Lösningsförslag tenta 2010-05-27

1. X = 100000101; Y = 000011001 (9 bitars ordlängd)

a) $[-2^{n-1}, +2^{n-1}-1] = [-2^{9-1}, +2^{9-1}-1] = [-256, +255]$ **(1p)**

b) $[0, 2^n-1] = [0, 2^9-1] = [0, 511]$ **(1p)**

c) $R = X+Y_{1k}+1$

```
9876543210       bitnummer
1000001111    1  carry
 100000101    X
+111100110    Y1k
 011101100 =  R
```

(1p)

d) N = r_8 = 0;
Z = 0 (R ≠ 0);
V = x_8*y_8*r_8' = 1*1*0' = 1*1*1 = 1; (Vid "subtraktion" är y_8 motsvarande bit i Y_{1k}.)
C = c_9' = 1' = 0 **(1p)**

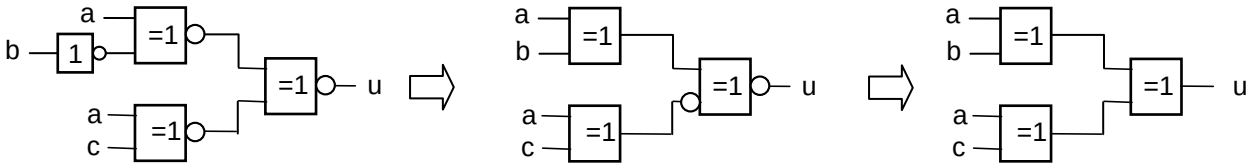
e) $\underline{R} = 011101100_2 = 0EC_{16} = 14*16 + 12 = 224 + 12 = \underline{236}$;
$\underline{X} = 100000101_2 = 105_{16} = 256 + 5 = \underline{261}$;
$\underline{Y} = 000011001_2 = 019_{16} = 16 + 9 = \underline{25}$; Korrekt resultat om C = 0. **(1p)**

f) $\underline{R} = \underline{236}$ (r_8 = 0, pos);
$\underline{X} = 2^9 - 251 = 512 - 251$ vilket motsvarar $\underline{-25}$ (x_8 = 1, neg);
$\underline{Y} = \underline{25}$ (y_8 = 0, pos); Korrekt resultat om V = 0. **(1p)**

g) -250 motsvarar 512 – 250 = 262 = 100000110_2 **(1p)**

h) $10^{15} = (10^3)^5 \approx (2^{10})^5 = 2^{50}$ Det skulle krävas 50-bitars mantissa. **(3p)**

i)

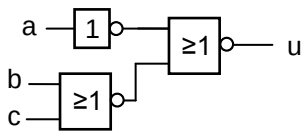


$\underline{u} = (a \oplus b) \oplus (a \oplus c) = a \oplus b \oplus a \oplus c = (a \oplus a) \oplus b \oplus c = (0) \oplus b \oplus c = \underline{b \oplus c}$ **(4p)**

2.

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

u = a(b + c)



(4p)

3.

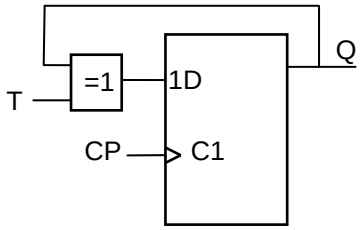
a)

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

b)

T \ D	0	1
0	0	1
1	1	0

$D = T'Q + TQ' = T \oplus Q$



(4p)

---

4. $4(A - 1) - 5B = 4(A - B - 1) - B$

CP	RTN	Styrsignaler (=1)
1	$B \rightarrow T$	OE_B, LD_T
2	$A - T - 1 \rightarrow R$	OE_A, f_3, f_2, LD_R
3	$2R \rightarrow R$	OE_R, f_3, f_1, f_0, LD_R
4	$2R \rightarrow R$	OE_R, f_3, f_1, f_0, LD_R
5	$R - T \rightarrow R$	OE_R, f_3, f_2, g_0, LD_R
6	$R \rightarrow A$	OE_R, LD_A

(5p)

5.

a)

State nr	S-term	RTN-beskrivning	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 → T	MR, LD_T.
Q_7	$Q_7 \cdot I_{xx}$	X + T→ R	OE_X, f_3, f_1, LD_R.
Q_8	$Q_8 \cdot I_{xx}$	R → MA	OE_R, LD_{MA}.
Q_9	$Q_9 \cdot I_{xx}$	M → MA	MR, LD_{MA}.
Q_{10}	$Q_{10} \cdot I_{xx}$	A → M, Next Fetch	OE_A, MW, NF.

(1p)

b) Q_5: Adressera ordet efter OP-koden i minnet. Öka PC till nästa OP-kod.
Q_6: Ordet efter OP-koden i minnet kopieras till register T.
Q_7: Innehållet i X-reg och det hämtade minnesordet adderas. Summan till R-registret.
Q_8: Summan i R-registret (en adress) kopieras till MA-registret.
Q_9: Dataordet på ”summaadressen” kopieras till MA-registret (en ny adress!).
Q_{10}: Innehållet i A-reg kopieras till minnet på den nya adressen.

Detta är STAA [n,X] **(2p)**

c)

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5 \cdot I_{xx}$	U = B, Flags → CC	OE_B, f_0, LD_{CC}
Q_6	$Q_6 \cdot I_{xx}$	PC → MA, T, PC + 1→ PC	OE_{PC}, LD_{MA}, LD_T, IncPC
Q_7	$Q_7 \cdot I_{xx}$ Z $Q_7 \cdot I_{xx}$ Z’	If Z = 1: M + T +1 → R else: Next Fetch	MR, f_3, f_1, g_0, LD_R NF
Q_8	$Q_8 \cdot I_{xx}$	R → PC, Next Fetch	OE_R, LD_{PC}, NF

(4p)

6.

a) Programräknarens (PC) funktion är att hålla reda på var i minnet programmet finns. PC pekar alltid på nästa instruktion eller del av instruktion. **(2p)**

b) När man har datavärden placerade i tabeller i minnet i på varandra följande adresser används X-registret för adressering av data. X-registret kan ökas eller minskas för att adressera nästa datavärde. **(2p)**

c) Innan ett villkorligt hopp utförs gör man en jämförelse mellan två tal som vi kan kalla α och β. Jämförelsen utförs som en subtraktion α – β som påverkar flaggorna. Instruktionerna BLO och BLT avser båda villkoret <, dvs. hopp utförs om α < β. Skillnaden är att BLO avser tal utan tecken medan BLT avser tal med tecken. För 8-bitars tal gäller att talområdet för tal utan tecken är [0, 255] medan det för tal med tecken är [-128, +127]. Detta innebär i praktiken att alla 8-bitars tal i intervallet [128, 255] tolkas som negativa och därför uppfattas som mindre än alla tal i intervallet [0, 127]. **(2p)**

d)

Adr	Data	~	Läge		
50	0F F6	4		LDAA #$F6	
52	11 FC	4	ALOOP	LDX #$FC	
54	E1	4	XLOOP	INX	
55	5E FD	5		BNE XLOOP	54 – 57 = FD
57	00	3		NOP	
58	41	4		INCA	
59	5E F7	5		BNE ALOOP	52 – 5B = F7
5B	00	3		NOP	

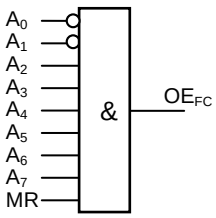
(3p)

e) N = 4 + (4 + (4 + 5)4 + 3 + 4 + 5)10 + 3 = 7 + (16 +36)10 = 527 **(3p)**

7.

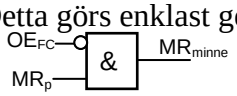
a) Inporten skall kopplas in vid läsning på adress FC_{16}. = 11111100_2

(Ring på ingången i figuren till höger betyder att signalen inverteras innan den når OCH-grinden.)



(3p)

b) Minnet får inte aktiveras vid läsning på adress FC_{16} eftersom data från inporten då skulle kollidera med data från minnet på databussen. MR-signalen från processorn, som vi kan kalla MR_p, måste därför hindras från att nå fram till minnet om adressen är FC_{16}. Detta görs enklast genom att blockera MR_p-signalen med en OCH-grind för just denna adress. Se figur.



(3p)

8.

```
CONV    PSHA              Spara register på stack
        PSHB
        PSHX

CLOOP   LDAA    ,X        Hämta data från textsträng
        TSTA              Strängslut?
        BEQ     CONEX     Ja, avsluta

        TFR     A,B       Nej, gör kopia av data
        ANDB    #$7F      Maska bit 7 i kopia
        CMPB    #'a'      a-z?
        BLO     NEXT      Nej, fortsätt
        CMPB    #'z'      a-z?
        BHI     NEXT      Nej, fortsätt

        ANDA    #%11011111 Ja, konvertera (nolla bit 5)
        STAA    ,X        Uppdatera sträng

NEXT    INX               Öka strängpekare
        BRA     CLOOP     Fortsätt

CONEX   PULX              Återställ register
        PULB
        PULA
        RTS
```

(8p)