Grundläggande datorteknik

- exempelsamling

Institutionen för Data och Informationsteknik
Chalmers tekniska högskola
Göteborg HT-2013

2 Talsystem och binära koder

2.1 Utför följande talomvandlingar:

a) $(8)_{10} = (?)_{2}$
b) $(8)_{10} = (?)_{16}$
c) $(8)_{10} = (?)_{NBCD}$
d) $(8)_{10} = (?)_{EXCESS\text{-}3}$
e) $(8)_{10} = (?)_{GRAY(4)}$

2.2 Utför följande talomvandlingar:

a) $(93)_{10} = (?)_{2}$
b) $(93)_{10} = (?)_{NBCD}$
c) $(93)_{10} = (?)_{16}$
d) $(93)_{10} = (?)_{EXCESS\text{-}3}$

2.3 Vilket, av följande alternativ är talet $(54{,}72)_{10}$ på naturlig binär form avrundat till 7 bråksiffror .

a) $(110100.1011100)_{2}$
b) $(110110.1011110)_{2}$
c) $(110110.1011101)_{2}$
d) $(110110.1011100)_{2}$
e) $(110010.\ 1011100)_{2}$

2.4 Ange ASCII-tecknet för följande tal:

a) $(3F)_{16}$
b) $(49)_{10}$
c) $(101011)_{2}$

2.5 Skriv följande ASCII-kodade textsträngar som en sekvens av hexadecimala kodord.

a) CHALMERS
b) tekniska
c) HOGSKOLA

2.6 Skriv följande tal som NBCD-kod

a) $(528{,}31)_{10}$
b) $(407{,}96)_{10}$

2.7 Antag 7 bitars binära tal (naturlig binärkod), där den mest signifikanta positionen är en paritetsbit. Ange de binära talen med jämn paritet (bitmönstren) för följande tal.

a) $(1100)_{2}$
b) $(32)_{10}$
c) $(16)_{16}$

2.8 Antag 7 bitars binära tal (naturlig binärkod), där den mest signifikanta positionen är en paritetsbit. Ange de binära talen med udda paritet (bitmönstren) för följande tal.

a) $(1100)_2$
b) $(32)_{10}$
c) $(16)_{16}$

2.9 Omvandla följande tal (naturlig binär kod) till decimal form:

a) $(101011)_2$
b) $(1100111)_2$
c) $(111100)_2$
d) $(101.011)_2$
e) $(.111)_2$
f) $(1000.01)_2$

2.10 Omvandla följande hexadecimala tal till decimal form:

a) $(AB)_{16}$
b) $(11)_{16}$
c) $(80)_{16}$
d) $(C.4)_{16}$
e) $(0.68)_{16}$
f) $(B3.D4)_{16}$

2.11 Omvandla följande tal till naturlig binär kod. Bråken avkortas till 6 st. bråksiffror.

a) $(45)_{10}$
b) $(33)_{10}$
c) $(87{,}65)_{10}$
d) $(122{,}18)_{10}$
e) $(45)_{16}$
f) $(33)_{16}$
g) $(4.8)_{16}$
h) $(10.6)_{16}$

2.12 Omvandla följande tal till hexadecimal form. Bråken avkortas till 2 st. bråksiffror.

a) $(255)_{10}$
b) $(330)_{10}$
c) $(19{,}37)_{10}$
d) $(132{,}43)_{10}$
e) $(1111110)_2$
f) $(100100001010)_2$
g) $(1111.1111)_2$
h) $(1010.0011)_2$

2.13 En axels vridningsvinkel [0-359°] ska kodas med Gray-kod. Hur många bitar måste kodorden minst ha för att ge upplösningen:

a) 1°
b) 2°

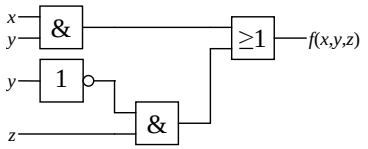
3 Switchnätalgebra

3.1 Åskådliggör följande booleska uttryck med hjälp av grindsymboler (NOT, AND, OR):

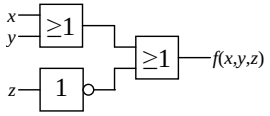
a) $f(x, y, z) = xy + \bar{x}z$
b) $f(x, y, z, e) = e(x + y + z)$
c) $f(x, y) = \bar{x}\,\bar{y} + xy$

3.2 Ange det booleska uttrycket för följande logiknät:

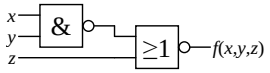
a)



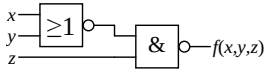
b)



c)

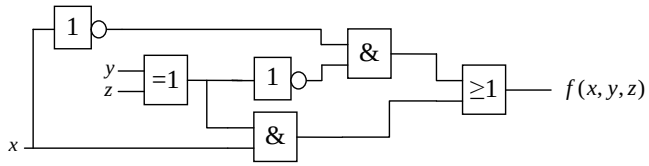


d)



3.3 Vilket (eller vilka) av följande booleska uttryck anger korrekt utsignalen *f* för nedanstående logiknät?

a) $f(x, y, z) = \bar{x}(\bar{y}\,\bar{z} + yz) + x(\bar{y}z + y\bar{z})$
b) $f(x, y, z) = x(\overline{y \oplus z}) + \bar{x}(y \oplus z)$
c) $f(x, y, z) = \overline{x \oplus y \oplus z}$
d) $f(x, y, z) = x \oplus y \oplus z$



3.4 Antag att $x\bar{y} + \bar{x}y = z$ och visa att $x\bar{z} + \bar{x}z = y$

3.5 Avgör för vart och ett av följande fall om uttrycket är sant, dvs. om VL = HL:

a) $(x+y)(\bar{x}+z) = xz + \bar{x}y + yz$

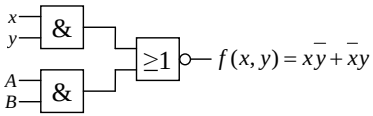
b) $(x+\bar{y})(\bar{x}+z) = \bar{x}z + x\bar{y}z + \bar{x}y\bar{z}$

c) $(x+\bar{x}+y) = 1$

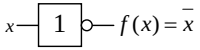
d) $(x\bar{x})(x+\bar{x}+y+\bar{y}) = 0$

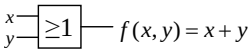
e) $\bar{x}yz + x\bar{y}z + \bar{x}y\bar{z} = (x+\bar{x})(\bar{x}+y)(x+\bar{z})$

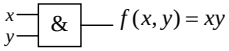
3.6 Bestäm *A* och *B* i följande logikkoppling:



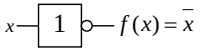
3.7 Visa hur följande logikgrindar kan ersättas med enbart 2-ingångars NAND-grindar.

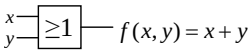
a) 

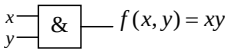
b) 

c) 

3.8 Visa hur följande logikgrindar kan ersättas med enbart 2-ingångars NOR-grindar.

a) 

b) 

c) 

3.9 Realisera följande booleska funktioner med hjälp av 2-ingångars NAND-grindar.

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

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

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

3.10 Realisera följande booleska funktioner med hjälp av 2-ingångars NOR-grindar.

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

b) $f(x,y,z) = x + yz$

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

3.11 Skriv följande funktioner på disjunktiv normal form. Bestäm mintermerna med hjälp av funktionstabell och binär evaluering.

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

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

c) $f(x,y,z,w) = xw + yz$

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

3.12 Vilket av följande alternativ utgör den disjunktiva normalformen av den booleska funktionen:

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

a) $xz + \bar{x}y + yz$

b) $\bar{x}\,\bar{y}z + x\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz$

c) $\bar{x}yz + x\bar{y}z + \bar{x}y\bar{z} + \bar{x}y\bar{z}$

d) $xyz + x\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz$

e) $x\bar{y}z + \bar{x}y\bar{z} + \bar{x}yz$

3.13 Vilket av följande alternativ utgör den disjunktiva normalformen av den booleska funktionen:

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

a) $\bar{x}\bar{y}zw + x\bar{y}zw + xyzw$

b) $\bar{x}\bar{y}zw + x\bar{y}z\bar{w} + xyzw$

c) $\bar{x}\bar{y}\bar{z}w + x\bar{y}zw + xyzw$

d) $\bar{x}yzw + x\bar{y}zw + xyzw$

e) $x\bar{y}zw + x\bar{y}zw + xyzw$

3.14 Skriv följande funktioner på konjunktiv normal form. Bestäm maxtermerna med hjälp av funktionstabell och binär evaluering.

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

b) $f(x,y,z) = xy + xz$

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

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

3.15 Vilket av följande alternativ utgör den konjunktiva normalformen av den booleska funktionen:

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

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

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

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

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

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

3.16 Tag fram en minimal disjunktiv form till var och en av funktionerna beskrivna av följande Karnaughdiagram:

a) f(yz)

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

b) f(yz)

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

c) f(zw)

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

d) f(zw)

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

e) f(zw)

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) f(zw)

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

g) f(zw)

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

h) f(zw)

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

i) f(zw)

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

j) f(zw)

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

3.17 Tag fram en minimal konjunktiv form till var och en av funktionerna beskrivna av följande Karnaughdiagram:

a) f(yz)

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

b) f(yz)

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

c) f(zw)

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

d) f(zw)

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

e) f(zw)

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) f(zw)

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

g) f(zw)

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

h) f(zw)

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

i) f(zw)

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

j) f(zw)

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

4 Aritmetik

4.1 Ange decimala motsvarigheten till följande 8-bitars tal givna på *binärform*.

a) `01010101`
b) `00111100`
c) `10101010`
d) `11000011`

4.2 Ange decimala motsvarigheten till följande 8-bitars tal givna på *tvåkomplementsform*.

a) `01010101`
b) `00111100`
c) `10101010`
d) `11000011`

4.3 Ange följande decimala tal som 8-bitars binära tal på *tvåkomplementsform*. (Ledning, bestäm först binär form av talet utan minustecken, tvåkomplementera därefter)

a) `-10`
b) `-38`
c) `-42`
d) `-56`

4.4 Utför (visa med papper och penna) följande additioner (R=X+Y) av 8-bitars tal givna på *binärform*. Ange X,Y och R på decimal form, ange dessutom hur flaggorna C och Z påverkas av operationerna.

a) `X = 00100100  Y= 01001010`
b) `X = 10111100  Y= 01000100`
c) `X = 10000001  Y= 10000001`
d) `X = 01001010  Y= 00110101`
e) `X = 01001010  Y= 01001010`

4.5 Utför (visa med papper och penna) följande additioner (R=X+Y) av 8-bitars tal givna på *tvåkomplementsform*. Ange X,Y och R på decimal form, ange dessutom hur flaggorna N, V och Z påverkas av operationerna.

a) `X = 00100100  Y = 01001010`
b) `X = 10111100  Y = 01000100`
c) `X = 10000001  Y = 10000001`
d) `X = 01001010  Y = 00110101`
e) `X = 01001010  Y = 01001010`

4.6 Utför (visa med papper och penna) följande subtraktioner (R=X-Y) av 8-bitars tal givna på *binärform*. Ange X,Y och R på decimal form, ange dessutom hur flaggorna C och Z påverkas av operationerna. Flaggan C förutsätts vid denna operation representera en lånesiffra (”borrow”) till den mest signifikanta positionen.

a) `X = 01111110  Y = 01010000`
b) `X = 01010000  Y = 01111110`
c) `X = 11011110  Y = 00100010`
d) `X = 00100010  Y = 11011110`

4.7 Utför (visa med papper och penna) följande subtraktioner (R=X-Y) av 8-bitars tal givna på *binärform*. Subtraktionerna ska utföras som addition av 2-komplement, dvs. R = X + (2~Y). Ange X,Y och R på decimal form, ange dessutom hur flaggorna C och Z påverkas av operationerna. Eftersom subtraktionen utförs som addition av 2-komplement förutsätts flaggan C vid denna operation representera *inversen* av en lånesiffra (”borrow”) till den mest signifikanta positionen.

a) `X = 01111110  Y = 01010000`
b) `X = 01010000  Y = 01111110`
c) `X = 11011110  Y = 00100010`
d) `X = 00100010  Y = 11011110`

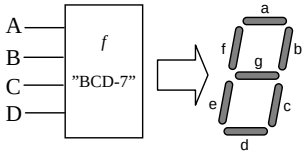
4.8 Utför (visa med papper och penna) följande subtraktioner (R=X-Y) av 8-bitars tal givna på *tvåkomplementsform*. Subtraktionerna ska utföras som addition av 2-komplement, dvs. R = X + (2~Y). Ange X,Y och R på decimal form, ange dessutom hur flaggorna N, V och Z påverkas av operationerna.

a) `X = 01111110  Y = 01010000`
b) `X = 01010000  Y = 01111110`
c) `X = 11011110  Y = 00100010`
d) `X = 00100010  Y = 11011110`
e) `X = 01000000  Y = 10111111`

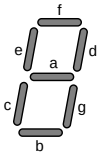
5 *Kombinatoriska nät*

5.1 Konstruera ett minimalt kombinatoriskt nät ”BCD-7”, som konverterar ett binärt kodat decimaltal till så kallad ”Sju-segment kod” enligt följande tabell:

decimal siffra	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	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	-	-	-	-	-	-	-

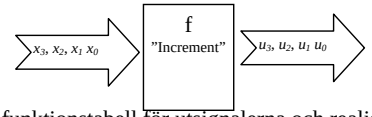


5.2 En sju-segments indikator har segmenten fördelade enligt följande figur:



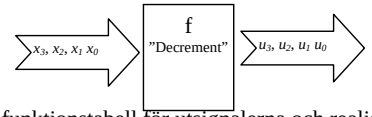
a) Ställ upp en funktionstabell för utsignalerna a-g.
b) Konstruera ett minimalt kombinatoriskt nät för översättning från de decimala siffrorna 0-9, från BCD-form till indikatorn.

5.3 Konstruera ett kombinatoriskt nät ”Increment” som adderar 1 till ett 4-bitars tal.
Nätet består av 4 insignaler x_3, x_2, x_1 och x_0 samt 4 utsignaler u_3, u_2, u_1 och u_0.



Ställ upp funktionstabell för utsignalerna och realisera ett minimalt kombinatoriskt nät. Använd AND/OR-logik och förutsätt att även insignalernas inverser finns tillgängliga.

5.4 Ett kombinatoriskt nät ”Decrement”, som subtraherar 1 från ett 4-bitars tal (tvåkomplements-form) ska konstrueras. Nätet består av 4 insignaler x_3, x_2, x_1 och x_0 samt 4 utsignaler u_3, u_2, u_1 och u_0.

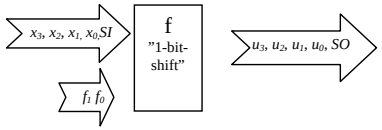


Ställ upp funktionstabell för utsignalerna och realisera ett minimalt kombinatoriskt nät. Använd AND/OR-logik och förutsätt att även insignalernas inverser finns tillgängliga.

5.5 Ett kombinatoriskt nät för olika typer av skiftoperationer av ett 8–bitars tal ska konstrueras. Nätets funktion beror av bitarna f_1 och f_0 enligt följande:

f_1	f_0	funktion
0	0	Ingen operation
0	1	Skift vänster
1	0	Skift höger
1	1	Aritmetiskt skift höger

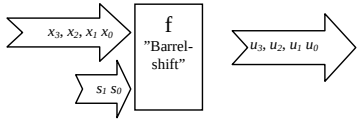
Insignaler: $x_7,x_6,x_5,x_4,x_3,x_2,x_1,x_0$ 8-bitars ord
SI bit som skiftas in i vakant position
Utsignaler: $u_7,u_6,u_5,u_4,u_3,u_2,u_1,u_0$ 8-bitars skiftat ord
SO bit som skiftats ut



a) Ställ upp en funktionstabell där alla utsignaler uttrycks som insignaler eller konstanter.
b) Realisera det kombinatoriska nätet med hjälp av ”1 av 4 väljare”.

5.6 Ett logiknät som skiftar bitar flera positioner under en klockcykel kallas för ”*Barrel-shifter*”. Konstruera ett sådant kombinatoriskt nät för högerskift. Du har tillgång till 4 st. ”1 av 4 väljare”.

Insignaler: x_3,x_2,x_1,x_0 är det ord som skall skiftas.
s_1, s_0 selektor, anger antal skift enligt:
00 →1 skift
01 →2 skift
10 →3 skift
11 →4 skift
Utsignaler: u_3,u_2,u_1,u_0 är resultatet, dvs. det skiftade ordet



a) Konstruera ett kombinatoriskt nät som utför högerskift av ett 4-bitars ord. Mest signifikanta inskiftade bitar är alltid 0. Utskiftade bitar ignoreras.
b) Konstruera ett kombinatoriskt nät som utför höger rotation av ett 4-bitars ord. Den minst signifikanta biten skiftas här till den mest signifikanta positionen.
c) Konstruera ett kombinatoriskt nät som utför aritmetiskt högerskift av ett 4-bitars ord. Mest signifikanta inskiftade bitar är alltid 0. Utskiftade bitar ignoreras.

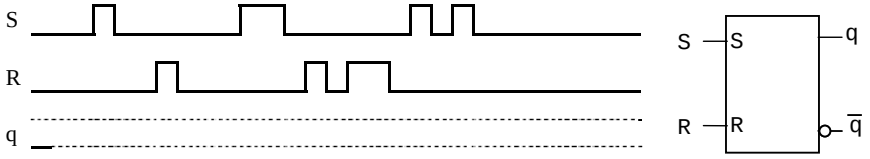
5.7 Funktionen $f(x,y,z) = \bar{y}\bar{z} + xy$. ska implementeras med hjälp av en 1 av 2 väljare. Detta kan göras på tre olika sätt. Visa den lösning som använder minimalt antal grindar utöver väljaren.

5.8 **Härled uttryck för ett kombinatoriskt nät som realiserar “inkrement”-funktion för ett 8-bitars register. Varje utsignal ska uttryckas explicit.

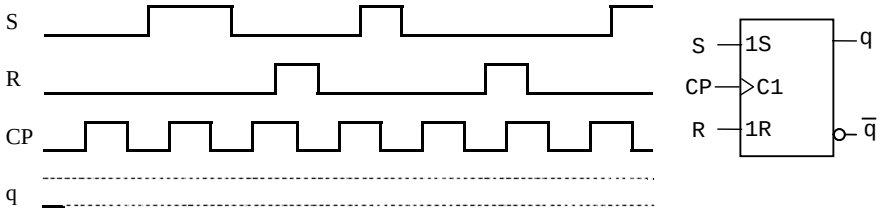
5.9 **Härled uttryck för ett kombinatoriskt nät som realiserar “dekrement”-funktion för ett 8-bitars register. Varje utsignal ska uttryckas explicit.

6 Sekvensnät

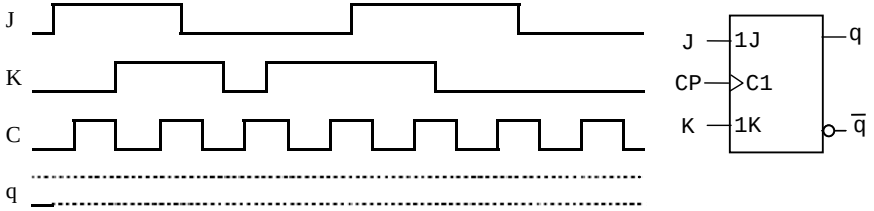
6.1 I figuren visas en SR-latch jämte tidsdiagram för S-, och R- signalerna. Komplettera tidsdiagrammet med q, då q =0 från början.



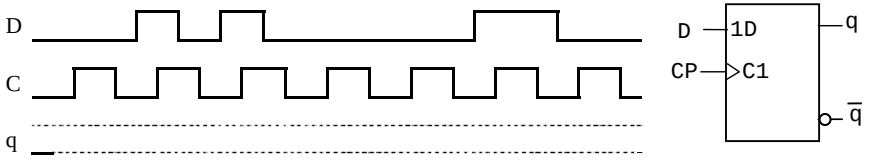
6.2 I figuren visas en SR-vippa jämte tidsdiagram för S-, R- och CP-signalerna. Komplettera tidsdiagrammet med q, då q =0 från början.



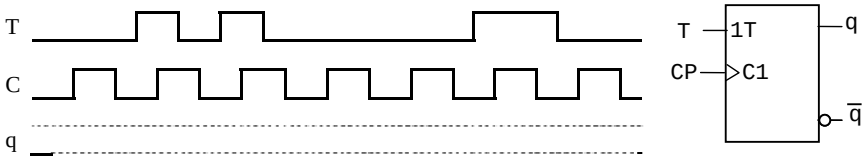
6.3 I figuren visas en flanktriggad JK-vippa jämte tidsdiagram för J-, K- och CP-signalerna. Komplettera tidsdiagrammet med q, då q =0 från början.



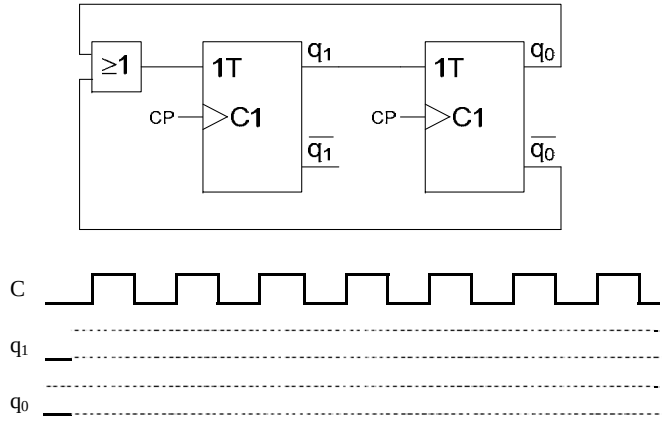
6.4 I figuren visas en D-vippa jämte tidsdiagram för D- och CP-signalerna. Komplettera tidsdiagrammet med q, då q =0 från början.



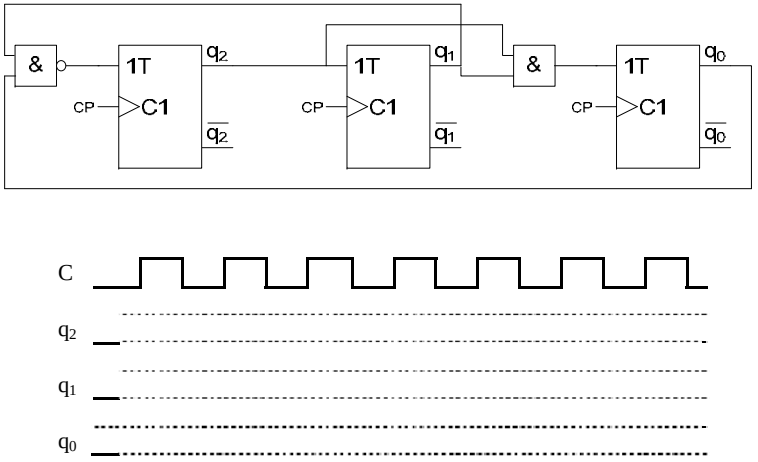
6.5 I figuren visas en T-vippa jämte tidsdiagram för T- och CP-signalerna. Komplettera tidsdiagrammet med q, då q =0 från början.



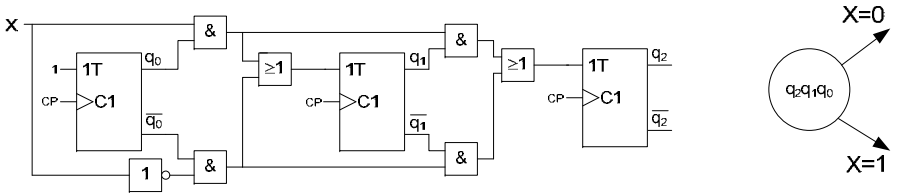
6.6 Ställ upp funktionstabell, rita tillståndsgraf och pulsdiagram för den autonoma räknaren. Pulsdiagrammet skall utgå från tillståndet Q = $(q_1q_0)_2$ = 0



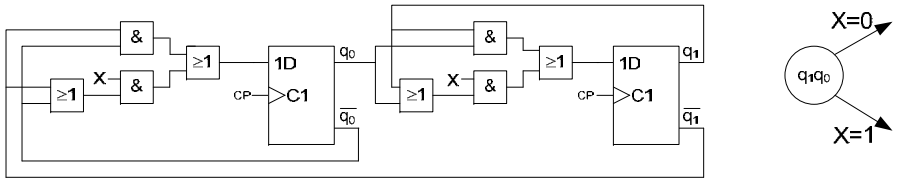
6.7 Ställ upp funktionstabell, rita tillståndsgraf och pulsdiagram för den autonoma räknaren nedan. Pulsdiagrammet skall utgå från tillståndet Q = $(q_2q_1q_0)_2$ = 0



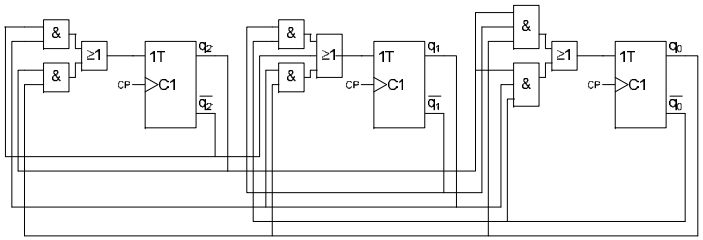
6.8 Ställ upp funktionstabell och rita tillståndsgrafen för följande räknare med räknevillkoret x.



6.9 Ställ upp funktionstabell och rita tillståndsgrafen för följande räknare med räknevillkoret x.

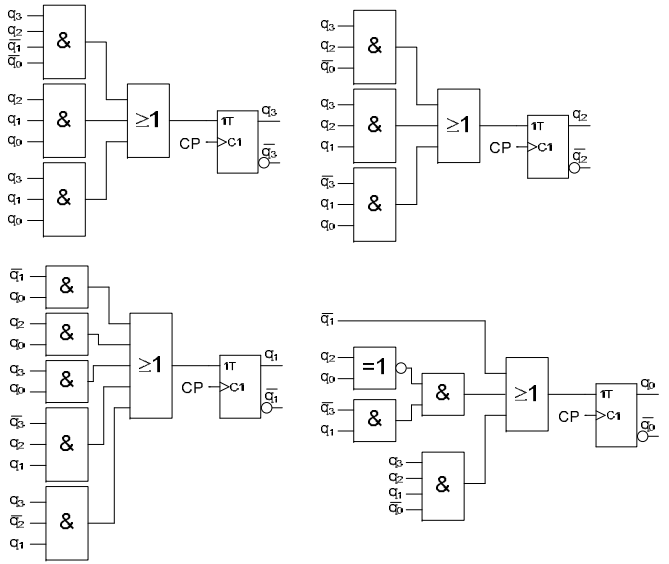


6.10 Analysera räknaren nedan. Utgå från initialtillståndet 0, vilken räknesekvens motsvarar räknarens tillståndsgraf? Tillstånden kodas som binärt tal: $q_2\ q_1\ q_0$.

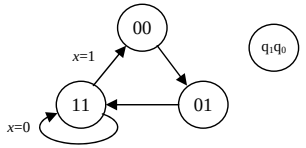


a) 0,1,2,3,4,5,6,7,0...
b) 0,2,4,6,7,1,3,5,0...
c) 0,2,4,7,6,1,3,5,0...

6.11 Analysera räknaren nedan. Visa dess fullständiga tillståndsgraf. Tillstånden kodas som binärt tal: $q_2\ q_1\ q_0$.

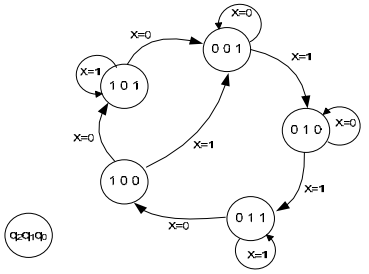


6.12 En 2-bitars tillståndsmaskin illustreras av följande tillståndsgraf:



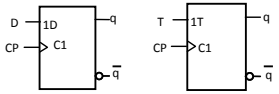
Konstruera en synkron räknare som realiserar tillståndsgrafen. Använd T-vippor, AND-, OR- och INVERTERAR-grindar.

6.13 Konstruera en räknare med styrsignal x, som realiserar följande tillståndsgraf. Använd D-vippor, och grindarna AND, OR och INVERTERARE. Förutsätt att räknaren alltid startar i tillstånd 001.



6.14 Konstruera en autonom synkron modulo-6 räknare med räknesekvens:
$q_2\,q_1\,q_0$ = 000, 001, 101, 111, 011, 010, 000 och vipptyp enligt nedan.
Dessutom får AND-, OR- och INVERTERAR-grindar användas.

a) Vipptyp: D.
b) Vipptyp: T.



6.15 Konstruera en synkron räknare med en styrsignal *x*, som bestämmer räknarens funktion på följande sätt

x = 0 Räknaren räknar enligt sekvensen q_1q_0 = 00, 01, 11, 10, 00
x = 1 Räknaren räknar enligt sekvensen q_1q_0 = 00, 10, 11, 01, 00

Av de två sekvenserna ovan framgår att räknaren är reversibel dvs. den kan räkna en bestämd sekvens framåt eller bakåt. Använd T-vippor och valfria grindar.

6.16 Konstruera en synkron, reversibel räknare med en styrsignal *x*, vilken bestämmer räknarens funktion enligt:

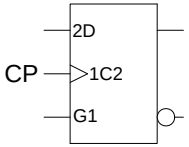
x = 0: Sekvensen 000,001,011,010,110,111,101,100,000 (Gray-kod)
x = 1: Sekvensen 000,100,101,111,110,010,011,001,000 (Gray-kod baklänges)
T-vippor och valfria grindar får användas.

6.17 Konstruera en synkron räknare med styrsignaler *x* och *y* som bestämmer räknarens funktion på följande sätt:

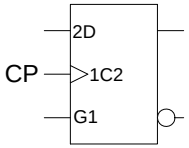
y = 1: Räknaren försätts i tillstånd 000.
y = 0:
x = 0: Räknaren räknar enligt sekvensen $q_2\,q_1q_0$ = 000,001,010,011,100,101,110,111,000...
x = 1: Räknaren försätts i tillstånd 011.

D-vippor och valfria grindar får användas. Rita också en tillståndsgraf som beskriver räknaren.

6.18 Konstruera ett 4-bitars synkront skiftregister ($q_3q_2q_1q_0$) med D-vippor enligt figuren till höger och valfria komponenter för den kombinatoriska delen. Kretsen ska ha en styrsignal *e* (*enable*):
e = 0: skiftregister kvarstannar i befintligt tillstånd
e = 1: ett aritmetiskt högerskift utförs.
Utskiftad bit q_0 ignoreras.



6.19 Konstruera ett 4-bitars synkront skiftregister ($q_3q_2q_1q_0$) med D-vippor och valfria kombinatoriska komponenter. Kretsen ska ha två styrsignaler *e* (*enable*) och *d* (*direction*):



e = 0: skiftregister kvarstannar i befintligt tillstånd
e = 1: ett skift utförs.
d = 0: "vänsterskift"
d = 1: "högerskift"
Utskiftade bitar (q3 vid vänsterskift och q0 vid högerskift) placeras i en D-vippa med beteckningen **c**. Inskiftade bitar (q0 vid vänsterskift och q3 vid högerskift) tas från D-vippa **c**.

7 Dataväg ALU och minne

För uppgifter i detta kapitel där ALU'n ingår gäller ALU-funktioner enligt följande tabell.

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

Anm:
D_{1k} = 1-komplement D = D'
$(E+C_{in})_{2k}$ = 2-komplement $E+C_{in}$ = 0-E-C_{in}

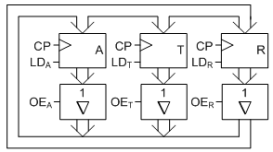
(1) $Z = \overline{u_7} \wedge \overline{u_6} \wedge \overline{u_5} \wedge \overline{u_4} \wedge \overline{u_3} \wedge \overline{u_2} \wedge \overline{u_1} \wedge \overline{u_0}$, dvs. Z=1 då samtliga bitar i register U är 0, Z=0 annars.

(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.

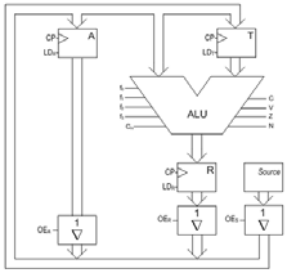
7.1 *Styrsignaler för registeröverföringar*: Använd figurens dataväg och ange styrsignalssekvenser för följande registeröverföringar beskrivna enligt RTN.

	Operation (RTN)	Aktiva styrsignaler
a)	$T \rightarrow A$	
b)	$T \rightarrow A,R$	
c)	$R \rightarrow T$, $A \rightarrow R$ $T \rightarrow A$ $= A \leftrightarrow R$	



7.2 *Enkla unära operationer*: Använd figurens dataväg och ange styrsignalssekvenser för följande operationer.

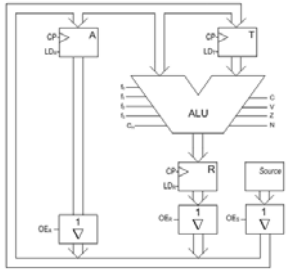
a) Nollställ innehållet i register A.
b) Addera 1 till innehållet i register A.
c) Bitkomplementera innehållet i register A.
d) Negera (tvåkomplementera) innehållet i register A.



Ge svaren i en tabellform enligt följande:

Cykel	Operation (RTN)	Aktiva styrsignaler
1		
2		
…		

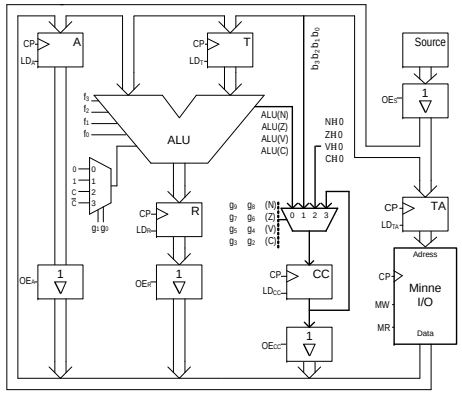
7.3 Använd figurens dataväg och ange en styrsignalssekvens enligt följande RTN-beskrivning:



Cykel	Operation (RTN)	Aktiva styrsignaler
1	$20_{16} \rightarrow A$	
2	$A\text{-}1 \rightarrow R$	
3	$R \rightarrow A$	
4	$F0_{16} \rightarrow T$	
5	$A \oplus T \rightarrow R$	
6	$R \rightarrow A$	

7.4 *Enkla minnesoperationer*: Använd figurens dataväg och ange styrsignalssekvenser för följande operationer.

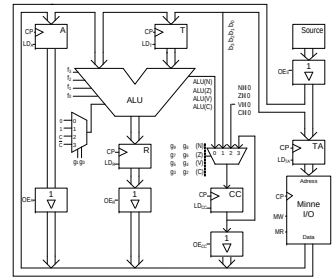
a) Kopiera innehållet på adress $(10)_{16}$ till register A.
b) Kopiera innehållet indirekt från adress $(10)_{16}$ till register A.
c) Utgå från att register A innehåller en adress och kopiera innehållet från den adressen till register A.
d) Kopiera innehållet från register A till adress $(10)_{16}$.



7.5 *Register/minne addition.* Använd figurens dataväg och ange styrsignalssekvenser för operationen:

$A+M(14_{16}) \rightarrow A$

Ge svaret i tabellform enligt följande:

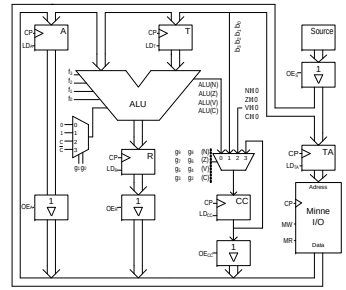


Cykel	Operation (RTN)	Aktiva styrsignaler
1		
2		
3		
4		
5		
6		
…		

7.6 *Minnesaddition.* Använd figurens dataväg och ange styrsignalssekvenser för operationen:

$M(12_{16})+M(13_{16}) \rightarrow M(14_{16})$

Ge svaret i tabellform enligt följande:

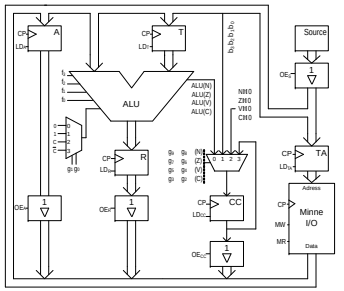


Cykel	Operation (RTN)	Aktiva styrsignaler
1		
2		
3		
4		
5		
6		
…		

7.7 *8/16 bitars addition.* Använd figurens dataväg och ange styrsignalssekvenser för operationen:

$A+M(12_{16}):M(13_{16}) \rightarrow M(20_{16}) : M(21_{16})$

Ge svaret i tabellform enligt följande:



Cykel	Operation (RTN)	Aktiva styrsignaler
1		
2		
3		
4		
5		
6		
…		

8 Styrenheten

8.1 I följande tabell visas RTN-beskrivningen för EXECUTE-sekvensen för en av FLIS-processorns instruktioner.

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	M(SP)→Y;	
Q_5	SP+1→SP;	

a) Ange för varje tillstånd de styrsignaler som är aktiva.
b) Vad är instruktionens namn (med assemblerspråk)?

8.2 I följande tabell visas RTN-beskrivningen för EXECUTE-sekvensen för en av FLIS-processorns instruktioner.

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	SP-1→SP;	
Q_5	Y→M(SP);	

a) Ange för varje tillstånd de styrsignaler som är aktiva.
b) Vad är instruktionens namn (med assemblerspråk)?

8.3 I följande tabell visas RTN-beskrivningen för EXECUTE-sekvensen för en av FLIS-processorns instruktioner.

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	n→T; PC+1→PC	
Q_5	Y→M(T+X);	

a) Ange för varje tillstånd de styrsignaler som är aktiva.
b) Vad är instruktionens namn (med assemblerspråk)?

8.4 I följande tabell visas RTN-beskrivningen för EXECUTE-sekvensen för en av FLIS-processorns instruktioner.

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	n→T;	
Q_5	M(T+X)→R;	
Q_6	R→PC;	

a) Ange för varje tillstånd de styrsignaler som är aktiva.
b) Vad är instruktionens namn (med assemblerspråk)?

8.5 I följande tabell visas RTN-beskrivningen för EXECUTE-sekvensen för en av FLIS-processorns instruktioner.

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	A'→R;	
	ALU(N,Z) →CC(n,Z); 0→V;	
Q_5	R→A;	

a) Ange för varje tillstånd de styrsignaler som är aktiva.
b) Vad är instruktionens namn (med assemblerspråk)?

8.6 I följande tabell visas RTN-beskrivningen för EXECUTE-sekvensen för en av FLIS-processorns instruktioner.

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	(d7)A>>1→R;	
Q_5	R→A;	

a) Ange för varje tillstånd de styrsignaler som är aktiva.
b) Vad är instruktionens namn (med assemblerspråk)?

8.7 I följande tabell visas RTN-beskrivningen för EXECUTE-sekvensen för en av FLIS-processorns instruktioner.

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	A→T	
Q_5	(C)(M(A+Y))>>1→R;	
Q_6	R→ M(A+Y);	

a) Ange för varje tillstånd de styrsignaler som är aktiva.
b) Vad är instruktionens namn (med assemblerspråk).

8.8 I följande tabell visas RTN-beskrivningen för EXECUTE-sekvensen för en av FLIS-processorns instruktioner.

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	PC→TA,T;	
Q_5	M(TA)+1→R; PC+1→PC;	
Q_6	if(N=1) R→ PC;	

a) Ange för varje tillstånd de styrsignaler som är aktiva.
b) Vad är instruktionens namn (med assemblerspråk)?

8.9 I följande tabell visas RTN-beskrivningen för EXECUTE-sekvensen för en av FLIS-processorns instruktioner.

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	PC→TA,T;	
Q_5	M(TA)+1→R; PC+1→PC;	
Q_6	if(C+Z=1) R→ PC;	

a) Ange för varje tillstånd de styrsignaler som är aktiva.
b) Vad är instruktionens namn (med assemblerspråk)?

8.10 I instruktionslistan för FLIS-processorn finner vi en maskininstruktion som beskrivs enligt följande:

Instruktion	Adressering				Operation	Flaggor			
COM									
Variant	metod	OP	#	~		N	Z	V	C
`COM Adr`	Absolute	3A	2	4	M(Adr) ' → M(Adr)	Δ	Δ	0	-

Visa hur instruktionens EXECUTE-fas implementeras hos en FLIS-processor med fast kopplad logik. Lösningen ska tydligt visa varje tillstånd med RTN beskrivning och angivande av aktiva styrsignaler.

8.11 I instruktionslistan för FLIS-processorn finner vi en maskininstruktion som beskrivs enligt följande:

Instruktion	Adressering				Operation	Flaggor			
JMP									
Variant	metod	OP	#	~		N	Z	V	C
`JMP A,X`	Indexed	63	1	4	A+X → PC	-	-	-	-

Visa hur instruktionens EXECUTE-fas implementeras hos en FLIS-processor med fast kopplad logik. Lösningen ska tydligt visa varje tillstånd med RTN beskrivning och angivande av aktiva styrsignaler.

8.12 I instruktionslistan för FLIS-processorn finner vi en maskininstruktion som beskrivs enligt följande:

Instruktion	Adressering				Operation	Flaggor			
ROL									
Variant	metod	OP	#	~		N	Z	V	C
`ROL n,SP`	Indexed	4D	2	4	M(n+SP)<<1 → M(n+SP)	Δ	Δ	Δ	Δ

Visa hur instruktionens EXECUTE-fas implementeras hos en FLIS-processor med fast kopplad logik. Lösningen ska tydligt visa varje tillstånd med RTN beskrivning och angivande av aktiva styrsignaler.

8.13 I instruktionslistan för FLIS-processorn finner vi en maskininstruktion som beskrivs enligt följande:

Instruktion	Adressering				Operation	Flaggor			
BLT									
Variant	metod	OP	#	~		N	Z	V	C
`BLT Adr`	Relativ	2F	2	4	If (N⊕V) = 1: PC+Offset → PC	-	-	-	-

Visa hur instruktionens EXECUTE-fas implementeras hos en FLIS-processor med fast kopplad logik. Lösningen ska tydligt visa varje tillstånd med RTN beskrivning och angivande av aktiva styrsignaler.

9 Assemblerprogrammering

Komplettera följande tabeller som anger operander och operatorer, med de resulterande uttryckens värden.

A	B	A+B	A-B	A*B	A/B	A%B
32	17					
32	32					
16	17					

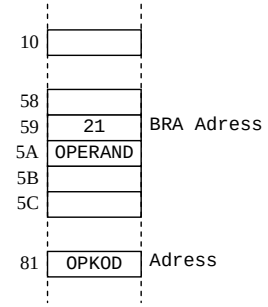
A	B	~A	A&B	A|B	A^B
32	17				
32	32				
16	17				

A	B	A>>B	A>>B
32	17		
32	32		
16	17		

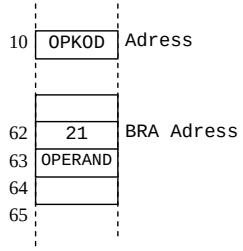
A	B	A==B	A!=B	A<=B	A>=B	A<B	A>B
32	17	0	1	0	1	0	1
32	32	1	0	1	1	0	0
16	17	0	1	1	0	1	0

A	B	!A	A&&B	A||B
32	17			
32	32			
16	17			

9.1 Följande figur illustrerar några fragment minne hos en FLISP-dator. I minnet finns en ”branch”-instruktion samt dess mål indikerade. Bestäm instruktionens operand, dvs. det värde som finns lagrat på adress 5A.



9.2 Följande figur illustrerar några fragment minne hos en FLISP -dator. I minnet finns en ”branch”-instruktion samt dess mål indikerade. Bestäm instruktionens operand, dvs. det värde som finns lagrat på adress 63.



9.3 För vilka värden på U utförs hoppet nedan? Betrakta U som ett tal [0,255].

```
LDA             #$85
CMPA            #U
B(Villkor)      Hopp
```

om den villkorliga instruktionen är

a)	BHI	e)	BGT
b)	BHS	f)	BGE
c)	BLS	g)	BLE
d)	BLO	h)	BLT

9.4 Översätt följande sekvens av assemblerdirektiv för FLIS-assemblatorn till maskinkod (assemblera programmet). Komplettera figuren så att det klart framgår hur maskinkod och assemblerkod hör ihop och vilka minnesadresser maskinkoden är placerad på.

Adress	Innehåll	Assemblerkod		
			ORG	$E0
			FCB	%10000111
			RMB	2
			FCB	$7A
			FCB	´a´
			FCS	”ABCD”

9.5 Översätt följande assemblerprogram för FLISP till maskinkod (assemblera programmet). Komplettera figuren så att det klart framgår hur maskinkod och assemblerkod hör ihop och vilka minnesadresser maskinkoden är placerad på.

Adress	Innehåll	Assemblerkod (disassemblering)		
			ORG	$30
		Start:	LDX	#$0C
		Loop:	LDA	,X+
			STA	$FC
			TSTA	
			BPL	Loop
		Stop:	JMP	Stop

9.6 Översätt följande assemblerprogram för FLISP till maskinkod (assemblera programmet). Komplettera figuren så att det klart framgår hur maskinkod och assemblerkod hör ihop och vilka minnesadresser maskinkoden är placerad på.

Adress	Innehåll	Assemblerkod (disassemblering)		
			ORG	$20
		Length:	EQU	4
		Start:	LDX	#Table
			LDA	Length
			STA	Count
		Loop:	LDA	,X+
			STA	$FC

```
                DEC         Count

                LDA         Count

                BNE         Loop

Stop            Jmp         Stop

Count           RMB         1
Table:          FCB         $10,%10,10,0
```

9.7 Ur en FLISP-dators minne avläser vi följande minnesinnehåll (hexadecimalt):

`90,48,91,94,F5,24,03,EB,21,FA,21,FE`

Utgå i från att ett program är lagrat här med början på adress 20_{16}. Identifiera instruktionssekvensen, dvs. disassemblera minnesinnehållet.

9.8 Ur en FLISP-dators minne avläser vi följande minnesinnehåll (hexadecimalt):

`F1,7A,06,0E,A6,C4,A5,C5,24,02,F0,01,21,FE`

Utgå i från att ett program är lagrat här med början på adress 20_{16}. Identifiera instruktionssekvensen, dvs. disassemblera minnesinnehållet.

9.9 Följande program finns lagrat i en FLISP-dators minne:

```
;    Huvudprogram
20: LDSP  #$F0
22: LDX   #$50
24: PSHX
25: JSR   $40
27: STA   $E0
29: JMP   $10

; Subrutin 1
40: PSHA
41: PSHX
42: PSHC
43: LDX   #$E0
45: LDA   #$03
47: JSR   $60
49: PULC
4A: PULX
4B: PULA
4C: RTS

; Subrutin 2
60: PSHA
```



```
61: PSHC
62: PSHX
63: LDX   #$15
65: INCA
66: PULX
67: PULC
68: PULA
69: RTS
```

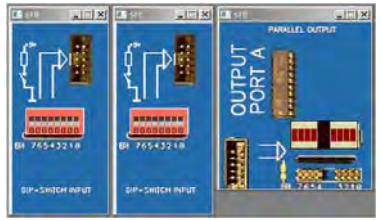
Beskriv stackens utseende, dvs. komplettera följande tabell, då programräknaren PC har följande värden:

PC	22	25	45	60	49	27
SP						
(SP)	-					
(SP+1)	-	-				-
(SP+2)	-	-				-
(SP+3)	-	-				-
(SP+4)	-	-	-		-	-

9.10 En 8-bitars strömbrytare, ”DIP_SWITCH” är ansluten till adress FB_{16} och en displayenhet ”HEXDISPLAY” som visar en byte i form av två hexadecimala siffror är ansluten till adress FC_{16} i en FLISP dator.

Konstruera en subrutin `DipHex` som läser av strömbrytaren och indikerar den minst signifikanta påslagna biten genom att skriva dess position, räknat från höger, till displayenheten. Om exempelvis bitarna 2 och 4 utgör ettställda strömbrytare ska positionen för bit 2, (dvs. 3) skrivas till displayenheten.
Om ingen strömbrytare är ettställd ska siffran 0 skrivas till displayen.
Speciellt gäller att symboler ska definieras och användas för absoluta adresser.

9.11 Två strömbrytare är anslutna till inportar med adresser FB_{16} och FC_{16} i en FLISP-dator. Dessutom är en ljusdiodramp ansluten till en utport med adress FB_{16}.



Konstruera en subrutin `DipSwitchOr` som bildar logisk ELLER av värdena som läses från strömbrytarna.

Subrutinen ska utformas så att avläsningen och indikering görs en gång. Kontinuerlig funktion fås genom att subrutinen, oupphörligt anropas från ett huvudprogram "main". Speciellt gäller att symboler ska definieras och användas för absoluta adresser.

9.12 I simulatorn för FLIS-datorn kan man ansluta strömbrytarmodulen DIPSWITCH till en inport och sifferindikatorn 7-SEGMENT till en utport. Skriv ett program som kontinuerligt läser av två 4-bitars tal från de åtta strömbrytarna på DIPSWITCH via inporten $FB, adderar de två talen och visar summan (4 bitar) som en decimal siffra på sifferindikatorn 7-SEGMENT, ansluten till utport $FC. Från inporten läses två 4-bitars binära tal P och Q samtidigt. P finns på bit 7654 och Q på bit 3210. Summan P+Q skall omvandlas till segmentkod och visas som en siffra 0-9 på sifferindikatorn. Om summan är större än nio skall ett ’E’ (ERROR) visas.

Du har tillgång till en tabell med segmentkoder och följande definitioner:

```
Error     EQU  $5D                ; Segmentkod för E (Error)
SegCode   FCB  xx,yy,zz,etc       ; Tabell med segmentkoder för [0,9]
```

9.13 Skriv en subrutin CNTONE i assemblerspråk för FLIS-processorn som räknar antalet ettor i register A. Vid återhopp skall register A innehålla antalet ettor som fanns i registret vid anropet. Om registret innehåller 5 st ettor vid anropet så skall det alltså innehålla talet %00000101 vid återhoppet. Endast register A och CC får vara förändrade vid återhopp från subrutinen. Hur skall programmet ändras om man istället vill räkna antalet nollor?

9.14 Skriv en subrutin ASCBIN i assemblerspråk för FLIS-processorn som översätter ett 7-bitars ASCII-tecken för en hexadecimal siffra (0-9 eller A-F) till motsvarande binära tal %00000000-%00001111. Vid anrop av subrutinen innehåller register A ASCII-tecknet i bitarna b6-b0 medan b7 är en checkbit med okänt innehåll. Vid återhopp skall det binära talet finnas i register A. Om innehållet i register A vid anropet ej är ASCII-tecknet för en hexadecimal siffra så skall talet $FF finnas i register A vid återhopp. Endast register A och CC får vara förändrade vid återhopp från subrutinen.

9.15 Skriv en subrutin, CCOUNT, i assemblerspråk för FLIS-processorn, som tar reda på hur många gånger ASCII-tecknet för bokstaven 'C' förekommer i en nollterminerad textsträng. Vid anrop av subrutinen skall startadressen till textsträngen finnas i X-registret och vid återhopp skall antalet C-tecken finnas i A-registret. ASCII-tecknen i textsträngen är lagrade med udda paritet med paritetsbiten som bit nummer 7. Den avslutande nollan har ingen paritets-bit. Endast register A och CC får vara förändrade vid återhopp från subrutinen.

9.16 Skriv en subrutin, AaCNT, i assemblerspråk för FLIS-processorn, som tar reda på hur många gånger ASCII-tecknen för bokstäverna A och a förekommer i en nollterminerad textsträng. Vid anrop av subrutinen skall startadressen till textsträngen finnas i X-registret och vid återhopp skall antalet ASCII-tecken finnas i A-registret. ASCII-tecknen i textsträngen är lagrade med jämn paritet med paritetsbiten som bit nummer 7.

Endast register A och CC får vara förändrade vid återhopp från subrutinen.

9.17 Skriv en subrutin SMALLCNT i assemblerspråk för FLIS-processorn som söker igenom en textsträng med 7- bitars ASCII-tecken och räknar alla ASCII-tecken som motsvarar små bokstäver (a-z). Vid anrop av subrutinen skall adressen till textsträngen finnas i X-registret. Textsträngen avslutas med datavärdet 0. Det sökta antalet skall returneras i A-registret. För övrigt får endast CC-registret vara förändrat vid återhopp. Ett ASCII-tecken är lagrat i bit6-bit0 på varje adress i textsträngen. Bit7 i textsträngens dataord har okänt värde.

9.18 I minnet i ett datorsystem med FLIS-processorn finns en nollterminerad sträng med sjubitars ASCII-tecken. Varje ASCII-tecken har kompletterats med en paritetsbit som åttonde bit (bit nr 7). Skriv en subrutin LCOUNT i assemblerspråk för FLIS-processorn som tar reda på hur många av ASCII-tecknen i strängen som motsvarar stora eller små bokstäver A - Z eller a - z. Antalet bokstäver skall finnas i A-registret vid återhopp. Vid anrop av subrutinen skall startadressen till strängen finnas i X-registret. Endast register A och CC får vara förändrade vid återhopp från subrutinen.

9.19 Skriv en subrutin PRINT i assemblerspråk för FLIS-processorn som matar ut ASCII-tecken (7 bitars ASCII lagrade som 8-bitars ord med mest signifikant bit nollställd) till en skrivare enligt följande beskrivning:

- ASCII-tecknen som skall matas ut finns lagrade i minnet med första tecknet på adressen som finns i X-registret och följande tecken på ökande adresser.

- Skrivarens dataingång (8 bitar) är ansluten till utport $FB på FLIS-datorn.
- Skrivarens statussignal READY är ansluten till bit 7 (mest signifikant) på inport $FC. Om signalen READY har värdet 1 är skrivaren beredd att ta emot ett nytt ASCII-tecken.
- Utmatning av ett dataord (ASCII-tecken) till skrivarens dataingång medför automatiskt att skrivaren nollställer signalen READY.
- Utmatning av ASCII-tecken skall fortsätta tills ett dataord med värdet $00 läses från minnet. Dataordet med värdet $00 används som slutmarkering och skall inte matas ut till skrivaren.
- Återhopp från subrutinen skall göras när utmatningen är färdig. Vid återhoppet skall samtliga interna register (utom PC) ha samma värden som vid inhoppet.
- Subrutinen skall fungera även om första dataordet som läses från minnet är $00, dvs återhopp skall då ske direkt.

9.20 En ramp med ljusdioder, enligt figuren till höger, är ansluten till adress FC_{16} i en FLISP-dator.

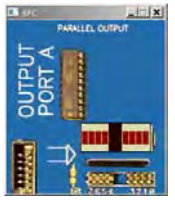
a) Skriv en subrutin `BLINK` som får samtliga dioder att blinka genom att först tända och sedan släcka dom. Kontrollera funktionen genom att stega igenom subrutinen instruktionsvis.

b) Utforma, som en ny subrutin `BLINKDELAY`, en fördröjning så att dioderna blinkar även då programmet exekveras normalt.

c) Beskriv subrutinerna `BLINK` och `BLINKDELAY` i form av flödesplaner.

d) Skapa ett enkelt huvudprogram `main`, som kontinueraligt anropar subrutinen `BLINK`.

9.21 En ramp med ljusdioder, enligt figuren till höger, är ansluten till en utport med adress FC_{16} i en FLISP-dator.

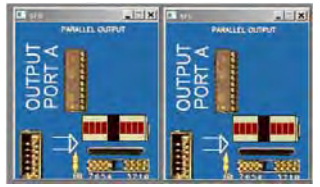


a) Skriv en subrutin `RLJUSH` som får dioderna att bete sig som ett "rinnande ljus" där dioderna tänds upp en och en från vänster till höger. Kontrollera funktionen genom att stega igenom subrutinen instruktionsvis.

b) Använd subrutinen `BLINKDELAY`, så att man tydligt kan se det rinnande ljuset även då programmet exekveras normalt.

c) Beskriv lösningen från b) i form av en flödesplan.

9.22 Två ramper med ljusdioder, enligt figuren till höger, är anslutna till utportarna med adress FB_{16} och FC_{16} i en FLISP-dator.



Konstruera en subrutin `RLJUSH16` som får dioderna att bete sig som ett kontinuerligt "rinnande ljus" där dioderna tänds upp en och en från vänster till höger. Efter det att bit 0 hos diodrampen på adress FB_{16} släckts ska bit 7 hos diodrampen på adress FC_{16} tändas. Då dioden för bit 0 på adress FC_{16} släckts, ska det rinnande ljuset börja om från bit 7 på adress FB_{16}, osv.

Använd en given subrutin `BLINKDELAY`, så att man tydligt kan se det rinnande ljuset även då programmet exekveras normalt. Implementera, dvs. skriv subrutinen i assemblerspråk.

9.23 I simulatorn för FLIS-datorn kan man ansluta strömbrytarmodulen DIPSWITCH till en inport och sifferindikatorn 7-SEGMENT till en utport.

Skriv ett program som läser av ett värde på strömbrytarna, visar värdet som en hexadecimal siffra på sifferindikatorn och sedan utför en fördröjning. Därefter skall sifferindikatorn släckas,

en ny fördröjning utföras och förloppet upprepas från början. Programmet som skall ha startadressen $20 beskrivs nedan. Det finns en färdig subrutin DELAY1S som utför en fördröjning på 1 s.
Programmet skall i tur och ordning:

1. Läsa strömbrytarna på inport $FB.
2. Maska fram bit 3-0 av det inlästa värdet, som motsvarar den önskade fördröjningen i sekunder.
3. Visa detta värde x som tillhör intervallet [0,$F] på sifferindikatorn på utport $FC.
4. Utföra fördröjningen. (Hoppa till DELAY1S x gånger).
5. Släcka sifferindikatorn.
6. Utföra fördröjningen igen. (Hoppa till DELAY1s x gånger).
7. Hoppa tillbaka till 1.

Du har tillgång till en tabell med segmentkoder och följande definitioner:

```
SegCode    FCB  xx,yy,zz,etc    Tabell med segmentkoder för [0,F]
DELAY1S    EQU  wwww       Startadress för Delay-rutinen som utför 1s fördröjning
```

9.24 En FLIS-dator skall användas för att styra ett elektroniskt lås. I låsprogrammet skall ingå en subrutin KEYCMP som jämför en sträng med inmatade ASCII-tecken med en annan sträng med 8-bitars ord, som innehåller "nyckeln" till låset. Bit 7 i varje inmatat ASCII-tecken har okänt värde medan motsvarande bit i "nyckel-strängen" är noll. Bägge strängarna är lika långa, relativt korta och noll-terminerade. Skriv en subrutin i assemblerspråk för FLIS-processorn som jämför de två strängarna och returnerar antal fel i A-registret. Vid anrop av subrutinen skall startadressen till den inmatade strängen finnas i X-registret och startadressen till "nyckelsträngen" i Y-registret. Endast register A och CC får vara förändrade vid återhopp från subrutinen.

9.25 En FLIS-processor skall användas i styrenheten för en maskin. Till inport $FC är ett antal givare och switchar anslutna. En operatör skall styra maskinen via switcharna. Huvudprogrammet för maskinstyrningen skall utformas som en evighetsslinga där inporten läses av en gång i början på varje varv. Huvudprogrammet skall inledas med att stackpekaren först sätts till värdet $FA. Sedan skall inporten läsas av och beroende på switcharnas och givarnas värden skall en av fyra olika färdiga subrutiner anropas om motsvarande villkor i tabellen nedan är uppfyllt. Efter återhopp från subrutinerna eller om inget av villkoren i tabellen är uppfyllt skall ett nytt varv i slingan påbörjas. Rita en flödesplan som beskriver huvud-programmet och skriv huvud-programmet i assemblerspråk för FLIS-processorn. Huvudprogrammets startadress skall vara $20. Subrutinernas startadresser antas vara givna.

Villkor	Inport $FC bitnr								Anropa subrutin
	7	6	5	4	3	2	1	0	
1	?	?	?	1	?	?	?	1	SUB1
2	?	?	0	1	?	?	1	0	SUB2
3	?	?	?	0	?	?	1	?	SUB3
4	0	0	0	0	1	1	0	0	SUB4

9.26 I en styrenhet för en maskin används FLIS-datorn. Ett avsnitt av styrprogrammet skall läsa av värdet CASE (8 bitar), som finns på inporten $FB, och därefter utföra en av åtta olika subrutiner, SUB0-SUB7. Vilken subrutin som utförs bestäms av värdet på variabeln CASE. Om CASE = 0 utförs SUB0, om CASE = 1 utförs SUB1 osv. Om CASE > 7 skall ingen av subrutinerna utföras. Adresserna till de åtta olika subrutinerna skall finnas lagrade i minnet i en tabell med början på adressen $C0. Vid laddning av programmet i minnet skall också tabellen med subrutinadresserna laddas. Skriv ett huvudprogram i assemblerspråk som först initierar stackpekaren till $20 och sedan läser värdet som finns på inporten $FB (CASE). Därefter anropas en av subrutinerna

enligt ovan om CASE < 8. Programmet utformas som en evighetsslinga. Det skall utformas som flerval och inte som upprepade tvåval.

Subrutinernas hexadecimala startadresser skall vara 80, 67, 75, 52, 90, B9, AF och E0.

9.27 Styrenheten i en maskin innehåller en FLIS-dator. I en styrsekvens skall styrenheten invänta en negativ flank på en binär signal från en givare, ansluten till bit nr 3 på inport $FC. Skriv en generell subrutin NEDGE i assemblerspråk för FLIS-processorn som detekterar en negativ flank på en av bitarna på inporten $FC. Vid anrop av subrutinen skall register A innehålla numret för den bit på inport $FC som man vill undersöka. Övriga bitar på inporten $FC har okända värden. Återhopp från subrutinen skall göras så snart en negativ flank har upptäckts. Endast register CC får vara förändrat vid återhoppet.

Ledning: 1. Tänk på att signalen från början kan ha värdet 0 eller 1.
2. Man kan använda en tabell med bitmasker för de 8 olika fallen (bit 0-7).
Visa också hur subrutinanropet ser ut i huvudprogrammet i det aktuella fallet.
Hur skall programmet ändras om man istället vill detektera en positiv flank?

9.28 Skriv en subrutin PCNT i assemblerspråk för FLIS-processorn, som söker igenom en sträng med 8-bitars dataord i minnet och räknar alla dataord där den vänstra halvan bildar ett fyrabitars binärt tal som är mindre än 6, dvs bitarna 7654 bildar ett binärt tal < 6. Vid anrop av subrutinen skall adressen till strängen finnas i X-registret. Strängen avslutas med datavärdet $FF. Det sökta antalet skall returneras i A-registret. Endast A-registret och flaggregistret får vara förändrat vid återhopp.

9.29 Skriv en subrutin SWAP för FLIS-processorn, som byter plats på databitarna i ett block i minnet så att bitarna 76543210 ersätts med 32107654 i hela blocket. Vid anrop av subrutinen finns antalet 8-bitars dataord i blocket i A-registret (högst 50 st) och begynnelseadressen i X-registret. Vid återhopp från subrutinen får endast CC-registret vara förändrat.

9.30 Skriv en subrutin EXCHG för FLIS-processorn, som byter plats på minnesinnehållen i två lika stora minnesareor med 8-bitars dataord. Vid anrop av subrutinen finns antalet dataord i vardera minnesarean i A-registret (högst 30 st) och adresserna till de två dataareorna i X-registret och Y-registret. Vid återhopp från subrutinen får endast register CC vara förändrat.

9.31 I minnet i ett datorsystem med FLIS-processorn finns ett antal 8-bitars tal med tecken lagrade i en tabell på adressen DTAB och framåt (ökande adress). Tabellen avslutas med talet 0. Skriv en subrutin ODDCNT i assemblerspråk som tar reda på hur många av de positiva talen i tabellen som är udda. Antalet tal i tabellen som uppfyller detta skall finnas i A-registret vid återhopp. Endast register A och register CC får vara förändrade vid återhopp från subrutinen.

Ledning: Man vet här vad bit 0 och 7 skall ha för värden.

9.32 Skriv en subrutin BLKMAN som nollställer bit 7 och inverterar bit 5 i ett block med 16 st 8-bitars dataord i minnet. Adressen till dataordet med lägst adress finns i X-registret vid anrop av subrutinen. Vid återhopp från subrutinen får endast CC-registret vara förändrat. Assemblerspråk för FLIS-processorn skall användas.

10 Adressavkodning

10.1 Ett datorsystem med 64 kbyte adressrymd ska konstrueras. Konstruera adressavkodningen för följande:

2 kbyte RWM från adress 0
8 kbyte ROM på de högsta adresserna
En I/O-area på 8 byte med start på adress $C5A0

Du har tillgång till
8 kbyte ROM-modul
2 kbyte RWM-modul

a) Använd fullständig adressavkodning
b) Använd ofullständig adressavkodning

10.2 Ett datorsystem med 64 kbyte adressrymd ska konstrueras. Konstruera adressavkodningen för följande:

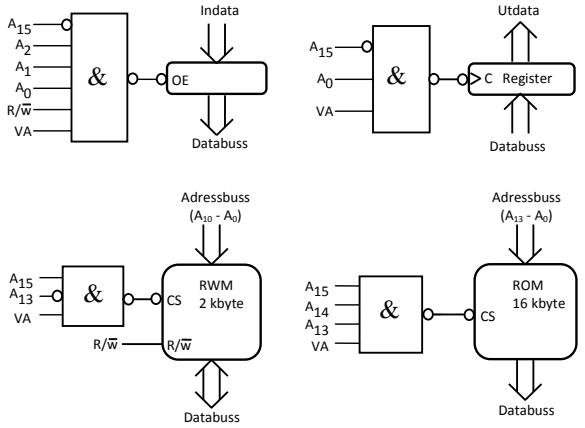
4 kbyte RWM från adress 0
8 kbyte ROM på de högsta adresserna
En I/O-area på 256 byte med start på adress $6000

Du har tillgång till
8 kbyte ROM-modul
4 kbyte RWM-modul

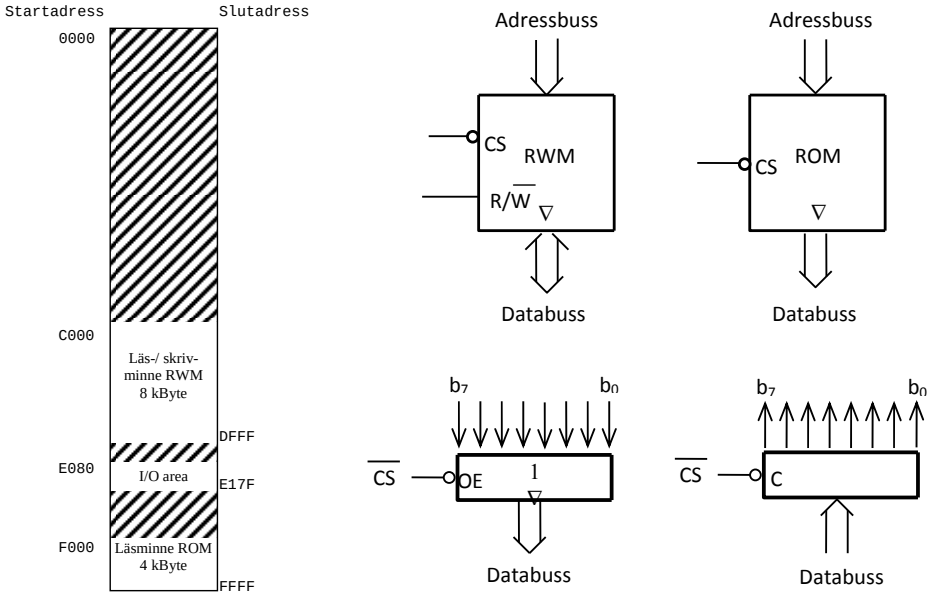
a) Använd fullständig adressavkodning
b) Använd ofullständig adressavkodning

10.3 I ett datorsystem, med synkron buss, 64 kbyte adressrymd och med 8 bitars databuss används en 16 kbyte ROM-kapsel för att lagra program och en 2 kbyte RWM-kapsel för att lagra variabler. Dessutom används en 8-bitars "three-state"-buffert som inport och ett 8-bitars register som utport. I nedanstående figur visas hur "chip-select"-avkodningen för dessa är utförd.

a) Redogör för var i adressrummet processorn kan läsa eller skriva i de olika enheterna. Alla adresser där en enhet kan nås skall redovisas.
b) Konstruktören upptäcker vid provning av systemet att märkliga saker ibland händer vid läsning av indata från inporten. Hjälp konstruktören reda ut vad som händer.



10.4 Ett datorsystem, med synkron buss, 64 kbyte adressrymd och med 8 bitars databuss ska konstrueras. Centralenheten genererar en *VA* signal för att indikera att innehållet på adressbussen är giltigt och en R/W-signal för att indikera läsning respektive skrivning. Figuren nedan visar hur adressrummet ska användas (alla adresser anges på hexadecimal form). Systemet skall bestyckas med 1 st. 8 kbyte RWM-kapsel, 1 st. 4 kbyte ROM-kapsel, 4 st. inportar och 4 st. utportar. Modulernas modeller ges också i figuren. Med tanke på framtida utbyggnad skall alla moduler, som ansluts till adressbussen, vara fullständigt avkodade.

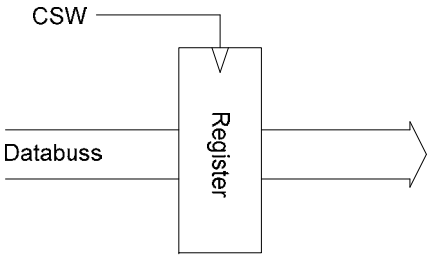


a) Konstruera "chip select"-logiken för de två minneskapslarna.

b) Konstruera "chip select"-logiken för in- och utportarna så att de ligger på adresserna $E0D0 - $E0D7, med inportar på udda adresser och utportar på jämna. Konstruktionen skall ha en gemensam del som skapar "chip select" för hela I/O arean.

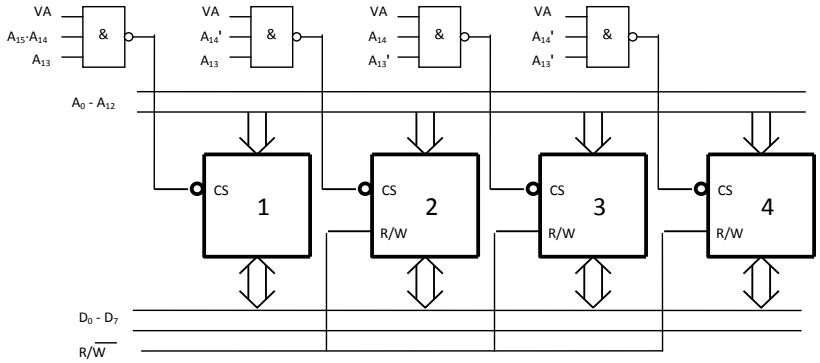
10.5 En utport där gränssnittet utgörs av endast ett register är normalt inte läsbar. Föreslå hur en icke-läsbar utport på adress $E0D1 skall "byggas om" så att den också blir läsbar.

I figuren förutsätts CSW vara en korrekt bildad CS-signal för porten, visa hur CSW bildas. Visa även den kompletta adressavkodningslogiken.



10.6 I ett datorsystem, med synkron buss, 64 kbyte adressrymd och med 8 bitars databuss har minneskretsar kopplats enligt figuren nedan. Primärminnet utgörs av ROM och RWM uppdelat på fyra minneskapslar.

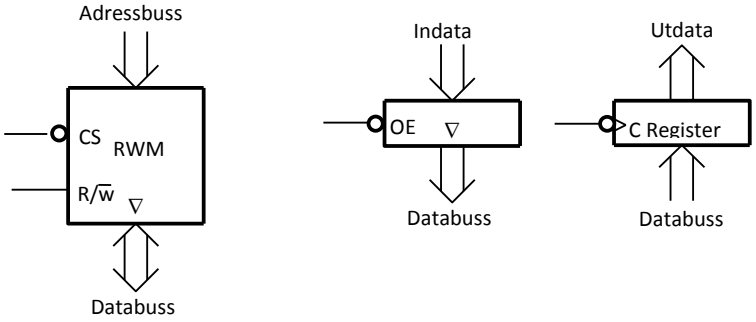
a) Vad betyder förkortningarna ROM och RWM?
b) Hur stort är systemets ROM respektive RWM?
c) Vilka adressområden (hexadecimalt) upptar respektive minneskapsel (1 t o m 4)?
d) Vilken roll har användningen av signalen "VA" vid adresseringen av en minneskapsel?



10.7 Föreställ dig en I/O-modul för vilken RWM-modulen till vänster i figuren nedan har stått modell, dvs. de har samma insignal- och utsignalstruktur mot en processorbuss. I I/O-modulen finns två inportar (ingångs-buffertar) och två utportar (utgångsregister) av den typ som visas till höger i figuren. Den tar upp två adresser, vilket innebär att på varje adress finns en inport och en utport.

a) I ett mikrodatorsystem, med synkron buss, 64 kbyte adressrymd och med 8 bitars databuss har I/O-modulen adresserna $B800 och $B801. Visa hur modulen är ansluten till centralenheten. Adressavkodningen är fullständig. Signalerna VA respektive R/W används för att indikera giltig adress respektive läs- eller skriv- /operation.

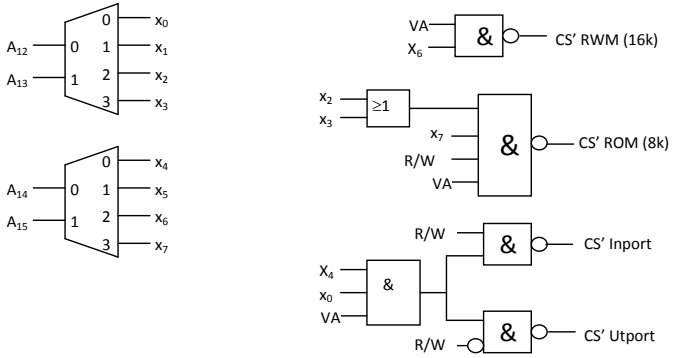
b) Visa hur I/O-modulen principiellt är kopplad internt, dvs. hur de två inportarna och de två utportarna tillsammans med annan logik är ihopkopplade så att modulen kan representeras med en RWM-liknande modul.



10.8 I CS-avkodningen för ett datorsystem med 16 bitars adressbuss används två st. ”2 till 4”-avkodare enligt figuren. Insignalerna till avkodarna är adressbitar från adressbussen. Utsignalerna (x0 – x7) från avkodarna används för att bilda CS-signalerna.

a) Analysera kopplingen och bestäm på vilka adresser minnesmodulerna och portarna är placerade.

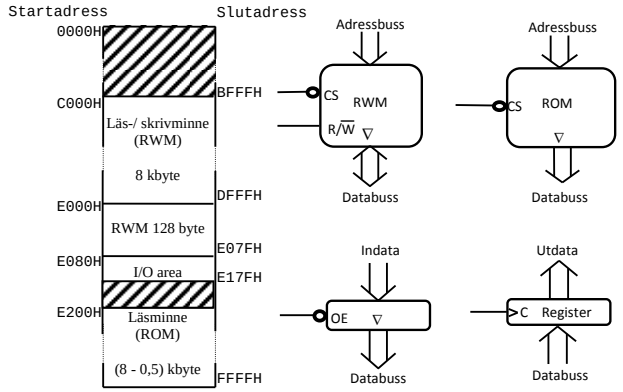
b) Man vill placera en andra ROM-modul i direkt anslutning till den som redan finns. Visa CS-logiken och adressintervallet för denna modul.



10.9 Adressrummet för ett givet datorsystem, med synkron buss, 64 kbyte adressrymd och med 8 bitars databuss visas i bilden till vänster i figuren. Centralenheten genererar en *VA* signal för att indikera att innehållet på adressbussen är giltigt och en R/W-signal för att indikera läsning respektive skrivning. Som läs-/skrivminne används 2 st. RWM-moduler om vardera 8 kbyte och 128 byte. Som läsminne används 1 st. ROM-modul om 8 kbyte. In- och utportar är och kan (vid utökning) placeras i den givna I/O-arean. Modeller av dessa moduler och portar ges också till höger i figuren. Användningen av I/O-arean kan variera från en tillämpning till en annan. Man skall med lätthet kunna byta ut och lägga till portar. Adressavkodningen för enheter i denna area bör därför vara indelad i två steg där det första steget avkodar adresser inom själva I/O-arean och det andra steget dessutom väljer ett specifikt gränssnitt. All avkodning nedan skall vara fullständig och får göras med standardgrindar (INVERTERARE, OCH, ELLER, NAND, NOR, XOR).

a) Konstruera adressavkodningslogiken för RWM-modulerna.

b) Konstruera en tvåstegs adressavkodare för inplaceringen av ett gränssnitt ”IE” som upptar adresserna $E0E0 och $E0E1.



Lösningsförslag

2 Talsystem och binära koder

2.1

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

2.2

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

2.3

d)

2.4

a) '?'
b) '1'
c) '+'

2.5

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

2.6

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

2.7

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

2.8

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

2.9

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) $(1000.01)_2 = (8,25)_{10}$

2.10

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) $(B3.D4)_{16} = (179,828125)_{10}$

2.11

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) $(10.6)_{16} = (10000.011)_2$

2.12

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) $(1010.0011)_2 = (A3)_{16}$

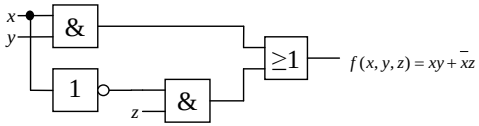
2.13

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

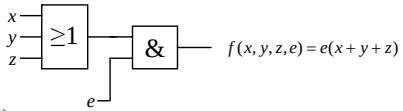
3 Switchnätalgebra

3.1

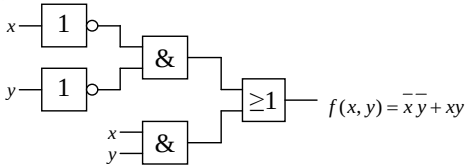
a)



b)



c)



3.2

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

3.3

a) och c).

3.4

3.5

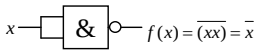
a),c) och d)

3.6

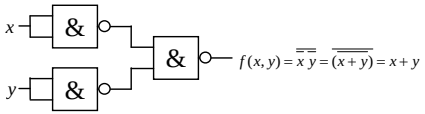
$A=\bar{x}, B=\bar{y}$

3.7

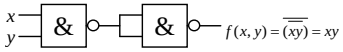
a)



b)

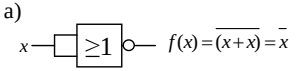


c)

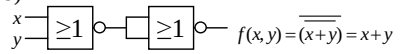


3.8

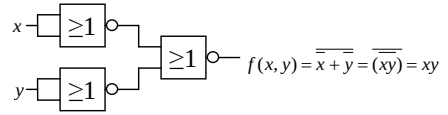
a)



b)

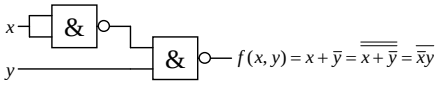


c)

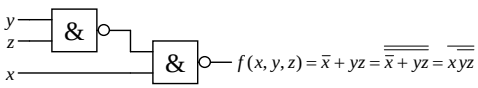


3.9

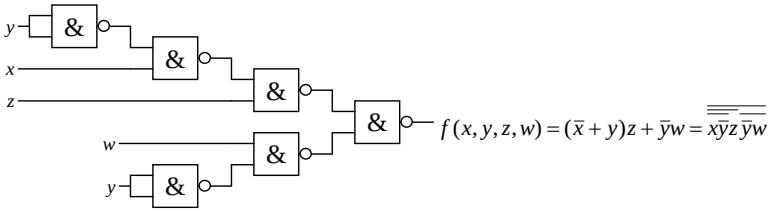
a)



b)

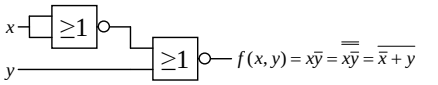


c)

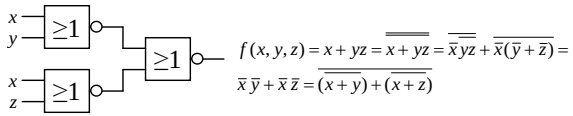


3.10

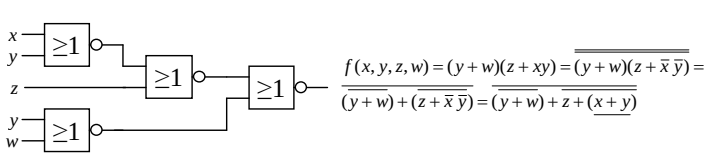
a)



b)



c)



3.11

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

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

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

3.12

d)

3.13

a)

3.14

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

d)

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

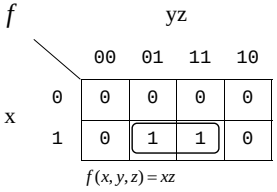
Maxtermer: $\prod$ M (6,10,14) = $(x+\bar{y}+\bar{z}+w)(\bar{x}+y+\bar{z}+w)(\bar{x}+\bar{y}+\bar{z}+w)$

3.15

e)

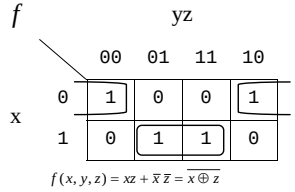
3.16

a)



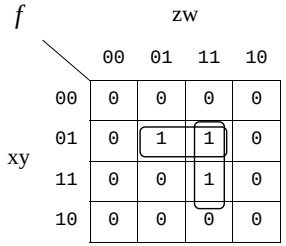
$f(x,y,z)=xz$

b)



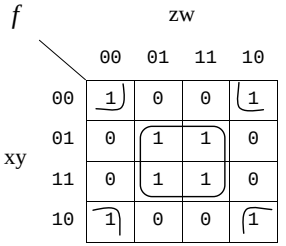
$f(x,y,z)=xz+\bar{x}\,\bar{z}=\overline{x\oplus z}$

c)



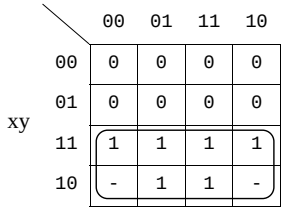
$f(x,y,z,w)=\bar{x}yw+yzw$

d)



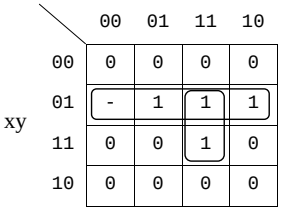
$f(x,y,z,w)=\bar{y}\,\bar{w}+yw=\overline{y\oplus w}$

e)



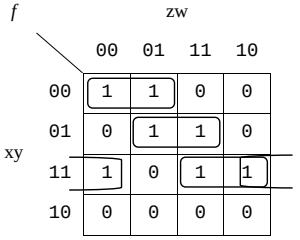
$f(x,y,z,w)=x$

f)



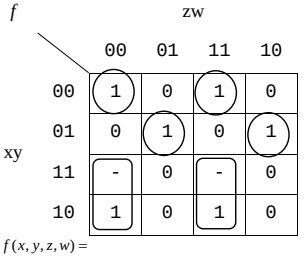
$f(x,y,z,w)=\bar{x}y+yzw$

g)



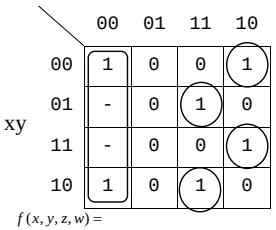
$f(x,y,z,w)=\bar{x}\,\bar{y}\,\bar{z}+\bar{x}yw+xyz+xy\bar{w}$

h)



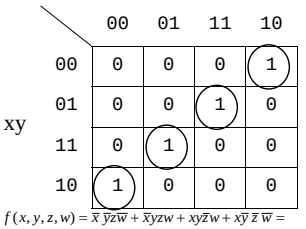
$f(x,y,z,w)=$
$\bar{x}\,\bar{y}\,\bar{z}\,\bar{w}+\bar{x}\bar{y}z\bar{w}+\bar{x}\,\bar{y}zw+\bar{x}yz\bar{w}+x\bar{z}\,\bar{w}+xzw=$
$x(\overline{z\oplus w})+\bar{x}(\overline{z\oplus y\oplus w})$

i)



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

j)



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

3.17

a) f yz

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

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

b) f yz

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

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

c) f zw

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

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

d) f zw

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

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

e) f zw

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 zw

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

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

g) f zw

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

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

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

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

h) f zw

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

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

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

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

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

4 Aritmetik

4.1

```
a) %01010101 = ____85
b) %00111100 = ____60
c) %10101010 = ___170
d) %11000011 = ___195
```

4.2

```
a) %01010101 = ____85
b) %00111100 = ____60
c) %10101010 = ___-86
d) %11000011 = ___-61
```

4.3

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

4.4 a)

Operation	Decimalt	
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≠0 → Z=0
c_8=0 → 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	
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 → Z=1
c_8=1 → 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	
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≠0 → Z=0
c_8=1 → 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.

4.5 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	0	1
+	Y	+	-127	+	1	0	0	0	0	0	0	1
=	R		2		0	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.

4.6 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	
		~~1~~ ~~0~~ 1 ~~0~~ ~~1~~ 0 ~~1~~ ~~0~~ ~~1~~ 0
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	
		~~0~~0
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	
		~~1~~ ~~0~~ ~~1~~ 0 ~~1~~ ~~0~~ ~~1~~ 0
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.

4.7 a)

Operation	Decimalt	
		1 1 1 1
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	
		0
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 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

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

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.

4.8 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

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

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

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

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

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

Det gäller att X=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 0 1 1 1 0 1 0
+ 2~Y	+ -34	+ 1 1 0 1 1 1 1 0
= R	-68	1 0 1 1 1 1 0 0

(bärtal: 1 1 / 1 1 1 1 1)

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

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

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 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

(bärtal: 0 1 1)

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

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

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

(bärtal: 0 1)

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

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

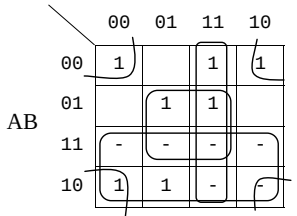
Det gäller att X=64, Y=-65, korrekt resultat R=129. Operationen innebär X=64, 2~Y=65, R=-127. Operationens resultat är fel. Av teckenöverläggningen (två positiva tal adderas men resultatet blir negativt) ser vi att V-flaggan sätts till 1 som indikator på spillet. Resultatet är skilt från 0 varför Z-flaggan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

5 Kombinatoriska nät

5.1

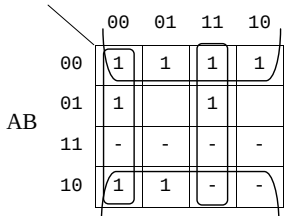
Skapa funktionerna: a,b,c,d,e,f,g och minimera med Karnaughdiagram:

a



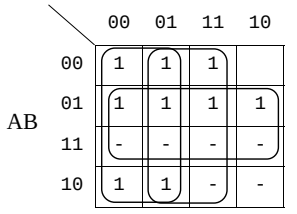
$a = \bar{B}\bar{D} + CD + BD + A$

b



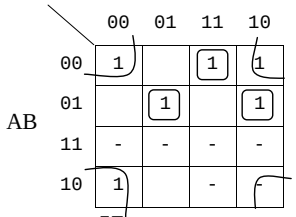
$b = \bar{C}\bar{D} + \bar{B} + CD$

c



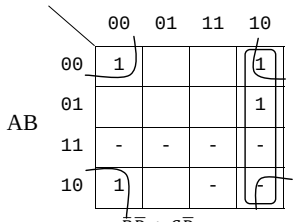
$c = \bar{C} + D + B$

d



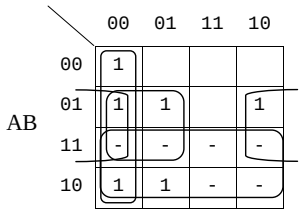
$d = \bar{B}\bar{D} + B \oplus C \oplus D$

e



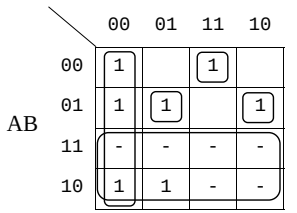
$e = \bar{B}\bar{D} + C\bar{D}$

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:

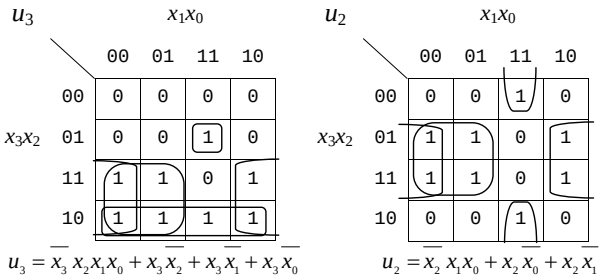
$a = \bar{B}\bar{D} + CD + BD + A$
$b = \bar{C}\bar{D} + \bar{B} + CD$
$c = \bar{C} + D + B$
$d = \bar{B}\bar{D} + B \oplus C \oplus D$
$e = \bar{B}\bar{D} + C\bar{D}$
$f = \bar{C}\bar{D} + B\bar{D} + A\bar{C} + A$
$g = \bar{C}\bar{D} + A + B \oplus C \oplus D$

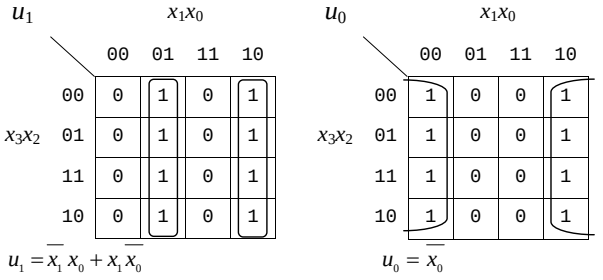
5.2

5.3

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

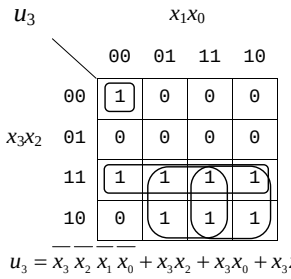


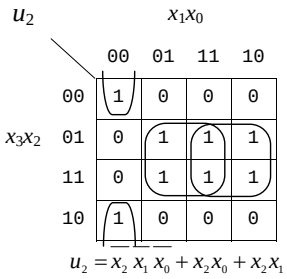


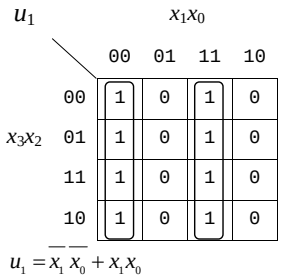
5.4

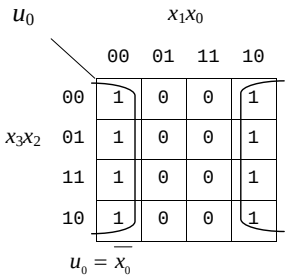
Funktionstabell för "4-bitars DECREMENT":

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







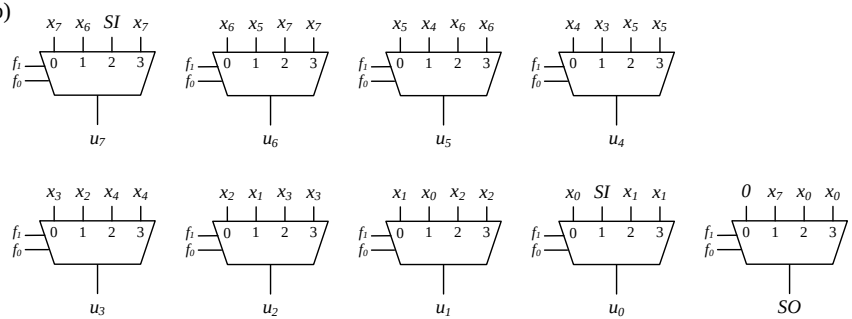


5.5

a)

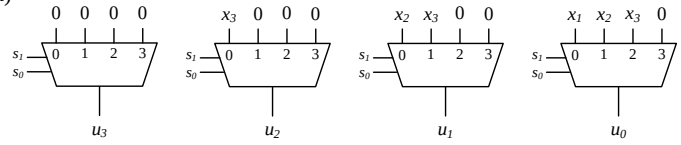
f_1	f_0	u_7	u_6	u_5	u_4	u_3	u_2	u_1	u_0	SO
0	0	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	0
0	1	x_6	x_5	x_4	x_3	x_2	x_1	x_0	SI	x_7
1	0	SI	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
1	1	x_7	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0

b)

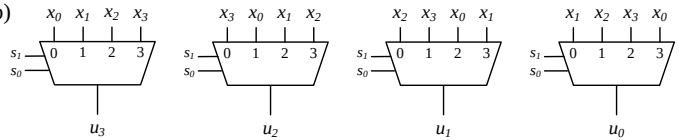


5.6

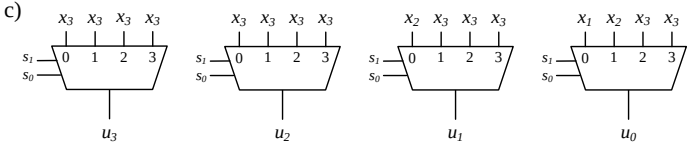
a)



b)

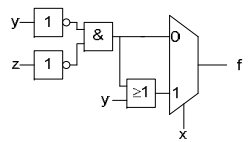
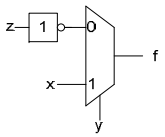
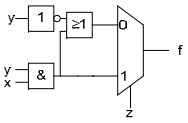


c)



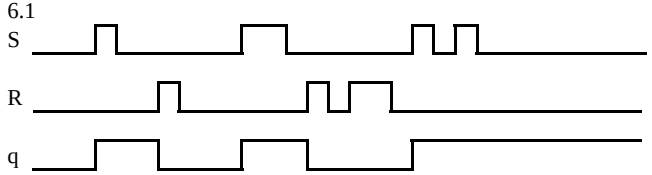
5.7

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

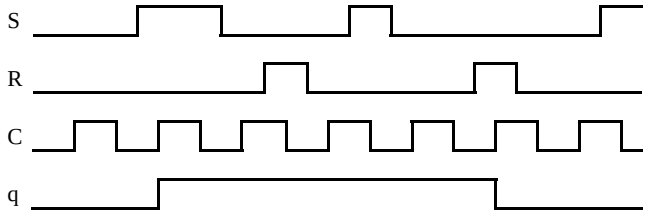
x är selektorvariabel:	*y* är selektorvariabel:	*z* är selektorvariabel:
$f = (\bar{x} + x)(\bar{y}\bar{z}) + xy =$ $\bar{x}(\bar{y}\bar{z}) + x(\bar{y}\bar{z}) + xy =$ $= \bar{x}(\bar{y}\bar{z}) + x(\bar{y}\bar{z} + y)$	$f = \bar{y}(\bar{z}) + y(x)$	$f = \bar{z}\bar{y} + (z + \bar{z})(xy) =$ $= \bar{z}(\bar{y} + xy) + z(xy)$
		
Totalt 5 grindar	Totalt 1 grind	Totalt 3 grindar

6 Sekvensnät

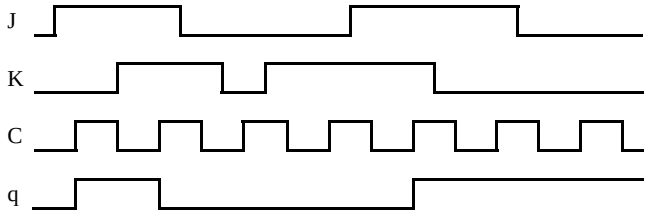
6.1



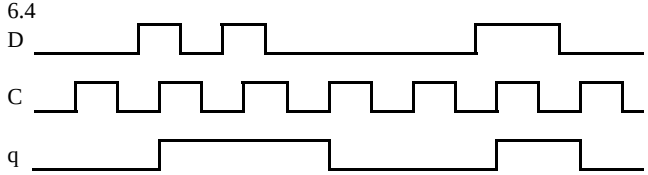
6.2



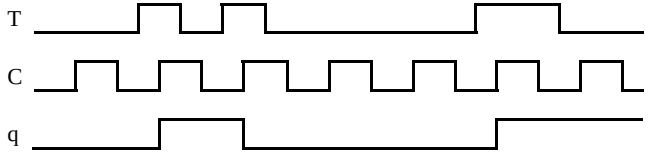
6.3



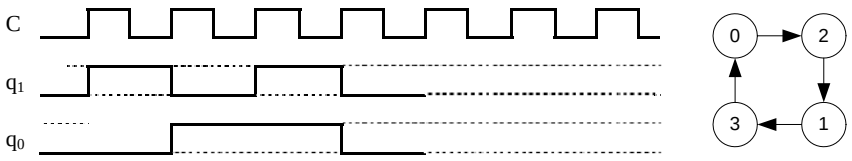
6.4



6.5

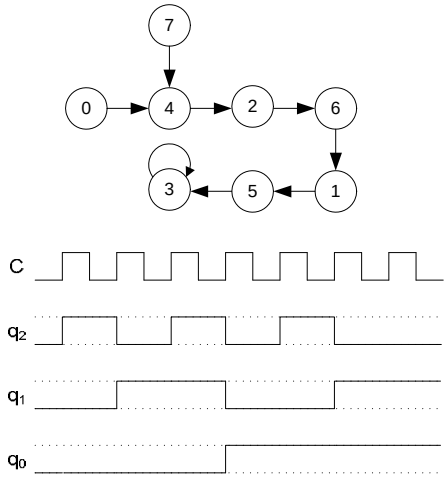


6.6



6.7

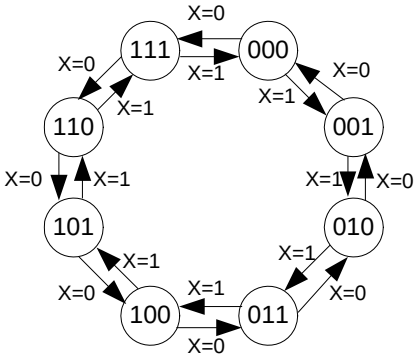
$T_2 = q_0' + q_1'$
$T_1 = q_2$
$T_0 = q_1\, q_2$



6.8

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

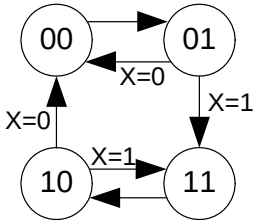
x	q_2	q_1	q_0	T_2	T_1	T_0	q_2^+	q_1^+	q_0^+
0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	1	0	0	0
0	0	1	0	0	1	1	0	0	1
0	0	1	1	0	0	1	0	1	0
0	1	0	0	1	1	1	0	1	1
0	1	0	1	0	0	1	1	0	0
0	1	1	0	0	1	1	1	0	1
0	1	1	1	0	0	1	1	1	0
1	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	1	0	1	0
1	0	1	0	0	0	1	0	1	1
1	0	1	1	1	1	1	1	0	0
1	1	0	0	0	0	1	1	0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



6.9

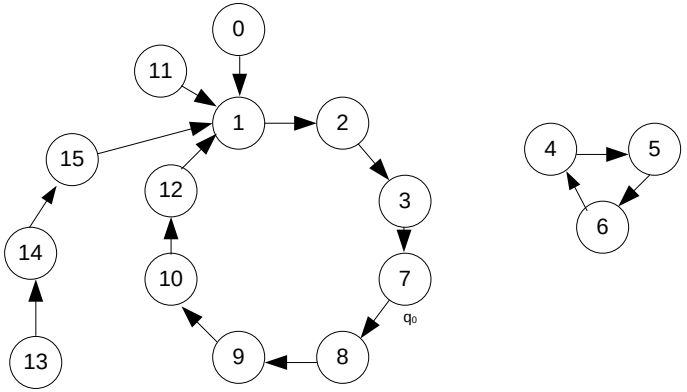
$q_0^+ = q_1'q_0' + xq_1' + xq_0'$
$q_1^+ = q_1q_0 + xq_1 + xq_0$

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

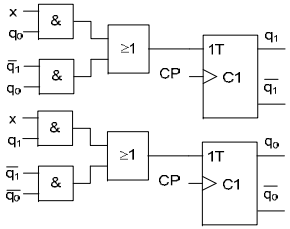


6.10 Alternativ b).

6.11



6.12



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 \ q_1q_0	00	01	11	10
0	0	1	0	-
1	0	1	1	-

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

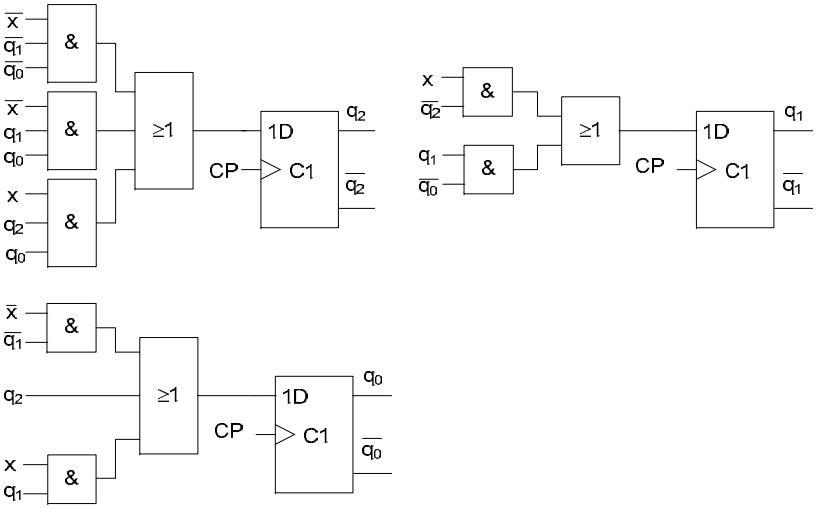
T_0 | q_1q_0

x \ q_1q_0	00	01	11	10
0	1	0	0	-
1	1	0	1	-

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

6.13

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



6.14

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	00	01	11	10
0	0	1	0	0
1	-	1	0	-

$D_1 = \bar{q}_2 q_3$

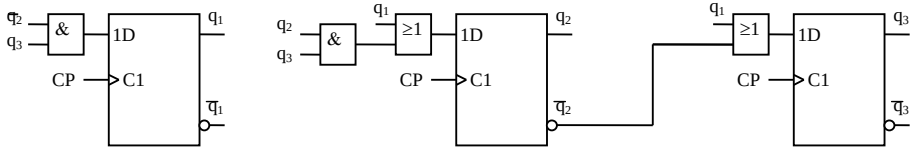
D_2	00	01	11	10
0	0	0	1	0
1	-	1	1	-

$D_2 = q_1 + q_2 q_3$

D_3	00	01	11	10
0	1	1	0	0
1	-	1	1	-

$D_3 = q_1 + \bar{q}_2$

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	00	01	11	10
0	0	1	0	0
1	-	0	1	-

$T_1 = \bar{q}_1\bar{q}_2 q_3 + q_1 q_2$

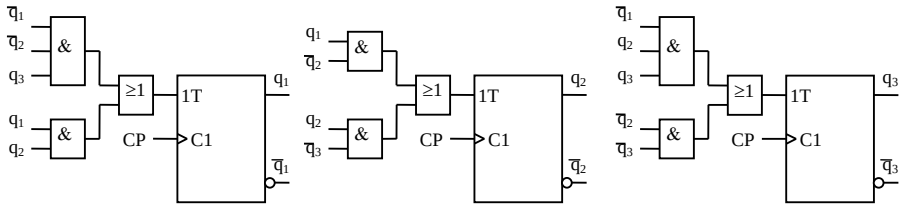
T_2	00	01	11	10
0	0	0	0	1
1	-	1	0	-

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

T_3	00	01	11	10
0	1	0	1	0
1	-	0	0	-

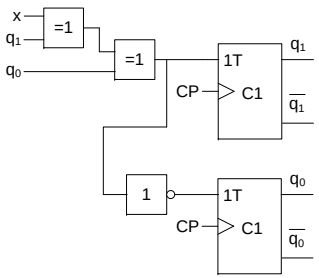
$T_3 = \bar{q}_2\bar{q}_3 + q_1 q_2 q_3$

Realisering med T-vippor



6.15

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



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

$T_1 = x \oplus q_1 \oplus q_0$

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

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

6.16

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

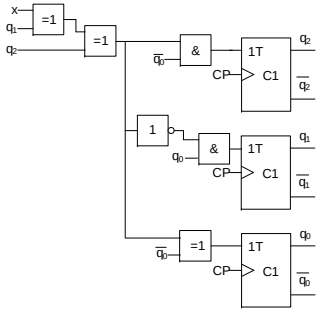
xq_2 \ q_1q_0	00	01	11	10
00		1		
01			1	
11		1		
10			1	

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

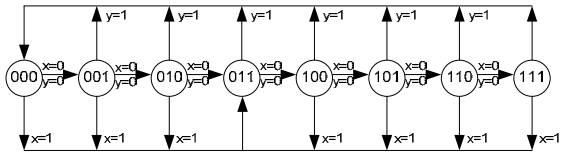
T_0

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

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



6.17



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

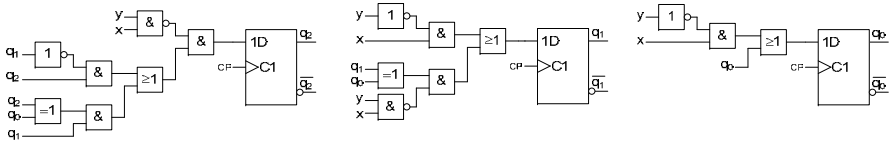
$D_2 = \overline{xy}[q_2\overline{q_1} + (q_2 \oplus q_0)q_1]$

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

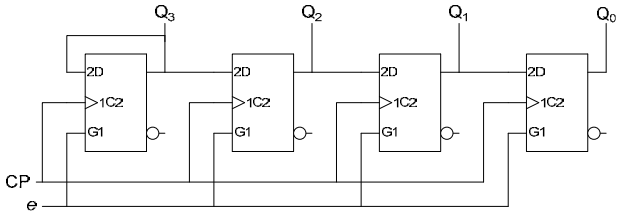
$D_1 = x\bar{y} + \overline{xy}(q_1 \oplus q_0)$

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

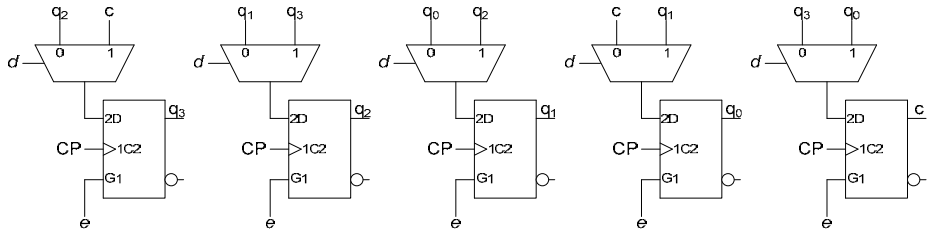
$D_0 = x\bar{y} + \overline{q_0}$



6.18



6.19



7 Dataväg ALU och minne

7.1

	Operation (RTN)	Aktiva styrsignaler
a)	T → A	LD_A,OE_T
b)	T → A,R	LD_A, LD_R, OE_T
c)	R → T, A → R T → A = A ↔ R	1) OE_R, LD_T 2) OE_A, LD_R 3) OE_T, LD_A

7.2 a)

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

b)

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

c)

Cykel	Operation (RTN)	Aktiva styrsignaler
1	~A → R	OE_A, F(0,1,0,1), LD_R
2	R → A	OE_R, LD_A

d)

Cykel	Operation (RTN)	Aktiva styrsignaler
1	2~A → R	OE_A, F(0,1,0,1), C_{in}, LD_R
2	R → A	OE_R, LD_A

7.3

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

7.4 a)

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

b)

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

c)

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

d)

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

7.5

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

7.6

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

7.7

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

8 Styrenheten

8.1 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`

8.2 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`

8.3 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`

8.4 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`

8.5 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`

8.6 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`

8.7 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`

8.8 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`

8.9 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`

8.10

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

8.11

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

8.12

Tillstånd	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_4	n→T; PC+1→PC	MR; LDT; INC_{PC}
Q_5	M(A+Y)>>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(A+Y);	OE_R; MW; g_{12}; NF

8.13

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

9 Assemblerprogrammering

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

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

9.3

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

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

9.4

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		

9.5

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			

9.6

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			

9.7

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

9.8

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

9.9

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	-	-

9.10

```
; 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
```

9.11

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

main:      JSR       DipSwitchOr
           BRA       main

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

9.12

```
;    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
```

9.13

```
;    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
```

9.14

```
;    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
```

9.15

```
;    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"
```

9.16

```
;    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
```


9.17

```
;    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
```

9.18

```
;    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
```

9.19

```
*    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
```

9.20

uppgift d)

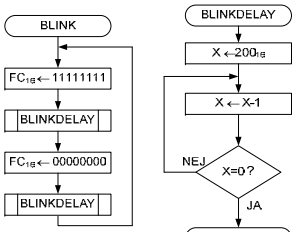
```
LedDisplay:     EQU     $FC
                ORG     $20
main:           JSR     Blink
                BRA     main

; 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)

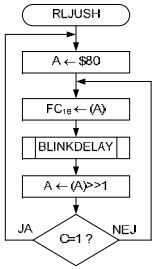


9.21

a) b)

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

c)



9.22

```
; 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
```

9.23

```
;    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
```

9.24

```
;    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
```

9.25

```
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
```


9.26

```
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
```

9.27

```
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
;    vänteloopparna 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
```

9.28

```
;    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
```

9.29

```
;    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
```


9.30

```
;    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
```

9.31

```
;    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
```

9.32

```
;    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
```

10 Adressavkodning

10.1 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	

10.2 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	

10.3 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	0	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.

10.4

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:

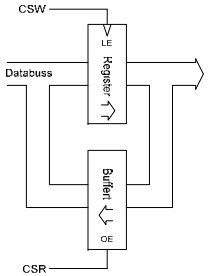
utport: $\overline{CS_{E0D0}} = \overline{\overline{CS_{IO}} \cdot \overline{A2} \cdot \overline{A1} \cdot \overline{A0} \cdot \overline{R/W}}$

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

10.5

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

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			

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

10.6

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	

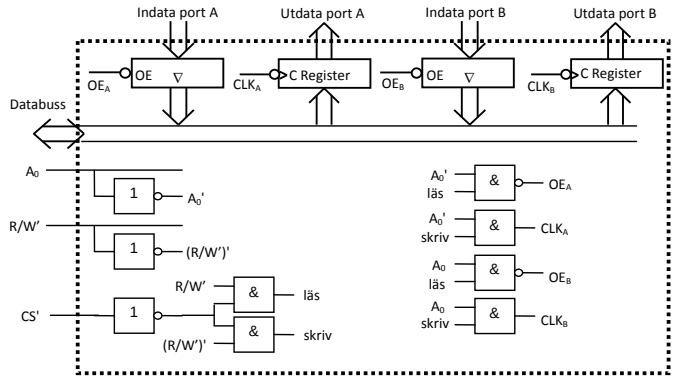
d) VA = 1 (*Valid Memory Address*) anger att adressbussens värde är stabilt och därför används denna signal för att grinda adresserna till minnet. Då kommer de adresser som finns ut på bussen under tiden som adressledningar håller på att ändras aldrig att nå primärminnet.

10.7 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	

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

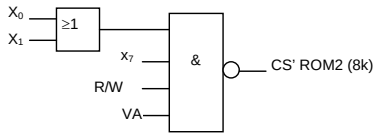
b) I/O-modulens interna struktur bör utformas på följande sätt:



10.8 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)



10.9

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:

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