LABORATION

DATORTEKNIK Y,C,I
DATORTEKNIK D

Beskrivning av MIA-systemet

Version: 2.5

1983 (BL)
1998 (TS)
2010 (AE,OVA)
2013 (OVA)
2019 (KP)

INNEHÅLL

1. Inledning

Det mest primitiva sättet för en datoranvändare att programmera sin dator är att använda s k maskinspråk. Maskinspråksprogrammering, även kallat assemblerprogrammering, innebär att användaren kombinerar datorns grundläggande maskininstruktioner till ett program. Maskininstruktionerna och deras funktion varierar från dator till dator beroende på datorns uppbyggnad, dess arkitektur.

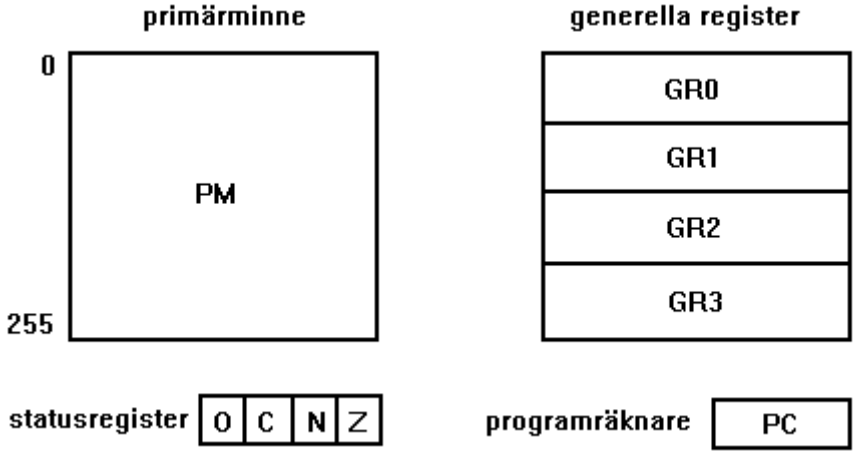
Steget från maskininstruktionernas bitmönster i primärminnet till att dessa utförs på önskat sätt av datorns olika digitala komponenter kan förefalla stort.

Ett sätt att utföra detta är att använda mikroprogrammering. Mikroprogrammet är en sekvens av binära ord, där i princip varje bit i orden styr en enstaka komponent i datorn: räkna upp ett registerinnehåll med ett, skriva i primärminnet o s v.
Genom att kombinera mikroprogrammets ettor och nollor på rätt sätt, kan man hämta maskininstruktioner ur primärminnet och utföra dessa.

Det är tämligen sällsynt att användaren själv kan mikroprogrammera sin dator. Mikroprogrammet och därmed också maskininstruktionernas utseende och funktion bestäms oftast av datortillverkaren. Den dator som används i laborationen är dock helt mikroprogrammerbar.

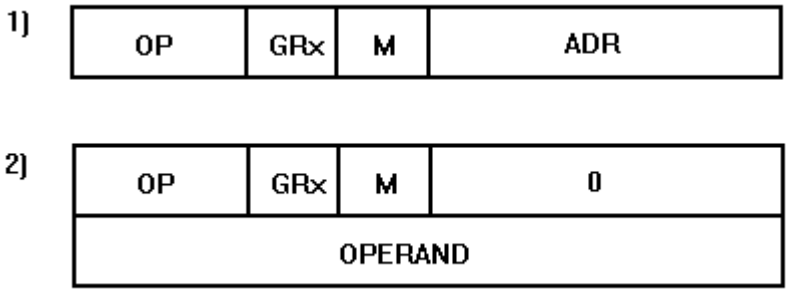
2. Maskinspråksprogrammerarens datormodell

Eftersom lab-datorn är mikroprogrammerbar kan man inom vissa gränser bestämma hur maskininstruktionerna ska se ut och hur de ska utföras. På detta sätt kan man påverka instruktionsrepertoaren hos datorn.
I den här laborationen ska datorn mikroprogrammeras så att en maskinspråks-programmerare får följande modell.



Datorn har 16 bitars ordbredd d v s alla operationer arbetar på 16 bitars ord och registren och primärminnet är 16 bitar breda.
Primärminnet innehåller 256 ord a 16 bitar.
I de fyra generella registren (GRx) kan operander hämtas och resultat lämnas.
Register GR3 kan dessutom användas som adressregister vid indexerad adressering.
Programräknaren (PC= Program Counter) används för att peka ut nästa maskininstruktion i primärminnet.
PC är 8 bitar bred eftersom PM består av 256 ord.

Maskininstruktionerna har följande två format:



Fält	antal bitar	anger
Op	4 bitar	vilken instruktion som avses.
GRx	2 bitar	vilket av de fyra generella registren som ska användas.
M	2 bitar	anger adresseringsmetod.
ADR	8 bitar	adressfält
Operand	16 bitar	operand

M fältet anger vilken adresseringsmetod som ska användas och därmed indirekt det aktuella instruktionsformatet. Följande möjligheter finns:

M=00 Direkt adressering, instruktionsformat 1. Innehållet i ADR fältet är adressen till den minnescell i vilken operanden finns.

M=01 Omedelbar operand, instruktionsformat 2. Det efterföljande ordet innehåller operanden.

M=10 Indirekt adressering, instruktionsformat 1. ADR fälet innehåller adressen till den minnescell där adressen till operanden finns.

M=11 Indexerad adressering med GR3, instruktionsformat 1. Summan av innehållet i ADR fältet och innehållet i GR3 är adressen till operanden.

Operandens verkliga adress som fås efter adressberäkningen kallas effektiv adress.

Instruktioner som arbetar med underförstådd operand ska ha M fältet satt till 00.

De instruktioner som styr programförloppet är alltid relativadresserande. M fältet ska även här sättas till 00. För dessa instruktioner ska ADR-fältet tolkas som ett 8 bitars tvåkomplementtal, vilket adderas till PC vid hopp i programmet.

3. Mikroprogrammerarens datormodell

För att man ska kunna mikroprogrammera en dator krävs att man har en fullständig kännedom om dess arkitektur och dessutom känner till utseendet av styrorden i mikroprogrammet. Av figuren på nästa sida framgår att mikroprogrammeraren jämfört med maskinspråksprogrammeraren måste känna till betydligt fler detaljer i datorn och hur dessa samverkar med varandra.

En dator kan grovt delas upp i tre huvuddelar: styrenhet, aritmetisk enhet och minnesenhet. Dessa tre delar måste kunna överföra data mellan varandra och det sker i vår dator via en buss. Till bussen ansluts ett sändande och ett mottagande register varefter data överförs från det sändande till det mottagande registret.

Minnesenheten består i vår dator av primärminnet (PM), adressregistret (ASR) och programräknaren (PC). I PM lagras maskinprogrammet och de data programmet ska arbeta på. ASR används för att adressera PM. PC pekar ut nästa maskininstruktion i PM.

Aritmetiska enheten är den del som utför själva "jobbet". Den är inblandad så fort någon adressberäkning eller operandmodifiering ska göras. Den består främst av ALUn (Arithmetic Logic Unit) och ackumulatorregistret (AR). ALUn är ett kombinatoriskt nät som kan utföra ett flertal operationer på sina två operander. Resultatet av en operation hamnar alltid i AR. I aritmetiska enheten ingår också hjälpregistret (HR) och de generella registren (GRx).

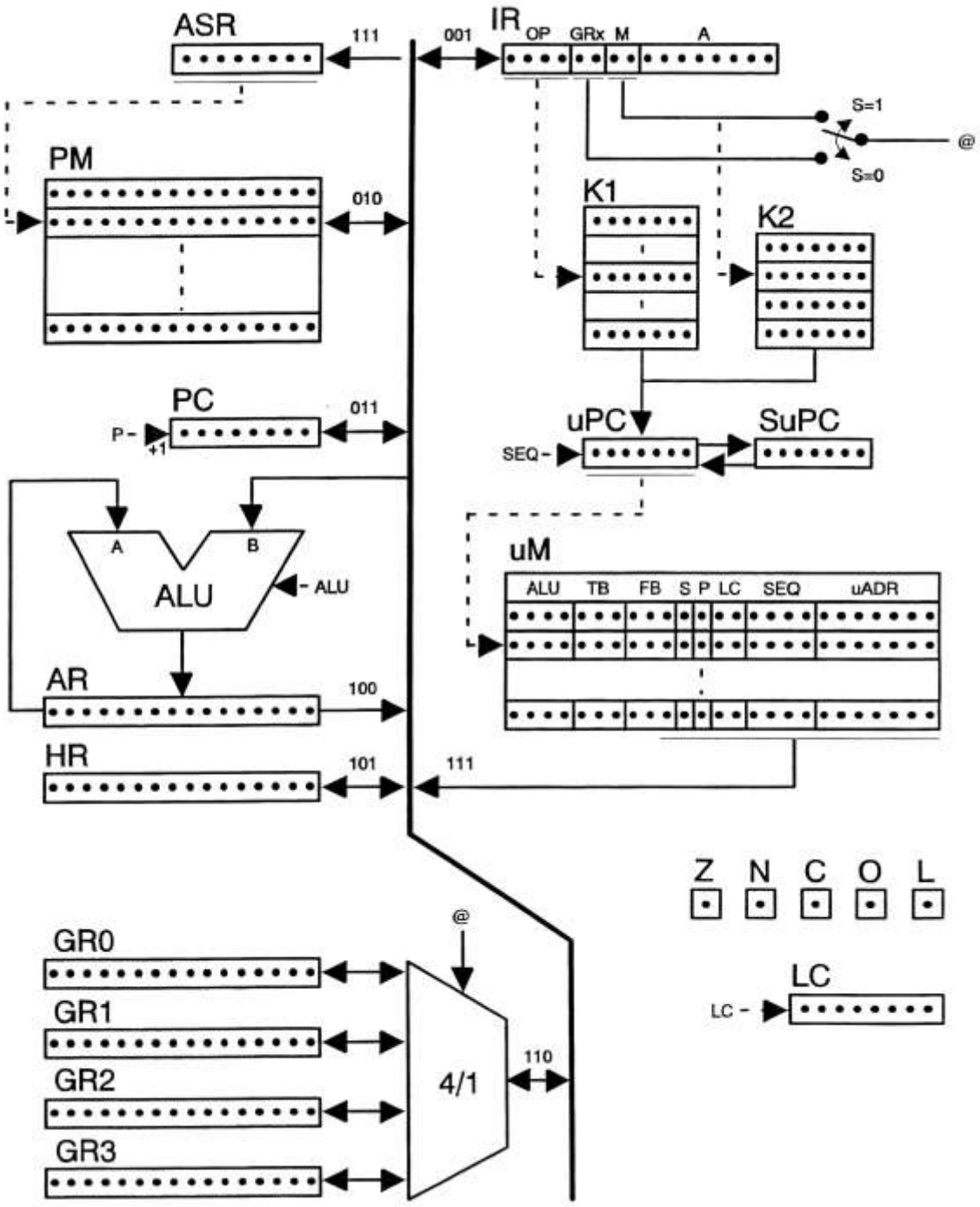
Den för oss viktigaste delen i datorn är dock styrenheten. Som namnet antyder styr den de övriga delarna. Den bestämmer mellan vilka register dataöverföringen via bussen ska ske, vilken operation ALUn ska utföra o s v. Dessutom styr den sig själv. Styrningen består i att en kontinuerlig ström av styrord eller mikroinstruktioner genereras. Dessa mikroinstruktioner har av mikroprogrammeraren lagts in i mikroprogramminnet (uM). uM adresseras av mikroprogramräknaren (uPC). uPC räknas hela tiden upp av en klocka och mikroinstruktionerna (styrorden) läggs alltså ut i takt med denna klocka. Vissa mikroinstruktioner kan också styra uPC så att en ny adress laddas in i uPC och hopp utförs i mikroprogrammet.

Förutom uM och uPC ingår i styrenheten instruktionsregistret (IR), operations-avkodningsnätet (K1), märkfältsavkodningsnätet (K2), registret för sparad uPC(SuPC), loopräknaren (LC) och statusvipporna (Z,N,C,O,L).

En dator arbetar som bekant i tvåtakt:
1) maskininstruktionen hämtas. 2) maskininstruktionen utförs.
Instruktionshämtningen sker likadant för alla maskininstruktioner, däremot är exekveringen givetvis olika.

I datorn finns ett speciellt register, instruktionsregistret (IR), till vilket den maskininstruktion som ska utföras hämtas. Nätet K1 ger sedan en adress i mikroprogrammet som motsvarar instruktionens OP-fält. Mikroprogrammet hoppar dit och motsvarande styrord genomlöps.

Mikroprogrammerarens datormodell



En mer detaljerad beskrivning av mikroprogrammförloppet skulle kunna se ut så här:

Instruktionshämtning

PC ansluts som sändare till bussen och ASR som mottagare. Därvid kommer PCs innehåll att överflyttas till ASR och vi adresserar nästa maskininstruktion i PM.
I nästa styrord ansluter vi PM och IR till bussen och överför instruktionen till IR.
Vi förutsätter här att minnets accesstid är kortare än periodtiden hos klockan.
Vi räknar upp PC med ett så att den pekar på nästa instruktion.

Innan vi utför exekveringsdelen är det lämpligt att beräkna instruktionens effektivadress enligt M fältet och placera denna i ASR.
Adressberäkningen är lika för alla instruktioner som adresserar minnet. Detta görs genom att m h a K2 hoppa till olika styrordssekvenser beroende på innehållet i M-fältet. Efter adressberäkningen sker ett hopp, m h a K1, till den styrordssekvens som utför exekveringen.

Exekvering

Den styrordssekvens som motsvarar maskininstruktionens OP-fält genomlöps.
Därefter sker ett hopp tillbaka till den del som utför en ny instruktionshämtning.

Nedanstående exempel visar hur operationen ADD skulle kunna utföras:

AR:= GRx
AR:= M(EA)+AR
GRx:= AR, hopp till instruktionshämtning

EA är den tidigare beräknade effektivadressen.

3.1 Beskrivning av de ingående komponenterna

3.1.1 Bussen

All datatransport i datorn sker över bussen. Den är 16 bitar bred.
Fälten TB (Till Buss) och FB (Från Buss) i styrordet anger vilket register som ska sända respektive mottaga data från bussen.
Endast ett register åt gången kan fungera som sändare och endast ett register kan fungera som mottagare vid varje enskild bussöverföring.

3.1.2 Primärminnet och adressregistret

Primärminnet består av 256 stycken ord om 16 bitar. PM har en dubbelriktad kommunikation med bussen. ASR pekar ut det aktuella ordet i PM. ASR är 8 bitar brett och kan endast skrivas i. Vid skrivning överförs bussens 8 minst signifikanta bitar till ASR.

3.1.3 Programräknaren

PC är ett 8 bitars register som innehåller adressen till nästa maskininstruktion i PM. PC har dubbelriktad kommunikation med bussen, men endast de 8 minst signifikanta bitarna används. Vid sändning till bussen nollställs de 8 mest signifikanta bitarna. PC räknas upp med ett genom att aktivera biten P i styrordet.

3.1.4 ALU, ackumulatorregistret och hjälpregistret

ALUn (Arithmetic Logic Unit) utför alla beräkningar. Den kan utföra aritmetiska, logiska och skiftoperationer. Vilken operation som ska utföras anges av ALU fältet i styrordet.

ALUn tar vid operationer av två-operandtyp den ena operanden från bussen och den andra från AR. Detta innebär att något register alltid måste vara anslutet till bussen då ALUn används på detta sätt.
Resultatet av en operation placeras i AR.

Vid ALU operationer påverkas statusflaggorna Z,N,C och O, se 3.1.8.

AR är 16 bitar brett och kan endast sända till bussen. ALUn arbetar oftast på 16 bitars operander men vissa skiftoperationer kan även utföras på 32 bitar. I detta fall innehåller HR de 16 minst signifikanta bitarna och AR de 16 mest signifikanta bitarna.

HR har dubbelriktad kommunikation med bussen och kan därmed också användas som mellanlagringsregister av mikroprogrammet.

3.1.5 Generella register

De 16 bitar breda generella registren har dubbelriktad kommunikation med bussen. Vilket register som avses bestäms av 4/1-multiplexern. Denna styrs av antingen GRx- eller M-fältet i instruktionsregistret IR. Biten S i styrordet väljer ut GRx-fältet eller M-fältet som styrsignal. GRx-fältet får styra då man hämtar/lämnar en operand medan M-fältet ska styra vid adressberäkningar. M=11 motsvarar ju att GR3 används som indexregister.

3.1.6 Instruktionsregistret

IR är 16 bitar brett och har dubbelriktad kommunikation med bussen. IR ska innehålla den maskininstruktion som för tillfället utförs. IR har uppdelats i fält efter maskininstruktionsformatet. OP-fältet är insignal till K1, GRx-fältet kan vara styrsignal till 4/1-muliplexern och M-fältet är insignal till K2 och kan vara styrsignal till 4/1 multiplexern.
ADR-fältet kan användas genom att ansluta IR till bussen. ADR-fältet blir då de 8 minst signifikanta bitarna på bussen.

3.1.7 Mikroprogramräknaren, K1, K2 och registret SuPC

uPC är ett 7 bitars register som pekar ut aktuellt styrord i uM. Fältet SEQ i styrordet bestämmer om uPC ska räknas upp med ett (vilket är det vanligaste), nollställas (vilket motsvarar att instruktionshämtningen börjar), eller laddas med det värde som K1 eller K2 anger.

Nätet K1 som är ett minne med 16 ord a 7 bitar används för att hoppa till den styrordssekvens som motsvarar OP-fältet i IR. Varje maskininstruktion ska alltså, på den plats i K1 som motsvarar dess OP-kod, ha adressen till den styrordssekvens i uM som exekverar instruktionen.

Nätet K2 är ett minne med 4 ord a 7 bitar och används på liknande sätt vid effektivadressberäkningen. Effektivadressen ska ju beräknas olika för olika värden i märkfältet M. K2 ska alltså innehålla adresserna till de olika styrordssekvenser som beräknar effektivadressen.

Andra typer av hopp i mikroprogrammet kan också göras. Fältet uADR i styrordet anger då en hoppadress. Hopp till denna adress kan ske ovillkorligt, villkorligt eller som subrutinhopp. Vid ovillkorligt hopp sker alltid hopp till uADR. Vid villkorligt hopp måste vissa av statusvipporna vara ettställda för att hopp ska ske.
Om vipporna inte är ettställda räknas istället uPC upp med ett.
En enkel form av subrutinanrop i mikroprogrammet kan också göras.
Då räknas uPC upp med ett, sparas i SuPC och ovillkorligt hopp görs till uADR.
Vid subrutinåterhopp laddas uPC från SuPC och mikroprogrammet fortsätter med styrordet efter det som åstadkom subrutinanropet.

3.1.8 Statusvipporna Z,N,C,L och loopräknaren

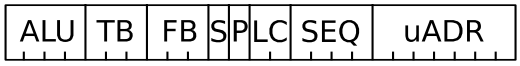
Varje gång en operation utförs i ALUn sätts vipporna Z,N,C och O efter resultatet. Dessa vippor kan testas i villkorliga hopp i mikroprogrammet och olika styrordssekvenser kommer att väljas beroende på resultatet av en operation.

Z=1	då resultatet i AR=0
N=1	då den mest signifikanta biten av resultatet i AR är ettställd, d v s då resultatet tolkat som tvåkomplementtal är negativt.
O=1	då en aritmetisk operation orsakat spill.
C=1	vid minnessiffra från addition eller subtraktion. Efter en skiftoperation innehåller C den utskiftade biten.
L=1	Då loopräknaren (LC) = 0.

Loopräknaren (LC) är en 8 bitars räknare som används vid loopar i mikroprogrammet. Den styrs av LC-fältet i styrordet. LC kan räknas ned med ett, laddas från uADR-fältet (endast 7 bitar, den mest signifikanta biten blir noll), eller laddas från bussens 8 minst signifikanta bitar.

3.2 Mikroinstruktionsformat

Mikroinstruktionerna eller styrorden ligger i mikrominnet uM. uM består av 128 ord a 25 bitar och är liksom K1 ch K2 ett programmerbart permanentminne.



Styrordets 25 bitar är uppdelade i 8 fält med olika funktion. Se figuren ovan.

fält	funktion
ALU	Väljer funktion hos ALUn.
TB	Vilket register som ska sända till bussen.
FB	Till vilket register informationen på bussen ska skrivas.
S	Väljer instruktionens GRx- eller M-fält som styrsignal till 4/1 multiplexern.
P	Räknar upp PC med ett.
LC	Styr loopräknaren.
SEQ	Styr mikroprogrammets sekvens d v s hopp i mikroprogrammet.
uADR	Innehåller adressen till nästa styrord vid vissa typer av hopp i mikroprogrammet. Fältet kan alternativt innehålla det värde som ska laddas till loopräknaren.

3.2.1 ALU-fältet, bit 0-3

b0-b3	funktion	påverkar flaggor
0000	Ingen funktion	nej
0001	AR:= buss	nej
0010	AR:= buss', (ettkomplementet)	nej
0011	AR:= 0	Z,N
0100	AR:= AR+buss	Z,N,O,C
0101	AR:= AR-buss	Z,N,O,C
0110	AR:= AR AND buss	Z,N
0111	AR:= AR OR buss	Z,N
1000	AR:= AR+buss	nej
1001	skifta AR ett steg vänster, noll skiftas in	N, Z, utskiftad bit till C
1010	skifta ARHR ett steg vänster (32 bitars skift), noll skiftas in	N, Z, utskiftad bit till C (N, Z gäller för 32 bitar)
1011	skifta AR ett steg höger aritmetiskt, teckenbiten skiftas in	N, Z, utskiftad bit till C
1100	skifta ARHR ett steg höger aritmetiskt (32 bitars skift), teckenbiten skiftas in	N, Z, utskiftad bit till C (N, Z gäller för 32 bitar)
1101	skifta AR ett steg höger logiskt, noll skiftas in	N, Z, utskiftad bit till C
1110	rotera AR ett steg vänster	N, Z, utskiftad bit till C
1111	rotera ARHR ett steg vänster(32 bitars skift)	N, Z, utskiftad bit till C (N, Z gäller för 32 bitar)

3.2.2 TB-fältet, bit 4-6

b4-b6	till bussen	kommentar
000	inget	data på bussen odefinierat
001	IR	
010	PM	
011	PC	
100	AR	
101	HR	
110	GRx	
111	styrordets b9-b24	endast ALU- och TB- fälten i styrordet fungerar, men uPC räknas upp med ett (anv. för konstanter)

3.2.3 FB-fältet, bit 7-9

b7-b9	skrivning till
000	inget
001	IR
010	PM
011	PC
100	odefinierad
101	HR
110	GRx
111	ASR

3.2.4 S-biten, bit 10

b10	funktion
0	Instruktionsregistrets GRx-fält styr 4/1 multiplexern
1	Instruktionsregistrets M-fält styr 4/1 multiplexern

3.2.5 P-biten, bit 11

b11	funktion
0	PC ändras ej
1	PC räknas upp med ett

3.2.6 LC-fältet, bit 12-13

b12,b13	funktion
00	LC påverkas ej
01	LC räknas ned med ett
10	LC laddas från bussens 8 minst signifikanta bitar
11	LCs 7 minst signifikanta bitar laddas från uADR fältet, msb=0

3.2.7 SEQ-fältet, bit 14-17

b14-b17	funktion
0000	uPC räknas upp med ett
0001	uPC laddas från K1
0010	uPC laddas från K2
0011	uPC nollställs
0100	hopp till uADR om Z=0, annars uPC+1
0101	ovillkorligt hopp till uADR
0110	subrutinhopp till uADR (SuPC:= uPC+1, uPC:= uADR)
0111	subrutinåterhopp (uPC:= SuPC)
1000	hopp till uADR om Z=1, annars uPC+1
1001	hopp till uADR om N=1, annars uPC+1
1010	hopp till uADR om C=1, annars uPC+1
1011	hopp till uADR om O=1, annars uPC+1
1100	hopp till uADR om L=1, annars uPC+1
1101	hopp till uADR om C=0, annars uPC+1
1110	hopp till uADR om O=0, annars uPC+1
1111	nollställ uPC och avbryt exekveringen (HALT)

3.2.8 uADR-fältet, bit 18-24

De 7 bitarna representerar nästa mikroprogramadress vid hopp i mikroprogrammet. De kan också vara det värde som loopräknaren ska laddas med.

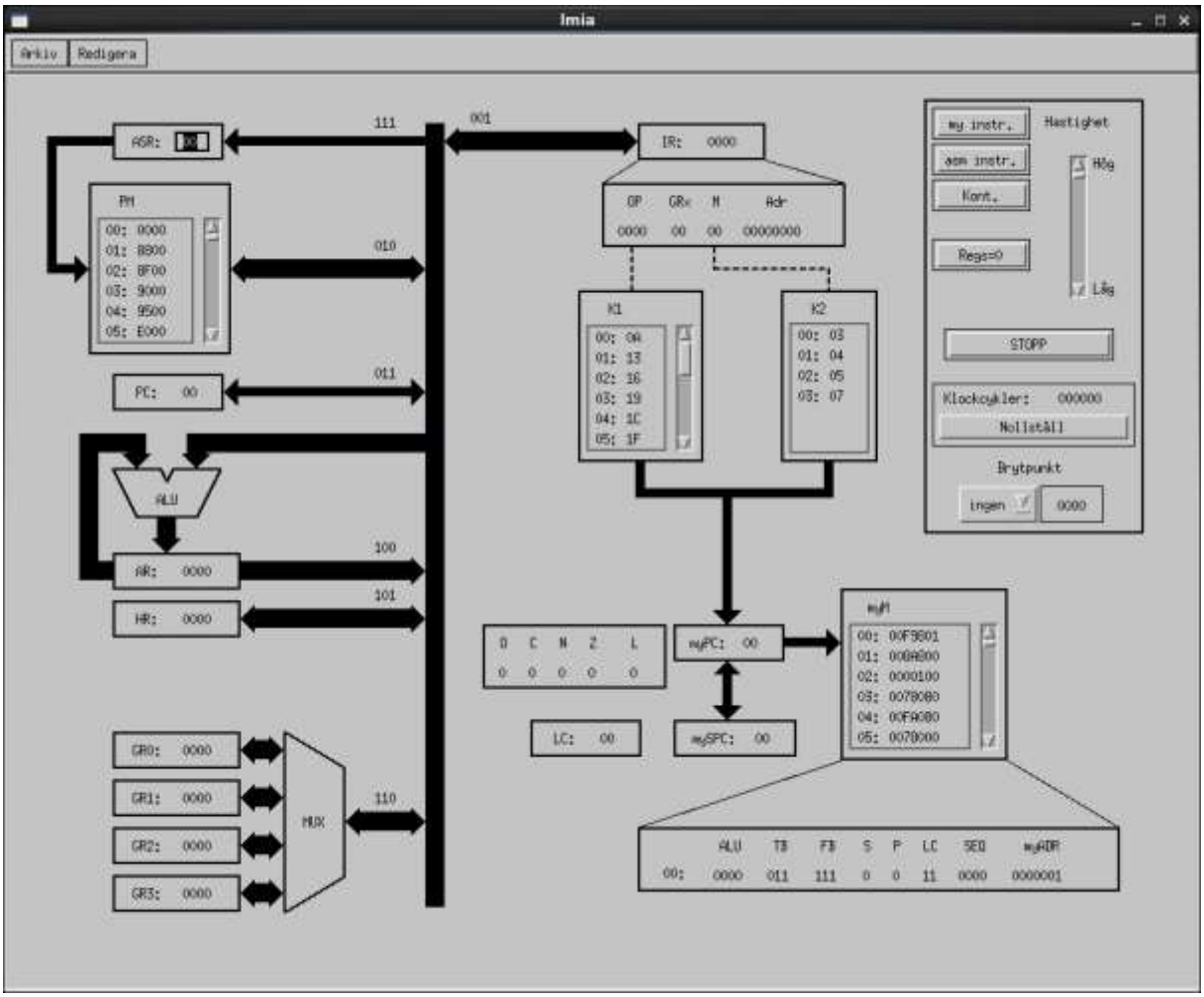
4. Laborationsutrustningen

Labdatorn körs på ISY/IDA:s elevsystem som kör linux. Labdatorn startas med följande kommandosekvens:

```
module add courses/TSEA28
lmia
```

Det går även att köra detta program hemifrån, se
http://www.ida.liu.se/local/students/remote
för mer information.

Skärmen ger en fullständig bild av labdatorns tillstånd.
Samtliga register, minnen och flaggor kan avläsas och ändras.
Skärmen visar alltid labdatorns status, också när ett program exekveras.



4.1 Programmering

All programmering sker hexadecimalt eller binärt beroende på var markören befinner sig. Man väljer var man vill ändra genom att peka med markören på önskat ställe och "klicka" en gång. Om hela registerinnehållet ska ändras så "dubbelklickar" man på det valda registret, varefter nya tecken kan matas in.

Under mikrominnet finns ett fält där binärrepresentationen av den utvalda mikroprogramraden visas. Varje delfält (t ex ALU fältet) kan här ändras enligt samma principer som ovan. Samma sak beträffande uppdelning i binära fält gäller även för instruktionsregistret.

I Redigeramenyn erbjuds en möjlighet att flytta hela block av data inom primärminnet eller mikrominnet. Det finns också en möjlighet att ladda testdata lämligt för sorteringsuppgiften.

I Arkivmenyn erbjuds en möjlighet att spara den inprogrammerade datormodellen inklusive registerinnehåll och flaggor. En sparad modell kan sedan återladdas vid ett senare tillfälle.

Regs=0 Ger en nollställning av alla register.

4.2 Exekvering

Efter det att man matat in sina värden kan man starta en exekvering genom att aktivera någon av kontrollfunktionerna.

my instr Exekverar en rad i taget i mikrominnet (den som myPC pekar på). Resultatet av operationen kan följas genom att studera hur de olika registren ändras.

asm instr. Exekverar en maskininstruktion i PM. Här kommer således samtliga mikroinstruktioner som behövs för denna maskininstruktion att utföras.

Kont. Kontinuerlig exekvering av det inmatade maskinprogrammet i PM. Maskinprogrammet avbryts då en mikroinstruktion utförs som innehåller 1111 i SEQ fältet.

STOPP Avbryter all exekvering.

Brytpunkt Om valt register uppnår valt värde avbryts pågående exekvering.

Det går även att justera hastigheten på exekveringen, samt avläsa hur många klockpulser som hittills har förflutit.