

2012-02-14

Assembler- programmering

Övningsuppgifter

2011

Lösningar

(Med reservation för diverse fel!)

1. Hur många "E-klockperioder" använder CPU12 (HCS12) för att köra programsekvensen nedan?

Lösning:

```

                ORG      $1000
                LDAA     #5          1
ALOOP  LDX      #1000      2*5
XLOOP  DBNE     X,XLOOP    3*1000*5
                LBRN     ALOOP     3*5
                DBNE     A,ALOOP    3*5

```

$$* N = 1 + (2 + 3 * 1000 + 3 + 3) * 5 = 15041 \text{ st}$$

* (Simulatorn "tror" att DBNE-istruktionen tar 6 maskincykler

* istället för 3 så den får antalet cykler till 30056 st

$$* N = 1 + (2 + 6 * 1000 + 3 + 6) * 5 = 30056 \text{ st}$$

2. Hur många "E-klockperioder" använder CPU12 (HCS12) för att köra programsekvensen nedan? Översätt assemblerkoden i till maskinkod och visa hur den placeras i minnet. Det skall framgå hur offset beräknas.

Lösning:

```

1000                                ORG      $1000      ~
1000  86 05                          LDAA     #5          1
1002  CE 00 64  ALOOP  LDX      #100      2*6
1005  04 35 FD  XLOOP  DBNE     X,XLOOP    3*100*6    1005 - 1008 = -3 ($FD)
1008  43                                DECA                     1*6
1009  2A F7                          BPL     ALOOP     3*6-2 (Hopp om N=0)
                                           1002 - 100B = -9 ($F7)

```

$$* N = 1 + (2 + 3 * 100 + 1 + 3) * 6 - 2 = 1835 \text{ st}$$

* (Simulatorn "tror" att DBNE-istruktionen tar 6 maskincykler

* istället för 3 så den får antalet cykler till 3635 st

$$* N = 1 + (2 + 6 * 100 + 1 + 3) * 6 - 2 = 3635 \text{ st}$$

3. Hur många "E-klockperioder" använder CPU12 (HCS12) för att köra programsekvensen nedan?

Lösning:

```

                ORG      $1000      ~
                LDX     #-5          2
XLOOP  LDY     #-100      2*5
YLOOP  IBNE     Y,YLOOP    3*100*5
                IBNE     X,XLOOP    3*5
                NOP                     1

```

$$* N = 2 + (2 + 3 * 100 + 3) * 5 + 1 = 1528 \text{ st}$$

* (Simulatorn "tror" att IBNE-istruktionen tar 6 maskincykler

* istället för 3 så den får antalet cykler till 3043 st

$$* N = 2 + (2 + 6 * 100 + 6) * 5 + 1 = 3043 \text{ st}$$

4. Programsekvensen nedan visar ett sätt att överföra inparametern 45H till en subrutin. Visa med en programsekvens hur subrutinen kan hämta inparametern och senare göra ett återhopp till RETURN. Subrutinen SUBRUT skall kunna anropas från flera olika ställen i samma program.

```

      .
      .
      JSR   SUBRUT
      FCB   $45      Inparameter
RETURN .
      .

```

Lösning:

```

SUBRUT LDY   ,SP      Hämta lagrad "återhopsadress" från stacken.
      LDAA  ,Y        Hämta inparametern
      INY                    Ny återhopsadress
      STY   ,SP      Skriv tillbaka återhopsadressen till stacken

```

5. Programsekvensen nedan visar ett sätt att överföra inparametern 53H till en subrutin. Visa med en programsekvens hur subrutinen kan hämta inparametern och senare göra ett återhopp till RETURN.

```

      .
      .
      BRA   SUBCALL
      FCB   $53      Inparameter
SUBCALL JSR   SUBRUT
RETURN

```

Lösning:

```

SUBRUT LDY   ,SP      Hämta lagrad återhopsadress från stacken.
      LDAA  -4,Y      Hämta inparametern

```

6. Skriv i assemblerspråk för CPU12 en subrutin BITCOUNT som räknar antalet ettor i ackumulator A och returnerar detta antal i ackumulator B. Vid återhopp från BITCOUNT får endast ackumulator B och flaggregistret vara förändrade. Programmet skall vara "korrekt" radkommenterat.

Lösning:

```

BITCOUNT PSHA                    Spara på stack
      CLRB                    Nolla 1-räknare
CTLOOP  TSTA                    Reg A tomt?
      BEQ   CNTOUT            Ja, återvänd
      ASLA                    Nej, skifta ut bit i carry
      BCC   CTLOOP            Carry=0? Ja, fortsätt
      INCB                    Carry= 1, öka 1-räknare
      BRA   CTLOOP            Fortsätt
CNTOUT  PULA                    Återställ A-reg
      RTS

```

7. Skriv en subrutin i assemblerpråk för CPU12 som räknar antalet nollor i ackumulator A. Vid återhopp skall ackumulator A innehålla det binära talet för antalet nollor som fanns i registret vid anropet. Endast ackumulator A och register CC får vara förändrade vid återhopp från subrutinen.

Lösning:

ZCOUNT	PSHA		Spara på stack
	COMA		Gör om nollor till ettor
	CLRB		Nolla 1-räknare
CTLOOP	TSTA		Reg A tomt?
	BEQ	CNTOUT	Ja, återvänd
	ASLA		Nej, skifta ut bit i carry
	BCC	CTLOOP	Carry=0? Ja, fortsätt
	INCB		Carry= 1, öka 1-räknare
	BRA	CTLOOP	Fortsätt
CNTOUT	PULA		Återställ A-reg
	RTS		

8. I minnet i ett mikrodatorsystem med CPU12 finns 150 st 8-bitars tal lagrade på adressen DBLOCK och framåt (ökande adress). De lagrade värdena är heltal i intervallet [0,255]. Skriv en subrutin i assemblerpråk för CPU12 som tar reda på hur många av talen som tillhör intervallet [70H,97H]. Antalet värden i detta intervall skall finnas i B-registret vid återhopp. Endast B-registret och flaggregistret får vara förändrade vid återhopp från subrutinen.

Lösning:

ICNT	PSHX		Spara register
	PSHA		
	LDX	#DBLOCK	Pekare till datablock
	LDA	#150	Antal dataord
	STAA	DCNT	Byteräknare
	CLRB		Nollställ träffräknare
LOOP	LDA	1,X+	Hämta tabellvärde
	CMPA	#\$70	Testa undre gräns
	BLO	OUTSIDE	
	CMPA	#\$97	Testa övre gräns
	BHI	OUTSIDE	
	INCB		Träff
OUTSIDE	DEC	DCNT	Minska byteräknare
	BNE	LOOP	Färdigt? Nej!
	PULA		Återställ register
	PULX		
	RTS		Returvärde i B-reg
DCNT	RMB	1	Byteräknare

9. I minnet i ett mikrodatorsystem med CPU12 finns 100 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 för CPU12 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. Inga andra register får vara förändrade vid återhopp från subrutinen.

Lösning:

ICNT2	PSHX		Spara register
	PSHA		
	LDX	#DVECTOR	Pekare till datablock
	LDAA	#100	Antal dataord
	STAA	DCNT	Byteräknare
	CLRB		Nollställ träffräknare
LOOP	LDAA	1,X+	Hämta tabellvärde
	CMPA	#-15	Testa undre gräns
	BLT	OUTSIDE	
	CMPA	#20	Testa övre gräns
	BGT	OUTSIDE	
	INCB		Träff
OUTSIDE	DEC	DCNT	Minska byteräknare
	BNE	LOOP	Färdigt? Nej!
	PULA		Återställ register
	PULX		
	RTS		Returvärde i B-reg
DCNT	RMB	1	Byteräknare

10. I minnet i ett datorsystem med processorn CPU12 finns ett antal 8-bitars tal lagrade i en tabell på adressen DTAB och framåt (ökande adress). Tabellen avslutas med talet FFH. Skriv en subrutin i assemblerpråk för CPU12-processorn som tar reda på hur många av talen som har värdet "1" i bitpositionerna 6, 4 och 1 samt värdet "0" i bitpositionerna 7 och 0. Antalet tal i tabellen som uppfyller detta skall finnas i A-registret vid återhopp. Endast register A och register CC får vara förändrade vid återhopp från subrutinen.

Lösning:

PATCNT1	PSHX PSHB		Spara register
	LDX #DTAB		Pekare till datablock
	CLRA		Nollställ träffräknare
LOOP	LDAB 1,X+		Hämta tabellvärde
	CMPB #\$FF		Slutmarkering?
	BEQ PCNTEX		
	ANDB #%11010011		Maska "dont't care"-bitar 5, 3, 2
	CMPB #%01010010		Testa bitmönster
	BNE LOOP		Ej träff, hämta nästa
	INCA		Träff, öka träffräknare
	BNE LOOP		Hämta nästa
PCNTEX	PULB PULX RTS		Återställ register Returvärde i A-reg

- 11.** I minnet i ett datorsystem med CPU12 finns ett antal 8-bitars tal lagrade i en tabell på adressen DTAB och framåt (ökande adress). Tabellen avslutas med talet FFH. Skriv en subrutin i assemblerspråk för CPU12 som tar reda på hur många av talen som har bit 7 = 1, bit 3 = 0 och bit 0 = 1. Antalet tal i tabellen som uppfyller detta skall finnas i A-registret vid återhopp. Endast register A och register CC får vara förändrade vid återhopp från subrutinen.

Lösning:

PATCNT2	PSHX PSHB		Spara register
	LDX #DTAB CLRA		Pekare till datablock Nollställ träffräknare
LOOP	LDAB 1,X+ CMPB #\$FF BEQ PCNTEX		Hämta tabellvärde Slutmarkering?
	ANDB #%10001001		Maska "dont't care"-bitar 6, 5, 4, 2, 1
	CMPB #%10000001 BNE LOOP		Testa bitmönster Ej träff, hämta nästa
	INCA BNE LOOP		Träff, öka träffräknare Hämta nästa
PCNTEX	PULB PULX RTS		Återställ register Returvärde i A-reg

- 12.** Skriv en subrutin i assemblerpråk för CPU12 som räknar antalet udda 8-bitars dataord i en dataarea i minnet. Vid anrop av subrutinen finns den lägsta adressen till dataarean i X-registret och antalet dataord i dataarean i A-registret. Antalet udda dataord skall finnas i A-registret vid återhopp. Endast A-registret och flaggregistret får vara förändrade vid återhopp från subrutinen.

Lösning:

ODDCNT	PSHX PSHB PSHB		Spara register
	LDX #DTAB CLRA		Pekare till datablock Nollställ träffräknare
LOOP	LDAB 1,X+ CMPB #\$FF BEQ PCNTEX		Hämta tabellvärde Slutmarkering?
	ANDB #%10001001		Maska "dont't care"-bitar 6, 5, 4, 2, 1
	CMPB #%10000001 BNE LOOP		Testa bitmönster Ej träff, hämta nästa
	INCA BNE LOOP		Träff, öka träffräknare Hämta nästa
PCNTEX	PULB PULX RTS		Återställ register Returvärde i A-reg

- 13.** Skriv en subrutin i assemblerpråk för CPU12 som räknar antalet udda 8-bitars dataord i en dataarea i minnet. Vid anrop av subrutinen finns den lägsta adressen till dataarean i X-registret och antalet dataord i dataarean i A-registret. Antalet udda dataord skall finnas i A-registret vid återhopp. Endast A-registret och flaggregistret får vara förändrade vid återhopp från subrutinen.

Lösning:

ODDCNT	PSHX PSHY PSHB		Spara register
	LDY #0 TFR A,B		Nollställ träffräknare Ordräknare till B-reg
LOOP	TSTB BEQ ODDEX		
	LDAA 1,X+ DECB ANDA #1 BEQ LOOP		Hämta dataord Minska ordräknare Bit 0 = 1 i dataord om udda Ej träff, hämta nästa

	INY		Träff, öka träffräknare
	BRA	LOOP	Nästa varv
ODDEX	TFR	Y,A	YL till A-reg (innehåller antal träffar)
	PULB		Återställ register
	PULY		
	PULX		
	RTS		Returvärde i A-reg

- 14.** I minnet i ett datorsystem med CPU12 finns ett antal 8-bitars tal med tecken lagrade i en tabell på adressen DTAB och framåt (ökande adress). Tabellen avslutas med talet 0. Skriv en subrutin i assemblerpråk för CPU12 som tar reda på hur många av de positiva talen i tabellen som är udda. Antalet tal i tabellen som uppfyller detta skall finnas i A-registret vid återhopp. Endast register A och register CC får vara förändrade vid återhopp från subrutinen.

Lösning:

PATCNT3	PSHX PSHB		Spara register
	LDX #DTAB CLRA		Pekare till datablock Nollställ träffräknare
LOOP	LDAB 1,X+ BEQ PCNTEX		Hämta tabellvärde Nollterminering? Ja
	ANDB #%10000001 CMPB #%00000001 BNE LOOP INCA BNE LOOP		Nej, maska "dont't care"-bitar 6, 5, 4, 3, 2, 1 Testa bitmönster (bit7=0 och bit0=1?) Ej träff, hämta nästa Träff, öka träffräknare Hämta nästa
PCNTEX	PULB PULX RTS		Återställ register Returvärde i A-reg

- 15.** Skriv en subrutin PCNT i assemblerspråk för CPU12, som söker igenom en sträng med 8-bitars dataord i minnet och räknar alla dataord där den vänstra halvan bildar ett fyrabitars binärt tal som är mindre än 6, dvs $(b_7b_6b_5b_4)_2 < 6$. Vid anrop av subrutinen skall adressen till strängen finnas i X-registret. Strängen avslutas med datavärdet FFH. Det sökta antalet skall returneras i D-registret. För övrigt får endast flaggregistret vara förändrat vid återhopp. Programmet skall vara ”korrekt” radkommenterat.
- 16.** I minnet i ett datorsystem med CPU12 finns en tabell med 15 st 8-bitars tal lagrade på adressen 2090H och framåt (ökande adress). De lagrade värdena är tal utan tecken. Skriv en subrutin i assemblerpråk för CPU12 som adderar samtliga tal i tabellen. Vid återhopp från subrutinen skall summan av alla tabellvärdena finnas som ett 16-bitars tal i D-registret. Endast register D och CC får vara förändrade vid återhopp från subrutinen.
- 17.** Skriv en subrutin, CCOUNT, i assemblerpråk för CPU12, som tar reda på hur många gånger ASCII-tecknet för bokstaven C förekommer i en nollterminerad textsträng. Vid anrop av subrutinen skall startadressen till textsträngen finnas i X-registret och vid återhopp skall antalet C-tecken finnas i B-registret. ASCII-tecknen i textsträngen är lagrade med udda paritet med paritetsbiten som bit nummer 7. Den avslutande nollan har ingen paritetsbit. Endast register B och register CC får vara förändrade vid återhopp från subrutinen.
- 18.** Skriv en subrutin, AaCNT, i assemblerpråk för CPU12, som tar reda på hur många gånger ASCII-tecknen för bokstäverna A och a förekommer i en nollterminerad textsträng. Vid anrop av subrutinen skall startadressen till textsträngen finnas i X-registret och vid återhopp skall antalet ASCII-tecken finnas i B-registret. ASCII-tecknen i textsträngen är lagrade med jämn paritet med paritetsbiten som bit nummer 7. Endast register B och register CC får vara förändrade vid återhopp från subrutinen.
- 19.** Skriv en subrutin SMALLCNT i assemblerspråk för CPU12 som söker igenom en textsträng med 7-bitars ASCII-tecken och räknar alla ASCII-tecken som motsvarar små bokstäver (a-z). Vid anrop av subrutinen skall adressen till textsträngen finnas i X-registret. Textsträngen avslutas med datavärdet 0. Det sökta antalet skall returneras i D-registret. För övrigt får endast flaggregistret vara förändrat vid återhopp. Ett ASCII-tecken är lagrat i bit6-bit0 på varje adress i textsträngen. Bit7 i textsträngens dataord har okänt värde. Programmet skall vara ”korrekt” radkommenterat.

- 20.** I minnet i ett datorsystem med CPU12 finns en nollterminerad sträng med sju bitars ASCII-tecken. Varje ASCII-tecken har kompletterats med en paritetsbit som åttonde bit (bit nr 7). Skriv en subrutin LCOUNT i assemblerpråk för CPU12 som tar reda på hur många av ASCII-tecknen i strängen som motsvarar stora eller små bokstäver A – Z eller a – z. Antalet bokstäver skall finnas i B-registret vid återhopp. Vid anrop av subrutinen skall startadressen till strängen finnas i X-registret. Endast register B och register CC får vara förändrade vid återhopp från subrutinen.
- 21.** Skriv en subrutin DIGCNT i assemblerpråk för CPU12 som söker igenom en textsträng med 7-bitars ASCII-tecken och räknar antalet tecken som motsvarar hexadecimala siffror. Vid anrop av subrutinen skall adressen till textsträngen finnas i X-registret och vid återhopp skall X-registret innehålla antalet hexadecimala siffror. Textsträngen avslutas med datavärdet 0. Endast X-registret och flaggregistret får vara förändrade vid återhopp. Av bokstäverna i alfabetet anses endast A-F (stora bokstäver) motsvara hexadecimala siffror. Ett ASCII-tecknen är lagrat i bit6-bit0 på varje adress i textsträngen. Bit7 i textsträngens dataord har okänt värde. Programmet skall vara ”korrekt” radkommenterat.
- 22.** Skriv en subrutin DIGCNT i assemblerpråk för CPU12 som söker igenom en textsträng med 7-bitars ASCII-tecken och räknar antalet tecken som motsvarar hexadecimala siffror. Vid anrop av subrutinen skall adressen till textsträngen finnas i X-registret och vid återhopp skall X-registret innehålla antalet hexadecimala siffror. Textsträngen avslutas med datavärdet 0. Endast X-registret och flaggregistret får vara förändrade vid återhopp. Av bokstäverna i alfabetet anses både A-F och a-f motsvara hexadecimala siffror. Ett ASCII-tecknen är lagrat i bit6-bit0 på varje adress i textsträngen. Bit7 i textsträngens dataord för ASCII-tecken har okänt värde. Programmet skall vara ”korrekt” radkommenterat.
- 23.** Skriv en subrutin PCHECK i assemblerpråk för CPU12, som söker igenom en nollterminerad sträng med ASCII-tecken i minnet och kontrollerar att paritetsbiten (b_7) för udda paritet är korrekt. Udda paritet innebär att kodordet inklusive paritetsbiten har ett udda antal ettor. Vid anrop av subrutinen skall startadressen till strängen finnas i X-registret. När ett paritetsfel upptäcks skall återhopp göras direkt med C-flaggan ettställd och adressen till det felaktiga tecknet i X-registret. Om strängen är felfri skall återhopp göras med C-flaggan nollställd. Vid återhopp får endast flaggregistret och X-registret vara förändrat. Du får använda en färdig subrutin, ACNT, som räknar antalet ettor i A-registret och returnerar detta antal i A-registret. Subrutinen ACNT påverkar endast flaggorna och A-registret.
- 24.** Skriv en subrutin CONV i assemblerpråk för CPU12 som söker igenom en nollterminerad textsträng med 7-bitars ASCII-tecken och ändrar alla ASCII-tecken (A-Z) till motsvarande ASCII-tecken (a-z). Vid anrop av subrutinen skall adressen till textsträngen finnas i X-registret. Endast flaggregistret får vara förändrat vid återhopp. Ett ASCII-tecken är lagrat i bit6-bit0 på varje adress i textsträngen. Bit7 i textsträngens dataord har okänt värde.

- 24.** Skriv i assemblerspråk för CPU12 en subrutin ODDPAR som förser en nollterminerad textsträng med 7-bitars ASCII-tecken med udda paritetsbit i bitposition 7. Vid anrop av subrutinen skall adressen till textsträngen finnas i X-registret. Endast flaggregistret får vara förändrat vid återhopp. ASCII-tecknen är lagrade i bit6-bit0 på varje adress i textsträngen. Bit7 i textsträngens dataord har från början värdet noll. Du får använda en färdig subrutin, ACNT, som räknar antalet ettor i A-registret och returnerar detta antal i A-registret. Subrutinen ACNT påverkar endast flaggorna och A-registret. Programmet skall vara ”korrekt” radkommenterat.
- 25.** En subrutin, KRYPT, för ”kryptering” av nollterminerade textsträngar med ASCII-tecken skall skrivas. Vid krypteringen av textsträngarna skall talet 3CH adderas till varje ASCII-tecken i textsträngen och sedan skall bitarna 0, 2, 4 och 6 inverteras. Skriv subrutinen KRYPT i assemblerspråk för CPU12. Vid anrop av subrutinen skall X-registret innehålla minnesadressen till det första ASCII-tecknet i en textsträng. Följande tecken finns på ökande adresser. Efter återhoppet från subrutinen skall textsträngen i minnet vara ”krypterad”. Den avslutande nollan skall inte krypteras. Endast register CC får vara förändrat vid återhopp från subrutinen. För att krypteringsprincipen ovan skall fungera måste en ASCII-kod (förutom 0) förbjudas. Vilken?
- 26.** En subrutin, DEKRYPT, för ”dekryptering” av textsträngar med ASCII-tecken skall skrivas. Vid krypteringen av textsträngarna har talet 3CH adderats till varje ASCII-tecken i textsträngen och sedan har bitarna 0, 2, 4 och 6 inverterats. De krypterade textsträngarna har olika längd och avslutas därför med en byte med värdet 0, som alltså inte är krypterad. Skriv subrutinen DEKRYPT i assemblerspråk för CPU12. Vid anrop av subrutinen skall X-registret innehålla minnesadressen till det första tecknet i en krypterad textsträng. Följande tecken finns på ökande adresser. Efter återhoppet från subrutinen skall textsträngen i minnet vara ”dekrypterad” till vanliga ASCII-tecken. Endast register CC får vara förändrat vid återhopp från subrutinen. För att krypteringsprincipen ovan skall fungera måste en ASCII-kod (förutom 0) förbjudas. Vilken?
- 27.** Skriv en subrutin BYT_PLATS för processorn CPU12, som byter plats på minnesinnehållen i två minnesareor med 16-bitars dataord. Vid anrop av subrutinen finns antalet 16-bitars dataord i vardera minnesarean i A-registret (högst 255 st) och adresserna till de två dataareorna i X- och Y-registren. Endast flaggregistret får vara förändrat vid återhopp från subrutinen. Skriv subrutinen i assemblerspråk för CPU12.
- 28.** Skriv en subrutin SWAP för processorn CPU12, som byter plats på databitarna i ett block i minnet så att $b_7b_6b_5b_4b_3b_2b_1b_0$ ersätts med $b_3b_2b_1b_0b_7b_6b_5b_4$ i hela blocket. Vid anrop av subrutinen finns antalet 8-bitars dataord i blocket i A-registret (högst 255 st) och begynnelseadressen i X-registret. Endast flaggregistret får vara förändrat vid återhopp från subrutinen. Skriv subrutinen i assemblerspråk för CPU12.
- 29.** Skriv en subrutin för CPU12 som matar ut en positiv flank på en av bitarna på utport OUT. Vid anrop av subrutinen innehåller register B bitnumret där flanken skall matas ut. Om flanken skall matas ut på bit nummer 5 så innehåller register B värdet 5 vid anropet. Vid återhopp från subrutinen får endast flaggregistret vara förändrat. Inga färdiga subrutiner finns tillgängliga.

- 30.** I en styrsekvens skall en CPU12-baserad styrenhet i en maskin invänta en positiv flank på en binär signal från en givare, ansluten till bit nr 5 på en inport, med den kända adressen INPORT1. En subrutin PEDGE i programmet för styrenheten skall detektera en positiv flank på en av bitarna på INPORT1. Vid anrop av subrutinen skall ackumulator A innehålla numret för den bit på INPORT1 som man vill undersöka. Övriga bitar på inporten har okända värden. Återhopp från subrutinen skall göras så snart en positiv flank har upptäckts. Endast register CC får vara förändrat vid återhoppet

Ledning: Fundera på hur man kan upptäcka en positiv flank hos en signal. Signalen kan från början ha värdet 0 eller 1. För att välja vilken bit man vill undersöka kan man använda en tabell med bitmasker för de 8 olika fallen.

Rita en flödesplan för subrutinen PEDGE. Skriv sedan subrutinen PEDGE i assemblerspråk för CPU12. Visa hur subrutinanropet ser ut i huvudprogrammet i det aktuella fallet.

- 31.** Styrenheten i en maskin innehåller en CPU12-dator. I en styrsekvens skall styrenheten invänta en negativ flank på en binär signal från en givare, ansluten till bit nr 3 på datorns inport med adress 600H. Skriv en subrutin i assemblerspråk för CPU12 som detekterar en negativ flank på en av bitarna på inporten 600H. Vid anrop av subrutinen skall ackumulator A innehålla numret för den bit på inporten som man vill undersöka. Övriga bitar på inporten har okända värden. Återhopp från subrutinen skall göras så snart en negativ flank har upptäckts. Endast register CC får vara förändrat vid återhoppet. Visa hur subrutinanropet ser ut i huvudprogrammet i det aktuella fallet.

- 32.** Skriv en subrutin i assemblerspråk för CPU12 som matar ut ASCII-tecken (7 bitars ASCII lagrade som 8-bitars ord med mest signifikant bit nollställd) till en skrivare enligt följande beskrivning:

ASCII-tecknen som skall matas ut finns lagrade i minnet med första tecknet på adress C880H och följande tecken på ökande adresser. I de lagrade orden är den mest signifikanta biten (bit 7) nollställd.

Skrivarens dataingång (8 bitar) är ansluten till en av datorns utgångar (8 bitar) på adressen 1000H.

Skrivarens statusutgång är ansluten till bit nummer 7 (mest signifikant) på en av datorns ingångar (8 bitar) med adressen 1001H. Om statusutgången har värdet 1 är skrivaren beredd att ta emot ett ASCII-tecken på sin dataingång. Utmatning av ett dataord (ASCII-tecken) till skrivarens dataingång medför automatiskt att skrivarens statusutgång nollställs. Detta är en egenskap hos skrivaren.

Utmatning av ASCII-tecken skall fortsätta tills ett dataord med värdet 00H läses från minnet. Dataordet med värdet 00H används som slutmarkering och skall inte matas ut till skrivaren. Återhopp från subrutinen skall göras när utmatningen är färdig. Vid återhoppet skall samtliga interna register ha samma värden som vid inhoppet. Subrutinen skall fungera även om första dataordet som läses från minnet är 00H, dvs återhopp skall då ske direkt. Programmet skall vara ”korrekt” radkommenterat.

- 33.** En CPU12-dator skall användas för att styra ett elektroniskt lås. I låsprogrammet skall ingå en subrutin som jämför en sträng med inmatade ASCII-tecken med en annan sträng med 8-bitars ord, som innehåller ”nyckeln” till låset. Bit 7 i varje inmatat ASCII-tecken har okänt värde medan motsvarande bit i ”nyckelsträngen” är noll. Bägge strängarna är lika långa, relativt korta och nollterminerade. Skriv en subrutin i assemblerpråk för CPU12 som jämför de två strängarna och returnerar antal fel i A-registret. Vid anrop av subrutinen skall startadressen till den inmatade strängen finnas i X-registret. Startadressen till ”nyckelsträngen” finns i minnet på adressen KEY.
- 34.** Skriv en subrutin som nollställer bit 7 och inverterar bit 5 i ett block med 16 st 8-bitars dataord i minnet. Adressen till dataordet med lägst adress finns i X-registret vid anrop av subrutinen. Vid återhopp från subrutinen skall samtliga processorregister ha samma innehåll som vid anropet. Assemblerspråk för CPU12 skall användas.
- 35.** I programmeringsspråket C finns den sk switch-satsen med vilken man kan göra flerval. Det innebär att man från ett ställe i programmet kan fortsätta på ett av flera olika ställen. Ett exempel på ett sådant flerval ges nedan.
I en styrenhet för en maskin används CPU12. Ett avsnitt av styrprogrammet skall läsa av värdet CASE (8 bitar utan tecken), som finns på en inport på den symboliska adressen INPORT, och därefter utföra ett hopp till en av sex olika adresser, ADR0-ADR5. Vilken av adresserna som väljs bestäms av värdet på variabeln CASE. Om CASE = 0 utförs hopp till ADR0, om CASE = 1 utförs hopp till ADR1 osv. Om CASE > 5 skall hopp ske till den symboliska adressen ERRADR. De sex olika hoppadresserna skall finnas lagrade i minnet i en tabell med början på adressen 1900H. Vid laddning av programmet i minnet skall också tabellen med hoppadresserna laddas. Skriv ett huvudprogram som först initierar stackpekaren till 1FF0H och sedan läser värdet CASE som finns på inporten på adressen INPORT. Om CASE < 6 utförs hopp till en adress enligt beskrivningen i föregående stycke. Om CASE > 5 utförs hopp till adressen ERRADR. Programmet skall skrivas i assemblerpråk för CPU12 och startadressen skall vara 1000H. Programmet skall utformas som flerval och inte som upprepade tvåval. Subrutinernas startadresser ADR0-ADR5 skall vara 1080H, 1167H, 1275H, 147FH, 14AFH och 18E0H. Radkommentarer skall finnas!
- 36.** Under laborationerna har du skrivit subrutinerna OUTZERO och OUTONE som nollställer resp. ettställer en av styrbitarna för bormaskinen via utporten DRCTRL I subrutinerna uppdateras också en kopia DRCOPY, av det ”styrord” som styr bormaskinen.

Skriv en ny subrutin STPULS som genererar en positiv puls (0-1-0) på en av bitarna på utporten DRCTRL och nollställer samma bit i styrordets kopia på adressen DRCOPY. Vilken bit (0-7) på utporten (och i kopian DRCOPY) som skall påverkas bestäms av register B:s värde vid anropet av subrutinen STPULS. Ifall värdet i B-registret vid anropet är större än 7 skall inget göras. Inga register, utom flaggregistret, får vara förändrade vid återhopp från subrutinen.

Subrutinerna OUTZERO och OUTONE skall inte användas. Assemblerspråk för CPU12 skall användas.

- 37.** CPU12 skall användas i styrenheten för en maskin. Till inport 600H är ett antal givare och switchar anslutna. En operatör skall styra maskinen via switcharna. Huvudprogrammet för maskinstyrningen skall utformas som en evighets slinga där inporten läses av en gång i början på varje varv. Huvudprogrammet skall inledas med att stackpekaren först sätts till värdet 2FFFH. Sedan skall inporten läsas av och beroende på switcharnas och givarnas värden skall en av fyra olika färdiga subrutiner anropas om motsvarande villkor i tabellen nedan är uppfyllt. Efter återhopp från subrutinerna eller om inget av villkoren i tabellen är uppfyllt skall ett nytt varv i slingan påbörjas. Rita en flödesplan som beskriver huvudprogrammet. Skriv huvudprogrammet i assemblyspråk för CPU12. Huvudprogrammets startadress skall vara 1200H. Subrutinernas startadresser antas vara givna. Programmet skall vara ”korrekt” radkommenterat.

Villkor	Inport 600H								Anropa subrutin
	b7	b6	b5	b4	b3	b2	b1	b0	
0	?	?	?	1	?	?	?	1	SUB0
1	?	?	0	1	?	?	1	0	SUB1
2	?	?	?	0	?	?	1	?	SUB2
3	0	0	0	0	1	1	0	0	SUB3

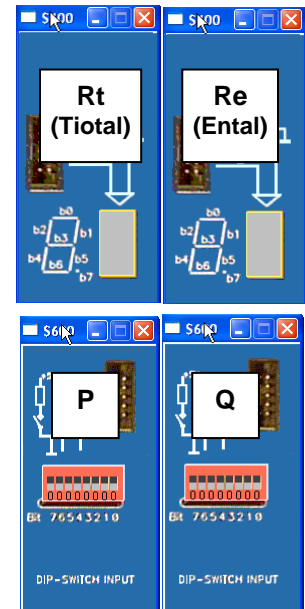
- 38.** Laborationsenheten ML4 innehåller flera yttre enheter som kan kopplas till inportarna och utportarna på laborationsdatoren MC12. De yttre enheterna ML4 INPUT och ML4 OUTPUT som används nedan finns också i simulatören.

Skriv ett program för CPU12 som i en evighets slinga läser två tvåsiffriga NBCD-tal P_{NBCD} och Q_{NBCD} från två 8 bitars inportar, utför additionen $R_{NBCD} = P_{NBCD} + Q_{NBCD}$ och sedan visar summan R_{NBCD} som ett tvåsiffrigt NBCD-tal på två sifferindikatorer via utportar.

Observera att indata tolkas som NBCD-tal när P och Q läses. Om indata inte är NBCD-tal skall EE (Error) skrivas till utportarna. Om summan R är större än 99 skall också EE skrivas till utportarna.

Exempel:

Om $P = 27_{10}$ och $Q = 48_{10}$ läses från strömbrytarna så skall 75_{10} visas på sifferindikatorerna.



Tabellen nedan med segmentkoder och definitioner finns tillgänglig:

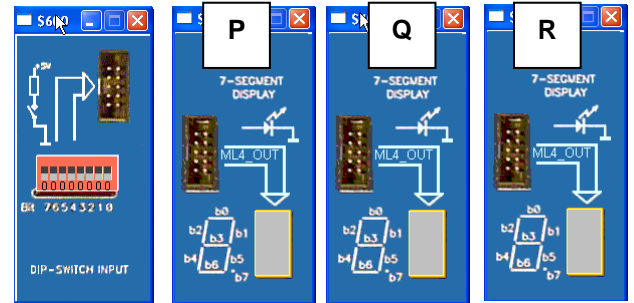
```

DipSwP EQU $600      Adress för strömbrytare P
DipSwQ EQU $601      Adress för strömbrytare Q
OutRt  EQU $400      Adress för Sifferindikator tiotal
OutRe  EQU $401      Adress för Sifferindikator ental
SegCode FCB xx,yy,zz,etc Tabell med segmentkoder för [0,9]
Error  FCB qq        Segmentkod för Error

```

Rita flödesplan och dokumentera programmet!

39. Laborationsenheten ML4 innehåller flera yttre enheter som kan kopplas till inportarna och utportarna på laborationsdatorn MC12. De yttre enheterna ML4 INPUT och ML4 OUTPUT som används nedan finns också i simulatorn.



Skriv ett program för CPU12 som i en evighetsslinga läser två NBCD-siffror P och Q från en 8-bitars inport med strömbrytare, visar siffrorna på två sifferindikatorer via två utportar, utför additionen $R = P + Q$ och till sist skriver summan R till den tredje sifferindikatorn via en utport.

De två NBCD-siffrorna P och Q läses alltså samtidigt som två 4-bitars binära tal från inporten (8 bitar).

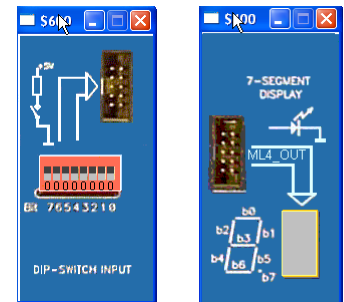
P finns på b_7 - b_4 och Q på b_3 - b_0 . Summan skall placeras i b_3 - b_0 för att omvandlas till segmentkod och skrivs till sifferindikatorn. Om summan är större än nio skall ett E (ERROR) skrivas ut. Du får förutsätta att $P \leq 9$ och $Q \leq 9$.

Tabellen nedan med segmentkoder och definitioner finns tillgänglig:

```
Inport EQU xxxx      Adress för inport
UtportP EQU yyyy     Adress för utport 1
UtportQ EQU zzzz     Adress för utport 2
UtportR EQU wwww     Adress för utport 3
Error EQU pp         Segmentkod för E (Error)
SegCode FCB xx,yy,zz,etc Tabell med segmentkoder för [0,9]
```

40. Laborationsenheten ML4 innehåller flera yttre enheter som kan kopplas till inportarna och utportarna på laborationsdatorn MC12. De yttre enheterna ML4 INPUT och ML4 OUTPUT som används nedan finns också i simulatorn.

Skriv ett assemblerprogram för CPU12 som i en evighetsslinga läser inporten (strömbrytare) och skriver värden till utporten (7-sifferindikator). När bit 7 på inporten är ettställd skall sifferindikatorn släckas helt. När bit 7 på inporten är nollställd skall sifferindikatorn tändas enligt följande beskrivning:



Bit 3-0 på inporten anger vad som skall visas på sifferindikatorn. Om indata är i intervallet [0,9] skall motsvarande decimala siffra visas på sifferindikatorn. Om indata är i intervallet [A,F] skall ett E (Error) visas på sifferindikatorn. Bitarna 6-4 på inporten kan anta vilka värden som helst.

Tabellen nedan med segmentkoder och definitioner finns tillgänglig:

```
Inport EQU xxxx      Adress för inport
Utport EQU yyyy     Adress för utport 1
Error EQU pp         Segmentkod för E (Error)
SegCode FCB xx,yy,zz,etc Tabell med segmentkoder för [0,9]
```