

Lösningsförslag tenta 2011-12-12 (v4 med reservation för eventuella fel!)

1. $X = 1100_2$; $Y = 0101_2$ (4 bitars ordlängd)

a) $[0, 2^n - 1] = [0, 2^4 - 1] = [0, 15]$ (1p)

b) $[-2^{n-1}, +2^{n-1} - 1] = [-2^{4-1}, +2^{4-1} - 1] = [-8, +7]$ (1p)

c) $S = X + Y$

$$\begin{array}{r} 43210 \text{ bitnummer} \\ 11000 \text{ carry} \\ 1100 \text{ X} \\ +0101 \text{ Y} \\ \hline 0001 = S \end{array}$$
 (1p)

d) $\underline{N} = s_3 = \underline{0}$
 $\underline{Z} = \underline{0} \text{ (} S \neq 0 \text{)}$
 $\underline{V} = x_3 * y_3 * s_3' + x_3' * y_3' * s_3 = 1 * 0 * 0' + 1' * 0' * 0 = \underline{0}$
 $\underline{C} = c_4 = \underline{1}$ (1p)

e) $D = X + Y_{1k} + 1$

$$\begin{array}{r} 43210 \text{ bitnummer} \\ 10001 \text{ carry (} c_0=1 \text{)} \\ 1100 \text{ X} \\ +1010 \text{ Y}_{1k} \\ \hline 0111 = D \end{array}$$
 (1p)

f) $\underline{N} = d_3 = \underline{0}$
 $\underline{Z} = \underline{0} \text{ (} D \neq 0 \text{)}$
 $\underline{V} = x_3 * y_{3ik} * d_3' + x_3' * y_{3ik}' * d_3 = 1 * 1 * 0' + 1' * 1' * 0 = \underline{1}$
 $\underline{C} = c_4' = 1' = \underline{0}$ (1p)

g) $\underline{X} = 1100_2 = C_{16} = \underline{12}$
 $\underline{Y} = 0101_2 = \underline{5}$
 $\underline{S} = 0001_2 = \underline{1}$ Resultatet S är felaktigt eftersom $C = 1$.
 $\underline{D} = 0111_2 = \underline{7}$ Resultatet D är korrekt eftersom $C = 0$. (1p)

h) ($x_3 = 1$, neg) $X_{2k} = 2^4 - 12 = 16 - 12 = 4$ $\underline{X}$ motsvarar $\underline{-4}$
 ($y_3 = 0$, pos) $\underline{Y} = \underline{5}$
 ($s_3 = 0$, pos) $\underline{S} = 0001_2 = \underline{1}$ Resultatet S är korrekt eftersom $V = 0$.
 ($d_3 = 0$, pos) $\underline{D} = \underline{7}$. Resultatet S är felaktigt eftersom $V = 1$. (1p)

i) 4 decimala sifferpositioner krävs (teckensiffra + 3 siffror). $X = 0351_{10}$; $-Y = 0594_{10}$.
 Y motsvaras av $(-Y)_{10k} = 9999 - 0594 + 1 = 9405 + 1 = 9406$

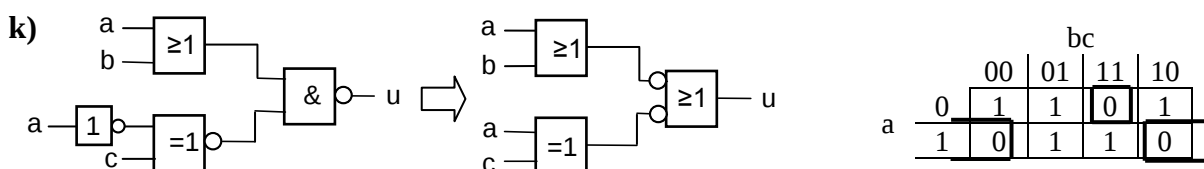
Princip: $S = X + Y$

3210	siffernummer
0000	0 carry
0351	X
+9406	Y
<u>9757</u>	$\underline{S}$ (Resultatet är negativt då $s_3=9$.)

För att kontrollera resultatet kan man 10-komplementera det för att se beloppet.

$\underline{S}_{10k} = S_{9k} + 1 = 9999 - 9757 + 1 = 0242 + 1 = \underline{243}$ (Summan är alltså -243) (2p)

j) Fraction är den normaliserade mantissan med inledande ettan struken. Normaliseringen medför att den inledande biten alltid är en etta och den behöver därför inte tas med i det lagrade talet. (1p)

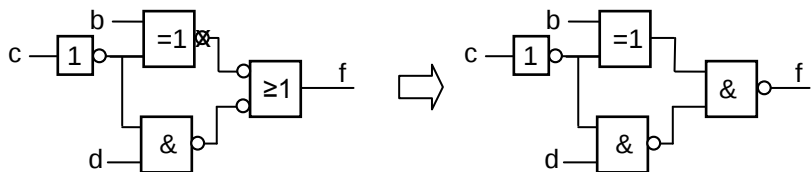


$\underline{u} = (a+b)' + (a \oplus c)' = a'b' + ac + a'c' = (\underline{a + b' + c'})'(\underline{a' + c})$ (enligt k-diagrammet ovan) (4p)

2.

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

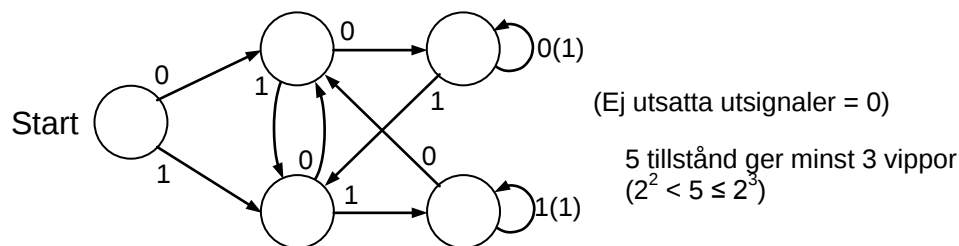
$$f = bc' + b'c + c'd = (b \oplus c) + c'd$$



(5p)

3.

a)



(5p)

b)

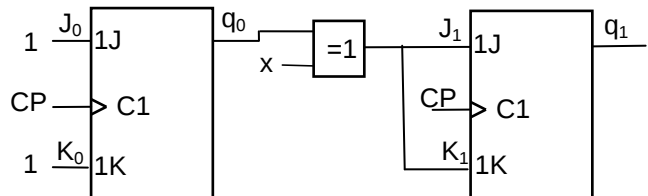
x	q ₁	q ₀	q ₁ ⁺	q ₀ ⁺	J ₁	K ₁	J ₀	K ₀
0	0	0	0	1	0	-	1	-
0	0	1	1	0	1	-	-	1
0	1	0	1	1	-	0	1	-
0	1	1	0	0	-	1	-	1
1	0	0	1	1	1	-	1	-
1	0	1	0	0	0	-	-	1
1	1	0	0	1	-	1	1	-
1	1	1	1	0	-	0	-	1

$$J_1 = x'q_0 + xq_0' = x \oplus q_0 = K_1$$

$$J_0 = K_0 = 1$$

J ₁	q ₁ q ₀			
	00	01	11	10
x	0	0	1	-
	1	1	0	-

K ₁	q ₁ q ₀			
	00	01	11	10
x	0	-	-	1
	1	-	0	1



(5p)

4.

CP	Styrsignaler (=1)	RTN	Reg A	Reg B	Reg T	Reg R
1	OE _A , LD _T	A → T	05	09	?	?
2	OE _B , f ₃ , f ₂ , LD _R	B - T - 1 → R	05	09	05	?
3	OE _R , f ₃ , f ₁ , f ₀ , LD _R , LD _T	2R → R, R → T	05	09	05	03
4	OE _R , f ₃ , f ₁ , LD _R	R + T → R	05	09	03	06
5	OE _A , LD _T	A → T	05	09	03	09
6	OE _R , f ₃ , f ₂ , g ₀ , LD _R	R - T → R	05	09	05	09
7	OE _R , f ₃ , f ₂ , g ₀ , LD _R	R - T → R	05	09	05	04
8	OE _R , LD _B	R → B	05	09	05	FF
9	?	?	05	FF	05	FF

(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→T	MR, LD _T
Q ₇	Q ₇ ·I _{xx}	X+T→R	OE _X , f ₃ , f ₁ , LD _R
Q ₈	Q ₈ ·I _{xx}	R→MA	OE _R , LD _{MA}
Q ₉	Q ₉ ·I _{xx}	M→MA	MR, LD _{MA}
Q ₁₀	Q ₁₀ ·I _{xx}	M→B, Next Fetch	MR, LD _B , NF

Instruktionen består av två ord. Ordet efter OP-koden adderas till värdet i X-reg och summan används som adress till minnet. Dataordet på adressen läses och placeras i MA-reg. Ny läsning på den nya adressen görs och data från minnet placeras i B-reg. Detta är LDAB [n,X] (2p)

b)

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q ₅	Q ₅ ·I _{FA}	X → MA, X+1→ R, SP-1→SP	OE _X , LD _{MA} , f ₃ , g ₀ , LD _R , DecSP
Q ₆	Q ₆ ·I _{FA}	R → X	OE _R , LD _X
Q ₇	Q ₇ ·I _{FA}	M → R	MR, f ₀ , LD _R
Q ₈	Q ₈ ·I _{FA}	SP → MA	OE _{SP} , LD _{MA}
Q ₉	Q ₁₀ ·I _{FA}	R → M, Next Fetch	OE _R , MW, NF

(4p)

6.

a) BHI och BGT medför båda hopp om skillnaden > 0.

Vid BHI tolkas talen som "tal utan tecken", som har talintervallet [0, FF₁₆]:

85₁₆ > 67₁₆ eller 85₁₆ - 67₁₆ > 0 medför alltså att hoppet utförs.

Vid BGT tolkas talen som "tal med tecken", som har talintervallet [-80₁₆, 7F₁₆]:

85₁₆ tolkas som det negativa talet - [85₁₆]_{2k} = -7B₁₆. Eftersom -7B₁₆ < 67₁₆ utförs inte hoppet. (2p)

b) Funktionen hos signalen NF är att ladda tillståndräknaren med värdet 3 som motsvarar första tillståndet i FETCH. Om sista tillståndet i EXECUTE inte aktiverar NF så fortsätter tillståndsräknaren räkna tills den varvar och börjar om med RESET följt av FETCH, dvs processorn startar om på det aktuella programmets startadress. (2p)

c) Under FETCH-fasen hämtas nästa OP-kod från minnet och laddas i instruktionsregistret I. Dessutom ökas programräknaren PC med ett. Eller uttryckt med RTN:

PC → MA, PC + 1 → PC

M → I.

(2p)

d)

Adr	Data	~	Läge		
80	0F F6	4		LDAA #-10	
82	7C	5*11	LOOP	LDAB 1,X+	
83	39	5*11		NEGB	
84	1A 0F	6*11		ANDB #15	
86	14 FE	5*11		STAB \$FE	
88	41	4*11		INCA	
89	67 F7	5*11		BLE LOOP	82 - 8B = F7
8B	00	3		NOP	

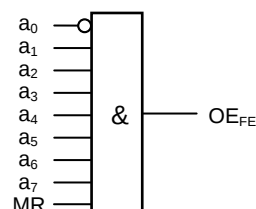
(3p)

e) $t = [4 + (5 + 5 + 6 + 5 + 4 + 5) * 11 + 3] \mu s = [7 + 30 * 11] \mu s = 337 \mu s$

(2p)

f) Inportens "three-state driver" skall aktiveras (OE_{FE}) vid läsning på adress FE₁₆ = 11111110₂

(Ring på ingången till OCH-grinden betyder att signalen inverteras innan den når grinden. Signalerna a₀-a₇ är adressbitarna från adressbussen.)



(2p)

7. NIBADD	PSHA		Spara på stack
	PSHB		
	PSHX		
	STAB	TABCNT	Init räknare för antal dataord
	TST	TABCNT	Färdigt?
	BEQ	NIBEX	Ja, förbered återhopp
LOOP	LDAA	,X	Nej, hämta nästa tabellvärde
	TFR	A,B	Kopia till B
	ANDB	#\$0F	Maska bort hög nibble i B
	STAB	NCOPY	Spara låg nibble
	ROLA		Högerjustera hög nibble
	ROLA		
	ROLA		
	ROLA		
	ROLA		
	ANDA	#\$0F	Nollställ bit 4-7
	ADDA	NCOPY	Utför addition med låg nibble
	STAA	1,X+	Uppdatera tabell och adressera nästa tabellvärde
	DEC	TABCNT	Minska dataordsräknare
	BNE	LOOP	Färdigt? Nej!
NIBEX	PULX		Hämta från stack
	PULB		
	PULA		
	RTS		
TABCNT	RMB	1	Räknare för antal dataord
NCOPY	RMB	1	Kopia av låg nibble

(6p)