Lösningsförslag tenta 2011-08-18 (Med reservation för eventuella fel)

Uppgift 4, steg 6 och 7 styrsignaler rättade 2012-05-21

1. $X = 00010111_2$; $Y = 11101001_2$ (8 bitars ordlängd)

a) $[-2^{n-1}, +2^{n-1}-1] = [-2^{8-1}, +2^{8-1}-1] = [-128, +127]$ **(1p)**

b) $[0, 2^n-1] = [0, 2^8-1] = [0, 255]$ **(1p)**

c)
```
S = X+Y
876543210   bitnummer
111111110   carry
 00010111   X
+11101001   Y
 00000000 = S
```
(1p)

d) $\underline{N} = s_7 = \underline{0}$
$\underline{Z} = \underline{1}$ (S = 0)
$\underline{V} = x_7*y_7*s_7' + x_7'*y_7'*s_7 = 0*1*0' + 0'*1'*0 = \underline{0}$
$\underline{C} = c_8 = \underline{1}$
(1p)

e)
```
D = X+Y1k+1
876543210   bitnummer
000101111   carry
 00010111   X
+00010110   Y1k
 00101110 = D
```
(1p)

f) $\underline{N} = d_7 = \underline{0}$
$\underline{Z} = \underline{0}$ ($D \neq 0$)
$\underline{V} = x_7*y_{7lk}*d_7' + x_7'*y_{7lk}'*d_7 = 0*0*0' + 0'*0'*0 = \underline{0}$
$\underline{C} = c_8' = 0' = \underline{1}$
(1p)

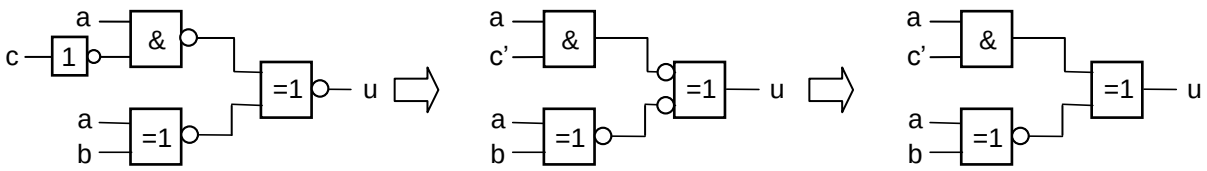
g) $\underline{X} = 00010111_2 = 17_{16} = = 1*16 + 7 = \underline{23}$
$\underline{Y} = 11101001_2 = E9_{16} = 14*16 + 9 = 224 + 9 = \underline{233}$
$\underline{S} = 00000000_2 = \underline{0}$ Resultatet S är felaktigt eftersom C = 1.
$\underline{D} = 00101110_2 = 2E_{16} = 2*16 + 14 = 32 + 14 = \underline{46}$ Resultatet D är felaktigt eftersom C = 1. **(1p)**

h) ($x_7 = 0$, pos) $\underline{X} = 00010111_2 = \underline{23}$
($y_7 = 1$, neg) $Y_{2k} = 2^8 - 233 = 256 - 233 = 23$ $\underline{Y}$ motsvarar $\underline{-23}$
($s_7 = 0$, pos) $\underline{S} = \underline{0}$ Resultatet S är korrekt eftersom V = 0.
($d_7 = 0$, pos) $\underline{D} = \underline{46}$ Resultatet D är korrekt eftersom V = 0. **(1p)**

i) $N_{min} = 1.000...0*2^{-126} = 16*2^{-4}*2^{-126} = 16*2^{-130} = 16*(2^{-10})^{13} \approx 16*(10^{-3})^{13} = 16*10^{-39} = 1{,}6*10^{-38}$ **(2p)**

j) Funktionstabellen för tre variabler har $2^3 = 8$ rader. Funktionen kan ha värdet 0 eller 1 på varje rad, dvs 2 möjligheter. Då blir antalet olika bitmönster (= olika funktioner) $2^8 = \underline{256}$. **(2p)**

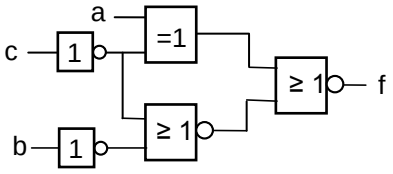
k)



$\underline{u} = (ac')\oplus(a\oplus b') = (ac')\oplus(ab+a'b') = (ac')'(ab+a'b')+ac'(a'b+ab') =$
$= (a'+c)(ab+a'b')+ac'(a'b+ab') = a'b'+abc+a'b'c+ab'c' = \underline{a'b'+b'c'+abc}$ **(3p)**

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

ab \ cd	00	01	11	10
00	0	0	1	1
01	0	0	0	0
11	1	1	0	0
10	1	1	0	0



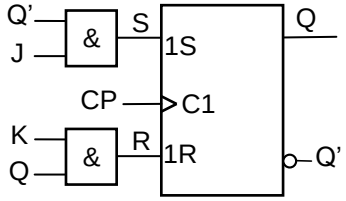
(4p)

3.

J	K	Q	Q^+	S	R
0	0	0	0	0	-
0	0	1	1	-	0
0	1	0	0	0	-
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	-	0
1	1	0	1	1	0
1	1	1	0	0	1

JK \ S Q	0	1
00	0	-
01	0	0
11	1	0
10	1	-

JK \ R Q	0	1
00	-	0
01	-	1
11	0	1
10	0	0



$S = JQ'$ $R = KQ$ **(5p)**

4. $5(A+1) + 3(B-1) = 3(A+B) + 2(A+1)$

CP	RTN	Styrsignaler (=1)
1	$B \rightarrow T$	OE_B, LD_T
2	$A + T \rightarrow R$	OE_A, f_3, f_1, LD_R
3	$2R \rightarrow R$, $R \rightarrow T$	OE_R, f_3, f_1, f_0, LD_R, LD_T
4	$R + T \rightarrow R$	OE_R, f_3, f_1, LD_R
5	$A \rightarrow T$	OE_A, LD_T
6	$R + T + 1 \rightarrow R$	OE_R, f_3, f_1, g_0, LD_R
7	$R + T + 1 \rightarrow R$	OE_R, f_3, f_1, g_0, LD_R
8	$R \rightarrow A$	OE_R, LD_A

(5p)

5. a)

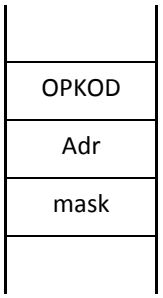
State	S-term	RTN-beskrivning	Styrsignaler (= 1)
Q_5	$Q_5 \cdot I_{xx}$	$PC \rightarrow MA$, $PC+1 \rightarrow PC$	OE_{PC}, LD_{MA}, IncPC
Q_6	$Q_6 \cdot I_{xx}$	$M \rightarrow MA$	MR, LD_{MA}
Q_7	$Q_7 \cdot I_{xx}$	$M_{1k} \rightarrow R$, Flags $\rightarrow$ CC	MR, f_1, f_0, LD_R, LD_{CC}
Q_8	$Q_7 \cdot I_{xx}$	$R \rightarrow M$, (Next Fetch)	OE_R, MW, NF

(1p)

b) 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: Alla bitar i minnesordet som adressen pekar på inverteras och laddas i R. Flaggorna uppdateras.
Q_8: Det inverterade minnesordet skrivs tillbaka på samma adress där det hämtades.
Detta är COM Adr **(2p)**

c)

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5 \cdot I_{FB}$	$PC \rightarrow MA$, $PC+1 \rightarrow PC$	OE_{PC}, LD_{MA}, IncPC
Q_6	$Q_6 \cdot I_{FB}$	$M \rightarrow T$	MR, LD_T
Q_7	$Q_7 \cdot I_{FB}$	$X+T \rightarrow R$	OE_X, f_3, f_1, LD_R
Q_8	$Q_8 \cdot I_{FB}$	$PC \rightarrow MA$, $PC+1 \rightarrow PC$	OE_{PC}, LD_{MA}, IncPC
Q_9	$Q_9 \cdot I_{FB}$	$M \rightarrow T$	MR, LD_T
Q_{10}	$Q_{10} \cdot I_{FB}$	$R \rightarrow MA$, $T_{1k} \rightarrow R$	OE_R, LD_{MA}, f_2, LD_R
Q_{11}	$Q_{11} \cdot I_{FB}$	$R \rightarrow T$	OE_R, LD_T
Q_{12}	$Q_{12} \cdot I_{FB}$	M *AND* T $\rightarrow$ R, Flags $\rightarrow$ CC	MR, f_2, f_1, LD_R, LD_{CC}
Q_{13}	$Q_{13} \cdot I_{FB}$	$R \rightarrow M$, Next Fetch	OE_R, MW, NF



(6p)

6.

a) Vid ”Reset” läser processorn dataordet på adress FF_{16} i minnet, som skall vara startadressen för det program man vill köra, och placerar det i PC. **(1p)**

b) FLEX-datorn använder minnesorienterad I/O. Fördelen med det är att vanliga instruktioner som läser eller skriver i minnet kan användas för I/O. **(2p)**

c) Bit w_5-w_0 måste vara 0, dvs W = --00 0000_2 = n*64, där n = 0, 1, 2 eller 3.
Hoppet utförs alltså om W = 0, 64, 128 eller 192. **(2p)**

d) BLT avser tal med tecken. För 8-bitars tal gäller då talintervallet [-128, 127]. COMA utför –W–1.
Hoppvillkoret blir $-W - 1 - 55_{16} < 0$ eller $-W - 1 - 85 < 0$, dvs $W > -86$ eller $-86 < W \leq +127$.

Eftersom negativa tal ersätts av 2-komplementet av motsvarande positiva tal så delar vi upp talintervallet i en positiv (inklusive 0) och en negativ del: $\underline{0 \leq W \leq +127}$

$-86 < W \leq -1$ (tolkning); $256 - 86 < W \leq 256 - 1$ (i verkligheten); $\underline{170 < W \leq 255}$ **(4p)**

e)

Adr	Data	~	Läge		
30	0F F6	4		LDAA #-10	
32	11 06	4	ALOOP	LDX #6	
34	10 05	4		LDAB #5	
36	45	4	XLOOP	DECB	
37	76 FE	6		LEAX $FE,X	
39	5E FB	5		BNE XLOOP	36 – 3B = FB
3B	41	4		INCA	
3C	5E F4	5		BNE ALOOP	32 – 3E = F4
3E	00	3		NOP	

(3p)

f) $\underline{N} = 4+(4+4+(4+6+5)*5+4+5)*10+3 = 7+(17+15*5)10 = 7+92*10 = \underline{927}$ klockpulser **(3p)**

7. (Först flödesplan) **(2p)**

```
BOS     EQU     $FA
INPORT  EQU     $FD
START   EQU     $10
        ORG     START
START   LDS     #BOS            Bottom of stack
LOOP    LDAA    INPORT          Läs indata (bitmönster)
        TFR     A,B             Spara en kopia av indata
TEST1   ANDA    #%01010011      Behåll bitarna 6, 4, 1 och 0, nolla övriga
        CMPA    #%01000011      Bit 4 = 0 och bit 6, 1 och 0 = 1?
        BNE     TEST2           Nej, nästa test
        JSR     SUB1            Ja
        BRA     LOOP            Nästa varv
TEST2   TFR     B,A             Återställ indata
        ANDA    #%00110010      Behåll bitarna 5, 4 och 1, nolla övriga
        CMPA    #%00010010      Bit 5 = 0 och bit 4 och 1 = 1?
        BNE     TEST3           Nej, nästa test
        JSR     SUB2            Ja
        BRA     LOOP            Nästa varv
TEST3   TFR     B,A             Återställ indata
        CMPA    #%00001100      $0C?
        BNE     NONE            Nej, ingen träff
        JSR     SUB3            Ja
        BRA     LOOP            Nästa varv
NONE    JSR     SUB4            Ja
        BRA     LOOP            Nästa varv
```

(5p)