Announcements

• Reminder: Clocks change on Sunday
  – You get 2am twice!

• Midterm logistics
  – Following the path-of-most-success from other faculty
    • Will be on canvas
    • Will be timed
    • Will allow forward progress only
      – N.b. This is how many standardized tests you will encounter (e.g. GRE) work
    • We will give you, in advance, a list of questions and suggested timing
    • Will not use the (IMHO creepy) remote proctoring, etc services
    • Open book, not open Google [but you will not have time to look everything up!]
  – The participation quizzes will be configured this way so you can get comfortable
    • Only difference is participation quizzes will let you see correct answers when you’re done
A brief aside: CSE 141 in the real world

• Literally, in the hour before lecture, during the TockOS project call...

• RISC-V does not support a branch Indexed instruction, which makes switch/case expensive in code size, which may change how our OS works
Is it really that easy?

• What happens when...
  add $3, $10, $11
  lw $8, 1000($3)
  sub $11, $8, $7
The Pipeline in Execution

Iw $8, 1000($3)  add $3, $10, $11  
Execute/ Address Calculation  
Memory Access  
Write Back

lw $8, 1000($3)  
add $3, $10, $11

Address Calculation  
Memory Access  
Write Back
The Pipeline in Execution

sub $11, $8, $7  
`lw $8, 1000($3)`  
`add $3, $10, $11`  
Memory Access  
Write Back
The Pipeline in Execution

add $10, $1, $2  sub $11, $8, $7  lw $8, 1000($3)  add $3, $10, $11  Write Back
When a result is needed in the pipeline before it is available, a **data hazard** occurs. What can we do?

- **sub** $2, $1, $3
- and $12, $2, $5
- or $13, $6, $2
- add $14, $2, $2
- sw $15, 100($2)
Data Hazards

- Data Hazards are caused by **data dependences**
- Not all data dependences result in data hazards
- A data hazard results when there is a data dependence between two instructions that appear too close together in the pipeline

- We will define a data hazard as any data dependence that requires either the software or hardware to take special action to get correct

```
  sub $2, $1, $3
  and $4, $2, $5
  or $8, $2, $6
  add $9, $4, $2
  slt $1, $6, $7
```
Dealing With Data Hazards – What can we do...

• ...in Software?
  – add no-op (nop)

• ...in Hardware?
  – use control signals to introduce a delay
  – "forward"

Data Hazards are caused by *instruction dependences*. For example, the add is data-dependent on the subtract:

- subi $5, $4, #45
- add $8, $5, $2
Dealing with Data Hazards in Software

sub $2, $1, $3

and $12, $2, $5
Dealing with Data Hazards in Software

- sub $2, $1, $3
- nop
- and $12, $2, $5

CC1  CC2  CC3  CC4  CC5  CC6  CC7  CC8

IM → Reg → ALU → DM → Reg

IM → Reg → ALU → DM → Reg

IM → Reg → ALU → DM → Reg

IM → Reg → ALU → DM → Reg

IM → Reg → ALU → DM → Reg

IM → Reg → ALU → DM → Reg

IM → Reg → ALU → DM → Reg
How Many No-ops?

sub $2, $1,$3
and $4, $2,$5
or   $8, $2,$6
add $9, $4,$2
slt  $1, $6,$7
Are No-ops Really Necessary?

sub $2, $1,$3
and $4, $2,$5
or $8, $3,$6
add $9, $2,$8
slt $1, $6,$7
Dealing with Data Hazards in Hardware
Part II - Pipeline Stalls

sub $2, $1, $3
add $14, $2, $2
sw $15, 100($2)

and $12, $2, $5
or $13, $6, $2
Dealing with Data Hazards in Hardware
Part II-Pipeline Stalls

sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
Dealing with Data Hazards in Hardware
Part II - Pipeline Stalls

sub $2, $1, $3

and $12, $2, $5

or $13, $6, $2

add $14, $2, $2

sw $15, 100($2)
Dealing with Data Hazards in Hardware
Part II-Pipeline Stalls

sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
Dealing with Data Hazards in Hardware
Part II - Pipeline Stalls

- sub $2, $1, $3
- and $12, $2, $5
- or $13, $6, $2
- add $14, $2, $2
- sw $15, 100($2)
Dealing with Data Hazards in Hardware
Part II-Pipeline Stalls (alt. View)

<table>
<thead>
<tr>
<th>CC1</th>
<th>CC2</th>
<th>CC3</th>
<th>CC4</th>
<th>CC5</th>
<th>CC6</th>
<th>CC7</th>
<th>CC8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sub $2, $1, $3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

and $12, $2, $5

or $13, $6, $2

add $14, $2, $2

sw $15, 100($2)
Dealing with Data Hazards in Hardware
Part II-Pipeline Stalls (alt. View)

```
sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
```
Dealing with Data Hazards in Hardware
Part II-Pipeline Stalls (alt. View)

sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
Dealing with Data Hazards in Hardware
Part II-Pipeline Stalls (alt. View)

sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
Dealing with Data Hazards in Hardware
Part II-Pipeline Stalls (alt. View)

```
sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
```
Dealing with Data Hazards in Hardware
Part II-Pipeline Stalls (alt. View)

sub $2, $1, $3
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($2)
Dealing with Data Hazards in Hardware
Part II-Pipeline Stalls (alt. View)

sub $2, $1, $3

and $12, $2, $5

or $13, $6, $2

add $14, $2, $2

sw $15, 100($2)
Poll Q: Try it yourself

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

How many bubbles?

A 5
B 6
C 7
D 8
E None of the above
Working this example...

```
sub $2, $1, $3
add $12, $3, $5
or $13, $6, $2
add $14, $12, $2
sw $14, 100($2)
```
Poll Q: How to actually implement this in hardware?

Once you detect the hazard in ID – what must you do to insert the nop and “stall”?

1. **Flush all instructions in the pipeline** (set control signals to 0).
2. Set all control signals going to ID/EX register to zero.
3. Set PCWrite to zero.
4. Set IF/ID register write to zero.

**Selection** | **Changes**
--- | ---
A | 1, 3, 4
B | 1, 2, 3
C | 2, 3, 4
D | 1
E | None of the above
Pipeline Stalls

• To ensure proper pipeline execution in light of register dependences, we must:
  – detect the hazard
  – stall the pipeline
Knowing When to Stall

- 6 types of data hazards
  - two reg reads * 3 reg writes
Knowing When to Stall

- 6 types of data hazards
  - two reg reads * 3 reg writes
The Pipeline

- What comparisons tell us when to stall?