

EEC 485 High Performance Arch. (Fall 2007)

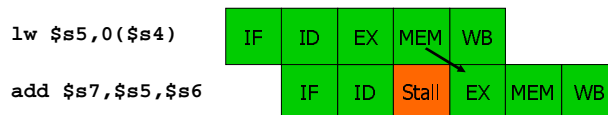
Chapter 6.5 Data Hazards and Stalls

Chansu Yu

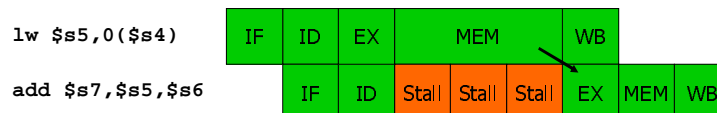
Cleveland State University

Data Hazards: All Considered ???

...but it doesn't eliminate all data hazards:

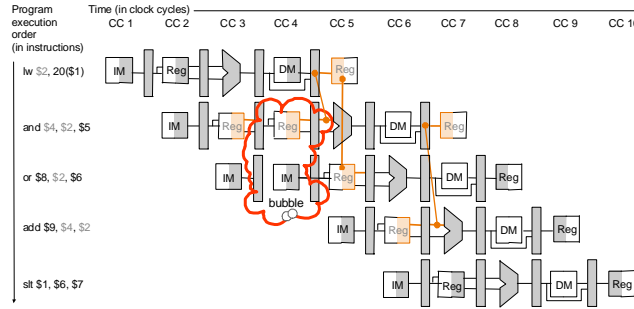


...especially when we remember that memory access is really often much longer than a single cycle:



Data Hazards: Stalling

- Stall the pipeline by keeping an instruction in the same stage



*lw-and
lw-or
At CC5, MEM stage is empty !!!*

Data Hazards: Stalling

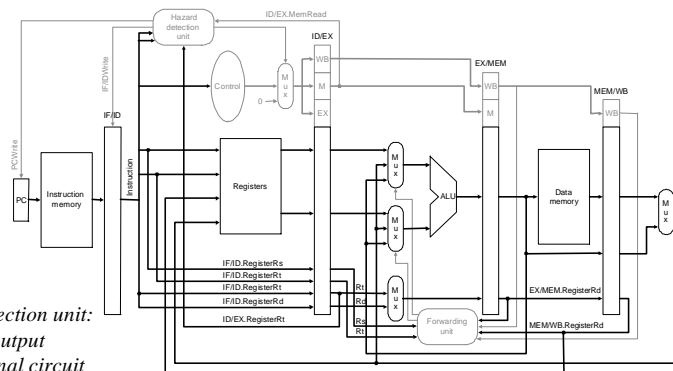
- Stalling detection and control
 - Detects during the ID stage when “lw” instruction is in EX stage
 - The following two instructions are in ID (“and”) and IF (“or”) stages, respectively
 - If detected,
 - Stall the following instruction (in ID stage, “and”) so that it repeats the ID stage again => IF/ID pipeline register should not be changed
 - Stall the second instruction (in IF stage, “or”) so that it repeats the IF stage again => PC should not be changed

Data Hazards: Stalling

- ❑ Hazard detection
 - If (ID/EX.MemRead and (ID/EX.Rt = IF/ID.Rs) or (ID/EX.Rt = IF/ID.Rt)) stall the pipeline
- ❑ Control signals generated from hazard detection unit
 - IF/IDWrite to prevent IF/ID register from changing
 - PCWrite to prevent PC from changing
 - MUX control to delay forwarding control signals (pass “null” signals)

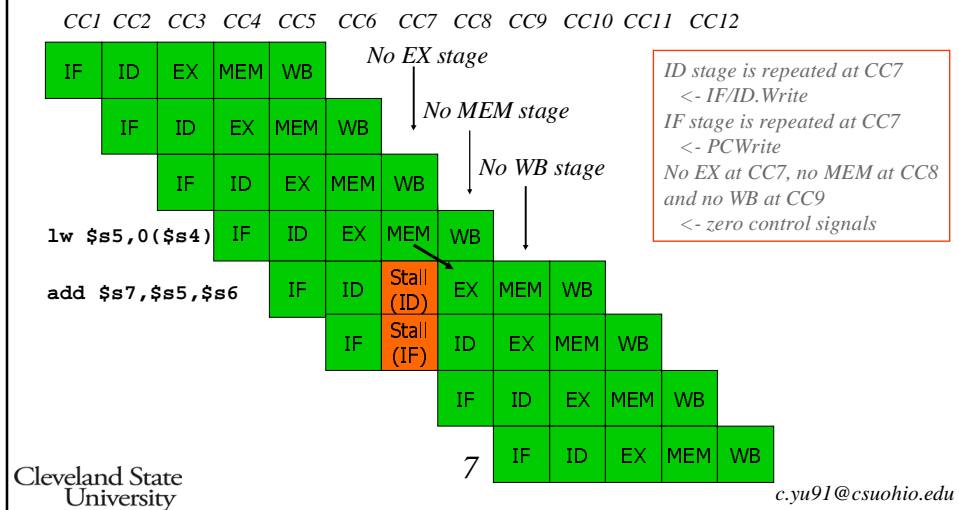
Stalling: Detection Unit

- ❑ Stall by letting an instruction that won't write anything go forward



Hazard detection unit:
4-input, 3-output
combinational circuit

Stalling: What happen in the pipleine?



Stalling: What happen in the pipleine?

- ❑ What does the pipeline do?
 - What does EX stage do at CC7?
 - simple add (ALUop1/ALUop0=00)
 - What does MEM stage do at CC8?
 - No MemRead/MemWrite (zero control signals)
 - What does WB stage do at CC9?
 - No RegWrite (zero control signals)

Data Hazards: Stalling

□ Data forwarding for stalled instructions

- They are needed; otherwise, we need one more stall cycle
 - “and” – forwarding to EX stage from WB stage (MEM/WB pipeline register)
 - “or” – no forwarding is required

- Then, is this forwarding already covered?
 - It seems not because there is no “Rd” for lw instruction
 - But, it is covered because “Rd” actually means destination register among Rd and Rt, selected at EX stage

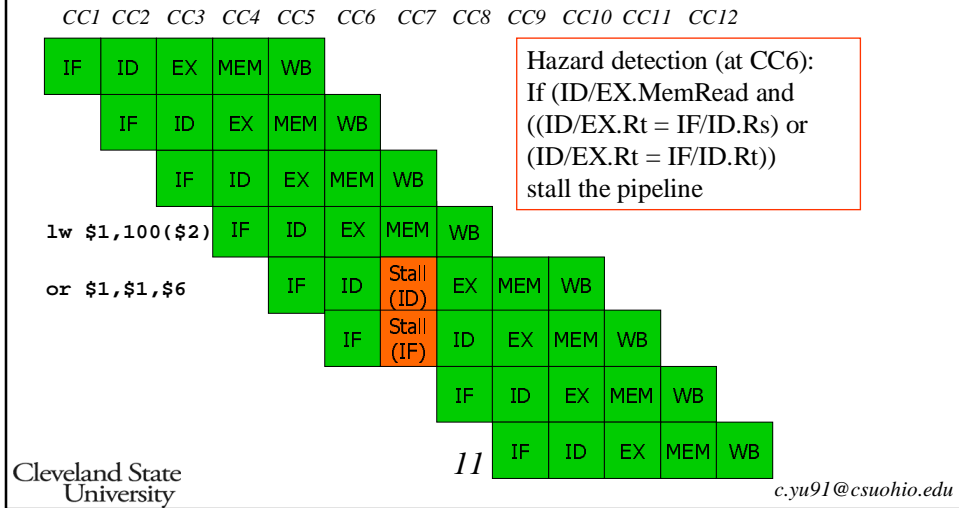
Data Hazards: Stalling

□ Any problems with

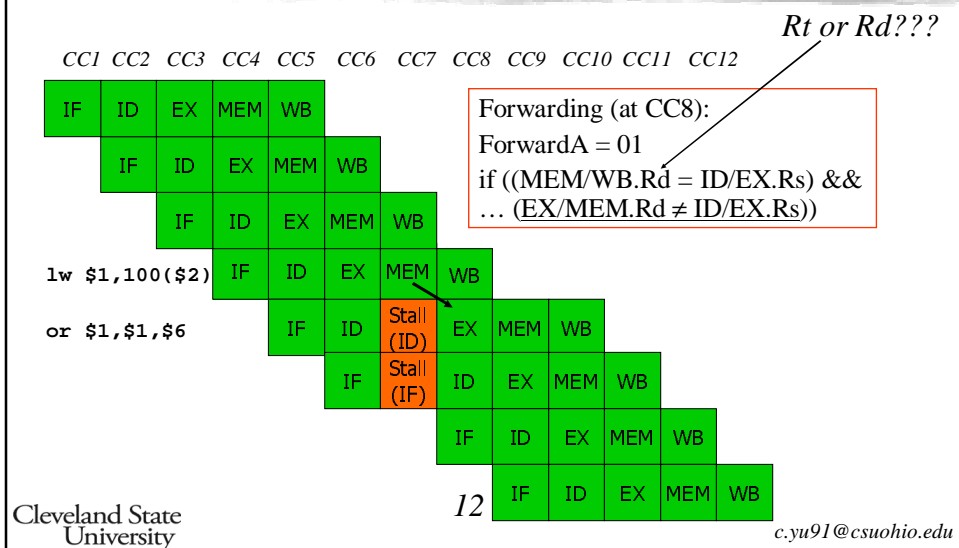
- Example 1
 - lw \$1, 100(\$2)
 - Or \$1, \$1, \$6
 - Sub \$1, \$1, \$1

- Example 2
 - lw \$1, 100(\$2)
 - lw \$1, 200(\$1)
 - Sub \$1, \$1, \$4

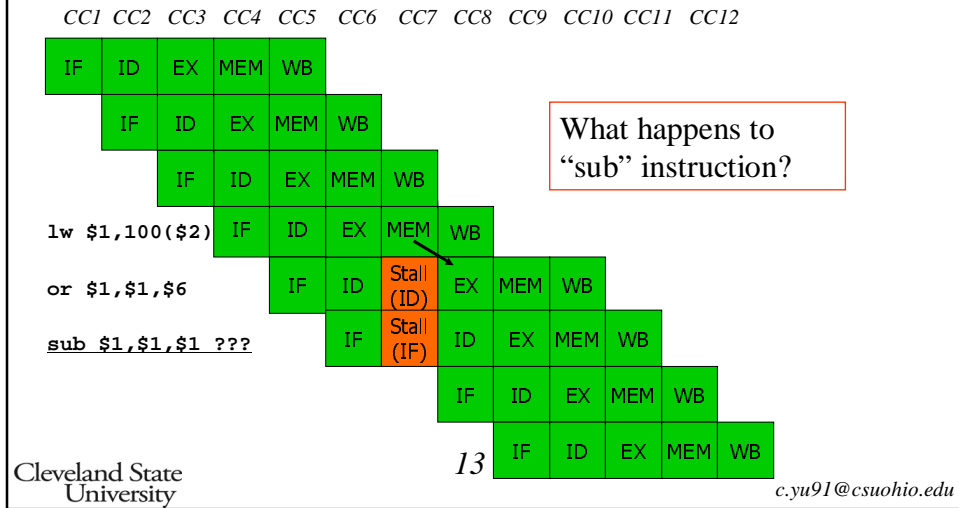
Data hazards: Stalling & Forwarding



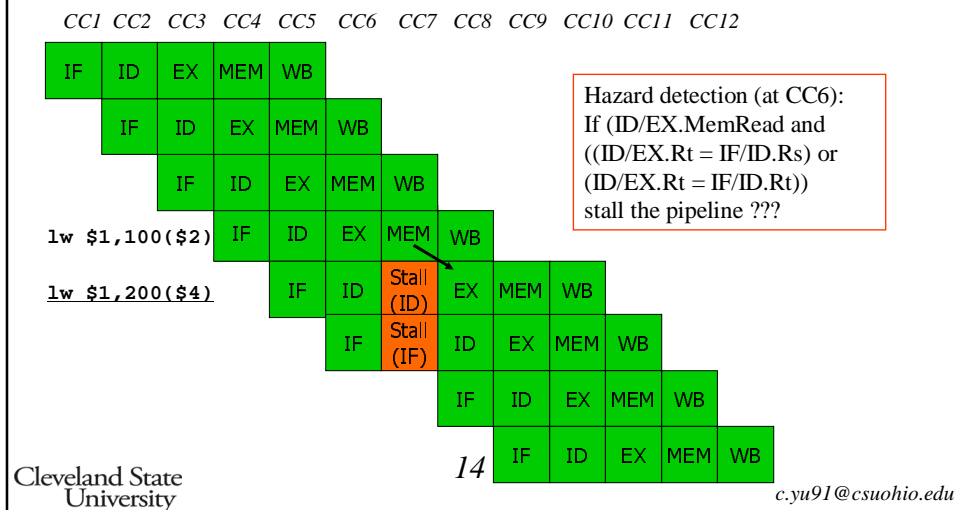
Data hazards: Stalling & Forwarding



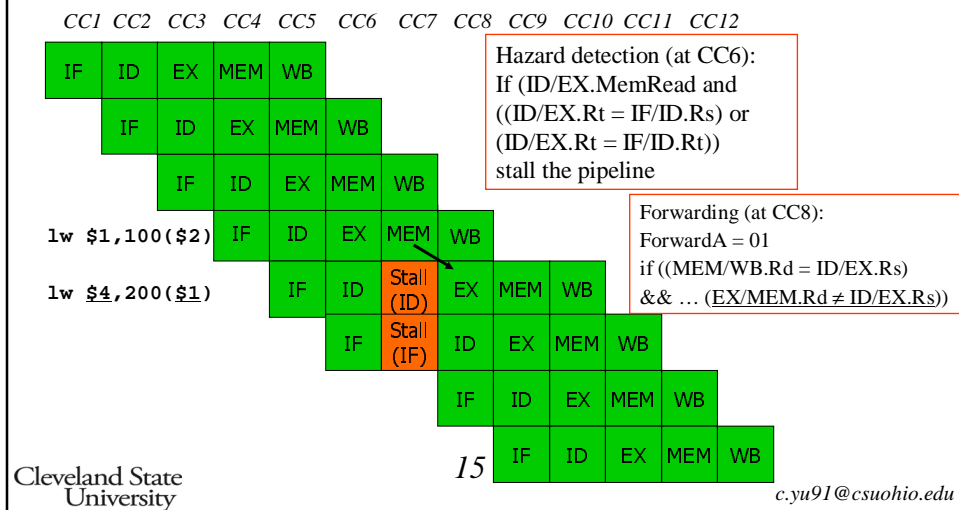
Data hazards: Stalling & Forwarding



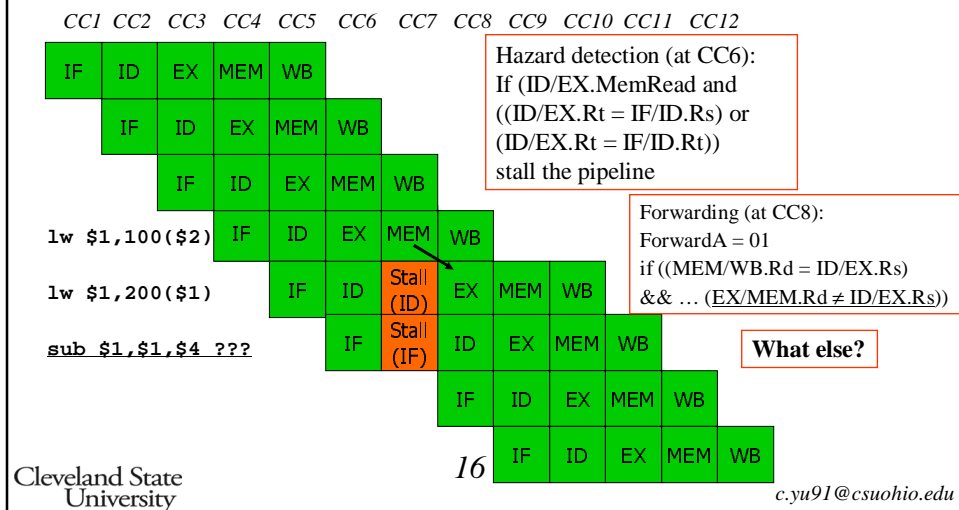
Data hazards: Stalling & Forwarding



Data hazards: Stalling & Forwarding



Data hazards: Stalling & Forwarding



Data Hazards: Stalling

- Consider the following two examples

lw \$1, 100(\$2)

lw \$1, 200(\$4)

lw \$1, 100(\$2)

sw \$1, 200(\$3)

Data Hazards: Static Reordering

Compiler reorders code to avoid stalls.

- I.e., separate dependent instructions.
- Particularly common technique for loads, since forwarding isn't sufficient.

Example:

MIPS:

```
lw $s1,0($s0)
add $s3,$s1,$s2
lw $s5,0($s4)
add $s7,$s5,$s6
```



```
lw $s1,0($s0)
lw $s5,0($s4)
add $s3,$s1,$s2
add $s7,$s5,$s6
```

Data Hazards: Dynamic Scheduling

- ❑ The hardware performs the “scheduling”
 - hardware tries to find instructions to execute
 - out of order execution is possible
 - speculative execution and dynamic branch prediction

- ❑ All modern processors are very complicated
 - DEC Alpha 21264: 9 stage pipeline, 6 instruction issue
 - PowerPC and Pentium: branch history table
 - Compiler technology important