IT för personligt arbete F6

Datalogi del 2

DSV Peter Mozelius

Datarepresentation

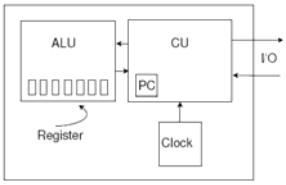
- Det som lagras i en dator representeras i grunden som 1:or och 0:or
- Dessa **binära värden** kan sedan tolkas på olika sätt i olika sammanhang
- Värdena **1** och **0** kan i sin tur ses som symboler för elektriska laddningar eller tillstånd i motsatsförhållanden

Datarepresentation

- Ström **ELLER Inte ström**
- Ljus **ELLER** Inte ljus
- Positiv laddning **ELLER** Negativ laddning
- Laddad kondensator **ELLER** Inte laddad
- Magnetisk laddning **ELLER** Icke-laddning
- Fördjupning **ELLER** Upphöjning
- Svart **ELLER** Vitt

Repetition från F5

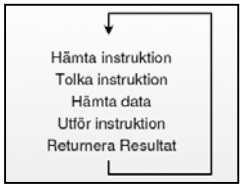
Processorns beståndsdelar



- **Styrenhet**
- **Aritmetisk logisk enhet**
- **Processorregister**
- **Systemklocka**

Repetition från F5

- The Fetch/Execute-cycle
  - Ett cykliskt arbetssätt
  - Fem steg som itereras



Repetition från F5

- De enstaka instruktionerna kombineras till att lösa komplexa problem
- Iteration
  - En eller flera instruktioner upprepas
- Selektion
  - Exekveringen väljer olika vägar

Att bygga program

- Program byggs genom att instruktioner kombineras till algoritmer
- Algoritmer implementeras i olika programmeringsspråk
- Lågnivåspråk
  - Assembler
- Högnivåspråk
  - Java, C, C++

Lågnivåspråk

- Med begreppet lågnivåspråk åsyftas olika former av **assembler**
- En **assemblerinstruktion** motsvarar ofta en **processorinstruktion**
- Assemblerinstruktionerna fungerar som **mnemonics** för maskinkoden
- **Mnemoteknik**
  - Teknik som underlättar hågkomst

Assembler

- Enkelt att ge exakta direktiv för datorns hårdvara
- Instruktioner för adresser i t ex
  - Grafikminnet
  - Portar
- Praktiskt för drivrutiner och annat som har direktkontakt med hårdvara
- Svåröverskådlig kod i större program

Högnivåspråk

- En instruktion i ett högnivåspråk motsvarar ofta 10 – 20 assemblerrader
- Lättare att strukturera koden
- Inbyggda säkerhetsmekanismer
- Inbyggda tekniker för modularisering
  - klasser
  - paket
  - funktioner/metoder/subrutiner/procedurer

Olika sorters programmering

- Imperativ programmering
  - Fortran, C, BASIC, Pascal m fl
- Funktionell programmering
  - LISP, Scheme
- Logikprogrammering
  - Prolog
- Objektorienterad programmering
  - C++, Eiffel, Simula, Java m fl

PAUS 15 min

Kompilering och interpretering

- Ett högnivåspråk kan kompileras (C)
  - Koden översätts till processorinstruktioner
- Ett högnivåspråk kan interpreteras (Javascript)
  - Koden tolkas vid körning
- Ett högnivåspråk görs om till bytekod (Java)
  - Bytekoden är plattformsoberoende

En addition

- Lågnivå – Assembler
  - **ADD 2000, 2080, 4000**
- Högnivå – Java, C eller C++
  - `int a, b, c;`
  - `a = 5;`
  - `b = 7;`
  - `c = a + b;`

En liten jämförelse i Java

```
int a;
int b;
a = 5;
b = 5;
if (a == b) {
  a = a + 1;
}
System.out.println("a = " + a );
```


En jämförelse Java - Javascript

- Java är ett objektorienterat språk
- Javascript är objektbaserat
- Java kompileras till bytekod
- Javascript interpreteras
- Java är ett starkt typat språk
- Javascript är inte starkt typat

Syntaxen är mycket lik i de båda språken

Javascript

- Användbart för internetprogrammering
- Klientsideprogrammering
  - Mindre beräkningar
  - Datavalidering
  - Internetprogrammering
- Jag har två exempel utlagda på:
  http://www.dsv.su.se/~mozelius/GKITP/javascript/javascript.htm

  http://www.dsv.su.se/~mozelius/schack/schack.htm (Rank 2200)

Javascript – booleska villkor

- Vi tittar närmare på ett villkor :

```
var surfare = "Mona";
var lasar = "2005";

if((surfare =="Lisa" || surfare =="Bernt" ||
    surfare =="Mona") && (lasar == "2006"))

  document.write("Hej ",surfare, ", Gott Nytt" ,lasar );

else

  document.write("Hej, det är fortfarande ", lasar );
```


Javascript

Från XHTML-dokumentet länkas javascriptet in enligt:

```
<script type="text/javascript" src="exempel.js">
</script>
```

MVC = Model – View - Controller

HTML – CSS - Javascript

Kort Paus?

Algoritmer del 2

- En algoritm är beskrivningen för hur man löser ett givet problem
- Behöver inte vara skriven i programkod
- Som ett recept i en kokbok
- Den datalogiska definitionen:

”Ett ändligt antal instruktioner som löser ett problem.”

Algoritmer del 2

- Algoritmer har funnits sedan länge inom områden som t ex arkitektur och matematik
- Här kommer en algoritm som är daterad till 1200-talet men som säkert är ännu äldre
- Leonardo Fibonaccis **Fibonaccital**:

```
public class Fibonacci {
   static final int ANTAL = 10;
```


Algoritmer del 2

```
public static void main(String[] args){
  int low   = 1;
  int high  = 1;
  for(int i = 1; i <= ANTAL; i++){
     System.out.println(high);
         high += low;
         low = high - low;
  }
}
```

Logiska operatorer

- Logiskt OCH: `&&`
- `(A && B)` är sant om både A **OCH** B är sanna

A	B	(A && B)
1	1	1
1	0	0
0	1	0
0	0	0

Logiska operatorer

- Logiskt ELLER: `||`
- `(A || B)` är sant om A **ELLER** B är sann

A	B	(A || B)
1	1	1
1	0	1
0	1	1
0	0	0

Iterativ programutveckling

- Att bygga programmen bit för bit
- Lägg till en rad kod i taget och kontrollera att allt fungerar innan något mera läggs till
- Enklare felsökning, bättre kontroll
- Att gå igenom programmen en gång till när all kod fungerar och optimera

Iterativ programutveckling

För att verkligen nå himlen,
så bygger vi en våning i taget

Felsökning

- Även vid mycket noggrann utveckling så uppstår alltid felaktigheter och buggar
  - Testa mjukvaran med varierande indata
  - Mata in udda värden/randvärden
  - Kontrollera resultat med spårutskrifter
  - Isolera problemen
  - Rätta felaktigheterna

Operativsystem

- Alla av er har hört begreppet operativsystem!
- Men vad är det egentligen?

- Vi tar nu 20 minuters paus och ni sätter er i små grupper och resonerar om några saker som ett operativsystem ska klara.

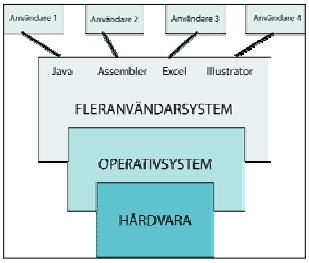
Att starta upp datorn

- **BIOS**
  1) När strömmen slås på startas en inläsning från det batteriuppbackade BIOS
  2) BIOS kör sedan den kontroll av hårdvaran som brukar kallas **POST**
  3) Om allt är som det ska med hårdvaran så finns det en liten kodsnutt som sedan lämnar över till operativsystemet

Att starta upp datorn

- **Operativsystemet**
  1) Ser till så att olika enheter i datorn kan samarbeta genom att ladda drivrutiner
  2) Startar upp de processer som sköter om användarinteraktionen
  3) Gör sig klart för att köra program och för adressering av primär- och sekundärminne

Operativsystem



Operativsystemet som ett gränssnitt mellan hårdvara och applikationer

Några viktiga OS-funktioner

- Minneshantering
- Administrera ett filsystem
- Kontakt med nätverk
- Kontakt med användare
- Kontakt med yttre enheter
- Köra program
- Köra flera program samtidigt

Köra flera program samtidigt

- **Single tasking** operating systems
  - att köra en process i taget
- **Multitasking** operating systems
  - multikörning, flera processer parallellt
- **Multiuser operating systems**
  - Fleranvändarsystem som naturligtvis måste klara multikörning

Fleranvändarsystem

- Som här i IT-Forums datasalar
- Personliga användarkonton
- Personlig login
- Personlig miljö
- Personlig åtkomstkontroll av program
- Personlig åtkomstkontroll av data
- Gemensamma fora som t ex First Class

Olika typer av RAM-minne

- SRAM = Statiska RAM-minnen
  - Var förr i tiden klart snabbare än DRAM
  - Men är också dyrare att tillverka
- DRAM = Dynamiska RAM-minnen
  - Billigare och tar mindre plats
  - Minnescellerna behöver uppdateras
- Läs mera på:

http://www.jonasweb.nu/sidor/datorn/tekniskt/minnen.html

Olika typer av SDRAM-minne

- **SDRAM - Synchronous DRAM**
  - Synkroniserat med datorns arbetspuls
  - Arbetar snabbare än vanliga DRAM
- **DDR SDRAM - Double Data Rate**
  - Dataöverföring 2 ggr per klockcykel
  - Mer strömsnåla än vanliga SDRAM
  - DDR SDRAM2 – skickar 64 bitar 2 ggr/cykel

http://www.webopedia.com/quick_ref/dram_memory.asp

Tentatips

- Läs på om booleska villkor
  - Kursboken sid 525 – 536
- Tentaplugga i grupp
  - I Forums grupprum
  - Eller via Internet

Tack för idag!