Extra programmeringsuppgifter för FLIS-processorn

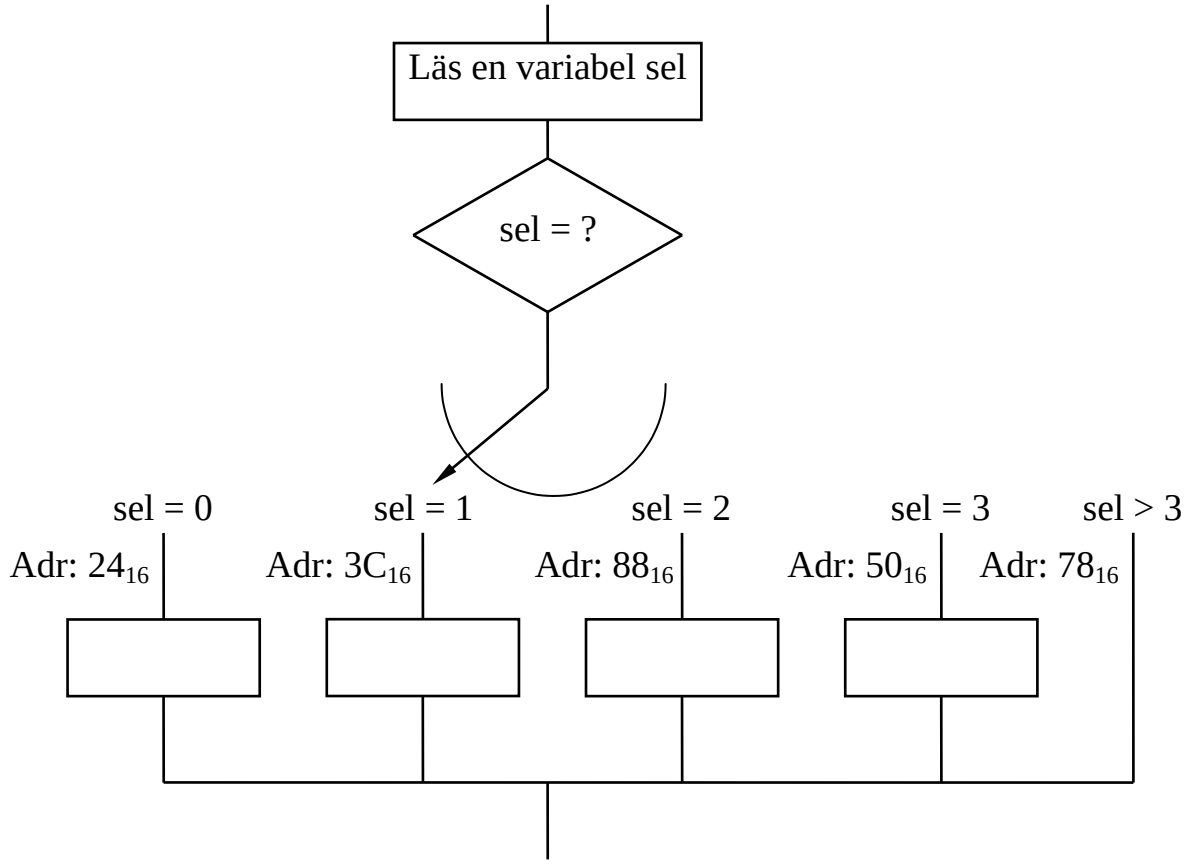
X.15

a) Skriv en instruktionssekvens för FLIS-processorn som nollställer bit 3-0 i alla minnesord i adressintervallet [35_{16}, 49_{16}]. Använd X-registret för adressering.

b) Skriv en subrutin för FLIS-processorn som maskerar (nollställer) bit 7 i varje byte inom adressområdet 40_{16}-$5F_{16}$.

X.16 Skriv en instruktionssekvens för FLIS-processorn 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.17 Skriv en instruktionssekvens för FLIS-processorn som implementerar ett flerval enligt flödesplanen nedan.



X.18 För vilka värden på W utförs hoppet nedan? Betrakta W som ett tal [0,255].

```
LDA      #$85
CMPA     #W
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.19 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 FF_{16} i A-registret.

De värden som finns i alla register, utom A och PC, vid anrop av subrutinen måste vara oförändrade vid återhopp från subrutinen.

X.20 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 bitnumret på den bit som skall nollställas i A-registret. Om bitnumret är större än 7 skall dataordet ej påverkas.

De värden som finns i alla register, utom PC, vid anrop av subrutinen måste vara oförändrade vid återhopp från subrutinen.

Exempel på anrop:

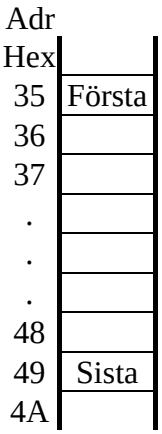
```
LDA     #5
LDX     #$20
JSR     BITZERO     Bit nr 5 i dataordet på adress 20₁₆ i minnet nollställs
```

Lösningar extrauppgifter X.15 - X.20

X.15a)

Lösning 1:

```
        .
        .
        LDX   #$35          Sätt pekare till tabellen
        LDA   #$15          Sätt varvräknare till antal dataord
        STA   COUNT         Startvärde för räknare i minnet
LOOP    LDA   0,X           Hämta data från tabellen
        ANDA  #%11110000    Nollställ bit 3-0
        STA   ,X+           Skriv tillbaka till tabell. Öka pekare
        DEC   COUNT         Minska varvräknare i minnet med ett
        BNE   LOOP          Fortsätt med nästa värde
        .                   Färdigt, fortsätt med något annat
        .
```



Lösning 2:

```
        .
        .
        LDX   #$35          Sätt pekare till tabellen
        LDA   #$15          Sätt varvräknare till antal dataord
        PSHA                Startvärde till stacken
LOOP    LDA   0,X           Hämta data från tabellen
        ANDA  #%11110000    Nollställ bit 3-0
        STA   ,X+           Skriv tillbaka till tabell. Öka pekare
        DEC   0,SP          Minska varvräknare på stacken med ett
        BNE   LOOP          Fortsätt med nästa värde
        PULA                (eller LEASP 1,SP) Återställ stackpekare
        .                   Färdigt, fortsätt med något annat
        .
```

Lösning 3:

```
        .
        .
        LDX   #$35          Sätt pekare till tabellen
LOOP    LDA   0,X           Hämta data från tabellen
        ANDA  #%11110000    Nollställ bit 3-0
        STA   ,X+           Skriv tillbaka till tabell. Öka pekare
        CMPX  #$4A          Kontrollera om pekaren har passerat hela tabellen
        BNE   LOOP          Nej, fortsätt med nästa värde i tabellen
        .                   Färdigt, fortsätt med något annat
        .
```

X.15b)
Lösning

```
MASKBIT  PSHA
         PSHC
         PSHX
         LDX   #$40      Startadress
LOOP     LDA   0,X       Hämta byten
         ANDA  #$7F      Nollställ ms bit
         STA   ,X+       Skriv tillbaks byte
         CMPX  #$60      Klart?
         BNE   LOOP      Om inte: fortsätt
MASKBIT  PULX
         PULC
         PULA
         RTS
```

X.16
Lösning

Numrera elementen från noll och uppåt, dvs första elementet har nummer 0, andra har nummer 1 osv.

Adress	-
Tabell+0	Första
Tabell+1	Andra
Tabell+2	Tredje
Tabell+3	Fjärde
Tabell+4	Femte
Tabell+5	Sjätte
Tabell+6	Sjunde
Tabell+7	Åttonde

```
         .
         .
LDX      #Tabell    Sätt pekare till tabellen
LDA      3,X        Hämta fjärde elementet från tabellen till A-reg
ADDA     5,X        Addera sjätte elementet från tabellen till fjärde elementet i A-reg
STA      4,X        Placera summan som femte element i tabellen
         .
         .
```

X.17
Lösning 1:

```
        .
LDA     SEL     Läs variabeln SEL från minnet
CMPA    #$03    Kontrollera om SEL > 3 eller SEL = 3
BHI     $78     SEL är > 3. Hoppa till adressen 78(16) enligt graf
BEQ     $50     SEL är = 3. Hoppa till adressen 50(16) enligt graf
TSTA            Kontrollera om SEL = 0
BEQ     $24     SEL är = 0. Hoppa till adressen 24(16) enligt graf
CMPA    #$01    Kontrollera om SEL = 1
BEQ     $3C     SEL är = 1. Hoppa till adressen 3C(16)
BRA     $88     SEL måste vara = 2. Hoppa till adressen 88(16)
        .
        .
```

I lösning 2 antas att hoppadresserna finns lagrade i en tabell i minnet på adressen JUMPTAB.

Adress	Data (Hex)
	-
JUMPTAB	24
JUMPTAB+1	3C
JUMPTAB+2	88
JUMPTAB+3	50

Lösning 2:

```
        .
LDA     SEL     Läs variabeln SEL från minnet
CMPA    #$03    Kontrollera om SEL > 3
BHI     $78     SEL är > 3. Hoppa till adressen 78(16) enligt graf

LDX     #JUMPTAB  SEL är ≤ 3. Sätt adress till tabell med hoppadresser
LDA     A,X       Hämta hoppadress till A från JUMPTAB
PSHA              Skriv hoppadressen i A på stack
PULX              Hämta hoppadressen på stacken till X
JMP     0,X       Hoppa till den adress som hämtades från JUMPTAB
        .
        .
        .
```

Istället för att använda stacken för att kopiera hoppadressen med "push" och "pull" i lösning 2 kan man mellanlagra den på en adress i minnet.

X.18

Lösning

```
LDA       #$85      Talet 85₁₆ till A
CMPA      #W        Bilda 85₁₆ - W
B(Villkor) Hopp
```

Storleksjämförelse görs alltså mellan 85_{16} = 133 och W.

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

Om det villkorliga hoppet är:

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

b) BHS (Higher or Same) utförs hoppet om $133 \geq W$, dvs $W \leq 133$
Svar: $0 \leq W \leq 133$

c) BLS (Lower or Same) utförs hoppet om $133 \leq W$, dvs $W \geq 133$
Svar: $133 \leq W \leq 255$

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

Alla hoppvillkoren i e - h avser tal med inbyggt tecken (2-komplementrepresentation).
För 8-bitars tal med inbyggt tecken (2-komplementrepresentation) gäller: $-128 \leq W \leq +127$
Talet 85_{16} tolkas som det negativa talet $-(100_{16} - 85_{16}) = -7B_{16} = -123$

Om det villkorliga hoppet är:

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

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

f) BGE (Greater or Equal) utförs hoppet om $-123 \geq W$, dvs $W \leq -123$

W skall alltså tillhöra intervallet [-128, -123] Svar: $128 \leq W \leq 133$

g) BLE (Less or Equal) utförs hoppet om $-123 \leq W$, dvs $W \geq -123$

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

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

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

Lösning: X.19

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   EXIT      Siffra 0-9
        ADDA  #$07      Bilda ASCII-tecken för bokstav A-F
EXIT    PULC
        RTS
FEL     LDA   #$FF      Signalera att talet för stort med A=FF16
        BRA   EXIT
```

Alt 2:

```
HEXASC  PSHX
        PSHC
        CMPA  #$0F      Kontrollera övre gräns
        BHI   FEL       Talet större än övre gräns
        LDX   #ASCTAB   Sätt pekare till tabell med samtliga 16 ASCII-tecken 0-F
        LDA   A,X       Hämta ASCII-tecken från tabell
EXIT    PULC
        PULX
        RTS
FEL     LDA   #$FF      Signalera att talet för stort med A=FF16
        BRA   EXIT

ASCTAB  FCB   $30       ASCII-tecken för 0
        FCB   $31       -"-          1
         .
         .
        FCB   $39       -"-          9
        FCB   $41       -"-          A
         .
         .
        FCB   $46       -"-          F
```

Lösning: X.20

```
BITZERO  PSHA                Spara register på stack
         PSHC
         PSHY

         CMPA  #$07          Kontrollera övre gräns för bitnummer
         BHI   EXIT          Bitnumret är större än 7

         LDY   #MSKTAB       Sätt pekare till tabell med masker
         LDA   A,Y           Hämta bitmask från tabell till A-reg
         ANDA  0,X           Nollställ bit i dataordet
         STA   0,X           Skriv tillbaka dataordet till minnet

EXIT     PULY                Hämta sparade register från stack
         PULC
         PULA
         RTS

MSKTAB   FCB   %11111110     Mönster för nollställning av bit nummer  0
         FCB   %11111101                    -"-                        1
         FCB   %11111011                    -"-                        2
         FCB   %11110111                    -"-                        3
         FCB   %11101111                    -"-                        4
         FCB   %11011111                    -"-                        5
         FCB   %10111111                    -"-                        6
         FCB   %01111111                    -"-                        7
```