

## 61C In the News

### Siri outage continues for some

Some iPhone 4S users say Siri is still out of commission, days after [it was first reported](#) that the popular, voice-activated artificially intelligent assistant appeared to be ailing.

Initially, when the iPhone 4S went on sale Oct. 14, many users couldn't get Siri to work because so many people were trying at the same time. **Siri needs to go through Apple's cloud-based servers to work**, and Siri's popularity caused a bit of a traffic jam.



11/7/11 Fall 2011 - Lecture #31 1

## CS 61C: Great Ideas in Computer Architecture (Machine Structures)

### Lecture 31: Pipeline Parallelism 2

Instructors:  
Mike Franklin  
Dan Garcia

<http://inst.eecs.Berkeley.edu/~cs61c/fa11>

11/7/11 Fall 2011 - Lecture #31 2

## Instruction Level Parallelism (ILP)

- Another parallelism form to go with Request Level Parallelism and Data Level Parallelism
- RLP – e.g., Warehouse Scale Computing
- DLP – e.g., SIMD, Map Reduce
- ILP – e.g., Pipelined instruction Execution
- 5 stage pipeline => 5 instructions executing simultaneously, one at each pipeline stage

11/7/11 Fall 2011 - Lecture #31 3

## You Are Here!

- **Parallel Requests**  
Assigned to computer  
e.g., Search "Katz"
- **Parallel Threads**  
Assigned to core  
e.g., Lookup, Ads
- **Parallel Instructions**  
>1 instruction @ one time  
e.g., 5 pipelined instructions
- **Parallel Data**  
>1 data item @ one time  
e.g., Add of 4 pairs of words
- **Hardware descriptions**  
All gates functioning in parallel at same time

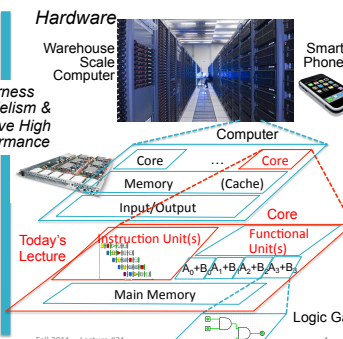
**Software**

**Hardware**

Warehouse Scale Computer

Smart Phone

*Harness Parallelism & Achieve High Performance*



Computer

Core ... Core

Memory (Cache)

Input/Output

Core

Today's Lecture

Instruction Unit(s)

Functional Unit(s)

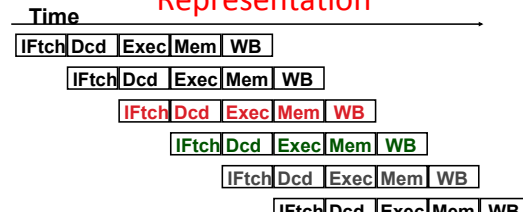
Main Memory

Logic Gate

11/7/11 Fall 2011 - Lecture #31 4

## Pipelined Execution Representation

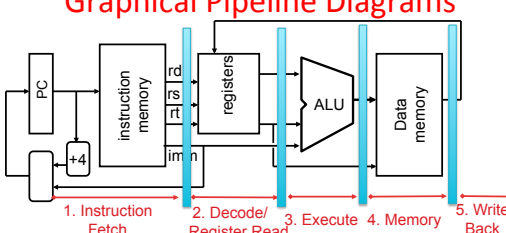
Time →




- Every instruction must take same number of steps, also called pipeline "stages", so some will go idle sometimes

11/7/11 Fall 2011 - Lecture #31 5

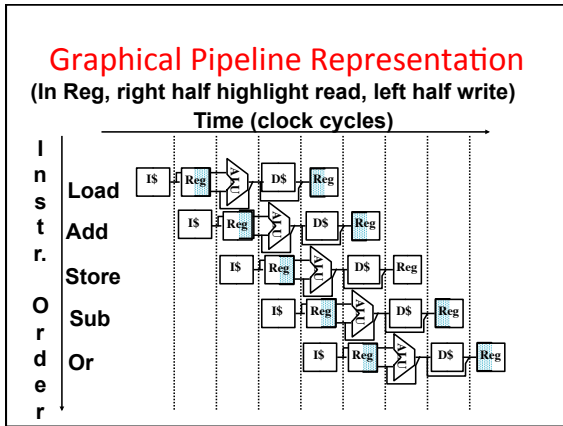
## Graphical Pipeline Diagrams



- Use datapath figure below to represent pipeline



11/7/11 Fall 2011 - Lecture #31 6

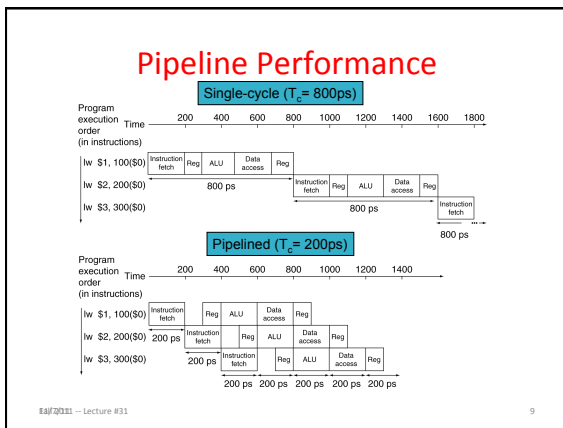


### Pipeline Performance

- Assume time for stages is
  - 100ps for register read or write
  - 200ps for other stages
- What is pipelined clock rate?
  - Compare pipelined datapath with single-cycle datapath

Instr	Instr fetch	Register read	ALU op	Memory access	Register write	Total time
lw	200ps	100 ps	200ps	200ps	100 ps	800ps
sw	200ps	100 ps	200ps	200ps		700ps
R-format	200ps	100 ps	200ps		100 ps	600ps
beq	200ps	100 ps	200ps			500ps

EE/20211 - Lecture #31 8



### Pipeline Speedup

- If all stages are balanced
  - i.e., all take the same time
- Time between instructions<sub>pipelined</sub> =  $\frac{\text{Time between instructions}_{\text{nonpipelined}}}{\text{Number of stages}}$
- If not balanced, speedup is less
- Speedup due to increased throughput
  - Latency (time for each instruction) does not decrease

EE/20211 - Lecture #31 10

### Hazards

Situations that prevent starting the next logical instruction in the next clock cycle

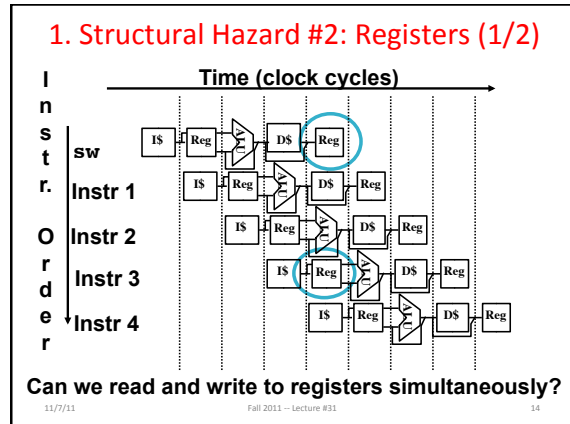
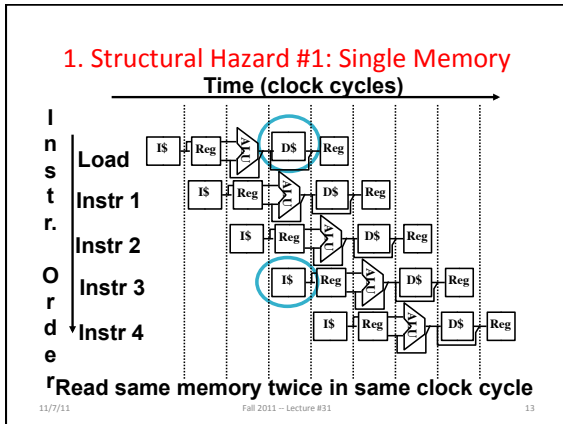
- Structural hazards
  - Required resource is busy (e.g., roommate studying)
- Data hazard
  - Need to wait for previous instruction to complete its data read/write (e.g., pair of socks in different loads)
- Control hazard
  - Deciding on control action depends on previous instruction (e.g., how much detergent based on how clean prior load turns out)

11/7/11 Fall 2011 - Lecture #31 11

### 1. Structural Hazards

- Conflict for use of a resource
- In MIPS pipeline with a single memory
  - Load/Store requires memory access for data
  - Instruction fetch would have to *stall* for that cycle
    - Causes a pipeline "bubble"
- Hence, pipelined datapaths require separate instruction/data memories
  - In reality, provide separate L1 I\$ and L1 D\$

11/7/11 Fall 2011 - Lecture #31 12



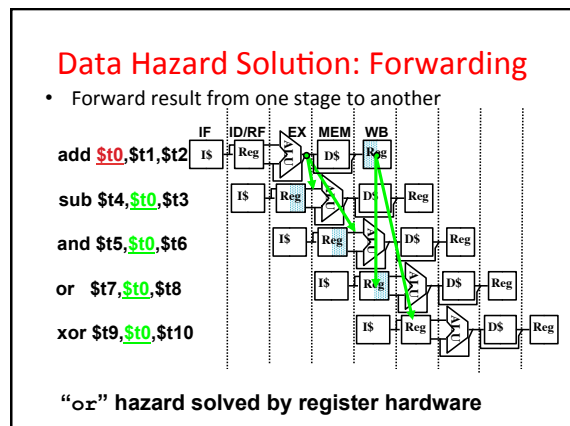
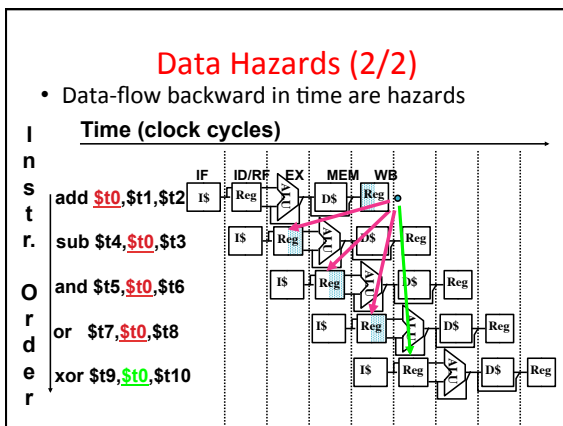
- ### 1. Structural Hazard #2: Registers (2/2)
- Two different solutions have been used:
    - RegFile access is *VERY* fast: takes less than half the time of ALU stage
      - Write to Registers during first half of each clock cycle
      - Read from Registers during second half of each clock cycle
    - Build RegFile with independent read and write ports
  - Result: can perform Read and Write during same clock cycle**
- 11/7/11 Fall 2011 - Lecture #31 15

### Data Hazards (1/2)

Consider the following sequence of instructions

```

add $t0, $t1, $t2
sub $t4, $t0, $t3
and $t5, $t0, $t6
or $t7, $t0, $t8
xor $t9, $t0, $t10
    
```



### Data Hazard: Load/Use (1/4)

- Dataflow backwards in time are hazards

**lw \$t0, 0(\$t1)**  
**sub \$t3, \$t0, \$t2**

- Can't solve all cases with forwarding
- Must stall instruction dependent on load, then forward (more hardware)

### Data Hazard: Load/Use (2/4)

Hardware stalls pipeline (Called "interlock")

**lw \$t0, 0(\$t1)**  
**sub \$t3, \$t0, \$t2**  
**and \$t5, \$t0, \$t4**  
**or \$t7, \$t0, \$t6**

Not in MIPS: (MIPS = Microprocessor without Interlocked Pipeline Stages)

### Data Hazard: Load/Use (3/4)

- Instruction slot after a load is called "load delay slot"
- If that instruction uses the result of the load, then the hardware interlock will stall it for one cycle.
- Alternative: If the compiler puts an unrelated instruction in that slot, then no stall
- Letting the hardware stall the instruction in the delay slot is equivalent to putting a nop in the slot (except the latter uses more code space)

### Data Hazard: Load/Use (4/4)

- Stall is equivalent to nop

**lw \$t0, 0(\$t1)**  
**nop**  
**sub \$t3, \$t0, \$t2**  
**and \$t5, \$t0, \$t4**  
**or \$t7, \$t0, \$t6**

### Pipelining and ISA Design

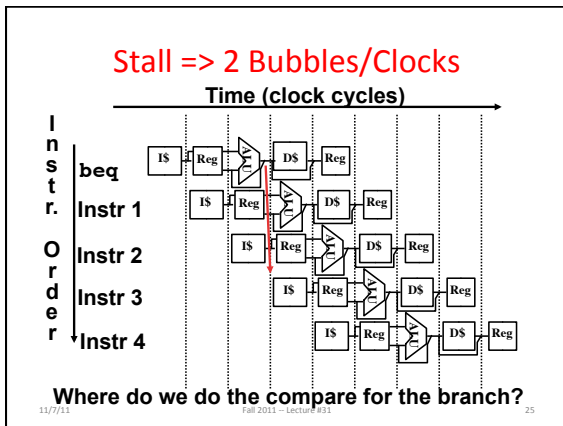
- MIPS Instruction Set designed for pipelining
- All instructions are 32-bits
  - Easier to fetch and decode in one cycle
  - x86: 1- to 17-byte instructions (x86 HW actually translates to internal RISC instructions!)
- Few and regular instruction formats, 2 source register fields always in same place
  - Can decode and read registers in one step
- Memory operands only in Loads and Stores
  - Can calculate address 3<sup>rd</sup> stage, access memory 4<sup>th</sup> stage
- Alignment of memory operands
  - Memory access takes only one cycle

11/7/11 Fall 2011 – Lecture #31 23

### 3. Control Hazards

- Branch determines flow of control
  - Fetching next instruction depends on branch outcome
  - Pipeline can't always fetch correct instruction
    - Still working on ID stage of branch
- BEQ, BNE in MIPS pipeline
- Simple solution Option 1: **Stall** on every branch until have new PC value
  - Would add 2 bubbles/clock cycles for every Branch! (~ 20% of instructions executed)

11/7/11 Fall 2011 – Lecture #31 24



Until next time ...

**The BIG Picture**

- Pipelining improves performance by increasing instruction throughput: exploits ILP
  - Executes multiple instructions in parallel
  - Each instruction has the same latency
- Subject to hazards
  - Structure, data, control
- Stalls reduce performance
  - But are required to get correct results
- Compiler can arrange code to avoid hazards and stalls
  - Requires knowledge of the pipeline structure

11/7/11

Fall 2011 - Lecture #31

26