Pipelining

Pipelining Defined

- Metaphor: assembly line
  - Split a job into N sequential sub-jobs, each taking roughly the same amount of time.
  - Each sub-job is processed by a different resource (station).
  - The job moves from one station to the next when it is completed.
- Example: doing laundry
  - Sub-jobs: washing, drying, folding/ironing
  - Each job takes 30 minutes
  - How long does one load take? 5 loads? N loads?

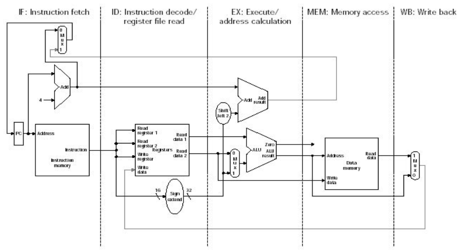
Performance

- Each pipe stage must take equal time.
- Throughput is enhanced, latency may be longer. Why?
- What is the ideal speedup?
- What factors keep us from attaining this ideal?

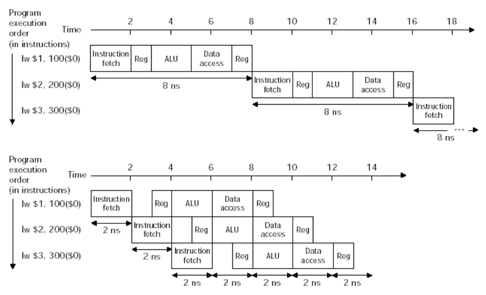
MIPS Pipelining

- Break the single cycle up into 5 sub steps.
  - IF: Instruction Fetch – get the next instruction
  - ID: Instruction Decode – decode the instruction & read the registers
  - EX: Execute – use the ALU, calculate branch address
  - MEM: Memory – read/write memory
  - WB: Write Back – write results back to the register file
- On each cycle, the machine does a little work on each instruction in the pipeline.
- How many cycles does a single instruction take?
- What is the CPI?

The 5 Stages



Performance vs Single Cycle



Why It's Not So Simple

- We need to remember information between stages.
- There may be dependencies between instructions:
  - Data dependencies:

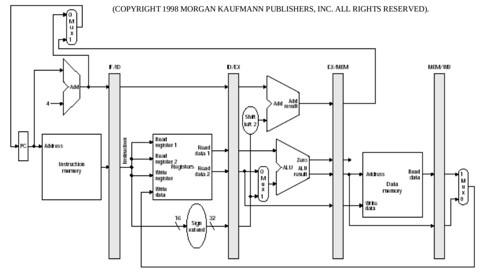
```
addi    $t0, $0, 13
add     $t1, $t0, $t0
```

  - Control dependencies:

```
beq     $t0, $t2, somewhere      # what if the branch is taken?
add     ....
div     ....
```

- Ignore the dependencies for now…

Adding Pipeline Registers



Pipeline Register Contents

- What information is needed in each pipeline reg?
- IF/ID:
- ID/EX:
- Ex/Mem:
- Mem/Wb:

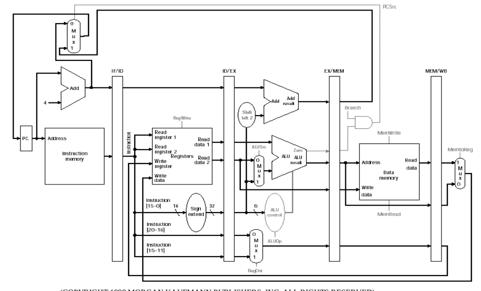
Stages: Fetch, Decode, Execute

- Instruction Fetch:
  - Get (and save) the next instruction; get (and save) nPC
- Instruction Decode:
  - Read the two registers (save these values)
  - Save the sign-extended immediate
  - Save the RD and RT register numbers
- Execute:
  - Use the ALU, save the result
  - Decide which is the real Destination register.
  - Calculate branch target

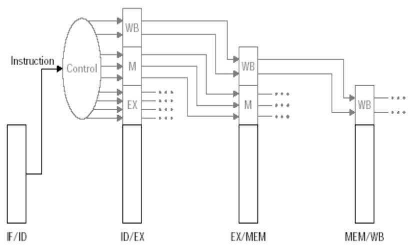
Stages: Memory & WriteBack

- Memory:
  - Branches: set the PC to the new target (if taken)
  - Memory instructions: access memory and save the result (on loads)
  - Other: just pass the ALU result along
- WriteBack:
  - Select which result to write to the register file (if necessary)

Control Lines



Adding Control Bits to the Pipe Regs


Instruction
Control
WB
M
EX
WB
M
WB
IF/ID
ID/EX
EX/MEM
MEM/WB