Moderne prosessorarkitektur

* Hver eneste instruksjon som skal utføres må gjennom hentefasen og utføringsfasen.
* **Hentefasen:** Instruksjonen hentes fra minnet og legges i instruksjonsregisteret.
* **Utføringsfasen:** Tolke instruksjoner, evt. Hente data fra minnet dersom instruksjonen trenger slike data, utføre instruksjonen, evt. Skrive data til minnet dersom instruksjonen lager et resultat som skal skrives dit.
* Om vi deler instruksjonssyklusen inn i finere deler enn de to delene:
  + **FI -** Fetch Instruction (Hent instruksjon)
  + **DI -** Decode Instruction (Dekod instruksjonen)
  + **FO -** Fetch Operand (Hent evt innoperand (i minnet))
  + **EI -** Execute Instruction (Utfør instruksjon)
  + **WO -** Write Operand (Skriv evt utoperand (til minnet))
  + Dette er en (av mange) mulige måter å dele opp instruksjonssyklusen i mindre deler.
  + Det er ganske vanlig med oppdeling i 4 - 8 trinn, selv om det finnes eksempler på CPU-er med langt flere trinn (for eksempel bruker Pentium II 12 trinn, Pentium IV brukte 20)
* **Instruction-level parallelism**
  + Parallell utføring av instruksjoner innfører nye problemer:
    - Flere instruksjoner kan trenge samme ressurs
    - Resultater som trengs i en instruksjon er ikke ferdig beregnet
  + Slike situasjoner kalles hasard, disse deler vi inn i tre ulike hasards:
    - **Strukturell hasard:** Flere instruksjoner trenger samme ressurs samtidig.
    - **Data-hasard:** En instruksjon trenger resultatet fra en annen instruksjon som ikke har resultatet klart enda
    - **Kontroll-hasard:** Hopp i programmet gjør at vi ikke vet hvile instruksjoner som skal utføres før hoppinstruksjonen er utført
      * **Se presentasjon moderne prosessorarkitektur for eksempler.**
    - Disse hasardene oppstår ikke så ofte som man skulle trodd siden ikke alle instruksjoner skriver og henter fra minnet. Men om hasard skulle oppstå løser vi dette med tomgangstrinn (setter trinn på vent).
* **Superskalare prosessorer**
  + Superskalare prosessorer har **flere** parallelle pipelines.
  + Antall parallelle pipelines kalles prosessorens grad av superskalar.
  + Eks: en superskalar prosessor av grad 2 har to parallelle pipelines. Instruksjonene utføres parvis:



* **Dynamisk utføring**
  + Dersom en instruksjon må vente på minnet eller L2-cache vil CPU begynne å jobbe på instruksjoner videre i programmet selv om den ikke vet helt sikkert at instruksjonene virkelig skal utføres.
  + Viktige begreper:
    - **Dataflytanalyse:** 
      * Finn ut hvilke instruksjoner som er avhengige av hverandre
    - **Gjetting ved hopp:**
      * Effektiviteten blir svært avhengig av det gjettes riktig ved betingede hopp. Feil gjetting gjør at det utføres feil instruksjoner
    - **Spekulativ utføring:**
      * Instruksjoner utføres gjerne i en helt annen rekkefølge enn den sekvensielle. Resultatene legges i “foreløpige registre”. Deretter frigis de (retired) i samme rekkefølge som de er programmert.