Lösningsförslag tenta 2009-01-14

1. X = 100101; Y = 001010 (6 bitars ordlängd)

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

```
6543210      bitnummer
1001011   1  carry
 100101   X
+110101   Y1k
 011011 = R
```

(1p)

b) N = r_5 = 0;
Z = 0 (R ≠ 0);
V = x_5*y_5*r_5' = 1*1*0' = 1*1*1 = 1; (Vid "subtraktion" är y_5 motsvarande bit i Y_{1k}.)
C = c_6' = 1' = 0 **(1p)**

c) $\underline{R} = 011011_2 = 1B_{16} = 16 + 11 = \underline{27}$
$\underline{X} = 100101_2 = 25_{16} = 32 + 5 = \underline{37}$
$\underline{Y} = 001010_2 = 0A_{16} = \underline{10}$
Korrekt resultat om C = 0. **(1p)**

d) (r_5 = 0, pos) $\underline{R} = \underline{27}$
(x_5 = 1, neg) $X_{2k} = 2^6 - 37 = 64 - 37 = 27$; $\underline{X}$ motsvarar $\underline{-27}$
(y_5 = 0, pos) $\underline{Y} = \underline{10}$
Korrekt resultat om V = 0. **(1p)**

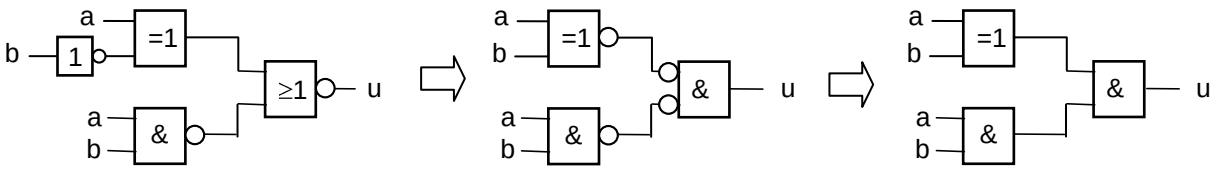
e) 5 decimala sifferpositioner krävs (4 siffror + teckensiffra). $A = 05872_{10}$; $B = 07594_{10;}$ $B_{9k} = 92415$
Princip: $D = A+B_{9k}+1$

```
43210      siffernummer
01001    1 carry
05872    A
+92405   B9k
98278 =  D (Resultatet är negativt då d4=9.)
```

För att kontrollera resultatet kan man 10-komplementera det för att se beloppet.
$D_{10k} = D_{9k} + 1 = 01721 + 1 = 01722$ **(3p)**

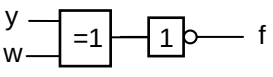
f) $10\ 1110\ 1111\ 0111_2 = 2EF7_{16}$ **(1p)**

g)



u = (a'b + ab')ab = a'bab + ab'ab = 0 + 0 = 0 **(3p)**

2. f(x,y,z,w) = (x'+ y + w')(y' + z'+ w)(y' + z + w)(x + y +w')
f = (y + w')(y' + w) = yw + y'w' = (y ⊕ w)'

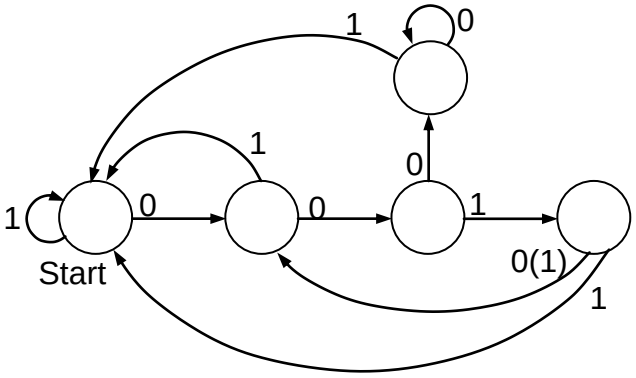


xy \ zw	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	0
10	1	0	0	1

(4p)

3.

a)



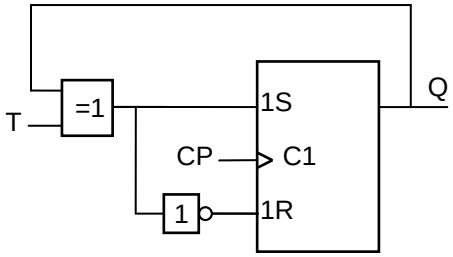
(Ej utsatta utsignaler = 0)

(4p)

b)

T	Q	Q^+	S	R
0	0	0	0	‡
0	1	1	‡	0
1	0	1	1	0
1	1	0	0	1

$S = R' = T'Q + TQ' = T \oplus Q$



(4p)

4. $4(A + 1) – 5(B + 1) = 4A + 4 – 5B – 5 = 4(A – B) – 1 – B$

CP	RTN	Styrsignaler (=1)
1	$B \rightarrow T$	OE_B, LD_T
2	$A – T \rightarrow R$	OE_A, f_3, f_2, g_0, LD_R
3	$2R \rightarrow R$	OE_R, f_3, f_1, f_0, LD_R
4	$2R \rightarrow R$	OE_R, f_3, f_1, f_0, LD_R
5	$R – T – 1 \rightarrow R$	OE_R, f_3, f_2, LD_R
6	$R \rightarrow B$	OE_R, LD_B

(5p)

5.

a)

State	Summaterm	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5 \cdot I_{xx}$	$PC \rightarrow MA$	OE_{PC}, LD_{MA}
Q_6	$Q_6 \cdot I_{xx}$	$M \rightarrow MA$	MR, LD_{MA}
Q_7	$Q_7 \cdot I_{xx}$	$M \rightarrow MA$	MR, LD_{MA}
Q_8	$Q_8 \cdot I_{xx}$	$M \rightarrow PC$, NF	MR, LD_{PC}, NF

(1p)

b) Q_5: Minnesadressen efter OP-koden placeras i MA.
Q_6: Dataordet efter OP-koden i minnet placeras i MA. (Alltså en adress.)
Q_7: Dataordet som pekas ut av adressen efter OP-koden i minnet placeras i MA. (Ny adress!)
Q_8: Dataordet på den nya adressen placeras i PC. (Alltså ett hopp!)
Detta skulle innebära ett indirekt-indirekt hopp, dvs JMP [[Adress]] **(2p)**

c)

State	Summaterm	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5 \cdot I_{F0}$	$B + 1 \rightarrow R$, Flags $\rightarrow$ CC	OE_B, f_3, g_0, LD_R, LD_{CC}
Q_6	$Q_6 \cdot I_{F0}$	$R \rightarrow B$	OE_R, LD_B
Q_7	$Q_7 \cdot I_{F0}$ Z $Q_7 \cdot I_{F0}$ Z'	If Z = 1: $PC + 1 \rightarrow PC$, NF else: $PC \rightarrow MA$, T	IncPC, NF OE_{PC}, LD_{MA}, LD_T
Q_8	$Q_8 \cdot I_{F0}$	$M + T + 1 \rightarrow R$	MR, f_3, f_1, g_0, LD_R
Q_9	$Q_9 \cdot I_{F0}$	$R \rightarrow PC$, NF	OE_R, LD_{PC}, NF

(5p)

6.

a) Q_3: PC → MA, PC + 1 → PC Kopiera PC-värdet till MA. Öka PC.
Q_4: M → IR Läs där ”gamla” PC pekade (OP-kod) och placera den i IR. **(2p)**

b) JSR placerar adressen till nästa instruktion (dvs. återhoppsadressen till huvudprogrammet) överst på stacken innan hoppet utförs. JMP påverkar ej stacken. **(2p)**

c) FLEX-datorn använder minnesorienterad I/O. Fördelen med det är att vanliga instruktioner som läser eller skriver i minnet kan användas för I/O. I fallet separatadresserad I/O krävs speciella I/O-intruktioner. **(2p)**

d) Före ett villkorligt hopp antas att flaggorna har påverkats av en subtraktion mellan två tal som skall jämföras. Storleksrelationen mellan talen kan då bestämmas med hjälp av flaggornas värden. För villkorliga hopp där hoppvillkoret avser tal utan tecken används flaggorna C och Z. C visar om lånesiffra uppstod och Z visar om talen var lika. För motsvarande hopp som avser tal med tecken används istället N ⊕ V som visar om skillnaden blev negativ och Z som visar om talen var lika. Uttrycket N ⊕ V har den goda egenskapen att det visar skillnadens korrekta tecken även om overflow uppstod vid subtraktionen. **(4p)**

e)

Adr	Data	~	Läge	
00	11 0F	4		LDX #DATA
02	81 03	7		LDAA 3,X
04	7A	4	WAIT1	LDAB ,X
05	45	4	WAIT2	DECB
06	00	3		NOP
07	5E FC	5		BNE WAIT2
09	00	3		NOP
0A	44	4		DECA
0B	5E F7	5		BNE WAIT1
0D	5A 09	5		BRA P2
0F	00 01 02 04 08 10 20 40 80	-	DATA	FCB 0,1,2,4,8,16,32,64,128
18	0B FD	5	P2	LDAA $FD

05 – 09 = FC; 04 – 0D = F7; 18 – 0F = 09 **(3p)**

f) t = [4 + 7 + (4 + (4 + 3 + 5)*256 + 3 + 4 + 5)*4 + 5 + 5] μs = [21 + (16 + 12*256)*4] μs = 12373 μs **(3p)**

7.

```
KEY     EQU   $C0

KEYCMP  PSHB              Spara register på stack
        PSHX

        CLRA              Nollställ felräknare

        PSHX              Spara pekare till ”User string”
        LDX   #KEY        Initiera adressen till nyckelsträngen
        STX   KEYADR      Uppdatera nyckelpekare
        PULX              Återställ pekaren till ”User string”

KLOOP   LDAB  1,X+        Hämta data från ”User string”
        TSTB              Kolla om strängslut
        BEQ   KEXIT       Ja, strängslut

        ANDB  #$7F        Nej, fortsätt med att jämföra med nyckelsträng

        PSHX              Spara pekare till ”User string”
        LDX   KEYADR      Hämta nyckelpekare
        INC   KEYADR      Uppdatera nyckelpekare
        CMPB  ,X          Jämför ”User data” med nyckeldata
        PULX              Återställ pekare till ”User string”

        BEQ   KLOOP       Data matchar, testa nästa

        INCA              Data stämmer ej med nyckel. Öka felräknare
        BRA   KLOOP       Testa nästa

KEXIT   PULX              Återställ register
        PULB
        RTS

KEYADR  RMB   1           Variabel för pekare till nyckelsträng
```

(8p)