Lösningsförslag tenta 2013-12-16

(Version 4 med reservation för eventuella fel. Uppdaterad 131219.)

1. $X = 1010\ 0101_2$; $Y = 0101\ 1011_2$ (8 bitars ordlängd)

a) $[0, 2^n–1] = [0, 2^8–1] = [0, 255]$ **(1p)**

b) $[–2^{n-1}, +2^{n-1}-1] = [–2^{8-1}, +2^{8-1}–1] = [–128, +127]$ **(1p)**

c)
```
S = X+Y
876543210   bitnummer
111111110   carry
 10100101   X
+01011011   Y
 00000000 = S
```
(1p)

d) $\underline{N} = s_7 = \underline{0}$
$\underline{Z} = \underline{1}$ (S = 0)
$\underline{V} = x_7*y_7*s_7' + x_7'*y_7'*s_7 = 1*0*0' + 1'*0'*0 = \underline{0}$
$\underline{C} = c_8 = \underline{1}$

NZVC = 0101 **(1p)**

e)
```
D = X+Y1k+1
876543210   bitnummer
101001011   carry
 10100101   X
+10100100   Y1k
 01001010 = D
```
(1p)

f) $\underline{N} = d_7 = \underline{0}$
$\underline{Z} = \underline{0}$ (D ≠ 0)
$\underline{V} = x_7*y_{7_{1k}}*d_7' + x_7'*y_{7_{1k}}'*d_7 = \underline{1*1*0'} + 1'*1'*0 = \underline{1}$
$\underline{C} = c_8' = 1' = \underline{0}$

NZVC = 0010 **(1p)**

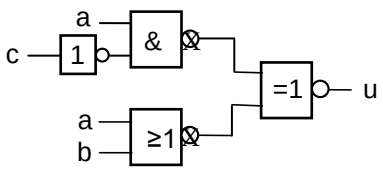
g) $\underline{X} = 1010\ 0101_2 = A5_{16} = 10*16 + 5 = \underline{165}$
$\underline{Y} = 0101\ 1011_2 = 5B_{16} = 5*16 + 11 = \underline{91}$
$\underline{S} = 0000\ 0000_2 = \underline{0}$ Resultatet S är felaktigt eftersom C = 1.
$\underline{D} = 0100\ 1010_2 = 4A_{16} = 4*16 + 10 = \underline{74}$ Resultatet D är korrekt eftersom C = 0. **(1p)**

h) (x_7 = 1, neg) $X_{2k} = 2^8 – 165 = 256 – 165 = 91$ $\underline{X}$ motsvarar $\underline{–91}$
(y_7 = 0, pos) $Y = \underline{91}$
(s_7 = 0, pos) $\underline{S} = 0000\ 0000_2 = \underline{0}$ Resultatet S är korrekt eftersom V = 0.
(d_7 = 0, pos) D = $\underline{74}$ Fel eftersom V = 1. **(1p)**

i) $N = 125{,}125_{10} = 1111101.001 = 1.111101001*2^6$ Format: s/c/f
s = 0 (pos); $c = exp + 2^{8-1} - 1 = 6 + 127 = 133 = 10000101_2$; f = mantissa - 1
$N_{flyt} = 0/100\ 0010\ 1/111\ 1010\ 0100\ 0000\ 0000\ 0000_2 = 42FA4000_{16}$ **(2p)**

2.

a)



$\underline{u}$ = [(ac')⊕(a+b)]' =
= (ac')(a+b) + (ac')'(a+b)' =
= ac'+abc'+ (a'+c)a'b' =
= ac'+~~abc~~'+ a'b'+~~a'b'c~~ =
= ac'+ a'b' = $\underline{(a + b')(a'+ c')}$ ($\underline{\text{Nr 4}}$)

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

(3p)

b)

ab \ cd	00	01	11	10
00	0	(1)	0	(1)
01	(1)	0	(-)	0
11	-	1	-	0
10	0	-	1	0

NAND-gindar ↔ "ettor"
$\underline{f} = \underline{a'(b⊕c⊕d) + ad}$ ($\underline{\text{Nr 5}}$)

[Nr 8 ger också korrekt funktion, men utgår från "nollor" och är därför inte lämpligt för NAND-grindar. **(2p)]**

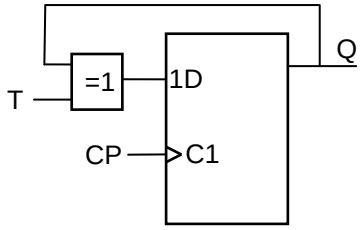
(4p)

3.

a)

T	Q	Q^+	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

T \ Q	0	1
0	0	1
1	1	0

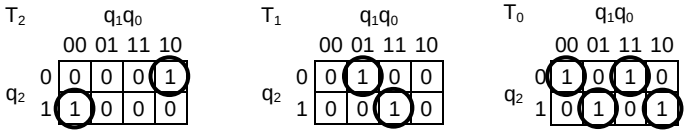


$D = T'Q + TQ' = T \oplus Q$ **(2p)**

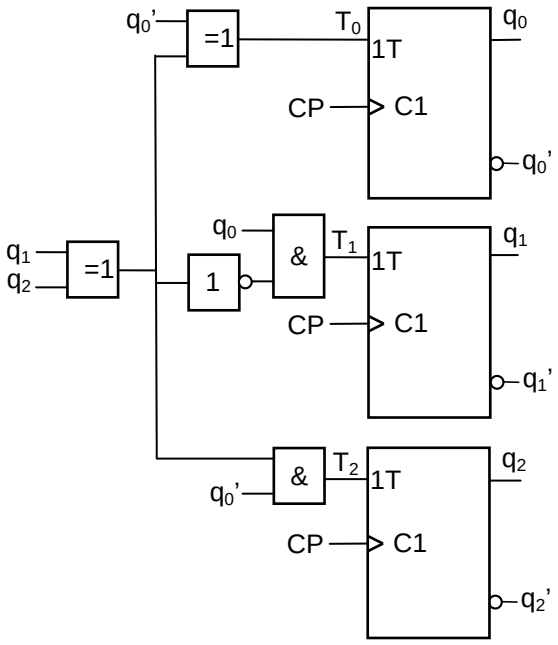
b)

För T-vippor gäller att $q^+ = q$ för T = 0 och $q^+ = q'$ för T = 1.

q_2	q_1	q_0	T_2	T_1	T_0	q_2^+	q_1^+	q_0^+
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	1	0	0	1	1	0
0	1	1	0	0	1	0	1	0
1	0	0	1	0	0	0	0	0
1	0	1	0	0	1	1	0	0
1	1	0	0	0	1	1	1	1
1	1	1	0	1	0	1	0	1



$T_2 = q_2'q_1q_0' + q_2q_1'q_0' = (q_2 \oplus q_1)q_0'$
$T_1 = q_2'q_1'q_0 + q_2q_1q_0 = (q_2 \oplus q_1)'q_0$
$T_0 = (q_2 \oplus q_1 \oplus q_0)' = (q_2 \oplus q_1) \oplus q_0'$



(6p)

4.

CP	Styrsignaler (=1)	RTN	A(8)	B(8)	T(8)	R(8)
1	OE_B, LD_T	B → T	C8	64	?	?
2	OE_A, f_3, f_2, LD_R	A – T – 1 → R	C8	64	64	?
3	OE_R, f_3, f_1, f_0, LD_R, LD_T	2R → R, R → T	C8	64	64	63
4	OE_R, f_3, f_1, LD_R	R + T → R	C8	64	63	C6
5	OE_R, LD_B	R → B	C8	64	63	29
6	?	?	C8	29	63	29

(4p)

5. a) FLISP:

State	Summaterm	RTN-beskrivning	Styrsignaler
Q_4	$Q_4 \cdot I_{xx}$	M(PC) → T, PC+1 → PC	MR, LD_T, INC_{PC}
Q_5	$Q_5 \cdot I_{xx}$	X + T → R	OE_X, f_3, f_1, f_0, LD_R
Q_6	$Q_6 \cdot I_{xx}$	R → X, (New Fetch)	OE_R, LD_X, NF

FLEX:

State	Summaterm	RTN-beskrivning	Styrsignaler
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(MA) → 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 → X, (Next Fetch)	OE_R, LD_X, NF

Detta är: LEAX n,X **(2p)**

b) FLISP:

State	S-term	RTN-beskrivning	Styrsignaler (=1)
Q_4	$Q_4 \cdot I_{DF}$	M(PC) → TA, PC+1 → PC	MR, LD_{TA}, INC_{PC}
Q_5	$Q_5 \cdot I_{DF}$	M(X) → T	MR, g_{13}, g_{12}, LD_T
Q_6	$Q_6 \cdot I_{DF}$	M(TA) + T → R, 0 → T, ALU(NZVC) → CC	MR, g_{14}, f_3, f_1, f_0, LD_R, LD_{CC}, CLR_T
Q_7	$Q_7 \cdot I_{DF}$	R → M(X)	OE_R, g_{13}, g_{12}, MW
Q_8	$Q_8 \cdot I_{DF}$	X + 1 → R	OE_X, f_3, f_0, LD_R
Q_9	$Q_9 \cdot I_{DF}$	R → X, (New Fetch)	OE_R, LD_X, NF

FLEX:

State	S-term	RTN-beskrivning	Styrsignaler (=1)
Q_5	$Q_5 \cdot I_{DF}$	PC → MA, PC+1 → PC	OE_{PC}, LD_{MA}, IncPC
Q_6	$Q_6 \cdot I_{DF}$	M(MA) → MA	MR, LD_{MA}
Q_7	$Q_7 \cdot I_{DF}$	M(MA) → T	MR, LD_T
Q_8	$Q_8 \cdot I_{DF}$	X → MA	OE_X, LD_{MA}
Q_9	$Q_9 \cdot I_{DF}$	M(MA) + T→ R, Flags → CC	MR, f_3, f_1, LD_R, LD_{CC}
Q_A	$Q_A \cdot I_{DF}$	R → M(MA)	OE_R, MW
Q_B	$Q_B \cdot I_{DF}$	X + 1 → R	OE_X, f_3, g_0, LD_R
Q_C	$Q_C \cdot I_{DF}$	R → X, (Next Fetch)	OE_R, LD_X, NF

(4p)

6.

a) 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 efter 15 och börjar om med RESET följt av FETCH, dvs processorn startar om på det aktuella programmets startadress. **(2p)**

b) Vi förutsätter att båda talen har ordlängden 8 bitar och därmed talområdet $[-128_{10},+127_{10}]$, eftersom hoppvillkoret för BPL avser tal med tecken.

Det villkorliga hoppet BPL Adr utförs om N-flaggan = 0, dvs. om subtraktionen ger ett resultat ≥ 0. Värdet $97_{16} = 151_{10}$ motsvarar det negativa talet $-(256 - 151)_{10} = -105_{10}$.

Hoppvillkoret blir $W - (-105_{10}) \geq 0$; $W \geq -105_{10}$; $-105_{10} \leq W \leq 127_{10}$

Om overflow inträffar får N-flaggan i detta fall värdet 1 vilket medför att hoppet då inte utförs.

Overflow inträffar om $W - (-105_{10}) \geq 128_{10}$ dvs. $W \geq 128_{10} - 105_{10} = 23_{10}$.

Hoppet görs därför om W tillhör intervallet: $-105_{10} \leq W \leq 22_{10}$.

Eftersom 2-komplementrepresentation används för negativa tal delar vi upp intervallet i en negativ och en positiv del, $-105_{10} \leq W \leq -1$ och $0 \leq W \leq 22_{10}$.

Det verkliga intervallet för den negativa delen blir:
$256_{10} - 105_{10} \leq W \leq 256_{10} - 1$ dvs. $151_{10} \leq W \leq 255_{10}$.

Hoppet utförs alltså om W tillhör något av intervallen: $0 \leq W \leq 22_{10}$ eller $151_{10} \leq W \leq 255_{10}$. **(3p)**

c)

Adr	Data	~	Assemblerkod			Offset
			Läge	Operation	Operand	
				LDA	#10	
		4		JSR	DELAY	
			CONST	EQU	5	
				ORG	$80	
80	10	3	DELAY	PSHA		
81	10	3		PSHA		
82	F0 05	2	YLOOP	LDA	#CONST	
84	08	3	ILOOP	DECA		
85	25 FD	4		BNE	ILOOP	84 – 87 = FD
87	48 00	4		DEC	0,SP	
89	25 F7	4		BNE	YLOOP	82 – 8B = F7
8B	14	3		PULA		
8C	14	3		PULA		
8D	43	2		RTS		

(3p)

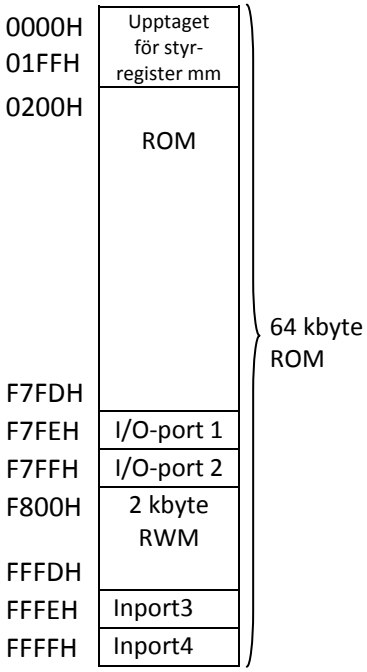
d) JSR DELAY tar 4 klockpulser.
A-reg innehåller värdet 10 vid anropet, så yttre slingan (YLOOP) utförs 10 gånger.
$\underline{T} = 4+3+3+[2+(3+4)\cdot 5+4+4]\cdot 10+3+3+2 = 18+[10+7\cdot 5\ ]\cdot 10 = \underline{468 \text{ st } (\mu s)}$ **(3p)**

7.

```
BOS       EQU   $FA          Bottom of stack
INPORT    EQU   $FB          DIPSWITCH
UTPORT    EQU   $FC          Ljusdiodramp (LED)
DELAY     EQU   $80          Färdig fördröjningsrutin (200 ms)
          ORG   $10          Huvudprogrammets startadress
START     LDSP  #BOS         Initiera stackbotten
          CLR   UTPORT       Alla ljusdioder släckta
PLOOP     LDX   #PATTERN     Pekare till tabell för diodmönster
WAIT      TST   INPORT       Starta visning?
          BPL   WAIT         Nej, bit7 = 0
          LDA   ,X+          Ja, bit7 = 1. Hämta bitmönster från tabell
          STA   UTPORT       Visa bitmönster
          JSR   DELAY        Vänta 200 ms
          CMPX  #PATTERN+8   Hela tabellen avverkad?
          BEQ   PLOOP        Ja, börja om från startläget
          BRA   WAIT         Nej, kolla om nästa bitmönster skall visas
;
;                            Tabell med samtliga bitmönster
PATTERN   FCB   %10000001,%01000010,%00100100,%00011000
          FCB   %00011000,%00100100,%01000010,%10000001
```

(7p)

8.



Passiv
$512 = 2^9$

	A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Start: 0000H	=	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Slut: 01FFH	=	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
		CS							9 st								

RWM
$2k = 2^{11}$

	A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Start: F800H	=	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
Slut: FFFFH	=	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
		CS					11 st										

I/O-port 1

	A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Adr: F7FEH	=	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0
									CS								

I/O-port 2

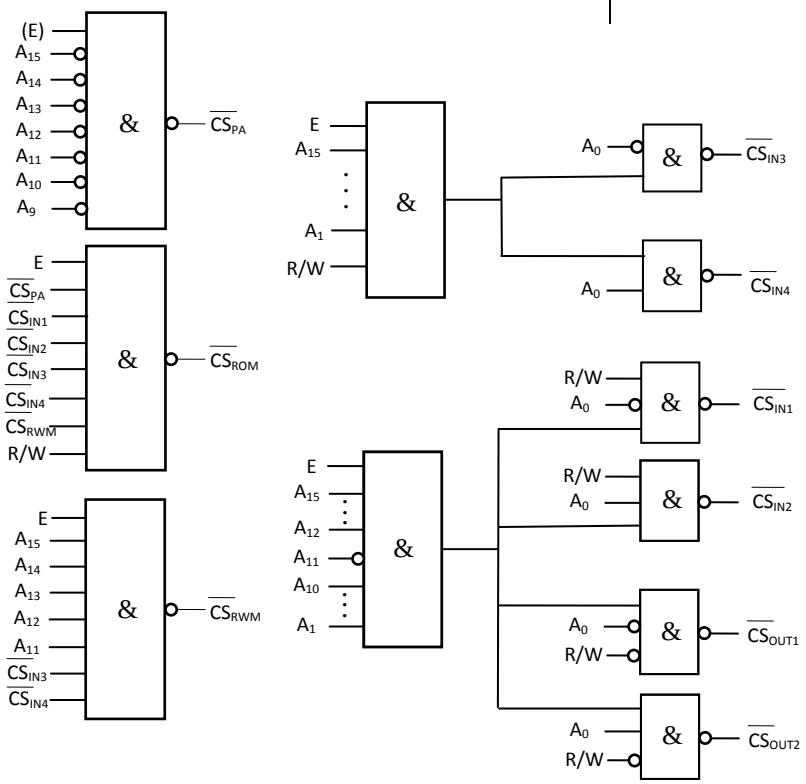
	A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Adr: F7FFH	=	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
									CS								

Inport 3

	A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Adr: FFFEH	=	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
									CS								

Inport 4

	A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Adr: FFFFH	=	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
									CS								



(7p)