

Lösningförslag tenta 2010-10-18

1. $X = 01010$; $Y = 11101$ (5 bitars ordlängd)

a) $R = X + Y$

543210	bitnummer
110000	0 carry
01010	X
+11101	Y
00111=	R

(1p)

b) $N = r_4 = 0$;
 $Z = 0$ ($R \neq 0$);
 $V = x_4 * y_4 * r_4' + x_4' * y_4' * r_4 = 0 * 1 * 0' + 0' * 1' * 0 = 0$; (Alltid 0 vid addition av tal med olika tecken!)
 $C = c_5 = 1$

(1p)

c) $R = 00111_2 = 07_{16} = \underline{7}$;
 $X = 01010_2 = 0A_{16} = \underline{10}$;
 $Y = 11101_2 = 1D_{16} = 16 + 13 = \underline{29}$;
 Korrekt resultat om $C = 0$.

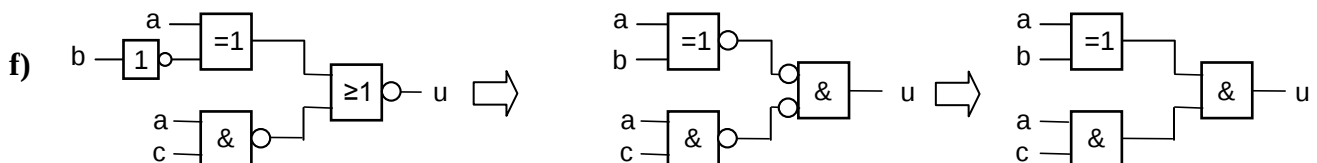
(1p)

d) ($r_4 = 0$, pos) $R = \underline{7}$
 ($x_4 = 0$, pos) $X = \underline{10}$
 ($y_4 = 1$, neg) $Y_{2k} = 2^5 - 29 = 32 - 29 = 3$ Y motsvarar $\underline{-3}$
 Korrekt resultat om $V = 0$.

(1p)

e) $560,375 = +1000110000.011_2 = +1.000110000011 * 2^9$;
 $s/c/f$; $s = 0$; $c = 9 + 127 = 136 = 128 + 8 = 10001000$; $f = 0001100000110...0$;
 Flyttalet blir: $0/10001000/0001100000110000000000_2 = 440C1800_{16}$

(2p)



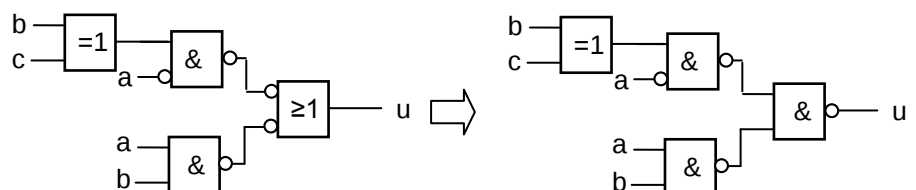
$$u = (a \oplus b)ac = (a'b + ab')ac = a'bac + ab'ac = 0 + ab'c = \underline{ab'c}$$

(3p)

2.

	bc			
	00	01	11	10
a	0	0	1	1
	1	0	0	1

$$u = ab + a'b'c + a'bc' = ab + a'(b'c + bc') = ab + a'(b \oplus c)$$

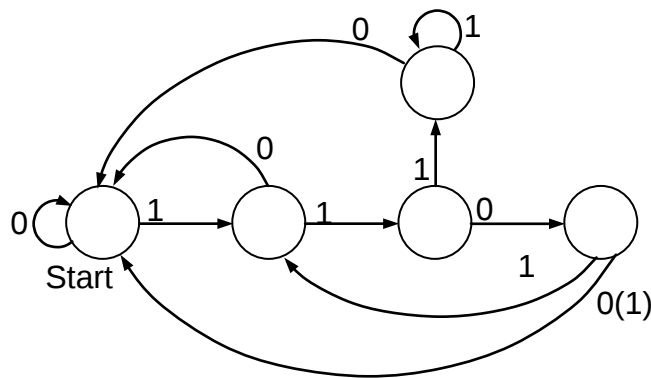


(Ring på ingångar i figuren ovan betyder att signalen inverteras innan den når grinden.)

(4p)

3.

a)



(Ej utsatta utsignaler = 0)

5 tillstånd ger minst 3 vippor
($2^2 < 5 \leq 2^3$)

(4p)

b)

q_2	q_1	q_0	q_2^+	q_1^+	q_0^+	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	1	1	0	-	1	-	1	-
0	0	1	0	0	0	0	-	0	-	-	1
0	1	0	-	-	-	-	-	-	-	-	-
0	1	1	1	1	1	1	-	-	0	-	0
1	0	0	0	0	1	-	1	0	-	1	-
1	0	1	-	-	-	-	-	-	-	-	-
1	1	0	1	0	0	-	0	-	1	0	-
1	1	1	1	1	0	-	0	-	0	-	1

$$J_2 = q_1$$

$$K_2 = q_1'$$

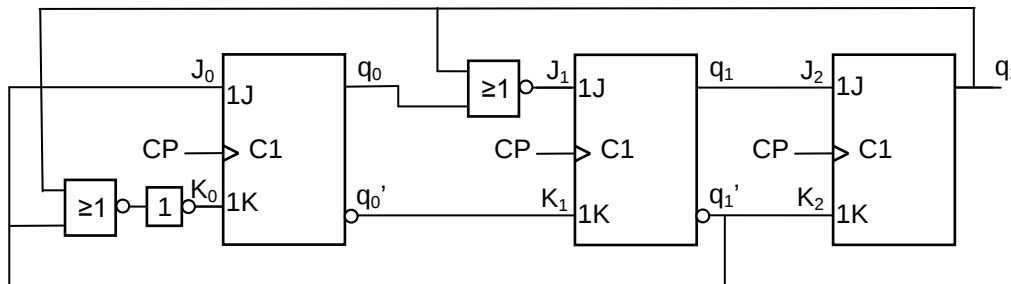
$$J_1 = q_2' q_0' = (q_2 + q_0)'$$

$$K_1 = q_0'$$

$$J_0 = q_1'$$

$$K_0 = q_2 + q_1'$$

J_2	$q_1 q_0$	K_2	$q_1 q_0$	J_1	$q_1 q_0$	K_1	$q_1 q_0$	J_0	$q_1 q_0$	K_0	$q_1 q_0$
0	00	0	00	0	00	1	00	1	00	1	00
0	01	-	01	1	01	-	01	-	01	-	01
0	10	-	10	-	10	-	10	-	10	-	10
1	11	-	11	-	11	-	11	-	11	-	11



(6p)

4. $7 \cdot A - 6 \cdot (B + 1) = 6 \cdot (A - B - 1) + A$

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	$2R \rightarrow R, R \rightarrow T$	$OE_R, f_3, f_1, f_0, LD_R, LD_T$
4	$R + T \rightarrow R$	OE_R, f_3, f_1, LD_R
5	$2R \rightarrow R$	$OE_R, f_3, f_1, f_0, LD_R$
6	$A \rightarrow T$	OE_A, LD_T
7	$R + T \rightarrow R$	OE_R, f_3, f_1, LD_R
8	$R \rightarrow B$	OE_R, LD_B

(5p)

5. a)

State	S-term	RTN-beskrivning	Aktiva 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	$Q_7' \cdot I_{xx}$	$2M \rightarrow R, Flaggor \rightarrow CC$	$MR, f_3, f_1, f_0, LD_R, LD_{CC}$
Q_8	$Q_8' \cdot I_{xx}$	$R \rightarrow M, NF$	OE_R, MW, NF

Instruktionen består av två ord, där det andra är en adress. Data på adressen dubblas och flaggorna påverkas. Detta är ASL Adr.

(2p)

b)

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5' \cdot I_{0F} \cdot Z$ $Q_5' \cdot I_{0F} \cdot Z'$ $Q_5' \cdot I_{0F}$	$PC \rightarrow MA, T, SP-1 \rightarrow SP$ Next Fetch $PC+1 \rightarrow PC$	$OE_{PC}, LD_{MA}, LD_T, DecSP$ NF IncPC
Q_6	$Q_6' \cdot I_{0F}$	$M + T + 1 \rightarrow R$	MR, f_3, f_1, g_0, LD_R
Q_7	$Q_7' \cdot I_{0F}$	$SP \rightarrow MA$	OE_{SP}, LD_{MA}
Q_8	$Q_8' \cdot I_{0F}$	$PC \rightarrow M$	OE_{PC}, MW
Q_9	$Q_8' \cdot I_{0F}$	$R \rightarrow PC, Next Fetch$	OE_R, LD_{PC}, NF

(5p)

6.

- a) $N \oplus V$ används istället för teckenflaggan N i logikuttryck för hoppvillkor då villkoret avser tal med tecken. Detta beror på att N-flaggan får fel värde vid overflow. Eftersom V-flaggan samtidigt ettställs kommer uttrycket $N \oplus V$ alltid att ha samma värde som den korrekta teckenbiten. (2p)
- b) Eftersom varje ORG-direktiv ger "location counter, LC" i assemblern ett nytt värde så är det risk att LC får ett värde som redan har använts. Vid laddning av program och data i minnet kan det då inträffa att tidigare laddade dataord skrivs över av senare laddade dataord. (2p)
- c) På stacken kan man lagra ett stort antal datavärden på ett mycket enkelt sätt. Stackpekaren SP innehåller alltid adressen till det översta ordet i stacken. Vid lagring av ett nytt datavärde (push) minskas först adressen i SP med ett och sedan skrivs datavärdet där SP "pekar". Vid läsning av översta värdet på stacken (pull eller pop) läser processorn datavärdet där stackpekaren pekar och därefter ökas adressen i SP med ett. Stacken används vid subrutinanrop för att lagra återhopp-adressen och för att tillfälligt lagra registervärden och variabler. (3p)

d)

Adr	Data	~	Läge		
60	11 72	4		LDX #TAB	
62	81 03	7		LDAA 3,X	
64	82 04	7	LOOP1	LDAB 4,X	
66	45	4	LOOP2	DECB	
67	5E FD	5		BNE LOOP2	66 – 69 = FD
69	41	4		INCA	
6A	5B F8	5		BMI LOOP1	64 – 6C = F8
6C	5A 0B	5		BRA NEXT	79 – 6E = 0B
6E	-- -- -- --	-		RMB 4	
72	00 FE 0A FC 14 F8 1E	-	TAB	FCB 0,-2,10,-4,20,-8,30	
79	00	3	NEXT	NOP	

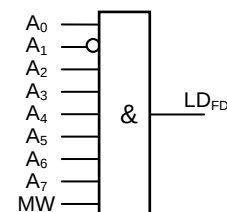
(3p)

e) $t = [4+7+(7+(4+5)*20+4+5)*4+5+3] \mu s = [19+(16+9*20)*4] \mu s = [19+784] \mu s = 803 \mu s$ (3p)

7.

- a) Utportens register skall laddas med data från databussen vid skrivning på adress $FD_{16} = 11111101_2$

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



(3p)

- b) Man kommer att skriva samma data i minnet på adress FD_{16} som man skriver på utporten. Har man ingen inport ansluten på samma adress kan man där läsa det man skrev på utporten, vilket ibland kan vara praktiskt. (2p)

- c) Man använder t ex instruktionen STAB \$FD (1p)

8.

ASCCNT	PSHA PSHX		Spara register på stack
	CLRB		Nollställ hexräknare
ALOOP	LDAA 1,X+ TSTA BEQ ASCEX		Hämta data från textsträng. Öka pekare Strängslut? Ja, avsluta
	ANDA #\$7F CMPA #'0' BLO ALOOP CMPA #'9' BLS COUNT		Maska paritetsbit (bit 7) Kolla undre gräns Ej hexsiffra. Testa nästa Decimal siffra? Ja, räkna
	CMPA #'A' BLO ALOOP		"Bokstavssiffra"? Ej hexsiffra. Testa nästa
	CMPA #'F' BHI ALOOP		"Bokstavssiffra"? Ej hexsiffra. Testa nästa
COUNT	INCB BRA ALOOP		Hexsiffra. Öka siffreräknare Testa nästa
ASCEX	PULX PULA RTS		Återställ register

(6p)