Assemblerprogrammeringsuppgifter för FLEX-processorn

1. I simulatorn för FLEX-datorn kan man ansluta strömbrytarmodulen DIPSWITCH till en inport och sifferindikatorn 7-SEGMENT till en utport.

Skriv ett program som kontinuerligt läser av två 4-bitars tal från de åtta strömbrytarna på DIPSWITCH via inporten FD_{16}, adderar de två talen och visar summan (4 bitar) som en decimal siffra på sifferindikatorn 7-SEGMENT, ansluten till utport FE_{16}.

Från inporten läses två 4-bitars binära tal P och Q samtidigt. P finns på $b_7b_6b_5b_4$ och Q på $b_3b_2b_1b_0$. Summan P+Q skall omvandlas till segmentkod och visas som en siffra 0-9 på sifferindikatorn. Om summan är större än nio skall ett E (ERROR) visas.

Du har tillgång till en tabell med segmentkoder och följande definitioner:

```
Error     EQU  $5D          Segmentkod för E (Error)
SegCode   FCB  xx,yy,zz,etc Tabell med segmentkoder för [0,9]
```

2. I simulatorn för FLEX-datorn kan man ansluta strömbrytarmodulen DIPSWITCH till en inport och sifferindikatorn 7-SEGMENT till en utport.

Skriv ett program som läser av ett värde på strömbrytarna, visar värdet som en hexadecimal siffra på sifferindikatorn och sedan utför en fördröjning. Därefter skall sifferindikatorn släckas, en ny fördröjning utföras och förloppet upprepas från början. Programmet som skall ha startadressen 20_{16} beskrivs nedan. Det finns en färdig subrutin DELAY1S som utför en fördröjning på 1 s.

Programmet skall i tur och ordning:

1. Läsa strömbrytarna på inport FD_{16}.
2. Maska fram bit 3-0 av det inlästa värdet, som motsvarar den önskade fördröjningen i sekunder.
3. Visa detta värde x som tillhör intervallet $[0,F_{16}]$ på sifferindikatorn på utport $FE.
4. Utföra fördröjningen. (Hoppa till DELAY1S x gånger).
5. Släcka sifferindikatorn.
6. Utföra fördröjningen igen. (Hoppa till DELAY1S x gånger).
7. Hoppa tillbaka till 1.

Du har tillgång till en tabell med segmentkoder och följande definitioner:

```
SegCode   FCB  xx,yy,zz,etc  Tabell med segmentkoder för [0,F]
DELAY1S   EQU  wwww         Startadress för Delay-rutinen som utför 1s fördröjning
```

Rita också flödesplan och dokumentera ditt program!

3. Skriv en subrutin CNTONE i assemblerspråk för FLEX-processorn som räknar antalet ettor i register A. Vid återhopp skall register B innehålla antalet ettor som fanns i registret vid anropet. Om register A innehåller 5 st ettor vid anropet så skall alltså register B innehålla talet 00000101_2 vid återhoppet. Endast register B och CC får vara förändrade vid återhopp från subrutinen. Hur skall programmet ändras om man istället vill räkna antalet nollor?

4. Skriv en subrutin ASCBIN i assemblerspråk för FLEX-processorn 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 FF_{16} finnas i register A vid återhopp. Endast register A och CC får vara förändrade vid återhopp från subrutinen.

5. Skriv en subrutin PRINT i assemblerspråk för FLEX-processorn 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 adressen som finns i X-registret och följande tecken på ökande adresser.
- Skrivarens dataingång (8 bitar) är ansluten till utport FD_{16} på FLEX-datorn.
- Skrivarens statussignal READY är ansluten till bit 7 (mest signifikant) på inport FE_{16}. Om signalen READY har värdet 1 är skrivaren beredd att ta emot ett nytt ASCII-tecken.
- Utmatning av ett dataord (ASCII-tecken) till skrivarens dataingång medför automatiskt att skrivaren nollställer signalen READY.
- Utmatning av ASCII-tecken skall fortsätta tills ett dataord med värdet 00_{16} läses från minnet. Dataordet med värdet 00_{16} 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 (utom PC) ha samma värden som vid inhoppet.
- Subrutinen skall fungera även om första dataordet som läses från minnet är 00_{16}, dvs återhopp skall då ske direkt.

6. Skriv en subrutin, CCOUNT, i assemblerspråk för FLEX-processorn, 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 CC får vara förändrade vid återhopp från subrutinen.

7. Skriv en subrutin, AaCNT, i assemblerspråk för FLEX-processorn, 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 CC får vara förändrade vid återhopp från subrutinen.

8. Skriv en subrutin SMALLCNT i assemblerspråk för FLEX-processorn 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 B-registret. För övrigt får endast CC-registret vara förändrat vid återhopp. Ett ASCII-tecken är lagrat i bit6-bit0 på varje adress i textsträngen. Bit7 i textsträngens dataord har okänt värde.

9. I minnet i ett datorsystem med FLEX-processorn finns en nollterminerad sträng med sjubitars ASCII-tecken. Varje ASCII-tecken har kompletterats med en paritetsbit som åttonde bit (bit nr 7). Skriv en subrutin LCOUNT i assemblerspråk för FLEX-processorn 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 CC får vara förändrade vid återhopp från subrutinen. För full poäng på uppgiften skall programmet vara korrekt kommenterat.

10. I minnet i ett datorsystem med FLEX-processorn finns en nollterminerad (= sista dataordet har värdet 0) sträng med sjubitars ASCII- tecken. Varje ASCII-tecken har en paritetsbit för jämn paritet som bit nr 7. Skriv en subrutin HEXCNT i assemblerspråk för FLEX-processorn som tar reda på hur många av ASCII-tecknen i strängen som motsvarar hexadecimala siffror, dvs. 0-9 och A-F (stora bokstäver). Antalet sådana tecken skall finnas i B-registret vid återhopp. Vid anrop av subrutinen skall startadressen till strängen finnas i X-registret. Endast register B och CC får vara förändrade vid återhopp från subrutinen.

11. Skriv en subrutin CONV i assemblerspråk för FLEX-processorn som söker igenom en textsträng med 7- bitars ASCII-tecken och ändrar alla ASCII-tecken (a-z) i strängen till motsvarande ASCII-tecken (A-Z). Textsträngen avslutas med dataordet 0.

Vid anrop av subrutinen skall adressen till textsträngen finnas i X-registret. ASCII-tecknen är lagrade i bit6-bit0 i varje dataord i textsträngen. Bit7 i textsträngens dataord har okänt värde och får inte ändras. Processorns flaggregister får vara förändrat vid återhopp. Övriga register skall vara oförändrade.

12. Skriv en subrutin BINASC i assemblerspråk för FLEX-processorn 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 CC får vara förändrade vid återhopp från subrutinen.

13. Skriv en subrutin ODDPAR i assemblerspråk för FLEX-processorn 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. Det finns en färdig subrutin CNTONE, som får användas. Den räknar antalet ettor i register A och returnerar detta antal i register B. Vid återhopp från CNTONE är endast register B och CC förändrade.

14. En subrutin KRYPT för ”kryptering” av nollterminerade textsträngar med ASCII-tecken skall skrivas. Vid krypteringen av textsträngarna skall talet $3C_{16}$ adderas till varje ASCII-tecken i textsträngen och sedan skall bitarna 0, 2, 4 och 6 inverteras. Skriv subrutinen KRYPT i assembler-språk för FLEX-processorn. 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 krypterings-principen ovan skall fungera måste en ASCII-kod (förutom 0) förbjudas. Vilken?

15. En subrutin DEKRYPT för ”dekryptering” av textsträngar med ASCII-tecken skall skrivas. Vid krypteringen av textsträngarna har talet $3C_{16}$ 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 FLEX-processorn. 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.

16. I ett datorsystem med FLEX-processorn behövs en subrutin, BLKASC, som omvandlar ett block med 8-bitars dataord i minnet till en nollterminerad sträng med ASCII-tecknen för hexadecimala siffror. Omvandlingsprincipen framgår av följande exempel:

Minnesblocket 10110101, 01111000, 01100010, 10100011.

ger ASCII-strängen 42_{16}, 35_{16}, 37_{16}, 38_{16}, 36_{16}, 32_{16}, 41_{16}, 33_{16}, 0

Skriv subrutinen BLKASC. Vid anrop av subrutinen skall startadressen till minnesblocket resp. ASCII-strängen finnas i X-registret resp. i minnet på adressen ADATA och antalet ord i datablocket i B-registret. Av registren får endast flaggregistret får vara förändrat vid återhopp.

17. I minnet i en FLEX-dator finns 50 st 8-bitars tal lagrade på adressen DBLOCK och framåt (ökande adress). De lagrade värdena är heltal i intervallet [0,255]. Skriv en subrutin ICNT1 i assemblerspråk för FLEX-datorn som tar reda på hur många av talen som tillhör intervallet [75_{16},90_{16}]. Antalet värden i detta intervall skall finnas i B-registret vid återhopp. Endast B- och CC-registret får vara förändrat vid återhopp från subrutinen.

18. I en FLEX-dators minne finns en tabell med 40 st 8-bitars tal lagrade på adressen DVECTOR och framåt (ökande adress). De lagrade värdena är tal med tecken och tillhör därför intervallet [-128, 127]. Skriv en subrutin ICNT2 i assemblerspråk 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.

19. I en FLEX-dators minne finns en tabell med 50 st 8-bitars tal lagrad. De lagrade värdena är tal utan tecken. Skriv en subrutin TABSUM i assemblerspråk för FLEX-datorn som adderar samtliga tal i tabellen. Vid återhopp från subrutinen skall summan av alla tabellvärdena finnas som ett 16-bitars tal i A- och B-registren med mest signifikant del i A-registret. Endast register A, B och CC får vara förändrade vid återhopp från subrutinen. Vid anrop av subrutinen skall startadressen till tabellen finnas i X-registret.

20. I minnet i ett datorsystem med FLEX-processorn finns ett antal 8-bitars tal lagrade i en tabell på adressen DTAB och framåt (ökande adress). Tabellen avslutas med talet FF_{16}. Skriv en subrutin PATCNT i assemblerspråk för FLEX-processorn 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 CC får vara förändrade vid återhopp från subrutinen.

21. Skriv en subrutin BLKMAN 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 får endast CC-registret vara förändrat. Assemblerspråk för FLEX-processorn skall användas.

22. I minnet i ett datorsystem med FLEX-processorn 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 ODDCNT i assemblerspråk för FLEX-processorn 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.

Ledning: Man vet här vad bit 0 och 7 skall ha för värden.

23. Skriv en subrutin EXCHG för FLEX-processorn, som byter plats på minnesinnehållen i två lika stora minnesareor med 8-bitars dataord. Vid anrop av subrutinen finns antalet dataord i vardera minnesarean i A-registret (högst 50 st) och adresserna till de två dataareorna i X-registret och på minnesadressen ADATA. Adressen till den andra dataarean finns alltså på adressen ADATA i minnet. Vid återhopp från subrutinen får inga register eller innehållet på adressen ADATA vara förändrade.

24. Skriv en subrutin SWAP för FLEX-processorn, 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 50 st) och begynnelseadressen i X-registret. Vid återhopp från subrutinen får endast CC-registret vara förändrat. Rita en flödesplan för subrutinen och skriv den i assemblerspråk.

25. Skriv en subrutin PCNT i assemblerspråk för FLEX-processorn, 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 FF_{16}. Det sökta antalet skall returneras i B-registret. Endast B- och flaggregistret får vara förändrat vid återhopp.

26. Styrenheten i en maskin innehåller en FLEX-dator. I en styrsekvens skall styrenheten invänta en negativ flank på en binär signal från en givare, ansluten till bit nr 3 på inport FE_{16}.

Skriv en generell subrutin NEDGE i assemblerspråk för FLEX-processorn som detekterar en negativ flank på en av bitarna på inporten FE_{16}. Vid anrop av subrutinen skall register A innehålla numret för den bit på inport FE_{16} som man vill undersöka. Övriga bitar på inporten FE_{16} har okända värden. Återhopp från subrutinen skall göras så snart en negativ flank har upptäckts. Endast register CC får vara förändrat vid återhoppet.

Ledning: 1. Tänk på att signalen från början kan ha värdet 0 eller 1.

2. Man kan använda en tabell med bitmasker för de 8 olika fallen (bit 0-7).

Visa också hur subrutinanropet ser ut i huvudprogrammet i det aktuella fallet.

Hur skall programmet ändras om man istället vill detektera en positiv flank?

27. I en styrenhet för en maskin används FLEX-datorn. Ett avsnitt av styrprogrammet skall läsa av värdet CASE (8 bitar), som finns på inporten FD_{16}, och därefter utföra en av åtta olika subrutiner, SUB0-SUB7. Vilken subrutin som utförs bestäms av värdet på variabeln CASE. Om CASE = 0 utförs SUB0, om CASE = 1 utförs SUB1 osv. Om CASE > 7 skall ingen av subrutinerna utföras. Adresserna till de åtta olika subrutinerna skall finnas lagrade i minnet i en tabell med början på adressen $C0_{16}$. Vid laddning av programmet i minnet skall också tabellen med subrutinadresserna laddas.

Skriv ett huvudprogram som först initierar stackpekaren till FC_{16} och sedan läser värdet som finns på inporten FD_{16} (CASE). Därefter anropas en av subrutinerna enligt ovan om CASE < 8. Programmet utformas som en evighetsslinga. Det skall skrivas i assemblerspråk för FLEX-processorn och startadressen skall vara 0. Programmet skall utformas som flerval och inte som upprepade tvåval.

Subrutinernas startadresser skall vara 80_{16}, 67_{16}, 75_{16}, 52_{16}, 90_{16}, $B9_{16}$, AF_{16} och $E0_{16}$.

28. En FLEX-processor skall användas i styrenheten för en maskin. Till inport FE_{16} är ett antal givare och switchar anslutna. En operatör skall styra maskinen via switcharna. Huvudprogrammet för maskinstyrningen skall utformas som en evighetsslinga där inporten läses av en gång i början på varje varv. Huvudprogrammet skall inledas med att stackpekaren först sätts till värdet FC_{16}. 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 och skriv huvudprogrammet i assemblerspråk för FLEX-processorn. Huvudprogrammets startadress skall vara 20_{16}. Subrutinernas startadresser antas vara givna.

Villkor	Inport FE_{16}								Anropa subrutin
	b7	b6	b5	b4	b3	b2	b1	b0	
1	?	?	?	1	?	?	?	1	SUB1
2	?	?	0	1	?	?	1	0	SUB2
3	?	?	?	0	?	?	1	?	SUB3
4	0	0	0	0	1	1	0	0	SUB4

29. En FLEX-dator skall användas för att styra ett elektroniskt lås. I låsprogrammet skall ingå en subrutin KEYCMP 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 assemblerspråk för FLEX-processorn 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 KSTRING. Endast register A och CC får vara förändrade vid återhopp från subrutinen.

Tabell 1 Assemblatordirektiv (Pseudoinstruktioner)

Direktiv	Förklaring
ORG N	Placerar den efterföljande koden med början på adress N. (ORG för ORiGin = ursprung)
L RMB N	Avsätter N bytes i följd i minnet (utan att ge dem värden), så att programmet kan använda dem. Följden placeras med början på adressen L. (RMB för Reseve Memory Bytes)
L EQU N	Ger symbolen L konstantvärdet N. (EQU för EQUates = beräknas till)
L FCB N1, N2	Avsätter en byte för varje argument i följd i minnet. Respektive byte ges konstantvärdet N1, N2 etc. Följden placeras med början på adressen L. (FCB för Form Constant Byte)
L FCS ”ABC”	Avsätter en byte för varje tecken i teckensträngen ”ABC” i följd i minnet. Respektive byte ges ASCII-värdet för A B C, etc. Följden placeras med början på adressen L. (FCS för Form Character String)

Tabell 2 7-bitars ASCII

0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1	$b_6b_5b_4$ $b_3b_2b_1b_0$
NUL	**DLE**	**SP**	**0**	**@**	**P**	**`**	**p**	0 0 0 0
SOH	**DC1**	**!**	**1**	**A**	**Q**	**a**	**q**	0 0 0 1
STX	**DC2**	**“**	**2**	**B**	**R**	**b**	**r**	0 0 1 0
ETX	**DC3**	**#**	**3**	**C**	**S**	**c**	**s**	0 0 1 1
EOT	**DC4**	**$**	**4**	**D**	**T**	**d**	**t**	0 1 0 0
ENQ	**NAK**	**%**	**5**	**E**	**U**	**e**	**u**	0 1 0 1
ACK	**SYN**	**&**	**6**	**F**	**V**	**f**	**v**	0 1 1 0
BEL	**ETB**	**‘**	**7**	**G**	**W**	**g**	**w**	0 1 1 1
BS	**CAN**	**(**	**8**	**H**	**X**	**h**	**x**	1 0 0 0
HT	**EM**	**)**	**9**	**I**	**Y**	**i**	**y**	1 0 0 1
LF	**SUB**	*****	**:**	**J**	**Z**	**j**	**z**	1 0 1 0
VT	**ESC**	**+**	**;**	**K**	**[Ä**	**k**	**{ä**	1 0 1 1
FF	**FS**	**,**	**<**	**L**	**\Ö**	**l**	**|ö**	1 1 0 0
CR	**GS**	**-**	**=**	**M**	**]Å**	**m**	**}å**	1 1 0 1
S0	**RS**	**.**	**>**	**N**	**^**	**n**	**~**	1 1 1 0
S1	**US**	**/**	**?**	**O**	**_**	**o**	**RUBOUT (DEL)**	1 1 1 1