Laboration nr 4 behandlar

Assemblerprogrammering

Följande upp gifter ur ”Arbetsbok för DigiFlisp” ska vara utförda som förberedelse för laborationen. Du ska på begäran av laborationshandledare redogöra för dessa.
(Signera själv i rutorna nederst när motsvarande uppgifter är utförda!)

Uppg.	16.8 16.9	16.10 16.11	16.16
Sign.			

Hemuppgifter, i detta PM, inför som ska vara utförda innan laborationen påbörjas.

Hem-Uppgifter	4.1	4.2

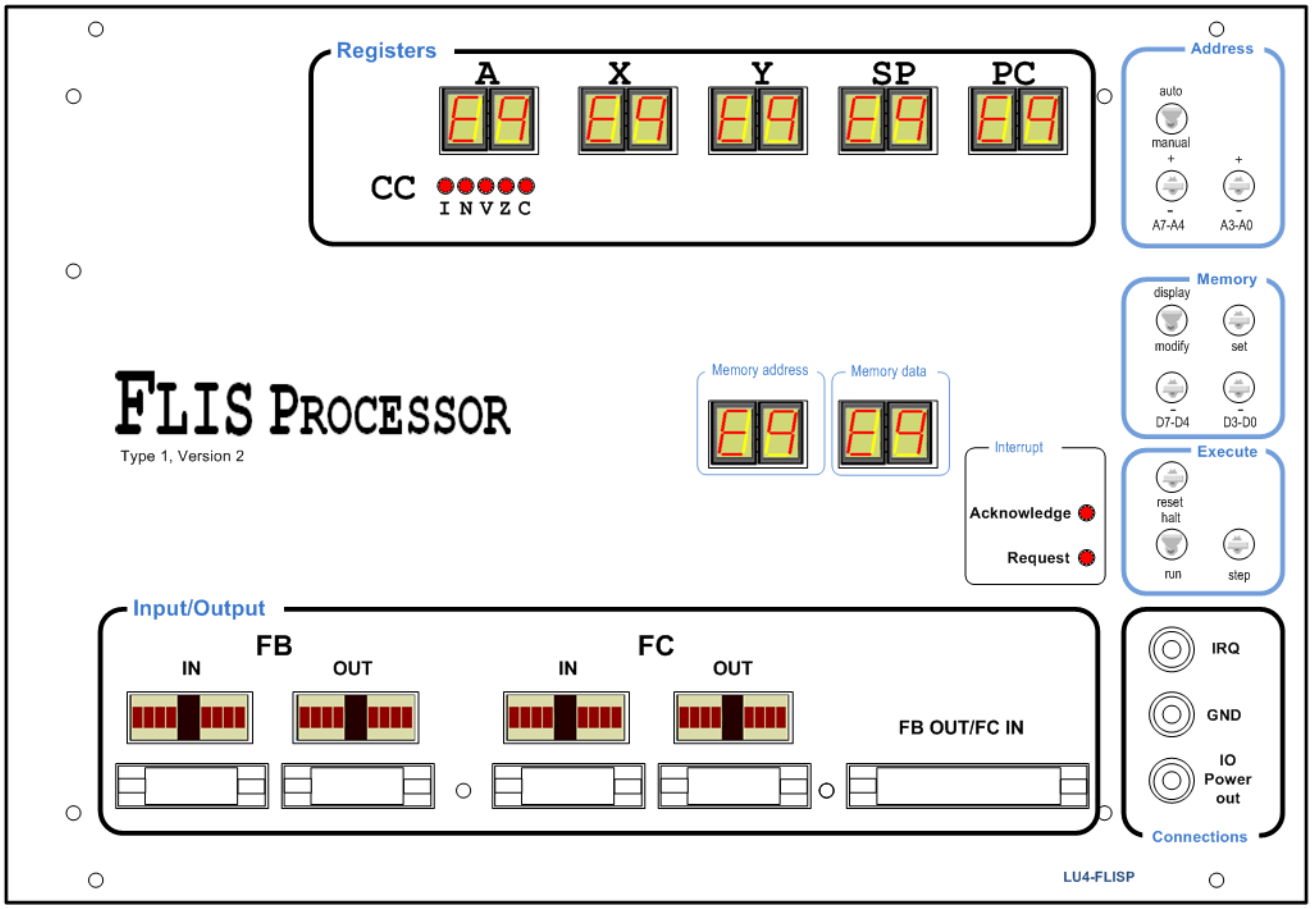
Följande laborationsuppgifter skall redovisas för en handledare för godkännande under laborationen.
(Handledare signerar!)

Laborations-uppgift	4.1	4.2	4.3	4.4
Sign.				

Beskrivning av laborationssystemet

Laborationssystemets panel är indelad i olika sektioner:

- Registers, visar innehållet i FLIS-processorns register.
- Input/Output, anslutningar av externa enheter till FLIS- processorns, här visas också de värden som för tillfället finns hos portarna.
- Address
  auto - minnesenhetens adressväljare (Memory address) följer programräknaren PC.
  manual - minnesenhetens adressväljare sätts med hjälp av omkopplarna A7-A4, de fyra mest signifikanta adressbitarna och A3-A0, de fyra minst signifikanta adressbitarna.
- Memory
  display - innehållet i minnesenhetens dataindikator (Memory data) ges av minnesenhetens adressväljare
  modify - minnesenhetens dataindikator sätts med hjälp av omkopplarna D7-D4/D3-D0
  set - då omkopplaren står i modify-läge förs innehållet i dataindikatorn in i minnet på den adress som anges av adressväljaren.
- Execute, här manövreras FLIS-processorn.
  reset- processorn utför ett återstartsförlopp.
  halt- I detta läge kan ett program utföras instruktionsvis med omkopplaren step.
  run- i detta läge exekverar processorn instruktioner kontinuerligt och exekveringshastigheten (3 olika) kan väljas med omkopplaren step.
- Interrupt, indikerar om en avbrottsbegäran (Request) finns på FLIS-processorns avbrottsingång. Dessutom indikeras (Acknowledge) då FLIS-processorn utför avbrottshantering.
- Connections, anslutningar, IRQ är direkt kopplad till FLIS-processorns avbrottsingång. IO Power out och GND kan användas för att strömförsörja yttre enheter.



Nedladdning av program

ETERM har inbyggd funktion för att underlätta nedladdning av program och data till laborationsdatorn. Funktionen filtrerar en fil av typen `.hwflisp` (skapas då du assemblerar din källtext) och skickar setMemory- och setRegister- kommandon till laborationsdatorn.

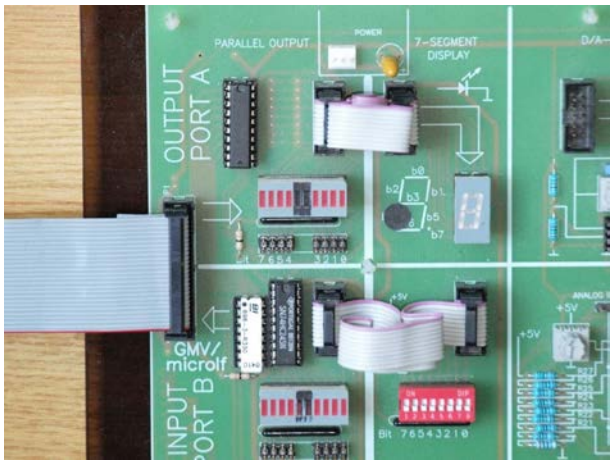
- Välj Debug | Terminal och sedan den COM-port som anvisats av laborationshandledare. Ett terminalfönster (blå färg) öppnas nu.
- Kontrollera att laborationsdatorn står i läge halt.
- Gör reset på laborationsdatorn.
- Placera markören i terminalfönstret och högerklicka, välj Load New och sedan fil för nedladdning.

Laborationsuppgift 4.1.

Sjusegmentsindikator

I denna uppgift ska du testa din lösning på uppgift 16.9 (`DisplaySegE`), i arbetsboken. Du ska tidigare ha provat den i simulator så du vet att ditt program beter sig som det ska.
Du måste dock göra ett litet tillägg till din tidigare lösning: för att programräknaren (PC) i laborationsdatorn ska få rätt värde (20_{16}) från början ska du lägga till assemblerdirektiv som placerar programmets startadress i RESET-vektorn.

- Laborationsdatorn ska vara ansluten till laborationskort ML4 via en 26-polig flatkabel.
- På ML4 ska sektionen DIPSWITCH input vara ansluten (10-polig flatkabel) till INPUT-sektionen.
- Sektionen 7-SEGMENT DISPLAY ansluts till OUTPORT-sektionen.



- Assemblera `DisplaySegE.sflisp`, det ska inte finnas några felmeddelanden.
- Kontrollera att laborationsdatorn står i läge halt.
- Placera markören i terminalfönstret och högerklicka, välj Load New och sedan filen `DisplaySegE.hwflisp` för nedladdning.
- Gör reset på laborationsdatorn. Kontrollera att programmets startadress nu finns i PC.
- Öppna listfilen (`DisplaySegE.lst`) i texteditorn.
- Sätt visningsadressen (Address) på laborationsdatorn i läge auto.

- Tryck en gång på step-omkopplaren och observera hur PC och Memory address uppdateras med programmets startadress.
- Läs av Memory data och se vad som finns i minnet på denna adress.
- Fortsätt med att utföra programmet instruktionsvis (step), följ med i listfilen, gör detta ett helt varv i programmet, dvs. tills PC på nytt får värdet 22_{16}.

- Starta exekvering av programmet genom att ställa omkopplaren i läge run.
- Öka exekveringshastigheten successivt genom att trycka på step-omkopplaren.

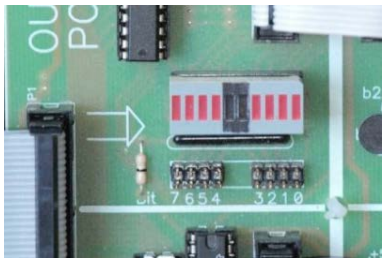
Laborationsuppgift 4.2.

Realtidsegenskaper, rinnande ljus.

Inför denna uppgift ska du ha utfört och testat uppgift 16.11 (`RunDiodeDelay`), i arbetsboken. Du ska alltså tidigare ha provat den i simulator så du vet att ditt program beter sig som det ska.

OBS: I simulatorn har du använt värdet 255 som "fördröjningskonstant" men det är allt för stort för laborationsdatorn. Använd i stället värdet 10.
Du måste även här lägga till assemblerdirektiv som placerar programmets startadress i RESET-vektorn för att laborationsdatorn ska starta korrekt.

- Laborationsdatorn ska vara ansluten till laborationskort ML4 via en 26-polig flatkabel på samma sätt som i föregående uppgift.
- I denna uppgift använde vi ljusdiodrampen på ML4's OUTPUT-sektion. Vi använder inte 7-SEGMENT-DISPLAY denna gång men du kan ändå låta den 10-poliga flatkabeln sitta kvar (den gör ingen skada).



- Assemblera `RunDiodeDelay.sflisp`, det ska inte finnas några felmeddelanden.
- Kontrollera att laborationsdatorn står i läge halt.
- Placera markören i terminalfönstret och högerklicka, välj Load New och sedan filen `RunDiodeDelay.hwflisp` för nedladdning.

- Gör reset på laborationsdatorn. Kontrollera att programmets startadress nu finns i PC.
- Starta programmet genom att ställa omkopplare i run-läge.
- Variera laborationsdatorns exekveringshastighet och observera skillnader hos ljusdiodrampen.

Laborationsuppgift 4.3.
Mekatronik, borra ett hål

Inför denna uppgift ska du ha utfört och testat uppgift 16.16 (`DrillHole`), i arbetsboken. Du ska ha provat programmet i simulator så du vet att det beter sig som det ska. Lägg också till assemblerdirektiv som placerar programmets startadress i RESET-vektorn så att laborationsdatorn startar korrekt.

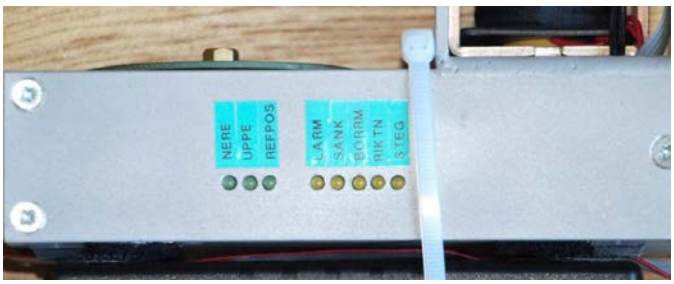
- Laborationsdatorn ska vara ansluten till laborationskort ML4 via en 26-polig flatkabel på samma sätt som i föregående uppgift.
    - Sektionen INPUT ska ansluten till sektionen DIPSWITCH. Omkopplaren används för att starta och stoppa borrmaskinprogrammet.
    - Sektion OUTPUT används inte i denna uppgift.

- Borrmaskinen ansluts till laborationsdatorn via en 26-polig kabel (ansluten till borrmaskin) som grenar sig i två stycken 10-poliga flatkablar:
    - Laborationssystemets port FC out ansluts till kabeln märkt FLISP OUT.
    - Laborationssystemets port FB in ansluts till kabeln märkt FLISP IN.

OBS: Växla INTE dessa anslutningar eftersom detta kan skada utgångarna hos laborationssystemet/borrmaskinen

- Utför programmet instruktionsvis (step), kontrollera att borrmaskinen utför de enskilda styrkommandona korrekt.

Nivåerna hos borrmaskinens styr- och status- signaler kan avläsas av dioderna på borrmaskinens ena sida:



- Låt laborationsdatorn utföra programmet (run) och ställ in den snabbaste exekveringshastigheten.

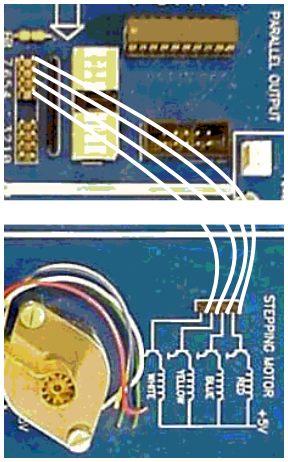
- Tänk på att simulatorn och laborationssystemet har olika tidsegenskaper, modifiera eventuellt Delay-rutinen så att rätt beteende uppnås.

Laborationsuppgift 4.4.
Variabel reglering för stegmotor

ML4 är utrustad med en stegmotor.

- Laborationsdatorn ska vara ansluten till laborationskort ML4 via en 26-polig flatkabel på samma sätt som i föregående uppgift.
    - Sektionen INPUT ska ansluten till sektionen DIPSWITCH. Omkopplaren används för att styra stegmotorns hastighet.
    - Sektion OUTPUT (bit7 - bit4) ansluts till ML4:s stegmotor.

Stegmotorn som är avsedd för unipolär drivning är ansluten via fyra stycken enpoliga kopplingskablar till PORT A's stiftlist SL1. Stegmotorns axel fås att rotera genom att de olika faserna (RED, BLUE, YELLOW och WHITE) styrs ut. Betrakta figuren i marginalen. Faserna ansluts via stiftlist SL4.



Observera att dessa faser är anslutna till +5V. För att få stegmotorn att rotera skall 0V kopplas till två av faserna medan de två andra faserna skall kopplas till +5V. Riktningen som stegmotorn roterar ges av följande tabell.

Stegmotorns rotationsriktning				
Fas	MEDURS → ← MOTURS			
	1	**2**	**3**	**4**
RÖD	+5V	+5V	GND	GND
BLÅ	GND	GND	+5V	+5V
GUL	GND	+5V	+5V	GND
VIT	+5V	GND	GND	+5V

Som tabellen visar har vi fyra olika tillstånd. Vi sätter först BLÅ och GUL fas till logiknivånivå 0 (GND) samtidigt som faserna RÖD och VIT ges logiknivå 1 (+5V). Detta motsvarar tabellens första kolumn.

- Om stegmotorn ska rotera medurs, ska sedan faserna ges de nivåer som anges i kolumn två, dvs. GUL och VIT ändras medan RÖD och BLÅ lämnas som de är. Efter kolumn två används i tur och ordning kolumn tre, kolumn fyra, kolumn ett, osv.
- Om stegmotorn ska roteras moturs, regleras faserna i stället genom att kolumnerna genomlöps: kolumn ett, kolumn fyra, kolumn tre, kolumn två, kolumn ett, osv.

Vid laborationsplatsen

Kontrollera att sektion OUTPUT är ansluten via stiftlisten SL1 direkt till stegmotorns stiftlist, SL4 enligt följande:

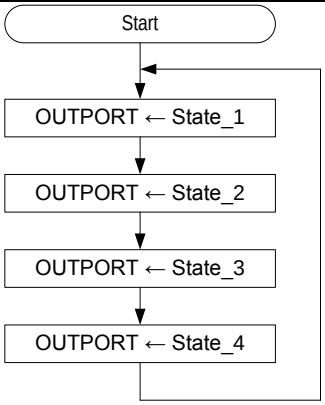
	b7	b6	b5	b4	b3	b2	b1	b0
OUTPORT	RÖD	BLÅ	GUL	VIT	x	x	x	x

Hemuppgift 4.1

Fyll i följande tabell som anger stegmotorns faser för att rotera medurs.

	b7	b6	b5	b4	b3	b2	b1	b0	HEX
State_1					0	0	0	0	
State_2					0	0	0	0	
State_3					0	0	0	0	
State_4					0	0	0	0	

Skriv en instruktionssekvens som får stegmotorn att rotera medurs, utforma instruktionssekvensen efter följande flödesdiagram:



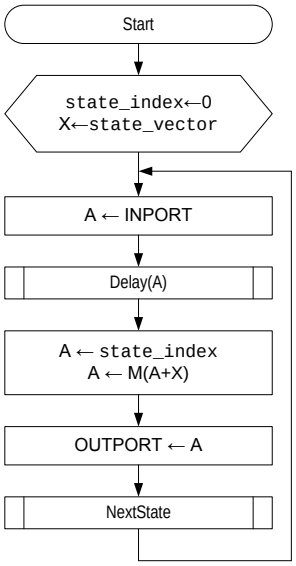
- Redigera en källtextfil `Lab_4-1.sflisp`, enligt flödesplanen, assemblera filen och rätta eventuella fel.

Vid laborationsplatsen

- Kontrollera instruktionssekvensens funktion genom att använda step-funktionen hos FLISP.
- Fungerar nu detta? Vrider stegmotorn sig ett steg i taget? *Om inte*, kontrollera att faserna ges rätt logiknivåer genom att observera vad lysdioderna för b_7, b_6, b_5 och b_4 på port OUTPUT visar.
- Rätta eventuella fel, stegmotorn skall vrida sig ett steg för varje nytt värde som matas ut.
- Starta därefter exekvering av instruktionssekvensen genom att ställa omkopplare i run-läge.
- Variera laborationsdatorns exekveringshastighet och observera eventuella skillnader i stegmotorns beteende.

Hemuppgift 4.2

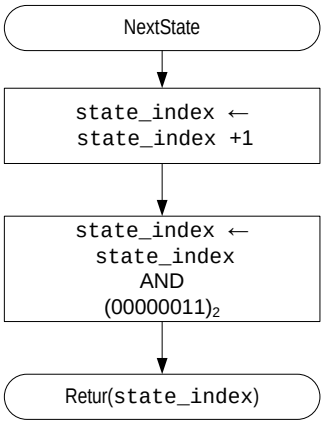
Konstruera ett program för variabel reglering av stegmotorns rotationshastighet enligt följande flödesdiagram:



- Redigera en källtextfil `Lab_4-2.sflisp`, enligt flödesplanen, lägg till subrutinerna Delay och NextState, assemblera filen och rätta eventuella fel.

Utforma subrutinen, Delay, så att fördröjningen anges i register A vid anrop, jämför med uppgift 16.11 i arbetsboken.

Konstruera en subrutin `NextState` som en "modulo-4 räknare", dvs. bestämmer "kolumn" för stegmotorns nästa tillstånd då den ska vridas ett steg medurs. Tillståndet ska returneras i A. Subrutinen måste ges någon form av "minne", dvs. det värde som rutinen ska returnera beror av det föregående värdet. Ett sätt att göra detta är att använda en global variabel, vi kallar den `state_index`, och denna variabel kan enbart anta värdena 0,1,2,3.



Vi definierar också subrutin och data i pseudospråk:

```
initialt värde indexvariabel
state_index = 0;
konstant vektor
state_vector={State_1, State _2, State _3, State _4};

NextState:
    state_index = state_index+1;
    state_index = state_index & 3;
    return (state_index);
```

Vid laborationsplatsen

- Kontrollera programmets funktion i run-läge.
- Variera stegmotorns hastighet genom att ställa in olika värden på DIPSWITCHEN.
- Då programmet fungerar som det ska tillkallar du handledare för att redovisa lösningen på laborationsuppgiften.

Tillägg till PM laboration 4:

Du kan ge kommandon till FLIS-processorn genom att klicka på terminalfönstret i ETERM och skriva in något av följande:

Kommando	Betydelse
i	Interaktiv mode, alla tecken skickas tillbaks och syns i terminalfönstret
q	Tyst mode, inga tecken skickas tillbaks
v	Om mode=i, skriv versionsnummer till terminal
t	Testsekvens, testar FLIS-processorns indikatorer genom att tända dessa succesivt, avsluta genom att ge kommandot 't' igen.
a	reset, utför åtesrtällning av FLIS-processorn
s	Utför hel instruktion (till nästa NF)
e	Utför program utan uppehåll, exekvera, avbryt exekvering genom att ge ytterligare ett ’e’-kommando.
wrZXX	Skriv värdet XX till register Z. Värdet XX anges på hexadecimal form med precis två siffror. Registret, Z, kan vara något av datavägens register enligt: a,x,y,s=sp,p=pc, r,c=cc.
wmXXYY	Skriv värdet XX till minnesadress YY. Såväl värdet XX som adressen YY anges på hexadecimal form med precis två siffror.
rrZ	Om mode=i, Skicka värdet (hexadecimal form) i register Z till terminalen. Registret, Z, kan vara något av datavägens register enligt: a,x,y,s=sp,p=pc, c=cc.
rmXX	Om mode=i, Skicka värdet (hexadecimal form) på minnesadress XX till terminalen

FLIS-datorn är i ”interaktiv mode” efter RESET.