11.1
a) C := 0, V := 0, Z := 0, N := 0 **b)** C := 0, V := 1, Z := 0, N := 1
c) C := 0, V := 0, Z := 0, N := 0 **d)** C := 1, V := 0, Z := 0, N := 1
e) C := 0, V := 0, Z := 0, N := 1 **f)** C := 1, V := 1, Z := 0, N := 1
g) C := 1, V := 1, Z := 0, N := 0 **h)** C := 1, V := 1, Z := 0, N := 1

11.2 **a)** 0000 **b)** 1010 **c)** 0101 **d)** 0001

11.3 **a)** 0000 **b)** 0100 **c)** 1011 **d)** 0001

11.4 **a)** B6 40 00 4A B7 40 02 **b)** B6 80 00 1F 89 53 F7 80 01

11.5
```
a)  CLRA          b)  TFR   A,B      c)  LDAA  $8364
    CLRB              STAA  $1000        CMPA  $1918
    ANDA  $0020       ASRA               BEQ   $120C   (=PC+4)
    ORAB  $0022       ORAA  $1000        TSTA
```

11.6 **a)** 0,125 µs **b)** 0,125 µs **c)** 0,375 µs **d)** 0,125 µs **e)** 0,375 µs **f)** 0,25 µs

---

11.7
```
TPA  =  TFR  CCR, A
TSX  =  TFR  SP, X      TSY  =  TFR  SP, Y
TXS  =  TFR  X, SP      TYS  =  TFR  Y, SP
```

11.8
```
XGDX =  EXG  D, X       EXDY =  EXG  D, Y
```

11.9 MOVEB $6892, $1010 gör detsamma som instruktionerna

```
LDA  $6892
STA  $1010
```

11.10 (A) = 12H, (PC) = 448AH, C=0

11.11 ASLD gör detsamma som instruktionerna

```
ASLB        * b7 till C
ROLA        * C till a0
```

11.12
```
negd  COMA
      COMB
      ADDD  #1
```

11.13
```
DES   DEX   DEY
INS   INX   INY
```

11.14
```
Nollställning av flagga:  CLC  CLI  CLV
Ettställning av flagga:   SEC  SEI  SEV
```

11.15
```
CPD  operandinfo    CPS  operandinfo
CPX  operandinfo    CPY  operandinfo
```

11.16 CBA A-B → flaggpåverkan

11.17
```
Dcnt1  LDY   #0     * Temporär räknare för antal ettor = 0.
LOOP   CPD   #0     * Har D-registret någon etta?
       BEQ   END    * Nej!
       LSRD         * Ja, för över lsb till C-flaggan
       BCC   LOOP   * var lsb = 0?
       INY          * lsb var = 1, öka räknaren med 1
       BRA   LOOP   * fortsätt undersöka det som är kvar i D-registret
END    TFR   Y, A   * antalet ettor lämnas i A-registret
```


11.18
```
LDAA  $C500
ADDA  $C501
STAA  $C502
```

11.19
```
LDD   $C500
ADDD  $C502
STD   $C504
```

11.20
```
COPY   LDAB  #$10     * räknare
       LDX   #$1000   * pekare 1
       LDY   #$2000   * pekare 2
LOOP   LDAA  ,X+      * läs data
       STAA  ,Y+      * skriv data
       DECB           * räkna ner
       BNE   LOOP     * fortsätt tills klart
```

11.21

		SP	M(SP)	M(SP+1)	M(SP+2)
		1098H	40H	98H	FFH
PSHY		1096H	Yhigh	Ylow	40H
BSR	SUBR	1094H	80H	10H	Yhigh
PSHX		1092H	Xhigh	Xlow	80H
PULA		1093H	Xlow	80H	10H
PULB		1094H	80H	10H	Yhigh
RTS		1096H	Yhigh	Ylow	40H
PULY		1098H	40H	98H	FFH

11.22
```
LDS   #$F120
LDAA  #$06
PSHA
LDAA  #$35
PSHA
LDAA  #$28
PSHA          * SP efter = F11DH
```

11.23 a)
```
MASKBit  LDX   #$A300      * startadress
LOOP     BCLR  X+,%01111111 * nollställ ms bit
         CPX   #$A600
         BNE   LOOP        * fortsätt om inte färdigt
         RTS
```

b)
```
MASKBit  LDX   #$A300      * startadress
LOOP     BCLR  X+,%01111111
         LDAA  ,X          * hämta byten
         ANDA  #$7F        * nollställ ms bit
         STAA  ,X+         * skriv tillbaks byte
         CPX   #$A600
         BNE   LOOP        * fortsätt om inte färdigt
         RTS
```

11.24
```
ADD40B   LDAB  #5          * 5 bytes längd
         ANDCC #$FE        * nollställ carry
ADD40B2  LDAA  ,X
         ADCA  ,Y+
         STAA  ,X+
         DECB
```

```
                BNE     ADD40B2
                RTS
```

11.25 Lösningsförslag 1:

```
NBCDbin PSHA
        LSRA            * skifta fram...
        LSRA            * ... mest signifikanta
        LSRA            * ... nibble
        LSRA            * ... Vilket är:
        LDAB    #10     * antal "tio-tal"
        MUL
        PULA            * minst signifikanta ...
        ANDA    #$0F    * .. nibble
        PSHB            * temporärt på stack ...
        ADDA    ,S      * ... för att addera till A
        INS             * balansera stack
        RTS             * binärtal nu i A
```

Lösningsförslag 2:

```
NBCDbin PSHA
        LDAB    #16
        MUL             * frigör msb i A
        LDAB    #10
        MUL             * resultat i B
        PULA
        ANDA    #$0F    * frigör lsb i A
        PSHB            * temporärt på stack ...
        ADDA    ,S      * ... för att addera till A
        INS             * balansera stack
        RTS
```

Lösningsförslag 3:

```
NBCDbin LDX     #tabell * tabellen med bin tal för  0 ... 99
        LDAA    A,X
        RTS

tabell  FCB     0,1,2,3,4,5,6,7,8,9,$A,$B,$C,$D,$E,$F
        FCB     $10,$11 .....$ 63

        * här används assemblerdirektiv FCB, beskrivs i Arbetshäfte 4
```

11.26

```
* Subroutine CHALPHA
* Checks for alpha character, allowed: A-Z, a-z
* Input      (X) = ptr to character
* Output     (A) = character
*           (CC)= Carry Clear if Ok     Set otherwise

CHALPHA LDAA    ,X              * get char
        CMPA    #'A'            * lower limit
        BLO     CHACS           * illegal if lower
        CMPA    #'z'            * upper limit
        BHI     CHACS           * illegal if higher
        CMPA    #'Z'            * intermediate A-Z ?
        BLS     CHACC           * branch if
        CMPA    #'a'            * intermediate a-z ?
        BHS     CHACC           * branch if
```



```
        CHACS   ORCC    #1              * set carry
                BRA     EXIT
        CHACC   ANDCC   #$FE            * clear carry
        EXIT    RTS
```

11.27

```
SEARCH  LDAA    #$55            * sökmönster
        LDX     #$2000          * startadress
LOOP    CMPA    ,X+             * jämför, peka på nästa
        BEQ     EXIT            * om lika: korrigera pekare
        CMX     #$3000          * fältet klart?
        BNE     LOOP
        BRA     EXIT2           * ingen korrigering
EXIT    DES
EXIT2   RTS
```

11.28

```
CreSeg  LDX     #SEGTAB
        LDAA    A,X
        RTS

SEGTAB  FCB   $FC, $60, $DA, $F2, $66, $B6, $BE, $E0, $FE, $F6

              * här används assemblerdirektiv FCB, beskrivs i Arbetshäfte 4
```

11.29

```
* Subroutine ADDVEC
* Adds vectors V and W, size i, stores result in V
* Input   (X) = ptr to V
*         (Y) = ptr to W
*         (B) = size i
* No Regs affected

ADDVEC  PSHD
        PSHX
        PSHY
ADDV1   LDAA    ,X
        ADDA    ,Y+
        STAA    ,X+
        DECB
        BNE     ADDV1
        PULY
        PULX
        PULD
        RTS
```

11.30

```
* Subroutine DOTPRD
* computes dot product of two 2-complement vectors V and W,
* leaves the result at DTPD.
* Vector components are 1-bytes unsigned numbers.
* Typical Calling Sequence:   LDX     #V
*                             PSHX
*                             LDX     #W
*                             PSHX
*                             LDX     #DTPD
*                             PSHX
*                               :
*                             BSR     DOTPRD
```

```
*                       LEAS    6,S
* No registers affected

DOTPRD  PSHD
        PSHX
        PSHY
        PSHC
        LDX     15,S            * 2+9+4 bytes ned i stacken
        LDY     13,S            * 2+9+2 bytes ned
        LDAA    ,X              * komponent V1
        LDAB    ,Y              * komponent W1
        MUL                     * 1:a produkten
        LDX     11,S            * 2+9+4 bytes ned i stacken
        STD     ,X              * lagra 1:a produkten
        LDX     15,S
        LDY     13,S
        LDAA    ,+X             * komponent V1
        LDAB    ,+Y             * komponent W1
        MUL                     * 2:a produkten
        LDX     11,S
        ADDD    ,X
        STD     ,X
        PULC
        PULY
        PULX
        PULD
        RTS
```

11.31

```
* Subroutine SORT - sorts entries in table
* Typical Calling Sequence:  LDX    #SCRATCH        * 10 bytes area
*                           LDY     #TABLE          * 10 bytes table
*                           PSHX
*                           PSHY
*                             :
*                           BSR     SORT
*                           LEAS    4,S
* D,X,Y,U affected

SORT    PSHD
        PSHX
        PSHY
        PSHC
        LDY     9,S             * get TABLE pointer; 2+7 bytes ned
        LDX     11,S            * get pointer to SCRATCH area; 2+7+2 bytes ned
        LDAA    #10             * SORT-counter value
        PSHA                    * SORT-counter on stack

SORT10  LDAB    #10             * element counter value to FNDMAX
        BSR     FNDMAX          * get max
        STAA    ,X+             * store it, increment pointer
        CLR     B,Y             * clear used element in table
        DEC     ,S              * Done?
        BNE     SORT10          * if not: Continue
        INS                     * adjust stack
        LDY     9,S             * get TABLE pointer
```



```
        LDX     11,S            * get pointer to SCRATCH area
        LDAA    #10             * restore SORT-counter value

SORT20  LDAB    ,X+             * copy scratch table
        STAB    ,Y+
        DECA
        BNE     SORT20          * table sorted ?
        PULC
        PULY
        PULX
        PULD
        RTS     DONE
```

11.32 Obalanserad stack, fel återhoppsadress

11.33 **a)** 20 03 **b)** 20 64 **c)** 20 FE **d)** 20 F4 **e)** 20 D7

11.34 **a)** De fyra instruktionerna är

DBEQ	r, hoppadress	r - 1 → r,	if r = 0 PC + offset → PC
DBNE	r, hoppadress	r - 1 → r,	if r ≠ 0 PC + offset → PC
IBEQ	r, hoppadress	r + 1 → r,	if r = 0 PC + offset → PC
IBNE	r, hoppadress	r + 1 → r,	if r ≠ 0 PC + offset → PC
TBEQ	r, hoppadress	r - 0 ,	if = 0 PC + offset → PC
TBNE	r, hoppadress	r - 0 ,	if ≠ 0 PC + offset → PC

och har alla 9-bitars offset.

b) Dessa två instruktioner testar om ett antal bitar i ett minnesord är noll resp ett. De är

BRCLR	operandinfo, mask, hoppadress	$\underline{M} \wedge$ mask , if = 0 PC + offset → PC
BRSET	operandinfo, mask, hoppadress	$M \wedge$ mask , if = 0 PC + offset → PC

och har båda 8-bitars offset.