Quiz 1

- Average: 56 points (70%)
- Stddev: 12.4 points (15.5%)
Agenda

- Quiz 1
- Cache Review
- A dab of VM
- Lab 2
Quiz 1: q1– strcpy

• copy string from one memory location to another
• start at M[rd] ← M[rs], and increment both pointers until M[rs] == 0
Quiz 1: q1– strcpy

• Some points:
  – ldIR (must be zero for all instructions)
  – spec states rd,rs will hold address of null character once finished
  – ldMA must be zero when accessing Memory
Microcoded MIPS
strcpy - pseudo-psuedo code

loop:

    temp <- M[rs]
    M[rd] <- temp
    if (temp==0) done;
    rs += 4
    rd += 4
    jmp loop

strcpy:

    MA <- rs
    B <- Mem
    MA <- rd
    Mem <- B; Z Fetch0
    A <- rs
    rs <- A+4
    A <- rd
    rd <- A+4; jmp strcpy
strncpy – pseudo code

strcpy:
0  MA <- rs
1  B  <- Mem
2  MA <- rd
3  Mem <- B; Z Fetch0
4  A  <- rs
5  rs <- A+4
6  A  <- rd
7  rd <- A+4; jmp strcpy0

strcpy_optimized:
MA <- rs; A <- rs
B  <- Mem
MA <- rd
Mem <- B; Z Fetch0
rs <- A+4
A  <- rd
rd <- A+4; jmp strcpy0

optimization

Friday, February 18, 2011
strcpy – final solution

0: MA <- rs; A <- rs
1: B <- Mem
2: MA <- rd
3: Mem <- B; Z Fetch0
4: rs <- A+4
5: A <- rd
6: rd <- A+4; jmp strcpy0

| State    | PseudoCode                  | ldIR | Reg Sel | Reg Wr | en Reg | ldA | ldB | ALUOp | en ALU | Id MA | Mem Wr | en Mem | Ex Sel | en Imm | µBr | Next State |
|----------|-----------------------------|------|---------|--------|--------|-----|-----|-------|--------|-------|--------|--------|--------|------|------------|
a better microarchitecture

0:  MA <- rs; B <- rs
1:  A  <- Mem
2:  MA <- rd
3:  Mem<- A; Z Fetch0
4:  rs <- B+4; MA <- B+4; B <- B+4
5:  A  <- rd
6:  rd <- A+4; jmp strcpy1

Modification:
  add INC_B_4
  (keep rs in register B)
saves a cycle in the dynamic loop

-OR-

0:  MA <- rs; B <- rs
1:  C  <- Mem
2:  MA <- rd; A <- rd
3:  Mem<- C; Z Fetch0
4:  rs <- B+4; MA <- B+4; B <- B+4
5:  rd <- A+4; jmp strcpy1

Modification:
  add ALU source C
  add COPY_C
  add INC_B_4
  (keep rd in A, rs in B, temp in C)
saves one static uop
saves two cycles in the dynamic loop

of course, you can do better if you don’t finish with Rs,Rd pointing to the null terminator

Friday, February 18, 2011
How to calculate CPI?

\[
\text{CPI} = \frac{\# \text{ of cycles to execute } N \text{ instructions}}{N \text{ instructions}}
\]

- i.e., CPI is calculated over an entire program

Pipelined machine (5-stage)

5 instructions, 5 cycles, CPI=1
How to calculate CPI? – q2.A.iii

CPI = \frac{\text{# of cycles to execute N instructions}}{N \text{ instructions}}

If 20% of all instructions are loads, and 50% of these loads are followed by dependent instructions, what is the CPI of the typical fully bypassed 5-stage pipeline?

over N instructions... 0.1*N cause an extra cycle stall

CPI = \frac{N + 0.1N}{N} = \frac{1+0.1}{1}
= 1.1

Another way....

CPI = 1 + 1(\% \text{ of one cycle stalls}) + 2(\% \text{ of two cycles stalls}) + ...
CPI = 1 + 1(0.1) = 1.1
What is $\text{is\_speculated}_{\text{dec}} = ...$

$LW \ R1, 0(R2)$
$ADD \ R2, R1, R0$
Question 2. Part B

What is is_speculated?_{dec} =

\[
(C_{\text{OPC}_{\text{EXE}}} == \text{LW}) \\
& \& [(C_{\text{RD}_{\text{EXE}}}==C_{\text{RS1}_{\text{DEC}}} && C_{\text{RE1}_{\text{DEC}}})  \\
| (C_{\text{RD}_{\text{EXE}}}==C_{\text{RS2}_{\text{DEC}}} && C_{\text{RE2}_{\text{DEC}}})] \\
& \& (C_{\text{RD}_{\text{EXE}}} != 0)
\]

LW R1, 0(R2)  \\
ADD R2, R1, R0

Friday, February 18, 2011
I1: LW  R1, 0(R2)  
I2: ADD R3, R1, R0  
I3: SUB R5, R0, R0  
I4: XOR R6, R0, R0  
I5: AND R7, R0, R0  
I6: OR  R8, R0, R0
Question 2. Part C

I1: LW  **R1**, 0(R2)
I2: ADD  **R3**, **R1**, R0
I3: SUB  R5, R0, R0
I4: XOR R6, R0, R0
I5: AND R7, R0, R0
I6: OR  **R8**, R0, R0

(regular fully bypassed 5-stage pipeline)

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
<th>t8</th>
<th>t9</th>
<th>t10</th>
<th>t11</th>
<th>t12</th>
<th>t13</th>
<th>t14</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td></td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td></td>
<td>F</td>
<td>D</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td></td>
<td></td>
<td>F</td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XOR</td>
<td></td>
<td></td>
<td></td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND</td>
<td></td>
<td></td>
<td></td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Friday, February 18, 2011
Question 2. Part C

I1: LW  R1, 0(R2)
I2: ADD R3, R1, R0
I3: SUB R5, R0, R0
I4: XOR R6, R0, R0
I5: AND R7, R0, R0
I6: OR  R8, R0, R0

(With load speculation, correctly speculated)

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
<th>t8</th>
<th>t9</th>
<th>t10</th>
<th>t11</th>
<th>t12</th>
<th>t13</th>
<th>t14</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XOR</td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND</td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OR</td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Friday, February 18, 2011
Question 2. Part C

I1: LW, 0(R2)
I2: ADD R3, R1, R0
I3: SUB R5, R0, R0
I4: XOR R6, R0, R0
I5: AND R7, R0, R0
I6: OR R8, R0, R0

(With load speculation, misspeculated)

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
<th>t5</th>
<th>t6</th>
<th>t7</th>
<th>t8</th>
<th>t9</th>
<th>t10</th>
<th>t11</th>
<th>t12</th>
<th>t13</th>
<th>t14</th>
</tr>
</thead>
<tbody>
<tr>
<td>LW</td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADD</td>
<td>F</td>
<td>D</td>
<td>X</td>
<td></td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>F</td>
<td>D</td>
<td></td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XOR</td>
<td>F</td>
<td></td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND</td>
<td></td>
<td>F</td>
<td></td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OR</td>
<td></td>
<td></td>
<td></td>
<td>F</td>
<td>D</td>
<td>X</td>
<td>M</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

kill 4 bubbles (4 stages got killed)
How to calculate CPI? – q2.D.i

\[
\text{CPI} = \frac{\text{# of cycles to execute N instructions}}{N \text{ instructions}}
\]

Assume that 100% of all load values that are speculated are correctly speculated (absolute best case). If, for a given program, 20% of all instructions are loads, and 50% of those loads are immediately followed by a dependent instruction, what is the CPI?

CPI = 1

(since the load-uses are being perfectly predicted, no stalls occur)

Another way....

CPI = 1 + 1(% of one cycle stalls) + 2(% of two cycles stalls) + ...
CPI = 1 + 1(0.0) = 1
How to calculate CPI? – q2.D.ii

CPI = \frac{\text{# of cycles to execute N instructions}}{N \text{ instructions}}

Let us be more realistic. Assume for a given program that 20% of all instructions are loads, and 25% of all loads in the program return a load value of zero.

Also assume that 50% of all loads are followed immediately by a dependent instruction:

15% of all instructions are loads that return a non-zero value but...
7.5% of all instructions cause a misspeculated load-use.

For a program of N instructions,
CPI = \frac{(N+0.075NX)}{N}, \text{ where } X \text{ is the number of bubbles that get inserted (4)}

so CPI = \frac{(N+0.3N)}{N} \quad \text{or} \quad CPI = 1 + 4(7.5\%) = 1.3
How to calculate CPI? – q2.D.ii

\[
\text{CPI} = \frac{\text{# of cycles to execute N instructions}}{\text{N instructions}}
\]

What fraction of loads must be correctly speculated for the new datapath to be worthwhile?

\[
\text{CPI} = 1.1 = 1 + 4(\% \text{ of insts that cause a misspeculation})
\]
\[
\text{CPI} = 1.1 = 1 + 4Y
\]

\[
Y = 2.5\%
\]

Assuming 20\% are loads, 50\% followed by dependent instructions....

10\% can cause a misspeculation

---\> 75\% of all loads must be zero
### Question 3: Iron Law

<table>
<thead>
<tr>
<th>Instructions / Program</th>
<th>Cycles / Instruction</th>
<th>Seconds / Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>decrease (64-bit ops won't require being synthesizing from multiple 32-bit versions)</td>
<td>unchanged (widening the datapath doesn't change the logic in the pipeline)</td>
<td>increases (bigger registers, wider datapaths, and larger ALUs will all work to increase Secs/Cycle)</td>
</tr>
<tr>
<td>-1 for “no change” unless the answer was qualified by stating no 64-bit arithmetic occurs in the program</td>
<td>- OR - increases more cache misses occur (because ints and address pointers are now 64bits which decrease the number of variables that will fit inside the caches)</td>
<td></td>
</tr>
</tbody>
</table>

Change the ISA from 32-bits to 64-bits (i.e., all registers in the Register File are now 64-bits wide and the ALU performs 64-bit operations).
<table>
<thead>
<tr>
<th><strong>Question 3: Iron Law</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modifying the ISA (and thus the micro-architecture) to use hardware interlocking instead of software interlocking for both branch delay slots and load-use delay slots</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
**Question 3: Iron Law**

<table>
<thead>
<tr>
<th>Instructions / Program</th>
<th>Cycles / Instruction</th>
<th>Seconds / Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Removing a complex instruction from the hardware implementation, and instead execute it by throwing an illegal opcode trap and letting the exception handler execute the instruction in software</td>
<td>unchanged (not ISA visible)</td>
<td>increase (a lot of software instructions will have to be executed to perform a single complex instruction)</td>
</tr>
</tbody>
</table>

“An Operating System call will appear to be a single instruction that takes many, many cycles to execute”
Administrivia

Newsgroup
ucb.class.cs152
news.csua.berkeley.edu

Can use EECS web UI:
http://inst.eecs.berkeley.edu/webnews

• http://www.piazzza.com/
  – you should’ve gotten an email about this
  – I will post notes and answer questions (you can post answers too!)
  – suggest a daily digest setting
Lab 2

- Is out now!
- Due March 2nd

- Simulate memory hierarchy using g-cache
- Get started now!
  - simulations will take much longer
• pset 2 will go out later today
Direct Mapped Cache

32-k-b  k  b

2^k entries
4-way Set-associative Cache

What if, for legacy reasons, the tag size was fixed.... but you got more transistors for the next version of your cache.

What would you do?
How to handle a TLB fill

Hierarchical Page Table

Virtual Address
31 22 21 12 11 0
p1 p2 offset
10-bit L1 index L2 index

Root of the Current Page Table
(Processor Register)

Level 1 Page Table
p1

Level 2 Page Tables
offset

Physical Memory

Data Pages

page in primary memory
page in secondary memory
PTE of a nonexistent page

February 16, 2011
CS152, Spring 2011
Example Machine: MIPS R10k

- 0.35 micron process
- $16.6 \times 17.9$ mm$^2$
- 298 mm$^2$
- 6.8 million transistors
  - 4.4 million cache
  - 2.4 million logic
- Full-custom design for datapaths and control logic
- Semi-custom design for less critical control logic

Jan 1996 (150-200 MHz) $3,000$ for 200 MHz
1997 -> 250nm @ 250 MHz


Friday, February 18, 2011
MIPS R10k: Memory Hierarchy

Main Memory

4 Instruction words

virtual indexed
physical tagged

Inst Fetch Virtual Address (PC)

• Study the effect of virtual vs. physical tagging/indexing
• Analyze differences in memory traffic with different cache replacement policies
• Analyze differences in memory traffic for different write-back schemes
• design a memory hierarchy to made a specific benchmark go fast
• tune a benchmark to go fast given a memory hierarchy
Questions?