ECE 3401 Lecture 24

Pipeline Design (II)

Outline

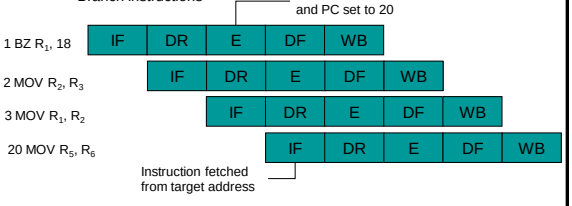
- Pipelined Design
  - Basic 5-stage pipe
    - Speedup of pipelined vs. non-pipelined implementations
  - Pipeline hazards
    - Structural, data, control
- Parallel digital systems

Control Hazard

- For branch or jump instruction, the correct instruction to execute is not known in time (at the start of the IF stage of the instruction after the Branch)
  - Condition not yet determined (for conditional branch instruction)
  - Target instruction address not yet calculated

Control Hazards

- Conflicts that arise from changes in the Program Counter
  - Branch instructions



- IF of 2 instructions following branch happens before you know whether or not to branch and where to branch
- ISA may allow code to run useful instructions
- Can use branch prediction to improve performance

Control Hazard Solutions

- Solution 1: stall
- The instructions after the branch are stalled, until the branch condition is checked and target address is generated

```
Add   |IF|DR|E |DF|W |
Beq      |IF|DR|E |DF|W |
and         |- |- |IF|DR|E |DF|W |
xor            |- |- |IF|DR|E |DF|W |
```

2 (stall) penalty cycles,

Sol 2:

- Perform target address calculation earlier in DR stage

```
Add   |IF|DR|E |DF|W |
Beq      |IF|DR|E |DF|W |
and         |- |IF|DR|E |DF|W |
xor            |- |IF|DR|E |DF|W |
```

Reduced penalty cycles from 2 to 1,

Need a separate branch adder in the DR stage to calculate PC+displacement

Sol. 3a

- Assume branch not taken, fixup if taken

```
Add    |IF|DR|E |DF|W |
Beq       |IF|DR|E |DF|W |
and          |IF|DR|E |DF|W |
xor             |IF|DR|E |DF|W |     if not taken

Add    |IF|DR|E |DF|W |
Beq       |IF|DR|E |DF|W |
and          |IF|-  |-  |-  |-  |     squash this
target instruction  |IF|DR|E |DF|W |     if taken
```

If branch is not taken (as predicted), then no penalty
else 1 penalty cycle.
Assumes branch address calculation in the DR stage

Sol.3b

- Assume branch taken, fixup if not taken

```
Add    |IF|DR|E |DF|W |
Beq       |IF|DR|E |DF|W |
target       |-  |IF|DR|E |DF|W |        if taken
target+1        |IF|DR|E |DF|W |

Add    |IF|DR|E |DF|W |
Beq       |IF|DR|E |DF|W |
and          |-  |IF|DR|E |DF|W |     if not taken
```

Not much benefit, since 1 cycle penalty in either case.
Assumes branch address calculation in the DR stage - also have to select between fall-thru address or target address.

Sol.4

- Delayed branch (ISA change)

```
Add          |IF|DR|E |DF|W |
Beq             |IF|DR|E |DF|W |
(delay slot)       |IF|DR|E |DF|W |
fallthru or target    |IF|DR|E |DF|W |
```

ISA specifies that the branch, if taken, only takes place after a delay of x instructions. (Above, x=1)

Branch adder in DR stage to calculate PC+displ, or PC+4+displ (depending on ISA definition)

Solutions to Control Hazards

- Micro-architecture solution
  - Stall the pipes
  - Calculate target address and condition earlier in pipeline (DR)
  - Assume branch always goes untaken (taken) and fix up pipe if it is actually taken (untaken)
- ISA solution
  - Have delay branches
    - Branch target takes place after n (delay) instructions
    - For n penalty cycles, must have (n-1) stages between IF and the stage where target and condition are determined
  - Have instruction which separate target address calculation and condition generation from actual branching, so these can be executed earlier (e.g., IA-64)
- Branch prediction

Handling Hazards - Summary

- Avoid some hazards “by design”
  - Eliminate DF-IF structural hazard by having separate I-cache and D-cache
  - Eliminate WAR by always fetching operands earlier in pipe (DR)
  - Eliminate WAW by doing all Ws in order (last stage, static) – not always the best micro-architecture decisions though!
  - Delayed branch in ISA to reduce control hazard penalty
- Detect and resolve remaining ones
  - Stall or forward (if possible)

Pipelining Cautions

- Superpipelining can cause long latencies
  - A large number of pipeline stages
  - High frequency (GHz)
    - Clock limited by slowest stage in the pipe
- Long pipes can also cause more stalls
- Dependencies can be tolerated as long as there is work to overlap the dependency

Parallelism in CPUs

- Instruction Level Parallelism
  - Superscalar
    - Multiple functional units in a CPU support multiple instructions fetch and issue simultaneously
  - VLIW: single instruction, but multiple executions

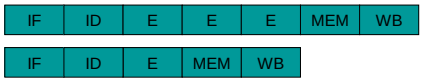
Pipelined CPUs

mult r1, r2, r3
add r4, r5, r6

3 cycle multiply execute stage

Superscalar CPUs

mult r1, r2, r3
add r4, r5, r6



3 cycle multiply execute stage, with multiple ALUs

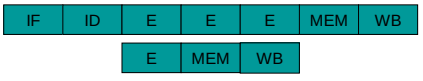
4-Way Superscalar

- Today's microprocessors are typically 4-way superscalar

Cycle	1	2	3	4	5	6	7	8
	IF	DR	E	DF	W			
	IF	DR	E	DF	W			
	IF	DR	E	DF	W			
	IF	DR	E	DF	W			
		IF	DR	E	DF	W		
		IF	DR	E	DF	W		
		IF	DR	E	DF	W		
		IF	DR	E	DF	W		
			IF	DR	E	DF	W	
			IF	DR	E	DF	W	
			IF	DR	E	DF	W	
			IF	DR	E	DF	W	
				IF	DR	E	DF	W
				IF	DR	E	DF	W
				IF	DR	E	DF	W
				IF	DR	E	DF	W

VLIW CPUs

mult r1, r2, r3; add r4, r5, r6



VLIW: very large instruction word
Single instruction fetched, multiple operations executed at the same time

4-way VLIW

Cycle	1	2	3	4	5	6	7	8
	IF	DR	E	DF	W			
			E		W			
			E		W			
			E		W			
		IF	DR	E	DF	W		
				E		W		
				E		W		
				E		W		
			IF	DR	E	DF	W	
					E		W	
					E		W	
					E		W	
				IF	DR	E	DF	W
						E		W
						E		W
						E		W