Pipelining (HW)

Hvad er pipelining?

- En **implementeringsteknik** hvor flere instruktioner kan overlappe hinanden i eksekvering
- Flere operationer kører samtidig
- Formålet med Pipelining er at **gøre processorer hurtige**
- Eksempel: **Vaskeeksemplet**

Hvilke fordele/ulemper har pipelining?

- Fordele:

- CPU'en udnyttes optimalt ved at køre **flere operationer samtidig**

- Ulemper:

- Langsomste instruktion bestemmer clock cycle og bruger derved l**ængere tid på korte instruktioner** end nødvendigt
- **Dyrere** at fremstille -> Bl.a flere flip-flops

Nævn nogle hazard typer og mulige løsninger på disse?w

- Hazard er **betegnelse** der sker i Pipelining når en **instruktion ikke kan køres** fordi den er **afhængig** af andet som ikke er klar endnu.
- Hazard **forhindrer** næste instruktion i at blive kørt
- Der findes **3 typer:**
  - **Data:** Når næste instruktion er **afhængig** af den **forriges resultat**. -> **Vent** til færdiggørelse
    - **Løsning:**
      - **Forwarding:** Intern buffer bliver brugt til at gemme data -> Ikke vente på memory access (write/read operationer)
  - **Structural:** Har med **hardware begrænsninger** at gøre. Eks1: Der tillades kun én write af gangen.
  - **Control:** Har med **branch afhængighed** at gøre. Næste instruktion er afhængig af en branch
    - **Løsning:**
      - **Antage** at det går godt og fortsætte. -> **Branch prediction**
      - **Stall** indtil der branches -> Bruge **Bubbles**
- **Bubbles** bruges til at stalle en operation i en Pipeline

Hvordan kan en "branch" predikteres?

- Der er **forskellige måder** at gøre det på:
  - **Stall:** Antag at branch ikke er taget -> Stall Pipeline hvis branch er taget
  - **Dynamic prediction:** Brug **log** til at gemme tidligere branching -> HW afhængig
  - **Delayed decision:** Kun MIPS. Eksekver en **betingelsesløs** instruktion lige efter en branch -> **Mere tid**

Pipelining (HW)