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
43210   bitnummer
11000   carry
 1100   X
+0101   Y
 0001 = S
```
(1p)

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

(1p)

e)
```
D = X+Y1k+1
43210   bitnummer
10001   carry (c0=1)
 1100   X
+1010   Y1k
 0111 = D
```
(1p)

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

(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)**

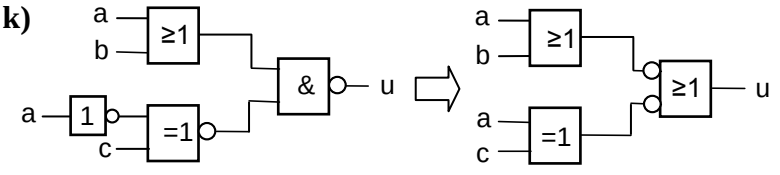
h) ($x_3 = 1$, neg) $X_{2k} = 2^4 – 12 = 16 – 12 = 4$ $\underline{X}$ motsvarar $\underline{–4}$
($y_3 = 0$, pos) $\underline{Y} = \underline{5}$
($s_3 = 0$, pos) $\underline{S} = 0001_2 = \underline{1}$ Resultatet S är korrekt eftersom V = 0.
($d_3 = 0$, pos) $\underline{D} = \underline{7}$. Resultatet 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
 9757 = S (Resultatet är negativt då s3=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)**

k)



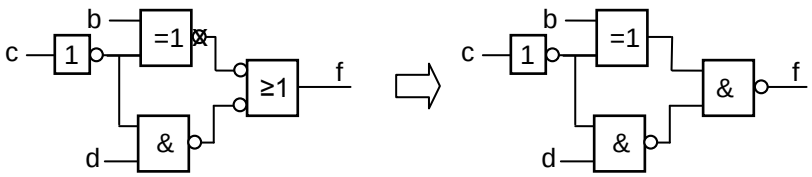
a \ bc	00	01	11	10
0	1	1	0	1
1	0	1	1	0

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

2.

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

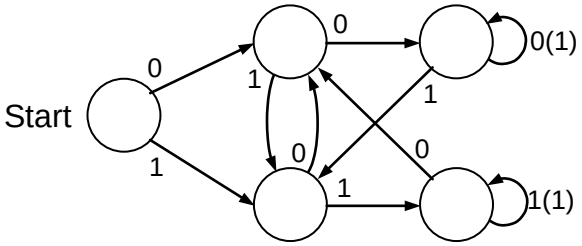
$\underline{f} = bc' + b'c + c'd = (b \oplus c) + c'd$



(5p)

3.

a)



(Ej utsatta utsignaler = 0)

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

(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

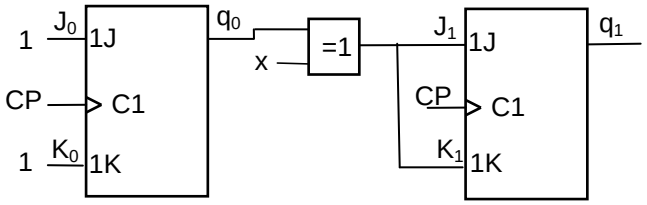
$J_1 = x'q_0 + xq_0' = x \oplus q_0 = K_1$
$J_0 = K_0 = 1$

J_1

x \ q_1q_0	00	01	11	10
0	0	1	-	-
1	1	0	-	-

K_1

x \ q_1q_0	00	01	11	10
0	-	-	1	0
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	$Q_5 \cdot I_{xx}$	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}	$Q_{10} \cdot I_{xx}$	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	$Q_5 \cdot I_{FA}$	X → MA, X+1→ R, SP-1→SP	OE_X, LD_{MA}, f_3, g_0, LD_R, DecSP
Q_6	$Q_6 \cdot I_{FA}$	R → X	OE_R, LD_X
Q_7	$Q_7 \cdot I_{FA}$	M →R	MR, f_0, LD_R
Q_8	$Q_8 \cdot I_{FA}$	SP → MA	OE_{SP}, LD_{MA}
Q_9	$Q_{10} \cdot I_{FA}$	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, FF_{16}]:
$85_{16} > 67_{16}$ eller $85_{16} - 67_{16} > 0$ medför alltså att hoppet utförs.

Vid BGT tolkas talen som ”tal med tecken”, som har talintervallet [-80_{16}, $7F_{16}$]:
85_{16} tolkas som det negativa talet $- [85_{16}]_{2k} = -7B_{16}$. Eftersom $-7B_{16} < 67_{16}$ utförs inte hoppet. **(2p)**

b) Funktionen hos signalen NF är att ladda tillståndsräknaren 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:

$PC \rightarrow MA,\ PC + 1 \rightarrow PC$
$M \rightarrow I.$ **(2p)**

d)

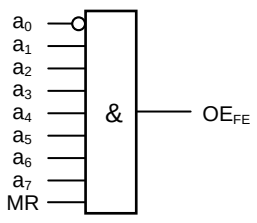
Adr	Data	~	Läge		
80	0F F6	4		LDAA #-10	
82	7C	5*11	LOOP	LDAB 1,X+	
83	39	5*11		NEGB	
84	1A 0F	6*11		ANDB #15	
86	14 FE	5*11		STAB $FE	
88	41	4*11		INCA	
89	67 F7	5*11		BLE LOOP	82 – 8B = F7
8B	00	3		NOP	

(3p)

e) $\underline{t} = [4+(5+5+6+5+4+5)*11+3]\ \mu s = [7+30*11]\ \mu s = \underline{337\mu s}$ **(2p)**

f) Inportens ”three-state driver” skall aktiveras (OE_{FE}) vid läsning på adress $FE_{16} = 11111110_2$

(Ring på ingången till OCH-grinden betyder att signalen inverteras innan den når grinden. Signalerna a_0-a_7 är adressbitarna från adressbussen.)



(2p)

7.

```
NIBADD  PSHA
        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)