Lösningsförslag tenta 2011-01-12 (Med reservation för eventuella fel)

1. X = 1001010101; Y = 0101101010 (10 bitars ordlängd)

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

```
          109876543210      bitnummer
           10000101011   1  carry
            1001010101   X
          + 1010010101   Y1k
            0011101011 = R
```
(1p)

b) $\underline{N} = r_9 = \underline{0}$;
$\underline{Z} = \underline{0}$ ($R \neq 0$);
$\underline{V} = x_9 * y_9 * r_9' + x_9' * y_9' * r_9 = \underline{1*1*0'} + 1'*1'*0 = \underline{1}$; (Vid ”subtraktion” är y_9 motsvarande bit i Y_{1k}.)
$\underline{C} = c_{10}' = 1' = \underline{0}$ **(1p)**

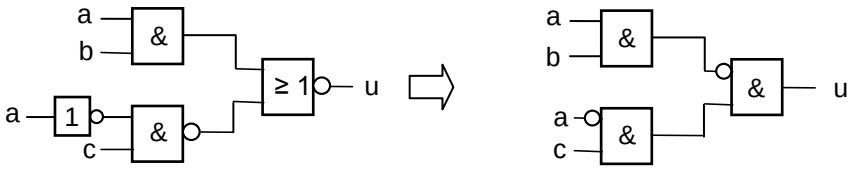
c) $\underline{R} = 0011101011_2 = 0EB_{16} = 14*16 + 11 = 224 + 11 = \underline{235}$;
$\underline{X} = 1001010101_2 = 255_{16} = = 2*256 + 5*16 + 5 = 512 + 80 + 5 = \underline{597}$;
$\underline{Y} = 0101101010_2 = 16A_{16} = 1*256 + 6*16 + 10 = 256 + 96 + 10 = \underline{362}$;
Resultatet R är korrekt (C = 0). Korrekt resultat om C = 0. **(1p)**

d) ($r_9 = 0$, pos) $\underline{R} = \underline{235}$
($x_9 = 1$, neg) $X_{2k} = 2^{10} - 597 = 1024 - 597 = 427$ $\underline{X}$ motsvarar $\underline{-427}$
($y_9 = 0$, pos) $\underline{Y} = \underline{362}$
Resultatet R är felaktigt (V = 1). Korrekt resultat om V = 0. **(1p)**

e) $X \geq Y$ om $X - Y \geq 0$, som gäller om C = 0 eller Z = 1, dvs $\underline{f} = \underline{C' + Z}$, efter subtraktionen. **(2p)**

f) Det högsta värdet ($11111111_2 = 255$) är reserverat för att representera oändligheten och används när absolutbeloppet av talet som skall representeras är större än maxbeloppet för 32-bitars flyttal. Det lägsta värdet ($00000000_2 = 0$) är (tillsammans med f = 0) reserverat för att representera talet 0 och används när absolutbeloppet av talet som skall representeras är mindre än minbeloppet för 32-bitars flyttal. **(2p)**

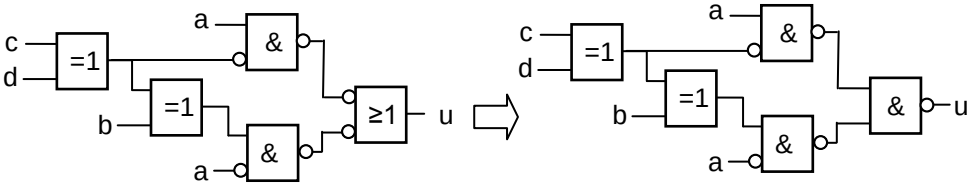
g)



$\underline{u} = (ab)'a'c = (a' + b')a'c = a'a'c + a'b'c = a'c + a'b'c = a'c(1 + b') = \underline{a'c}$ **(2p)**

2.

ab \ cd	00	01	11	10
00	0	(1)	0	(1)
01	(1)	0	(1)	0
11	[1]	0	[1]	0
10	[1]	0	[1]	0



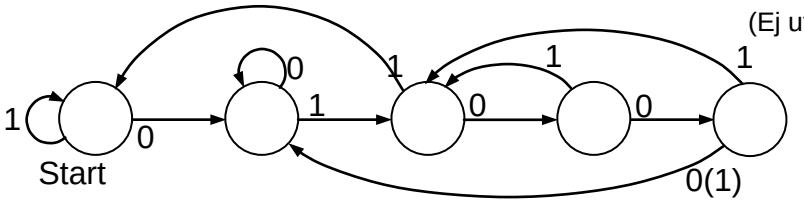
(Ring på ingångar i figuren ovan betyder att signalen inverteras innan den når grinden.)

$\underline{u} = ac'd' + acd + a'b'c'd + a'b'cd' + a'bc'd' + a'bcd = a(c'd' + cd) + a'(b'c'd + b'cd' + bc'd' + bcd) =$
$= a(c \oplus d)' + a'[b'(c'd + cd') + b(c'd' + cd)] = a(c \oplus d)' + a'[b'(c \oplus d) + b(c \oplus d)'] =$
$= a(c \oplus d)' + a'[b \oplus (c \oplus d)]$

(6p)

3.

a)



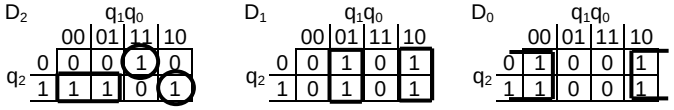
5 tillstånd ger minst 3 vippor
($2^2 < 5 \leq 2^3$)

(4p)

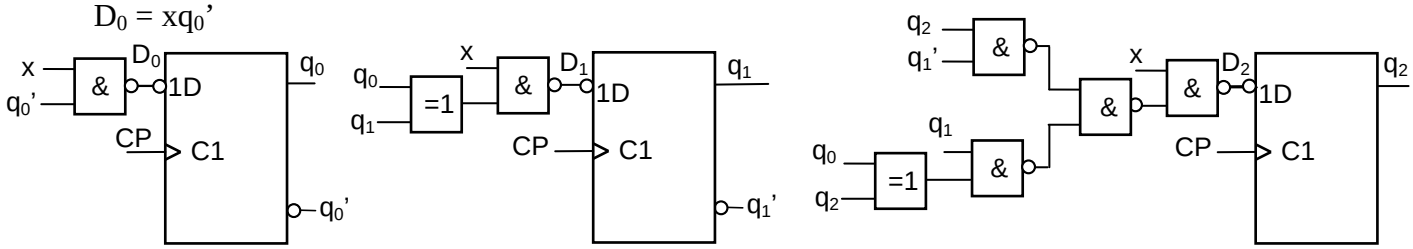
b)

x	q_2	q_1	q_0	q_2^+	q_1^+	q_0^+	D_2	D_1	D_0
1	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	0
1	0	1	0	0	1	1	0	1	1
1	0	1	1	1	0	0	1	0	0
1	1	0	0	1	0	1	1	0	1
1	1	0	1	1	1	0	1	1	0
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	0	0	0	0	0

För D-vippor gäller att $q^+ = D$
För x = 0 gäller att $D_2D_1D_0 = 000$



$D_2 = x(q_2q_1' + q_2'q_1q_0 + q_2q_1q_0') = x[q_2q_1' + q_1(q_2'q_0 + q_2q_0')] = x[q_2q_1' + q_1(q_2 \oplus q_0)]$
$D_1 = x(q_1'q_0 + q_1q_0') = x(q_1 \oplus q_0)$
$D_0 = xq_0'$



(Ring på D-ingångarna i figuren ovan betyder att signalen inverteras innan den når vippan.)

(6p)

4. $A - 7 \cdot (B + 1) = A - [2 \cdot 3(B + 1) + (B + 1)]$

CP	RTN	Styrsignaler (=1)
1	$B + 1 \rightarrow R$	OE_B, f_3, g_0, LD_R
2	$2R \rightarrow R$, $R \rightarrow T$	OE_R, f_3, f_1, f_0, LD_R, LD_T
3	$R + T \rightarrow R$	OE_R, f_3, f_1, 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_1, LD_R
6	$R \rightarrow T$	OE_R, LD_T
7	$A - T \rightarrow R$	OE_A, f_3, f_2, g_0, LD_R
8	$R \rightarrow A$	OE_R, LD_A

(5p)

5. a)

State	S-term	RTN-beskrivning	Styrsignaler (=1)
Q_5	$Q_5 \cdot I_{xx}$	$PC \rightarrow MA$, $PC+1 \rightarrow PC$	OE_{PC}, LD_{MA}, IncPC
Q_6	$Q_6 \cdot I_{xx}$	$M \rightarrow T$	MR, LD_T
Q_7	$Q_7 \cdot I_{xx}$	CC *OR* $T \rightarrow R$	OE_{CC}, f_2, f_0, LD_R
Q_8	$Q_8 \cdot I_{xx}$	$R \rightarrow$ CC, Next Fetch	OE_R, LD_{CC}, g_2, NF

Instruktionen består av två ord. Andra ordet läses och placeras i T-registret. Innehållen i CC- och T-registret OR:as och resultatet placeras i R-registret. Resultatet placeras sedan i CC-registret.
Detta är ORCC #Data.

(2p)

b)

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5 \cdot I_{F8}$	$PC \rightarrow MA$, $PC+1 \rightarrow PC$	OE_{PC}, LD_{MA}, IncPC
Q_6	$Q_6 \cdot I_{F8}$	$M \rightarrow MA$	MR, LD_{MA}
Q_7	$Q_7 \cdot I_{F8}$	$M \rightarrow R$	MR, f_0, LD_R
Q_8	$Q_8 \cdot I_{F8}$	$PC \rightarrow MA$, $PC+1 \rightarrow PC$	OE_{PC}, LD_{MA}, IncPC
Q_9	$Q_9 \cdot I_{F8}$	$M \rightarrow MA$	MR, LD_{MA}
Q_{10}	$Q_{10} \cdot I_{F8}$	$R \rightarrow M$, Next Fetch	OE_R, MW, NF

(4p)

6.

a) Under processorns FETCH-fas hämtar den nästa OP-kod från minnet och placerar den i instruktions-registret IR. FETCH-fasen inleds med att PC-värdet placeras i MA-registret. PC-värdet ökas också med ett. Sedan läser processorn minnesinnehållet som MA pekar på och placerar det i IR. **(1p)**

b) Bägge instruktionerna utför ett hopp om villkoret ”större än” är uppfyllt. Före det villkorliga hoppet förutsätts att två tal har jämförts och därmed påverkat flaggorna. För BHI tolkas de jämförda talen som ”tal utan tecken” med hoppvillkoret C’ + Z’, dvs skillnaden är större än noll för tal utan tecken. För BGT tolkas de jämförda talen som ”tal med tecken” med hoppvillkoret (N ⊕ V)’ + Z’, dvs skillnaden är större än noll för tal med tecken. **(2p)**

c) C-värdet används av ALU’n när tal som är bredare än 8 bitar (ALU’ns bredd) skall adderas eftersom additionen då måste utföras i två eller flera steg och den minnessiffra som uppstår i föregående steg skall ingå i nästa addition. C’-värdet används på motsvarande sätt vid ”subtraktion med borrow” (SBC-instruktionerna) som utförs genom addition av 1-komplement. C’ skall användas på grund av att ALU’n inverterar C-värdet vid subtraktion och den andra inverteringen därför återställer det ursprungliga C-värdet. C-flaggan används också när ROL-instruktionerna utförs. **(2p)**

d)

Adr	Data	~	Läge			
30	10 03	4		LDAB	#%11	
32	11 42	4		LDX	#DATA	
34	81 04	7		LDAA	4,X	
36	88	6	LOOPA	LDX	B,X	
37	E2	4	LOOPX	DEX		
38	5C FD	5		BPL	LOOPX	37 – 3A = FD
3A	11 42	4		LDX	#DATA	
3C	2A 01	6		ADDA	#1	
3E	5E F6	5		BNE	LOOPA	36 – 40 = F6
40	5A 0C	5		BRA	NEXT	4E – 42 = 0C
42	00 01 F9 04 FE 05 EC	-	DATA	FCB	0,1,-7,4,-2,5,-20	
49	-- -- -- -- --	-	TAB	RMB	5	
4E	00	3	NEXT	NOP		

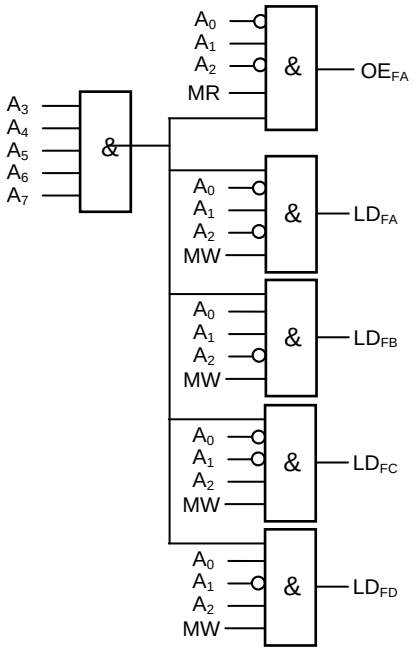
(3p)

e) $\underline{t}$ = [4+4+7+(6+(4+5)*5+4+6+5)*2+5+3] µs = [23+(21+45)*2] µs = [23 + 132] µs = $\underline{155\mu s}$ **(3p)**

7.

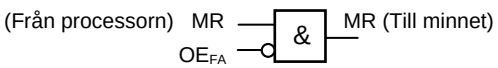
a) Inportens ”three-state driver” skall aktivera (OE_{FA}) vid läsning på adress FA_{16} och utportarnas register skall laddas (LD_{FA} - LD_{FD}) med data från databussen vid skrivning på adresserna FA_{16} - FD_{16} = - 11111010_2 - 11111101_2.

(Ring på ingången till OCH-grindarnai figuren till höger betyder att signalen inverteras innan den når resp. grind. Signalerna A_0-A_7 är adressbitarna från adressbussen.)



(3p)

b) Vid läsning från inporten kommer indata att kollidera med data från minnets adress FA_{16}. Man måste därför förhindra att minnet lägger ut sin data på databussen vid läsning på inportsadressen FA_{16}. Nedan visas hur detta kan fixas.



(3p)

8. (Först flödesplan)

```
BOS     EQU   $FC
INPORT  EQU   $FD
START   EQU   $20

        ORG   START

START   LDS   #BOS          Bottom of stack

LOOP    LDAA  INPORT        Läs indata (bitmönster)
        TFR   A,B           Spara en kopia av indata

TEST1   ANDA  #%11010000    Behåll bitarna 7, 6 och 4, nolla övriga
        CMPA  #%00010000    Bit 7 och 6 = 0 och bit 4 = 1?
        BNE   TEST2         Nej, nästa test

        JSR   SEND          Ja
        BRA   LOOP          Nästa varv

TEST2   TFR   B,A           Återställ indata
        ANDA  #%10100000    Behåll bitarna 7 och 5, nolla övriga
        CMPA  #%10100000    Bit 7 och 5 = 1?
        BNE   TEST3         Nej, nästa test

        JSR   REC           Ja
        BRA   LOOP          Nästa varv

TEST3   TFR   B,A           Återställ indata
        ANDA  #$0F          Behåll bitarna 3-0, nolla övriga
        CMPA  #5            < 5?
        BHS   LOOP          Nej, nästa varv

        JSR   WAIT          Ja
        BRA   LOOP          Nästa varv
```

(6p)