Risc og Cisc

* En RISC kan i utgangspunktet utføre relativt enkle instruksjoner, der r står for redusert i navnet.
* Grunnen til at de to forskjellige oppstå er fordi antall transistorer har økt veldig noe som har gjort at vi kan få utført kompliserte instruksjoner. Før i tiden da man ville ha mer kompliserte instruksjoner men de hadde ikke transistorer til å gjøre dette derfor brukte de mikroprogrammerte kontrollenheter.
* Siden vi kan oppnå så mye med antall transistorer vi har i dag har vi gått tilbake til hardlogikk.

**CISC**

* + Effektivt med avanserte instruksjoner som forsøkte å tette «det sematiske gap» mellom høynivå og lavnivå programmering
    - Man hadde lyst på den tiden å kunne programmere kontrollenheten i et språk som lignet programmeringsspråk på den tiden
  + Svært fleksible løsninger for funksjonskall som ikke begrenser programmererens arbeid
  + Mikroprogrammerte kontrollenheter gir svært fleksible instruksjonsformater
* **Avanserte instruksjoner**
  + Fordel:
    - effektiv programmering
  + Ulempe:
    - Utføringstiden varierer: enkle instruksjoner trenger få, mikrooperasjoner kompliserte krever svært mange mikrooperasjoner. Vanskelig å lage effektiv pipeline
    - Implementasjon av de kompliserte instruksjonene gjør at også enkle instruksjoner går langsomt. Vanskelig å bruke hardlogikk.
* **Svært fleksible løsninger for funksjonskall**
  + Ingen begrensinger på antall parametere
  + Ingen begrensinger på antall lokale variabler
  + **Fordel:**
    - Legger få begrensninger på programmeringen
  + Ulempe:
    - Hvert funksjonskall krever minnetrafikk fordi parametere og lokale variable ligger i minnet.
* **Mikroprogrammert kontrollenhet gir fleksibelt instruksjonsformat**
  + Fordel:
    - Mye brukte instruksjoner kan være korte – noe som gir lite minnetrafikk
  + Ulempe:
    - Komplisert og tidskrevende å tolke instruksjoner
    - Mye vertikal koding. Feltet er avhengig og kan ikke tolkes i parallell

**RISC**