP     ;Ej liten bokstav, fortsätt med nästa

          INC       COUNT     ;Ja, öka bokstavsräknare
          BRA       LLOOP     ;Fortsätt med nästa

LEXIT:    LDA       COUNT     ;Hämta resultat
          PULX                ;Återställ register
          RTS

COUNT:    RMB       1         ;Plats för bokstavsräknare
```

8.19

```
*     Subrutin PRINT
PRINT:    PSHA                ;Spara register på stack
          PSHX

PLOOP:    LDA       ,X+       ;Hämta data från textsträng
          BEQ       PREXIT    ;Strängslut
          ANDA      #$7F      ;Nej, maska bort ev bit 7

WAIT:     TST       STATUS    ;Vänta tills skrivaren redo. Bit 7 (N-flaggan=0?)
          BPL       WAIT      ;Ja, vänta tills N = 1
          STA       UTPORT
          BRA       PLOOP

PREXIT:   PULX                ;Återställ register
          PULA
          RTS
```

8.20
uppgift d)

```
LedDisplay:     EQU         $FC
                ORG         $20
main:           JSR         Blink
                BRA         main
```

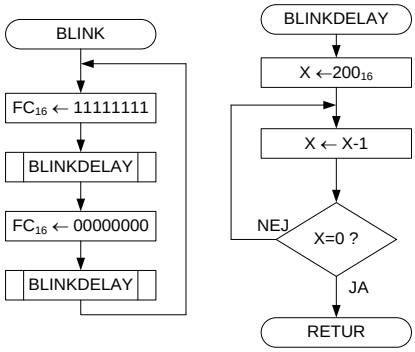
; uppgift a)

```
Blink:          LDA         #$FF
                STA         LedDisplay
                JSR         BLINKDELAY
                LDA         #0
                STA         LedDisplay
                JSR         BLINKDELAY
                BRA         Blink
```

; uppgift b)

```
BLINKDELAY:
                LDX         #$200
BLINKDELAY1:
                LEAX        -1,X
                CPX         #0
                BNE         BLINKDELAY1
                RTS
```

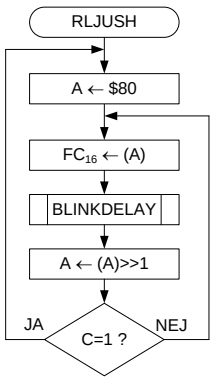
; uppgift c)



8.21
a) b)

```
LedDisplay:     EQU         $FC
RLJUSH:         LDA         #$80
RLJUSH1:
                STA         LedDisplay
                JSR         BLINKDELAY
                RORA
                BCS         RLJUSH
                BRA         RLJUSH1
```

c)



8.22

```
; Subrutin RLJUSH16
LedHigh:   EQU        $FB
LedLow:    EQU        $FC

RLJUSH16:  LDA        #$80
           CLR        TMP
RLJUSH16_1:
           LDX        TMP
           STX        LedLow
           STA        LedHigh
           JSR        BLINKDELAY
           RORA
           ROR        TMP
           BCS        RLJUSH16
           BRA        RLJUSH16_1
TMP:       RMB        1
```

8.23

```
;     Huvudprogram
BOS:       EQU        $20
INPORT:    EQU        $FB
UTPORT:    EQU        $FC

           ORG        $20
START:     LDSP       #BOS
           LDX        #SegCode   ;Segmenttabell för 7-segmentdisplay

LOOP:      LDA        INPORT     ; Läs inport $FB
           ANDA       #$0F       ; Maska bort bit 4-7
           STA        COPY       ; Spara kopia av avläst värde
           LDA        A,X        ; Hämta segmentkod för hexsiffra
           STA        UTPORT     ; Visa hexsiffra på 7-segmentdisplay
           LDA        COPY       ; Hämta tillbaka sparad kopia
           JSR        DELAY      ; Vänta ett antal sekunder som bestäms av A-reg
           CLR        UTPORT     ; Släck display
           JSR        DELAY      ; Vänta ett antal sekunder som bestäms av A-reg
           BRA        LOOP       ; Upprepa

COPY:      RMB        1          ; Plats för kopia
SegCode:   FCB  $77,$22,$5B,$6B,$2E
           FCB  $6D,$7D,$23,$7F,$6F
           FCB  $3F,$7C,$55,$7A,$5D,$1D

           ORG        $FF
           FCB        START      ; RESET-vektor
```

8.24

```
;     Subrutin KEYCMP som jämför två strängar och returnerar antal fel
KEYCMP:    PSHX                  ;Spara register på stack
           PSHY
           CLR        ERRNUM     ;Nollställ felräknare

KLOOP:     LDA        ,X+        ;Hämta data från "User string"
           BEQ        KEXIT      ;Strängslut
           ANDA       #$7F       ;Maska bort bit 7
           STA        TEMP
           LDA        ,Y+        ;Hämta tecken från nyckelsträng
           CMPA       TEMP       ;Jämför med nyckel
           BEQ        KLOOP      ;Data matchar, testa nästa
           INC        ERRNUM     ;Data stämmer ej med nyckel. Öka felräknare
           BRA        KLOOP      ;Testa nästa

KEXIT:     LDA        ERRNUM     ;Hämta antal fel
           PULY                  ;Återställ register
           PULX
           RTS

TEMP:      RMB        1          ;Temporärvariabel
ERRNUM:    RMB        1          ;Variabel för antal fel
```

8.25

```
BOS:      EQU       $20
DIPSW:    EQU       $FC
HEXDIS:   EQU       $FB

          ORG       $20
START:    LDSP      #BOS       ;Initiering av stackpekare

LOOP:     LDA       DIPSW
          STA       TEMP       ;Gör kopia
          CMPA      #$0C       ;00001100 SUB4
          BNE       NEXT1
          JSR       SUB4
          BRA       LOOP

NEXT1:    ANDA      #%00110011
          CMPA      #%00010010 ;??01??10 SUB2
          BNE       NEXT2
          JSR       SUB2
          BRA       LOOP

NEXT2:    LDA       TEMP       ;Hämta kopia av A-reg
          ANDA      #%00010001
          CMPA      #%00010001 ;???1???1 SUB1
          BNE       NEXT3
          JSR       SUB1
          BRA       LOOP

NEXT3:    LDA       TEMP       ;Hämta kopia av A-reg
          ANDA      #%00010010
          CMPA      #%00000010 ;???0??1? SUB3
          BNE       NEXT4
          JSR       SUB3
          BRA       LOOP

NEXT4:    CLR       HEXDIS
          BRA       LOOP

TEMP:     RMB       1          ;För kopia av A-reg

;*******************************************************************************
SUB1:     PSHA
          LDA       #1
          STA       HEXDIS
          PULA
          RTS
;*******************************************************************************

SUB2:     PSHA
          LDA       #2
          STA       HEXDIS
          PULA
          RTS
;*******************************************************************************

SUB3:     PSHA
          LDA       #3
          STA       HEXDIS
          PULA
          RTS
;*******************************************************************************

SUB4:     PSHA
          LDA       #4
          STA       HEXDIS
          PULA
          RTS
;*******************************************************************************
          ORG       $FF
          FCB       START      ;RESET-vektor
```

8.26

```
BOS:      EQU       $20
DIPSW:    EQU       $FB
HEXDIS:   EQU       $FB

          ORG       $20
START:    LDSP      #BOS       ;Initiering av stackpekare
LOOP:     LDX       #JTAB

          LDA       DIPSW      ;Läs inport DIPSW
          CMPA      #7         ;Maxvärde
          BHI       ERROR      ;Ogiltigt. Visa E på HEXDISPLAY

          LDA       A,X        ;Ladda adress till subrutin från tabell
          STA       TEMP
          LDX       TEMP
          JSR       0,X        ;Utför vald subrutin
          BRA       LOOP       ;Upprepa

ERROR:    LDA       #$E        ;Felkod
          STA       HEXDIS
          BRA       LOOP

TEMP:     RMB       1

JTAB:     FCB       $80,$67,$75,$52,$90,$B9,$AF,$E0
```

8.27

```
BOS:      EQU       $20
INNUMB:   EQU       $FB
INEDGE:   EQU       $FC
UTPORT:   EQU       $FB

          ORG       $20
START:    LDSP      #BOS       ;Initiering av stackpekare
          CLR       EDGECT     ;Nollställ flankräknare
          CLR       UTPORT     ;Antal negativa flanker är 0
LOOP:     LDA       INNUMB     ;Läs inport för bitnummer (Avkänn bit nr x)
          JSR       NEDGE      ;Vänta på negativ flank på inport INEDGE, bit x
          INC       EDGECT     ;Öka flankräknare
          LDA       EDGECT     ;Hämta antal flanker
          STA       UTPORT     ;Mata ut antal negativa flanker på UTPORT
          BRA       LOOP       ;Upprepa
;*******************************************************************************
;     Subrutin som väntar på negativ flank
;     Om positiv flank skall registreras skall
;     vänteslooparna WAIT0 och WAIT1 kastas om.

NEDGE:    PSHA                 ;Spara på stack
          PSHX

          LDX       #BMASK
          LDA       A,X        ;Hämta bitmask

WAIT0:    BITA      INEDGE     ;Läs givare och maska fram bit med nr enl INNUMB
          BEQ       WAIT0

WAIT1:    BITA      INEDGE     ;Läs givare och maska fram bit med nr enl INNUMB
          BNE       WAIT1

          PULX                 ;Återställ reg
          PULA
          RTS
;*******************************************************************************
EDGECT:   RMB       1          ;Flankräknare

BMASK:    FCB       1,2,4,8,16,32,64,128      ;Mask för bit 0-7

          ORG       $FF
          FCB       START
```

8.28

```
;     Subrutin PCNT

PCNT:       PSHX                ;Spara register

            CLR         HITCNT  ;Nollställ träffräknare

LOOP:       LDA         ,X+     ;Hämta tabellvärde

            CMPA        #$FF    ;Slutmarkering?
            BEQ         PEXIT   ;Ja

            ANDA        #$F0    ;Maska fram bit 7-4
            CMPA        #$60    ;Testa villkor
            BHS         LOOP    ;Ej träff, testa nästa

            INC         HITCNT  ;Träff, öka räknare
            BRA         LOOP    ;Testa nästa

PEXIT:      LDA         HITCNT  ;Hämta resultat

            PULX                ;Återställ register
            RTS

HITCNT:     RMB         1       ;Träffräknare

            ORG         $FF
            FCB         START
```

8.29

```
;     Subrutin SWAP som byter ordning på nibblar i en dataarea

SWAP:       PSHA                ;Spara register på stack
            PSHX
            STA         SWCOUNT
            BEQ         SWEX

SWLOOP      LDA         0,X     ;Hämta data från dataarea
            STA         TEMP    ;Gör kopia av data

            LSLA                ;16-bitars skift
            ROL         TEMP
            LSLA
            ROL         TEMP
            LSLA
            ROL         TEMP
            LSLA
            ROL         TEMP    ;Nu är TEMP swappad

            LDA         TEMP
            STA         ,X+     ;Skriv tillbaka till dataarea och öka pekare

            DEC         SWCOUNT
            BNE         SWLOOP

SWEX:       PULX                ;Återställ register
            PULA
            RTS

SWCOUNT:    RMB         1       ;Räknare för antal bytes
TEMP:       RMB         1       ;Temp variabel
```

8.30

```
;     Subrutin EXCHG
EXCHG      PSHA                ;Spara register på stack
           PSHX
           PSHY

           TSTA                ;Några dataord att flytta?
           BEQ       BLKEX     ;Nej. Färdigt
           STA       DCOUNT    ;Räknare för antal dataord kvar att flytta

EXLOOP:    LDA       0,X       ;Hämta dataord från DATA1
           STA       TEMP      ;Mellanlagra
           LDA       0,Y       ;Hämta dataord från DATA2
           STA       ,X+       ;Skriv dataord från DATA2 till DATA1, öka pekare2
           LDA       TEMP      ;Hämta dataord från DATA1
           STA       ,Y+       ;Skriv dataord från DATA2 till DATA1, öka pekare2
           DEC       DCOUNT    ;Minska ordräknare
           BNE       EXLOOP    ;Färdigt? Nej

;     Nu är det färdigt
BLKEX:     PULY      ;Återställ register
           PULX
           PULA
           RTS

DCOUNT:    RMB       1         ;Räknare för antal dataord kvar att flytta
TEMP:      RMB       1         ;Plats fär mellanlagring av dataord
```

8.31

```
;     Subrutin
ODDCNT:    PSHX                ;Spara register på stack
           CLR       HITCNT    ;Nollställ Träffräknare

ODDLOOP    LDA       ,X+       ;Hämta data från textsträng. Öka pekare
           BEQ       ODDEX     ;Strängslut
           ANDA      #%10000001 ;Maska dont't care-bitar
           CMPA      #%00000001 ;Testa bitmönster
           BNE       ODDLOOP   ;Ej träff, testa nästa

COUNT:     INC       HITCNT    ;Träff öka räknare
           BRA       ODDLOOP   ;Testa nästa

ODDEX:     LDA       HITCNT
           PULX                ;Återställ register
           RTS

HITCNT:    RMB       1         ;Träffräknare
```

8.32

```
;     Subrutin BLKMAN
BLKMAN:    PSHA                ;Spara register på stack
           PSHX

           LDA       #16       ;Sätt byteräknare
           STA       COUNT

PATLOOP:   LDA       0,X       ;Hämta data från tabell. Öka pekare
           ANDA      #%01111111 ;Maska bort bit 7
           EORA      #%00100000 ;Invertera bit 5
           STA       ,X+       ;Skriv tillbaka till tabell
           DEC       COUNT     ;Minska byteräknare
           BNE       PATLOOP   ;Nästa ord

PATEX:     PULX                ;Färdigt återställ register
           PULA
           RTS

COUNT:     RMB       1         ;Byteräknare
```

9 Adressavkodning

9.1 a)

Modul		Adressbuss																
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Anm.
RWM	$0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$07FF	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	
ROM	$E000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
I/O	$C5A0	1	1	0	0	0	1	0	1	1	0	1	0	0	0	0	0	
	$C5A7	1	1	0	0	0	1	0	1	1	0	1	0	0	1	1	1	

b)

Modul		Adressbuss																
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Anm.
RWM	$0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$07FF	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	
ROM	$E000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
I/O	$C5A0	1	1	0	0	0	1	0	1	1	0	1	0	0	0	0	0	
	$C5A7	1	1	0	0	0	1	0	1	1	0	1	0	0	1	1	1	

9.2 a)

Modul		Adressbuss																
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Anm.
RWM	$0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	
ROM	$E000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
I/O	$6000	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$60FF	0	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	

b)

Modul		Adressbuss																
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Anm.
RWM	$0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	
ROM	$E000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
I/O	$6000	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$60FF	0	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	

9.3 a) Modulerna tar upp följande adressområden:

Modul		Adressbuss																
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Anm.
ROM	$FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	$E000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
RWM	$BFFF	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Modulen avbildas på totalt åtta intervall
	$B800	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
	.....																	
	$87FF	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	
	$8000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
utport	$7FFF	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	alla udda i intervallet
	$0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
inport	$7FFF	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	alla som slutar på 111
	$0007	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	

b) Vid läsning av inporten på adresserna 0xxx xxxx xxxx x111 så adresseras också utporten eftersom inte R/W-signalen används i dess avkodning. Data som läses från inporten skrivs då också i utporten.

9.4

a)

Modul			Adressbuss															
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Anm.
ROM	$FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	$F000	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
RWM	$DFFF	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
	$C000	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
I/O	$E0D7	1	1	1	0	0	0	0	0	1	1	0	1	0	1	1	1	
	$E0D0	1	1	1	0	0	0	0	0	1	1	0	1	0	0	0	0	

CS-signalerna bildas ur de för respektive modul konstanta adressdelarna, som grindas med *VA*-signalen.

$$\overline{CS_{ROM}} = \overline{A15 \cdot A14 \cdot A13 \cdot A12 \cdot VA}$$

$$\overline{CS_{RWM}} = \overline{A15 \cdot A14 \cdot \overline{A13} \cdot VA}$$

b) För in- och utportarna skapas den gemensamma CS-signalen:

$\overline{CS_{IO}} = \overline{A15 \cdot A14 \cdot A13 \cdot \overline{A12} \cdot \overline{A11} \cdot \overline{A10} \cdot \overline{A9} \cdot \overline{A8} \cdot A7 \cdot A6 \cdot \overline{A5} \cdot A4 \cdot \overline{A3} \cdot VA}$:

CS-signalerna för respektive port blir nu:

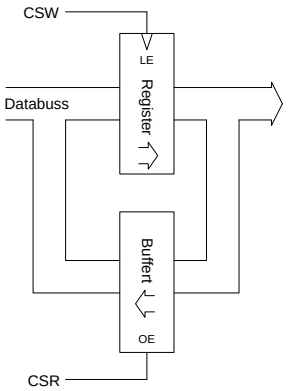
utport: $\overline{CS_{E0D0}} = \overline{\overline{CS_{IO}} \cdot \overline{A2} \cdot \overline{A1} \cdot \overline{A0} \cdot \overline{R/\overline{W}}}$

inport: $\overline{CS_{E0D1}} = \overline{\overline{CS_{IO}} \cdot \overline{A2} \cdot \overline{A1} \cdot A0 \cdot R/\overline{W}}$

9.5

$$CSW = A15 \cdot A14 \cdot A13 \cdot \overline{A12} \cdot \overline{A11} \cdot \overline{A10} \cdot \overline{A9} \cdot \overline{A8} \cdot A7 \cdot A6 \cdot \overline{A5} \cdot A4 \cdot \overline{A3} \cdot \overline{A2} \cdot \overline{A1} \cdot A0 \cdot \overline{R/\overline{W}}$$

Placera en ”inport” i form av en buffert, på samma adress, men med en CSR (ChipSelectRead)



CSR och CSW är aktivt höga:

		Adressbuss															
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
CS	$FFFF	1	1	1	0	0	0	0	0	1	1	0	1	0	0	0	1
		E				0				D				1			

$$CSR = A15 \cdot A14 \cdot A13 \cdot \overline{A12} \cdot \overline{A11} \cdot \overline{A10} \cdot \overline{A9} \cdot \overline{A8} \cdot A7 \cdot A6 \cdot \overline{A5} \cdot A4 \cdot \overline{A3} \cdot \overline{A2} \cdot \overline{A1} \cdot A0 \cdot R/\overline{W}$$

9.6

a) ROM (*Read Only Memory*) minne som enbart är läsbart. RWM (*Read Write Memory*) minne som är både läsbart och skrivbart.

b) Man ser att varje minneskapsel har 13 adressledningar, A0 - A12, som insignaler. Det innebär att varje kapsel innehåller 2^{13} ord = 8 kbyte. Man ser också att en av kapslarna, nr 1) är av ROM-typ, ty den använder ej R/W-signalen. Datorn har således 8 kbyte ROM och 24 kbyte RWM.

c) Minneskapslarna upptar följande adressområden:

Modul		Adressbuss																
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Anm.
1	$FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	$E000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	$BFFF	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Modulen avbildas på två intervall
	$A000	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$2FFF	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	
	$2000	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	$DFFF	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	Modulen avbildas på två intervall
	$C000	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$5FFF	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
	$4000	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	$9FFF	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	Modulen avbildas på två intervall
	$8000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$1FFF	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
	$0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

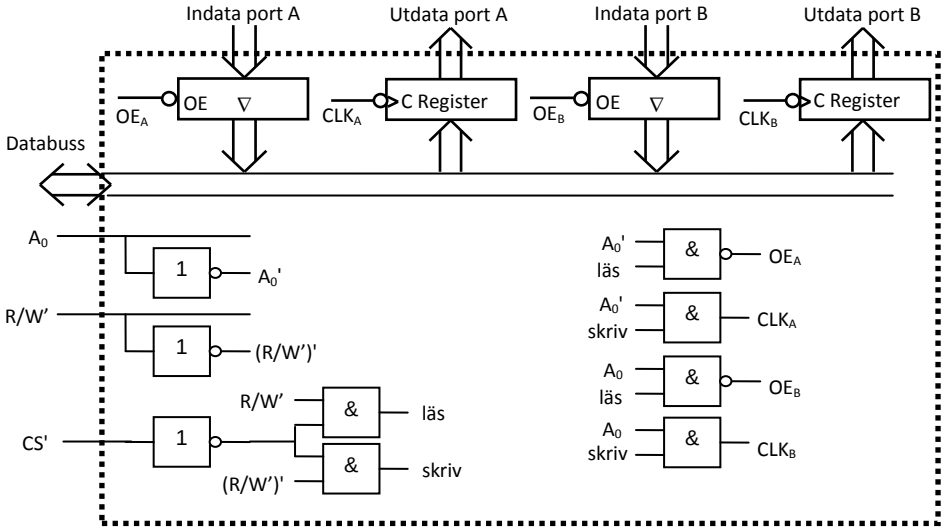
d) VA = 1 (*Valid Memory Address*) anger att adressbussens värde är stabilt och därför används denna signal för att grinda adresserna till minnet. Då kommer de adresser som finns ut på bussen under tiden som adressledningar håller på att ändras aldrig att nå primärminnet.

9.7 a)

Modul			Adressbuss															
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Anm.
I/O	$B801	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	1	
	$B800	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	

$$\overline{CS_O} = \overline{A15 \cdot \overline{A14} \cdot A13 \cdot A12 \cdot A11 \cdot \overline{A10} \cdot \overline{A9} \cdot \overline{A8} \cdot \overline{A7} \cdot \overline{A6} \cdot \overline{A5} \cdot \overline{A4} \cdot \overline{A3} \cdot \overline{A2} \cdot \overline{A1} \cdot VA}$$

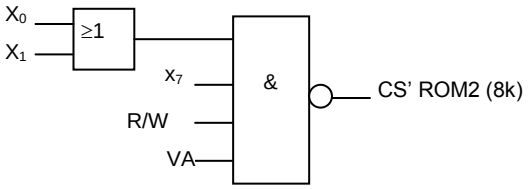
b) I/O-modulens interna struktur bör utformas på följande sätt:



9.8 a) Minnesmodulerna i adressrum M tar upp följande adressområden:

Modul			Adressbuss															
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Anm.
ROM	$FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	$E000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
RWM	$BFFF	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
	$8000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
I/O	$0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	
	$0000	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	

b)



9.9

Minnesmodulerna och I/O-arean i adressrummet tar upp följande adressområden:

Modul			Adressbuss															
		A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Anm.
RWM1	$C000	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$DFFF	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
RWM2	$E000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
	$E07F	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	
I/O	$E080	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	
	$E17F	1	1	1	0	0	0	0	1	0	1	1	1	1	1	1	1	
IE	$E0E0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	
	$E0E1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	1	
ROM	$E200	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	
	$FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	

a)

$$\overline{CS_{RWM1}} = \overline{A15 \cdot A14 \cdot \overline{A13} \cdot VA}$$

$$\overline{CS_{RWM2}} = \overline{A15 \cdot A14 \cdot A13 \cdot \overline{A12} \cdot \overline{A11} \cdot \overline{A10} \cdot \overline{A9} \cdot \overline{A8} \cdot \overline{A7} \cdot VA}$$

b) Steg 1:

$$\overline{CS_{IO}} = \overline{A15 \cdot A14 \cdot A13 \cdot \overline{A12} \cdot \overline{A11} \cdot \overline{A10} \cdot \overline{A9} \cdot (A8 \oplus A7) \cdot VA}$$

Steg 2:

$$\overline{CS_{IE}} = \overline{CS_{IO} \cdot A6 \cdot A5 \cdot \overline{A4} \cdot \overline{A3} \cdot \overline{A2} \cdot \overline{A1}}$$