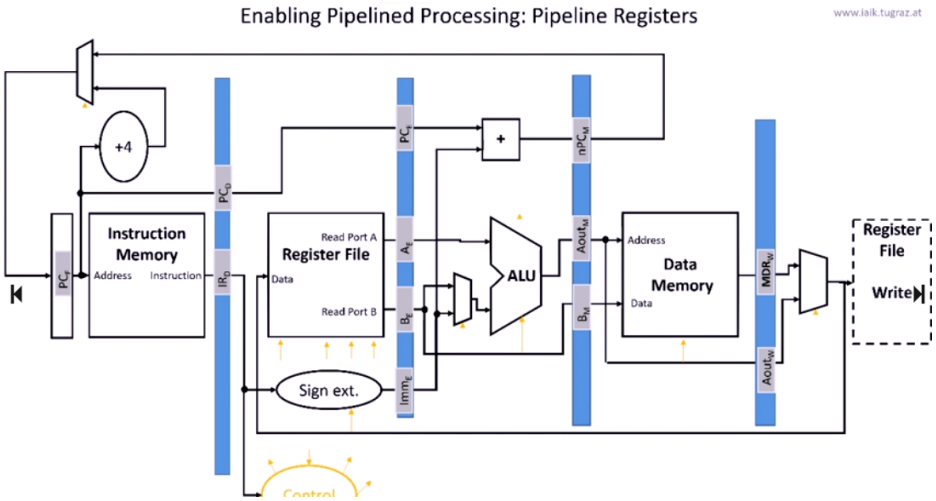
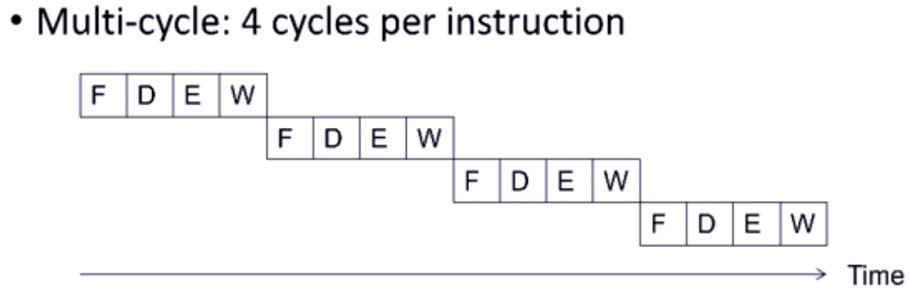
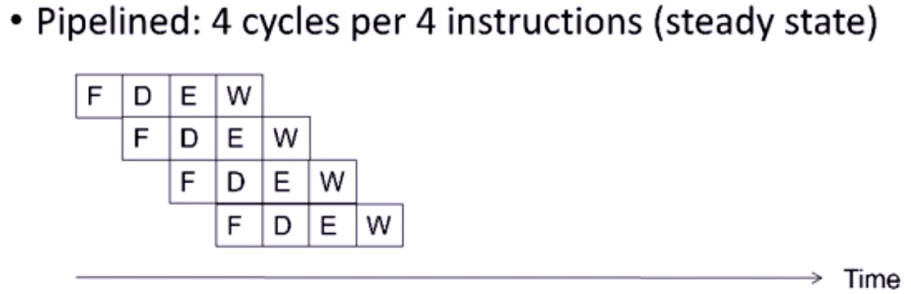
Motivation

- Fließbandprinzip
  - each component has own dedicated task
  - invented by Henry Ford
- pipeline the execution of multiple instructions
  - assembly line of instructions
- divide instruction processing into distinct stages
  - different instruction executed each stage
  - requires additional registers between each stage
  - 

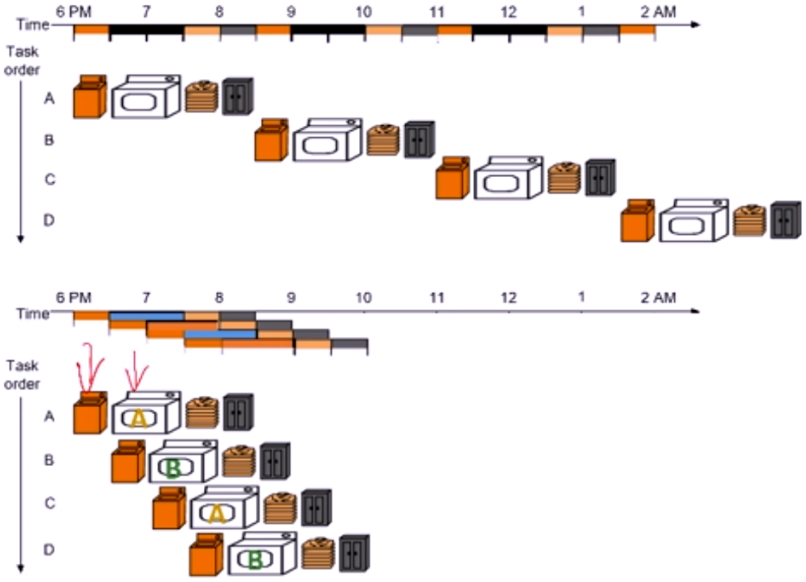
Example

- four independent ADDs
  - 

  - 

- laundry analogy

- compensate being slow on specific stage
- multiple instance of slow stage



- throughput restored (2 loads per hour) using 2 dryers

Ideal Pipelining

Goal: Increase throughput with little increase in cost
(hardware cost, in case of instruction processing)

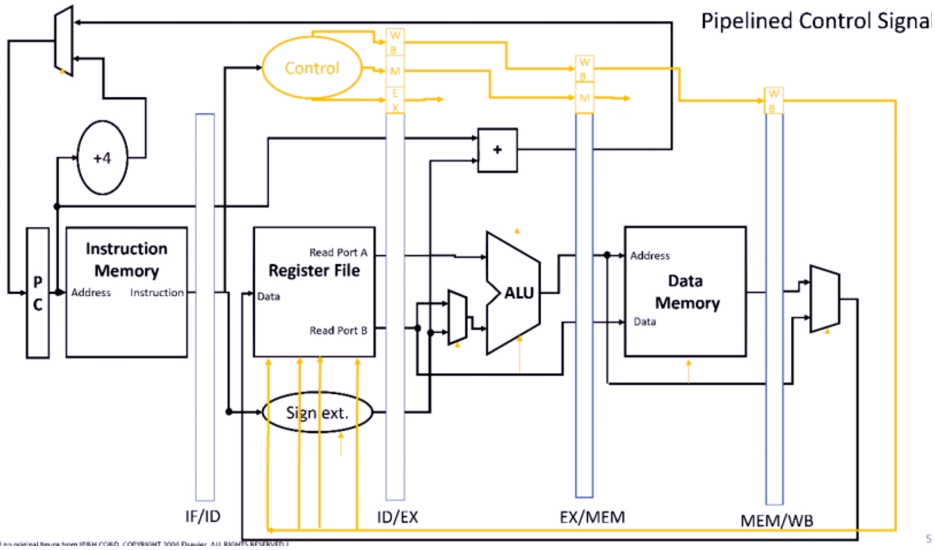
Repetition of identical operations
- The same operation is repeated on a large number of different inputs (e.g., all laundry loads go through the same steps)

Repetition of independent operations
- No dependencies between repeated operations

Uniformly partitionable suboperations
- Processing can be evenly divided into uniform-latency suboperations (that do not share resources)

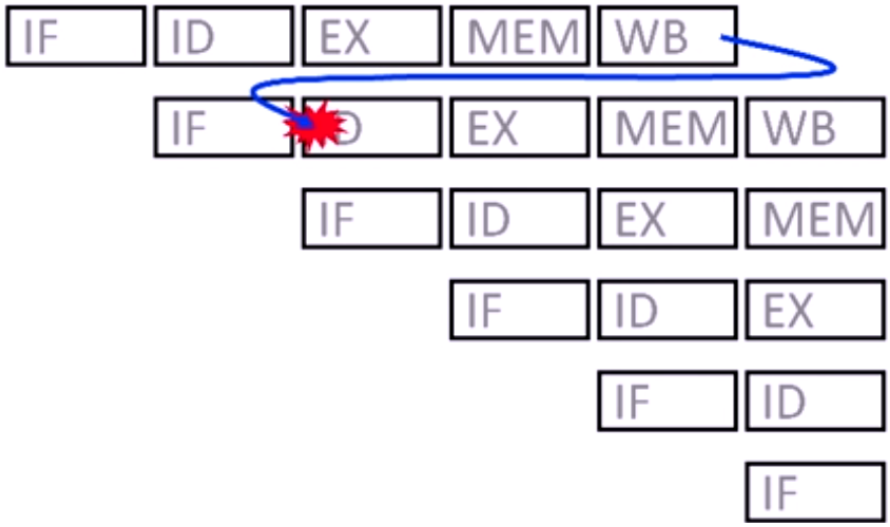
Control Signals

- signals piped through and used when needed



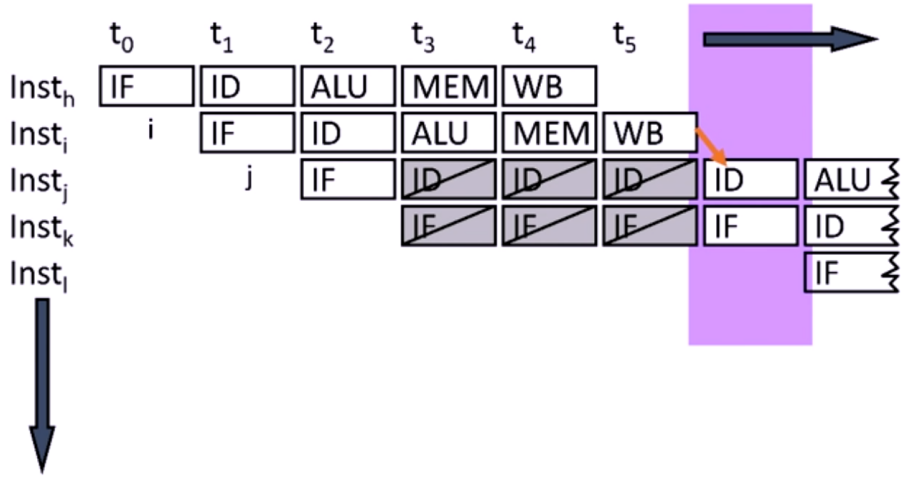
RAW Dependence Handling

- RAW - read after write
- sometimes result needed by instruction not calculated yet
  - e.g. 2nd instruction needs value from first
  - which has not finished yet



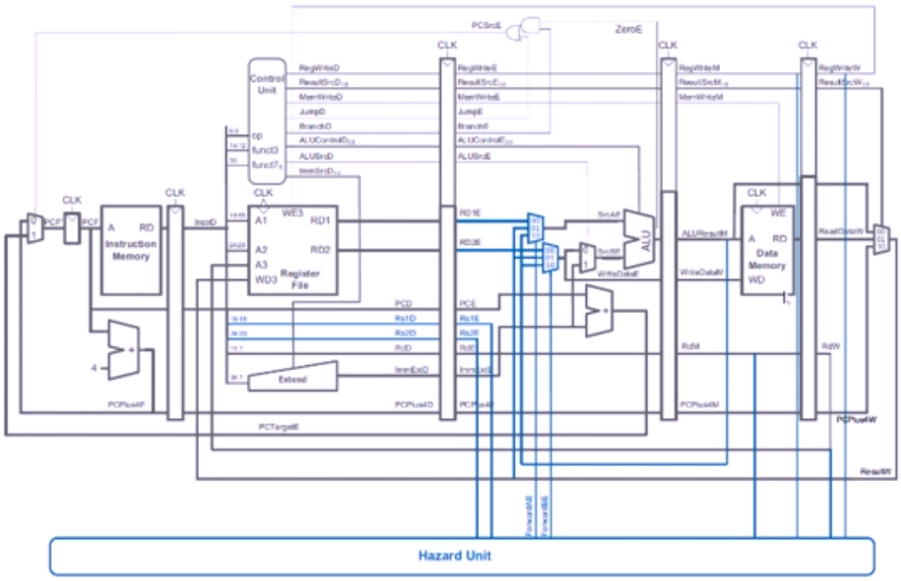
  -

- stall
  - wait until data value available



–

- dependence detection
  - scoreboarding
    - each registers has valid bit
    - instruction writing to register resets valid bit
    - instruction in decode stage needs to check if all source registers are valid
  - data forwarding/bypassing
    - forward result to dependent instruction as soon as available
    - hazard unit detects dependencis between instructions



*