CHALMERS
Institutionen för data och informationsteknik.
LP1 - 2015

Grundläggande datorteknik

Laborationer

Denna laborationsserie för Grundläggande datorteknik omfattar totalt fyra laborationsmoment som utförs i tur och ordning. Tiden vid laborationsplatsen förkortas avsevärt genom noggranna laborationsförberedelser. Inför varje laborationstillfälle ska du därför förbereda dig genom att noggrant läsa igenom anvisningarna för laborationsmomentet i detta PM. För de flesta uppgifterna förutsätts det även att du utfört laborationsförberedande hemuppgifter. Vilka dessa uppgifter är framgår av detta PM. Du ska vara beredd att visa upp och redogöra för dina förberedda lösningar inför laborationstillfället. ***Bristfälliga förberedelser kan medföra avvisning från bokad laborationstid.***

Underskrifterna på detta försättsblad är ditt kvitto på godkänt resultat på respektive laborationsmoment. Spara det, för säkerhets skull, tills slutbetyg på kursen rapporterats.

Börja med att skriva ditt namn och personnummer med bläck.

_____ _____
Personnummer Namn (textat)

Följande tabell fylls i av laborationshandledare efter godkänd laboration.

Laboration	Godkännande av laboration		Labbprov	
	Datum	Laborationshandledares underskrift	Datum	Signatur
1				
2				
3				
4				

Godkännande – hel laborationsserie:

_____ _____
Datum Laborationshandledares underskrift

Översikt av laborationsserien

Under laboration 1 bekantar du dig med laborationssystemet och konstruerar enkla kombinatoriska nät som exemplifierar användning av logikkretsar. Speciellt illustreras nät såsom kodomvandlare, väljare och adderare, vilka återkommer under senare labbar då du bygger upp en dators centralenhet (FLISP).

Under laboration 2 studerar du dataöverföring mellan register i datavägen och hur ALU:n används för att utföra operationer på data i register.

Under laboration 3 får du prova på att själv konstruera och testa instruktioner (i form av styrsignalsekvenser) för FLIS-processorn.

Laboration 4 omfattar grundläggande assemblerprogrammering och här får du också tillfälle att praktisera test, felsökning och ”avlusning” (eng. *debugging*).

Kompletterande material

För laborationernas genomförande behöver du, utöver kurslitteraturen, program och diverse tekniska beskrivningar.

Vid laborationerna används följande program:

DigiFlisp
ETERM 6 för Flisp

Du finner länkar till programmen via kurshemsidan under *Dokument/Simulatorer*.

Laboration nr 1

Logikgrindar och logiknivåer
Kombinatoriska nät: kodomvandlare, väljare, adderare

Följande uppgifter ur *Arbetsbok för DigiFlisp* ska vara utförda som förberedelse för laborationen. Du ska på begäran av laborationshandledare redogöra för dessa. Tillfälle för redovisning kommer att anordnas på simulationstillfällen.

Uppgifter inom parentes är inte strikt obligatoriska, men om man gjort dem blir arbetet med de obligatoriska uppgifterna lättare.

Uppg.	(3.1) 3.2 3.6 3.7 (3.8) 3.9 3.11 3.12 3.13	4.1 4.2 4.3	5.1 5.2 5.3 5.4 5.5	6.1 6.2 6.4 6.5 6.6
Sign.				

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

Hem-Uppgift	1.A	1.B
Sign.		

Följande laborationsuppgifter skall redovisas för en handledare för godkännande under laborationen, dvs. innan du kopplar ned.

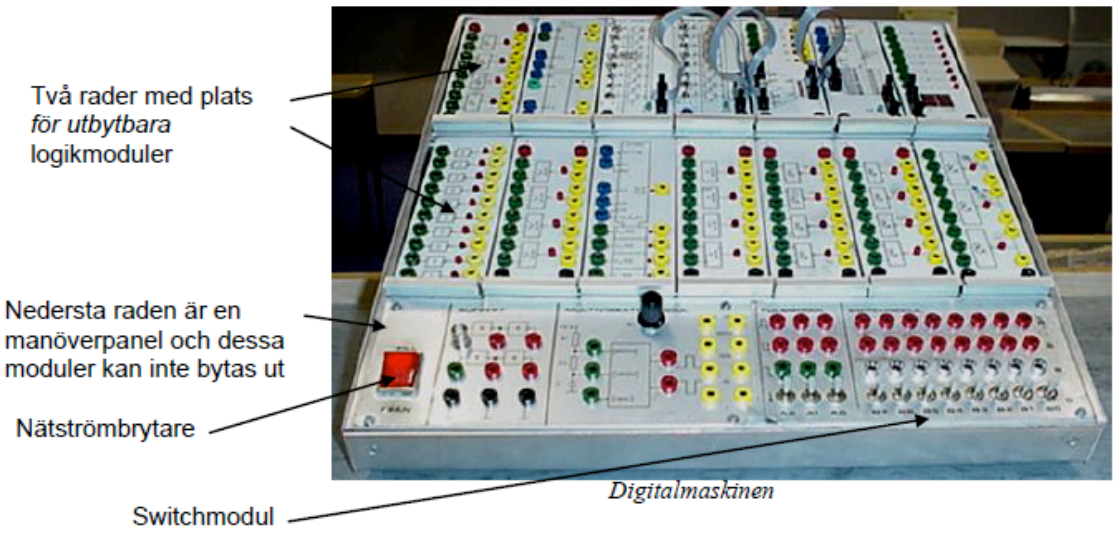
Laborations-uppgift	1.1	1.2	1.3	1.4	1.5	1.6	1.X
Sign.							

Laborationssystemet

Under denna laboration använder du:

- Digitalmaskinen med dess grundläggande moduler.
- Enkelt universalinstrument (”voltmeter”).

Digitalmaskinen består av tre rader moduler. De två översta raderna innehåller utbytbara logikmoduler och den nedersta raden består av en fast manöverpanel.



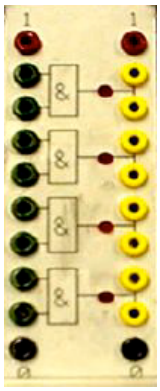
Digitalmaskinen

Läsanvisning:

Läs om digitalmaskinen i filen *Moduler till digitalmaskinen.pdf* som finns under *Dokument/Handböcker och datablad* på kurshemsidan.

Laborationsuppgift 1.1:

Mätning av logiknivåer



Som första praktiska uppgift i laboratoriet skall vi undersöka vilka spänningar relativt jord, som motsvarar logikvärdena "0" och "1" i digitalmaskinen. Eftersom alla spänningar i digitalmaskinen mäts relativt jord använder vi i fortsättningen endast ordet "spänning" i stället för "spänning relativt jord". Vi mäter spänningarna på en AND-grinds ingångar och utgång med hjälp av en voltmeter.

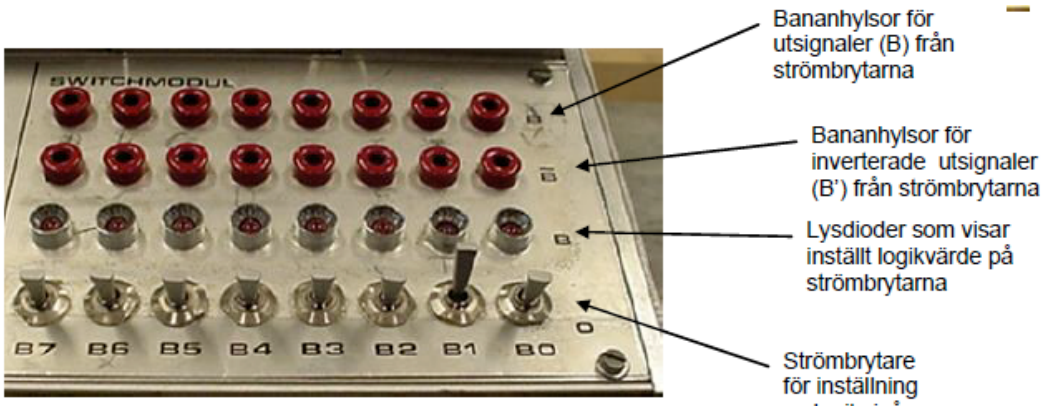
Fråga en handledare om du behöver hjälp med att använda en voltmeter.

Studera AND-modulen som innehåller fyra 2-ingångars AND-grindar.

Grindarna har gröna bananhylsor på ingångarna. De gula utgångarna är dubblerade för att man enklare skall kunna koppla utsignalen vidare till ingångar på andra grindar. Varje grind har en lysdiod på utgången som indikerar utgångens logiknivå (tänd diod=1, släckt diod=0). Överst finns två röda bananhylsor med +5V och nederst två svarta med 0V. Dessa kan man använda till att koppla logiknivå ett eller logiknivå noll till ingångar på kretsarna.

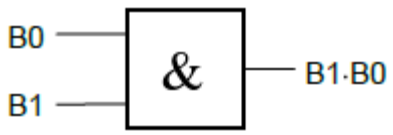
Slå av nätströmbrytaren på digitalmaskinen. Nätströmbrytaren på digitalmaskinen skall vara avstängd vid allt kopplingsarbete!

Insignaler till grindarna kan tas från switchmodulen: Anslut en kopplingskabel mellan den övre banankontakten på switch B0 (B-raden på switchmodulen och en av ingångarna på en av AND-grindarna.



Switchmodul

Anslut en annan kopplingskabel mellan den övre banankontakten på switch B1 (B-raden på switchmodulen) och den andra grindingången. Se figuren till höger.



Slå på nätströmbrytaren på digitalmaskinen.

Ställ in logikvärdena enligt tabellen nedan med switcharna B0 och B1. Mät spänningen på den använda grindens ingångar och utgång med en voltmeter och fyll i tabellen. Observera också utgångens logikvärde.

Insignal B1		Insignal B0		Utsignal B1·B0	
Logikvärde	Spänning [V]	Logikvärde	Spänning [V]	Logikvärde	Spänning [V]
0		0			
0		1			
1		0			
1		1			

Från mätvärdena i tabellen drar vi slutsatsen att:

Logikvärdet "0" motsvarar spänningen ____ V.

Logikvärdet "1" motsvarar spänningen ____ V.

De uppmätta värdena är typiska för den sorts mikroelektronik, HCMOS, som används i digitalmaskinen.

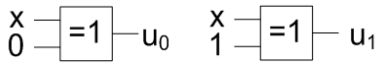
Vi observerar också att de AND-grindar som inte har någon yttre signal ansluten till sina ingångar via en labsladd har logikvärdet ____ på utgången.

Mätning av spänning på ingångarna till dessa grindar visar spänningen ____ V, dvs. logikvärdet ___, vilket förklarar utsignalvärdet. Det är samma logikvärde på (nästan) samtliga ingångar, som saknar yttre anslutning via en labsladd.

Laborationsuppgift 1.2:

Undersökning av XOR-grind

Slå av nätströmbrytaren på digitalmaskinen.



s	x	u_0	u_1
0	0		-
0	1		-
1	0	-	
1	1	-	

Koppla upp uppgift 3.11 från arbetsboken.

Slå på nätströmbrytaren på digitalmaskinen.

Använd switch B0 på switchmodulen som styrsignal **s** för att lägga på nollan eller ettan, och switch B1 som insignal **x**.

Fyll i funktionstabellen.

Laborationsuppgift 1.3:

NAND-logik

Slå av nätströmbrytaren på digitalmaskinen.

Koppla upp uppgift 3.9 där du använder NAND/NAND logik från arbetsboken.

Använd switch B3-B0 på switchmodulen som insignaler för **x**, **y**, **z** och **w**.

Slå på nätströmbrytaren på digitalmaskinen och kontrollera utsignalen **f** för olika värden hos insignalerna, fyll i funktionstabellen:

x	y	z	w	f
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Hemuppgift 1.A:

I laborationsuppgift 1.4 nedan, ska du koppla upp uppgift 4.3 från arbetsboken.

Här måste du dock tänka på att du inte har obegränsat med moduler för din koppling. Du måste därför eventuellt anpassa din lösning för användning av de moduler som finns tillgängliga på laborationsplatsen:

- 10 st. INVERTERARE
- 4 st. 2-ingångars AND
- 8 st. 2-ingångars NAND
- 3 st. 3-ingångars NAND
- 4 st. 2-ingångars NOR
- 8 st. XOR

Kontrollera att din lösning i arbetsboken inte omfattar fler grindar än du har tillgång till på laborationsplatsen. Modifiera eventuellt lösningen så att grindarna räcker och skriv här booleska uttryck för funktionerna.

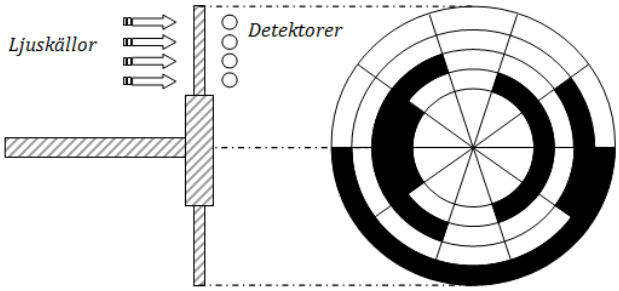
a=	b=
c=	d=

Laborationsuppgift 1.4:

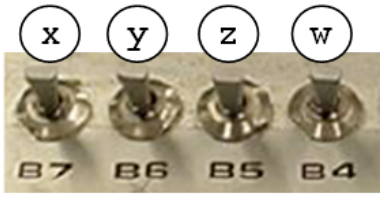
Konstruktion av digital vinkelgivare

Slå av nätströmbrytaren på digitalmaskinen.

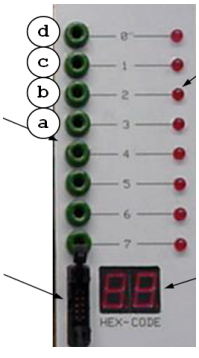
Koppla upp uppgift 4.3 från arbetsboken enligt din lösning i hemuppgift 1.A.



Använd switcharna B7 till B4 på switchmodulen för insignalerna **x**, **y**, **z** och **w** enligt vidstående figur, tänk på att även insignalernas inverser finns tillgängliga från switchmodulen.



Koppla vinkelgivarens utsignaler till displaymodulen så att det högra sifferfönstret används, dvs. utsignal **a** till insignal nr 3, utsignal **b** till insignal nr 2, utsignal **c** till insignal 1 och utsignal **d** till insignal 0 på displaymodulen.

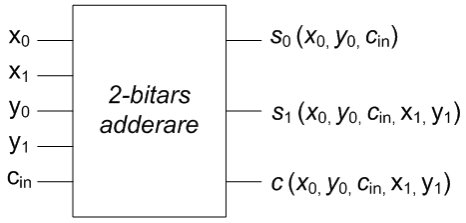


När du kopplat upp hela kodomvandlaren slår du på nätströmbrytaren på digitalmaskinen och testar vinkelgivaren för hela funktionstabellen i figur 4.3 i arbetsboken. Demonstrera sedan kopplingen för en handledare.

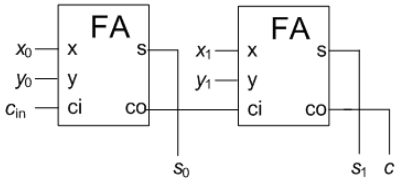
Hemuppgift 1.B:

Studera uppgift 5.1 i arbetsboken.

Rita schemat för en 2-bitars adderare i boxen nedan så att du har den lätt tillgänglig under kopplingsarbetet vid laborationen.



Schema för 2-bitars adderare.



Laborationsuppgift 1.5:

Konstruktion av 2-bitars adderare

Under denna laborationsuppgift ska du koppla upp den 2-bitars adderare du konstruerat som hemuppgift 1.B.

Slå av nätströmbrytaren på digitalmaskinen.

Koppla upp adderaren där du ansluter insignalerna $\mathbf{x_1}$ och $\mathbf{x_0}$ till B5 och B4 på switch-modulen. Vidare väljer du B1 och B0 för insignalerna $\mathbf{y_1}$ och $\mathbf{y_0}$. Slutligen väljer du B7 som $\mathbf{c_{in}}$.

När du kopplar upp ett lite större nät kan en kopplingssladd lätt hamna fel. Därför är det ett bra arbetssätt att först koppla upp en delmängd av konstruktionen.

Exempelvis kan du börja med att koppla nätet som utgör en 1-bits adderare, dvs. bilda $\mathbf{s_0}$ och carry ut från additionen av $\mathbf{x_0}$ och $\mathbf{y_0}$, och testar detta först.

Därefter kan du utöka kopplingen till en 2-bits adderare.

Efter att ha kontrollerat koppligen demonstrerar du den för en handledare.

Laborationsuppgift 1.6:

Väljare och dess användning

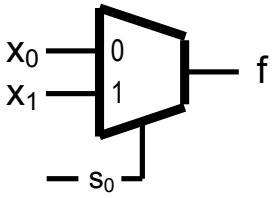
Slå av nätströmbrytaren på digitalmaskinen.

Koppla upp uppgift 6.1 från arbetsboken.

Använd switcharna B7, B6 och B0 på switchmodulen för signalerna x_0, x_1 och s_0 enligt funktionstabellen.

Slå på digitalmaskinens nätströmbrytare och komplettera tabellen med funktionsvärdena **f**.

Styrsignal s_0 (B0)	Insignaler x_1 (B6)	x_0 (B7)	Utsignal f
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



Laboration nr 2

ALU
Register med 3-state-utgång
Dataväg och minne
Räknare

Följande uppgifter ur *Arbetsbok för DigiFlisp* ska vara utförda som förberedelse för laborationen. Du ska på begäran av laborationshandledare redogöra för dessa.

Uppgifter inom parentes är inte strikt obligatoriska, men om man gjort dem blir arbetet med de obligatoriska uppgifterna lättare.

Uppg.	7.1 7.2	8.1 8.2 8.3 8.4 8.5 8.6	9.1	11.1 11.9 11.11	(12.1) 12.2 12.3	(13.1) 13.2
Sign.						

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

Hem-Uppgift	2.A	2.B
Sign.		

Följande laborationsuppgifter skall redovisas för en handledare för godkännande under laborationen, dvs. innan du kopplar ned.

Laborations-uppgift	2.1	2.2	2.3	2.4	2.5
Sign.					

Hemuppgift 2.A:

- Utför deluppgifterna 2.3.2, 2.3.3 och fyll i tabellen under 2.3.4.

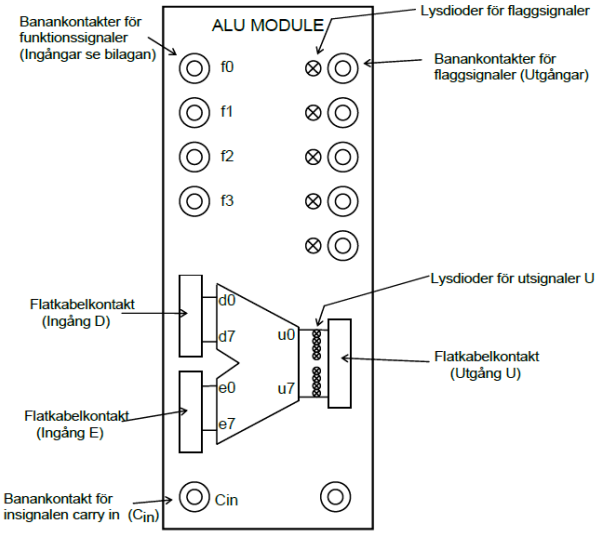
I några av laborationsuppgifterna används en ALU. Eftersom laborationssystemets ALU skiljer sig något från den som beskrivits i arbetsboken och FLISP-simulatorn (se laborationsuppgift 2.1) ska du förbereda laborationsuppgifterna i detta avsnitt med att

- Ersätta funktionskoderna i dina lösningar som du gjort med FLISP-simulatorn med de funktionskoder som gäller för laborationssystemets ALU. Dessa använder du sedan under laborationen.

Hemuppgift 2.B:

- Utför deluppgifterna 2.5.1 och 2.5.2.

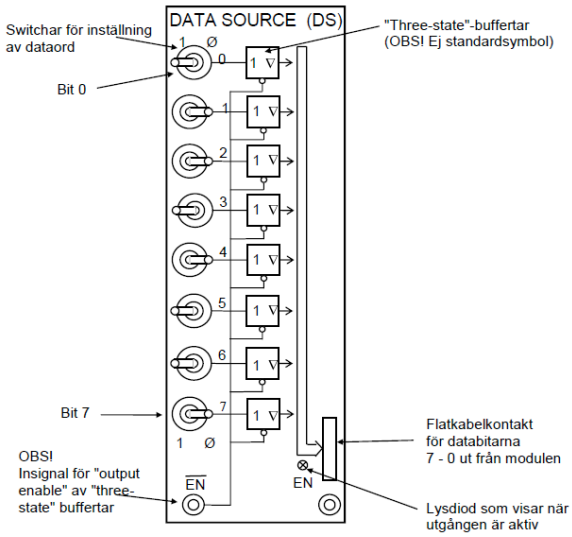
Beskrivning av ALU-, DATA SOURCE- och DISPLAY- moduler



Figuren till vänster visar frontpanelen för en 8-bitars ALU-modul som ingår i labbsystemet. Modulen har två 8-bitars dataingångar D och E samt en ingång för carry-in. Den har också fyra ingångar, f_0, f_1, f_2 och f_3 som bestämmer dess funktion.

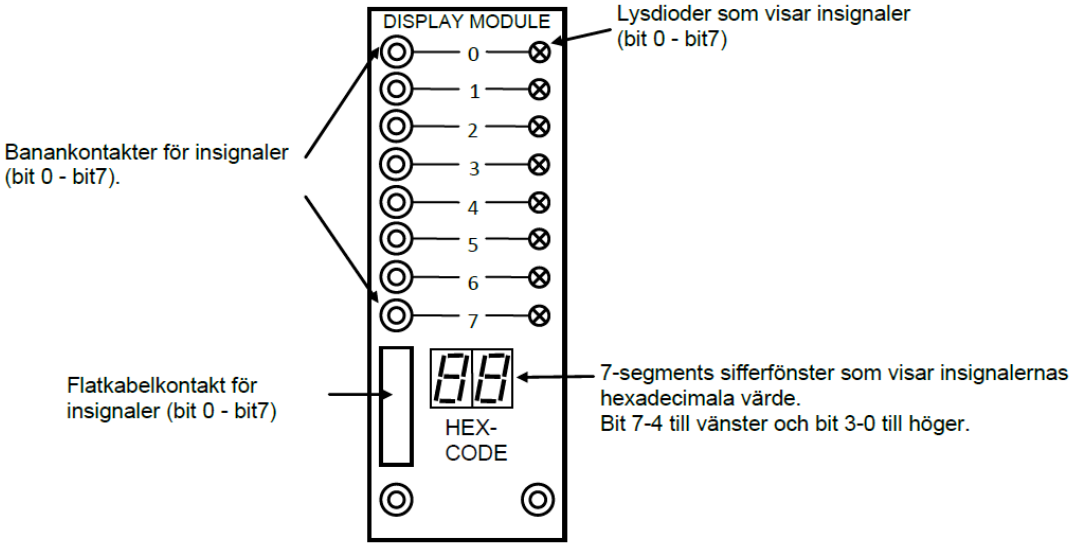
Förutom datautgången U med 8 bitar har den flaggutgångarna N, Z, V och C.

De båda dataingångarna D och E, samt datautgången U finns tillgängliga på frontpanelen i form av flatkabelkontakter. Insignalerna till D- och E-ingångarna och utsignalerna på U-utgången måste därför kopplas med flatkablar via dessa kontakter.



I labbsystemet ingår också två moduler med namnet "DATA SOURCE" (DS). Med hjälp av dem kan man koppla in 8-bitars dataord till olika enheter via flatkablar. Dataorden kan ställas in med åtta switchar, som finns på DS-modulens framsida. Varje DS-modul har "three-state"-buffertar på sina åtta datautgångar mot flatkabelkontakten och kan därför användas som en av flera datakällor på samma buss. "Three-state"-bufferterna i en DS-modul kan aktiveras med signalen EN' via en banankontakt på frontpanelen. Figuren till höger visar "DATA SOURCE"-modulens frontpanel med dataordet 01101001 inställt på switcharna (bit 7 - bit 0). Genom att aktivera ingången EN' nere till vänster på modulen lägger man ut bitmönstret på flatkabelkontakten nere till höger.

För att man enkelt skall kunna se värdet på 8-bitars tal från till exempel ALU:n har labbsystemet försetts med en displaymodul med 2 st 7-segments sifferfönster. I sifferfönstren visas det binära 8-bitarsvärdet på flatkabelkontakten som ett hexadecimalt tal. Se figur nedan.



Laborationsuppgift 2.1:

Analys av 8-bitars ALU

OBSERVERA: I laborationssystemet skiljer sig ALU:n åt något från den ALU som beskrivs i arbetsboken. Följande funktionstabell gäller laborationssystemets ALU:

funktion				operation	utsignaler											
f_3	f_2	f_1	f_0	RTN	u_7	u_6	u_5	u_4	u_3	u_2	u_1	u_0	N	Z	V	C
0	0	0	0	$0 \rightarrow U$	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	1	$D \rightarrow U$	d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0	u_7	(1)	0	0
0	0	1	0	$E \rightarrow U$	e_7	e_6	e_5	e_4	e_3	e_2	e_1	e_0	u_7	(1)	0	0
0	0	1	1	$\overline{D} \rightarrow U$	$\overline{d_7}$	$\overline{d_6}$	$\overline{d_5}$	$\overline{d_4}$	$\overline{d_3}$	$\overline{d_2}$	$\overline{d_1}$	$\overline{d_0}$	u_7	(1)	0	0
0	1	0	0	$\overline{E} \rightarrow U$	$\overline{e_7}$	$\overline{e_6}$	$\overline{e_5}$	$\overline{e_4}$	$\overline{e_3}$	$\overline{e_2}$	$\overline{e_1}$	$\overline{e_0}$	u_7	(1)	0	0
0	1	0	1	$D \vee E \rightarrow U$	$d_7 \vee e_7$	$d_6 \vee e_6$	$d_5 \vee e_5$	$d_4 \vee e_4$	$d_3 \vee e_3$	$d_2 \vee e_2$	$d_1 \vee e_1$	$d_0 \vee e_0$	u_7	(1)	0	0
0	1	1	0	$D \wedge E \rightarrow U$	$d_7 \wedge e_7$	$d_6 \wedge e_6$	$d_5 \wedge e_5$	$d_4 \wedge e_4$	$d_3 \wedge e_3$	$d_2 \wedge e_2$	$d_1 \wedge e_1$	$d_0 \wedge e_0$	u_7	(1)	0	0
0	1	1	1	$D \oplus E \rightarrow U$	$d_7 \oplus e_7$	$d_6 \oplus e_6$	$d_5 \oplus e_5$	$d_4 \oplus e_4$	$d_3 \oplus e_3$	$d_2 \oplus e_2$	$d_1 \oplus e_1$	$d_0 \oplus e_0$	u_7	(1)	0	0
1	0	0	0	$D + C_{in} \rightarrow U$									u_7	(1)	(2)	(3)
1	0	0	1	$D + (FF)_{16} + C_{in} \rightarrow U$									u_7	(1)	(2)	(3)
1	0	1	0	$D + E + C_{in} \rightarrow U$									u_7	(1)	(2)	(3)
1	0	1	1	$D + D + C_{in} \rightarrow U$									u_7	(1)	0	(4)
1	1	0	0	$D + \overline{E} + C_{in} \rightarrow U$									u_7	(1)	(2)	(3)
1	1	0	1	$0 \rightarrow U$									0	1	0	0
1	1	1	0	$0 \rightarrow U$									0	1	0	0
1	1	1	1	$(FF)_{16} \rightarrow U$	1	1	1	1	1	1	1	1	1	0	0	0

(1) $Z = \overline{u_7} \wedge \overline{u_6} \wedge \overline{u_5} \wedge \overline{u_4} \wedge \overline{u_3} \wedge \overline{u_2} \wedge \overline{u_1} \wedge \overline{u_0}$, dvs. Z=1 då samtliga bitar i register U är 0, Z=0 annars.

(2) $V = (\overline{u_7} \wedge d_7 \wedge e_7) \vee (u_7 \wedge \overline{d_7} \wedge \overline{e_7})$, dvs. V-flaggan sätts enligt reglerna för tvåkomplementsaritmetik.

(3) $C = c_8$, dvs. carry ut från additionen av de mest signifikanta siffrorna.

(4) C = utskiftad bit, dvs. bit d_7 före vänsterskiftet.

För ALU:ns samtliga operationer gäller att innehållen på ingångarna D och E inte kan påverkas. Varje operation, såsom bestämd av F, kan endast påverka utgången U och flaggorna som vi betecknar ALU(N, V, Z, C).

Jämför nu med arbetsbokens uppgift 7.1 och betrakta följande tabell. Identifiera motsvarande funktioner (funktionskoder) hos laborationssystemets ALU och fyll i dessa i tabellen. Använd sedan ALU:n för att fylla i resultaten i tabellen (kolumnen *Utgång U*).

	Operation	F					Ingång D	Ingång E	Utgång U
		f_3	f_2	f_1	f_0	C_{in}	Bin	Bin	Bin
1)	E → U						. . . . .	. . . . .	
2)	D∨E→U						. . . . .	. . . . .	
3)	D⊕E→U						1 1 0 1 0 0 1 1	0 1 0 1 1 0 0 0	
4)	D<<1(C_{in})→ U					1	. . . . .	. . . . .	
5)	D<<1(C_{in})→ U					0	. . . . .	. . . . .	

Slå av nätströmbrytaren på digitalmaskinen.

Koppla flatkablar mellan DS-modulerna och D- resp E-ingången på ALU-modulen. Glöm inte att DS-modulerna ska ha aktiva utgångar!

Koppla också en flatkabel mellan ALU-modulens utgång U och displaymodulen.

Anslut ALU:ns funktionsingångar f_3 -f_0 och C_{in} till switchmodulen nere till höger på digitalmaskinen enligt följande tabell:

B7	B6	B5	B4	B3	B2	B1	B0
f_3	f_2	f_1	f_0				C_{in}

Slå på nätströmbrytaren på digitalmaskinen.

Ställ in funktionskoderna och kontrollera operationerna enligt tabellen ovan. Jämför resultaten (U) med de resultat du fick för motsvarande operationer i uppgift 7.1.

Utgå nu från arbetsbokens uppgift 7.2. Identifiera laborationssystemets funktionskoder för operationerna addition och subtraktion, fyll i dessa i följande tabell.

Koppla upp följande operationer, komplettera tabellen, jämför slutligen med resultaten för motsvarande operationer i arbetsbokens uppgift 7.2

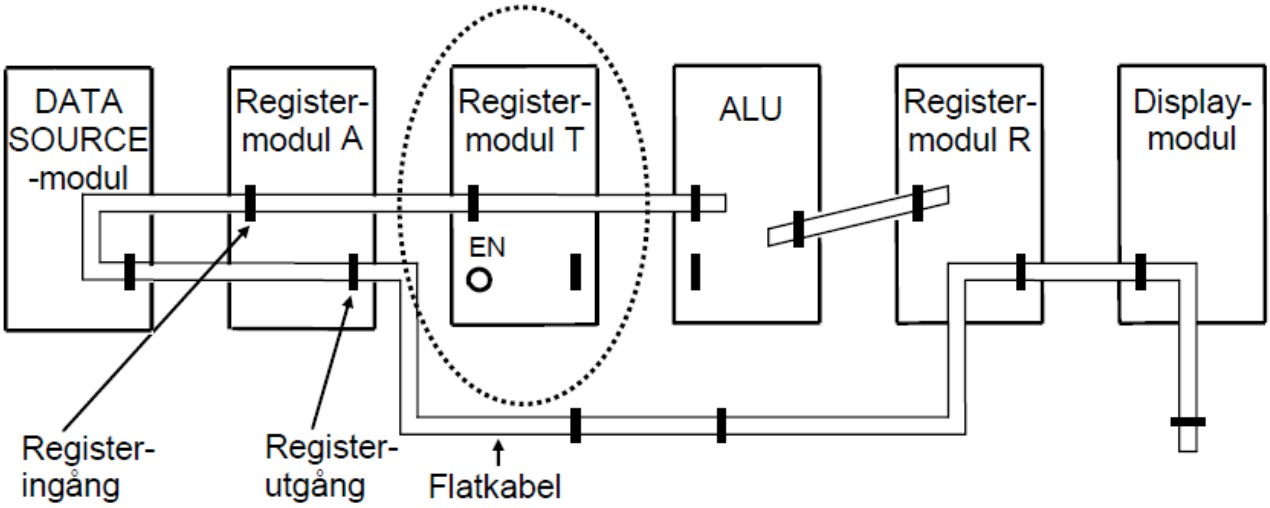
	Ingång D		Ingång E		Op	ALU-funktion		Utgång U		Flaggor			
	Dec	Bin	Dec	Bin		F	C_{in}	Bin	Dec	N	Z	V	C
1)	27	0 0 0 1 1 0 1 1	55	0 0 1 1 0 1 1 1	+			0 1 0 1 0 0 1 0	82				
2)	55		–27		+				28				
3)	–55		–27		+				-82				
4)	55		27		-				28				
5)	27	0 0 0 1 1 0 1 1	55	0 0 1 1 0 1 1 1	-				-28				
6)	27	0 0 0 1 1 0 1 1	–55		-				82				

Laborationsuppgift 2.2:

Register med 3-state utgång

Studera följande figur över en dataväg och se till att en handledare kopplar upp den i labbsystemet. Kopplingen skall senare användas för att visa hur man kan flytta och behandla data i en enkel dataväg. Först skall vi dock studera hur en registermodul (REG8) fungerar.

När du skall avlägsna en kabel, får du aldrig dra i själva kabeln, utan använd istället utkastar-armarna som finns på stifttagen på modulerna!



- Slå på nätströmbrytaren på digitalmaskinen.
- Register T:s utgångar är inte anslutna till någonting. Studera modulen. Lägg märke till att lysdioderna på registermodulens utgångssida (till höger) visar ett odefinierat värde. Detta beror på att registermodulens utgångar inte är aktiverade och därför befinner sig i flytande tillstånd. De får alltså inget logikvärde från registrets Q-utgångar vars värden man ser på lysdioderna i mittraden.
- Vidrör utgångskontaktens metallstift lätt med fingerspetsen och iakttag hur lysdioderna på utgångssidan ändrar ljusstyrka. Fenomenet förklaras av att laddningar omfördelas mellan fingret och de olika kontaktstiften. Detta visar också att "three-state"-utgångar som befinner sig i flytande tillstånd har mycket hög impedans (resistans) till matningsspänningen och jord.
- Om nödvändigt, lyft bort en modul på digitalmodulens mittersta rad. I ”hålet” där modulen saknas hittar du två spännings-skenor. Den ena är +5V och den andra är jord (0V). Vidrör utgångskontaktens metallstift på registermodulen lätt med fingerspetsen samtidigt som du håller ett finger på en av spännings-skenorna. Testa detta ett antal gånger när du vidrör spänningsskenorna.
- Anslut T-registermodulens klockingång och Load Enable-ingång till switch-modulen (exempelvis till switcharna B0 och B1) på digitalmaskinen.
- Anslut DS-modulens Output Enable-signal till switch-modulen på digitalmaskinen.
- Ställ in värdet 37_{16} på DS-modulen.

Undersök hur de olika styrsignalerna ska vara inställda för att värdet 37_{16} ska kopieras från DS-modulen till T-registret. Skriv ner dina slutsatser.

_____

_____

Inaktivera Output Enable-signalen för DS-modulen och aktivera Output Enable-signalen för T-registermodulen. Vidrör utgångskontaktens metallstift på registermodulen lätt med fingerspetsen. Observera vad som händer på registrets utgång. Upprepa förfarandet när Output Enable för T-registermodulen inte är aktiverad. Varför ändras värdet på utgången ibland och ibland inte?

_____

_____

Gör en sammanställning i tabellen nedan över registermodulens styrsignaler Load Enable, Output Enable, klockpuls etc. Ange alla ingångar M1, M2, M3, C4 och EN. Ange om signalerna är aktiva som nollor eller ettor. Är du osäker testar du genom att klocka in nya värden i registret och att belasta utgången (beröra utgångskontaktens metallstift) på registermodulen.

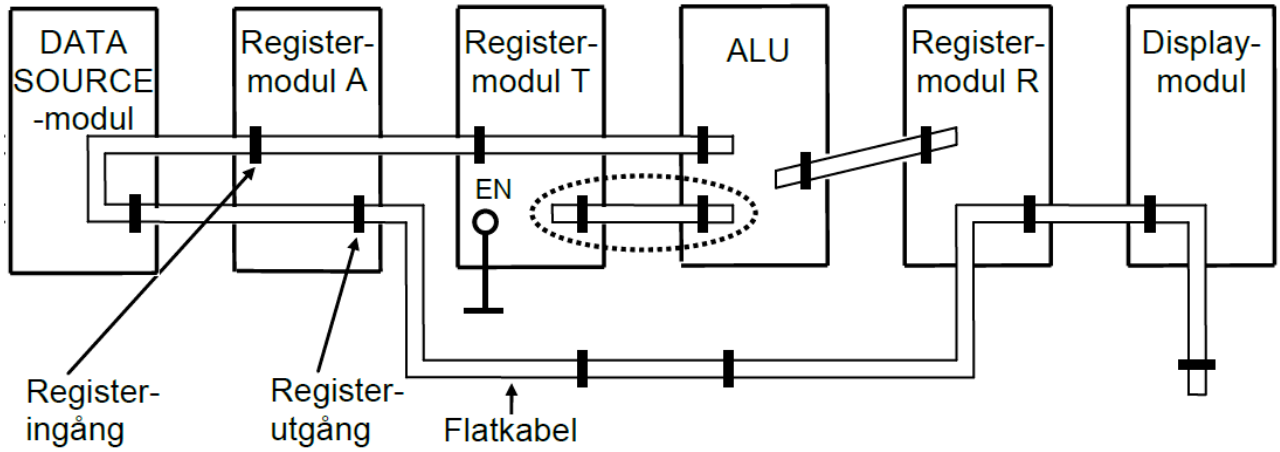
Ingång	Beteckning	Aktiv	Kommentar
M1			
M2			
M3	Load Enable	Låg (0)	
C4			
EN			

Slå av nätströmbrytaren, låt flatkablarna sitta kvar och avlägsna övriga kablar.

Laborationsuppgift 2.3:

Dataväg med ALU

Studera figur 11.1 på sidan 57 i arbetsboken och jämför med uppkopplingen för denna laborationsuppgift. Notera att en flatkabel nu ska anslutas mellan T-registret och ALU:n i labbsystemet och att Output Enable för T-registret ska anslutas till jord.



En flatkabel med flertalet kontakter kopplar samman de tre registermodulerna, Datasource och Displaymodulen med ALU:n. Flatkabeln fungerar nu som en databuss.

OBSERVERA: Notera att hylsdonen (hon-kontakterna) på flatkabeln endast kan kopplas i stifttagen på ett sätt på grund av styrstiftet.

Utöver de nämnda modulerna använder du här också en ”MANUELL STYRENHET” med vars hjälp den lilla datavägen kan styras.

Om vi jämför laborationssystemets moduler med arbetsbokens ”Dataväg med ALU” noterar vi bland annat följande skillnader:

- Hos laborationssystemet återfinns några styrsignaler som du inte hittar i arbetsbokens styrenhet (LDB och OEB), dessa kommer vi *inte* att använda. Dessutom finns styrsignalerna MEM_W, MEM_R som vi kommer att använda först under nästa laborationsuppgift.
- Arbetsbokens styrsignal OE_S, betecknas hos laborationssystemet OEDS.
- Arbetsbokens manuella styrenhet har 6 st uppsättningar omkopplare för styrsignalerna. Laborationssystemets manuella styrenhet har bara en uppsättning omkopplare. Detta innebär att du måste ställa om styrsignalerna inför varje klockpuls då du manövrerar laborationssystemet.
- I laborationssystemet skiljer sig ALU:n åt något från den ALU som beskrivs i arbetsboken. För laborationssystemets ALU använder du de funktionskoder som gavs i laborationsuppgift 2.1.

Deluppgift 2.3.1:

Koppla samman den manuella styrenhetens signaler: OEDS, OEA,OER,LDA, LDT,LDR Cin, f3,f2,f1,f0 och CLOCK till rätt kopplingspunkter i datavägen. Anslut Output Enable-signalen på T-registermodulen till jord.

Deluppgift 2.3.2:

Ange en styrsignalsekvens som placerar värdet 12_{16} i register A och värdet 62_{16} i register R.

RTN	steg	*Source*	OE_S	OE_A	OE_R	LD_A	LD_T	LD_R	C_{in}	f_3	f_2	f_1	f_0
$12_{16} \rightarrow A$	*1*												
$62_{16} \rightarrow R$	*2*												

Utför styrsignalsekvensen i laborationssystemet och kontrollera funktionen.

Deluppgift 2.3.3:

Skriv in RTN-beskrivning och styrsignalerna nedan för att byta innehållen i register A och R. Register A skall alltså ha värdet 62_{16} och register R 12_{16} efter bytet. Under detta moment får du inte utnyttja DS-modulen. Testa förloppet i kopplingen.

RTN	steg	OE_A	OE_R	LD_A	LD_T	LD_R	C_{in}	f_3	f_2	f_1	f_0
	1										
	2										
	3										

Deluppgift 2.3.4:

Ange en styrsignalsekvens som utför operationer enligt nedanstående RTN-beskrivning, vilken sammantaget utför 4A – T $\rightarrow$ A. (I den här uppgiften ska alla värden och bitmönster tolkas som tal utan tecken.) Utför alla steg i den ordning som anges. Fyll i styrsignalsekvensen i den vänstra delen av tabellen på nästa sida.

$45_{16} \rightarrow A$

$16_{16} \rightarrow T$

$2A \rightarrow A$

$2A \rightarrow A$

$A - T \rightarrow A$

RTN	Steg	*Source*	OE_S	OE_A	OE_R	LD_A	LD_T	LD_R	C_{in}	f_3	f_2	f_1	f_0	c	A	R	T
	1																
	2																
	3																
	4																
	5																
	6																
	7																
	8																

Vid laborationsplatsen:

Utför styrsignalsekvensen i laborationssystemet och kontrollera funktionen. Notera i den högra delen av tabellen c-flaggan från ALU:n och värdet i A-, R- och T-registren.

Vad blev slutresultatet i register A efter sekvensen?

_____

Är resultatet lika med $4_{16}*45_{16} - 16_{16}$? (Kontrollräkna med papper och penna eller miniräknare.)

_____

Kommentera och förklara dina svar på de föregående två frågorna; förklara sambandet mellan hur c-flaggan tändes och huruvida slutresultatet var korrekt.

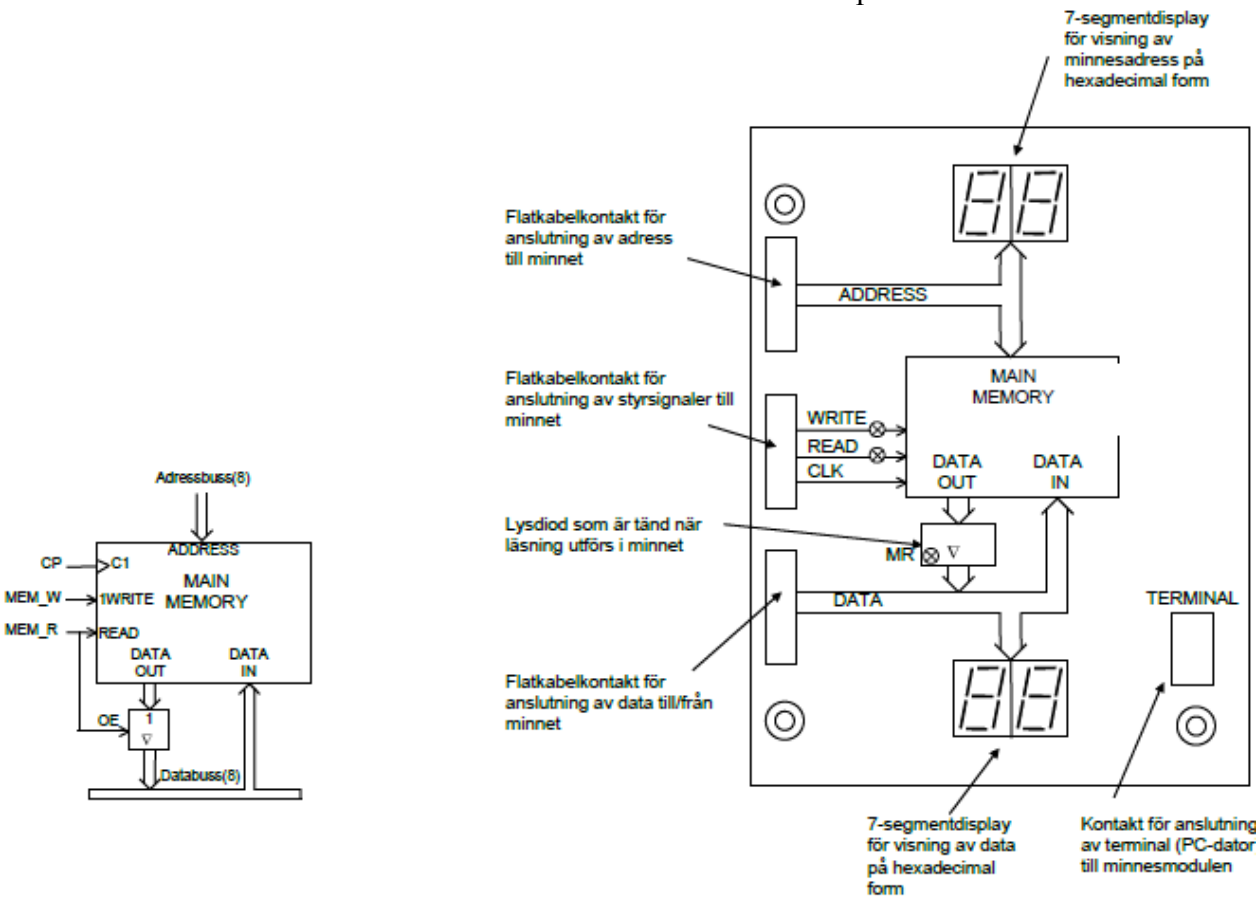
_____

_____

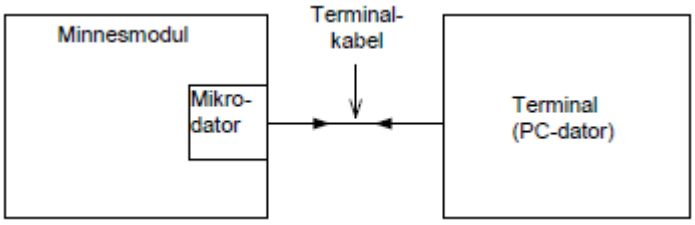
_____

Beskrivning av minnesmodul och terminalanslutning

I labsystemet ingår en modul med läs- och skrivbart minne, RWM, med kapaciteten $2^8 = 256$ st 8-bitars minnesord. Blockschemat för RWM-modulen och dess frontpanel visas nedan.



För att man lättare skall kunna studera och ändra minnesinnehållet har en mikrodator byggts in i minnesmodulen som finns på labplatsen. Mikrodatorn kan kommunicera med en terminal (PC-dator) så som visas i figuren nedan. Kommunikationen sker via terminalkontakten nere till höger på frontpanelen.



Ett program i minnesmodulens mikrodator visar innehållet på hexadecimal form på alla 256 adresser på terminalens skärm. Man kan enkelt ändra ett dataord på en minnesadress genom att först flytta markören på skärmen till den önskade adressen med piltangenterna på terminalens tangentbord. Därefter skriver man in det nya dataordet på hexadecimal form med siffer- och bokstavstangenterna.

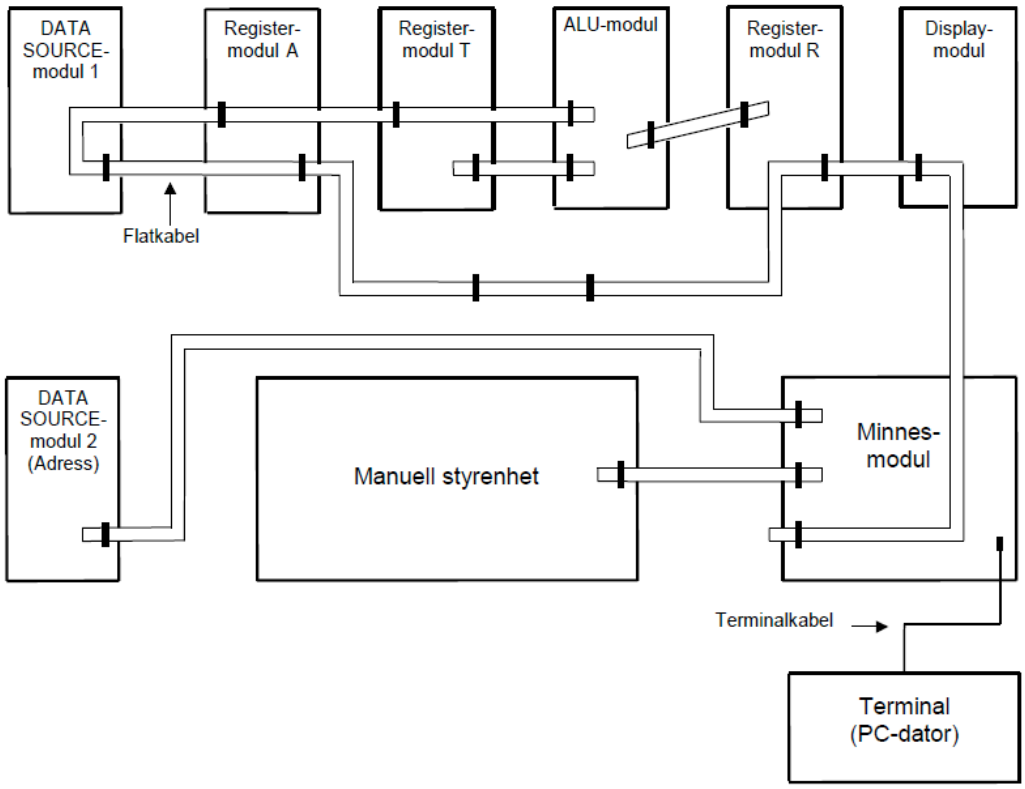
Laborationsuppgift 2.4:

Anslutning av en minnesmodul till datavägen

Du förebereder dig för denna laborationsuppgift genom att studera arbetsbokens kapitel 12.1-12.4, och utföra uppgifterna 12.1 till och med 12.5.

Slå av nätströmbrytaren på digitalmaskinen.

Vi ska nu komplettera dataväg och manuell styrenhet med en minnesmodul.



- Koppla in minnesmodulen enligt figuren ovan. Aktivera OEDS2.
- Starta programmet ”Flexminne” (eller ”HyperTrm”). Det ska finnas en genväg till programmet på skrivbordet. Slå på nätströmbrytaren på digitalmaskinen.
- Observera hur minnets innehåll skrivs till bildskärmen. För att uppdatera ”Flexminne” tryck `Ctrl-A` upprepade gånger. Ange därefter, i kolumn 1, vilka data som är lagrade på följande minnesadresser:

Adress	1	2	3
20_{16}			
$3B_{16}$			
70_{16}			
AA_{16}			
CF_{16}			
FF_{16}			

- Slå av nätströmbrytaren på digitalmaskinen, vänta några sekunder och slå på nätströmbrytaren igen. (Det är aldrig bra att *snabbt* slå av och på nätströmbrytaren på elektroniska apparater!!!)
- Ange därefter, i kolumn 2, vilka data som är lagrade på minnesadresserna. Upprepa förfarandet och fyll slutligen i kolumn 3 med data lagrade på minnesadresserna.
- Diskutera med din laborationspartner; Hittas något ”mönster” i utskriften på skärmen? Jämför med någon annan laborationsgrupp.

Nu skall minnesmodulens funktion undersökas. Följ instruktionerna:

1. Tryck på terminalens ENTER-knapp och iakttag bildskärmen. Fyll hela minnet med nollor genom att hålla siffertangenten 0 nedtryckt tills minnet är fullt.

2. Ställ sedan in adressen 30_{16} på DS-modul 2, som är ansluten till adresskontakten, samt dataordet $5A_{16}$ på DS-modul 1, som är ansluten till datakontakten.

3. Aktivera DS-modul 1 genom att sätta switchen OEDS = 1 på styrenheten. Kontrollera att dataordet $5A_{16}$ finns på databussen.

4. Ställ switchen MEM_W i läget 1 på den manuella styrenheten medan du ser på minnesmodulen.

5. Ge en klockpuls medan du ser på bildskärmen.

6. Fortsätt att skriva och läsa några olika dataord med hjälp av DS-modulerna och styrenheten. Kontrollera att resultatet överensstämmer med informationen på bildskärmen.

7. Prova sedan med att ändra minnesinnehållet med hjälp av terminalens pil- och siffertangenter.

8. Ställ in en viss minnesadress på DS-modul 2 och ställ switchen MEM_R = 1 (läsning) och ändra minnesinnehållet på den valda adressen med hjälp av terminalen. Observera ändringen på minnesmodulens datadisplay.

Du ska nu utföra uppgifterna 12.2 och 12.3 från arbetsboken i laborationssystemet. Eftersom simulator och laborationssystem inte är helt identiska måste du anpassa styrsignalsekvensen för laborationssystemet. Översätt därför resultaten från uppgifterna i arbetsboken för följande manuella styrenhet. Låt laborationssystemets modul DS2 motsvara simulatorns register TA. Eftersom adressen nu läggs direkt till minnets adressbuss räcker det med ett steg. Styrsignaler i simulatorn som inte återfinns i följande tabell kan du bortse från.

Läscykeln

Fyll i laborationssystemets styrsignalsekvens för manuell styrenhet (uppgift 12.2)

M(10_{16}) → A

RTN	steg	*Data Source2*	OE_A	OE_R	OE_{CC}	LD_A	LD_T	LD_R	f_3	f_2	f_1	f_0	MEM_R	MEM_W
10_{16} → MainMemory:ADRESS; M(MainMemory:ADRESS)) → A	*1*													

- Använd terminalen för att lägga in värdet 15_{16} på adress 10_{16} i minnet.
- Nollställ register A
- Utför styrsignalsekvensen och kontrollera att register A nu innehåller 15_{16}.

Skrivcykeln:

Fyll i laborationssystemets styrsignalsekvens för manuell styrenhet (uppgift 12.3)

A → M(11_{16})

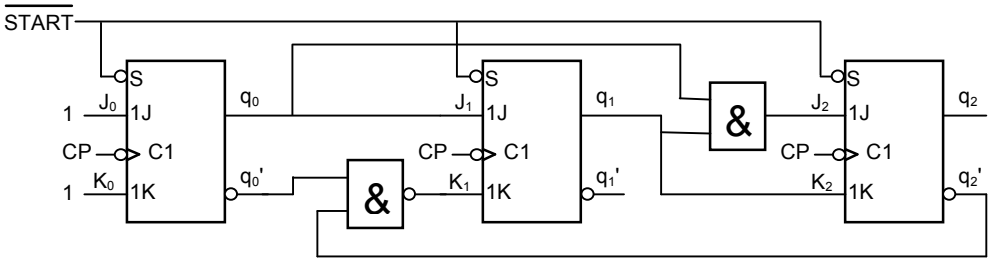
RTN	steg	*Data Source2*	OE_A	OE_R	OE_{CC}	LD_A	LD_T	LD_R	f_3	f_2	f_1	f_0	MEM_R	MEM_W
11_{16} → MainMemory:ADRESS; A → M(MainMemory:ADRESS)	*1*													

- Använd terminalen för att lägga in värdet FF_{16} på adress 11_{16} i minnet.
- Utför styrsignalsekvensen och kontrollera att minnesinnehållet på adress 11_{16} är det samma som innehållet i register A.

Laborationsuppgift 2.5:

Vippor och räknare

Du förebereder dig för denna laborationsuppgift genom att studera arbetsbokens kapitel 8 och 13.



Deluppgift 2.5.1:

Bestäm uttryck för J- och K-ingångarna för vipporna i figuren ovan.

$J_0 =$
$K_0 =$
$J_1 =$
$K_1 =$
$J_2 =$
$K_2 =$

Deluppgift 2.5.2:

Fyll i följande tabell nedan med hjälp av J- och K-uttrycken.

Detta tillstånd				**Insignaler**						**Nästa tillstånd**			
Q	q_2	q_1	q_0	J_2	K_2	J_1	K_1	J_0	K_0	q_2^+	q_1^+	q_0^+	Q^+
0	0	0	0					1	1				
1	0	0	1					1	1				
2	0	1	0					1	1				
3	0	1	1					1	1				
4	1	0	0					1	1				
5	1	0	1					1	1				
6	1	1	0					1	1				
7	1	1	1					1	1				

Deluppgift 2.5.3:

Koppla upp räknaren från föregående sida på digitalmaskinen. Koppla utgångarna q_0, q_1 och q_2 till ingångarna 0, 1 och 2 på displaymodulen. Kontrollera att flatkabeln ej är inkopplad på displaymodulen. Koppla insignalerna START och CP till switchmodulen på digitalmaskinen.

Ge räknaren signalen START och stega sedan igenom med hjälp av CP. Ange utsekvensen för räknaren:

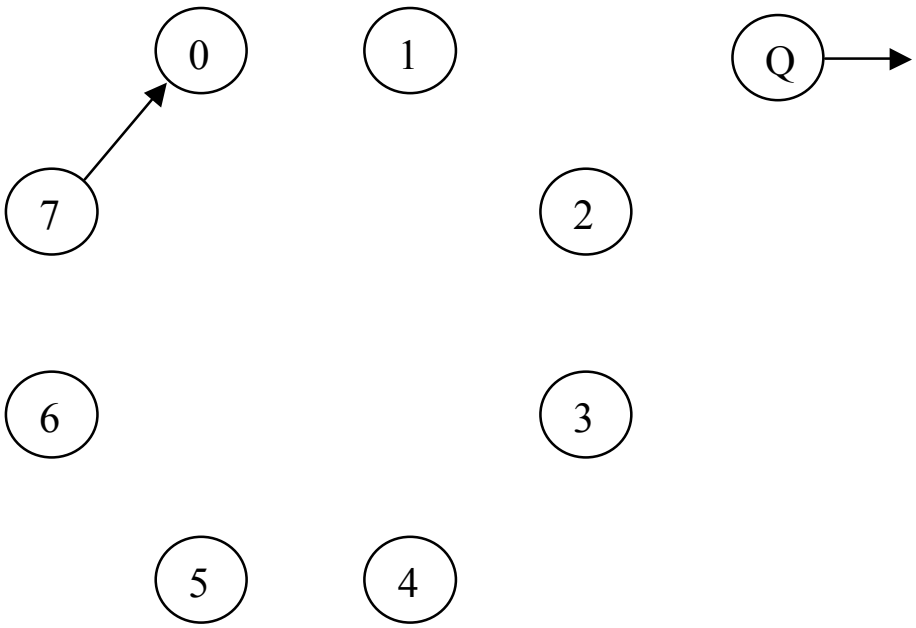
q_2 q_1 q_0: 111, 000, _____

Ange räknarens tillstånd när signalen START aktiveras: _____

Verifiera dina svar här gentemot vad du fick i föregående uppgift.

Deluppgift 2.5.4:

Ange utsekvensen för räknaren i form av en tillståndsgraf:



Slå av nätströmbrytaren, avlägsna alla kablar.

Glöm inte städa upp på din laborationsplats innan du lämnar den.

Laboration nr 3

Konstruktion och test av instruktioner (styrsignalsekvenser) för FLISP

Följande uppgifter ur *Komplement till Arbetsbok för DigiFlisp* ska vara utförda som förberedelse för laborationen. Du ska på begäran av laborationshandledare redogöra för dessa. Uppgifter inom parentes är inte strikt obligatoriska, men arbetet med de obligatoriska uppgifterna blir lättare om man gjort dem.

Uppgifter	14.3	(14.4) LDA #	14.7	14.10	14.11	14.12	14.13
Sign.							

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

Hem-Uppgifter	3.A	3.B	3.C	3.D	3.E
Sign.					

Följande laborationsuppgifter skall redovisas för en handledare för godkännande under laborationen.

Laborations-uppgift	3.3	3.4	3.5
Sign.			

Inledning

Denna laboration består av fem deluppgifter. Under uppgifterna 3.1, 3.2 och 3.3 har du möjlighet att bekanta dig med laborationssystemet och lära dig att använda de grundläggande funktionerna för att därefter, under uppgifterna 3.4 och 3.5 självständigt implementera och testa två helt nya FLISP-instruktioner.

Laborationssystemet består av två delar:

- Dataväg med FLISP styrenhet, "DV-modul" eller kortare "dataväg" (LU3-FLISP-DV)
- Kopplingsplatta för att bilda styrsignaler externt (LU3-FLISP-SE)

Med DV-modulen kan du detaljstudera hur styrsignalsekvenser sätts samman i maskininstruktioner för FLISP. Kopplingsplattan ansluts till DV-modulen via en 64-polig flatkabel. Med kopplingsplattan, som innehåller ett antal AND/OR-nät, kan du bilda styrsignaler och på så sätt skapa godtyckliga styrsignalsekvenser, dvs. nya instruktioner för FLISP.

Två nya FLISP-instruktioner ska konstrueras och testas. Det är helt nya instruktioner så de finns inte i den ordinarie instruktionslistan:

`MOVE  #Data,Adr` *"move immediate data to memory"*

`CMJEQ  #Data,Adr` *"compare register and data, jump if equal"*

För att klara av laborationen under utsatt tid krävs att du förberett dig genom att göra flera hemuppgifter. Observera att det inte är tillräckligt att bara göra uppgifterna från arbetsboken som anges ovan utan du måste också följa anvisningar om hemuppgifterna du får genom att studera detta PM.

Hemuppgift 3.A

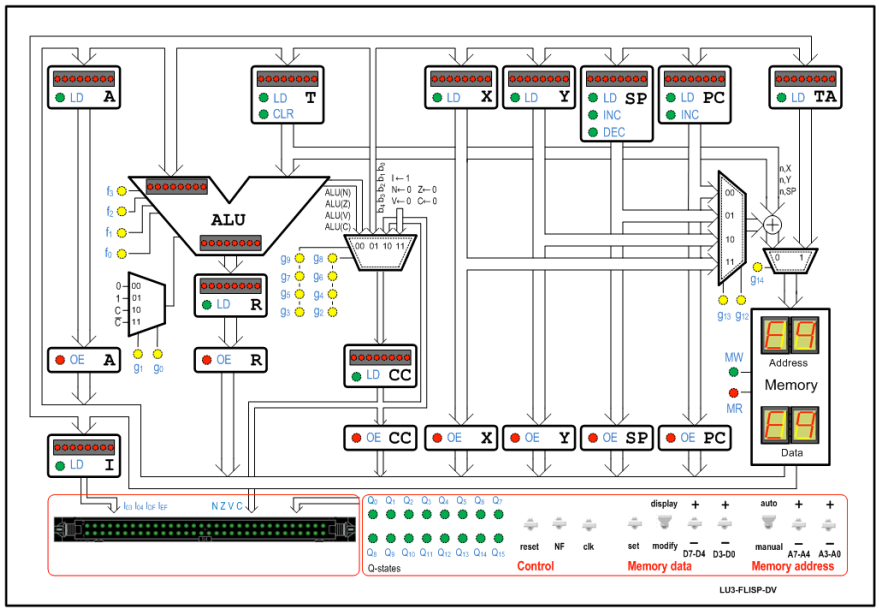
För att kunna testa dina styrsignalsekvenser med simulatorn använder du de enkla testprogram som ges senare i detta PM. Eftersom DV-modulen redan implementerar hela FLISP:s instrukktionsuppsättning behöver du därför normalt sett inte tillhandahålla dessa.

I ett laborationsuppgift 3.3 ska du dock ersätta DV-modulens styrsignalsekvenser med dina egna, för instruktionerna: `JMP Adr`, `DECA`, `STA Adr`.

Du ska därför ha konstruerat och testat dessa instruktioner (se kapitel 14 i arbetsboken) med andra operationskoder (se uppgift 3.3) och spara styrsignalsekvenserna i filen "`lab3_3.fcs`".

Beskrivning av laborationssystemet

Följande bild visar LU3-FLISP-DV, eller kortare ”DV-modul” (datavägsmodul):



LU3-FLISP-DV, ”DV-modul”

OBSERVERA:
På laborationssystemet har trycket av väljaren för CC-registret blivit fel. Ingångarna 3 och 4 är här växlade. Detta gäller dock INTE funktionen.

DV-modulen har dessutom en inbyggd styrenhet för att kunna utföra samtliga FLISP:s instruktioner, 256 bytes primärminne och funktioner för att kunna övervaka och modifiera minnesinnehållet.

De odefinierade operationskoderna $E0_{16}$ och FF_{16} hanterar styrenheten helt enligt FLISP-specifikationen, dvs. med undantagshantering.

De odefinierade operationskoderna 03_{16}, 04_{16}, DF_{16} och EF_{16} hanteras på följande sätt av laborationsenheten:

Då någon av dessa operationskoder finns i instruktionsregistret:

1. aktiveras respektive signal I_{03}, I_{04}, I_{DF} eller I_{EF} till kopplingsplattan
2. alla interna styrsignaler till datavägen inaktiveras, styrsignalerna hämtas nu i stället från kopplingsplattan.

Datavägen och kopplingsplattan kan därför användas för att skapa nya instruktioner för FLISP genom att styrsignalsekvenser bildas med hjälp av Q- och I-signaler som via AND/OR näten återkopplas från kopplingsplattan till datavägen i form av styrsignaler.

För ytterligare beskrivningar av datavägens indikatorer för register, styrsignaler och tillståndssignaler hänvisas till arbetsboken och annan dokumentation av FLISP.

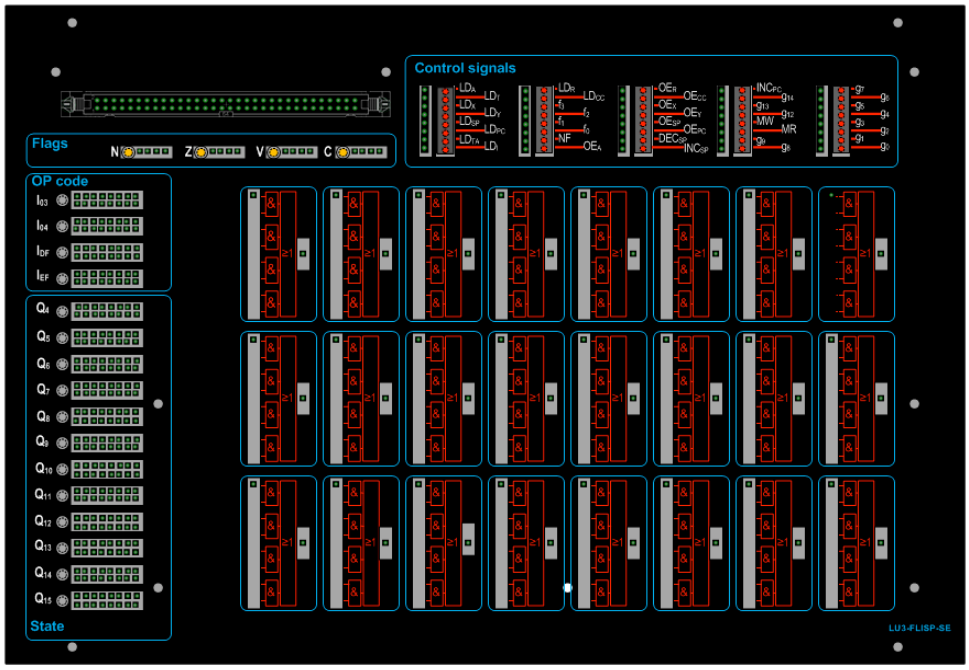
Laborationsenheten kan styras med ett antal strömställare:

Control
- reset - DV-modulen försätts i återställningstillstånd Q_0
- NF - DV-modulen klockas fram till nästa FETCH-fas (stega en instruktion)
- clk - en klockpuls ges till DV-modulen

Memory address
- auto - innehållet i minnesenhetens adressindikator är det värde som kopplats till minnesenheten med styrsignaler g_{12}, g_{13} och g_{14}
- manual - minnesenhetens adressindikator sätts med hjälp av omkopplarna A7-A4, de fyra mest signifikanta adressbitarna och A3-A0, de fyra minst signifikanta adressbitarna.

Memory data
- display - innehållet i minnesenhetens dataindikator ges av innehållet på adressen som finns i adressindikatorn
- modify - minnesenhetens dataindikator sätts med hjälp av omkopplarna D7-D4/D3-D0
- set - om omkopplaren står i modify-läge förs innehållet i dataindikatorn in i DV-modulens minne på den adress som anges i adressindikatorn.

Följande bild visar LU3-FLISP-SE, eller kortare "kopplingsplattan":



LU3-FLISP-SE, ”kopplingsplatta”

Kopplingsplattan har tre sektioner med ingångar, dvs. signaler som kommer *från* DV-modulen, dessa är:

OP code, då någon av de odefinierade operationskoderna 03_{16},04_{16},DF_{16} eller EF_{16} finns i DV-modulens instruktionsregister aktiveras också motsvarande signal I_{03}, I_{04}, I_{DF} eller I_{EF} till kopplingsplattan. Det finns 16 stiftlist för varje signal.

State, anger vilket exekveringstillstånd Q_4-Q_{15} som DV-modulen är i. Även här finns 16 stiftlist för varje signal.

Flags, (N, Z, V, C) från DV-modulens CC-register. Dessa kan användas för att bilda enklare flaggvillkor. Varje signal kan tas ut från något av de fyra stiften på den intilliggande stiftlisten.

Insignalerna kopplas med hjälp av 24, av varandra, oberoende AND/OR-grindnät för att bilda styrsignaler för DV-modulen. Ingångarna på AND-grindarna har en så kallad *weak pull-down*, så logiknivån är noll på en ingång som inte är ansluten. Varje utgång kan kopplas till maximalt tre olika styrsignaler om så skulle krävas.

Styrsignalerna kopplas tillbaks till DV-modulen via sektionen "Control signals" som också har ljusdiodindikatorer för att indikera styrsignalernas nivå.

Laborationsuppgift 3.1

I denna uppgift ska du manuellt lägga in en instruktionssekvens och resetvektor i DV-modulens minne och därefter kontrollera sekvensens funktion genom att utföra programmet cykelvis (klockcykel för klockcykel).

Hemuppgift 3.B

Disassemblera, dvs. tolka minnesinnehållet och använd FLISP:s instruktionslista för att komplettera tabellen med mnemonics. Ange också instruktionens sista tillstånd i exekveringsfasen (det tillstånd i vilket signalen NF aktiveras).

Adress	Maskin-kod	Assemblerkod	Instruktionens NF-tillstånd
20	F0		
21	7E		
22	07		
23	33		
24	22		
FF	20		

På laborationsplatsen:

Så här skriver du in data manuellt till FLISP:s primärminne:

- Ställ Memory address omkopplare i läge manual, och ställ in adressen 20 med omkopplarna A7-A4/A3-A0.
- Sätt Memory data omkopplaren i läge modify och ställ in data F0 med omkopplarna D7-D4/D3-D0. Tryck på set-omkopplaren för att skriva in värdet F0 på adress 20 i minnet.

Upprepa förfarandet för varje adress tills hela instruktionssekvensen och RESET-vektorn lagts in i minnet.

Du kontrollerar instruktionssekvensen på följande sätt:

- Ställ Memory address-omkopplaren i läge auto, Memory data-omkopplaren i läge display.
- Kontrollera att DV-modulen är i tillstånd Q_0, tryck reset annars.
- Utför instruktionssekvensen cykelvis genom att ge klocksignaler, dvs. tryck in clk-omkopplaren.

Laborationsuppgift 3.2

I denna uppgift får du goda tips om hur du testar ett program i DV-modulen.

- Utför programmet instruktionsvis ("stega" igenom programmet)
- Utför programmet utan uppehåll (exekvera programmet)

Hemuppgift 3.C

Använd FLISP:s instruktionslista och komplettera följande tabell genom att översätta sekvensen av mnemonics till maskinkod, dvs. assemblera programmet.

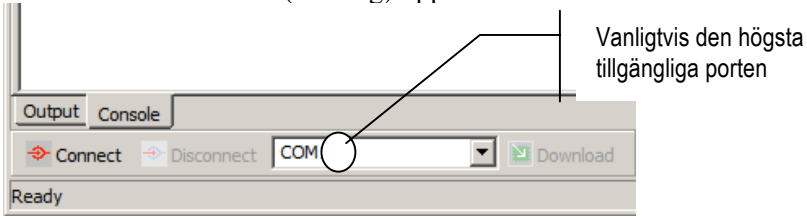
Adress	Maskin-kod	Assemblerkod		
20		L1:	LDA	3_{16}
21				
22		L2	DECA	
23			STA	10_{16}
24				
25			BEQ	L1
26				
27			JMP	L2
28				
FF	20			

Vid laborationsplatsen:

Skriv manuellt in maskinprogrammet i DV-modulens minne på samma sätt som i föregående laborationsuppgift.

DigiFlisp har en inbyggd terminalfunktion som kan användas för att kommunicera med DV-modulen via en USB-anslutning.

- Starta DigiFlisp och välj fliken Console och sedan den COM-port som anvisats av laborationshandledare. Klicka därefter på Connect. Ett terminalfönster (blå färg) öppnas nu.



- Ställ DV-modulens Memory address omkopplare i läge manual, och ställ in adressen 10 med omkopplarna A7-A4/A3-A0. Genom att observera innehållet på denna adress kan du senare se att DEC-instruktionen i programmet utförs korrekt.
- Gör reset på DV-modulen.
- Placera markören i terminalfönstret och ge kommandot **'s'** (*step instruction*) från tangentbordet. För varje gång du ger detta kommando utförs en hel instruktion, på detta sätt blir det enklare att följa programutförande genom instruktioner som är kända att fungera korrekt.
- Stega instruktionsvis några varv i programslingan, observera innehållet på adress 10_{16}.
- Kontrollera att markören är placerad i terminalfönstret och ge kommandot **'e'** (execute). Programmet utförs nu utan att stanna. Studera speciellt innehållet på adress 10_{16} i minnet.

Laborationsuppgift 3.3

I den här uppgiften ska du utföra samma program som i laborationsuppgift 3.2, fast den här gången ska du använda dina egna implementationer av instruktionerna `JMP`, `DECA` och `STA Adr`.

- Eftersom kopplingsplattan endast kan användas för att ange styrsignaler för OP-koderna 03_{16}, 04_{16}, DF_{16} och EF_{16}, kan du låta `JMP` använda OP-koden 03_{16}. Byt därför ut motsvarande OP-kod i programmet från laborationsuppgift 3.2.
- Implementera din lösning för `JMP` (uppgift 14.17 i blå arbetsboken, uppgift 14.12 i komplementet), genom att koppla upp den på kopplingsplattan.
- Bekräfta att programmet får samma beteende som i föregående laborationsuppgift.
    - Utför instruktionssekvensen för programmet cykelvis genom att ge klocksignaler, tills du ser operationskoden 03_{16} i instruktionsregistret.
    - Kontrollera nu, för exekveringsfasen dvs. Q_4, att styrsignalerna aktiveras korrekt.
- Byt ut OP-koden för `DECA` i testprogrammet mot 04_{16}.
- Koppla upp din implementation av instruktionen `DECA` (uppgift 14.15 i blå arbetsboken, uppgift 14.10 i komplementet) på kopplingsplattan. Både `DECA` och `JMP` ska alltså vara uppkopplade samtidigt.
- Bekräfta att programmet fortfarande har samma beteende som tidigare.
    - Utför instruktionssekvensen för programmet cykelvis genom att ge klocksignaler, tills du ser operationskoden 04_{16} i instruktionsregistret.
    - Kontrollera nu, för varje cykel i exekveringsfasen dvs. Q_4 och uppåt, att styrsignalerna aktiveras korrekt.
- Byt ut OP-koden för `STA Adr` i programmet mot DF_{16}.
- Koppla även upp din implementation av `STA Adr` (uppgift 14.12 i blå arbetsboken, uppgift 14.7 i komplementet)) på kopplingsplattan.
- Bekräfta att programmet fortfarande har samma beteende som tidigare.
    - Utför instruktionssekvensen för programmet cykelvis genom att ge klocksignaler, tills du ser operationskoden DF_{16} i instruktionsregistret.
    - Kontrollera nu, för varje cykel i exekveringsfasen dvs. Q_4 och uppåt, att styrsignalerna aktiveras korrekt.
- Visa upp din koppling för en handledare.

Laborationsuppgift 3.4

Data kan kopieras i minnet med en MOVE-instruktion utan användning av de ordinarie "synliga" registren hos FLISP, dvs. utan att förändra innehållet i något av A, X, Y, SP eller CC. Fördelarna med detta är att datakopiering går snabbare, framför allt då de synliga registren är upptagna för annat, eftersom man slipper spara/återställa register med hjälp av stacken. Samtidigt kan hela instruktionen kodas med endast tre bytes. Exempelvis kan instruktionssekvensen

```
PSHA
LDA   #FE16
STA   1016
PULA
```

ersättas av instruktionen

```
MOVE  #FE16,1016
```

Under denna uppgift ska du konstruera och testa instruktionen. Din styrsignalsekvens och ett tillhörande testprogram (visas nedan) ska fungera såväl i simulator som med laborationsutrustningen.

MOVE-instruktionen specificeras enligt följande:

MOVE #Data,Adr

RTN	Data → M(Adr)
Flaggor	Påverkas ej.
Beskrivning	Initierar en minnescell med en konstant.

Detaljer:

Instruktion **MOVE**	Adressering				Operation	Flaggor			
	metod	OP	#	~		N	Z	V	C
`MOVE #Data,Adr`	Imm/ Absolute	DF	3		Data → M(Adr)	-	-	-	-

Instruktionsformat:

DF	Adr	Data

Hemuppgift 3.D

1. Konstruera `MOVE`- instruktionen med hjälp av DigiFlisp:s "Instruction builder", spara styrsignalsekvensen i filen ”`lab_3-4-instruction.fcs`”
2. Skapa ett testprogram, enligt anvisningarna nedan, spara detta filen "`lab_3_4-testprogram.fmem`".
3. Kontrollera att MOVE-instruktionen utförs enligt specifikationen ovan, fyll därefter i tabellen med styrsignaler nedan, du använder tabellen vid laborationen.

- Skapa ett testprogram enligt följande (komplettera med saknad maskinkod), för också in maskinkoden som "setMemory"-direktiv i filen med testprogrammet.

Adress	Maskin-kod		Assemblerkod		Direktiv för att initiera minne
20	DF	L1	MOVE	#FE_{16},10_{16}	#setMemory 20=DF
21	10				#setMemory 21=10
22	FE				#setMemory 22=FE
23		L2	INC	10_{16}	#setMemory
24					#setMemory
25			BEQ	L1	#setMemory
26					#setMemory
27			BRA	L2	#setMemory
28					#setMemory
29					
FF	20				#setMemory FF=20

Vid laborationsplatsen

Du ska nu verifiera att din MOVE-instruktion fungerar även i hårdvara.

- Koppla upp MOVE-instruktionens styrsignaler på kopplingsplattan. Tänk speciellt igenom hur många (eller få) kopplingskablar som behövs. Varje OR-grind har tre utgångar för att kunna driva tre styrsignaler samtidigt.
- Använd DigiFlisp:s nedladdningsfunktion och ladda testprogrammet "lab_3_4-testprogram.fmem" till DV-modulen.
- Gör reset på DV-modulen.
- Utför nu instruktionssekvensen cykelvis genom att ge klocksignaler, tills du ser operationskoden DF_{16} i instruktionsregistret.
- Kontrollera nu, för varje cykel i exekveringsfasen dvs. Q_4 och uppåt, att styrsignalerna aktiveras korrekt. Se till att värdet på minnesadress 10_{16} utvecklas enligt FE_{16}, FF_{16}, 00_{16} för att sedan börja om på FE_{16}.

Då MOVE-instruktionen fungerar som den ska tillkallar du en handledare och redovisar laborationsuppgiften.

Därefter kopplar du ner denna instruktion och fortsätter med nästa uppgift.

Styrsignalsekvens, hemuppgift 3.D

MOVE #Data,Adr

Tillstånd	Summa-term	RTN-beskrivning	Styrsignaler =1

Laborationsuppgift 3.5

Denna uppgift ger exempel på en mer komplex och instruktion än den föregående. Jämförelse, test och villkorlig flödesändring kan utföras med en enda instruktion:

```
CMJEQ    #Data,Adress
```

Liknande funktion fås med instruktionsföljden

```
CMPA  #Data
BEQ   Adress
```

med skillnaden att CMJEQ implementerar ett absolut hopp istället för som i BEQ där hoppet är relativt. I vår nya instruktion anger vi alltså destinationsadressen som en absolut adress, vilket förenklar implementeringen av styrsignalsekvensen något.

Instruktionen specificeras av följande:

CMJEQ Compare register A with data, jump if equal

RTN	A – Data, If Z = 1: Adr → PC
Flaggor	N: Får värdet hos skillnadens teckenbit (bit 7). Z: Ettställs om skillnaden blir noll. V: Ettställs om 2-komplementoverflow uppstår vid subtraktionen C: Ettställs om borrow uppstår vid subtraktionen.
Beskrivning	Data subtraheras från innehållet i register A. Skillnaden lagras ej, utan påverkar endast flaggorna. Därefter testas Z-flaggans värde. Om Z=1 utförs ett hopp till adressen Adr. Om Z=0 utförs inget hopp. Nästa instruktion blir i så fall den direkt efter CMJEQ-instruktionen i minnet.

Detaljer:

Instruktion **CMJEQ** Variant	Adressering metod	OP	#	~	Operation	Flaggor N	Z	V	C
`CMJEQ #Data,Adr`	Immediate/Absolute	EF	3		A – Data, If (Z = 1) Adr → PC	Δ	Δ	Δ	Δ

Instruktionsformat:

EF	Adr	Data

Du får konstruera styrsignalsekvensen hur du vill; det finns inga "prestandakrav". Du kan exempelvis använda följande tips:

Gör en lösning som liknar dem för BEQ- och CMPA-instruktionerna, observera dock att `Adr` här är kodad som absolut adress, och inte relativ som i fallet med BEQ.

1. Läs in `Adr`, dvs. destinationsadress för uppfyllt villkor, placera i register R
2. Läs in `Data` till register T, jämför med register A och ladda CC-registret med flaggor från ALU:n
3. Överför adressen i R till PC (jfr BEQ) om Z=1; avsluta därefter styrsignalsekvensen.

Hemuppgift 3.E

1. Konstruera CMJEQ - instruktionen med hjälp av DigiFlisp:s "Instruction builder", spara styrsignalsekvensen i filen ”lab_3-5-instruction.fcs”
2. Skapa ett testprogram, enligt anvisningarna nedan, spara detta filen "lab_3_5-testprogram.fmem".
3. Kontrollera att CMJEQ -instruktionen utförs enligt specifikationen ovan, fyll därefter i tabellen med styrsignaler nedan, du använder tabellen vid laborationen.

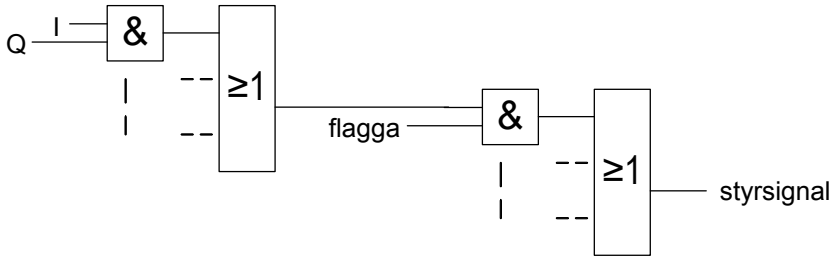
- Skapa ett testprogram enligt följande (komplettera med den saknade maskinkoden), för också in maskinkoden som "setMemory"-direktiv i filen med testprogrammet.

Adress	Maskin-kod		Assemblerkod		Direktiv för att initiera minne
20		L1	CLRA		#setMemory 20=
21		L2	INCA		#setMemory 21=
22			CMJEQ	#2,L1	#setMemory 22=
23					#setMemory 23=
24					#setMemory 24=
25			BRA	L2	#setMemory 25=
26					#setMemory 26=
27					
FF					#setMemory FF=

Vid laborationsplatsen

Verifiera att din CMJEQ -instruktion fungerar även i hårdvara:

- Koppla upp CMJEQ -instruktionens styrsignaler på kopplingsplattan.
  För att skapa villkor för programflöde krävs här en AND-grind med tre ingångar, men någon sådan finns inte på laborationsplatsen, du kan i stället använda två AND/OR-nät enligt följande:



- Gör reset på DV-modulen.
- Utför nu instruktionssekvensen cykelvis genom att ge klocksignaler, tills du ser operationskoden EF_{16} i instruktionsregistret.
- Kontrollera nu, för varje cykel i exekveringsfasen dvs. Q_4 och uppåt, att styrsignalerna aktiveras korrekt.
- Utför programmet tills värdet i register A är 2 och kontrollera att hoppet då blir till adress 20_{16}.

Då CMJEQ-instruktionen fungerar som den ska, tillkallar du en handledare och redovisar laborationsuppgiften.

Därefter kopplar du ner och snyggar till din laborationsplats, laborationen är klar.

Styrsignalsekvens, hemuppgift 3.E

CMJEQ #Data,Adr

Tillstånd	Summa-term	RTN-beskrivning	Styrsignaler =1

Tillägg till PM laboration 3:

Du kan ge kommandon till DV-modulen genom att klicka på terminalfönstret och skriva in något av följande

Kommando	Betydelse
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,t,x,y,s=sp,p=pc,u=ta,r,c=cc.
wmXXYY	Skriv värdet XX till minnesadress YY. Såväl värdet XX som adressen YY anges på hexadecimal form med precis två siffror.

Laboration nr 4

Assemblerprogrammering

Följande uppgifter ur *Arbetsbok för DigiFlisp* ska vara utförda som förberedelse för laborationen. Du ska på begäran av laborationshandledare redogöra för dessa.

Uppgifter inom parentes är inte strikt obligatoriska, men om man gjort dem blir arbetet med de obligatoriska uppgifterna lättare.

Uppg.	(16.6) (16.7) (16.8) 16.9	16.10 16.11
Sign.		

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

Hem-Uppgifter	4.A	4.B	4.C
Sign.			

Följande laborationsuppgifter skall redovisas för en handledare för godkännande under laborationen.

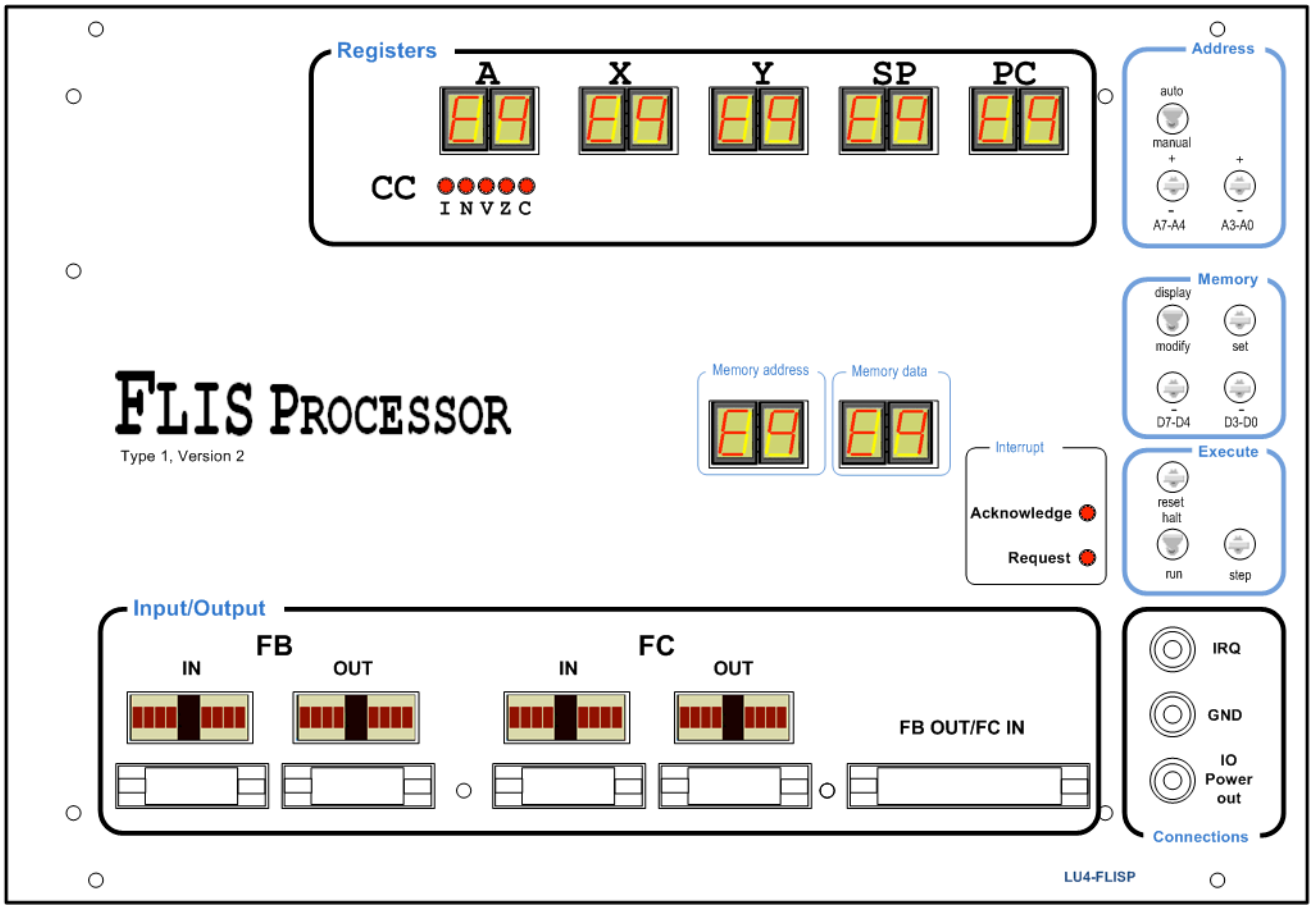
Laborations-uppgift	4.1	4.2	4.3	4.4
Sign.				

OBSERVERA: Om du använder den nya versionen av DigiFlisp kommer IO-enheter att se något annorlunda ut i simulatorn jämfört med laborationsutrustningen. Funktionsmässigt är dom dock lika.

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 (tre 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

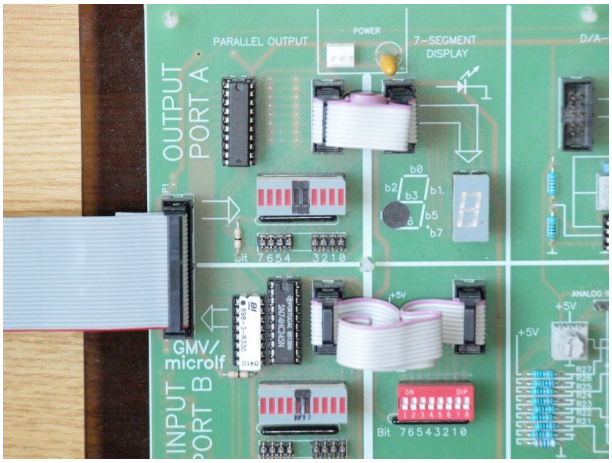
ETERM har inbyggd funktion för att underlätta nedladdning av program och data till laborationsdatorn. Funktionen filtrerar en fil av typen `.fmem` (skapas då du assemblerar din källtext) och skickar setMemory- och setRegister- kommandon till laborationsdatorn.

- 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.fmem` 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.

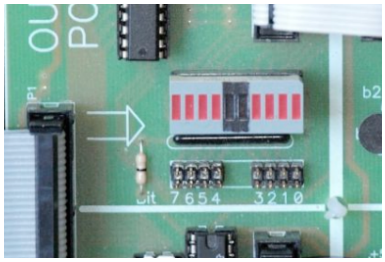
Realtidsegenskaper, rinnande ljus.

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.fmem` för nedladdning.

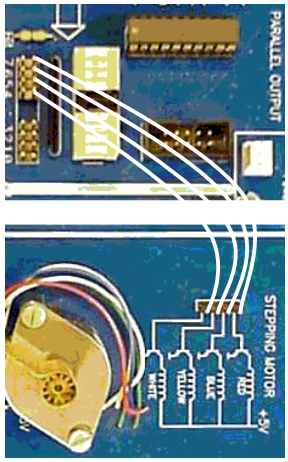
- Gör reset på laborationsdatorn. Kontrollera att programmets startadress nu finns i PC.
- Starta programmet genom att ställa omkopplare i run-läge.
- Variera laborationsdatorns exekveringshastighet och observera skillnader hos ljusdiodrampen.

Laborationsuppgift 4.3.
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 vara 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 i följande ordning: 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:

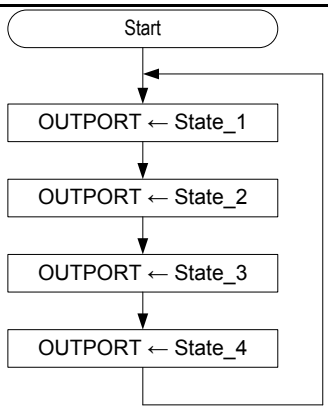
	b7	b6	b5	b4	b3	b2	b1	b0
OUTPORT	RÖD	BLÅ	GUL	VIT	x	x	x	x

Hemuppgift 4.A

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:



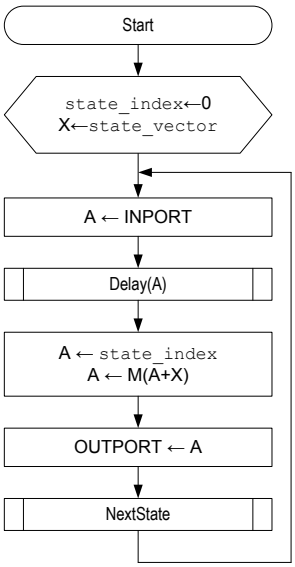
- Redigera en källtextfil `Lab_4-1.sflisp`, enligt flödesplanen, assemblera filen och rätta eventuella fel.
- Notera det inte finns en stegmotor i simulatorn. När du förbereder uppgiften, använd istället en hexdisplay som output för att se att rätt värden skrivs ut till porten i rätt ordning.

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.B

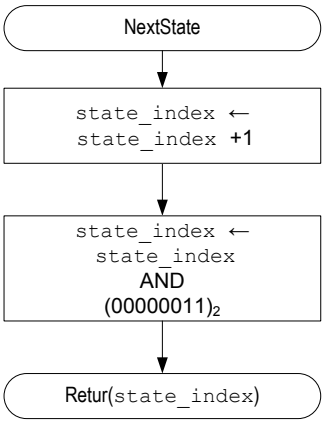
Konstruera ett program för variabel reglering av stegmotorns rotationshastighet enligt följande flödesdiagram:



- 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 pseudokod:

```
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.

Laborationsuppgift 4.4.
Översättning till morsekod

Hemuppgift 4.C

Förbered den här laborationsuppgiften genom att skriva kod för de rutiner som beskrivs nedanför. I beskrivningen av rutinerna nämns en DA-omvandlare och lampa som finns på ML4-kortet. Dessa finns inte i simulatorn – använd istället en diodramp som utenhet. I laborationssystemet är utporten ansluten till adress $FB och inporten till adress $FC.

Målet med den här uppgiften är att skriva ett program som får en lampa att blinka Morsekod för en bokstav, vars ASCII-kod matats in på Dipswitchar. Programmeringen är indelad i tre delar:

1) En rutin som läser in ASCII-kod från Dipswitcharna
2) En rutin som får lampan att blinka en gång, snabbt eller långsamt
3) Ett huvudprogram som använder de två ovanstående rutinerna för att översätta från ASCII-kod till blinkningar

Deluppgift 4.4.1. Inläsning av ASCII-tecken

Från Dipswitcharnas högsta sju bitar, bit1 till bit7, ska en ASCII-kod för en VERSAL bokstav läsas in. Stigande flank på den lägsta biten, bit0, används för att markera att man är klar med att mata in ett tecken. Pseudokod för inmatningsrutinen:

```
Medan bit0 på dipswitchen = 1     // De två looparna väntar på en stigande flank på bit0
      (Gör inget)
Medan bit0 på dipswitchen = 0
      (Gör inget)
A ← Dipswitch
A ← A >> 1                        // Placera ASCII-koden i de lägsta bitarna
Retur
```

Gör en subrutin av ovanstående, anropa den från ett huvudprogram, och säkerställ att rätt värde finns i A-registret när subrutinen returnerar.

Deluppgift 4.4.2. Blinkande lampa

På ML4 finns det en lampa, ansluten till en DA-omvandlare, som i sin tur är ansluten till utporten på FLISP. Ju högre binärtal som skrivs till utporten, desto starkare lyser lampan.

Skriv nu en subrutin som tänder lampan och sedan släcker den efter att en viss tid har gått. Om den mest signifikanta biten i A-registret är ettställd när rutinen anropas, så ska lampan vara tänd i tre sekunder innan den släcks, och om den är nollställd, ska lampan vara tänd i en sekund innan den släcks. Efter att lampan släckts, ska rutinen även skapa en fördröjning på en sekund innan den returnerar. Till er hjälp har ni en fördröjningsrutin, Delay1s, som orsakar en fördröjning på ungefär 1s i simulatorn.

Pseudokod för att blinka med lampan:

```
Pusha A                    // Pusha A-registret för att kunna återställa i slutet av rutinen
Pusha A
Tänd lampan
Pulla A
Om bit7 i A = 1
   Anropa Delay1s tre gånger
Annars
   Anropa Delay1s en gång
Släck lampan
Anropa Delay1s
Pulla A
Retur
```

Gör en subrutin av ovanstående, anropa den från ett huvudprogram, och testa att den fungerar som avsett, både med korta och långa blinkningar.

Deluppgift 4.4.3. En översättare från ASCII till Morse

På Pingpong finns en fil med en tabell med värden, svarande mot Morsekod för alfabetiska tecken. I denna tabell är det två byte för varje bokstav: en byte som anger antalet pulser i koden för en viss bokstav, och en byte med ett bitmönster som anger följden av långa/korta pulser för en viss bokstav. Exempelvis ser första raden i tabellen ut så här:

```
MorseCode     FCB   2,%01000000 ; Morsekod för bokstaven 'A'
```

Detta betyder att första bokstaven i Morsealfabetet har två pulser, och dessa är en kort puls följt av en lång. De sex sista bitarna i bitmönstret för andra byten är godtyckligt satta till 0 i exemplet ovan.

Nu är det dags att sätta samman allt till ett program som läser in ASCII-kod från dipswitcharna och sedan får lampan att blinka motsvarande Morsekod på lampan på ML4. Nedan finns pseudokod för detta.

```
Anropa inläsningsrutinen
A ← A - $61          // Räkna ut ordningstalet för bokstaven som lästes in. (Ordningstalet för 'A' = 0)
A ← 2A               // Eftersom det är två byte per bokstav i tabellen, måste offset dubbleras
Pusha A
X ← Startadress för tabell med Morsekoder
A ← M(X + A)         // Här hämtar vi antalet pulser för bokstaven
Count ← A            // Count är en variabel (en reserverad byte i minnet)
Pulla A
A ← A + 1            // Beräkna offset för pulsmönstret
A ← M(X + A)         // Här hämtar vi pulsmönstret (Morsekoden) för  bokstaven
Repetera så länge som Count ≠ 0
   Anropa blinkrutinen
   A ← A << 1
   Count ← Count – 1
Börja om från början
```

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.