Lösningsförslag tenta 2013-03-11 **(v3 med reservation för eventuella fel!)**

1. a) JMP $2345 → 06 23 45 **(1p)**

b) ORCC #1 ↔ SEC **(1p)**

c) LDAA -25,X → A6 E1 E7 **(2p)**

d) 3947_{16}: 04 51 00 = 04 lb rr. lb = 51 ⇒ TBEQ B,Adr med negativ 9-bitars offset, dvs. offseten = 1rr = 100_{16}. Offset = Tilladress – Frånadress ⇒ Tilladress = Offset + Frånadress. Frånadress är adressen till nästa OP-kod, dvs. $394A_{16}$. Vi måste använda 16 bitars ordlängd vid additionen som bildar ”Tilladress” så vi utvidgar offseten från 9 till 16 bitar dvs. 100_{16} → $FF00_{16}$. Tilladress = $FF00_{16}$ + $394A_{16}$ = $384A_{16}$. Instruktionen är alltså: TBEQ B,$384A **(3p)**

e) BEQ är ett programräknarrelativt hopp och använder en 8-bitars offset, som är ett tal med tecken. Hoppavståndet (offset) tillhör därför intervallet [–128,+127] = [$–80_{16}$,$+7F_{16}$].

1300_{16}: BEQ $1270; Offset = Tilladress – Frånadress = $1270_{16} – 1302_{16} = FF6E_{16}$.
Alt. Offset = –(Frånadress – Tilladress) = $1302_{16} – 1270_{16} = –92_{16}$ = – 0000000010010010_2.

Denna offset är alltså negativ och kräver minst 9 bitars ordlängd för att inrymma teckenbiten. Det finns dock bara plats för en 8-bitars offset, så assemblatorn ger ett felmeddelande. **(2p)**

f) BLT (<) avser tal med tecken. Det innebär att vi skall tolka data som tal i intervallet [-128, 127]. Talet $70_{16} = 7 \cdot 16 = 112_{10}$.

CMPA utför subtraktionen: 112 – W och hoppvillkoret blir: 112 – W < 0 eller W > 112.
När hänsyn tas till talområdet blir hoppvillkoret: $112 < W \leq 127$.

Flaggvillkoret tar hänsyn till overflow, så det fallet behöver man inte testa. **(2p)**

g) BLO (<) avser tal utan tecken. Det innebär att vi skall tolka data som tal i intervallet [0, 255]. Talet $70_{16} = 7 \cdot 16 = 112_{10}$.

CMPA utför subtraktionen: 112 – W och hoppvillkoret blir: 112 – W < 0 eller W > 112.
När hänsyn tas till talområdet blir hoppvillkoret: $112 < W \leq 255$. **(2p)**

h) Sk. stuffbitar med inverterat värde skjuts in efter en sekvens med ett antal (5) lika bitar på bussen och medför att det finns flanker tillräckligt ofta i bitströmmen för att noderna skall behålla synkronismen. **(2p)**

i) 53-bitars mantissa ger $2^{53} = 2^3 \cdot 2^{50}$ olika kombinationer. Vi utnyttjar att $2^{10} \approx 10^3$.
$2^3 \cdot 2^{50} = 8 \cdot 2^{50} = 8 \cdot (2^{10})^5 \approx 8 \cdot (10^3)^5 = 8 \cdot 10^{15}$. Vi kan därför räkna med att det är minst 15 (eller möjligen 16 pga approximationen) värdesiffror i ett motsvarande decimalt tal. **(2p)**

j) Primärminnesord får slumpvis placering i cache, så man vet inte var någonstans i cache man skall leta efter ”rätt” ord. Det krävs ett stort, dyrt och komplicerat associativt minne för att sökningen efter rätt ord inte skall ta för lång tid. **(2p)**

2. a)

```
MODIFY  PSHD                    Spara register på stack
        PSHX
        LDX     #DTAB           Pekare till tabell
MLOOP   LDAA    ,X              Hämta dataord från tabell
        CMPA    #$FF            Slutmarkering?
        BEQ     MODEX           Ja
        EORA    #%11000000      Invertera bit 7 o 6
        ORAA    #%00110000      Ettställ bit 5 o 4
        ANDA    #%11110011      Nollställ bit 3 o 2
        STAA    1,X+            Spara i tabell
        BRA     MLOOP           Fortsätt med nästa
MODEX   PULX                    Återställ register
        PULD
        RTS
```

(4p)

b)

```
        LDAB    #120            Antal dataord i BLOCK
        LDX     #BLOCK          Adress till BLOCK
        JSR     MIX             Bearbeta BLOCK
```

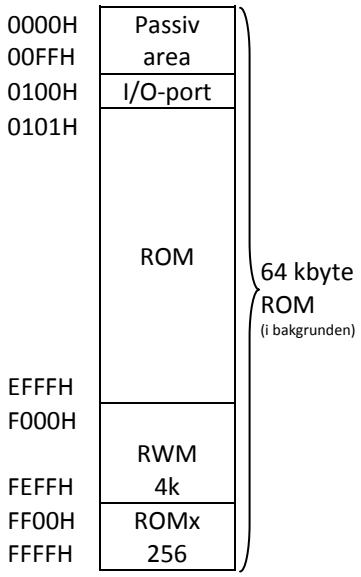
(2p)

c)

```
MIX     PSHX                    Spara register
        PSHY
        PSHD
        TSTB                    Färdigt?
        BEQ     MIXEX           Ja
        LDY     #4              Rotationsräknare
MLOOP   LDAA    ,X              Hämta dataord från tabell
        STAA    COPY0
        LSLA                    Flytta låg nibble till vänster
        LSLA
        LSLA
        LSLA
        STAA    COPY1           Höger nibble nu till vänster
        LDY     #4              Rotationsräknare
ROT     ROL     COPY0
        ROLA                    Hög nibblebit till A
        ROL     COPY1
        ROLA                    Låg nibblebit till A
        DEY                     Minska rotationsräknare
        BNE     ROT
        STAA    1,X+            Uppdatera tabell
        DECB                    Minska byteräknare
        BNE     MLOOP           Nästa byte om ej färdigt
MIXEX   PULD                    Återställ register
        PULY
        PULX
        RTS
COPY0   RMB  1
COPY1   RMB  1
```

(7p)

3.



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												

I/O-port:

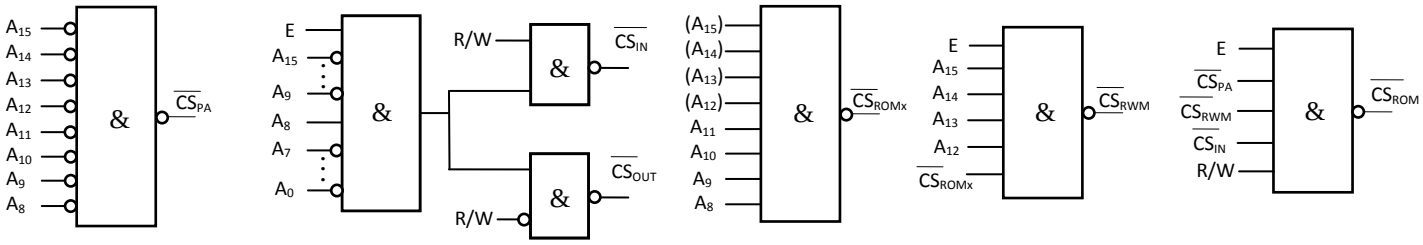
	A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Adr: 0100H	=	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
									CS								

RWM: $4k = 2^{12}$

	A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Start: F000H	=	1	1	1	1	0	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														

ROMx: $256 = 2^8$

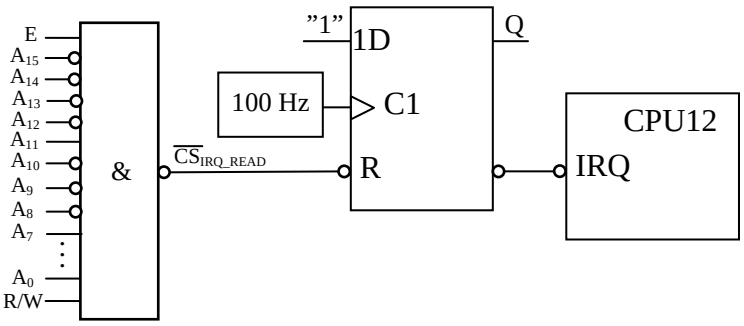
	A	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Start: FF00H	=	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
Slut: FFFFH	=	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
						CS											



(8p)

4. a) Adress för nollställning av avbrottsvippa: $08FF_{16} = 0000\ 1000\ 1111\ 1111_2$

Koppling:



(2p)

b) **Avbrottsrutin**

```
IRQR    TST     $8FF        Nollställ avbrottsvippan (T ex adr $8FF enligt figuren)
        LDX     ICNT1       Minska avbrottsräknare för 5 sekunder
        DEX
        STX     ICNT1       Uppdatera räknaren
        BNE     CHK10M      Ej 5 sekunder, kolla om 10 minuter har gått
        MOVW    #500,ICNT1  Ominitiera avbrottsräknare för 5 sekunder
        MOVB    $602,$3CD2  Läs givaren och uppdatera värdet
CHK10M  LDX     ICNT2       Minska avbrottsräknare för 10 minuter
        DEX
        STX     ICNT2       Uppdatera räknaren
        BNE     IRQEX       Ej 10 minuter, hoppa ut
        MOVW    #60000,ICNT2 Ominitiera avbrottsräknare för 10 minuter
        MOVW    $600,$3CD0  Läs 16-bitarsvärdet och uppdatera
IRQEX   RTI
```

(4p)

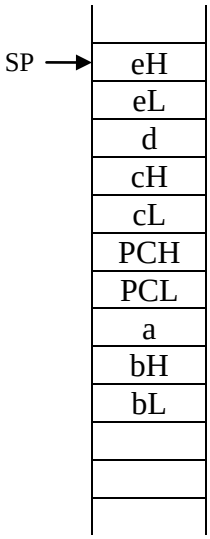
c) I**nitiering av variabler och avbrottssystem** (Stackpekaren antas initierad tidigare)

```
        TST     $8FF        Nollställ avbrottsvippan (T ex adr $8FF enligt figuren)
        MOVW    #500,ICNT1  Initiera avbrottsräknare för 5 sekunder
        MOVW    #60000,ICNT2 Initiera avbrottsräknare för 10 minuter
        MOVW    #IRQR,$FFF2 Avbrottsvektor
        CLI                 Aktivera avbrottssystem
        …
```

(2p)

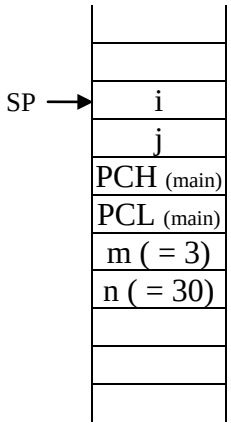
```
        ORG     $3CF0
ICNT1   RMB     2           Räknare för 5 sekunder
ICNT2   RMB     2           Räknare för 10 minuter
```

5. a)



(2p)

b)




```
main   LDAB   #30      Inparametrar till stack (n)
       PSHB
       LDAB   #3       (m)
       PSHB
       JSR    funcb
       LEAS   2,SP     Justera stack för inparametrar
       RTS
k      FCB    10       Global variabel
funcb: LEAS   -2,SP    Gör plats för lokala variabler
       LDAB   k        Global variabel
       STAB   1,SP     j = k
do
for    CLR    0,SP     i = 0 (Initiering av i för for-loop)
fortest LDAB  0,SP     Hämta i
       CMPB   #20      i < 20?
       BHS    whltst   Nej, lämna forloop
goon   LDAB   4,SP     Hämta m
       ADDB   1,SP     Bilda m + j
       STAB   1,SP     j = m + j
ominit INC    0,SP     i = i + 1 (Ominit av i i for-loop)
       BRA    fortest  Nytt varv
whltst LDAB   1,SP     Hämta j
       CMPB   #100     j <= 100? (while-test)
       BLE    do       Ja, ett varv till
       LDAB   5,SP     Nej, hämta n (bilda returvärde)
       ADDB   1,SP     n + j till B-reg
       LEAS   2,SP     Justera stack för lokala variabler
       RTS             Returvärde i B-reg
```

(7p)

c) for-satsen bildar j = k + x·3·20 = 10 + x·60, där x är antalet gånger den körs.

x bestäms av att j ≤ 100, där j är ett 8-bitars tal med tecken, dvs i intervallet [-128,+127]

j: 70, 130 (tolkas som –(256 – 130) = –126), –66, –6, 54, 114 (sista).

Returvärde: j + n = 114 + 30 = 144, men tolkas som –(256 – 144) = –112 **(3p)**