Lösningsförslag tenta 2011-10-17
(Version 4 med reservation för eventuella fel.)

1. $X = 0110101_2$; $Y = 1101011_2$ (7 bitars ordlängd)

a) $[-2^{n-1}, +2^{n-1}-1] = [-2^{7-1}, +2^{7-1}-1] = [-64, +63]$ **(1p)**

b) $[0, 2^n-1] = [0, 2^7-1] = [0, 127]$ **(1p)**

c)
```
S = X+Y
76543210    bitnummer
11111110    carry
 0110101    X
+1101011    Y
 0100000 =  S
```
(1p)

d) $\underline{N} = s_6 = \underline{0}$
$\underline{Z} = \underline{0}$ ($S \neq 0$)
$\underline{V} = x_6*y_6*s_6' + x_6'*y_6'*s_6 = 0*1*0' + 0'*1'*0 = \underline{0}$
$\underline{C} = c_7 = \underline{1}$

(1p)

e)
```
D = X+Y1k+1
76543210    bitnummer
01101011    carry
 0110101    X
+0010100    Y1k
 1001010 =  D
```
(1p)

f) $\underline{N} = d_6 = \underline{1}$
$\underline{Z} = \underline{0}$ ($D \neq 0$)
$\underline{V} = x_6*y_{6lk}*d_6' + x_6'*y_{6lk}'*d_6 = 0*0*1' + 0'*0'*1 = \underline{1}$
$\underline{C} = c_7' = 0' = \underline{1}$

(1p)

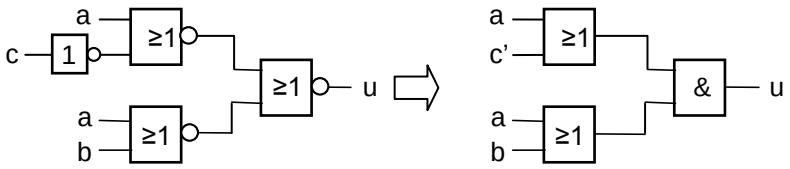
g) $\underline{X} = 0110101_2 = 35_{16} = 3*16 + 5 = \underline{53}$
$\underline{Y} = 1101011_2 = 6B_{16} = 6*16 + 11 = 96 + 11 = \underline{107}$
$\underline{S} = 0100000_2 = 20_{16} = 2*16 + 0 = \underline{32}$ Resultatet S är felaktigt eftersom C = 1.
$\underline{D} = 1001010_2 = 4A_{16} = 4*16 + 10 = 64 + 10 = \underline{74}$ Resultatet D är felaktigt eftersom C = 1. **(1p)**

h) ($x_6 = 0$, pos) $\underline{X} = 0110101_2 = \underline{53}$
($y_6 = 1$, neg) $Y_{2k} = 2^7 - 107 = 128 - 107 = 21$ $\underline{Y}$ motsvarar $\underline{-21}$
($s_6 = 0$, pos) $\underline{S} = 0100000_2 = \underline{32}$ Resultatet S är korrekt eftersom V = 0.
($d_6 = 1$, neg) $D_{2k} = 2^7 - 74 = 128 - 74 = \underline{54}$ $\underline{D}$ motsvarar $\underline{-54}$. Felaktigt eftersom V = 1. **(1p)**

i) Antal booleska funktioner av n variabler = $2^{2^n} = 2^{2^2} = 2^4 = \underline{16}$ **(1p)**

j) $N = -75{,}75_{10} = -1001011.11 = -1.00101111*2^6$ Format: s/c/f
s = 1 (neg); $c = \exp + 2^{8-1} - 1 = 6 + 127 = 133 = 10000101_2$; f = mantissa - 1
$N_{flyt} = 1/100\ 0010\ 1/001\ 0111\ 1000\ 0000\ 0000\ 0000_2 = C2978000_{16}$ **(2p)**

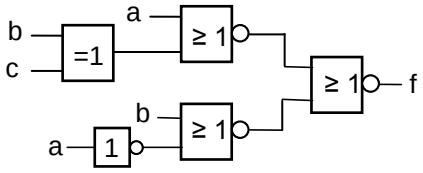
k)



$\underline{u} = (a+c')(a+b) = a + bc'$ **(2p)**

2. $\underline{f} = abc + a'b'c + abc' + a'bc' = (a+b+c)(a+b'+c')(a'+b) =$

$= [a+(b+c)(\ b'+c')](\ a'+b) = [a+(b\oplus c)](\ a'+b)$

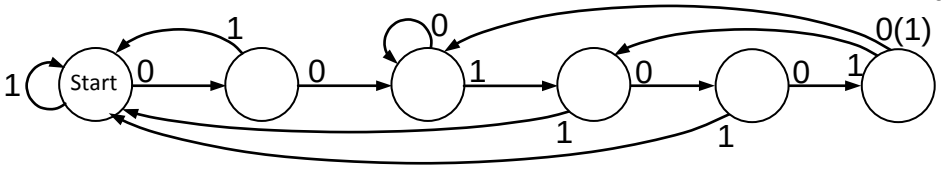


a \ bc	00	01	11	10
0	(0)	1	(0)	1
1	0	0	1	1

(4p)

3.

a)



6 tillstånd ger minst 3 vippor ($2^2 < 6 \leq 2^3$)

(Ej utsatta utsignaler = 0)

(4p)

b)

,

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

För T-vippor gäller att $q^+ = q$ för $T = 0$ och $q^+ = q'$ för $T = 1$.

T_2 = q_1q_0

q_2 \ q_1q_0	00	01	11	10
0	0	0	0	1
1	0	1	-	0

T_1 = q_1q_0

q_2 \ q_1q_0	00	01	11	10
0	1	0	1	1
1	1	1	-	1

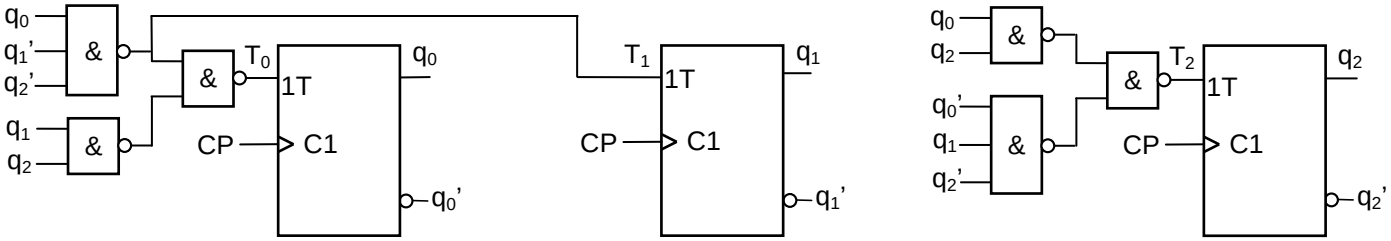
T_0 = q_1q_0

q_2 \ q_1q_0	00	01	11	10
0	0	1	0	0
1	0	0	-	1

$T_2 = q_2'q_1q_0' + q_2q_0$

$T_1 = q_2 + q_1 + q_0'$

$T_0 = q_2'q_1'q_0 + q_2q_1$



(6p)

4. $5A - 7(B + 1) = 5(A - B - 1) - 2(B + 1)$

CP	RTN	Styrsignaler (=1)
1	B → T	OE_B, LD_T
2	A - T - 1 → R	OE_A, f_3, f_2, LD_R
3	2R → R, R → T	OE_R, f_3, f_1, f_0, LD_R, LD_T
4	2R → R	OE_R, f_3, f_1, f_0, LD_R
5	R + T → R	OE_R, f_3, f_1, LD_R
6	B → T	OE_B, LD_T
7	R - T - 1 → R	OE_R, f_3, f_2, LD_R
8	R - T - 1 → R	OE_R, f_3, f_2, LD_R
9	R → A	OE_R, LD_A

(5p)

5. a)

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5 \cdot I_{xx}$	PC → MA, PC+1 → PC	OE_{PC}, LD_{MA}, IncPC
Q_6	$Q_6 \cdot I_{xx}$	M → MA	MR, LD_{MA}
Q_7	$Q_7 \cdot I_{xx}$	M → T	MR, LD_T
Q_8	$Q_8 \cdot I_{xx}$	A - T - C → R, Flags → CC	OE_A, f_3, f_2, g_1, g_0, LD_R, LD_{CC}
Q_9	$Q_9 \cdot I_{xx}$	R → A, (Next Fetch)	OE_R, LD_A, NF

Från början pekar PC på minnesordet efter OP-koden.
Q_5: Minnet adresseras med innehållet i PC, som också ökas med ett.
Q_6: Dataordet efter OP-koden som PC pekade på i minnet hämtas till MA. Det är alltså en adress.
Q_7: Dataordet som adressen pekar på i minnet hämtas till T.
Q_8: Dataordet från minnet och C subtraheras från A. Skillnaden laddas i R. Flaggorna uppdateras.
Q_9: Skillnaden laddas i A.
Detta är SBCA Adr (Subtraktion med borrow.) **(2p)**

b)

State	S-term	RTN-beskrivning	Styrsignaler (= 1)
Q_5	$Q_5 \cdot I_{F1} \cdot (N \oplus V)'$ $Q_5 \cdot I_{F1} \cdot (N \oplus V)$	PC → MA,T PC+1 → PC, (Next Fetch)	OE_{PC}, LD_{MA}, LD_T IncPC, NF
Q_6	$Q_6 \cdot I_{F1}$	M+T+1 → R	MR, f_3, f_1, g_0, LD_R
Q_7	$Q_7 \cdot I_{F1}$	R → PC, (Next Fetch)	OE_R, LD_{PC}, NF

(4p)

6.

a) Assemblatorn arbetar rad för rad, uppifrån och ner. När den skall bestämma värdet på SYMBOL2 är värdet på SYMBOL3 okänt och den borde därför ge ett felmeddelande. **(2p)**

b) JMP- och BRA-instruktioner lägger inte någon återhoppsadress på stacken, så när processorn kommer till RTS sist i subrutinen och ”tror” att den läser återhoppsadressen, så spårar den ur. **(2p)**

c) X-registret kan användas till att lagra ett datavärde, som även kan ökas eller minskas, men främst används det till att lagra en adress till t ex en tabell med datavärden i minnet. Det finns ett antal olika adresseringsmoder som då kan användas t ex ,X; n,X; A,X; B,X; 1,X+ och 1,-X. **(2p)**

d) BGT avser tal med tecken. För 8-bitars tal gäller då talintervallet [-128, 127]. COMA utför –W–1. Hoppvillkoret blir $-W - 1 - 50_{16} > 0$ eller $-W - 1 - 80 > 0$, dvs $W < -81$ eller $-128 \leq W < -81$. Eftersom negativa värden ersätts med 2-komplementet av motsvarande positiva värde blir det verkliga intervallet: $256 - 128 \leq W < 256 - 81$ eller $\underline{128 \leq W < 175}$ **(3p)**

e)

Adr	Data	~	Läge			
60	11 74	4		LDX	#NEXT	
62	81 FC	7		LDAA	-4,X	
64	82 FD	7	LOOP1	LDAB	-3,X	
66	45	4	LOOP2	DECB		
67	00	3		NOP		
68	5E FC	5		BNE	LOOP2	66 – 6A = FC
6A	41	4		INCA		
6B	5B F7	5		BMI	LOOP1	64 – 6D = F7
6D	5A 05	5		BRA	NEXT	74 – 6F = 05
6F	00 FE 0A FC 14	-	TAB	FCB	0,-2,10,-4,20	
74	00	3	NEXT	NOP		

(3p)

f) $\underline{N} = 4+7+(7+(4+3+5)*10+4+5)*2+5+3 = 19+(16+12*10)*2 = \underline{291 \text{ klockpulser (µs)}}$ **(3p)**

7.

```
NIBINC  PSHX            Spara på stack
        PSHA

        STAB    TABCNT  Init räknare för antal dataord
        CLR     OFLCNT  Nollställ räknare för antal overflow

LOOP    TST     TABCNT  Färdigt?
        BEQ     NIBEX   Ja, förbered återhopp

        LDAA    ,X      Nej, hämta nästa tabellvärde
        TFR     A,B     Kopia till B
        ANDB    #$F0    Maska bort låg nibble i B
        STAB    BCOPY   Spara hög nibble

        ORAA    #$F0    Ettställ hög nibble i A
        ADDA    #5      Utför additionen till låg byte
        BCC     NOCY    Ej overflow
        INC     OFLCNT  Det blev carry. Öka overflowräknare

NOCY    ANDA    #$0F    Maska bort hög nibble i A
        ORAA    BCOPY   Kombinera nibblar
        STAA    1,X+    Uppdatera tabell och adressera nästa tabellvärde
        DEC     TABCNT  Minska dataordsräknare
        BNE     LOOP    Färdigt? Nej!

NIBEX   LDAB    OFLCNT  Hämta antal overflow
        PULA            Hämta från stack
        PULX
        RTS

TABCNT  RMB 1           Räknare för antal dataord
OFLCNT  RMB 1           Räknare för antal overflow
BCOPY   RMB 1           Kopia av hög nibble
```

(7p)