CPUer og maskinkode

DM534

Rolf Fagerberg

Mål

Målet for disse slides er at beskrive, hvordan maskinkode ser ud og hvordan CPUen udfører et program i maskinkode.

Dette emne er et uddrag af kurset *DM548 Computerarkitektur og systemprogrammering* (3. semester).

CPUers opbygning

En CPU er bygget op af elektriske kredsløb (jvf. sidste forelæsning), som kan manipulere bits.

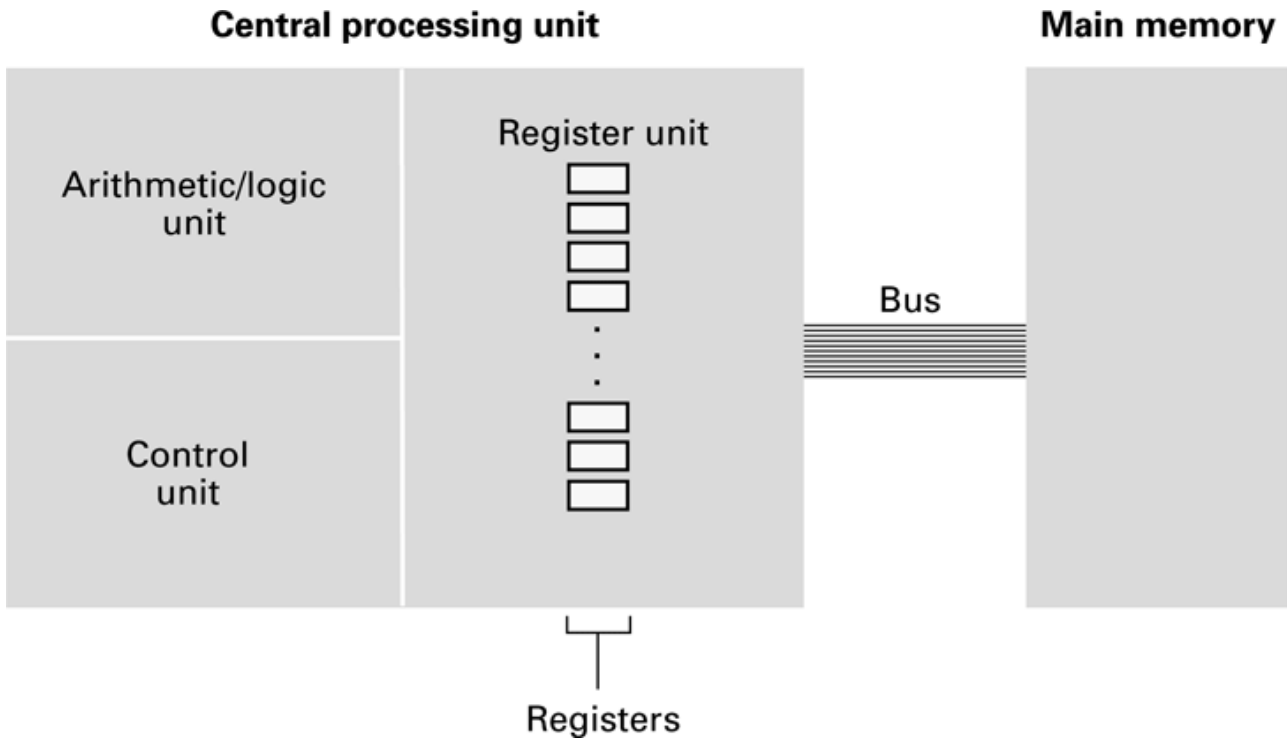
En CPU manipulerer flere bits ad gangen, deres antal kaldes CPUens *ordlænge* (typisk 32 eller 64).

CPUers opbygning

En CPU er bygget op af elektriske kredsløb (jvf. sidste forelæsning), som kan manipulere bits.

En CPU manipulerer flere bits ad gangen, deres antal kaldes CPUens *ordlænge* (typisk 32 eller 64).

Typisk er en CPU struktureret således:



Instruktionssæt

En CPU tilbyder en række ret simple kommandoer til at manipulere ord. Disse kaldes CPUens *instruktionssæt*.

Der er normalt kommandoer til

- Flytte ord mellem RAM (main memory) og CPUens registre.
- Lave simple beregninger på ord i CPUens registre (plus, minus, gange, dividere, AND, OR,...).
- Ændre hvorhenne i programmet den næste kommando læses fra (jump).

Instruktionssæt

En CPU tilbyder en række ret simple kommandoer til at manipulere ord. Disse kaldes CPUens *instruktionssæt*.

Der er normalt kommandoer til

- Flytte ord mellem RAM (main memory) og CPUens registre.
- Lave simple beregninger på ord i CPUens registre (plus, minus, gange, dividere, AND, OR,...).
- Ændre hvorhenne i programmet den næste kommando læses fra (jump).

Et eksekverbart program er en række af sådanne kommandoer.

Programmet ligger i RAM. Kommandoerne er, som alt andet i en computer, repræsenteret ved en række bits.

Instruktionssæt

En CPU tilbyder en række ret simple kommandoer til at manipulere ord. Disse kaldes CPUens *instruktionssæt*.

Der er normalt kommandoer til

- Flytte ord mellem RAM (main memory) og CPUens registre.
- Lave simple beregninger på ord i CPUens registre (plus, minus, gange, dividere, AND, OR,...).
- Ændre hvorhenne i programmet den næste kommando læses fra (jump).

Et eksekverbart program er en række af sådanne kommandoer.

Programmet ligger i RAM. Kommandoerne er, som alt andet i en computer, repræsenteret ved en række bits.

Man programmerer sjældent direkte i maskinsprog. Programmer skrevet i et normalt programmeringssprog såsom Python, Java, C++ eller C# oversættes til maskinsprog før (eller mens) det kører.

CPU cyclus

En CPU arbejder i en cyclus:

- Fetch: Hent næste kommando (en samling bits) fra programmet i hukommelsen til CPUens.
- Decode: Konverter kommandoen (en samling bits) til kontrolsignaler internt i CPUen.
- Execute: Disse kontrolsignaler aktiverer de relevante dele af CPUen.

Der er ca. 10^9 cykler i sekundet i en typisk CPU.

CPU cyclus

En CPU arbejder i en cyclus:

- Fetch: Hent næste kommando (en samling bits) fra programmet i hukommelsen til CPUens.
- Decode: Konverter kommandoen (en samling bits) til kontrolsignaler internt i CPUen.
- Execute: Disse kontrolsignaler aktiverer de relevante dele af CPUen.

Der er ca. 10^9 cykler i sekundet i en typisk CPU.

Adressen i RAM på den næste kommando som skal hentes, angives af et specielt register kaldet *program counter*. Normalt tælles program counter én op efter hver cyclus (sekventiel programudførsel).

CPU cyclus

En CPU arbejder i en cyclus:

- Fetch: Hent næste kommando (en samling bits) fra programmet i hukommelsen til CPUens.
- Decode: Konverter kommandoen (en samling bits) til kontrolsignaler internt i CPUen.
- Execute: Disse kontrolsignaler aktiverer de relevante dele af CPUen.

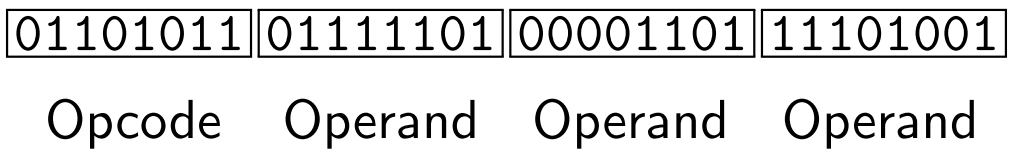
Der er ca. 10^9 cykler i sekundet i en typisk CPU.

Adressen i RAM på den næste kommando som skal hentes, angives af et specielt register kaldet *program counter*. Normalt tælles program counter én op efter hver cyclus (sekventiel programudførsel).

En jump kommando ændrer på indholdet af dette register, og kan dermed styre hvilke kommandoer i programmet som udføres. Sådan laves løkker, forgreninger (if/then/else) og metodekald.

Kommandoer

Et programs kommandoer er, som alt andet i en computer, repræsenteret ved en række bits. Disse er ofte struktureret således:



- *Opcode* angiver kommandotypen (flyt ord, plus, gange, jump,...).
- *Operand* angiver input til kommandoen. Det kan være f.eks. et registernummer, en hukommelsesadresse, eller et stykke data, alt efter typen af kommando.

En CPU simulator

Brookshear machine. Et program som simulerer en simpel CPU med:

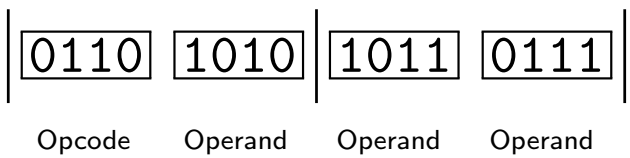
- 13 kommandotyper.
- 16 registre (plus en program counter).
- Ordlængde 8 bits.
- En RAM med 256 ord. Indeholder både program og data.

En CPU simulator

Brookshear machine. Et program som simulerer en simpel CPU med:

- 13 kommandotyper.
- 16 registre (plus en program counter).
- Ordlængde 8 bits.
- En RAM med 256 ord. Indeholder både program og data.

En kommando fylder to ord (16 bits), og er opbygget således:

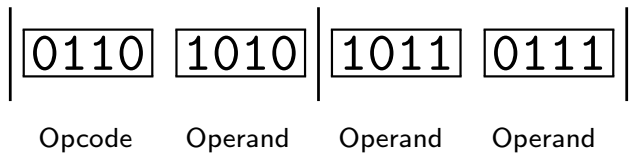


En CPU simulator

Brookshear machine. Et program som simulerer en simpel CPU med:

- 13 kommandotyper.
- 16 registre (plus en program counter).
- Ordlængde 8 bits.
- En RAM med 256 ord. Indeholder både program og data.

En kommando fylder to ord (16 bits), og er opbygget således:



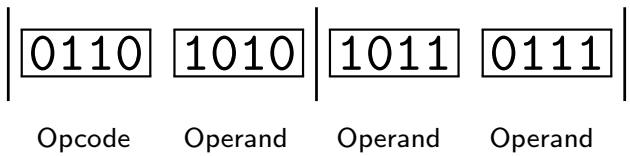
I simulatoren angives bits fire ad gangen via et hexadecimalt ciffer (0, 1, 2, 3, . . . , 9, A [= 10], B [= 11], C [= 12],. . . , F [= 15]).

En CPU simulator

Brookshear machine. Et program som simulerer en simpel CPU med:

- 13 kommandotyper.
- 16 registre (plus en program counter).
- Ordlængde 8 bits.
- En RAM med 256 ord. Indeholder både program og data.

En kommando fylder to ord (16 bits), og er opbygget således:



I simulatoren angives bits fire ad gangen via et hexadecimalt ciffer (0, 1, 2, 3, . . . , 9, A [= 10], B [= 11], C [= 12],. . . , F [= 15]).

En kommando fylder 4 hexadecimale cifre i alt. Heraf kan en kommando- og et registernummer hver angives med 1 ciffer ($2^4 = 16$), og en adresse i RAM med 2 cifre (da $16 \cdot 16 = 256$).

Instruktionssæt

Kommandoer: (hvor R, X og Y angiver et hexadecimalt ciffer).

Opcode	*Operands*	*Effekt*
1	*RXY*	Kopier indholdet af RAM celle XY til register R.
2	*RXY*	Læg bitmønstret angivet af XY ind i register R.
3	*RXY*	Kopier indholdet af register R til RAM celle XY.
4	*xRS*	Kopier indholdet af register R til register S.
5	*RST*	Lav addition af indholdet af register S og T og læg resultatet i register R. Indhold fortolkes som heltal i two's complement.
6	*RST*	Lav addition af indholdet af register S og T og læg resultatet i register R. Indhold fortolkes som flydende kommatal.

Instruktionssæt

Kommandoer:

Opcode	*Operands*	*Effekt*
7	*RST*	Lav bit-wise OR af indholdet af register S og T og læg resultatet i register R.
8	*RST*	Lav bit-wise AND af indholdet af register S og T og læg resultatet i register R.
9	*RST*	Lav bit-wise XOR af indholdet af register S og T og læg resultatet i register R.
A	*RxX*	Rotér indholdet af register R cyclisk mod højre X skridt.
B	*RXY*	Jump til instruktionen i RAM celle XY hvis indholdet i register R er lig indholdet i register 0.
C	*xxx*	Stop programmet.
D	*RXY*	Jump til instruktionen i RAM celle XY hvis indholdet i register R er større ($>$) end indholdet i register 0. Indhold fortolkes som heltal i two's complement.

Uddybning

Uddybning af et par af operationerne ovenfor:

- *Bit-wise OR* (eller AND eller XOR): Brug OR (eller AND eller XOR) på hver plads. Et eksempel med OR:

$$\begin{array}{r} 10011001 \\ 01010001 \\ \hline = 11011001 \end{array}$$

  På hver plads er den nye bit 1 hvis mindst én af de to gamle er 1 (jvf. definitionen af OR).

- *Cyclisk rotation mod højre*: Alle bits i ordet flyttes mod højre. Dem, som skubbes ud over højre ende af ordet, sættes ind igen i venstre ende i samme rækkefølge. Et eksempel på at rotere 3 skridt cyklisk mod højre:

$$01011001 \rightarrow 00101011$$

Eksempelprogrammer

Følgende program bytter om på indholdet af RAM celle 10 og 12:

1110
1212
3112
3210
C000

Eksempelprogrammer

Følgende program bytter om på indholdet af RAM celle 10 og 12:

```
1110
1212
3112
3210
C000
```

Her er programmet igen, med hver linie forklaret:

1110	Kopier indholdet af RAM celle 10 til register 1
1212	Kopier indholdet af RAM celle 12 til register 2
3112	Kopier indholdet af register 1 til RAM celle 12
3210	Kopier indholdet af register 2 til RAM celle 10
C000	Stop

(Hvis man skal se effekten når programmet kører, skal man først fylde RAM celler 10 og 12 med forskelligt indhold.)

Eksempelprogrammer

Følgende program illustrerer en løkke. Det skriver efter tur tallene $0, 1, 2, 3, \ldots, 6$ (dvs. indhold $00, 01, 02, 03, \ldots, 06$) i RAM celle 1E, hvis RAM celle 18 indeholder 07 til at starte med.

```
2000
2101
1518
301E
5001
D506
C000
```

Eksempelprogrammer

Her er programmet igen, med hver linie forklaret:

2000	Læg bitmønstret angivet (hexadecimalt) af 00 ind i register 0
2101	Læg bitmønstret angivet (hexadecimalt) af 01 ind i register 1
1518	Kopier indholdet af RAM celle 18 til register 5
301E	Kopier indholdet af register 0 til RAM celle 1E
5001	Lav addition af register 0 og 1, læg resultat i register 0
D506	Jump til instruktionen i RAM celle 06 (og 07) hvis indholdet i register 5 er større end indholdet i register 0. Når programmet placeres i starten af hukommelsen er denne instruktion 301E.
C000	Stop

(Husk først at fylde RAM celle 18 med indhold (f.eks. 07) inden programmet kører.)