1.) a) ANDCC #$FE kan skrivas som CLC (2p)

b) Maskinkod 18 09 5A 15 38 är assemblerinstruktionen MOVB $1538,-6,Y (2p)

c) ORAB -100,SP har maskinkoden EA F1 9C (2p)

d) IBNE D,$1742 på adressen (opkod) 1800₁₆ har maskinkoden 04 B4 3F (2p)

e) $63 = 99 BHI avser tal utan tecken [0,255] (2p)

Hoppvillkor: $99 > W \Rightarrow \underline{0 < W < 99}$

f) BPL avser tal med tecken. (Eftersom positiva tal och noll har N=0 blir villkoret $\geq$)

Hoppvillkor: $99 \geq W$

[-128, +127] $-128 \leq W \leq 99$ Men vid subtraktionen 99-W kan overflow inträffa och då visar teckenflaggan N fel värde!

Overflow inträffar om $99 - W \geq 128 \Rightarrow 99 - 128 \geq W$; $W \leq -29$

Intervallet för hopp begränsas därför nedåt till $W = -28$

Alltså: $-28 \leq W \leq 99$ ger hopp.

Eftersom negativa tal skrivs som 2-komplement till motsvarande positiva tal, så delar vi upp intervallet i en positiv del (inkl 0) och en negativ del

$\underline{0 \leq W \leq 99}$

$-28 \leq W \leq -1 \rightarrow 256 - 28 \leq W \leq 256 - 1$ d.v.s. $\underline{228 \leq W \leq 255}$ (4p)

g) Stuffbitar används för att undvika för långa sekvenser av samma bitvärde 0 eller 1. Stuffbiten intogas efter en sådan sekvens och har det inverterade värdet. Orsaken till att problemet uppstår är att bitarna sänds okodade. (2p)

1. (forts.) h) $-200{,}375 = -11001000.011 = \overset{S=1}{\ominus} 1.\underbrace{1001000011}_{f} \cdot 2^{\overset{(7)}{exp}}$

s/c/f $C = exp + 127 = 7 + 127 = 6 + 128 = 10000110$

$N_{flyt} = \underbrace{1}_{S}\underbrace{10000110}_{C}\underbrace{10010000110\ldots0}_{f} = C3486000_{16}$ (2p)

---

i) Om man stryker teckenbiten S, så kan man betrakta de resterande bitarna som ett heltal vars värde direkt representerar absolutbeloppet hos motsvarande flyttal. Man kan då t.ex. sortera ett antal sådana tal med avseende på storleken (beloppet). (2p)

2.) a)

```
        LDX  #TAB        ~ 2
        TFR  X,Y           1
        LDD  4,X           3
YLOOP   LDX  6,Y           3
XLOOP   DBNE X,XLOOP     3.5 } 10
        DBNE D,YLOOP       3
        BRA  NEXT          3
TAB     FDB  15,-7,10,5,20,4,8   -
NEXT    NOP                1
        0  2  4 6  8
```

(4p)

$N = 2 + 1 + 3 + (3 + 3.5 + 3) \cdot 10 + 3 + 1 = 10 + 21 \cdot 10 = 220$ st

---

b)

```
XCNT    PSHY          Spara register
        PSHA
        LDY  #0       Nollställ biträknare
        TFR  X,D      Bitmönster till D
XLOOP   CPD  #0       Någon etta?
        BEQ  XEXIT    Nej, färdigt
        LSLD          Ja, skifta moto till carry
        BCC  XLOOP    Etta? Nej, fortsätt tills D=0
        INY           Ja, räkna
        BRA  XLOOP    Fortsätt tills D=0
XEXIT   TFR  Y,B      Färdigt räknevärde till B (YL→B)
        PULA          Återställ register
        PULY
        RTS
```

(4p)

2. (forts.) c)

```
STRADD   PSHX            Spara register
         PSHY
         CLRB            Låg byte av overflowräknare VCNT
STRLOOP  LDAA  ,X        Hämta data från STRING1
         CMPA  #$FF      Slutmarkering?
         BEQ   STREX     Ja.
         ADDA  1,Y+      Nej, addera med data från STRING2, Y++
         BVC   STORE     Ej overflow
         INCB            Overflow. Öka räknare låg byte.
         BNE   STORE     Ej carry till hög byte av räknare
         INC   VCNTH     Carry till hög byte
STORE    STAA  1,X+      Lagra summan i STRING1. X++
         BRA   STRLOOP   Nästa dataord
STREX    LDAA  VCNTH     Hämta hög byte av VCNT
         PULY            Återställ register
         PULX
         RTS             Returnera VCNT i D-reg
VCNTH    RMB   1         Hög byte av VCNT
```

(7p)

3.)

Passiv area: $4k = 2^{12}$

Start 0000_{16} = 0000 | 0000 0000 0000 0000 (A15 14 13 12 | 11 ... 0)

Slut $0FFF_{16}$ = 0000 | 1111 1111 1111 (CS | 12 st)

RWM $32k = 2^{15}$

Start 0000_{16} = 0 | 000 0000 0000 0000

Slut $7FFF_{16}$ = 0 | 111 1111 1111 1111 (CS)

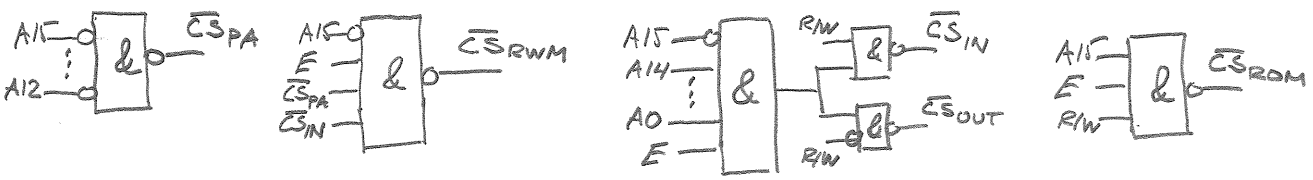
I/O

Adress: 0 | 111 1111 1111 1111 (CS)

ROM $32K = 2^{15}$

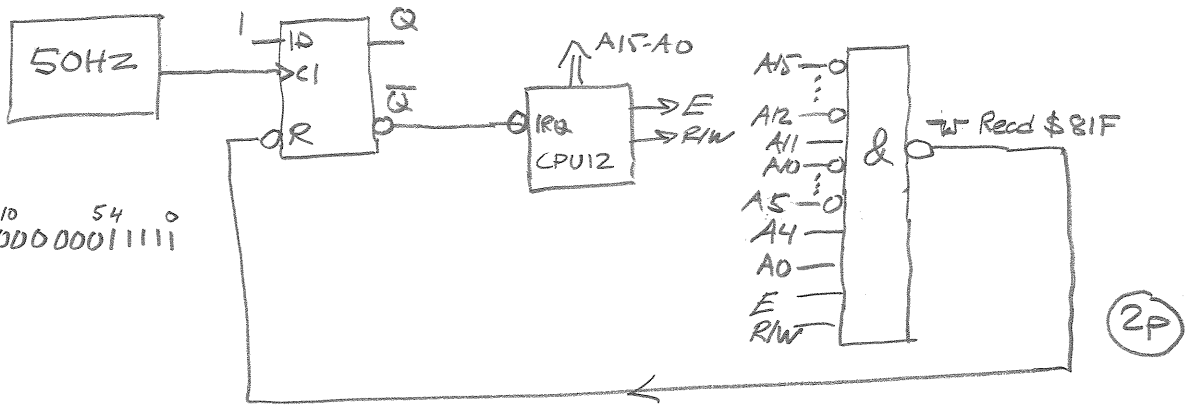
Start 8000_{16} = 1 | 000 0000 0000 0000

Slut $FFFF_{16}$ = 1 | 111 1111 1111 1111 (CS)



(8p)

4.) a)



$081F = 0000 1000 0001 1111 (15 12 11 10 5 4 0)

2p

b)

```
IRQRUT  TST   $81F      Nollställ avbrottsvippa
        DEC   CNT5      Minska 5-räknare
        BNE   NOT5      Räknare=0? Nej
        MOVB  #5,CNT5   Ominitiera till 5
        MOVB  INPORT1,GIV1  Läs Givare 1 och lagra värdet
NOT5    DEC   CNT50     Minska 50-räknare
        BNE   IRQEX     Räknare=0? Nej
        MOVB  #50,CNT50 Ominitiera till 50
        MOVB  INPORT2,GIV2  Läs Givare 2 och lagra värdet
IRQEX   RTI             Återhopp efter avbrott
```

4p

c)

```
(LDS   #BOS)
        :
MOVW  #IRQRUT,$FFF2   Sätt avbrottsvektorn
MOVB  #5,CNT5         Initiera räknare
MOVB  #50,CNT50
TST   $81F            Nollställ avbrottsvippan
CLI                   Aktivera avbrottssystem
        :
```

2p

5.

```
START   LDS   #$2000    BOS
        LDX   #JTAB     Pekare till adresstabell
        LDAA  INPORT    Läs SEL
        CMPA  #3        SEL > Max?
        LBHI  ERROR     Ja, fel
        LSLA            Nej. Dubbla A.
        LDX   A,X       Hämta adress från tabell till X-reg
        JSR   ,X        Utför subrutin
        BRA   NEXT
JTAB    FDB   $1100,$1150,$1200,$1380  Adresstabell
NEXT    :
```

7p

6. "Direct mapped cache"
Fördel: Enkel att implementera och snabb.
Nackdel: "Blockerar" adresserna som direktmappas => sannolikheten för "cachemiss" är stor.

2p