Lösningsförslag tenta 2009-05-27

1.

a) $[-2^{6-1},+2^{6-1}-1] = [-32, +31]$ **(1p)**

b) $[0,2^6-1] = [0, 63]$ **(1p)**

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

```
6543210      bitnummer
1011111   1  carry
 100111   X
+101110   Y1k
 010110 = R
```

(1p)

d) $N = r_5 = 0$;
$Z = 0$ ($R \neq 0$);
$V = x_5*y_5*r_5' = 1*1*0' = 1*1*1 = 1$; (Vid ”subtraktion” är y_5 motsvarande bit i Y_{1k}.)
$C = c_6' = 1' = 0$ **(1p)**

e) $\underline{R} = 010110_2 = 16_{16} = 16 + 6 = \underline{22}$; $\underline{X} = 100111_2 = 27_{16} = 32 + 7 = \underline{39}$; $\underline{Y} = 010001_2 = 16 + 1_{16} = \underline{17}$;
Korrekt resultat om C = 0. **(1p)**

f) $\underline{R} = \underline{22}$ ($r_5 = 0$, pos); $\underline{X} = 2^6 - 39 = 64 - 39 = \underline{-25}$ ($x_5 = 1$, neg); $\underline{Y} = \underline{17}$ ($y_5 = 0$, pos);
Korrekt resultat om V = 0. **(1p)**

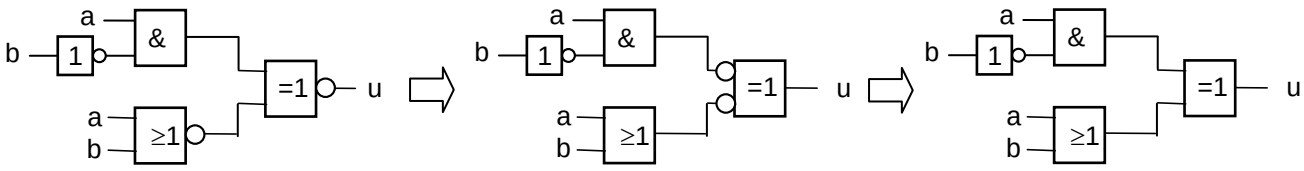
g) -32 motsvarar $64 - 32 = 32 = 100000_2$ **(1p)**

h) N_{flyt} = 45834200H = 0/100 0101 1/000 0011 0100 0010 0000 0000
s c f

s = 0 (+)
c = 139; exp = 139 – 127 = 12;
m = 1.f = 1.0000 0110 1000 0100 0000 000

$\underline{N_2} = \underline{1.0000\ 0110\ 1000\ 0100\ 0000\ 000*2^{12}} = 4096 + 64 + 32 + 8 + .25 = \underline{4200,25}$ **(2p)**

i)

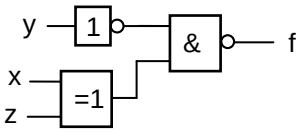


$\underline{u} = (ab')\oplus(a+b) = (ab')'(a+b) + ab'(a+b)' = (a'+b)(a+b) + \cancel{ab'a'b'} = \cancel{a'a} + a'b + ab + b =$
$= (a'+ a)b + b = \underline{b}$ **(2p)**

2. $f(x,y,z) = xz + yz' + x'y'z' + x'yz$

$f = y + x'z' + xz = y + (x\oplus z)' = (y'(x\oplus z))'$

x \ yz	00	01	11	10
0	1	0	1	1
1	0	1	1	1

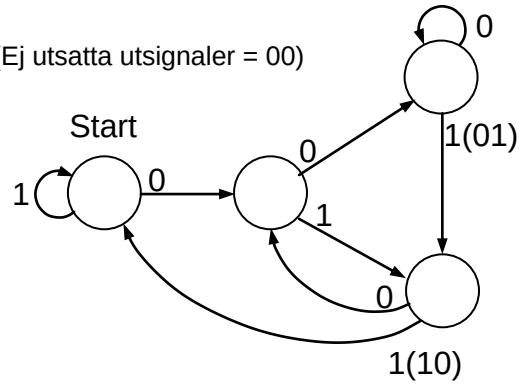


(4p)

3.

a)

(Ej utsatta utsignaler = 00)



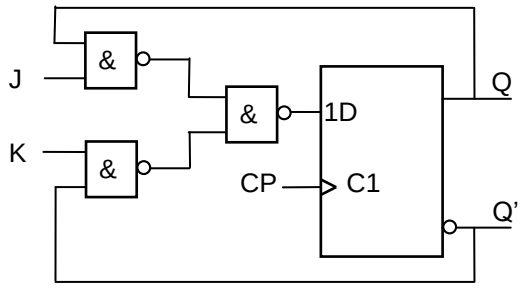
(6p)

b)

J	K	Q	Q^+=D
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$D=Q^+$	Q=0	Q=1
JK=00	0	1
JK=01	0	0
JK=11	1	0
JK=10	1	1

$D = JQ' + K'Q$



(4p)

---

4. $3A - 5(B + 1) = \rightarrow B$

CP	RTN	Styrsignaler (=1)
1	$B + 1 \rightarrow R$	OE_B, f_3, g_0, LD_R
2	$R \rightarrow T$, B	OE_R, LD_T, LD_B
3	$A - T \rightarrow R$	OE_A, f_3, f_2, g_0, LD_R
4	$2R \rightarrow R$, $R \rightarrow T$	OE_R, f_3, f_1, f_0, LD_R, LD_T
5	$R + T \rightarrow R$	OE_R, f_3, f_1, LD_R
6	$B \rightarrow T$	OE_B, LD_T
7	$R - T \rightarrow R$	OE_R, f_3, f_2, g_0, LD_R
8	$R - T \rightarrow R$	OE_R, f_3, f_2, g_0, LD_R
9	$R \rightarrow B$	OE_R, LD_B

(5p)

5.

a)

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5 \cdot I_{xx}$	$X \rightarrow MA$	OE_X, LD_{MA}
Q_6	$Q_6 \cdot I_{xx}$	$M \rightarrow T$	MR, LD_T
Q_7	$Q_7 \cdot I_{xx}$	B OR $T \rightarrow R$, Flags $\rightarrow$ CC	OE_B, f_2, f_0, LD_R, LD_{CC}
Q_8	$Q_8 \cdot I_{xx}$	$R \rightarrow B$, Next Fetch	OE_R, LD_B, NF

(1p)

b) Q_5: Värdet i X-reg (en adress) kopieras till MA-reg.
Q_6: Innehållet i minnesadressen som X pekar på placeras i T-reg.
Q_7: Innehållet i B-reg OR:as med det lästa minnesinnehållet i T-reg och placeras i R-reg.
Q_8: Resultatet av OR-operationen kopieras till B-reg.

Detta är ORAB ,X **(2p)**

c)

State	S-term	RTN-beskrivning	Aktiva styrsignaler (=1)
Q_5	$Q_5 \cdot I_{xx}$	$X + 1 \rightarrow R$, Flags $\rightarrow$ CC	OE_X, f_3, g_0, LD_R, LD_{CC}
Q_6	$Q_6 \cdot I_{xx}$	$R \rightarrow X$	OE_R, LD_X
Q_7	$Q_7 \cdot I_{xx}$ Z' $Q_7 \cdot I_{xx}$ Z	If Z = 0: PC $\rightarrow$ MA, T else: PC + $1 \rightarrow$ PC, Next Fetch	OE_{PC}, LD_{MA}, LD_T IncPC, NF
Q_8	$Q_8 \cdot I_{xx}$	$M + T + 1 \rightarrow R$	MR, f_3, f_1, g_0, LD_R
Q_9	$Q_9 \cdot I_{xx}$	$R \rightarrow$ PC, Next Fetch	OE_R, LD_{PC}, NF

(5p)

6.

a) Stackpekaren SP innehåller minnesadressen till det översta värdet på stacken. Man bestämmer stackens begynnelseadress, BOS ”bottom of stack”, genom att ladda detta adressvärde i SP. Vid skrivning av ett nytt värde på stacken minskas först adressvärdet i SP med ett (räkna ned ett steg). Vid läsning av översta värdet på stacken ökas adressvärdet i SP med ett efter läsningen (räkna upp ett steg). Eftersom SP är en upp/ner-räknare behöver man ej använda ALU:n för detta. **(2p)**

b) När man har datavärden placerade i tabeller i minnet i på varandra följande adresser används X-registret för adressering av data. X-registret kan ökas eller minskas för att adressera nästa datavärde. **(2p)**

c) Tillstånden bestäms av en 4-bitars räknare med $2^4 = 16$ olika tillstånd. RESET och FETCH kräver 3 + 2 = 5 tillstånd. 16 – 5 = 11 tillstånd är därför tillgängliga för EXECUTE. Svar: 11 tillstånd. **(2p)**

d) Subrutinen multiplicerar innehållen i A- och B-registren. Produkten (16 bitar) returneras i samma register som A:B. $10_{16}*11_{16} = 110_{16}$, dvs. A = 01_{16} och B = 10_{16}. **(3p)**

e)

Adr	Data	~	Läge	
30	49 4D	5	SUBR	CLR VAR1
32	49 4E	5		CLR VAR2
34	52	3		TSTA
35	5D 11	5		BEQ EXIT
37	53	3		TSTB
38	5D 0E	5		BEQ EXIT
3A	14 4E	5		STAB VAR2
3C	48	4		CLRB
3D	29 4E	7	LOOP	ADDB VAR2
3F	62 02	5		BCC GOON
41	43 4D	6		INC VAR1
43	44	4	GOON	DECA
44	5E F7	5		BNE LOOP
46	14 4E	5		STAB VAR2
48	0B 4D	5	EXIT	LDAA VAR1
4A	0C 4E	5		LDAB VAR2
4C	6A	4		RTS
4D	-		VAR1	RMB 1
4E	-		VAR2	RMB 1

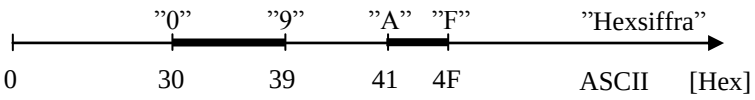
48 – 37 = 11; 48 – 3A = 0E; 43 – 41 = 02; 3D – 46 = F7 **(3p)**

f) t = [5 + 5 + 3 + 5 + 3 + 5 + 5 + 4 + 16(7 + 5 + 4 + 5) + **<u>6</u>** +5 + 5 + 5 + 4] μs = [35 + 16*21 + 25] μs = [60 + 336] μs = 396 μs

(Understrukna sexan i fet stil i uttrycket ovan är tiden för INC VAR1, som ingår i slingan LOOP men som bara utförs en gång. Produkten blir ju 0110_{16}. Den höga byten av produkten är alltså 01_{16} och den har bara ökats en gång.)

(3p)

7.




```
ASCBIN  ANDA  #$7F      Maska bort bit 7

        SUBA  #$30      Justera för siffror 0-9
        BLO   ASCERR    Ej siffra. Fel

        CMPA  #9        Siffra 0-9?
        BLS   ASCOK     Ja, återhopp

        SUBA  #7        Nej, justera för siffra A-F
        CMPA  #10       Siffra A-F?
        BLO   ASCERR    Nej. Fel

        CMPA  #15       Siffra A-F?
        BLS   ASCOK     Ja, återhopp

ASCERR  LDAA  #$FF      Ej siffra, sätt felflagga
ASCOK   RTS
```

(8p)