

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

e) $D = X+Y_{1k}+1$	
76543210	bitnummer
01101011	carry
0110101	X
+0010100	Y_{1k}
1001010	= D

(1p)

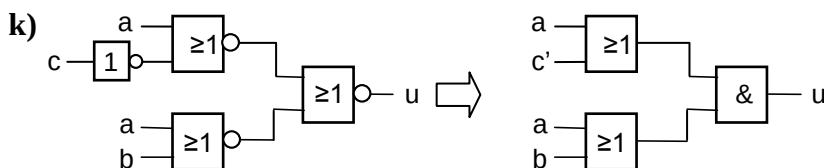
f) $\underline{N} = d_6 = 1$
 $\underline{Z} = 0$ ($D \neq 0$)
 $\underline{V} = x_6 * y_{6ik} * d_6' + x_6' * y_{6ik}' * d_6 = 0 * 0 * 1' + 0' * 0' * 1 = 1$
 $\underline{C} = c_7' = 0' = 1$ (1p)

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

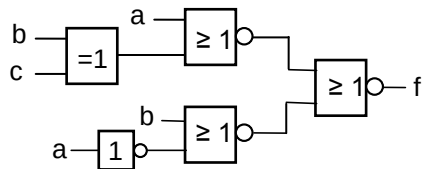
i) Antal booleska funktioner av n variabler $= 2^{2^n} = 2^{2^2} = 2^4 = \underline{16}$ (1p)

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



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

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

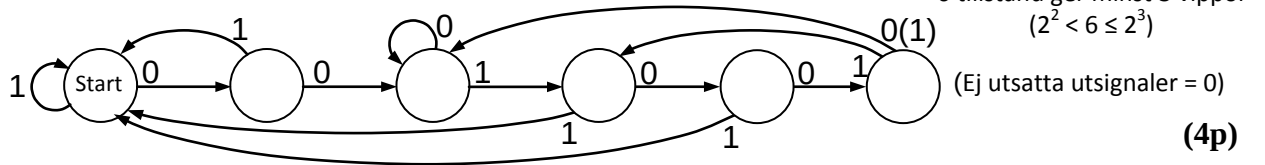


		bc			
		00	01	11	10
a	0	0	1	0	1
	1	0	0	1	1

(4p)

3.

a)



(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 $T = 1$.

		q ₁ q ₀			
		00	01	11	10
q ₂	0	0	0	0	1
	1	0	1	-	0

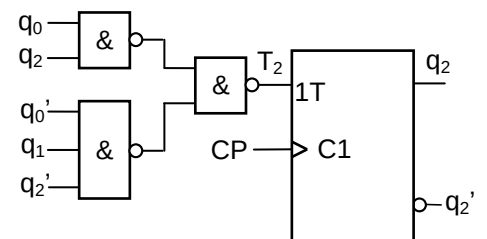
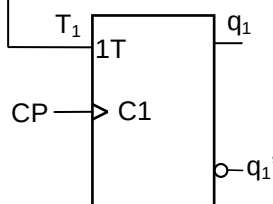
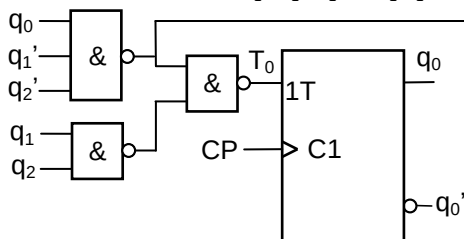
		q ₁ q ₀			
		00	01	11	10
q ₂	0	1	0	1	1
	1	1	1	-	1

		q ₁ q ₀			
		00	01	11	10
q ₂	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	$B \rightarrow T$	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	$2R \rightarrow R$	$OE_R, f_3, f_1, f_0, LD_R$
5	$R + T \rightarrow R$	OE_R, f_3, f_1, LD_R
6	$B \rightarrow T$	OE_B, LD_T
7	$R - T - 1 \rightarrow R$	OE_R, f_3, f_2, LD_R
8	$R - T - 1 \rightarrow R$	OE_R, f_3, f_2, LD_R
9	$R \rightarrow A$	OE_R, LD_A

(5p)

5. a)

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q ₅	Q ₅ ·I _{xx}	PC → MA, PC+1 → PC	OE _{PC} , LD _{MA} , IncPC
Q ₆	Q ₆ ·I _{xx}	M → MA	MR, LD _{MA}
Q ₇	Q ₇ ·I _{xx}	M → T	MR, LD _T
Q ₈	Q ₈ ·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 ·I _{xx}	R → A, (Next Fetch)	OE _R , LD _A , NF

Från början pekar PC på minnesordet efter OP-koden.

Q₅: Minnet adresseras med innehållet i PC, som också ökas med ett.

Q₆: Dataordet efter OP-koden som PC pekade på i minnet hämtas till MA. Det är alltså en adress.

Q₇: Dataordet som adressen pekar på i minnet hämtas till T.

Q₈: Dataordet från minnet och C subtraheras från A. Skillnaden laddas i R. Flaggorna uppdateras.

Q₉: Skillnaden laddas i A.

Detta är SBCA Adr (Subtraktion med borrow.)

(2p)

b)

State	S-term	RTN-beskrivning	Styrsignaler (= 1)
Q ₅	Q ₅ ·I _{F1} ·(N⊕V)'	PC → MA, T	OE _{PC} , LD _{MA} , LD _T
	Q ₅ ·I _{F1} ·(N⊕V)	PC+1 → PC, (Next Fetch)	IncPC, NF
Q ₆	Q ₆ ·I _{F1}	M+T+1 → R	MR, f ₃ , f ₁ , g ₀ , LD _R
Q ₇	Q ₇ ·I _{F1}	R → PC, (Next Fetch)	OE _R , LD _{PC} , NF

OPKOD
offset

(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 återhoppadress på stacken, så när processorn kommer till RTS sist i subrutinen och "tror" att den läser återhoppadressen, 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 - 50₁₆ > 0 eller -W - 1 - 80 > 0, dvs W < -81 eller -128 ≤ W < -81.

Eftersom negativa värden ersätts med 2-komplementet av motsvarande positiva värde blir det verkliga intervallet: 256 - 128 ≤ W < 256 - 81 eller 128 ≤ 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) $N = 4+7+(7+(4+3+5)*10+4+5)*2+5+3 = 19+(16+12*10)*2 = 291$ klockpulser (μs)

(3p)

7.

NIBINC	PSHX PSHA		Spara på stack
	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)