Lærevejledning

- en introduktion til maskinarkitektur

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

Indhold

Kapitel 1

Bekendtgørelsen og formelle krav

Undervisningsmaterialet dækker et område svarende til det af gymnasiebekendtgørelsen påkrævede for undervisning i maskinarkitektur for mellemniveau (C-niveau) datalogi. Ifølge gymnasiebekendtgørelsen inkludere det:

- virtuelle maskinniveauer og systemprogrammel
- dualitet mellem program og data.

Ud over det følger undervisningsforløbet også kravene for undervisningsmål. Netop for at holde sig til kravene for undervisningsmål, er det vigtigt at huske at eleven selv skal være i stand til at løse opgaverne på egen hånd, via. datalogiske metoder ved undervisningens afslutning.

Kapitel 2

Undervisningsforslag

Det må forventes at eleven særligt i starten af undervisningsforløbet vil befinde sig på det pre/konkret-operationelle stadie(Piaget), samt 2-3 niveau(Bloom). Derfor er det vigtigt at lade eleverne anvende deres nye viden. Timerne skal derfor bære mere præg af at være eksperimenterende, end belærende.

En måde at opnå dette er at lade eleverne sidde i grupper af 2-3 ved hver computer, og lade dem løse opgaver i fællesskab. Det er vigtigt at eleverne i de enkelte grupper ikke ligger langt fra hinanden niveau-mæssigt[1].

2.1 Gruppearbejde

Gruppearbejde kan være et særdeles effektivt middel til undervisning, dog er det til dette forløb kun nødvendigt at arbejde i små grupper, eller samarbejde med hele klassen. Det foreslås at eleverne sidder 2-3 personer pr. computer, således at de sammen kan arbejde sig frem til en løsning på problemet.

2.2 Forelæsninger

Det er vigtigt at huske at selvom forelæsninger er en meget let måde at formidle viden på , så er det stadigvæk den mest passiverende form for undervisning for eleverne. Sørg derfor for at holde elevernes opmærksomhed fanget, under hele forelæsningen. Eleverne lære ikke noget hvis de ikke finder det interessant, så derfor er det vigtigt at gøre forelæsningerne interessante. En ide kunne f.eks. være at starte forløbet med at få eleverne til at komme med deres bud på hvilke grundkomponenter en computer er

[1] Se evt ”bubblesort” for et eksempel på hvordan man kan sortere eleverne efter niveau

bygget op af. På den måde engagere man eleverne fra starten, og de har lige pludselig en interesse i at følge med i timen, frem for at falde i søvn.

2.3 Opgaver

De medfølgende opgaver er beregnet på at eleverne skal forsøge at løse dem i grupper af 2-3 personer. Nogle af opgaverne er så svære at de bør gennemgås på tavle og andre i fællesskab i klassen.

2.4 Lege

Lege, og anden fysisk aktivitet kan været et godt middel til at aktivere eleverne. Det kan ligefrem være en fordel at klargøre ting for eleverne på denne måde frem for en teoribog.

2.4.1 Bubblesort

Denne leg kan bruges til at opdele elever i små grupper.

Lad eleverne stille sig op på en række, derefter skal de afgøre om de har højere eller lavere niveau end deres sidekammarat, hvis de har lavere skal de skifte plads imod venstre, ellers imod højre. Tilsidst kan man så ”knække” rækken på midten, og lade de to rækker stille sig ved siden af hinanden. Man har nu et antal par, hvor der er en klar niveauforskel, dog uden at den bedste kommer sammen med den dårligste.

2.4.2 En elev-datamat

Eleverne skal i denne leg agere komponenter i en computer. På forhånd laves der en række små kort med beskrivelse af hvordan komponentet fungere, samt en række simple programmer. Eleverne skal nu trække et beskrivelses-kort hver. Beskrivelser kunne f.eks. være:

- Bus: Du kan modtage instruktioner eller data, og videregive den til en anden komponent. Du kan ikke ændre på noget.
- Alu: Du kan udføre aritmetriske operationer (addition, subtraktion, multiplikation og division). Du kan udføre disse operationer på indholdet af register x og y, og lægge resultatet i a. Du kan kun tilgå disse registre via bussen.
- Lager: Du kan modtage eller sende indholdet af specifikke celler. Du må ikke selv bestemme hvilke celler der tilgås, men må kun tilgå de celler du får besked på fra bussen.

Gruppen får nu udleveret et simpelt program, og får til opgave at afvikle det korrekt. Hvert medlem må kun udføre sin egen opgave, dog kan det være en ide at lade hele gruppen fungere som styreenhed.

Legens formål er at give eleverne en ide om hvordan komponenterne i en computer arbejder sammen, og kan forhåbentligt være med at give eleven en bedre forståelse.

Kapitel 3

Opsætning af ”LINDA”

3.1 Maskinkrav

LINDA er testet på Windows og Linux (testet på Intel ia32, men Intel ia64 og PPC platforme burde også være ok). Maskinkravene er ens for begge operativsystemer.

- Mininum hvad der svarer til en 166 MHz Pentium
- Ca. 26 Mb diskplads til LINDA incl source + 50 Mb til Java Development Kit
- Anbefalet minimum 64 Mb RAM, men 32 Mb kan gøre det.

3.2 Installation

LINDA kræver ingen installation. Alt hvad der er nødvendigt for at køre programmet ligger på cd’en og programmet kan også køres direkte fra cd’en. Dog er det nødvendigt at kopiere hele linda kataloget over på en eller anden form for medie med skriveadgang hvis man ønsker at rode i koden selv. Desuden er det nødvendigt at have et Java Development Kit installeret hvis man ønsker at compile source koden. På LINDA cd’en ligger Java Development Kit 1.3.1 til Windows fra Sun.

Hvis hovedformålet med at bruge LINDA er at eleverne skal kunne se hvordan det er implementeret anbefaler vi at hele projektet bliver importeret til et IDE eller at eleverne har adgang til en god editor. Der er på LINDA cd’en lagt to IDE’er; Eclipse fra IBM og Netbeans fra SUN. Begge IDE’er er open-source og derfor gratis.

For at kunne arbejde med sourcekoden eller afvikle programmet uden brug

af exe filen, skal man være opmærksom på at programmet er afhængig af følgende filer: swt.jar skal være i classpath både ved kørsel og kompilering. Swt-win32-2019.dll skal placeres i Java Runtime Enviroments /bin katalog.

3.3 Problemer?

Hvis LINDA ikke starter kan man prøve at starte programmet med ”linda.exe -debug”. Programmet skulle da gerne skrive nogle informationer på skærmen.

Hvis ikke det er til nogen hjælp kan man sende en email til Ulf Holm Nielsen på doktor@dyregod.dk.

Kapitel 4

Opgaveløsninger

Øvelse 2.1

Det revidere program bør se således ud:

Adresse	Indhold
00:	0205
01:	0406
02:	0407
03:	0308
04:	0100
05:	0023
06:	0016
07:	0065
07:	0000

Øvelse 2.2

Programmet indlæser tre tal fra brugeren og tager gennemsnittet.

Øvelse 2.3

Opgaven er klart sværere end de andre opgaver i hæftet. Men ikke umulig.

En løsning kan se således ud:

Programmet indlæser et andet program via IND instruktionen og gemmer det i lageret. Det indlæste program afvikles når brugeren indtaster 9999. Brugeren skal selv sørge for at placere en stop kommando!

Programmet kan ikke indlæse programmer der har brug for data fra lageret fra første instruktion. Det indlæste program bør gøre brug af IND instruktio-

nen til at få data programmet skal afvikles med.

F.eks. vil følgende program ikke egne sig:

```
HENT   04
ADD 05
GEM 06
```

Det vil derimod følgende:

```
IND 05
IND 06
HENT   05
ADD 06
GEM 07
```

Programmet til indlæsning kan ses på næste side.

Man angiver i celle 99 hvor det indlæste program skal indlæses til (start adresse).Celle 98 er en tom GEM instruktion som vi skal bruge senere. Celle 97 bruges til at opbevare de indlæste data. Celle 96 indeholder brugerens stop kode: her 9999. Celle 95 bruges som peger, dvs den tælles op efterhånden som fylder lageret op ved indlæsning. Celle 94 indholder tallet 1 da vi skal tælle 95 op en ad gangen.

Adresse	Indhold	Kommentar
00:	HENT 19	Henter indholdet af adresse 19 som er instruktionen der sørger for at hoppe til det indlæste program
01:	ADD 99	Lægger adresse hvorpå brugerprogrammet til. Derved kan vi hoppe til den rigtige adresse.
02:	GEM 19	Gemmer den nye instruktion
03:	HENT 99	Henter start adressen for brugerens program
04:	GEM 95	Gemmer i 95 så vi kan tælle op
05:	HENT 95	Henter 95 igen da det er her vi hopper tilbage til.
06:	ADD 98	Lægger 98 til (gem instruktionen) således at vi kan gemme det indlæste på den adresse 95 peger på
07:	GEM 17	Gemmer den nye gem instruktion
08:	HENT 95	Henter adressen i 95
09:	ADD 94	Lægger 1 til
10:	GEM 95	Gemmer i 95
11:	IND 97	Indlæser linie af brugers program
12:	HENT 97	Henter 97
13:	SUB 96	Trækker 9999 fra
14:	HNUL 19	Hvis resultatet er 0 betyder det at brugeren indtastede 9999 og han ønsker at afslutte og vi hopper til 19
15:	HENT 97	Ellers henter vi 97
16:	ADD 99	Lægger 99 dvs start offset i lageret til således at brugerens adressering passer
17:	GEM 00	Data gemmes til adressen specificeret tidligere.
18:	HOP 05	Løkken gentages
19:	HOP 00	Hopper til brugerens program
20:	TOM 00	
—		Fortsætter med tomme celler indtil..
94:	TOM 01	
95:	TOM 00	
96:	9999	
97:	TOM 00	
98:	GEM 00	Gem ”skabelon”
99:	TOM 21	Offset værdi

Øvelse 3.1

Se Maskine.java filen i source/dk/dyregod/linda kataloget.

Øvelse 3.2

```
public void DIV()
{
    sendTilBus(A);
    modtagFraBus(X);
    sendTilBus(ADR);
    modtagFraBus(LAR);
    laesLager();
    sendTilBus(LDR);
    modtagFraBus(Y);
    ALUDiv();
}
```

Øvelse 3.3

Der er mange mulige måde at lave en sådan metode på. En kunne se ud som herunder. Men eleverne er ganske givet opfindsomme nok til at deres løsning ikke ser sådan ud.

```
public void KVA()
{
    sendTilBus(ADR);
    modtagFraBus(LAR);
    laesLager();
    sendTilBus(LDR);
    modtagFraBus(A);
    sendTilBus(A);
    modtagFraBus(X);
    sendTilBus(A);
    modtagFraBus(Y);
    ALUMult();
}
```

Og ændringen til executeNext():

```
case 13 :
    KVA();
    break;
```