# Prosessorarkitektur

## CPUens deler og virkemåte

### CPUens oppbygging

CPU er en elektronisk krets som er i stand til utføre instruksjoner. Disse instruksjonene ligger på et lavt abstraksjonsnivå og er grunnleggende og enkle av natur.

#### Instruksjonssyklusen

Instruksjonssyklusen begynner med hentesyklusen. I hentesyklusen hentes en ny instruksjon fra minnet. Instruksjonen legges i IR. I utføringssyklusen utfører CPU denne instruksjonen. Senere innførte vi også avbruddsyklusen. Når du fortsetter å lese denne leksjonen, er det viktig å huske at CPU normalt veksler mellom hente- og utføringssyklusen. I avbruddssyklusen skjer det normalt ingenting. Unntaket er når det har kommer et avbrudd. Men der er tross alt relativ sjelden.

### Den viktigste delene til CPU

Den viktigste delen av CPU er:

* Registerblokken
  + Finnes mange ulike type registre. Noen er synlig for brukere, og programmere har tilgang til disse, andre er kun for OS.
  + Register for brukere:
    - Generell registre: Disse brukes til data som CPU trenger under utføring
    - Flagg-registeret: Dette er et register som bare kan lses, og som angir status for CPUen.
    - Adresseregistre: Adresseregistre inneholder adresser til minnet. Det er altså registre som peker til forskjellige deler av minnet. Indeksregistre med eller uten automatisk opptelling brukes til indeksering i tabeller og arrayer.
  + Register ikke tilgjengelig for bruker:
    - Programteller
    - Instruksjonsregister
    - Registre som brukes av operativsystemet og som kun kan aksesseres av privilegerte instruksjoner
    - Interne registre som brukes til buffring under utførelse av instruksjoner
* ALU
  + Er en komponent som kan utføre aritmetiske og logiske operasjoner på binære data.
  + ALU har to innganger og en utgang, hvor både inn og utgangene er registre.
  + Begge inngangene kan kombineres på mange måter
    - Aritmetiske Operasjoner
      * Adderes
      * Subtraheres
      * Multipliseres
      * Divideres
    - Logiske Operasjoner
      * Sammenlignes
* Kontrollenhet
  + Aktivitetene til CPU styres av en kontrollenhet som sender kontrollsignaler til de andre enhetene. Disse kontrollsignalene er binære signaler; de er enten av eller på. Eksempler på kontrollsignaler er:
    - Kontrollsignaler som bestemmer hvilken ALU-operasjon som skal foretas.
    - Kontrollsignaler som åpner/lukker for dataflyt mellom to registre
* Klokkekretsen
  + Elektroniske pulser bygd rundt en et kvartskrystall, som gir faste presise pulser.
  + Klokkefrekvensen er rundt 300-4000 MHz, som betyr ny handling mellom 300 millioner - 4 milliarder ganger i sekundet.
* Intern CPU-bussen
  + Ekstrem rask buss/hurtig parallell buss.
  + Den benytter CPUens klokkefrekvens, og er koblet til alle delene i CPUen som trenger å utveksle informasjon
  + De ulike delene snakker over den interne CPU-bussen.

### Mikrooperasjoner

* Aktivitetene til CPU styres av en kontrollenhet som sender kontrollsignaler(binære) til de andre enhetene.
* Aktivitetene til en CPU i løpet av en klokkepuls kalles mikrooperasjon.
* Å utføre en instruksjonen består i å utføre en sekvens av slike mikrooperasjoner.
* Typer av mikrooperasjoner:
  + Flytte data mellom to registre
  + Utføre aritmetiske eller logiske operasjoner med ALU. Input data til ALU ligger i registre og ALU legger resultatet i et register
  + Flytte data mellom registre og minne eller IO-registre (I/O-porter) via systembussen

### Flytte data mellom registre

* Det er kontrollenhet som styrer overføringen ved å sette den rette kontrollsignaler. Registrene har brytere som skrus av og på (åpen/lukke), som styres binært.
* For å overføre åpner bryteren ut fra reg1 og bryteren inn til reg2. Alle andre brytere sperrer for dataflyt

### Bruke ALU

* ALU kombinerer to bitmønste som ligger i hver sin inngang, og produserer et resultat.
* Hver inngang har fire binære signaler slik at vi får 16 ulike operasjoner den kan utføre.

### Kommuniserer med systembussen

* Kommunikasjon med omverden/resten av pcen skjer med systembussen. Da må den "Oversette" bitmønstre på bussen slik at cpu kan lese/skrive. Da bruker den 2 spesialagenter:
  + ABR-register (AdresseBuss Register): Et register som tilknyttet adressebussen på systembussen. Registret inneholder adressen til en minnelokasjon. CPU setter adresselinjene til bussen ved å skrive et bitmønster på ABR-registeret.
  + DBR-register (DataBuss Register): Et register som er tulknyttet databussen på systembussen.
    - Dersom CPU vil skrive til databussen, så legger CPU bitmønster på DBR
    - Dersom CPU skal lese fra minnet, vil minne legget bitmønster på databussen, og CPU kan lese det fra DBR-registeret

## Kontrollenhet

### Kontrollenhetens oppgaver

Kontrollenhetens oppgaver:

1. Sekvensering. Kontrollenheten skal sørge for at CPU stepper gjennom en serie av mikrooperasjoner i riktig rekkefølge.
2. Eksekvering. Kontrollenheten bruker kontrollsignaler for å sørge for at mikrooperasjonen blir fullført.

### Kontrollenhetens funksjon

Tar inn en av tre type signaler og gir ut et av to kontrollsignal. Av disse tre type innsignalene er det:

* IR-registeret som inneholder instruksjonen som skal utføres. Den er delt opp i op-kode og operand(er).
* Klokkesignal varsler når det skal settes nye kontrollsignaler.
* Kontrollsignalene på systembussen, blant annet avbruddssignalet

Av utsignalene har vi:

* Interne kontrollsignal som sendes til CPUens deler
* Eksterne kontrollsignal som sendes ut fra CPU via systembussens kontrollinjer til andre deler av datamaskinen. Et eksempel (se leksjonen om avbrudd) er linjen INT ACK (Avbruddsbekreftelse), som er et varsel til avbruddskildene om at et avbrudd er mottatt og registrert.

### Kontrollenhetens oppbygging

#### Hardlogikk

Hardlogikk kontrollenheter som bygges opp av såkalte kombinatoriske kretser hvor utgangene alltid er entydig gitt av inngangene. Ved design av hardlogikk tar man for seg kontrollsignalene ett for ett. For hvert av kontrollsignalene finner man et boolsk utrykk for den kombinasjon av alle innganger som skal sette akkurat det kontrollsignalet. Denne prosessen gjentas for hvert eneste kontrollsignal i prosessoren.

#### Mikroprogrammert kontrollenhet

Et sett av 12 bit mikroinstruksjon blir laget og lagret i et lite internminnet i CPU kalt kontrollminnet. Dette mønsteret blir sent til CPU som bruker mikroinstruksjonene verdier (0 er ikke utfør, 1 er utfør) og utfører de nødvendige kontrollsignalene.

### Hardkoding contra Mikroprogrammering

Fordelen med hardkodet er at den er veldig rask siden den har en definert mengde operasjoner den kan utføre, og den vet nøyaktig hvordan den skal utføres, men ettersom de nyeste datamaskinene har såpass komplekse operasjoner krever det stadig mer kompliserte intrigerte kretser. Fordel for Mikroprogrammering er at siden den er såpass dynamisk kan man utføre enormt mange operasjoner med de samme kretsene, men de er tregere.

### Kontrollenheten på moderne CPUer

### Optimalistering av utføringen

Hvordan øke utføringshastigheten til CPUen

#### Øke klokkefrekvensen

Klokkefrekvensen er viktig størrelse i alle prosessorer, men hvis den blir for stor vil prosessoren brenne opp. Men siden størrelse har mye å si når det gjelder varmegang har vi klart å lage mindre prosessorer som har ha høyere klokkefrekvens, siden den tåler mer varme.

#### Forbedret oppbygging og arkitektur

Flere og bedre intrigerte kretser fører til at man kan utføre samme oppgave på færre klokkesykluser. Også har man fått flere viktige mekanismer for å gjøre prosessoren bedre som Pipeline, superskalar arkitektur og dynamisk utføring.

# Moderne datamaskinarkitektur

## Pipeline

En instruksjon syklus består av en hentesyklus og en utføringssyklus. Vi antar at disse blir utført på to ulike deler av elektronikken. Dermed kan vi utføre en hentesyklus og når instruksjonen blir hentesyklusen utført samtidig. Da er det viktig at begge bruker like lang tid, ellers blir den ene ventende på den ander.

### Fem-trinns pipeline

#### Trinn 1, FI

Hentesyklusen (Fetch Instruction)

#### Trinn 2, DI

Tolke instruksjonen. Planlegge hva som skal skje. (Decode Instruction)

#### Trinn 3, FO

Hente data fra minnet, cache eller IO dersom instruksjonen trenger det. Slik data kalles inn-operander. (Fetch Operand)

#### Trinn 4, EI

Utføre instruksjonen. (Execute Instruction)

#### Trinn 5, WO

Enkelte ganger skrives det til minnet. Det er ikke nødvendig at alle instruksjoner skriver til minnet. (Write Operand)

### Utføring av en fem-trinn pipeline

Vi antar at hvert trinn tar en klokkesyklus. Dermed vil vi for hver klokkesyklus begynne på en ny instruksjon. Den første instruksjonen vil ta 5 klokkesykluser å fullføre, men deretter vil hver instruksjon kun ta en klokkesyklus. Alle trinn, uansett om de blir brukt eller ikke vil bli utført. Dette er for å holde samme format på teknologien

## Problemer med parallellitet på instruksjonsnivå