Upg 1

1a) R=X-Y utförs som R=X+Y_{1komp} +1 Y_{1komp} = 11010.

```
                110001
 X               11100
+Y1komp        + 11010
=R             = 10111
```

1b) N=1; Z=0; V=0; C_5=1 ⇒ C=0.

1c) Tal UTAN tecken:
X=28; Y=5; R=23. (Verkar rimligt ty C=0 och 28-5=23 är rätt)

1d) Tal MED tecken:
X=-4; Y=5; R=-9. (Verkar rimligt ty V=0 och -4-5=-9 är rätt)

1e) 1: J,N; 2: N,J; 3: J,J; 4: N,N

1f) Kännetecknet på att det å ena sidan är en **Graykod** är att kodorden för två närliggande siffror bara skiljer sig åt i en position och

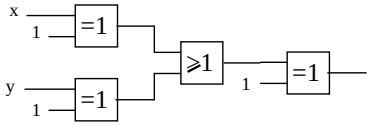
kännetecknet på att det är en **Excess-3 kod** är att kodordet för siffran 0 ej är 0000 utan förskjutet tre steg så att det i Graykoden blir 0010

OBS! Varje lösning på NY SIDA!

Upg 2

2a) De Morgan: (X•Y)' = X'+Y' kan skrivas om till X•Y = (X'+Y')'

Inverterare kan realiseras med XOR-grindar där ena ingången är ett.

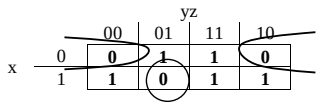


2b) Funktionstabell:

x	y	z	yz	xz'	x'y'z	f
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	0	0	0	0
0	1	1	1	0	0	1
1	0	0	0	1	0	1
1	0	1	0	0	0	0
1	1	0	0	1	0	1
1	1	1	1	0	0	1

Konjunktiv normal form:
f = (x+y+z)(x+y'+z)(x'+y+z')

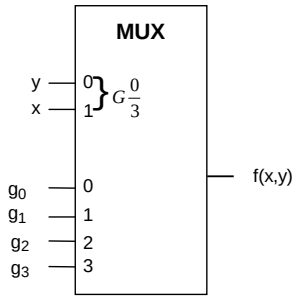
Konjunktiv minimal form → Minimera!



Minimal form: f = (x+z)(x'+y+z')

2c)
Väljarens funktionstabell och symbol

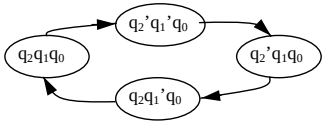
x	z	f
0	0	g_0
0	1	g_1
1	0	g_2
1	1	g_3



OBS! Varje lösning på NY SIDA!

Uppg 3

3a) Undersök räknarsekvensen. Tillståndsdiagram:



Studeras räknarens utsekvens ser vi att vi har 4 tillstånd och att q_0 har värdet ett i alla tillstånd. Detta innebär att räknaren kan realiseras med endast två vippor.

4 tillstånd; $q_0 = 1$.

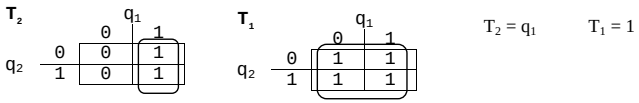
Sätter upp en tabell med q_2q_1. T-Vippa. Funktions och excitationstabell

Detta tillst	Nästa tillst		
q_2q_1	q_2q_1	T_2	T_1
00	01	0	1
01	10	1	1
10	11	0	1
11	00	1	1

T	Q^+
0	Q
1	Q'

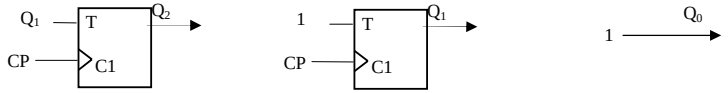
Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

Minimering:

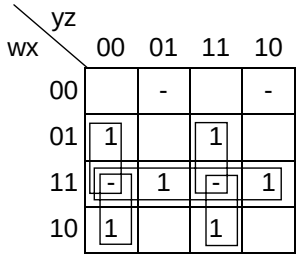


$T_2 = q_1$ $T_1 = 1$

Realisering

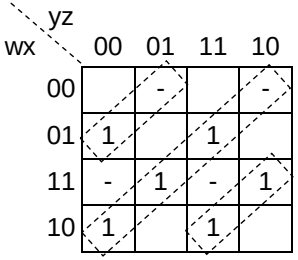


3b) Den givna funktionstabellen tjänar som funktionsbeskrivning.
Lösning - NAND-grindar:



Minimal disjunktiv form: f = wx + xy'z' + xyz + wy'z' + wyz
Realisering: 1 st 5-ingångars, 4 st 3-ingångars och 1 st 2-ingångars NAND

3c) **Lösning - fritt grindval**:



Man ser diagonalerna och vet då att det blir en XOR-lösning

Minimal blandad form: f = (w⊕x)⊕(y⊕z)
Realisering: 3 st 2-ingångars XOR

OBS! Varje lösning på NY SIDA!

Uppg 4

4a)

State nr	RTN-beskrivning	Styrsignaler (=1)
0	PC→MA, PC+1→PC	OE_{PC}, LD_{MA}, IncPC
1	M→MA	MR, LD_{MA}
2	M→T	MR, LD_T
3	A-T, Flaggor→CC, NF	OE_A, f_3, f_2, g_0, LD_{CC}, NF

4b)

0) Förbered för läsning av operand (ardess) i minnet, Öka PC
1) Förbered för läsning av operand (data) i minnet, Öka PC
2) Läs operand (data) från minnet och placera i T.
3) Utför en subtraktion och spara ALU:ns flaggor i flaggregistret.

Instruktionen är CMPA Adr

4c)

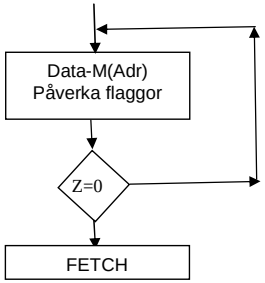
JSR Adr som består av OP-kod och adressoperand upptar 2 bytes i minnet. Instruktionen tar 7 klockcykler. (Utförandefasen tar7-FETCH =5 klockcykler)

RTN beskrivning:
S-1→S
PC→M(S)
Adr→PC

State nr	RTN-beskrivning	Styrsignaler (=1)
0	PC→MA, PC+1→PC	OE_{PC}, LD_{MA}, IncPC
1	M→T	MR, LD_T
2	S→MA	OE_S, LD_{MA}
3	PC→M, T→R	OE_{PC}, MW, f_1, LD_R
4	R→PC	OE_R, LD_{PC}, NF

4d) Instruktionen upptar 3 byte bestående av OP-kod, dataoperand och sist adressoperand.

Flödesplan:



Sekvens	Adress (Hex)	Hopp villkor G_K	Hopp adress (Hex)	RTN	Styrsignaler (aktiva)
WFP #Data,Adr	E3	G_{0F}=1	500	PC→MA, PC+1→PC	OE_{PC}, LD_{MA}, IncPC
	500			M→T	MR, LD_T
	501			PC→MA, PC+1→PC	OE_{PC}, LD_{MA}, IncPC
	502			M→MA	MR, LD_{MA}
	503			M-T, Flaggor→CC	MR, f_3, f_2, g_0, LD_{CC}
	504	G_{13}=Z'	503		
	505	G_{0F}=1	108	NF	(NF)

Uppg 5

5a) Ett assemblerdirektiv är direktiv till assemblatorn som översätter ett assemblerprogram till maskinkod. Exempel på assemblerdirektiv : ORG $30 som anger att maskinkodens startadress är adress 30_{16} i minnet.

En assemblerinstruktion består av en bokstavskombination, en förkortning, som anger vad assemblerinstruktionen utför. Exempel INCA som anger INcrement register A.

5b) BRA-instruktionen upptar två bytes i minnet, vilket innebär att ”från-adressen” är $CF. ”

Till-adressen” är $BE.

Då beräknad offset här blir $EF blir maskininstruktionens offset $EF.

Maskininstruktionen blir $5A $EF.

	± -10 10
Till Adr	$ B E
- Från Adr	- $ C F
= Offset	= $ E F

5c)

Adr	Kod		Assemblerprogram	
C0	??	T3	RMB	3
C1	??			
C2	??			
C3	03	T4	FCB	3,$A,%11
C4	0A			
C5	03			
C6	00		NOP	
C7	0B		LDA	128
C8	80			
C9	82		LDB	T1,X
CA	11			
CB	14		STB	T3
CC	C0			
CD	22		EORB	#T2
CE	3A			
CF	42		INCB	

OBS! Varje lösning på NY SIDA!

Uppg 6

```
* Programmet BORR borrar ett hål när operatörens startknapp aktiveras.
* Programmet inväntar att startknappen släpps innan ett nytt hål kan
* borras.
        org   $40
OutPort equ   $fd       Utport, styrsignaler till borrmaskinen
InPort  equ   $fe       Inport, statussignaler från borrmaskinen

Passiv  equ   $ff       Passiva styrsignaler
StartOn equ   %10000000 Startknapp från operatör
MotorOn equ   %11110111 Styrord för att starta borrmotorn
Drill   equ   %11100111 Styrord för att sänka borret med startad
*                       borrmotor
Lift    equ   %11110111 Styrord för att höja borret med startad
*                       borrmotor


Bottom  equ   %00000010


Start lda   #Passiv   Sätt passiva styrsignaler
      sta   OutPort
Next
B7is0 lda   InPort    Invänta start från operatör
      bita  #StartOn
      beq   B7is0

      lda   #MotorOn  Påbörja borrningen
      sta   OutPort
      lda   #Drill
      sta   OutPort

NotDown lda Inport    Invänta genomborrning
      bita  #Bottom
      bne   NotDown

      lda   #Lift     Avsluta borrningen
      sta   OutPort
      lda   #Passiv
      sta   OutPort

B7is1 lda   InPort    Invänta att operatören släpper startknappen
      bita  #StartOn
      bne   B7is1
      bra   Next
```