# Obligatorisk øving til modul 1 i datateknikk

Dette er den første obligatoriske øvingen i datateknikk.

## Oppgave 1

Alle (også lærere) vet at det finnes kalkulatorer som gjør beregningene nedenfor. Ikke bruk slike her, men få arbeidsmetoden inn i fingrene. Deretter kan svarene evt kontrolleres med kalkulatoren. Det er neppe gitt en eneste eksamen i datamaskintekniske fag noe sted i verden uten at sjonglering mellom tallformater er med. Derfor er det viktig stoff. Husk at det ikke er tilstrekkelig å angi rett svaret. Dere må også vise fremgangsmåten for å få full uttelling på eksamen.

a) Skriv følgende binære tall som hexadesimale og desimale tall

0100000011110011

Svar:

0\*215 + 1\*214 + 1\*27 + 1\*26 + 1\*25 + 1\*24 + 1\*21 + 1\*20

= 16384 + 128 + 64 + 32 +16 + 2 + 1

= 16627

Desimal: 1662710

HEXA

16627 / 16 = 1039 rest 3

1039 / 16 = 64 rest 15

64 / 16 = 4 rest 0

Hex: 4 + 0 + 15(er F i hex) + 3

Hex: 40F316

ELLER:

Gjør om til 4 bit

0100 0000 1111 0011

4 + 0 + F + 3

Hexa: 40F3

1110101100111101

Svar:

Desimal: 6022110

1110 1011 0011 1101

E + B + 3 + D

HEXA: EB3D

b) Skriv følgende hexadesimale tall som binære tall og desimale tall

DE

Svar:

Desimal: 2710

Binær: 110112

B0

Svar:

Desimal: 1110

Binært: 10112

c) Skriv følgende desimale tall som binære tall

128

Svar:

100000002 (8bit)

245

Svar:

111101012

## Oppgave 2 Tallsystemer

Digitale datamaskiner forholder seg til binære verdier. En samling av binære verdier kaller vi for et bitmønster. For eksempel er instruksjoner bitmønstre som forteller CPU hva den skal gjøre, data er bitmønstre som representerer verdier på ulike datatyper (heltall, tegn, strenger …) og adresser er bitmønstre som unikt identifiserer ulike deler av datasystemet.

Et eksempel på en adresse er MAC-adresse (Media Access Control Address eller maskinvareadresse). Denne adressen identifiserer utstyr som kan kobles til et nettverk. MAC-adressen er et 48-bits tall. MAC-adressen skrives gjerne på heksadesimalt form som seks grupper av tall med to sifre i hver gruppe. Gruppene er gjerne atskilt med kolon eller bindestrek for å øke lesbarheten ytterligere.

1. Hvor mange ulike MAC-adresser finnes det?

Svar:

48 bit 🡪 248 = 281 474 976 710 656 ulike adresser

1. Skriv følgende MAC-adresse på binær form: 00-20-18-6d-fe-8a

Svar:

00 = 0000 0000

20 = 0010 0000

18 = 0001 1000

6d = 0110 1011

Fe = 1111 1101

8a = 1000 1100

Svar: 0000 0000 0010 0000 0001 1000 0110 1011 1111 1101 1000 1100­2

## Oppgave 3

1. Når en CPU utfører en instruksjon gjennomløper den en instruksjonssyklus. I sin enkleste form kan vi si at instruksjonssyklusen består av to delsykluser, nemlig en *hentesyklus* og en *utføringssyklus*. Hva skjer i hentesyklusen, og hva skjer i utføringssyklusen?

Svar:

Hentesyklusen: Det første som skjer i en instruksjonssyklus er hentesyklusen. I hentesyklusen hentes instruksjonen fra minnet. Adressen til instruksjonen i minnet ligger i PC (programteller). Denne instruksjonen hentes fra denne adressen i minnet og legger i IR (instruksjon registeret, her ligger instruksjonen som CPU skal utføre i utføringssyklus. Etter dette legges det en til PC hvis ikke annet er bedt om, f.eks. som det vil i en for loop og if setning.

Utføringssyklus: Etter at instruksjonen er hentet og plassert i IR, skal den utføres eller executes. Instruksjonen er på binær form i et bestemt bitmønster som sier hva CPU skal gjøre. Denne utføringen kontrolleres av kontrollenheten som tolker instruksjonene og setter opp de riktige kontrollsignalene. Det er ALU-en som utfører disse oppgavene den får tilsendt.

1. Hvis du ikke har innbakt svaret på denne deloppgaven i oppgave a), så svar på følgende spørsmål: Hva er register, og hva brukes programtelleren (PC), instruksjonsregisteret (IR), og akkumulatorregisteret (AC) til?

Svar:

Register: CPU-en har et lite internt minne kalt et register. Her ligger data som det skal utføres operasjoner på.

PC: Holder styr på adressen til neste instruksjon, legges alltid til en etter instruksjon er utført med mindre annet er spesifisert.

IR: Er et instruksjon register som inneholder instruksjonene som skal utføres i CPU.

AC: Er et korttidslager for data, også kalt en akkumulator. Kan f.eks. brukes til å lagre et tall midlertidig slik at man kan gå til neste instruksjon og hente et annet tall som skal adderes med tallet i AC.

1. Når vi skal tolke bitmønsteret til en instruksjon – f.eks. slik figur 5 i leksjonen viser – snakker vi om at den består av en op-kode og (en eller flere) operander.   
     
   Hva er en op-kode, og hva er en operand?

Svar:

OP-kode: Er et bitmønster som sier hvilken operasjon som skal utføres i CPU, eksempler er addisjon, subtraksjon. Blir også kalt operasjonskode. En instruksjon kan ha ulik lengde etter hvilken bit man bruker, f.eks. hvis det brukes 1 bit til op-kode kan man ha to forskjellige operasjoner, men med 4 bit kan man ha 16 operasjoner (2n).

Operand: Den første delen av instruksjonen består av en OP-kode, men resten av instruksjonen er adressen til hvor dataen ligger. Vi kan ikke alltid anta at disse dataene ligger i minnet, ofte kan de ligge i et CPU register eller på en I/O modul, derfor kaller man det en operand. Den skal fortelle hvor data som skal behandles ligger.

## Oppgave 4 (viktig oppgave)

Ta utgangspunkt i den hypotetiske maskinen som ble beskrevet i leksjonen.

Denne hypotetiske maskinen skal utstyres med I/O-utstyr. I/O-utstyr styres av såkalte I/O-moduler (kontrollere), og CPU kommuniserer med I/O-modulene ved å skrive/lese bitmønstre til eller fra I/O-modulene på omtrent samme måte som CPU skriver/leser til/fra minnet. Datamaskinen kan ha mange I/O-moduler, og for å skille mellom dem får hver I/O-modul sitt eget nummer – omtrent som hver minnelokasjon i minnet har sin egen adresse. Oppbygningen er vist i figuren nedenfor:



I tillegg til de instruksjonene som ble beskrevet i leksjonen, finnes følgende to instruksjoner for å kommunisere med I/O-modulene:

|  |  |  |  |
| --- | --- | --- | --- |
| **Bitmønster** | **Hex-verdi** | **Symbolsk navn** | **Funksjon** |
| 00112 | 316 | READIO | Les data fra en I/O-modul til AC. Adressefeltet angir hvilken I/O-modul som skal leses. |
| 01112 | 716 | WRITEIO | Skriv til en I/O-modul fra AC. Adressefeltet angir hvilken I/O-modul det skal skrives til. |

I/O-modulene er altså nummerert, og man kan adressere hver enkelt av dem ved å angi dette nummeret i adressefeltet til instruksjonen.

**Eksempel:**

Les fra I/O-modul nr 9 og legge resultatet i AC.

Symbolsk kode: READIO 9

Heksadesimal kode: 3 0 0 9

Binær kode: 0011 0000 0000 1001

### Oppgaver:

a) Skriv et symbolsk program som

1. Leser data fra I/O-modul 5 og legger det i AC

**Eksempel:**

Les fra I/O-modul nr 5 og legge resultatet i AC.

Symbolsk kode: READIO 5

Heksadesimal kode: 3 0 0 5

Binær kode: 0011 0000 0000 0101

1. Adderer innholdet av minnelokasjon 940

**Eksempel:**

Adderer innholdet av minnelokasjon 940

Symbolsk kode: ADD 940

Heksadesimal kode: 5 0 0 9

Binær kode: 0101 0011 1010 1100

1. Skriver resultatet til I/O-modul 6.

**Eksempel:**

Skriv resultat til I/O modul 6

Symbolsk kode: WRITEIO 6

Heksadesimal kode: 7 0 0 6

Binær kode: 0111 0000 0000 0110

Du trenger både instruksjonene i tabellen ovenfor, og en av instruksjonene som er beskrevet i leksjonen.

Svar:

Ligger ovenfor under hver instruksjon.

1. Hva blir bitmønstrene for disse instruksjonene?

Svar:

Bitmønsteret: de første fire bitene i instruksjonen er OP-koden. Resten av de 12 bitene som er ledige brukes til å angi adressen / operanden.

0111 0000 0000 0110

1. Tegn en figur som viser hva som skjer når programmet kjører. Figuren skal bruke samme oppsett som figur 7 i leksjonen. Anta at programmet starter i minnelokasjon 300. Ta gjerne utgangspunkt i figuren nedenfor (evt på neste side):





*Utfører instruks. 302: STORE/lagre data i AC i I/O modul nr.6*

*Utfører instruks 301: add/legge til data fra minne adresse 940 til data i AC, og lagre i AC*

*Utfører instruks. 300: load/hent data fra I/o modul nr.5 og legg i AC*

*Henter instruksjon 302 og legger den i IR*

*Henter instruksjon 300 og legger den i IR*

*Henter instruksjon 301 og legger den i IR*

7006

5009

3005

7006

5009

3005

7006

7006

0001

0003

0003

0001

0004

0004

0004

303

302

0004

1 + 3 = 4

302

5009

5009

0001

3005

0001

301

5009

3005

7006

0003

0003

0003

0001

0001

0001

5009

3005

7006

5009

3005

7006

5009

3005

7006

301

3005

0003



Instruksjonene på symbolsk form ligger i minnet

Dataene ligger også i minnet på en annen adresse

300



1 ligger på I/O modul 5. Nr.6 er tom.

0001

Dette programmet har tre ulike deler, hvor den henter instruksjonen og utfører instruksjonen i hver del.

1. LOAD
2. ADD
3. STORE

PC: Holder styr på adressen til neste instruksjon, legges alltid til en etter instruksjon er utført med mindre annet er spesifisert.

IR: Er et instruksjon register som inneholder instruksjonene som skal utføres i CPU.

AC: Er et korttidslager for data, også kalt en akkumulator. Kan f.eks. brukes til å lagre et tall midlertidig slik at man kan gå til neste instruksjon og hente et annet tall som skal adderes med tallet i AC.