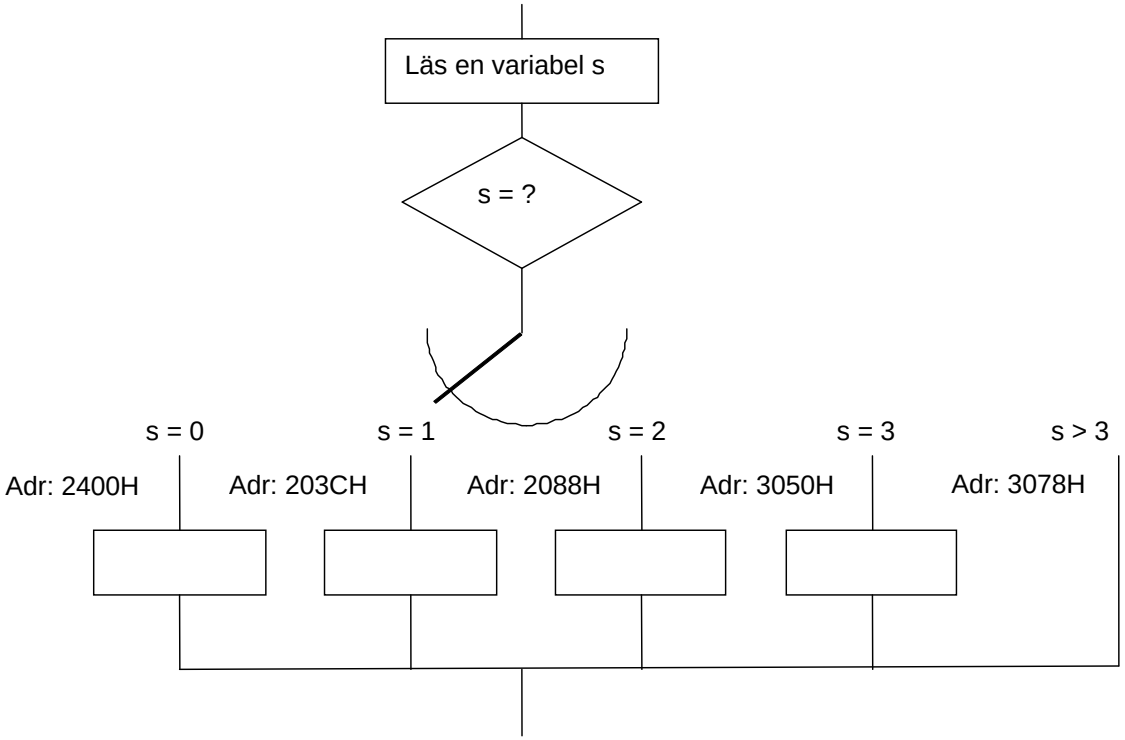
Extrauppgifter för CPU12

X.1a)
Skriv en instruktionssekvens som nollställer bit 3-0 i alla minnesord i adressintervallet [2035H, 2049H]. Använd X-registret för adressering.

X.1b)
Skriv en subrutin som maskerar (nollställer) bit 7 i varje byte inom adressområdet 3040H-3060H. Alla använda register sparas på stack.

X.2
Skriv en instruktionssekvens som adderar det fjärde och sjunde elementet i en tabell med åttabitars ord och placerar summan i tabellen som det femte elementet. Det gamla element fem skall skrivas över. Använd X-registret för adressering.

X.3
Skriv en instruktionssekvens som implementerar ett flerval enligt flödesplanen nedan.



X.4
För vilka värden på Y utförs hoppet nedan? Betrakta Y som ett tal [0,255].

```
        LDAA    #$85
        CMPA    #Y
        B(Villkor)  Hopp
```

om det villkorliga hoppet är

a) BHI
b) BHS
c) BLS
d) BLO
e) BGT
f) BGE
g) BLE
h) BLT

X.5

Skriv en subrutin som har ett binärt tal i A-registret som indata.

Om talet tillhör talområdet $[0000,1111]_2$ skall subrutinen returnera ASCII-tecknet för motsvarande hexadecimala siffra som utdata i A-registret.

Om talet är större än $(1111)_2$ skall subrutinen returnera talet FFH i A-registret.

De värden som finns i B-, CC-, X- och S-registren vid anrop av subrutinen måste vara oförändrade vid återhopp från subrutinen.

X.6

Skriv en subrutin som nollställer en specificerad bit i dataordet på en adress i minnet.

Vid anrop av subrutinen finns dataordets minnesadress i X-registret och bitnummret på den bit som skall nollställas i B-registret. Om bitnumret är större än 7 skall dataordet ej påverkas.

De värden som finns i A-, B-, CC-, X- och S-registren vid anrop av subrutinen måste vara oförändrade vid återhopp från subrutinen.

Exempel på anrop:

```
LDAB    #5
LDX     #$2038
JSR     BITZERO     Bit nr 5 i dataordet på adress 2038H i minnet nollställs
```

X.7

I minnet finns 25 st 8-bitars tal lagrade på adressen DVECTOR och framåt (ökande adress). De lagrade värdena är tal med inbyggt tecken (2-komplementrepresentation) och tillhör därför intervallet [-128, 127].

Skriv en subrutin i assemblerpråk som tar reda på hur många av talen som tillhör intervallet [-15, 20]. Antalet värden i detta intervall skall finnas i B-registret vid återhopp.

Endast register B och register CC får vara förändrade vid återhopp från subrutinen.

X.8

Skriv en subrutin HEXASC i assemblerpråk som översätter det binära talet $(b_3b_2b_1b_0)_2$ i register A till motsvarande 7-bitars ASCII-tecken. Innehållet i $b_7b_6b_5b_4$ i register A är okänt. Vid återhopp skall ASCII-tecknet finnas i register A med b_7 nollställd. Endast register A och register CC får vara förändrade vid återhopp från subrutinen.

X.9

Skriv en subrutin ASCHEX i assemblerpråk som översätter ett 7-bitars ASCII-tecken för en hexadecimal siffra (0-9 eller A-F) till motsvarande binära tal $(00000000)_2$-$(00001111)_2$. Vid anrop av subrutinen innehåller register A ASCII-tecknet i bitarna b_6-b_0 medan b_7 är en checkbit med okänt innehåll. Vid återhopp skall det binära talet finnas i register A. Om innehållet i register A vid anropet ej är ASCII-tecknet för en hexadecimal siffra så skall talet $(11111111)_2$ finnas i register A vid återhopp. Endast register A och register CC får vara förändrade vid återhopp från subrutinen.

X.10

I ett datorsystem med processorn CPU12 behövs en subrutin, STRCONV, som omvandlar en sträng med ASCII-tecknen för hexadecimala siffror i minnet till en annan sträng med motsvarande 8-bitars dataord. ASCII-strängen består av ett jämnt antal ASCII-tecken med dataordet 0 tillagt sist. Principen för omvandlingen framgår av följande exempel:

ASCII-strängen 42H, 35H, 37H, 38H, 36H, 32H, 41H, 33H, 0

ger binära strängen 10110101, 01111000, 01100010, 10100011.

Skriv subrutinen STRCONV. Vid anrop av subrutinen skall adressen till ASCII-strängen finnas i X-registret och adressen till den binära strängen i Y-registret. Då en korrekt omvandling har genomförts skall carryflaggan nollställas före återhopp. Om ett otillåtet ASCII-tecken upptäcks skall omvandlingen avbrytas och carryflaggan ettställas före återhopp. Endast flaggregistret får vara förändrat vid återhopp. ASCII-tecknen är lagrade i bit6-bit0 på varje adress i textsträngen. Bit7 i textsträngens dataord har värdet 0. Textsträngen avslutas med datavärdet 0.

X.11

I ett datorsystem med processorn CPU12 behövs en subrutin, BLKASC, som omvandlar ett block med 8-bitars dataord i minnet till en nollterminerad (= som avslutas med en byte med värdet 0) sträng med ASCII-tecknen för hexadecimala siffror. Omvandlingsprincipen framgår av följande exempel:

Minnesblocket 10110101, 01111000, 01100010, 10100011.

ger ASCII-strängen 42H, 35H, 37H, 38H, 36H, 32H, 41H, 33H, 0

Skriv subrutinen BLKASC. Vid anrop av subrutinen skall startadressen till minnesblocket resp. ASCII-strängen finnas i X-registret resp. Y-registret och antalet minnesord i A-registret. Endast flaggregistret får vara förändrat vid återhopp.

X.12

Visa hur stacken används när programmet nedan körs!

Minnesadr (Hex)

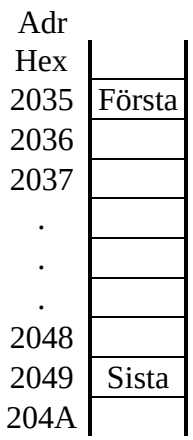
```
1000:   LDS   #$3C00
1013:     :       :
  :       :
1019:   LDX   #$1360
101C:   CLRA                    } Huvudprogram
101D:   JSR   $1040
1020:   STAA  $24E0
1023:
  :       :
  :       :

1040:   PSHA
1041:   PSHX
1042:   PSHC
1043:   LDX   #$1098
1046:     :       :
  :       :
104D:   LDAA  #$03              } Subrutin 1
104F:   JSR   $1060
1052:     :       :
  :
  :     PULC
  :     PULX
  :     PULA
  :     RTS
  :       :

1060:   PSHA
1061:   PSHC
1062:   PSHX
  :     LDX   #$2315
  :       :
  :     INCA                      } Subrutin 2
  :       :
  :     PULX
  :     PULC
  :     PULA
  :     RTS
```

Lösningar extrauppgifter X.1 - X.9

X.1a)

Lösning 1:

```
        .
        LDX     #$2035      Sätt pekare till tabellen
        LDAB    #$15        Sätt varvräknare till antal varv
LOOP    LDAA    ,X          Hämta data från tabellen
        ANDA    #%11110000  Nollställ bit 3-0
        STAA    1,X+        Uppdatera tabell. Öka pekare med 1
        DECB                Minska varvräknare med ett
        BNE     LOOP        Fortsätt med nästa värde
        .                   Färdigt, fortsätt med något annat
```



Lösning 2:

```
        .
        LDX     #$2035      Sätt pekare till tabellen
        LDAB    #$14        Sätt index till 14H
LOOP    LDAA    B,X         Hämta data från tabellen
        ANDA    #%11110000  Nollställ bit 3-0
        STAA    B,X         Skriv tillbaka till tabell
        DECB                Minska index med ett
        BPL     LOOP        Fortsätt med nästa värde så länge B inte är negativ
        .                   Färdigt, fortsätt med något annat
```

Lösning 3:

```
        .
        LDX     #$2035      Sätt pekare till tabellen
LOOP    LDAA    ,X          Hämta data från tabellen
        ANDA    #%11110000  Nollställ bit 3-0
        STAA    1,X+        Skriv tillbaka till tabell. Öka pekare med 1
        CPX     #$204A      Kontrollera om pekaren har passerat hela tabellen
        BNE     LOOP        Nej, fortsätt med nästa värde i tabellen
        .                   Färdigt, fortsätt med något annat
```

Lösning 4:

```
        .
        LDX     #$2035      Sätt pekare till tabellen
LOOP    BCLR    1,X+,#$0F   Nollställ bit 3-0 i tabellen och öka tabellpekare med 1
        CPX     #$204A      Kontrollera om pekaren har passerat hela tabellen
        BNE     LOOP        Nej, fortsätt med nästa värde i tabellen
        .                   Färdigt, fortsätt med något annat
```

Lösning 5:

```
        .
        LDAA    #$15        Sätt varvräknare till antal varv
        LDX     #$2035      Sätt pekare till tabellen
LOOP    BCLR    1,X+,#$0F   Nollställ bit 3-0 i tabellen och öka tabellpekare med 1
```

```
        DBNE    A,LOOP      Fortsätt med nästa värde i tabellen tills alla är färdiga
          .                 Färdigt, fortsätt med något annat
```

X.1b)
Lösning 1:

```
          .
MASKBIT   PSHA
          PSHC
          PSHX
*
          LDX   #$3040      Startadress
LOOP      LDAA  ,X          Hämta byten
          ANDA  #$7F        Nollställ ms bit
          STAA  1,X+        Skriv tillbaka byte. Öka pekare med 1
          CPX   #$3061
          BNE   LOOP        Om inte: fortsätt
*
MSKEX     PULX
          PULC
          PULA
          RTS
```

Lösning 2:

```
          .
MASKBIT   PSHA
          PSHC
          PSHX
*
          LDAA  #$21        Antal ord
          LDX   #$3040      Startadress
LOOP      BCLR  1,X+,#$80   Nollställ ms bit och öka pekare med 1
          DBNE  A,LOOP      Färdigt? Om inte: fortsätt
*
          PULX
          PULC
          PULA
          RTS
```

X.2
Lösning

Numrera elementen från noll och uppåt, dvs första elementet har nummer 0, andra har nummer 1 osv. Vi skall då addera element nr 3 och nr 6. Summan skall placeras som element nr 4.

```
        .
        .
    LDX     #Tabell     Sätt pekare till tabellen
    LDAA    3,X         Hämta element nummer 3 från tabellen
    ADDA    6,X         Addera element nr 6 till element nr 3 (A)
    STAA    4,X         Placera summan som element nr 4 i tabellen
        .
```

Adress	
	-
Tabell	Första
Tabell+1	Andra
Tabell+2	Tredje
Tabell+3	Fjärde
Tabell+4	Femte
Tabell+5	Sjätte
Tabell+6	Sjunde
Tabell+7	Åttonde

X.3
Lösning 1:

```
        .
    LDAA    Val         Läs variabeln s från minnet på adressen Val
    LBEQ    $2400       s är = 0. Hoppa till adressen 2400H enligt graf
    CMPA    #$03        Kontrollera om s > 3 eller s = 3
    LBHI    $3078       s är > 3. Hoppa till adressen 3078H enligt graf
    LBEQ    $3050       s är = 3. Hoppa till adressen 3050H enligt graf
    CMPA    #$01        Kontrollera om s = 1
    LBEQ    $203C       s är = 1. Hoppa till adressen 203CH
    LBRA    $2088       s måste vara = 2. Hoppa till adressen 2088H
        .
```

Lösning 2:

```
        .
        LDAA    Val         Läs variabeln s från minnet på adressen Val
        CMPA    #$03        Kontrollera om s > 3
        LBHI    $3078       s är > 3. Hoppa till adressen 3078H enligt graf
        ASLA                Dubbla tabelloffset för 16 bitars ord
        LDX     #JTAB       Sätt adress till tabell med hoppadresser
        LDX     A,X         Hämta hoppadress till X från JTAB
        JMP     ,X          Hoppa till den adress som hämtades från JTAB
JTAB    FDB     $2400,$203C,$2088,$3050 Hoppadresstabell
          .
```

Lösning 3:

```
          .
        LDAB    Val         Läs variabeln s från minnet på adressen Val
        CMPB    #$03        Kontrollera om s > 3
        LBHI    $3078       s är > 3. Hoppa till adressen 3078H enligt graf
        CLRA                D = B
        ASLB                Dubbla tabelloffset för 16 bitars ord
        LDX     #JTAB       Sätt adress till tabell med hoppadresser
        JMP     [D,X]       Hoppa till rätt adress via JTAB
JTAB    FDB     $2400,$203C,$2088,$3050 Hoppadresstabell
          .
          .
```

X.4
Lösning

```
LDAA      #$85      Talet 85H till A
CMPA      #Y        Bilda 85H - Y
B(Villkor) Hopp
```

Storleksjämförelse görs alltså mellan 85H = 133 och Y.

Alla hoppvillkoren i a - d avser tal utan inbyggt tecken.
För 8-bitars tal utan inbyggt tecken gäller: $0 \le Y \le 255$

Om det villkorliga hoppet är:

a) BHI (Higher) utförs hoppet om $133 > Y$, dvs $Y < 133$
Svar: $0 \le Y < 133$

b) BHS (Higher or Same) utförs hoppet om $133 \ge Y$, dvs $Y \le 133$
Svar: $0 \le Y \le 133$

c) BLS (Lower or Same) utförs hoppet om $133 \le Y$, dvs $Y \ge 133$
Svar: $133 \le Y \le 255$

d) BLO (Lower) utförs hoppet om $133 < Y$, dvs $Y > 133$
Svar: $133 < Y \le 255$

Alla hoppvillkoren i e - h avser tal med inbyggt tecken (2-komplementrepresentation).
För 8-bitars tal med inbyggt tecken (2-komplementrepresentation) gäller: $-128 \le Y \le +127$
Talet 85H tolkas som det negativa talet -(100H - 85H) = -7BH = -123

Om det villkorliga hoppet är:

e) BGT (Greater Than) utförs hoppet om $-123 > Y$, dvs $Y < -123$

Y skall alltså tillhöra intervallet [-128, -123) Svar: $128 \le Y < 133$

f) BGE (Greater or Equal) utförs hoppet om $-123 \ge Y$, dvs $Y \le -123$

Y skall alltså tillhöra intervallet [-128, -123] Svar: $128 \le Y \le 133$

g) BLE (Less or Equal) utförs hoppet om $-123 \le Y$, dvs $Y \ge -123$

Y skall alltså tillhöra intervallet [-123, +127] Svar: $0 \le Y \le 127$ eller $133 \le Y \le 255$

h) BLT (Less Than) utförs hoppet om $-123 < Y$, dvs $Y > -123$

Y skall alltså tillhöra intervallet (-123, +127] Svar: $0 \le Y \le 127$ eller $133 < Y \le 255$

Lösning: X.5

Alt 1:

```
HEXASC  PSHC
        CMPA    #$0F    Kontrollera övre gräns
        BHI     FEL     Talet större än övre gräns
        ADDA    #$30    Bilda ASCII-tecken för decimal siffra
        CMPA    #$39    Kontrollera övre gräns för decimal siffra
        BLS     OK      Siffra 0-9
        ADDA    #$07    Bilda ASCII-tecken för bokstav A-F
OK      PULC
        RTS
FEL     LDAA    #$FF    Signalera att talet för stort med A=FFH
        BRA     OK
```

Alt 2:

```
HEXASC  PSHX
        PSHC
        CMPA    #$0F        Kontrollera övre gräns
        BHI     FEL         Talet större än övre gräns
        LDX     #ASCTAB     Sätt pekare till tabell med alla 16 ASCII-tecken 0-F
        LDAA    A,X         Hämta ASCII-tecken från tabell
EXIT    PULC
        PULX
        RTS
FEL     LDAA    #$FF        Signalera att talet för stort med A=FFH
        BRA     EXIT

ASCTAB  FCB     $30,$31,$32,$33,$34,$35$36,$37,     ASCII för 0-7
        FCB     $39,$40,$41,$42,$43$44,$45,$46      ASCII för 8-F
```

Lösning: X.6

```
BITZERO PSHB
        PSHC
        CMPB    #$07        Kontrollera övre gräns för bitnummer
        BHI     EXIT        Bitnumret är större än 7
        PSHX                Rädda X på stack
        LDX     #MSKTAB     Sätt pekare till tabell med masker
        LDAB    B,X         Hämta mask från tabell till B-reg
        PULX                Återställ X
        ANDB    ,X          Nollställ bit i dataordet (i B-reg)
        STAB    ,X          Skriv tillbaka dataordet till minnet
EXIT    PULC
        PULB
        RTS

MSKTAB  FCB     %11111110,%11111101,%11111011,%11110111     0-3
        FCB     %11101111,%11011111,%10111111,%01111111     4-7
```

Lösning: X.7

```
ICOUNT  PSHA
        PSHX
*
        CLRB                    Nollställ antal inom intervall
        LDX     #DVECTOR        Sätt pekare till tabell med tal
*
TLOOP   LDAA    1,X+            Hämta tal från tabell
        CMPA    #20             Kontrollera övre gräns
        BGT     OUT             Större (med tecken)
        CMPA    #-15            Kontrollera undre gräns
        BLT     OUT             Mindre (med tecken)
*
        INCB                    Nytt tal inom intervall
*
OUT     CPX     #DVECTOR+25     Färdigt?
        BNE     TLOOP
*
EXIT    PULX
        PULA
        RTS
```

Lösning: X.8

```
HEXASC  PSHX
        ANDA    #$0F            Nollställ bit 4-7
        LDX     #ASCTAB         Sätt pekare till tabell med alla 16 ASCII-tecken 0-F
        LDAA    A,X             Hämta ASCII-tecken från tabell
        PULX
        RTS
*
ASCTAB  FCB     $30,$31,$32,$33,$34,$35$36,$37      ASCII för 0-7
        FCB     $38,$39,$41,$42,$43$44,$45,$46      ASCII för 8-F
```

Lösning: X.9

```
ASCHEX  ANDA    #$7F    Nollställ bit 7
        SUBA    #$30    Bilda binärtal om ASCII-tecken för decimal siffra
        BLO     FEL     Ej hexsiffra
        CMPA    #9      Decimal siffra?
        BLS     EXIT    Siffra 0-9
        SUBA    #$07    Bilda tal för hexsiffra A-F
        CMPA    #$0A    Kolla undre gräns
        BLO     FEL     Ej hexsiffra
        CMPA    #$0F    Kolla övre gräns
        BLS     EXIT    Hexsiffra
*
FEL     LDAA    #$FF    Signalera felaktigt hextal med A=FFH
*
EXIT    RTS
```