Computerarkitektur

- en introduktion til computerarkitektur med LINDA

faraz@butt.dk — **Faraz Butt**
mads@danquah.dk — **Mads Danquah**
doktor@dyregod.dk — **Ulf Holm Nielsen**

Roskilde Universitetscenter
Naturvidenskabelig Basisuddannelse Hus 14.2
3. semester

Forord

Dette hæfte er ment som en introduktion i maskinarkitektur til datalogiundervisningern i gymnasiet. Hæftet kan bruges selvstændigt eller i sammenhæng med det medfølgende program LINDA. Arbejdet med det tilhørende undervisningsforløb opfylder bekendtgørelsens krav om maskinarkitektur. Materialet forudsætter et begrænset kendskab til Java.

Materialet er lavet i forbindelse med et 3. semester projekt på Naturvidenskabelig Basisuddannelse på RUC.

I forbindelse med udarbejdelsen af dette materiale har Thorkild Skjeldborg været en stor hjælp.

Hvis du har kommentarer til materialet så send en mail til Ulf Holm Nielsen på doktor@dyregod.dk eller til Mads Danquah på mads@danquah.dk.

København, januar 2001

Faraz Butt
Mads Danquah
Ulf Holm Nielsen

Indhold

Forord **i**

1 En simpel computer og hvordan den virker **1**
1.1 Introduktion 1
1.2 En simpel model 2
1.3 Komponenterne 2
1.3.1 Lager 2
1.3.2 Bus 3
1.3.3 Processor 3

2 Programmering af computeren **5**
2.1 Maskinniveauer 5
2.2 Maskinkode 6
2.2.1 Instruktioner 7
2.3 Maskinkode i LINDA 8
2.3.1 Indtastning af maskinkode i LINDA 8
2.3.2 Afvikling af programmer i LINDA 9
2.4 Øvelser 9

3 Mikrokode **11**
3.1 En udvidet computermodel 11
3.2 Introduktion til mikrokode 12
3.3 Mikrokode i LINDA 14
3.4 Øvelser 16

A Instruktionssæt **18**

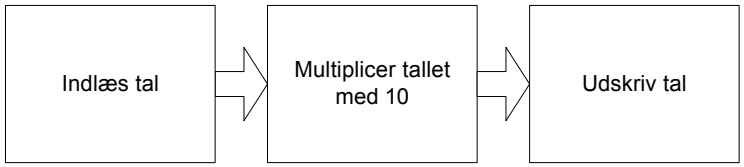
Kapitel 1

En simpel computer og hvordan den virker

Vi vil i det følgende kapitel gennemgå en simpel computermodel, der kun består af de mest basale komponenter. Desuden vil vi gennemgå, hvordan de enkelte komponenter virker.

1.1 Introduktion

Moderne computere, og for den sags skyld også umoderne, fungerer ved, at nogle elektroniske kredsløb afvikler en serie instruktioner i en bestemt rækkefølge. Ideen om en maskine, der afvikler en serie instruktioner, stammer tilbage fra 1834, hvor Charles Babbage lavede en såkaldt analytisk maskine. I 1834 var der ikke noget, der hed elektroniske kredsløb, og maskinen var derfor rent mekanisk.



Figur 1.1: En serie af instruktioner

Det var stort set umuligt at lave mekanisk, men i 1940'erne genoptog man ideen. Da havde man udviklet elektroniske kredsløb. En af de første egentlige computere, blev lavet af von Neumann i USA. Arkitekturen bag maskinen er den der stadig bruges, og den er siden blevet kaldt for von Neumann

arkitektur[1].

1.2 En simpel model

Et computerprogram består altså af en række instruktioner, der udføres i en bestemt rækkefølge. Disse instruktioner arbejder med nogle data i form af tal. For at opbygge en computer må vi have et komponent der kan forstå og udføre instruktioner. Vi skal også bruge en del, der kan indeholde instruktioner og data. Denne del skal blot indeholde tal, da både instruktioner og data er tal.

En basal **von Neumann** maskine består af følgende:

- Et lager til opbevarering af instruktioner og data.
- En kontrolenhed til at hente instruktioner og data fra lageret.
- En regneenhed til at udføre alle regneoperationer.
- Indlæsning- og udskrivningsdel, der kan skrive på skærmen eller registrere et tastetryk.

Kontrolenheden og regneenheden er næsten altid bygget sammen og kaldes en processor eller en CPU (**C**entral **P**rocessing **U**nit). Det er processoren, der afvikler instruktioner.

Vi mangler en del, der kan sørge for, at få data og instruktioner frem og tilbage mellem processoren, lageret og indlæsnings/udskrivnings-delen. Denne del kalder man en bus.

1.3 Komponenterne

1.3.1 Lager

Lageret er, hvor maskinen opbevarer data. Man kan forestille sig lageret som en masse små kasser, der hver kan indeholde et tal. Man kalder disse ”kasser” for celler. Vores maskine består af 100 celler, nummereret fra 0 til 99. En celles nummer kaldes også for dens adresse. Cellerne kan indeholde heltal mellem -9999 og 9999. Rigtige maskiner indeholder flere millioner celler, som kan indeholde tal helt op til 4 mia.

[1] Andre arbejdede med samme ide, men der var ikke mange der kendte til deres arbejde på den tid

1.3.2 Bus

En bus sørger for, at alle komponenterne i maskinen kan snakke sammen. Al data, der skal fra et komponent til et andet, skal over bussen.

1.3.3 Processor

Processoren er ”hjernen” i computeren, idet det er processoren der styrer afviklingen af programmer. Processoren indeholder to dele: en styreenhed og en regneenhed.

Regneenheden

Regneenheden, kaldes også en ALU (**A**rithmetic and **L**ogic **U**nit), udfører alle regneoperationer og gemmer resultatet i register A (A for akkumulator). Et register er det samme som en celle, men kaldes altså et register, når det ikke drejer sig om lageret. Et register kan da også ligesom hver af lagercellerne indeholde heltal fra -9999 til 9999. ALU’en kan i vores tilfælde gange, dividere, addere og subtrahere heltal. Normalt vil en ALU også kunne håndtere komma tal, men det er langt mere kompliceret og vil ikke blive behandlet yderligere.

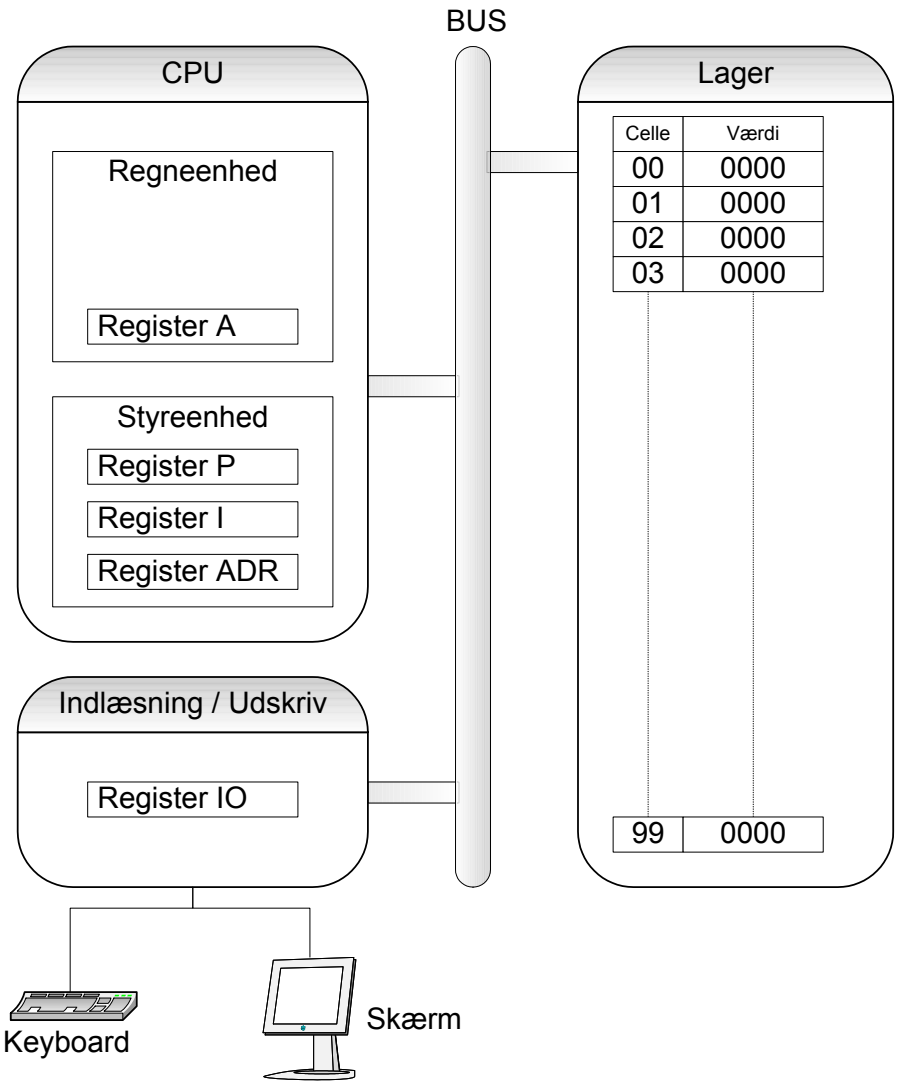
Styreenhed

Styreenheden sørger for at holde styr på, hvor i afviklingen programmet man er nået. Styreenheden har et P register (P for peger), der ”peger” på den adresse hvor den næste instruktion er placeret. Desuden har den et I register (I for instruktion), der indeholder instruktionskoden samt et ADR register (ADR for adresse), der indeholder den adresse, instruktionen skal behandle.

Indlæsning og udskrivning

Indlæsning og udskrivning er computerens eneste måde, at kommunikere til omverden på. Indlæsning til en computer, kan foregå via et tastatur, en mus, tale eller et lagermedie som f.eks. en CD. Udskrivning kan foregå til en skærm, et stykke papir eller en række lamper som på en lystavle. I vores maskine er den eneste mulighed for indlæsning og udlæsning **I/O** registret (**I**nput/**Output**). Når data flyttes til I/O registret, bliver det vist for brugeren. Hvis computeren beder om input fra brugeren, vil brugerens input det blive overført til I/O registret, hvorefter det kan viderebehandles.

Vi kan nu lave et diagram over computermodellen. Resultatet kan ses på figur 1.2 side 4.



Figur 1.2: Computermodellen

Kapitel 2

Programmering af computeren

Her vil vi gennemgå hvordan man programmerer maskinen vi lærte om i sidste kapitel.

2.1 Maskinniveauer

Man kan opdele en computer i flere niveauer; traditionelt opdeles de i følgende:

Niveau 6: Bruger programmer, som f.eks. tekstbehandling eller lign.

Niveau 5: Programmeringssprog

Niveau 4: Assemblersprog

Niveau 3: Operativsystem

Niveau 2: Maskinkode

Niveau 1: Mikroprogram

Niveau 0: Digitale kredsløb

Hvert niveau er bygget v.ha. af det foregående, således består maskinkode af mikroprogrammer, som igen bygger oven på den funktionalitet, som de digitale kredse har.

Hvis man ønsker at lave et program til vores simple computer, må man indtil videre, gøre det i maskinkode, dvs niveau 2.

2.2 Maskinkode

En computer forstår på maskinkode niveau kun meget få instruktioner. Det kan f.eks. være "`flyt data fra lagercelle 8 til register A`". Denne instruktion kalder vi for `HENT` efterfulgt af adressen, hvor der skal hentes fra. Instruktionen skal desuden ligge i lageret, hvor der jo kun er plads til 4 cifrede tal. Derfor oversætter vi instruktionen til et tal mellem 0 og 99, så den kan ligge i de to første cifre af lagercellen. De sidste to cifre bruger vi til den adresse, der skal hentes fra. En liste over alle instruktioner kan ses i bilaget.

Generelt vil alle instruktioner bortset fra `STOP`, der stopper afviklingen af programmet, skulle bruge en adresse i lageret, som de skal udføre operationen på.

Eksempel 2.1

For nemheds skyld bruger vi symbolske betegnelser for alle instruktionerne og først, når de skal indtastes i lageret, "oversætter" vi dem til tal. Et simpelt program, der adderer to tal, begge gemt i lageret på henholdsvis adresse 3 og 4, ser således ud i symbolsk maskinkode:

```
00:  HENT 03
01:  ADD  04
02:  STOP 00
03:  TOM  07
04:  TOM  05
```

I lageret vil det se sådan ud:

Adresse	Indhold
00:	0203
01:	0404
02:	0100
03:	0007
04:	0005

Programmet henter indholdet af cellen med adresse 03 (her er indholdet 7) til register A. Derefter adderes indholdet af cellen med adresse 04 (her er indholdet 5) til A registret, og resultatet gemmes i A registret (som så er lig 5 + 7 = 12) og overskriver altså den gamle værdi.

Da lageret indeholder både instruktioner og data, og da computeren ikke kan skelne mellem de to, er det op til programmøren at skelne og undgå

fejltagelser. Derfor er det nødvendigt at indsætte en `STOP` kommando, når programmet ikke må køre længere.

2.2.1 Instruktioner

Da vores computer blot er en model, har den også et ret begrænset instruktionssæt, 13 instruktioner ialt. Moderne computere som f.eks. en fra Apple eller en af de utallige PC producenter har langt større instruktionssæt og mange flere registre. Alle de 13 instruktioner, som vores computer, forstår kan ses i oversigten i bilaget.

Instruktionerne i vores computer kan opdeles i 5 kategorier:

Kontrolinstruktioner

Vi har i maskinen kun en instruktion til at styre afviklingen af programmet; det er `STOP`, som vi allerede har omtalt.

Regneinstruktioner

Regneinstruktioner i vores maskine udgøres af `ADD`, `SUB`, `DIV` og `MULT`, som kan henholdsvis addere, subtrahere, dividere og multiplicere.

Flytteinstruktioner

De to flytteinstruktioner i maskinen er `HENT` og `GEM`. De henholdsvis henter og gemmer til og fra register A. Adressedelen af instruktionen angiver hvilke celle der skal læses/skrives til.

Hopinstruktioner

Det er muligt at hoppe til en ny adresse, dvs angive, at næste instruktion, der skal afvikles skal hentes fra den angivne adresse. Der findes tre forskellige hopinstruktioner; to betingede og en ubetinget. den ubetingede `HOP` sætter blot P registret til adressedelen. De to betingede `HNUL` og `HNEG` hopper kun, hvis A registret er lig 0 eller for `HNEG`, at A registret er negativt.

Indlæsnings- og udlæsningsinstruktioner

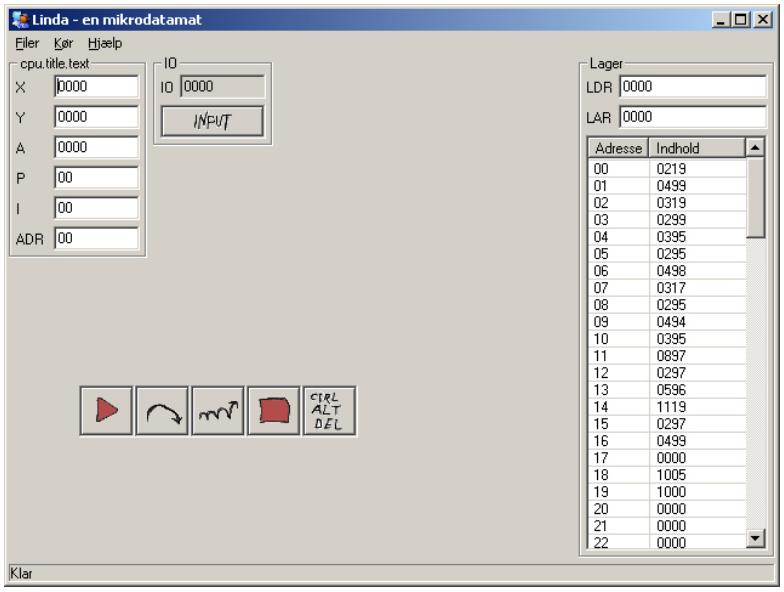
Der er en enkelt instruktion til indlæsning og en enkelt til udlæsning. `IND` venter på input fra brugeren og gemmer værdien på adressen angivet i adresse delen. `UD` skriver værdien på adressen i adressedelen til I/O registret.

2.3 Maskinkode i LINDA

Til dette hæfte hører også et program kaldet LINDA[1]. LINDA er et program til at simulere computermodellen i dette hæfte.

2.3.1 Indtastning af maskinkode i LINDA

Når programmet startes fremkommer følgende skærmbillede:



Figur 2.1: LINDA

Tabellen i højre side repræsenterer lageret. Det er muligt v.ha. dobbelt eller højre-klik at ændre indholdet af den enkelte celle. Når programmet er indtastet, er der mulighed for at afvikle programmet enten fra ende til anden eller en instruktion af gangen. Det er henholdsvis knappen længst til venstre og den næste i rækken. De er også tilgængelige fra ”Kør”-menuen. De to grupper af tekstbokse repræsenterer de registre forskellige registre i computeren.

Det er i LINDA muligt at gemme sit program, dette gøres ved at trykke gem i fil menuen. Åben henter et gemt program ind igen og overskriver hele lageret.

Tip Det er en god ide først at lave programmet på papir og vente med ind-

[1]Hvis ikke programmet fulgte med dette hæfte kan du hente det på http://www.dyregod.dk/linda

tastningen, til man ved hvor meget, det kommer til at fylde. Det er rigtig surt at have glemt i linie i starten af programmet.

2.3.2 Afvikling af programmer i LINDA

Ved afvikling af `IND` instruktionen vil det være muligt at indtaste en værdi i tekstboksen IO. Værdien indlæses først, når brugeren trykker på ”Input”-knappen.

Nulstil-knappen og -menupunktet sætter alle registre til nul, således kan maskinen afvikle programmet igen.

2.4 Øvelser

Øvelse 2.1

Indtast programmet fra eksempel 2.1, der adderer to tal, i LINDA og gå igennem programmet med enkelttrin.

Lav nu programmet så det istedet for at adderer to tal adderer tre tal. Resultatet skal gemmes på lageradresse 08.

Øvelse 2.2

Oversæt følgende program til maskinkode og indtast det i LINDA. Gå igennem programmet med enkelttrin.

Adresse	Indhold	
00:	IND	10
01:	IND	11
02:	IND	12
03:	HENT	10
04:	ADD	11
05:	ADD	12
06:	DIV	13
07:	GEM	9
08:	STOP	
09:	TOM	00
10:	TOM	00
11:	TOM	00
12:	TOM	00
13:	TOM	3

Hvad gør programmet?

Øvelse 2.3 - Svær

Lav et program der v.ha. `IND` instruktioner får brugeren til at indtaste et program. Når brugeren indtaster 9999 stopper indlæsningen og det indtastede program afvikles.

Kapitel 3

Mikrokode

Vi har i det foregående afsnit set på, hvordan man programmerer i maskinkode med LINDA. I det følgende vil vi bevæge os et niveau længere ned. Vi vil kigge på, hvordan man opbygger hver enkelt instruktion af mikrokode.

3.1 En udvidet computermodel

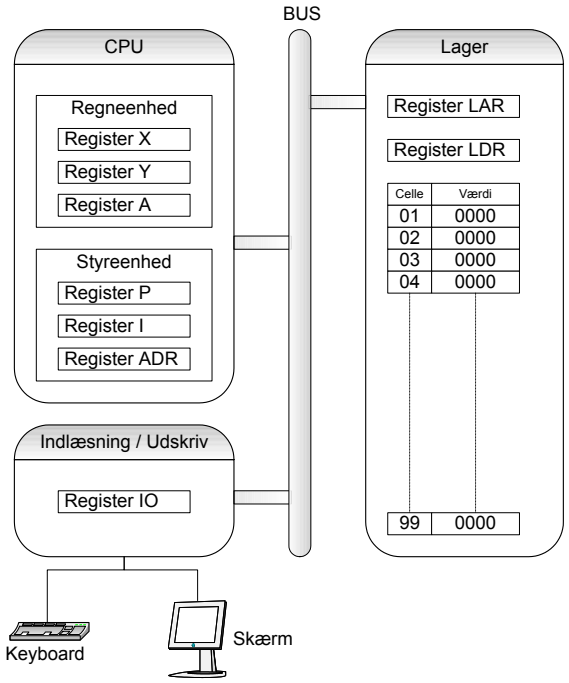
Men før vi beskæftiger os mere med mikrokode, bliver vi lige nødt til at udvide vores computermodel. Modellen, som blev introduceret i kapitel 2 skjuler nogle ting, som ikke var nødvendige at vide da vi lavede programmer på maskinniveau.

Vi vil nu se lidt nærmere på, hvad der egentlig sker, når man henter indholdet af en celle, og når man lægger to tal sammen i ALU'en.

I virkeligheden indholder lageret to registre: **LAR** og **LDR**, som står for henholdsvis **L**ager **A**dresse **R**egister og **L**ager **D**ata **R**egister. Desuden har man fra processoren ikke direkte adgang til lageret, og al kommunikation mellem processor og lager skal gå igennem bussen til de to registre LAR og LDR. Når man ønsker at hente indholdet af en celle via `HENT` instruktionen, sættes LAR til adressen på cellen. Derefter sendes besked om at man ønsker at lageret læser indholdet fra cellen, hvis adresse er angivet i LAR, ind i LDR. Derefter kan register A sættes lig register LDR. `GEM` instruktionen fungerer ligesådan. Blot bagvendt.

ALU'en indeholder ud over A registret to registre til, som kun bruges midlertidigt under regneoperationer, disse er register X og register Y. Ønsker vi f.eks. at udføre instruktionen `ADD`, starter maskinen med at overføre indholdet af register A til register X. Derefter følger den samme fremgangsmåde som ved `HENT`, blot med den forskel at indholdet af LDR lægges over i Y.

Derefter sørger elektroniske kredsløb i ALU'en for at lave de egentlige aritmetiske operationer. Og resultatet placeres i register A. Princippet er det samme for alle de aritmetiske operationer.



Figur 3.1: Den udvidede computermodel

3.2 Introduktion til mikrokode

Som nævnt i sidste kapitel så består hver instruktion af en række mikroinstruktioner. Computerens mikroinstruktioner er endnu mere begrænsede end de almindelige instruktioner, og der går flere mikroinstruktioner til at opbygge en enkelt instruktion.

Basalt set kan vores computer kun meget få ting. Den kan regne i ALU'en. Den kan flytte data mellem registre og lageret og eksterne enheder. Mikroinstruktionerne har ingen anden funktion end det komponenterne i computeren stiller til rådighed. Vi har derfor følgende instruktioner til rådighed:

Læs lager - Læser adressen i LAR og lægger indholdet i LDR

Skriv lager - Skriver indholdet af LDR på adressen i LAR

Send til bus - Sender indholdet af et register til bussen

Hent fra bus - Flytter indholdet af bussen til et register

Vent på input - Venter til IO registret er blevet sat af brugeren

Tæl op - lægger 1 til p registret

ALU Addition - adderer tallene i X og Y registrene sammen og lægger resultatet i A registret

ALU Subtraktion - subtraherer Y fra X og lægger resultatet i A

ALU Division - Dividerer X med Y og lægger resultatet i A

ALU Multiplikation - Multiplicerer X med Y og lægger resultatet i A

Det er disse mikroinstruktioner, alt i computermodellen skal opbygges af. Hvis vi bruger eksemplet fra tidligere, hvor vi ønsker at lægge to tal sammen:

HENT 03
ADD 04

Alle instruktioner har en del mikrokode til fælles. Koden til at hente en instruktion fra lageret er den samme hver gang. For at hente en instruktion fra lageret v.ha. mikrokode må man gøre nogenlunde følgende:

Overfør adressen i **P** til **LAR**, dvs adressen på den næste instruktion, der skal udføres lægges i **LAR** → læs indholdet af lagercellen med adressen i **LAR** ind i **LDR** → send indholdet af **LDR** til **I** → tæl **P** et op.

I mikrokode ser det derfor således ud:

Send indholdet af **P** registret til bussen
Modtag bus-indhold til register **LAR**
Læs lager adressen i **LAR** og læg resultatet i **LDR**
Send indholdet af **LDR** registret til bussen
Modtag bus-indhold til register **I** [1]
Tæl **P** en op

[1] Når I registret modtager data bliver det automatisk splittet op i to: en del til ADR registret og en til I

Samlet set ser de to liniers maskinkode således ud:

Send indholdet af **P** registret til bussen
Modtag bus-indhold til register **LAR**
Læs lager adressen i **LAR** og læg resultatet i **LDR**
Send indholdet af **LDR** registret til bussen
Modtag bus-indhold til register **I**
Tæl **P** en op

Send indholdet af **ADR** registret til bussen
Modtag bus-indhold til register **LAR**
Læs lager adressen i **LAR** og læg resultatet i **LDR**
Send indholdet af **LDR** registret til bussen
Modtag bus-indhold til register **A**

Send indholdet af **P** registret til bussen
Modtag bus-indhold til register **LAR**
Læs lager adressen i **LAR** og læg resultatet i **LDR**
Send indholdet af **LDR** registret til bussen
Modtag bus-indhold til register **I**
Tæl **P** en op

Send indholdet af **A** registret til bussen
Modtag bus-indhold til register **X**
Send indholdet af **ADR** registret til bussen
Modtag bus-indhold til register **LAR**
Læs lager adressen i **LAR** og læg resultatet i **LDR**
Send indholdet af **LDR** registret til bussen
Modtag bus-indhold til register **Y**
ALU Addition

Register A indeholder nu indholdet af adresse 02 og 03 adderet.

3.3 Mikrokode i LINDA

Det er muligt at arbejde med mikrokode på to måder i LINDA. Man kan udføre mikrotrin, det svarer til enkelttrin, blot tages der kun en mikroinstruktion af gangen. Statusbaren nederst i LINDA vinduet viser hvilke mikroinstruktioner maskinen udfører. Derudover er det muligt at ændre i implementeringen af maskinkodeinstruktionerne.

For at ændre i implementeringen skal man blot åbne Maskine.java filen i dk/dyregod/linda kataloget. Nederst i filen findes alle instruktionerne maskinen forstår med en metode pr. instruktion. F.eks. ser `HENT` instruktionen således ud:

```
public void HENT()
{
    sendTilBus(ADR);
    modtagFraBus(LAR);
    laesLager();
    sendTilBus(LDR);
    modtagFraBus(A);
}
```

De tilgængelige mikroinstruktioner er helt analogt med de tidligere definerede:

```
sendTilBus(int register)
modtagFraBus(int register)
laesLager()
skrivLager()
ventForInput()
ALUAdd()
ALUSub()
ALUDiv()
ALUMult()
```

sendTilBus og modtagFraBus tager som argument en int mellem 0 og 7 der angiver hvilket register, der skal arbejdes med. Alle registernumrene har dog også en konstant, som er navnet på registret med store bogstaver. Så hvis man f.eks. vil sende indholdet af register ADR til bussen ville kommandoen skulle se sådan ud (Husk semikolon efter hver linie i java):

```
sendTilBus(ADR);
```

Når man har ændret i Maskine.java filen skal LINDA genkompileres, dette gøres ved at køre compile.bat filen. Herefter kan LINDA programmet startes igen, og nu bruges koden med ændringer.

3.4 Øvelser

Øvelse 3.1

Hvordan vil følgende program se ud i mikrokode? Lav øvelsen uden at bruge LINDA.

Adresse	Indhold	
0:	HENT	06
1:	ADD	07
2:	DIV	08
3:	GEM	05
4:	STOP	
5:	TOM	00
6:	TOM	24
7:	TOM	23
8:	TOM	02

Øvelse 3.2

Åben Maskine.java filen i dk/dyregod/linda/ kataloget og implementer DIV instruktionen. Du finder metoden nederst i filen. Afprøv om instruktionen virker i LINDA ved at køre på compile.bat filen. [2]

Øvelse 3.3

Udvid LINDA maskinens instruktionssæt med endnu en instruktion `KVA` (**Kva**drer) der sætter indholdet af cellen i anden. Instruktionen skal have nummer 13.
HINT: Der skal tilføjes endnu en case sætning til executeNext() metoden, samt laves en ny metode af valgfrit navn til instruktionen.

[2] Hvis noget går galt kan du finde den originale Maskine.java fil i source kataloget

Indeks

ALU, 3
Assembler, 5

Bus, 3

CPU, 3

Instruktion, 7, 18
    Flytte-, 7
    Hop-, 7
    Indlæsnings-, 7
    Kontrol-, 7
    Regne-, 7
    Udlæsnings-, 7

Lager, 2
LINDA, 8, 11, 14, 18

Maskinkode, 5, 8
Mikroinstruktion, 12
Mikrokode, 5, 12

Processor, 3

Styreenhed, 3

von Neumann, 1

Bilag A

Instruktionssæt

En liste over alle de instruktioner LINDA forstår.

Nummer	Navn	Beskrivelse
0	TOM	ingen
1	STOP	stopper afviklingen af programmet
2	HENT	henter indholdet af celle til register A
3	GEM	gemmer indholdet af register A til celle
4	ADD	adderer indhold af celle til A
5	SUB	subtraherer indholdet af celle fra A
6	MULT	multiplicerer register A med indholdet af celle
7	DIV	dividerer register A med indholdet af celle
8	IND	indlæser tal til celle
9	UD	udskriver tal fra celle
10	HOP	hopper til adresse (register p sættes til adresse)
11	HNUL	hopper til adresse hvis A er 0
12	HNEG	hopper til adresse hvis A er negativ