CS 152
Computer Architecture and Engineering

Lecture 1 – The MIPS ISA

2005-8-30
John Lazzaro
(www.cs.berkeley.edu/~lazzaro)

TAs: David Marquardt and Udam Saini

www-inst.eecs.berkeley.edu/~cs152/
IBM Power 5 “die photo”: a die is an unpackaged part
Will we be fabricate CPU dies?

Back when I was taking classes (1984 @ Caltech) our project course did fab chips.

Intel XScale 80200: used in earlier HP PocketPCs
FPGAs: Field Programmable Gate Arrays

Xilinx
Virtex E

43,200 “parts” + 655,000 RAM bits

Write Verilog to “wire” parts.
Calinx: 150/152 boards in 125 Cory

- Download CPU machine code using TFTP
- Program Xilinx via PC
- DRAM
- Xilinx Virtex E FPGA
CS152: Approximate grading weights

Lab 4: 25%
Lab 3: 15%
Lab 2: 8%
Lab 1: 2%

Subject to fine tuning ...

Lab work
Evals
Midterm exams
HW
Grading: Peer and staff evaluations

Peer evals:
Teammates grade each other after Labs 2, 3, 4. Rewards good “team players”

Staff evals:
TAs are your “managers”
They observe how well you work on the team.

Evals
Midterm exams
Lab work
HW
Exams: Two mid-terms, no final ...

MT1: Oct 4th. About 10% of final grade.
MT2: Dec 6th. About 20% of final grade.

3 hours, early evening, no calculators or electronic devices.

In-class review session before each mid-term.
Homeworks: To prepare for midterms

- Midterm exams
- Lab work
- Evals

Homework due at mid-term review session.

Based on last year’s midterm.

Graded on effort, not correctness.

NO late homeworks accepted.

May discuss HWs with other students, but work handed in must be your own.
152: Semester Calendar

See the class webpage for the most up to date version! Changes daily!

www-inst.eecs.berkeley.edu/~cs152/
Week One: You are here.

Download Lab 1, done individually. It is a refresher lab for MIPS assembly language programming. Use 125 Cory machines, or use RDF (see web page “Resources”)

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Reading</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 8/30</td>
<td>The MIPS ISA</td>
<td>PDF</td>
<td>Ch 2, 3.1-3, 3.8</td>
</tr>
<tr>
<td>W 8/31</td>
<td>Single-Cycle Datapaths</td>
<td>PDF</td>
<td>5.1-4, 5.8, Appendix B</td>
</tr>
<tr>
<td>Th 9/1</td>
<td></td>
<td></td>
<td>HW 1 available</td>
</tr>
<tr>
<td></td>
<td>(due at Midterm I review session)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F 9/2</td>
<td></td>
<td></td>
<td>Teams Meet the TA, 12-2 PM, 125 Cory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>150 Lab Lecture</strong>, 2-3 PM, 125 Cory</td>
</tr>
<tr>
<td>Sa 9/3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Su 9/4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M 9/5</td>
<td>(Labor Day Holiday)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T 9/6</td>
<td>Single-Cycle Wrap-Up</td>
<td>PDF</td>
<td>Lab 1: Final Report Due, 1:59 PM, via the submit program</td>
</tr>
</tbody>
</table>

Lab due Tuesday.
Week One: Preparing for team labs

Lab 2, the first team lab, is up. I will be mailing out tentative team assignments soon. On Friday at noon, meet in 125 Cory w/ TAs, to finalize teams.

<table>
<thead>
<tr>
<th>Date</th>
<th>Activity</th>
<th>Format</th>
<th>Pages</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 8/30</td>
<td>The MIPS ISA</td>
<td>PDF</td>
<td>Ch 2, 3.1-3, 3.8</td>
<td>Start on Lab 1; Preview Lab 2</td>
</tr>
<tr>
<td>W 8/31</td>
<td>Single-Cycle Datapaths</td>
<td>PDF</td>
<td>5.1-4, 5.8, Appendix B</td>
<td>HW 1 available (due at Midterm I review session)</td>
</tr>
<tr>
<td>Th 9/1</td>
<td></td>
<td></td>
<td></td>
<td>Teams Meet the TAs 12-2 PM, 125 Cory 150 Lab Lecture, 2-3 PM, 125 Cory</td>
</tr>
<tr>
<td>F 9/2</td>
<td>Single-Cycle Wrap-Up</td>
<td>PDF</td>
<td></td>
<td>Lab 1: Final Report Due, 11:59 PM, via the submit program</td>
</tr>
<tr>
<td>Sa 9/3</td>
<td>(Labor Day Holiday)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Su 9/4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TAs: David Marquardt and Udam Saini. Undergrads, “A team” CS152 alumni. See web page for contact info.
**Week One: 150 boot-camp begins.**

Students w/o CAD skills **SHOULD** attend first 4 150 Lectures in order to learn Xilinx CAD and Verilog (in person, or via video), and **SHOULD** do the first 4 150 labs.

<table>
<thead>
<tr>
<th>Date</th>
<th>Activity</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>T 8/30</td>
<td>The MIPS ISA</td>
<td>Start on <a href="#">Lab 1</a>; Preview <a href="#">Lab 2</a></td>
</tr>
<tr>
<td>W 8/31</td>
<td></td>
<td><a href="#">HW 1</a> available (due at Midterm I review session)</td>
</tr>
<tr>
<td>Th 9/1</td>
<td>Single-Cycle Datapaths</td>
<td><a href="#">5.1-4, 5.8, Appendix B</a></td>
</tr>
<tr>
<td>F 9/2</td>
<td></td>
<td><a href="#">Teams Meet the TA, 12-2 PM, 125 Cory</a></td>
</tr>
<tr>
<td>Sa 9/3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Su 9/4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M 9/5</td>
<td>(Labor Day Holiday)</td>
<td></td>
</tr>
<tr>
<td>T 9/6</td>
<td>Single-Cycle Wrap-Up</td>
<td>Lab 1: Final Report Due, 11:59 PM, via the submit program</td>
</tr>
</tbody>
</table>

Do 150 labs self-study (not handed in or graded). Ask the 152 TAs questions, not the 150 TAs.
## Weeks Two and Three: Lab 2 Begins

<table>
<thead>
<tr>
<th>Day</th>
<th>Schedule/Activities</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Th</td>
<td>Testing and Teamwork</td>
<td>Preliminary Design Document due to TAs, 11:59PM</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>Design Document Review, 12-2PM or 3-5PM, 125 Cory</td>
</tr>
<tr>
<td>Sa</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Su</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>Timing</td>
<td>Final Design Document due to TAs, 11:59PM</td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Th</td>
<td>Performance</td>
<td>Lab 2: ModelSim Checkoff, 12-2PM or 3-5PM, 125 Cory</td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>W 9/21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>Th 9/22</td>
<td>Pipelining II</td>
<td>PDF</td>
</tr>
<tr>
<td>F 9/23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sa 9/24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Su 9/25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M 9/26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T 9/27</td>
<td>Pipelining III</td>
<td>PDF</td>
</tr>
<tr>
<td>W 9/28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Th 9/29</td>
<td>Midterm Review Session in Class</td>
<td>PDF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F 9/30</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lab access, remote work, lab rules

For card key access to 125 Cory: apply at 253 Cory (& maybe CS office).

Also work remotely -- see Remote Desktop info on Resources page.

Accounts: Handouts, also on 125 Cory wall ...

Rules: Eat/drink at round tables only, log equipment failures in log book (and email TAs), don’t share your account, logoff after each session, be considerate when doing non-152 work and for 150 scheduled labs, be crime/safety aware.

Printing: The first 250+ pages are free (count includes cover sheets). Then, $12 per 200 pages. Plan ahead ...

We expect you to obey the EECS Policy on Academic Dishonesty. See “Course Info” on website for info.
Office Hours, Mid-terms ...

David: TBA
Udam: TBA
John: Mon 9:30-10:30 AM

Mid-term 1: Tuesday October 4th, 6:00 to 9:00 PM, 310 Soda.

Mid-term 2: Tuesday December 6th, 6:00 to 9:00 PM, 310 Soda.

Check for schedule conflicts now, let us know by Thursday.
MIPS Instruction Set

Your CPU projects will use a subset of the MIPS ISA.

Lab 1 refreshes your MIPS machine language skills, uses SPIM.

<table>
<thead>
<tr>
<th>Lecture Topic</th>
<th>Notes</th>
<th>Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>The MIPS ISA</td>
<td></td>
<td>Ch 2, 3.1-3, 3.8</td>
</tr>
</tbody>
</table>

CS 152 L1: The MIPS ISA
MIPS Instruction Set

Use the MIPS ISA document as the final word on the ISA (+ labs). Not P&H!

MIPS ISA document available on Resources page on class website.
Syntax: ADD $8 $9 $10  Semantics: $8 = $9 + $10

Fieldsize: 6 bits  5 bits  5 bits  5 bits  5 bits  5 bits  6 bits
Bitfield:   opcode  rs  rt  rd  shamt  funct

Binary: 000000  01001  01010  01000  00000  100000

In Hexadecimal: 012A4020
Hardware implements semantics ...

Syntax: ADD $8 $9 $10  Semantics: $8 = $9 + $10

Fetch next inst from memory: 012A4020

<table>
<thead>
<tr>
<th>opcode</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>funct</th>
</tr>
</thead>
</table>

Decode fields to get: ADD $8 $9 $10

"Retrieve" register values: $9 $10

Add $9 to $10

Place this sum in $8

Prepare to fetch instruction that follows the ADD in the program.
Memory Instructions: \texttt{LW $1, 32($2)}

Fetch the load inst from memory

```
<table>
<thead>
<tr>
<th>opcode</th>
<th>rs</th>
<th>rt</th>
<th>offset</th>
</tr>
</thead>
</table>
```

"I-Format"

Decode fields to get: \texttt{LW $1, 32($2)}

"Retrieve" register value: $2

Compute memory address: 32 + $2

Load memory address contents into: $1

Prepare to fetch instr that follows the \texttt{LW} in the program. Depending on load semantics, new $1 is visible to that instr, or not until the following instr ("delayed loads").
Branch Instructions: \texttt{BEQ} \ $1, $2, 25

- **Fetch branch inst from memory**

  \begin{tabular}{c|c|c|c}
  \texttt{opcode} & \texttt{rs} & \texttt{rt} & \texttt{offset} \\
  \end{tabular}

  “I-Format”

- **Decode fields to get:** \texttt{BEQ} \ $1, $2, 25

  “Retrieve” register values: \ $1, $2

- **Compute if we take branch:** \ $1 == $2 ?

  ALWAYS prepare to fetch instr that follows the \texttt{BEQ} in the program (“delayed branch”). IF we take branch, the instr we fetch AFTER that instruction is PC + 4 + 100.

  \textbf{PC} == “Program Counter”
Conclusions: The Architect’s Contract

To the program, it appears that instructions execute in the correct order defined by the ISA.

As each instruction completes, the machine state (regs, mem) appears to the program to obey the ISA.

What the machine actually does is up to the hardware designers, as long as the contract is kept.

The primary challenge of 152 CPU projects!
Example: Lab 3, Week 2: Hazard-Free Code

On **Friday 10/7** in lab section, you will demonstrate the processor running on the Calinx board. During the demo, the TAs will provide you with test code whose instruction sequence does not require the use of forwarding muxes for correct operation, and whose load instructions assume the presence of one load delay slot. These programs will test basic pipelining operation, but will not test "hard" pipelining features: how to stall and how to forward. Apart from these exceptions, the semantics of all of the instructions in the table shown in Problem 1a must be supported for this checkoff. Referring to Figure 6.36 of COD/3e, the "hazard detection unit" and "forwarding unit" need not be implemented for this checkoff.

A sample program

```
ADD R4, R3, R2
OR  R7, R6, R5
SUB R10, R9, R8
```

R’s chosen so that instructions are independent - like cars on the line.
Week 3: Run TA’s “Hard Tests” on Xilinx

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>10/14</td>
<td></td>
</tr>
<tr>
<td>Sa</td>
<td>10/15</td>
<td></td>
</tr>
<tr>
<td>Su</td>
<td>10/16</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>10/17</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lab 3: Final Xilinx Checkoff, 12-2PM or 3-5PM, 125 Cory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lab 3: Final Report Due, 11:59 PM via the submit program</td>
</tr>
</tbody>
</table>

4. On Friday 10/14 in lab section, you will demonstrate the processor running on the Calinx board. Unlike the 10/7 checkoff, this demo will test all facets of the processor (including the issues discussed in Problem 4). During this demo, the TA will provide you with secret test code. If you are able to pass these tests on your first try, you will receive bonus points. You can also receive bonus points if you fix your processor to pass the tests within your section time. If your processor is not fixed by the end of section, your TA will provide source for the secret test code, for use in your weekend debugging sessions.

5. On Monday 10/17 at 11:59 PM the lab (including the lab report) is due.

New sample program

ADD R4, R3, R2
OR R5, R4, R2

An example of a “hazard” — we must (1) detect and (2) resolve all hazards to make a CPU that matches ISA.
Jump and Link (JAL).

Lab 3, final project.

TA JAL test code stresses pipeline logic.

Jump and Link

<table>
<thead>
<tr>
<th>JAL</th>
<th>000011</th>
<th>instr_index</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>25</td>
<td>0</td>
</tr>
</tbody>
</table>

**Format:** JAL target

**Purpose:**
To execute a procedure call within the current 256 MB-aligned region

**Description:**
Place the return address link in GPR 31. The return link is the address of the second instruction following the branch, at which location execution continues after a procedure call.

This is a PC-region branch (not PC-relative); the effective target address is in the “current” 256 MB-aligned region. The low 28 bits of the target address is the `instr_index` field shifted left 2 bits. The remaining upper bits are the corresponding bits of the address of the instruction in the delay slot (not the branch itself).

Jump to the effective target address. Execute the instruction that follows the jump, in the branch delay slot, before executing the jump itself.

**Restrictions:**
Processor operation is **UNPREDICTABLE** if a branch, jump, ERET, DERET, or WAIT instruction is placed in the delay slot of a branch or jump.

**Operation:**

\[
\begin{align*}
I & : \text{GPR}[31] \leftarrow \text{PC} + 8 \\
I+1 & : \text{PC} \leftarrow \text{PC}_{\text{GPRLEN-1..28}} | | \text{instr\_index} | | 0^2
\end{align*}
\]

**Exceptions:**
None

**Programming Notes:**

Forming the branch target address by catenating PC and index bits rather than adding a signed offset to the PC is an advantage if all program code addresses fit into a 256 MB region aligned on a 256 MB boundary. It allows a branch from anywhere in the region to anywhere in the region, an action not allowed by a signed relative offset.

This definition creates the following boundary case: When the branch instruction is in the last word of a 256 MB region, it can branch only to the following 256 MB region containing the branch delay slot.
“Overall the students need to stick to the lab spec sheet to do a better job. All the bugs could be found from going down the list of things to test, but many students took the approach of testing everything they could think of.”
Here is a list of things you should check for:

1. Register zero always has the value zero.
2. All the branch and jump instructions should jump to the right address under the right conditions.
3. All the branch and jump instructions with link should put the correct return address into the link register.
4. SLT(I) and STLU(I) must work properly with bit patterns representing negative two's complement numbers.
5. All the arithmetic and logical instructions with immediate field should extend it properly.
6. Make sure a load that follows a store to the same address reads the appropriate data.
7. Jump and branch instruction should have a delay slot.
8. Overflow should be detected correctly in arithmetic and logical instructions.
9. Shifting instructions work correctly and extend the MSB appropriately.

as you mentioned you wanted to speak with the students about future lab reports. They were good for the most part as far as the report itself, but they need to pay closer attention to the spec sheet and have more robust testing.
New successful instruction sets are rare

Implementors suffer with original sins of ISAs, to support the installed base of software.