DV1464/DV1493 Datorteknik

Introduktion

Erik Bergenholtz

Institutionen för Datalogi och Datorsystemteknik
Blekinge Tekniska Högskola

26 mars 2017

Innehåll

Innehåll

Vi som jobbar med kursen

- Kursansvarig och föreläsningar
  - Carina Nilsson
    carina.nilsson@bth.se
- Laborationer
  - Erik Bergenholtz
    erik.bergenholtz@bth.se

Kursmoment DV1464/DV1493

- Laborationer 4.5/4hp
  - Uppgift 1 0.5hp
  - Uppgift 2 2/1.5hp
  - Uppgift 3 2hp
- Tentamen 3/2hp
  - Tentamensbetyget blir kursbetyg

Laborationer

- Uppgift 1: Enkelt assemblerprogram för ARM
- Uppgift 2: Avbrottshantering för ARM
- Uppgift 3: Litet assemblerprojekt för Intel x86

Regler för laborationer

När du genomför laborationerna måste följande regler följas:

- En laborationsgrupp bör bestå av två personer. Efter överenskommelse med labbhandledaren kan undantagsfall med grupper på en eller tre personer accepteras.
- Det är **inte** tillåtet att dela med sig av laborationsresultat eller laborationsrapporter till andra grupper
- Det är inte tillåtet att ta, kopiera eller på annat sätt efterlikna en annan grupps resultat.
- Alla gruppmedlemmar måste aktivt delta i laborationernsarbetet. Detta inkluderar att skriva kod, rapporter, testa och felsöka, redovisa och experimentera.
- Examinationen är alltid baserad på individuella resultat

Tentamen

- Skriftlig tentamen, inga hjälpmedel
- Första tentamenstillfället är 27e maj, tid meddelas senare.

Litteratur

- Rekommenderad kursbok: D.A. Pattersson
  Computer Organization and Design - the Hardware/Software[1]
  Interface – ARM edition,
  Morgan Kaufmann.
  ISBN13: 978-0128017333
  ISBN10: 0128017333
- Alternativa böcker
  - Brorsson, M: Datorsystem: program- och maskinvara.
    Studentlitteratur AB
  - William Stallings: Computer Organization and Architecture,
    Prentice Hall International, Inc..
  - Sven Eklund, Avancerad datorarkitektur, Studentlitteratur, 1994.
  - Andrew S. Tanenbaum, Structured Computer Organization, 4th edition, Prentice Hall International, Inc., 1999.

[1] Boken använder ARMv8 i sina exempel, men vi kommer använda ARMv6. Referens till detta finns som .pdf på It's Learning

Upplägg

Föreläsningar

- Ge överblick, *underlätta studier i boken*
- Förberedelser: Läs gärna anvisade kaptiel i boken

Laborationer

- Syfte: ge praktisk kunskap och förståelse.
- Förberedelser: ha studerat laborationsmaterialet (laborationshandledning och beskrivning av systemet)

Kursmål

Kunskap och förståelse:

- klargörande kunna beskriva funktionen hos ett datorsystem och dess ingående delar
- klargörande kunna förklara samspelet mellan hårdvara, maskinspråk och högnivåspråk
- översiktligt kunna beskriva funktionen hos assembler och länkare
- översiktligt kunna redogöra för samspel mellan datorsystem och omvärlden

Kursmål, forts.

Färdighet och förmåga

- självständigt kunna konstruera en avbrottshanterare
- självständigt kunna skriva assemblerrutiner som kan ingå i ett högnivåspråksbibliotek
- självständigt programmera i assemblerspråk för minst två olika arkitekturer

Kursmål, forts.

Värderingsförmåga och förhållningssätt

- i generella termer kunna argumentera kring för och nackdelar med olika arkitekturer i datorsystem.

Förändringar i upplägg sedan förra året

- MIPS-datorerna i labbet har bytts ut mot Raspberry Pi's

OBS!

Rekommenderad arbetsinsats är minst 20h/vecka (16h/vecka för DV1493)

Innehåll

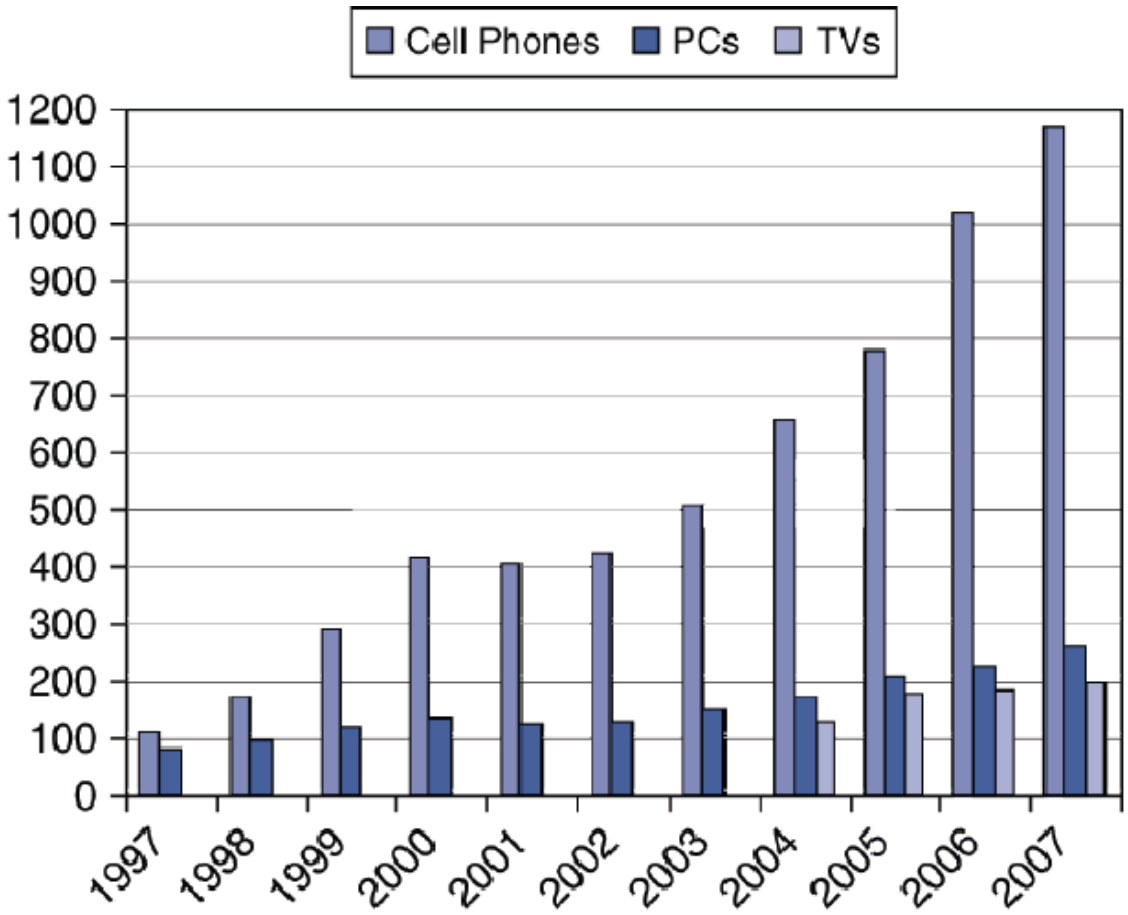
Treveckorsupprop

- Treveckorsuppropet går ut på att ni ska jobba lite med Linux
- www.netacad.com
- Kursen “Linux_Ess_VT_2017”
- Modul 4, 6 och 7 ska göras

Innehåll

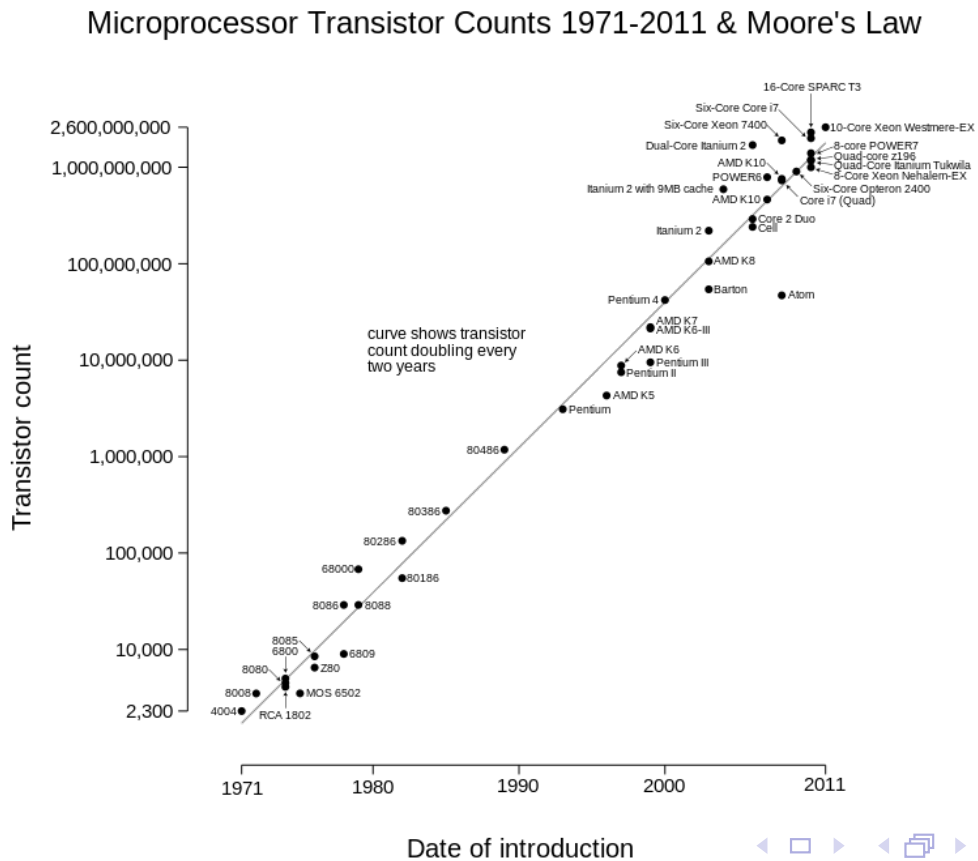
Var finns datorer?

Marknad för processorer



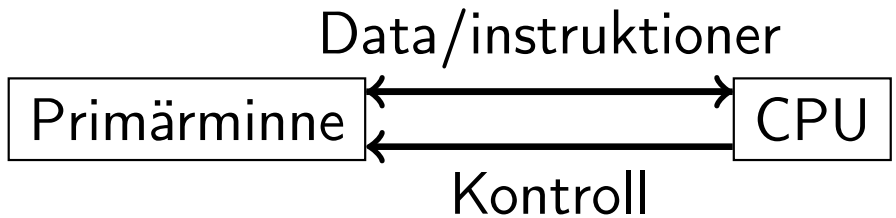
Moore's Lag

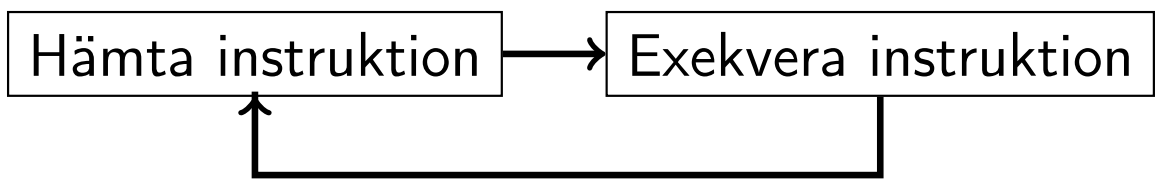
1965 formulerade Gordon E. Moore *Moore's lag*. Lagen säger att antalet transistorer på ett chip dubblas var 24:e månad.



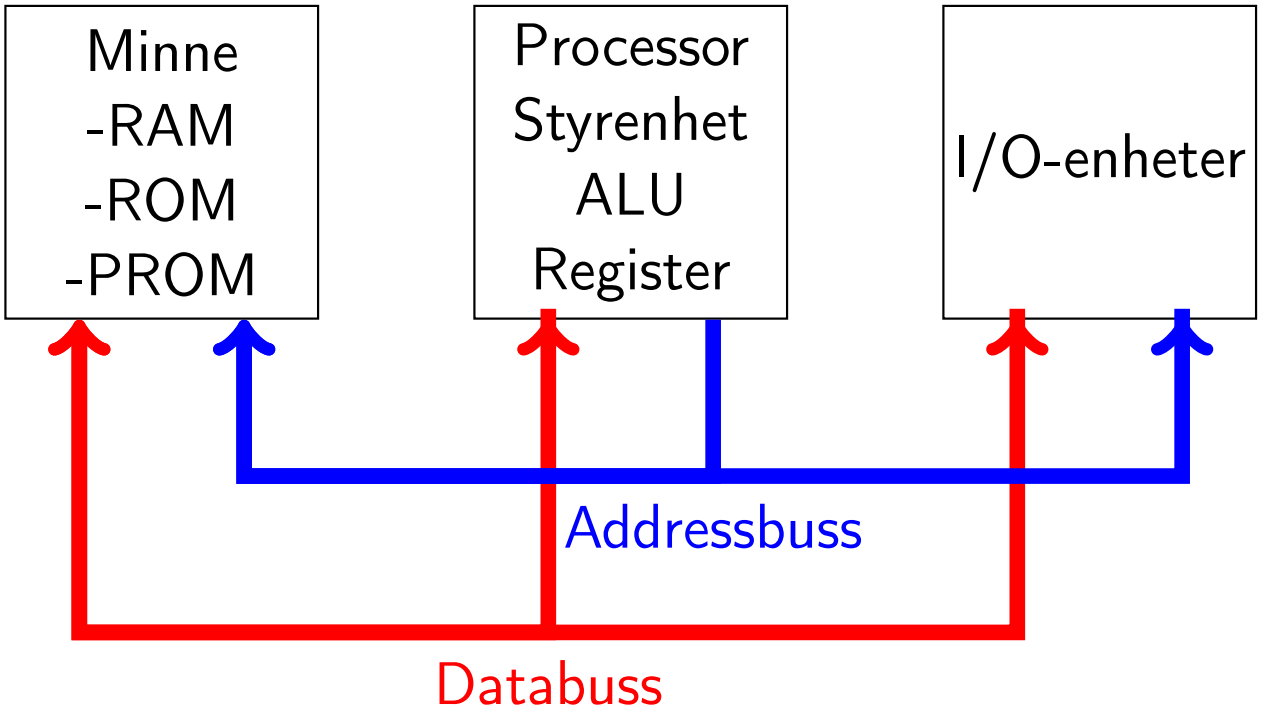
Vad "gör" en dator?

- Beror på programmet som körs
- Generellt gör datorn bara tre saker:
  - Läser ifrån minne (data eller instruktioner)
  - Exekverar instruktioner
  - Skriver till minne





Hårdvarans viktigaste delar



Användarprogram, systemprogram och hårdvara

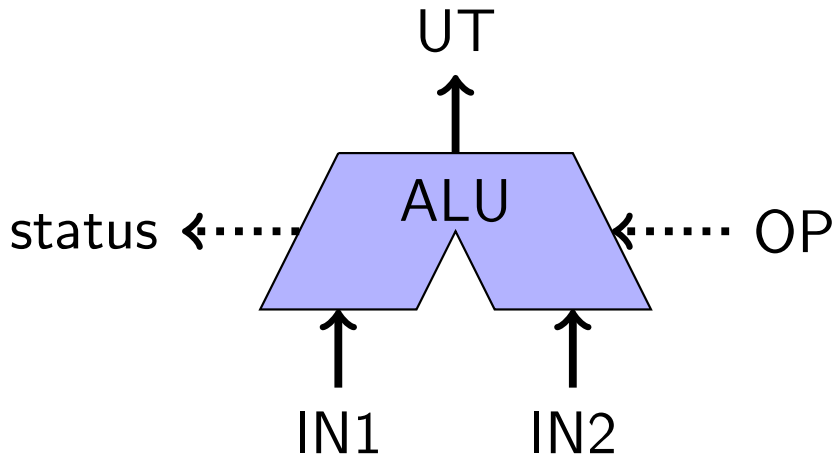
- Applikationsprogram är närmast användaren, t.ex. LibreOffice, Firefox eller Spotify. De skrivs i högnivåspråk som C++.
- Systemprogram
  - Kompilatorer: Översätter program ifrån högnivåspråk till lågnivåspråk.
  - Operativsystem: Hanterar minne, I/O, och schemalägger jobb. Exempel är Windows, Debian och Mac OSX.
- Hårdvara är t.ex. processorer och minne

Processorns delar

- Styrenhet - Ser till att alla instruktioner hämtas och att de gör det de ska
- ALU - **Arithmentic Logic Unit**, utför aritmetiska och logiska beräkningar
- Register - Små minnen som håller data processorn arbetar med
- Systemklocka - Slår an takten för datorns arbete

ALU - Arithmetic Logic Unit

- IN1, IN2: Ingångar
- UT: Utgång, resultat
- OP: Operation (ADD, SUB, AND...)
- Status: Overflow, Zero...



Register

- Alla processorer har någon typ av registerfil för temporär lagring
- Register är väldigt små minnesceller i CPU:n
- Tumregel: Ju mindre minnet är, desto snabbare är det
- ARM har 16 register, varav 13 är general purpose
  - Används för data som används ofta
  - Registren är numrerade r0-r15
  - Data på 32 bit kallar "word" (ordlängd=stolek på CPU reg.)
  - I assemblern har olika register olika användningsområden
    + `r0-r3` är scratch register
    + `r4-r12` är general purpose som behöver sparas
    + `r13` är stackpekaren (kallas även sp)
    + `r14` är länkregistret (lr)
    + `r15` är programräknaren (pc)

Register, forts

Register som är synliga för programmeraren kan vara:

- Helt generella
- Specifika för data/instruktioner
- Specifika för heltal eller flyttal
- Specifika för att hantera resultat vid division/multiplikation
- Specifika för att kunna hantera adressberäkningar (t.ex. basregister, stackpekare)

Status- och kontrollregister

Status- och kontrollregistren är inte direkt åtkomliga för programmeraren

- Programräknaren (PC)[2]: Innehåller adressen på den instruktion som ska hämtas
- Instruktionsregister (IR): Innehåller den senast hämtade instruktionen
- Memory Address Register (MAR): Minnesadress som ska läsas/skrivas
- Memory Buffer Register (MBR): Den data som ska skrivas till minnet eller precis lästs därifrån
- Program status word: Diverse flaggor (är avbrott på/av, är kernel mode påslaget...)

[2] I ARM går det att modifiera PC:n direkt

Register och primärminne

- Högnivåspråk:

  ```
  int c=a+b;
  ```

- Assemblerspråk:

  ```
  ADD c,a,b
  ```

- Kompilatorn (assemblern för att vara specifik) hittar en lämplig maskininstruktion
- 3 minnesadresser:
  - Hämta a
  - Hämta b
  - Skriv c

Register och primärminne

- Högnivåspråk:

```
int d=a+b+c;
```

- Assemblerspråk, alternativ 1:

```
ADD d,a,b # a+b sparas i d (minne)
ADD d,d,c # d+c sparas i d (minne)
```

- Totalt sex minnesoperationer
  - Om minnesaccess tar 100μs innebär det att additionen tar 600μs.
- Assemblerspråk, alternativ 2:

```
ADD r1,a,b # a+b sparas i r1 (register)
ADD d,r1,c # r1+c sparas i d (minne)
```

- Totalt fyra minnesoperationer, två registeroperationer
  - Om minnesaccess tar 100μs och registeraccess tar 1μs blir det totalt 402μs

Primärminne

Minnesoperationer

- **LOAD**
  Läser data ifrån minne och placerar datan i ett register
- **STORE**
  Skriver data ifrån ett register till minnet

Bitgrupper av olika storlek

- Byte - 8 bitar
- Ord (word) - 32 bitar (på 32-bits arkitekturer)
- Nibble - 4 bitar
- Halfword - 16 bitar

Byte- och word-adresserat minne

- Byteadresserat:
  - Måste hålla koll på vad som ska anses vara ett ord
  - Exempelvis, läs adress 4, som är en del i ett ord
- Wordadresserat:
  - Fragmentering om en byte ska lagras (tar upp 4 byte)
  - Läser ett word i taget

Hur sätts bytes ihop till words?

Givet ordet

(MSB) 1111 1111 0000 1111 1111 0000 1010 1010 (LSB)

Big Endian

Adress	Data
....	....
n	1111 1111
n+1	0000 1111
n+2	1111 0000
n+3	1010 1010
....	....

Little Endian

Adress	Data
....	....
n	1010 1010
n+1	1111 0000
n+2	0000 1111
n+3	1111 1111
....	....

Innehåll

Kort om Linux

- Linux är en operativssystemkärna
- Det finns många OS som bygger på Linux
  - Debian
  - Ubuntu
  - Arch
- I kursen kommer ni komma i kontakt med Ubuntu och Raspbian
- På många sätt likt Windows
  - Fönster
  - Skrivbord
  - Muspekare
- Den största skillnaden är **terminalen**

Linux terminalen

- Terminalen är Linux motsvarighet till Windows kommandotolk
- Istället för att använda musen för att göra saker skriver man kommandon
- Att kunna använda terminalen är viktigt när man jobbar med Linux
- Mycket kan göras mycket snabbare och smidigare
- För att komma igång behöver man kunna ett par grundläggande kommandon
- Treveckorsuppropet har med det att göra
- På It's Learning finns det ett dokument som ni kan använda som referens

Linux terminalen 101

Kommando	Exempel	Förklaring
`cd <DIR>`	`cd src`	Går in i en mapp
`mv <FILE1> <FILE2>`	`mv main.c main.cc`	Flyttar eller döper om en fil, första argumentet är originalfilen
`cp <FILE1> <FILE2>`	`cp main.c main.c.OLD`	Kopierar en fil, första argumentet är originalfilen
`cat <FILE>`	`cat main.c`	Skriver ut en fil i terminalen
`head <FILE>`	`head kernel.log`	Skriver ut första 10 raderna av en fil i terminalen
`tail <FILE>`	`head kernel.log`	Skriver ut sista 10 raderna av en fil i terminalen

Linux terminalen 101, forts.

`rm <FILE>`	`rm a.out`	Tar bort en fil
`rm -r <DIR>`	`rm -r bin`	Tar bort en mapp och allt innehåll
`mkdir <NAME>`	`mkdir bin`	Skapa en ny mapp
`touch <NAME>`	`touch main.c`	Skapa en ny fil
`./<NAME>`	`./a.out`	Kör en körbar fil
`chmod +x <FILE>`	`chmod +x a.out`	Gör en fil körbar
`nano <FILE>`	`nano main.c`	Öppna en fil för redigering med nano
`vim <FILE>`	`vim main.c`	Öppna en fil för redigering med vim

Linux terminalen - Textredigering

- Raspberry Pi labbarna kommer kräva att ni redigerar text i terminalen
- Det finns två texteditorer ni kan välja på: vim och nano
- vim
  - Extremt kraftfull när man lärt sig
  - Tar lite tid att komma in i
- nano
  - Lätt att komma igång med
  - Inte lika kraftfull som vim

Linux terminalen - Textredigering