Laboration nr 3 behandlar

Konstruktion och test av instruktioner (styrsignalsekvenser) för FLISP

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

Uppgifter	14.8	14.15	14.16	14.17	14.18
Sign.					

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

Hem-Uppgifter	3.0	3.1	3.2	3.3	3.4

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

Laborations-uppgift	3.3	3.4
Sign.		

Inledning

Denna laboration består av fyra deluppgifter. Under uppgifterna 3.1 och 3.2 har du möjlighet att bekanta dig med laborationssystemet och lära dig använda de grundläggande funktionerna för att därefter, under uppgifterna 3.3 och 3.4 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 dom finns inte i den ordinarie instruktionslistan:

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

`CMBEQ #Data,Adr` *"compare register and data, branch if equal"*

För att klara av laborationen under utsatt tid krävs att du förberett dig genom att göra flera hemuppgifter. Observera att det är inte 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 3.0

För att kunna testa dina styrsignalsekvenser med simulatorn använder du de enkla testprogram som ges senare i detta PM. Testprogrammen omfattar instruktioner ur FLISP:s instruktionsuppsättning, dessa är:

`INC Adr`, BEQ, BRA, `CLRA` och `INCA`.

Börja med att samla styrsignalsekvenser för dessa instruktioner i en konfigurationsfil:

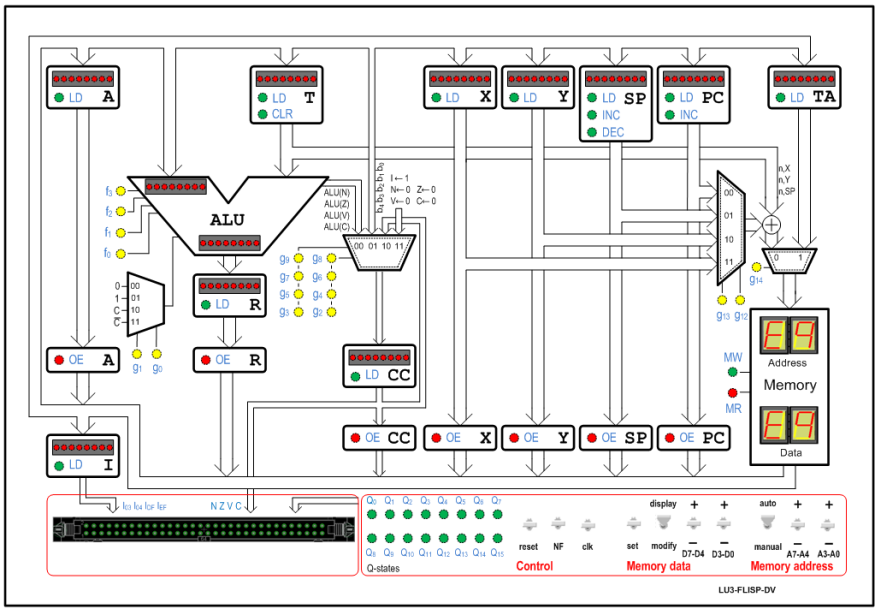
`"lab_3-ins.hwflisp"`.

Observera att förberedelseuppgifterna ovan är likartade, dock inte identiska, instruktioner nämligen DECA, `DEC Adr` och BNE.

Dessutom behöver du ha styrsignalsekvensen för FLISP:s RESET- och FETCH-fas, tillsammans med instruktionen NOP för att testa nya instruktioner i simulatorn. Dessa samlas i filen `"basic.hwflisp"`.

Beskrivning av laborationssystemet

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



LU3-FLISP-DV, ”DV-modul”

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 kopplas 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.

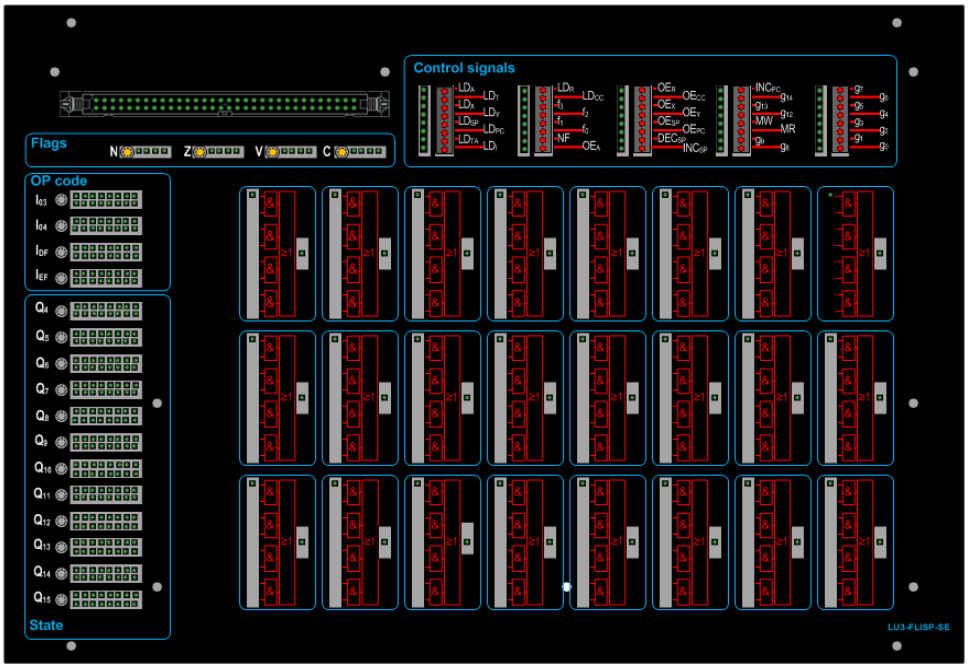
Laborationsenheten kan kontrolleras med ett antal strömställare:

Control
- reset - DV-modulen försätts i återställningstillstånd Q_0
- NF - DV-modulen försätts i FETCH-fas
- 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 någon 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 på en inte ansluten ingång är noll. 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 3.1

Disassemblera, dvs. tolka minnesinnehållet och använd FLISP:s instruktionslista för att komplettera tabellen med mnemonics. Ange också instruktionens sista tillstånd i exekveringsfasen.

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 program utan uppehåll (exekvera programmet)

Hemuppgift 3.2

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	$\#40_{16}$
21				
22			STA	10_{16}
23				
24		L2	INC	10_{16}
25				
26			BEQ	L1
27				
28			BRA	L2
29				
FF	20			

Vid laborationsplatsen:

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

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

- Starta ETERM för FLISP och välj Debug | Terminal och sedan den COM-port som anvisats av laborationshandledare. 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 INC-instruktionen i programmet utförs korrekt.
- Gör reset på DV-modulen.
- Placera markören i terminalfönstret och ge kommando **'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 kommando **'e'** (execute). Programmet utförs nu utan att stanna. Studera speciellt innehållet på adress 10_{16} i minnet.

Laborationsuppgift 3.3

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 3.3

- Du har sedan tidigare konfigurationsfilerna "`basic.hwflisp`" och "`lab_3-ins.hwflisp`" med instruktionerna `INC Adr`, BEQ och BRA som ingår i testprogrammet för den nya instruktionen.
- Skapa en ny konfigurationsfil "`lab_3-3-ins.hwflisp`" för MOVE- instruktionen.
- Konstruera instruktionen och fyll i instruktionens styrsignalsekvens i den avsedda tabellen nedan, för in direktiven i konfigurationsfilen. Observera att tabellens delvis ifyllda raderna endast är avsedda att underlätta arbetet. Dra inga slutsatser om antalet styrsignaler som krävs från detta.
- Skapa ytterligare en konfigurationsfil "`lab_3-3-test.hwflisp`" med ett testprogram enligt följande:

```
lab_3-3-test.hwflisp

# ClearAllMemory
# ClearAllRegisters
# load "basic.hwflisp"
# load "lab_3-ins.hwflisp"
# load "lab_3-3-ins.hwflisp"

... Här följer maskinkoden för testprogrammet, se nedan.
```

- Assemblera ett testprogram enligt följande (komplettera med saknad maskinkod), för också in maskinkoden som "setMemory"-direktiv i konfigurationsfilen 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

- Kontrollera MOVE-instruktionens funktion med hjälp av simulatorn. Rätta eventuella fel.

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 här tre utgångar för att kunna driva tre styrsignaler samtidigt.
- 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. Då hela instruktionen utförts, ska värdet 40_{16} finnas på adress 10_{16} i minnet.

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 3.3

MOVE #Data,Adr

Tillstånd	Summa-term	RTN-beskrivning	Styrsignaler =1	Direktiv i konfigurationsfil
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState

Laborationsuppgift 3.4

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
```

Samma funktion fås med instruktionsföljden

```
CMPA  #data
BEQ   adress
```

I vår nya instruktion anger vi destinationsadressen som en absolut adress vilket förenklar implementeringen av styrsignalsekvensen något.

Instruktionen specificeras av följande:

CMJEQ Compare register A with data, branch 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:

DF	Adr	Data

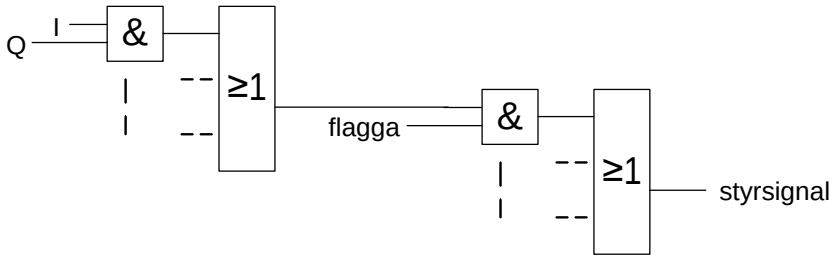
Du får konstruera styrsignalsekvensen som du vill, det finns inga "prestandakrav", du kan exempelvis använda följande tips:

Utgå från BEQ- och CMPA- instruktionerna, observera dock att `Adr` här är kodad som absolut adress.

1. Läs in `Adr`, dvs. destinationsadress för uppfyllt villkor, placera i register R
2. Läs in `Data` till register T, gör jämförelsen med register A och ladda CC-registret med flaggor från ALU:n
3. Villkorlig överföring av adressen i R till PC (jfr BEQ), därefter avslutas styrsignalsekvensen.

Tips vid laborationsplatsen:

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:



Hemuppgift 3.4

- Skapa en konfigurationsfil ”lab_3-4-ins.hwflisp” och konstruera den nya instruktionen. Fyll i instruktionens styrsignalsekvens i den avsedda tabellen nedan
- För att testa instruktionen skapar du ytterligare en konfigurationsfil "lab_3-4-test.hwflisp" med ett testprogram enligt följande:

```
lab_3-4-test.hwflisp

# ClearAllMemory
# ClearAllRegisters
# load "basic.hwflisp"
# load "lab_3-ins.hwflisp"
# load "lab_3-4-ins.hwflisp"

... Här följer maskinkoden för testprogrammet, se nedan.
```

- Assemblera ett testprogram enligt följande (komplettera med den saknade maskinkoden), för också in maskinkoden som "setMemory"-direktiv i konfigurationsfilen 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=

- Kontrollera CMJEQ -instruktionens funktion med hjälp av simulatorn. Rätta eventuella fel.

Vid laborationsplatsen

Verifiera att din CMJEQ -instruktion fungerar även i hårdvara:

- Koppla upp CMJEQ -instruktionens styrsignaler på kopplingsplattan. 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 3.4

CMJEQ #Data,Adr

Tillstånd	Summa-term	RTN-beskrivning	Styrsignaler =1	Direktiv i konfigurationsfil
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState
				# MergeState

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.