# Lösningsförslag tenta 2013-08-22

**(Version 4 med reservation för eventuella fel.)**

**1.** $X = 1\ 1011_2$; $Y = 1\ 0111_2$ (5 bitars ordlängd)

**a)** $[0, 2^n–1] = [0, 2^5–1] = [0, 31]$ **(1p)**

**b)** $[–2^{n-1}, +2^{n-1}-1] = [–2^{5-1}, +2^{5-1}–1] = [–16, +15]$ **(1p)**

**c)**
```
S = X+Y
543210    bitnummer
111110    carry
 11011    X
+10111    Y
 10010 =  S
```
**(1p)**

**d)** $\underline{N} = s_4 = \underline{1}$
$\underline{Z} = \underline{0}$ $(S \neq 0)$
$\underline{V} = x_4*y_4*s_4' + x_4'*y_4'*s_4 = 1*1*1' + 1'*1'*1 = \underline{0}$
$\underline{C} = c_5 = \underline{1}$

NZVC = 1001 **(1p)**

**e)**
```
D = X+Y1k+1
543210    bitnummer
110111    carry
 11011    X
+01000    Y1k
 00100 =  D
```
**(1p)**

**f)** $\underline{N} = d_4 = \underline{0}$
$\underline{Z} = \underline{0}$ $(D \neq 0)$
$\underline{V} = x_4*y_{4_{1k}}*d_4' + x_4'*y_{4_{1k}}'*d_4 = 1*0*0' + 1'*0'*0 = \underline{0}$
$\underline{C} = c_5' = 1' = \underline{0}$

NZVC = 0000 **(1p)**

**g)** $\underline{X} = 1\ 1011_2 = 1B_{16} = 1*16 + 11 = \underline{27}$
$\underline{Y} = 1\ 0111_2 = 17_{16} = 1*16 + 7 = \underline{23}$
$\underline{S} = 1\ 0010_2 = 12_{16} = 1*16 + 2 = \underline{18}$ Resultatet S är felaktigt eftersom C = 1.
$\underline{D} = 0\ 0100_2 = 04_{16} = 0*16 + 4 = \underline{4}$ Resultatet D är korrekt eftersom C = 0. **(1p)**

**h)** ($x_4$ = 1, neg ) $X_{2k} = 2^5 – 27 = 32 – 27 = \underline{5}$ $\underline{X}$ motsvarar $\underline{–5}$
($y_4$ = 1, neg) $Y_{2k} = 2^5 – 23 = 32 – 23 = 9$ $\underline{Y}$ motsvarar $\underline{–9}$
($s_4$ = 1, neg) $S_{2k} = 2^5 – 18 = 32 – 18 = 14$ $\underline{S}$ motsv $\underline{–14}$ Resultatet S är korrekt eftersom V = 0.
($d_4$ = 0, pos) D = 4 Resultatet D är korrekt eftersom V = 0. **(1p)**

**i)** $N_{flyt} = C4\ 00\ 00\ 00_{16} = 1/100\ 0100\ 0/000\ 0000\ 0000\ 0000\ 0000\ 0000_2$ Format: s/c/f
s = 1 (neg); exp = c – $(2^{8-1} – 1)$ = 136 – 127 = 9;
Mantissa = 1.f = 1.000 0000 0000 0000 0000 $0000_2$
$\underline{N} = –1.000\ 0000\ 0000\ 0000\ 0000\ 0000 \cdot 2^9 = \underline{– 512}$ **(2p)**

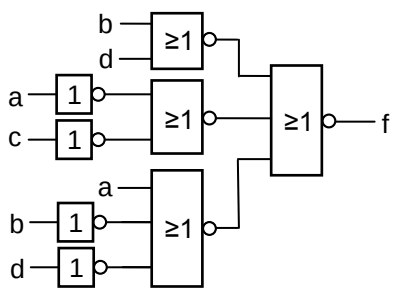
**2.**

**a)** VL: $x⊕(y·z) = x'(yz) + x(yz)' = x'yz + x(y' + z') = x'yz + xy' + xz' = x'yz + xy'z + xy'z' + xyz' + \sout{xy'z'}$
HL: $(x⊕y)(x⊕z) = (x'y + xy')(x'z + xz') = x'yz + xy'z'$
SP-normalformerna för VL och HL är olika. Alltså gäller inte lagen för XOR. **(3p)**

**b)**

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

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




**(4p)**

**3.**

**a)**

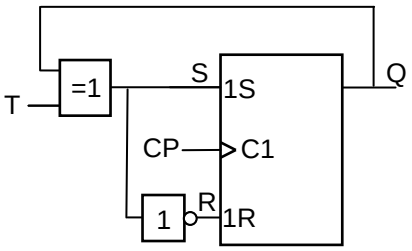
| T | Q | $Q^+$ | S | R |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | - |
| 0 | 1 | 1 | - | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |

S = T⊕Q

| T \ Q | 0 | 1 |
|---|---|---|
| 0 | 0 | - |
| 1 | 1 | 0 |

R = (T⊕Q)'

| T \ Q | 0 | 1 |
|---|---|---|
| 0 | - | 0 |
| 1 | 0 | 1 |




**(4p)**

**b)**

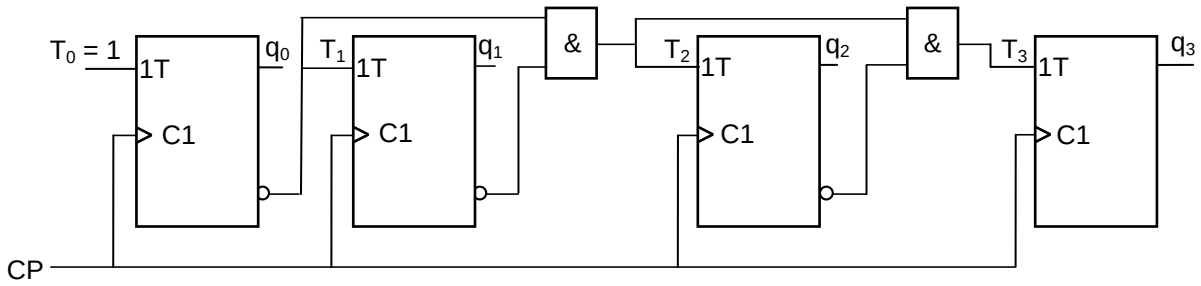
För T-vippor gäller att $q^+ = Q$ för T = 0 och $q^+ = Q'$ för T = 1. Q ändras alltså om T = 1.

Räknarens utsignaler : $q_3q_2q_1q_0$

När man betraktar de binära värdena vid uppräkning ser man att bit $q_0$ alltid ändras efter klockpuls och att övriga bitar bara ändrar sig om alla bitar med lägre nummer är nollor.

Det innebär att följande uttryck fås för T: $T_0 = 1$; $T_1 = q_0'$; $T_2 = q_1'q_0'$; $T_3 = q_2'q_1'q_0'$

Omskrivning ger: $T_0 = 1$; $T_1 = q_0'$; $T_2 = q_1'T_1$; $T_3 = q_2'T_2$




**(7p)**

**4.**

| CP | Styrsignaler (=1) | RTN | Reg A(8) | Reg B(8) | Reg T(8) | Reg R(8) |
|---|---|---|---|---|---|---|
| 1 | $OE_B$, $LD_T$ | B → T | 64 | C8 | ? | ? |
| 2 | $OE_A$, $f_3$, $f_2$, $LD_R$ | A – T – 1 → R | 64 | C8 | C8 | ? |
| 3 | $OE_R$, $f_3$, $f_1$, $f_0$, $LD_R$ | 2R → R | 64 | C8 | C8 | 9B |
| 4 | $OE_R$, $f_3$, $f_1$, $f_0$, $LD_R$ | 2R → R | 64 | C8 | C8 | 36 |
| 5 | $OE_R$, $f_3$, $f_2$, $g_0$, $LD_R$ | R – T → R | 64 | C8 | C8 | 6C |
| 6 | $OE_R$, $LD_B$ | R → B | 64 | C8 | C8 | A4 |
| 7 | | | 64 | A4 | C8 | A4 |

**(1+3p)**

**5. a)** FLISP:

| State | Summaterm | RTN-beskrivning | Styrsignaler |
|---|---|---|---|
| $Q_4$ | $Q_4 \cdot I_{xx}$ | PC→TA, PC→T | $OE_{PC}$, $LD_{TA}$, $LD_T$ |
| $Q_5$ | $Q_5 \cdot I_{xx}$ | M(TA)+T+1→R, PC+1→PC | MR, $f_3$, $f_1$, $f_0$, $g_0$, $g_{14}$, $LD_R$, IncPC |
| $Q_6$ | $Q_6 \cdot I_{xx} \cdot (C+Z)'$ | If (C+Z)=0: R→PC | $OE_R$, $LD_{PC}$ |
| | $Q_6 \cdot I_{xx}$ | New Fetch | NF |

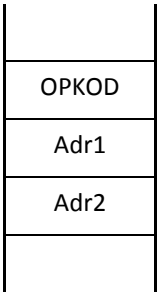
FLEX:

| State | Summaterm | RTN-beskrivning | Styrsignaler |
|---|---|---|---|
| $Q_5$ | $Q_5 \cdot I_{xx}$ | PC→MA, PC→T | $OE_{PC}$, $LD_{MA}$, $LD_T$ |
| $Q_6$ | $Q_6 \cdot I_{xx}$ | M+T+1→R, PC+1→PC | MR, $f_3$, $f_1$, $g_0$, $LD_R$, IncPC |
| $Q_7$ | $Q_7 \cdot I_{xx} \cdot (C+Z)'$ | If (C+Z)=0: R→PC | $OE_R$, $LD_{PC}$ |
| | $Q_7 \cdot I_{xx}$ | Next Fetch | NF |

Detta är: BHI Adr **(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(TA) → R | MR, $g_{14}$, $f_3$, $f_0$, $LD_R$ |
| $Q_6$ | $Q_6 \cdot I_{DF}$ | M(PC) → TA, PC+1 → PC | MR, $LD_{TA}$, $INC_{PC}$ |
| $Q_7$ | $Q_7 \cdot I_{DF}$ | M(TA) → T | MR, $g_{14}$, $LD_T$ |
| $Q_8$ | $Q_8 \cdot I_{DF}$ | R + T → R, ALU(NZVC) → CC | $OE_R$, $f_3$, $f_1$, $f_0$, $LD_R$, $LD_{CC}$ |
| $Q_9$ | $Q_9 \cdot I_{DF}$ | R → M(TA), (New Fetch) | $OE_R$, $g_{14}$, MW, 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 | MR, $LD_{MA}$ |
| $Q_7$ | $Q_7 \cdot I_{DF}$ | M → T | MR, $LD_T$ |
| $Q_8$ | $Q_8 \cdot I_{DF}$ | PC → MA, PC+1 → PC | $OE_{PC}$, $LD_{MA}$, IncPC |
| $Q_9$ | $Q_9 \cdot I_{DF}$ | M → MA | MR, $LD_{MA}$ |
| $Q_A$ | $Q_A \cdot I_{DF}$ | M + T → R, Flags → CC | MR, $f_3$, $f_1$, $LD_R$, $LD_{CC}$ |
| $Q_B$ | $Q_D \cdot I_{DF} \cdot Z$ | R → M, (Next Fetch) | $OE_R$, MW, NF |

**(4p)**

**6.**

**a)** Med hoppinstruktionen BRA Adr istället för JMP Adr blir inte programmet beroende av startadressen. Det kan då flyttas i minnet. **(1p)**

**b)** Subtraktionen $76_{16} - 81_{16} = F5_{16}$ ger flaggvärdena NZVC = 1011. Detta medför att BHI ej utförs (C+Z = 1), men att BGT utförs [(N⊕V)+Z=0]. **(2p)**

**c)**

| Adr | Data | ~ | Assemblerkod Läge | Operation | Operand | Offset |
|---|---|---|---|---|---|---|
| 80 | 10 | 3 | WAIT | PSHA | | |
| 81 | 10 | 3 | LOOP1 | PSHA | | |
| 82 | F0 04 | 2 | | LDA | #NUMB | |
| 84 | 08 | 3 | LOOP2 | DECA | | |
| 85 | 23 FD | 4 | | BPL | LOOP2 | 84 – 87 = FD |
| 87 | 14 | 3 | | PULA | | |
| 88 | 07 | 3 | | INCA | | |
| 89 | 25 F6 | 4 | | BNE | LOOP1 | 81 – 8B = F6 |
| 8B | 14 | 3 | DELX | PULA | | |
| 8C | 43 | 2 | | RTS | | |

**(3p)**

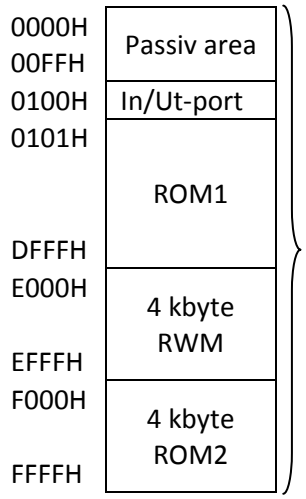
**d)** BSR WAIT tar 5 klockpulser.
A-reg innehåller värdet –10 vid anropet, så yttre slingan (LOOP1) utförs 10 gånger och inre slingan 4+1 = 5 gånger.
$\underline{T} = 5+3+[3+2+(3+4)\cdot5+3+3+4]\cdot10+3+2 = 13+[15+7\cdot5]\cdot10 = \underline{513\ \mu s}$ **(3p)**

**7.**

```
;*******************************************************************************
;  Subrutin DECNT som räknar antal ASCII-tecken som motsvarar decimala siffror i en noll-;
   terminerad sträng. Antal returneras i A-reg. X pekar på strängen vid anrop.
;  Påverkar A och CC.
;*******************************************************************************
DECCNT  PSHX            Spara register på stack
;
        CLR     COUNT
DELOOP  LDA     ,X+     Hämta data från stränggen
        BEQ     DECEX   Dataord = 0? Ja, avsluta
;
        ANDA    #$7F    Maska bort bit 7
        CMPA    #'0'    Undre gräns
        BLO     DELOOP  Ej decimal siffra
        CMPA    #'9'    Låg gräns
        BHI     DELOOP  Ej decimal siffra
        INC     COUNT   Bit7 = 1. Öka räknare
        BRA     DELOOP  Behandla nästa dataord
;
DECEX   LDA     COUNT   Hämta räknaren
        PULX            Återställ register
        RTS
;
COUNT   RMB     1       Räknare för decimala siffror
;
```

**(7p)**

**8.**




Passiv area
256 = $2^8$

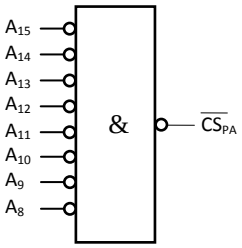
| | A | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Start: 0000H | = | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Slut: 00FFH | = | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | CS | | | | | | | | 8 st | | | | | | | |

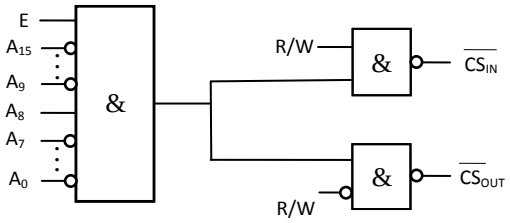
In/Utport:

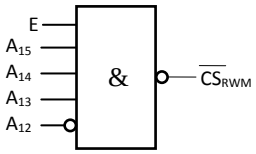
| | A | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Adr: 0100H | = | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | | | | | | | | CS | | | | | | | | |

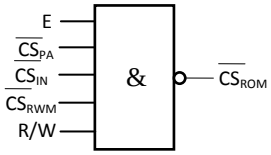
RWM
4k = $2^{12}$

| | A | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Start: E000H | = | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Slut: EFFFH | = | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | | CS | | | | 12 st | | | | | | | | | | | |




**(6p)**