Seminar 13 (oppsummering)

Pensumrepetisjon

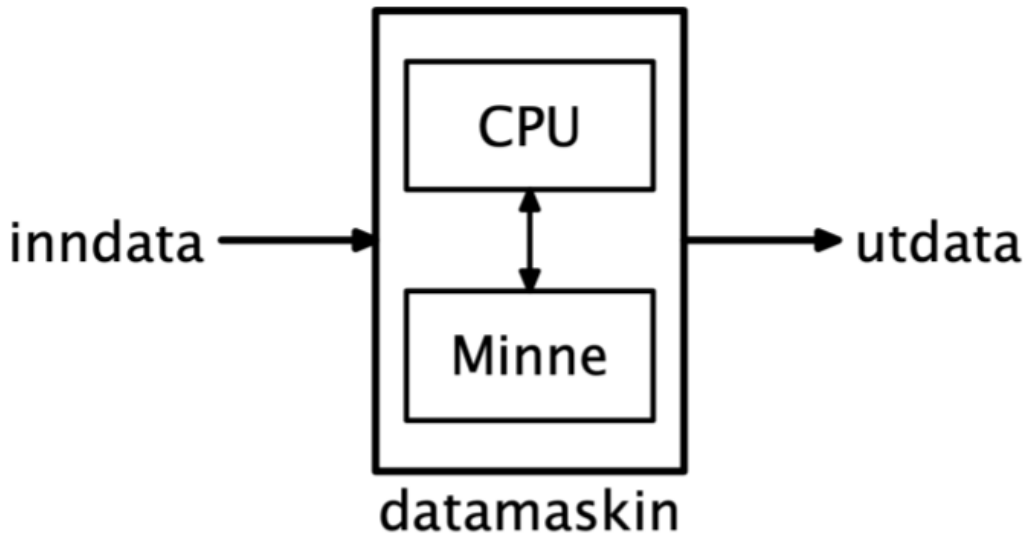
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2c2d4205-8b1f-42a9-84a1-b74d429e2c73/uke-46-oppsummering-datasikkerhet.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4484674b-5ec2-4a88-9276-9f7b6d19c61d/uke-45-nettverk-oppsummering.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/46a28809-701f-4587-8d9a-d73a060e5259/oppsummering-lmc-og-digital-representasjon.pdf

LMC og prosessorarkitektur

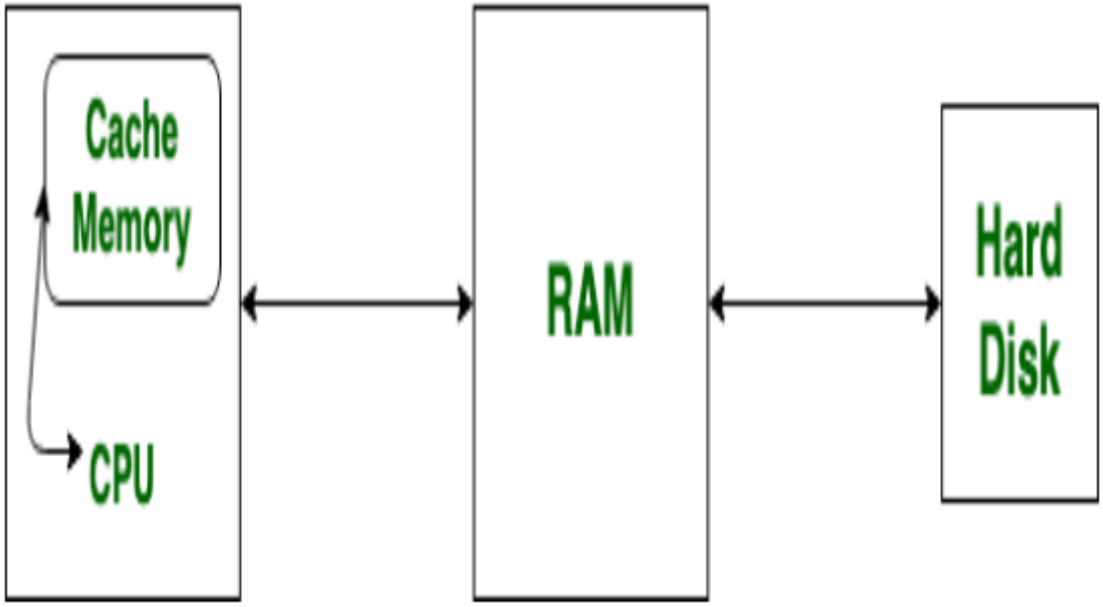
Von Neumann-akrutektur



- Minnet holder på både *data* og *instruksjoner* til CPU-en

CPU (Central Processing Unit)

- ALU (Arithmetic Logic Unit) - holder styr på aritmetiske og logiske operasjoner. LMC sin ALU kun adderer og subtraherer tall.
- Registre - små minneceller med hurtig tilgang med hver sin unik oppgave
- Cache vs Register



LMC

LMC

ASCII tabell (både base 10 og 16)

- Alle instruksjonene i LMC:

Kode	Navn	Beskrivelse
0xx	HLT	Stopper eksekveringen
1xx	ADD	Adderer verdien i angitt minnelokasjon med akkumulatoren
2xx	SUB	Subtraherer verdien i angitt minnelokasjon med akkumulatoren
3xx	STO	Lagrer akkumulatoren i angitt minnelokasjon
4xx	-	Ikke i bruk
5xx	LDA	Henter verdi fra minnet til akkumulatoren
6xx	BRA	Hopper til angitt adresse
7xx	BRZ	Hopper til angitt adresse hvis akkumulatoren er 0
8xx	BRP	Hopper til angitt adresse hvis akkumulatoren er ≥ 0
901	INP	Leser verdi fra input, og legger svaret i akkumulatoren
902	OUT	Skriver ut verdien i akkumulatoren
922	OTC	Skriver ut ASCII-tegn (ikke i boka)

Eksempler

1. Skriv kode som skriver ut "Hei" til terminal-vinduet.

   Tallkode:

   507
   922
   508
   922
   509
   922
   000
   072
   101
   105

   Assemblerkode:

   LDA *H* // Hent ASCII verdien til "H" fra akkumulatoren
   OTC // Skriv ut "H"
   LDA *e* // Hent ASCII verdien til "e" fra akkumulatoren
   OTC // Skriv ut "e"
   LDA *i* // Hent ASCII verdien til "i" fra akkumulatoren
   OTC // Skriv ut "i"
   HLT

H DAT 072 // = "H"
e DAT 101 // = "e"
i DAT 105 // = "i"

2. Skriv kode som leser inn et tall og skriver det ut uendret om input er positiv, og skriver det tilsvarende positive tallet om input er negativ.
   Tallkode:
   901
   805
   307
   207
   207
   902
   000
   Assemblerkode:

```
    INP          // Les inn et tall.
    BRP posi // Om tallet er positiv, hopp til posi. Om den er negativ, gjør ingenting.
    STO tall // Lagre tallet i minnecelle tall.
    SUB tall // Subtraher det tallet fra seg selv, resulterer i 0.
    SUB tall // Subtraher igjen til å få det tilsvarende positive tallet.
posi OUT      // print tallet
     HLT

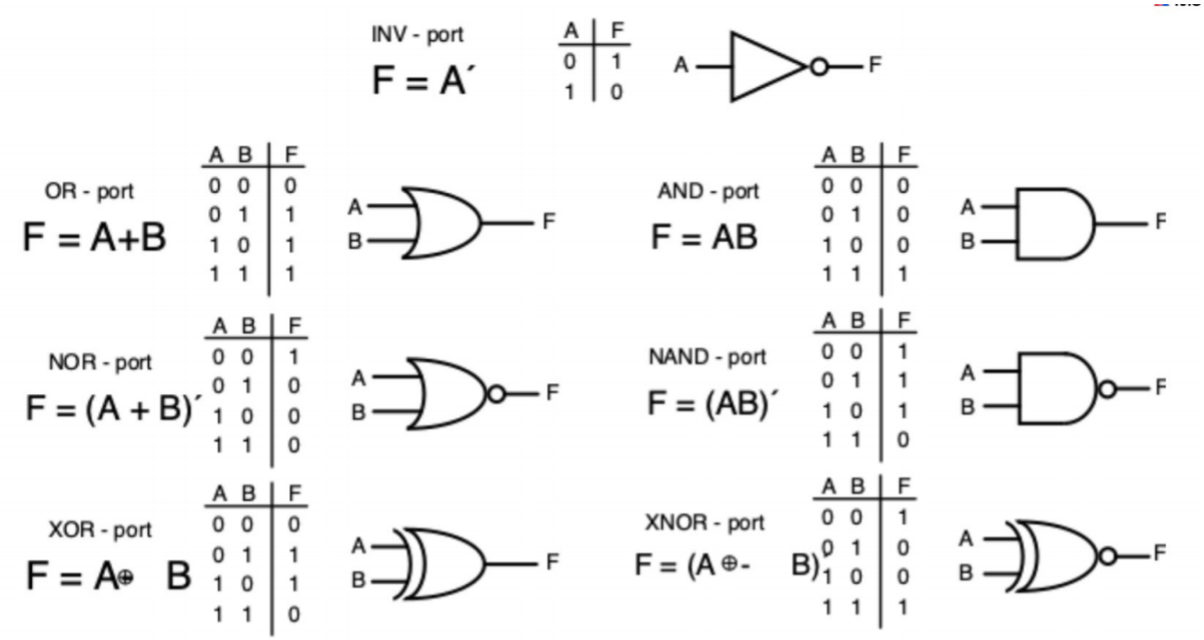
tall  DAT 0  // placeholder for tall
```

Eksempel 2 er tatt fra ukeoppgaver til uke 4. For mer LMC-koding, se her:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4f441ef4-258e-4653-9e90-dcba2dc571ad/oppgaver-lmc-1.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/27621cbe-764e-45e4-84ba-19d1945a18c3/fasit-lmc-1.pdf

Porter og funksjoner



Bindesterkhet rekkefølge

- A' - sterkest
- A*B
- A+B - svakest
- () - forsterker alt som er inni

To måter å løse kretsoppgaver

- **Venstre til høyre**. Brukes når vi bygger en krets fra en gitt formel.
  - Går fra de sterkeste til de svakeste (↓, topp til bunn), siden vi går fra enkelte uttrykk (A, B) til de komplekse (F).
- **Høyre til venstre**. Brukes når vi har en ferdig implementert krets og trenger å skrive en formel.
  - Går fra de svakeste til de sterkeste (↑, bunn til topp), siden vi går fra de komplekse uttrykk (F), til de enkelte (A, B).

Noe regler i boolsk algebra

- **Inverse law for addition (OR-port):** A + A' = 1
- I**nverse law for multiplication (AND-port):** A * A' = 0

- **Distributive law:** A + BC = (A + B)(A + C)
- **"Decyphering" (simplifying) XOR port:** F = A xor B = A'B + AB'

For mer eksamenslesing:

Eksamensnotat av Henrik Heiberg (Høst 2018)

Nettverksdel

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f49f65c3-7cf4-47d4-81b2-de91dd63f1b6/uke-9---historie-og-lagdeling-i-datakommunikasjon.pdf