IT för personligt arbete F5

Datalogi del 1

DSV Peter Mozelius

En dators beståndsdelar

1) Minne
2) Processor
3) Inmatningsenheter
   1) tangentbord
   2) scanner
   3) mus
4) Utmatningsenheter
   1) bildskärm
   2) ljudkort

Kommunikation utåt

- För att processorn ska kunna kommunicera med omvärlden, behöver den inmatnings- och utmatningsenheter (I/O-enheter)
- Datorn har I/O-portar där externa enheter kan anslutas
- För varje I/O-enhet finns det en *device driver/drivrutin* som tolkar datorns signaler till något som enheten kan förstå

Datorminne

- All data i datorn sparas som binära siffror i olika minnesfack med adresser
- Hur data sedan ska tolkas är upp till den som läser minnet
- En viss mängd data i varje fack och innehållet i varje fack kan skrivas över hur många gånger som helst

Datorns olika minnen

Olika typer av data lagras på olika platser

- Olika minnen är olika snabba
  - Externa lagringsenheter (t ex USB-minnen)
  - Hårddisk(ar) (sekundärminne)
  - RAM-minne (primärminne)
  - Processorns minnen och register

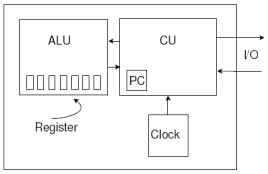
Processorn

- Processorn kan liknas vid datorns hjärna
- Enheten som fattar alla beslut, utför alla beräkningar samt ger de andra enheterna instruktioner
- Processorn utför många enkla beräkningar i mycket hög hastighet

Processorns beståndsdelar

1) **CU**, kontrollenhet som fattar besluten
2) **ALU**, som utför beräkningarna
3) **Processorregister** (scratchpad), små snabba minnen för mellanlagring
4) **Klocka**, som styr i vilken takt som instruktionerna utförs

Processorns beståndsdelar



- **Kontrollenhet**
- **Aritmetisk logisk enhet**
- **Processorregister**
- **Systemklocka**

CU/Kontrollenheten

- Kontrollenheten i processorn är den enhet som styr vad som ska hända
- Den har en minnesadress, **Program Counter** (PC), som talar om var den ska hitta nästa instruktion
- För varje steg den utför, hämtar den nästa instruktion där PC talar om att den ligger
- Kontrollenheten tolkar sedan instruktionen och hämtar de data den behöver
- Slutligen så utförs instruktionen och resultatet lagras på angiven minnesplats

Aritmetic Logic Unit

- **ALU** är processorns beräkningsenhet
- En ALU i en dator kan utföra ett begränsat antal enkla instruktioner:
  - jämföra två värden
  - öka med 1
  - minska med 1
  - logiskt OCH, logiskt ELLER
  - addera, subtrahera …

Processorregister

- All data som ALU arbetar med måste ligga i processorregistren
- Extremt snabba minnesplatser som finns internt i processorn
- För att ALU ska kunna utföra en beräkning så måste CU först se till att all data ligger i rätta register

Systemklockan

- Systemklockan bestämmer i vilken takt som instruktionerna exekveras
- Efter rasten ska vi titta på hur det går till i detalj när en instruktion exekveras
- Det som i kursboken kallas för

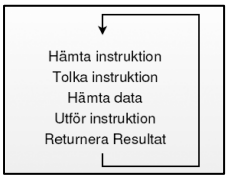
  The Fetch/Execute-cycle

  Processorns exekveringscykel

PAUS 15 minuter

The Fetch/Execute-cycle

- Ett cykliskt arbetssätt
- Fem steg som itereras



Ett instruktionsexempel

ADD 2000, 2080, 4000

- Hämta det värde som ligger på adress 2000 och det värde som ligger på adress 2080 och spara resultatet på adress 4000
- Notera att vi inte är intresserade av att addera talen 2000 och 2080
- Vi vill i stället addera de två tal som ligger på dessa två adresser

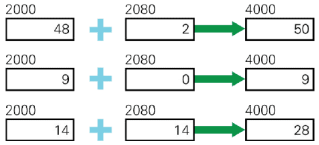
Ett instruktionsexempel

ADD 2000, 2080, 4000

- Genom att vi adderar talen som ligger på adresserna, kan samma instruktion ge olika resultat olika gånger
- I exemplet ovan adderar ALU de två tal som CU hämtat från adresserna 2000 och 2080
- Resultatet av additionen tas om hand av CU och läggs på adress 4000

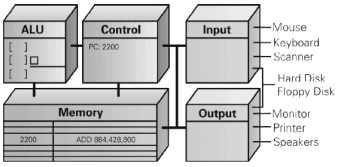
Ett instruktionsexempel

- Samma instruktion med varierande indata



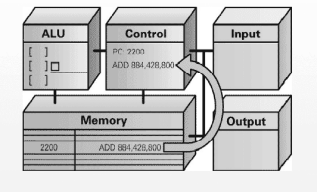
Instruktionsexempel 2

- **ADD 884,428,800**



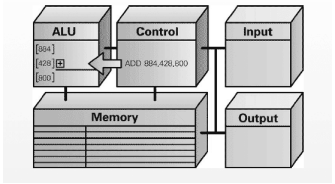
Instruktionsexempel 2

1) Från minnet till kontrollenheten



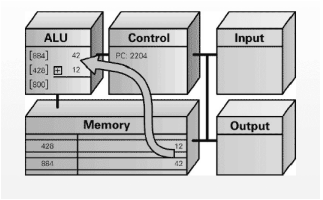
Instruktionsexempel 2

- Adresser till operander och lagring av resultatet läggs in i ALU



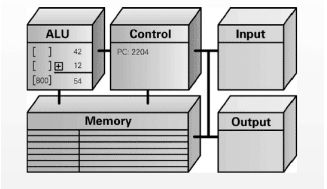
Instruktionsexempel 2

- Operanderna hämtas in från minnet



Instruktionsexempel 2

- Beräkningen utförs



Instruktionsexempel 2

- Resultatet lagras sedan i minnet och nästa cykel påbörjas sedan
- Till skillnad från en processor så behöver våra hjärnor lite vila ibland

PAUS 15 minuter

Hur snabbt arbetar processorn?

- Processorns hastighet bestäms av en klocka som med jämna mellanrum ger ifrån sig en elektrisk puls
- I teorin kan processorn utföra en sak i F/E-cykeln varje gång klockan tickar
- Om en instruktion har fem delar så tar varje instruktion fem klocktick
- Detta kan snabbas upp
  - *pipelining*
  - Superskaläritet
  - Multipla processorer/Dual core

Hur snabbt arbetar processorn?

- Klockans hastighet anges i Hertz (Hz)
- Hz = antal pulser per sekund
- Dagens mikrodatorer har en klockfrekvens på flera GigaHertz (GHz)
- Ett antal miljarder tick per sekund
- En dator arbetar inte snabbare än sina långsammaste komponenter tillåter

Hur arbetar processorn?

- Teoretiskt sett räcker det med sex olika sorters instruktioner för att en processor ska kunna lösa *alla* problem
- I praktiken är det dock opraktiskt att bara använda sig av sex olika instruktioner
- En modern processor kan utföra ett 100-tal olika instruktioner

Hur arbetar processorn?

- Exempel på olika instruktioner:
  - addera innehållet i två register
  - jämför innehållet i två register
  - kontrollera om värden är 0 eller inte
  - spara värdet i ett register till en viss minnesadress
  - hoppa till en ny instruktion (sätt PC till ett nytt värde) beroende på värdet av föregående operation

Hur arbetar processorn?

- Ofta ligger nästa instruktion på nästa plats i minnet

MEN

- För att kunna göra något intressant, och inte bara utföra alla instruktionerna i en enda följd, måste processorn kunna hoppa mellan olika instruktioner beroende på situationen
- Dessa hopp kallas för Branch och Jump. Det betyder att adressen på nästa instruktion som ska utföras kan vara ett resultat av den instruktion som just utförts

Att bygga program

- 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, Javascript, C, C++

Algoritmer

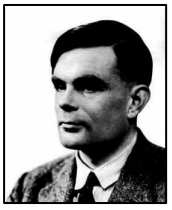
- En algoritm kan liknas vid
  - ett recept
  - en reparationsmanual
  - en vägbeskrivning
- Recept:
  - *Ta 1 dl socker, 4 msk kakao, och 5 dl mjölk*
  - *Blanda alla delar i en kastrull*
  - *Koka upp och låt sjuda i 5 minuter*

Algoritmer

- Den datalogiska definitionen:
  - *”Ett ändligt antal instruktioner som löser ett problem”*
- Definierat av Alan Turing
  - Ett av 1900-talets stora genier
  - Var bl a med och knäckte Enigma-kryptot

Alan Turing

Alan Turing
1912 - 1954



Alan Turing

- En av de forskare som lade den teoretiska grunden till datavetenskapen
- Uppfann flera grundläggande begrepp
- T ex **Turingtestet**

- Mera om algoritmer i nästa föreläsning
- Vi ska nu titta på ett litet Javascript

Javascript

Ett litet exempel som går att provköra från
http://www.dsv.su.se/~mozelius/GKITP/

```
var surfare = prompt("Vad heter du som besöker
                      denna sida?", "Anna");

var lasare = window.navigator.appName;


document.write(" <h3> Hej ",surfare, ", jag ser
           att du använder " ,lasare, ". </h3> ");
```


Javascriptet inlänkat i HTML

- Javascriptet läggs i en egen fil
- Om filen heter exempel.js
  så kan Javascriptet länkas in enligt:

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


Javascript

- Vi går vidare med programmering på nästa föreläsning
- Vi ses igen på torsdag!

Tack för idag!