# 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

0100 0000 1111 0011

Svar:

Desimal: 16627

Hex: 40F3

1110 1011 0011 1101

Svar:

Desimal: 60221

Hex: EB3D

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

DE

Svar:

Binær: 1101 1110

Desimal: 222

B0

Svar:

Binær: 1011 0000

Desimal: 176

c) Skriv følgende desimale tall som binære tall

128

Svar:

1000 0000

245

Svar:

1111 0101

## 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:

Det finnes 2^48 adresser.

Det er 281474976710656 adresser.

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

Svar:

0000 0000 0010 0000 0001 1000 0110 1101 1111 1110 1000 1010

## 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:

I hentesyklusen må CPUen hente en instruksjon fra minnet. PC-registeret holder styr på hvor i minnet instruksjonene ligger. Deretter legges instruksjonen i IR-registeret. Hvis instruksjonen ikke sier noe anne, øker PC-registeret med én.

I utføringssyklusen utfører CPUen instruksjonen i IR-registeret.

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:

Et register er et lager i CPUen med plass til et «ord» (instruksjon, data, adresse).

AC-registeret er register for kortidslagring av data.

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-koden er et bitmønster som forteller CPUen hvilken operasjon den skal utføre.

Operand er et annet ord for adresse, og forteller altså hvor data ligger hen, enten det er minnet, et register eller en I/O-modul.

## 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
2. Adderer innholdet av minnelokasjon 940
3. Skriver resultatet til I/O-modul 6.

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

Svar:

READIO 5

ADD 940

WRITEIO 6

1. Hva blir bitmønstrene for disse instruksjonene?

Svar:

0011 0000 0000 0101

0101 1001 0100 0000

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):

