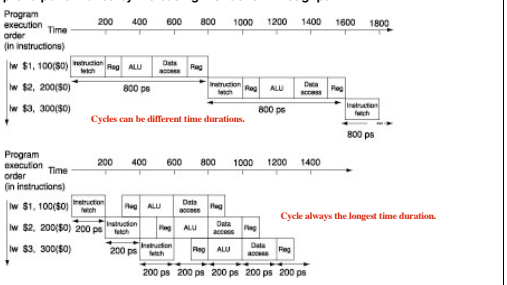
Chapter Six- 1st Half
Pipelined Processor
Delayed Controls, Hazards

EE3055

Web: www.csc.gatech.edu/copeland/jac/3055-05

Pipelining

- **Improve performance by increasing instruction throughput**

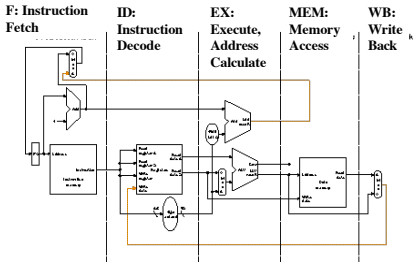


Ideal speedup is number of stages in the pipeline. Do we achieve this?

Pipelining

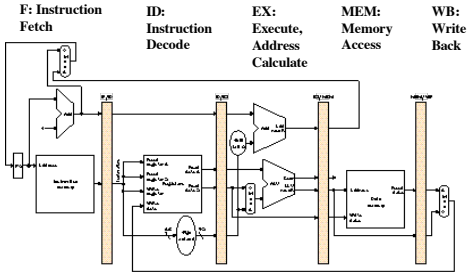
- **What makes it easy**
  - **all instructions are the same length**
  - **just a few instruction formats**
  - **memory operands appear only in loads and stores**
- **What makes it hard?**
  - **structural hazards: suppose we had only one memory**
  - **control hazards: need to worry about branch instructions**
  - **data hazards: an instruction depends on a previous instruction**
- **We'll build a simple pipeline and look at these issues**
- **We'll talk about modern processors and what really makes it hard:**
  - **exception handling**
  - **trying to improve performance with out-of-order execution, etc.**

Basic Idea



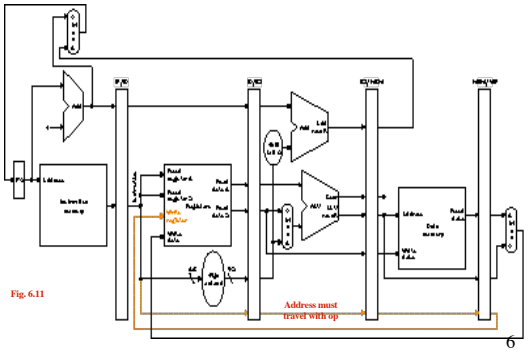
- *What do we need to add to actually split the datapath into stages?*

Pipelined Datapath

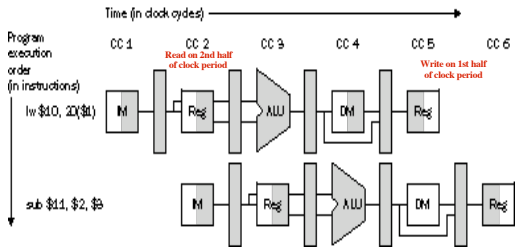


Can you find a problem even if there are no dependencies?
What instructions can we execute to manifest the problem?

Corrected Datapath



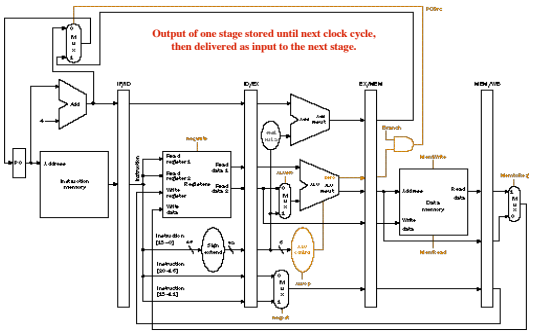
Graphically Representing Pipelines



- Can help with answering questions like:
  - how many cycles does it take to execute this code?
  - what is the ALU doing during cycle 4?
  - use this representation to help understand datapaths

7

Pipeline Control



8

Pipeline control

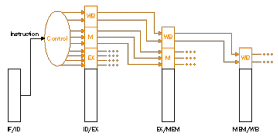
- We have 5 stages. What needs to be controlled in each stage?
  - Instruction Fetch and PC Increment
  - Instruction Decode / Register Fetch
  - Execution
  - Memory Stage
  - Write Back
- How would control be handled in an automobile plant?
  - a fancy control center telling everyone what to do?
  - should we use a finite state machine?

9

Pipeline Control

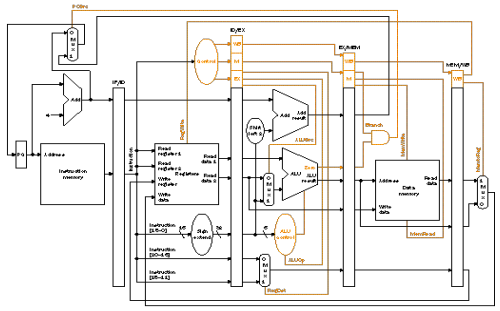
- **Pass control signals along just like the data**

Instruction	Execution/Address Calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg write	Mem to Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X



10

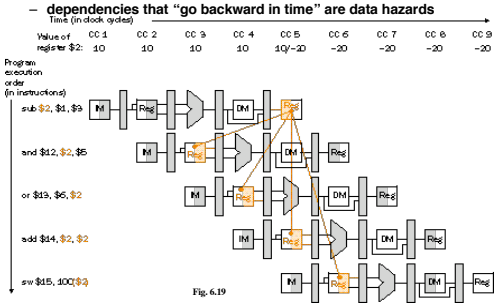
Datapath with Control



11

Dependencies

- **Problem with starting next instruction before first is finished**
  - **dependencies that "go backward in time" are data hazards**



12

Software Solution

- Have compiler guarantee no hazards
- Where do we insert the "nops" ?

```
sub    $2, $1, $3
and    $12, $2, $5
or     $13, $6, $2
add    $14, $2, $2
sw     $15, 100($2)
```

- Problem: this really slows us down!
- Hardware solutions - next set of slides.