3810 Review Session

Spring 2020
Unpipelined processor
CPI:
Clock speed:
Throughput:

Pipelined processor
CPI:
Clock speed:
Throughput:

Circuit Assumptions
Length of full circuit:
Length of each stage:
No hazards

Pipeline Performance
No Bypassing

Point of production: always RW middle
Point of consumption: always D/R middle

Bypassing

Point of production:
add, sub, etc.: end of ALU
lw: end of DM

Point of consumption:
add, sub, lw: start of ALU
sw $1, 8($2): start of ALU for $2, start of DM for $1

* PoP
I1 add: IF DR AL DM RW
I2 add: IF DR DR DR AL DM RW
* PoC

* PoP
I1 add: IF DR AL DM RW
I2 add: IF DR AL DM RW
* PoC
Assumptions

100 instructions
20 branches
14 Not-Taken, 6 Taken
Branch resolved in 6th cycle (penalty of 5)

Approach 1: Panic and wait

Approach 2: Fetch-next-instr

Approach 3: Branch Delay Slot
Option A: always useful
Option B: useful when the branch goes along common fork
Option C: useful when the branch goes along uncommon fork
Option D: no-op, always non-useful

Approach 4: Branch predictor
Accuracy of 90%

Control Hazards
Out of Order Processor

Branch prediction and instr fetch

R1 ← R1+R2
R2 ← R1+R3
BEQZ R2
R3 ← R1+R2
R1 ← R3+R2

Instr Fetch Queue

Decode & Rename

T1 ← R1+R2
T2 ← T1+R3
BEQZ T2
T4 ← T1+T2
T5 ← T4+T2

Issue Queue (IQ)

ALU

Results written to ROB and tags broadcast to IQ

Register File R1-R32

Reorder Buffer (ROB)

Instr 1
Instr 2
Instr 3
Instr 4
Instr 5
Instr 6
T1
T2
T3
T4
T5
T6
Assumptions

1000 instructions, 1000 cycles, no stalls with L1 hits
# loads/stores:
% of loads/stores that show up at L2:
% of loads/stores that show up at L3:
% of loads/stores that show up at mem:
L2 acc = 10 cyc, L3 acc = 25 cyc, mem acc = 200 cyc
Assumptions

512KB cache, 8-way set-associative, 64-byte blocks, 32-bit addresses

Data array size = #sets x #ways x blocksize
Tag array size = #sets x #ways x tagsize
Offset bits = log(blocksize)
Index bits = log(#sets)
Tag bits + index bits + offset bits = addresswidth
Assumptions

16 sets, 1 way, 32-byte blocks

Access pattern: 4 40 400 480 512 520 1032 1540
  Set #:
  Block:
6. Consider a 4-processor multiprocessor connected with a shared bus that has the following properties: (i) centralized shared memory accessible with the bus, (ii) snooping-based MSI cache coherence protocol, (iii) write-invalidate policy. Also assume that the caches have a writeback policy. Initially, the caches all have invalid data. The processors issue the following three requests, one after the other. Similar to slide 4 of lecture 25, fill in the following table to indicate what happens for every request. Also indicate if/when memory writeback is performed. \(\text{(12 points)}\)

<table>
<thead>
<tr>
<th>Request</th>
<th>Cache Hit/Miss</th>
<th>Request on bus</th>
<th>Who responds</th>
<th>State Cache 1</th>
<th>State Cache 2</th>
<th>State Cache 3</th>
<th>State Cache 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3: Read X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3: Write X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P2: Write X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(a) P3: Read X
(b) P3: Write X
(c) P2: Write X
Virtual Memory
Synchronization, GPUs