Övningsuppgifter kapitel 11

Alla uppgifter är avsedda att göras för processorn CPU12.

Tumregel vid skrivning av subrutiner i assemblerspråk:

När det i texten explicit uttrycks att något/några registers innehåll ej får ändras skall dessa sparas och återställas av subrutinen. Med "register" avses då också flaggregistret CC. Dessa registers innehåll är då relevanta för den fortsatta exekveringen av det/de program som kan använda subrutinen. Detta gäller självfallet bara om subrutinen använder dessa register.

11.1 Vilka värden får C-, V- Z- och N-flaggorna vid följande beräkningar? 8 bitars ordlängd används och tecknen är inbyggda i talen (2-komplementform). Talen är givna på decimalform.

a) 25 + 37 **b)** 112 + 59 **c)** 37 - 25 **d)** 59 –112

e) -35 - 43 **f)** 109 - (-33) **g)** -38 + (-96) **h)** 96 - (-38)

11.2 Antag att instruktion ADDA #$33 står i tur att exekveras. Före exekveringen är de fyra minst signifikanta bitarna i CC-registret (Condition Code Register) 0 och A-registret har värdet

a) A = 39H **b)** A = 77H **c)** A = CDH **d)** A = F0H

Ange innehållet, i binär form, i de fyra minst signifikanta bitarna i CC-registret när instruktionen exekverats.

11.3 Antag att instruktion SUBA #$84 står i tur att exekveras. Före exekveringen är de fyra minst signifikanta bitarna i CC-registret (Condition Code Register) 0 och A-registret har värdet

a) A = A0H **b)** A = 84H **c)** A = 10H **d)** A = 0

Ange innehållet, i binär form, i de fyra minst signifikanta bitarna i CC-registret när instruktionen exekverats.

11.4 Handassemblera följande sekvens av mnemonics.

a)
```
LDAA  $4000
DECA
STAA  $4002
```

b)
```
LDAA   $8000
TFR    A,B
COMB
STAB   $8001
```

11.5 Disassemblera följande hexadecimala, konsekutiva minnesinnehåll. Första byten är en Opkod. Startadress = 1200H

a) 87, C7, B4, 00, 20, FA, 00, 22

b) B7, 01, 7A, 10, 00, 47, BA, 10, 00

c) B6, 83, 64, B1, 19, 18, 27, 04, 97

11.6 Förutsätt en cykeltid av 0,125 μs (klockfrekvensen är således 8 MHz) och ange tiden det tar att exekvera följande instruktioner.

a) NOP **b)** LDAA #data **c)** LDAA adr

d) ASLA **e)** CMPA adr **f)** ABA

11.7 Med transferinstruktionen TFR r_1,r_2 flyttar man data mellan register i CPU12. För vissa vanliga flyttningar mellan register har Motorola skapat egna mnemoniska assemblerkoder. Man finner dem i instruktionslistan. Vilka är det?

11.8 Med EXG r_1,r_2 byter man innehållen mellan två register. För vissa vanliga byten mellan register har Motorola skapat egna mnemoniska assemblerkoder. Man finner dem i instruktionslistan. Vilka är det?

11.9 Med vilka instruktioner kan man kopiera en byte från 6892H till 1010H.

11.10 Följande register- och minnesinnehåll kända: PC = 4482H, A = 17H, minnesord: M(39H) = 7FH, M(42H) = 08H

just innan instruktionssekvensen

```
LDAA   #10
ADDA   $0042
STAA   $0039
```

exekveras. Ange därefter innehållet i A- och PC-registren. Vad innehåller C-flaggan?

11.11 Hur kan man göra ett aritmetiskt vänsterskift av D-registret.

11.12 Med vilka instruktioner kan man 2-komplementera innehållet i D-registret.

11.13 CPU12 har instruktionerna DEC resp INC för att minska resp öka operandvärden med 1. Den har även instruktioner för att minska resp öka värden i de register som oftast innehåller adresser (indexregister) med 1. Man finner dem i instruktionslistan. Vilka är det?

11.14 Med instruktionen ANDCC resp ORCC kan man nollställa resp ettställa individuella flaggor i flaggregistret hos CPU12. För att nollställa resp ettställa ofta använda flaggor har Motorola skapat egna mnemoniska assemblerkoder. Man finner dem i instruktionslistan. Vilka är det?

11.15 CPU12 har instruktionerna CMP för att jämföra två operandvärden. Den har även instruktioner för att jämföra adresser, dvs värden i register för adresser (indexregister). Man finner dem i instruktionslistan. Vilka är det?

11.16 CPU12 har även en instruktion för att jämföra de två värden som finns i ackumulatorerna A och B. Man finner den i instruktionslistan. Vilken är det?

11.17 Skriv en instruktionssekvens som räknar antalet 1-ställda bitar i D-registret. Resultatet läggs i A-registret.

11.18 Skriv en instruktionssekvens som adderar två 8-bitars tal på adresserna C500H resp. C501H och därefter placerar resultatet i C502H. ”Overflow” kan ej inträffa.

11.19 Skriv en instruktionssekvens som adderar två sextonbitars tal på adresserna C500:C501H resp. C502:C503H och därefter placerar resultatet i C504:C505H. ”Overflow” kan ej inträffa.

11.20 Skriv en instruktionssekvens som kopierar innehållet på minnesadresserna 1000H-100FH till 2000H-200FH. Lös uppgiften utan att använda MOVB-instruktionen. Använd register X och Y som "pekare", samt register A för data.

11.21 Ange, i lämplig tabellform, stackpekarens värde efter varje instruktion i följande sekvens. Ange också innehållet i de tre översta orden på stacken. De initiala värdena är:

SP=1098H, M(SP) = 40H, M(SP+1) = 98H, M(SP+2) = FFH.

```
$800C   PSHY
$800E   BSR     SUBR
$8010   PULY

SUBR    PSHX
        PULA
        PULB
        RTS
```

11.22 Ge en sekvens av instruktioner som definierar en stack med början på adress F120H, samt placerar operanderna 06H, 35H och 28H på stacken. Vad är stackpekarens innehåll när sekvensen exekverats?

11.23 Skriv en subrutin som nollställer (maskerar) bit 7 av orden i adressområdet A300H-A5FFH. Lös uppgiften genom att

a) använda BCLR-instruktionen **b)** ej använda BCLR-instruktionen

11.24 Skriv en subrutin som adderar två 40-bitars tal utan tecken, N och M. Resultatet skall ersätta N i minnet, dvs man kan se det som
N := N+M.

Indata: X = pekare till N LSB (minst signifikanta byte),
Y = pekare till M LSB.

N,M och resultatet organiseras i minnet enligt:

hög adress .... låg adress
MSB .... LSB

11.25 Skriv en subrutin som omvandlar ett NBCD-kodat tal, som består av två decimala siffror, till motsvarande binära tal.

Indata A = NBCD-kodat tal
Utdata A = binärt tal

11.26 Skriv en subrutin "CHALPHA" som kontrollerar om ett utpekat minnesord tillhör ASCII-intervallet [A-Z] eller [a-z]. Rutinen ska meddela resultatet genom att returnera C=0 för godkänt tecken, C=1 annars. Som inparameter till rutinen skickas en pekare till tecknet (X-registret). Pekaren får inte ändras av rutinen.

11.27 Skriv en subrutin som genomsöker minnesarean 2000H-2FFFH efter en byte med innehållet 55H. Rutinen ska lämna adressen till den första förekomsten av 55H i X-registret. Om hela arean genomsökts utan att någon byte med detta innehåll hittats skall X-registret innehålla 3000H.

11.28 Skriv en subrutin som omvandlar en NBCD-kodad siffra till motsvarande "segmentmönster" för en 7-segments display enligt följande:

0	FCH	2	DAH	4	66H	6	BEH	8	FEH
1	60H	3	F2H	5	B6H	7	E0H	9	F6H

Segmentmönstren skall finnas i en tabell som börjar på adress SEGTAB.

Indata: A = NBCD-siffra
Utdata: A = segmentmönster

11.29 Skriv en subrutin "ADDVEC" som adderar komponenterna hos två vektorer V(i) och W(i), placerar resultatet i V(i). Vektorernas komponenter utgörs av 1-bytes positiva tal, eventuellt spill ignoreras.

Indata: X = pekare till V(0).
Y = pekare till W(0)
B = Antal komponenter i vektorerna.

11.30 Skriv en subrutin DOTPRD, som beräknar skalärprodukten av två 2-dimensionella vektorer V och W. Varje komponent av vektorerna består av ett 1-byte positivt heltal. Adresser till V,W och deras skalärprodukt DTPD, tillhandahålles via stacken. Ett anrop av DOTPRD sker på följande sätt:

```
LDX     #V              startadress vektor V
PSHX
LDX     #W              startadress vektor W
PSHX
LDX     #DTPD           startadress skalärprodukt
PSHX                     ... på stacken
      :
  (ev ändring av register X)
   :
BSR     DOTPRD
LEAS    6,SP            balansera stack
```

Eventuellt overflow ignoreras. Inget register får påverkas.

11.31 Skriv en rutin "SORT" som sorterar positiva heltal (1-bytes) i sjunkande ordning. Rutinen anropas enligt:

```
LDX     #SCRATCH        10 bytes tillgänglig minnesarea
LDY     #TABLE          10 bytes tabell att sortera
PSHX
PSHY
      :
(ev ändring av register X ochY)
  :
BSR     SORT
LEAS    4,SP
```

Tabellen, som alltså består av tio tal, skall efter sortering inledas med det största talet på adressen TABLE+0, det näst största talet på adressen TABLE+1 osv.

Följande rutin ska användas av "SORT":

```
*************************************************
* Subroutine FndMax
* finds the largest entry in table
* Input:        Y = ptr to table
*               B = # of bytes in table
* Output:       A = largest data byte
*               B = within table, (offset from Y).
*************************************************

FndMax  PSHB            make space and init position
        DECB            adjust for ptr
        LDAA    B,Y     get last entry
FndM1   DECB            next entry...
        CMPA    B,Y     greatest so far ?
        BHI     FndM2   if so: do next
        LDAA    B,Y     substitute greatest
        STAB    ,SP     substitute position
FndM2   TSTB            table done ?
        BNE     FndM1   if not: continue
        PULB            get position
        RTS             exit
```

11.32 Vad är det för fel på följande subrutin?

```
SUBRTN  PSHA
        LDAA    TAL_1
        ADDA    TAL_2
        TFR     A,B
        RTS
```

11.33 OP-koden till instruktionen BRA LABEL har adressen 12AH. Handassemblera instruktionen när LABEL har adress enligt följande:

a) 12FH **b)** 190H **c)** 12AH **d)** 120H **e)** 103

11.34 De villkorliga hoppinstruktionerna föregås vanligen av en instruktion som påverkar flaggor. Hoppinstruktionen testar därefter ett flagguttryck och avgör om den skall hoppa eller ej. Utöver de vanliga villkorliga hoppinstruktionerna finns hos CPU12 även 6 st instruktioner som kombinerar en flaggsättande operation med själva hoppmöjligheten. Man kan säga att varje sådan instruktion ersätter två instruktioner.

a) Fyra av dessa använder sig av ett register som i sammanhanget tänks vara en räknare. Man finner dem i instruktionslistan. Vilka är det? Beskriv vad de gör!

b) Två av dessa riktar sig mot innehållet på en minnesadress och undersöker dess bitar. Man finner dem i instruktionslistan. Vilka är det? Beskriv vad de gör!