
CS 152

Computer Architecture and Engineering

Lecture 1 – The MIPS ISA

2005-8-30

John Lazzaro
(www.cs.berkeley.edu/~lazzaro)

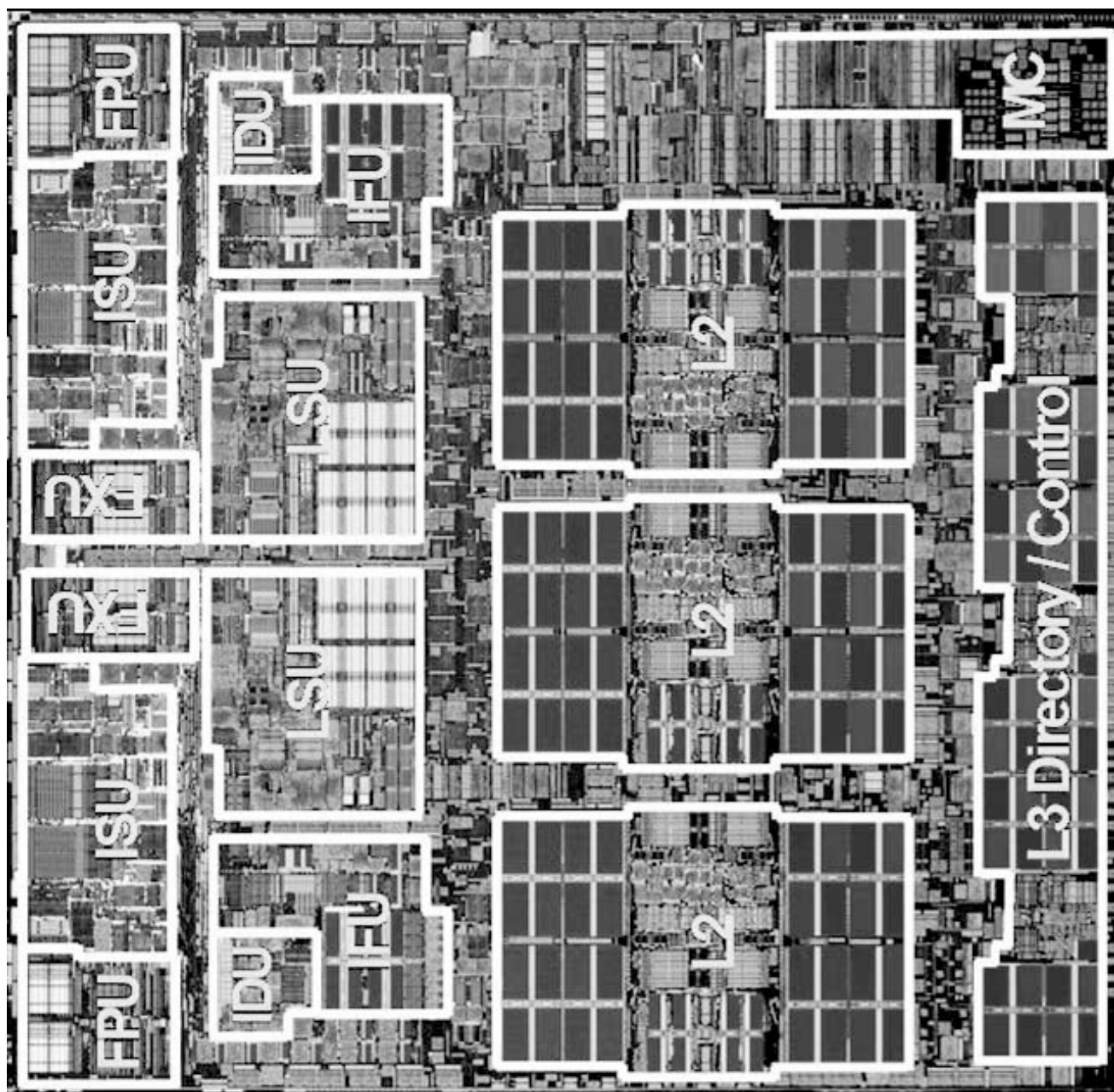
*And also, an
intro to the
course.*

TAs: David Marquardt and Udam Saini

www-inst.eecs.berkeley.edu/~cs152/



CS 152: Computer Design Team Projects



**Single-cycle
CPU project**

3 weeks

Pipelined CPU

3 weeks

Final Project

5 weeks

200 hr/student

Teams of

4-5 students

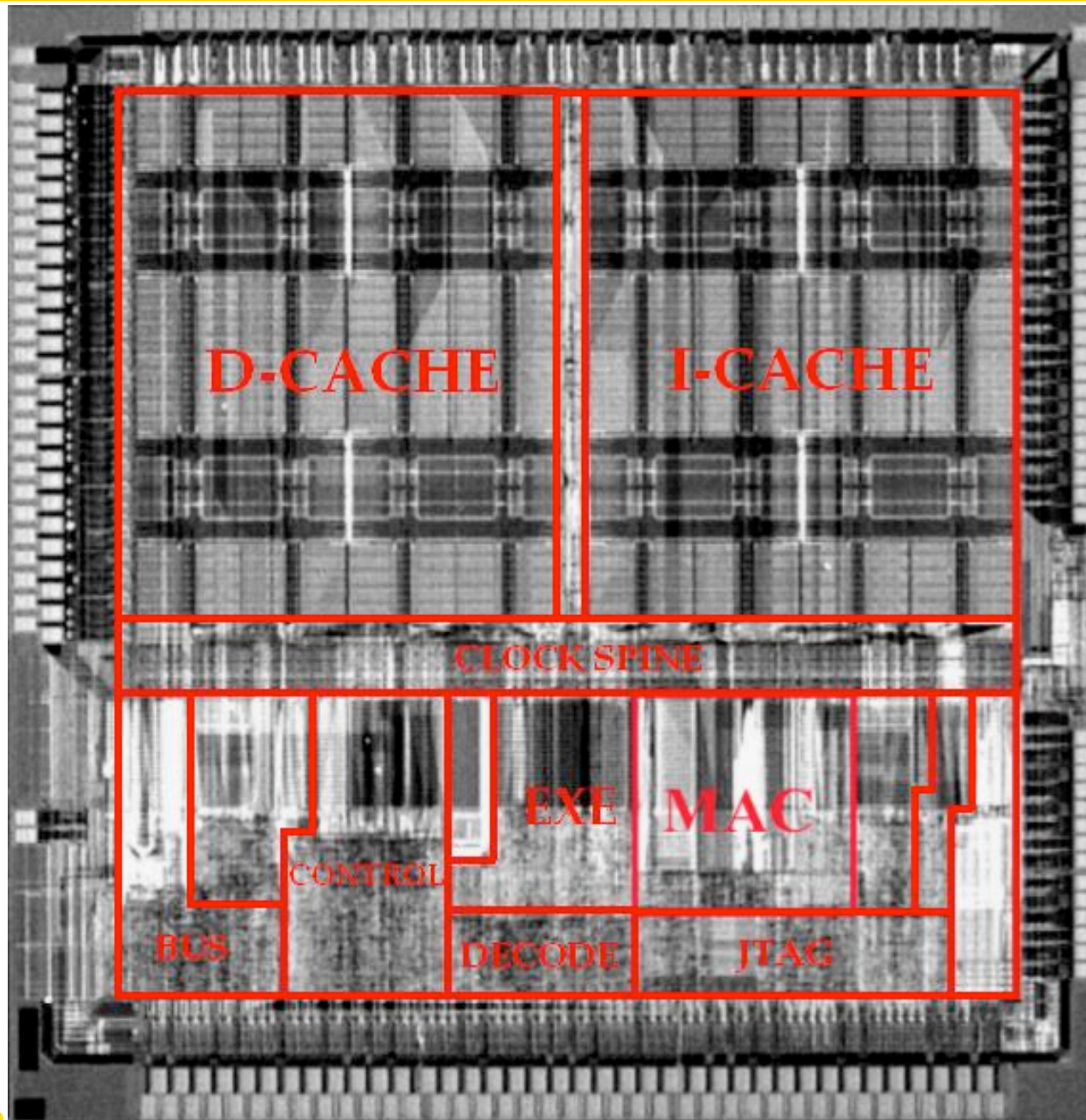


IBM Power 5 “die photo”: a die is an unpackaged part

CS 152 L1: The MIPS ISA

UC Regents Fall 2005 © UCB

CS 152: Real hardware, not simulation



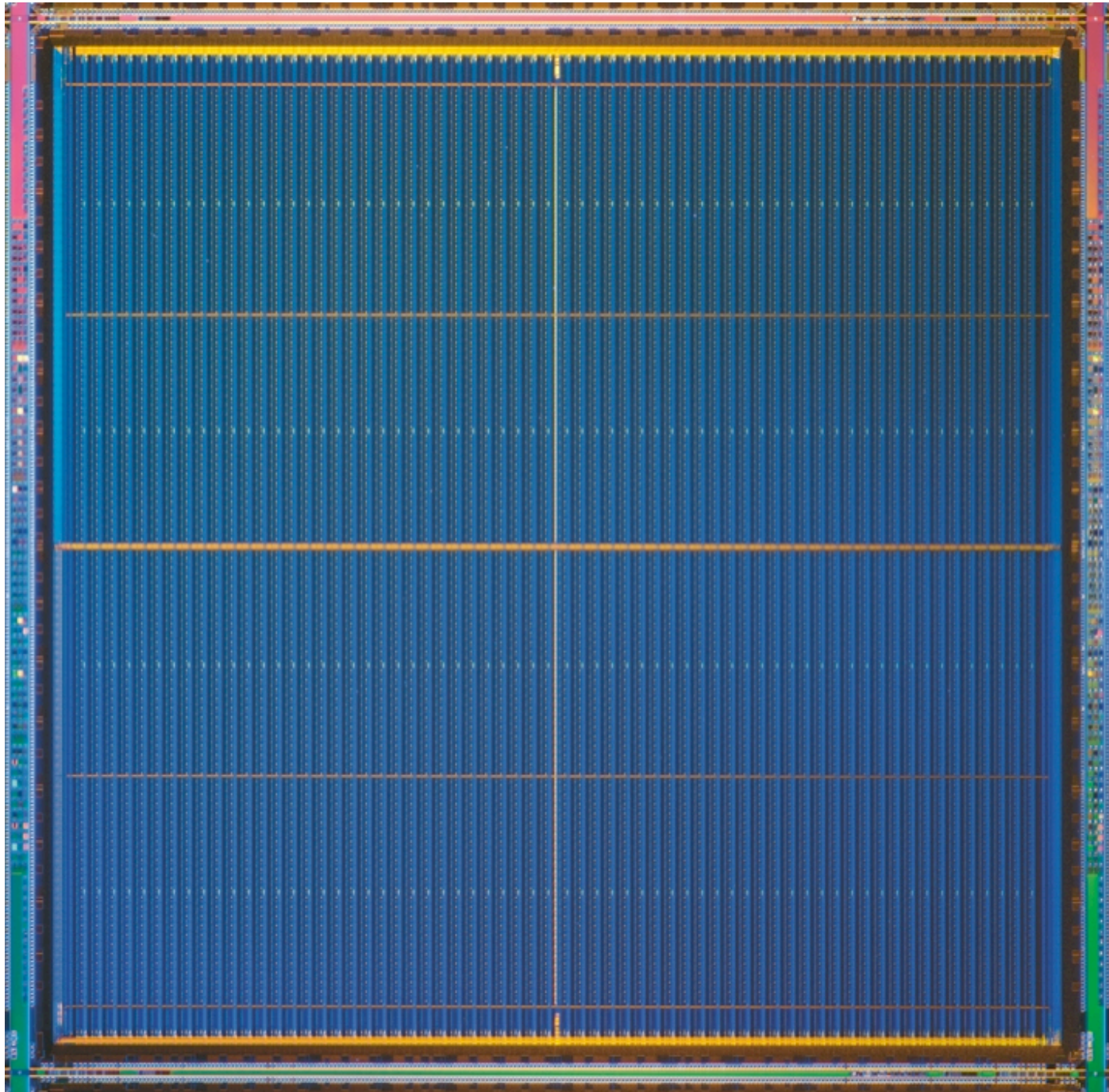
**Will we be
fabricate
CPU dies?**

**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.**

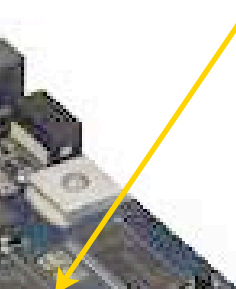


Calinx: 150/152 boards in 125 Cory

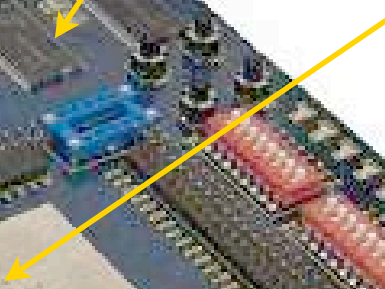
Download CPU
machine code
using TFTP



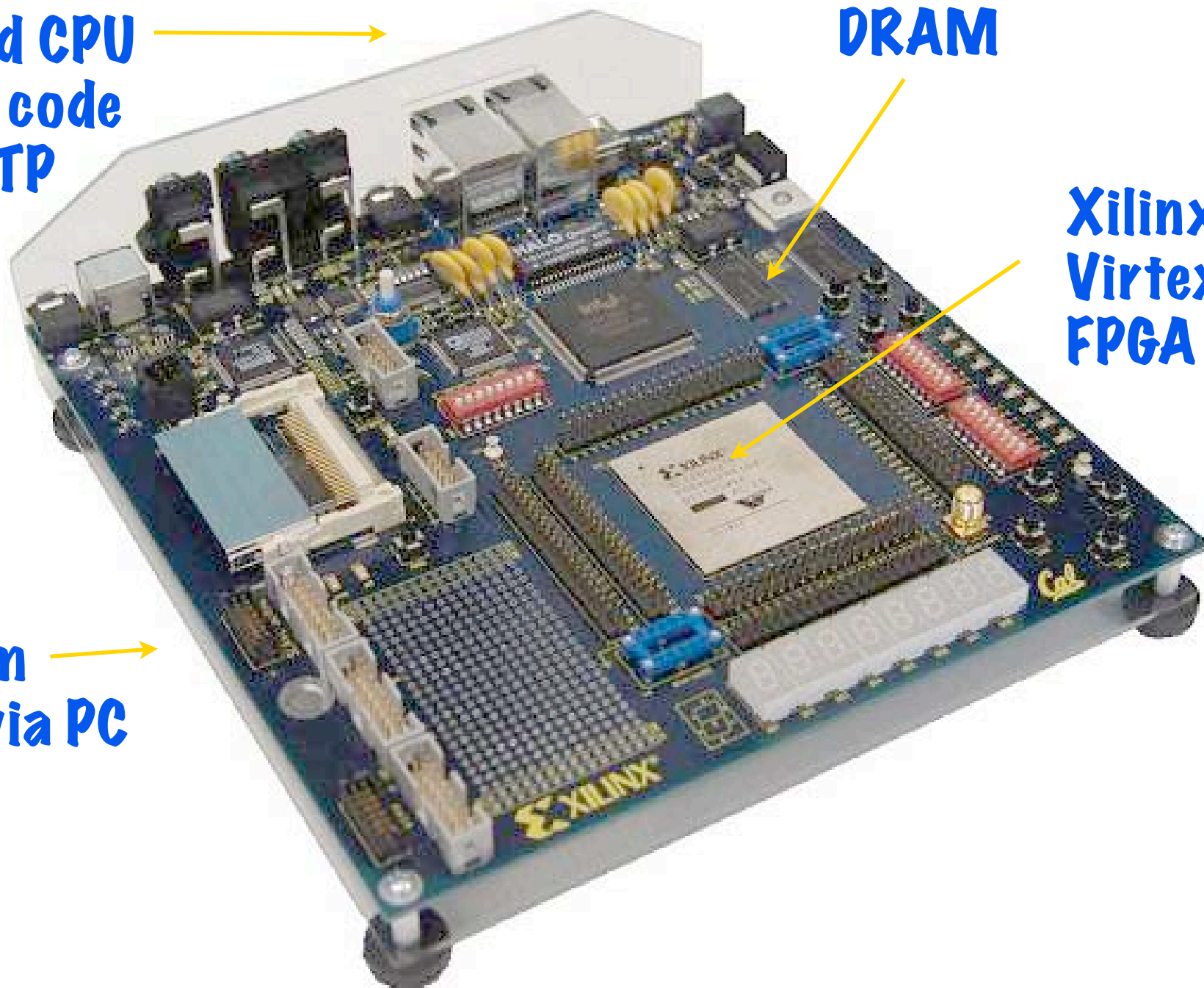
DRAM



Xilinx
Virtex E
FPGA



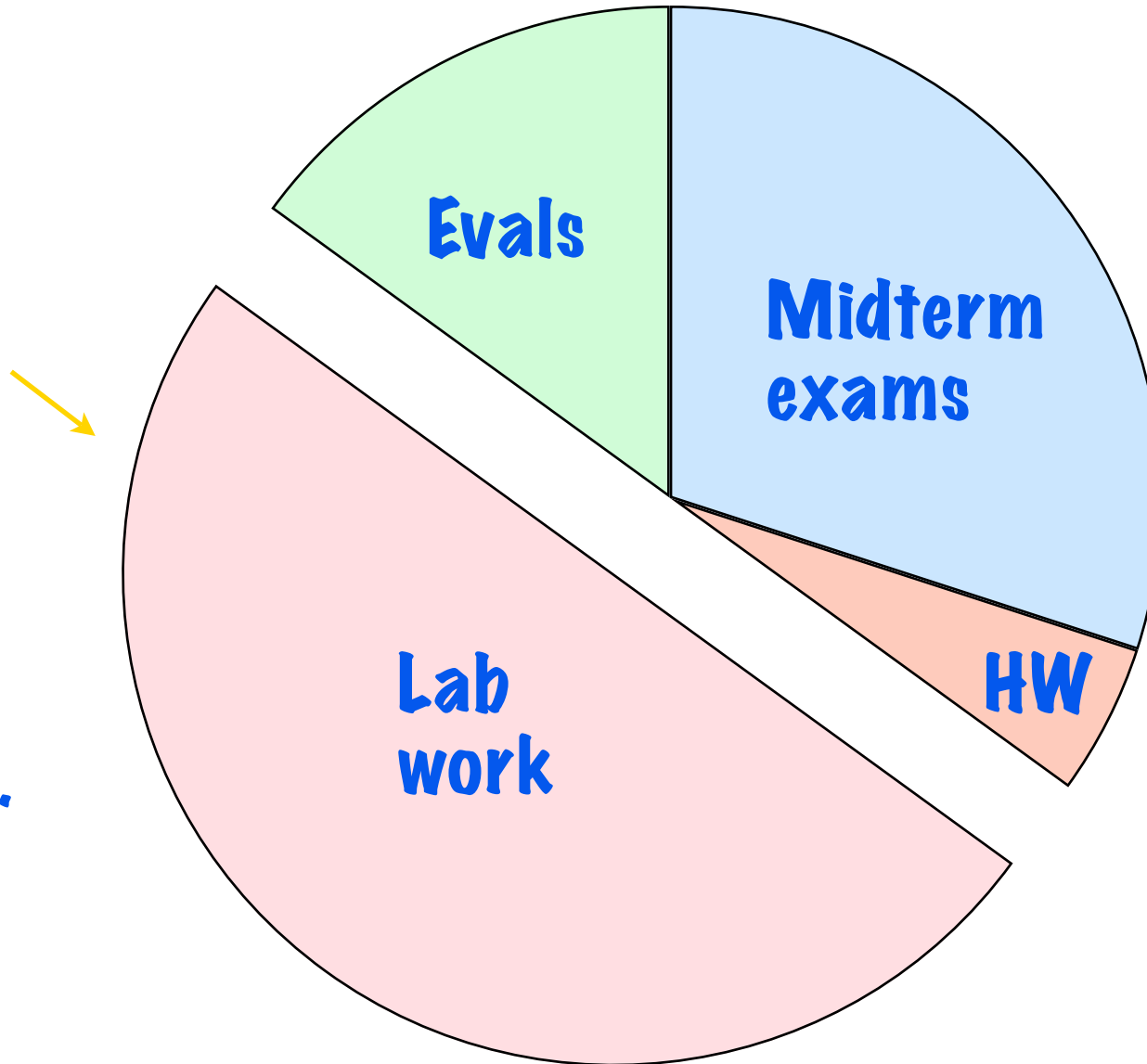
Program
Xilinx via PC



CS152: Approximate grading weights

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

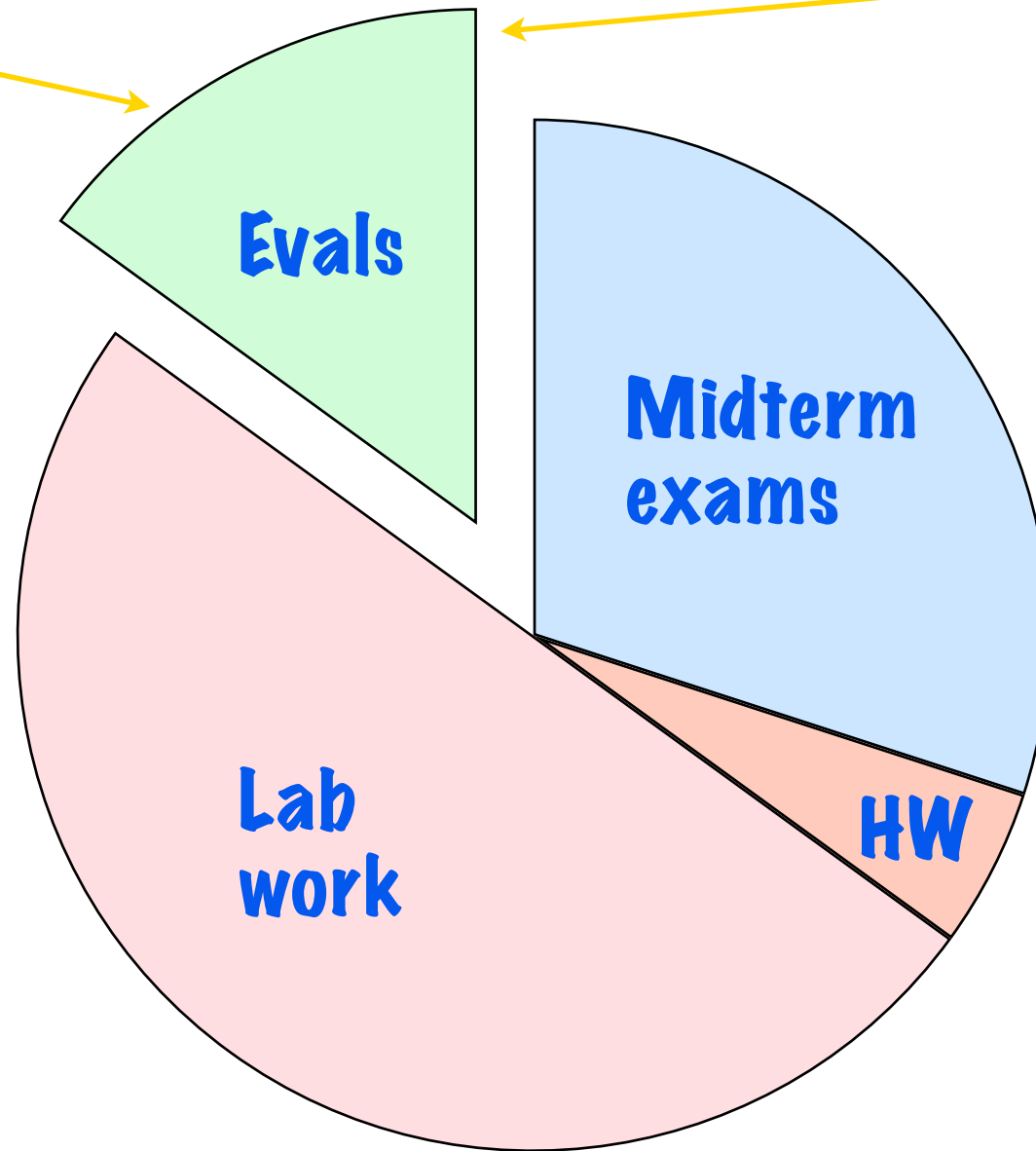
Subject to
fine tuning ...



Grading: Peer and staff evaluations

**Peer
evals:**

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



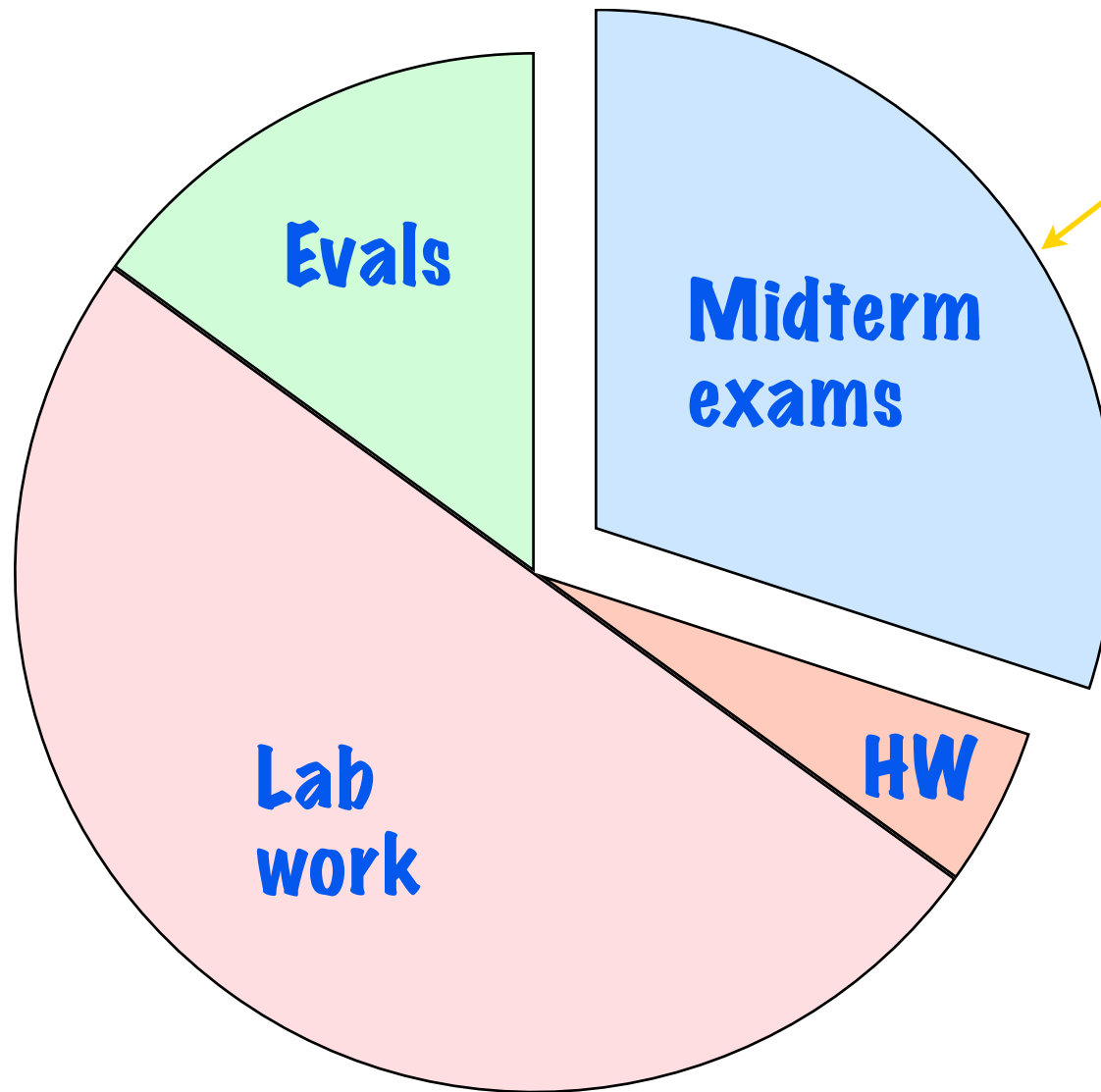
**Staff
evals:**

**TAs are
your
“managers”**

**They
observe
how well
you work
on the
team.**



Exams: Two mid-terms, no final ...



**MT1: Oct 4th.
About 10% of
final grade.**

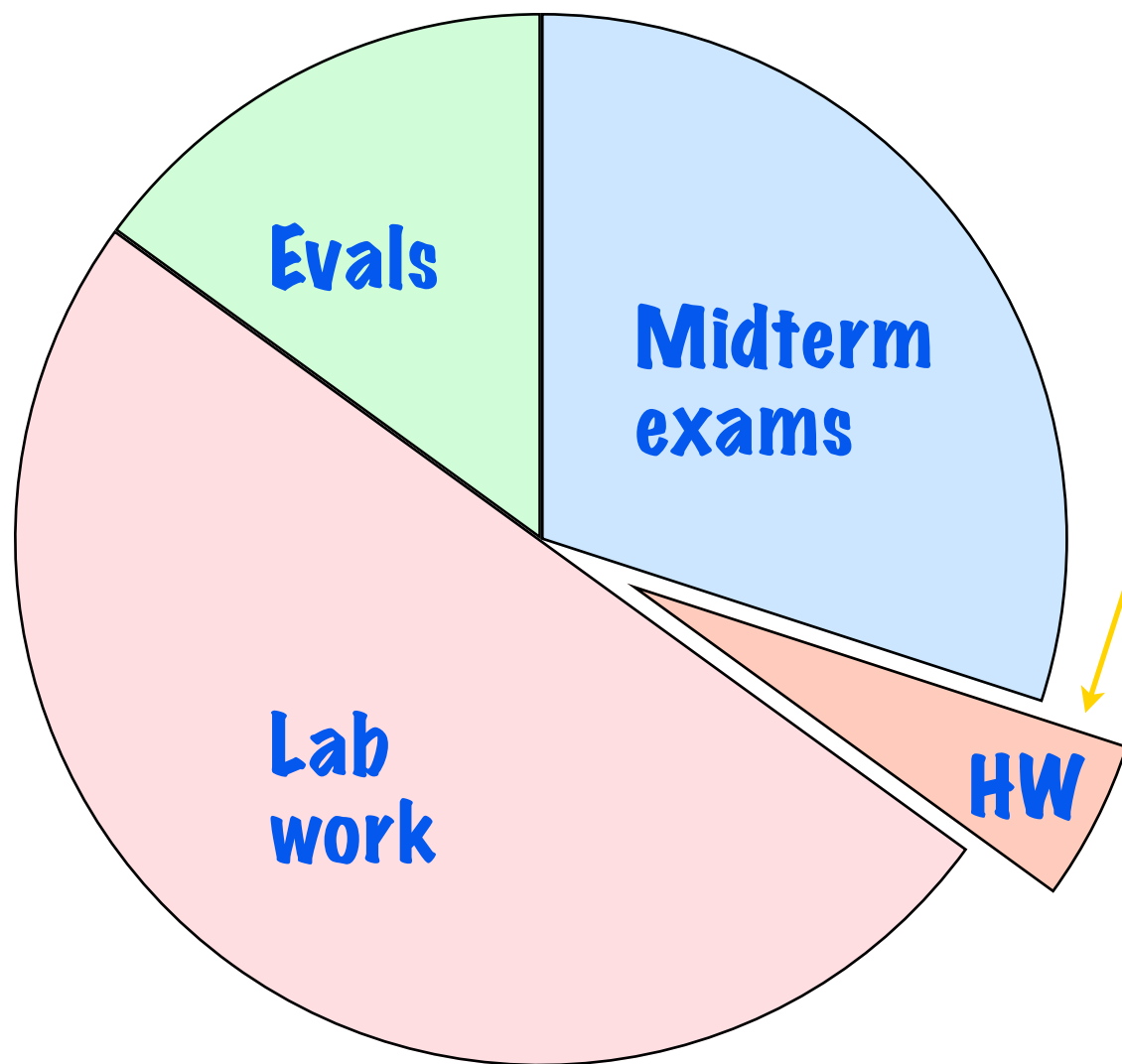
**MT2: Dec 6th.
About 20% of
final grade.**

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



In-class review session before each mid-term.

Homeworks: To prepare for midterms



Homework due at mid-term review session.

Based on last year's mid-term.

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")

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

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.

T 8/30	The MIPS ISA	PDF	Ch 2, 3.1-3, 3.8	Start on Lab 1 ; Preview Lab 2
W 8/31				HW 1 available (due at Midterm I review session)
Th 9/1	Single-Cycle Datapaths	PDF	5.1-4, 5.8, Appendix B	
F 9/2				Teams Meet the TA 12-2 PM, 125 Cory 150 Lab Lecture , 2-3 PM, 125 Cory
Sa 9/3	TAs: David Marquardt and Udam Saini. Undergrads, "A team" CS152 alumni. See web page for contact info.			
Su 9/4				
M 9/5	(Labor Day Holiday)			
T 9/6	Single-Cycle Wrap-Up	PDF		Lab 1: Final Report Due, 11:59 PM, via the submit program

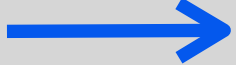
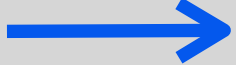
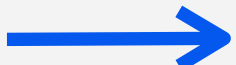
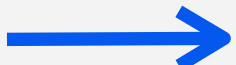
Week One: 150 boot-camp begins.

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.

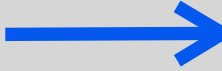
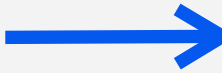
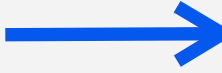
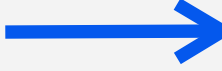
T 8/30	The MIPS ISA	PDF	Ch 2, 3.1-3, 3.8	Start on Lab 1 ; Preview Lab 2
W 8/31				HW 1 available (due at Midterm I review session)
Th 9/1	Single-Cycle Datapaths	PDF	5.1-4, 5.8, Appendix B	
F 9/2				Teams Meet the TA, 12-2 PM, 125 Cory 150 Lab Lecture , 2-3 PM, 125 Cory
Sa 9/3				
Su 9/4				
M 9/5	(Labor Day Holiday)			
T 9/6	Single-Cycle Wrap-Up	PDF		Lab 1: Final Report Due, 11:59 PM, via the submit program

**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

Th 9/8	Testing and Teamwork	PDF		Lab 2: Preliminary Design Document due to TAs, 11:59PM
F 9/9				Lab 2: Design Document Review, 12-2PM or 3-5PM, 125 Cory 150 Lab Lecture , 2-3 PM, 125 Cory
Sa 9/10				
Su 9/11				
M 9/12				Lab 2: Final Design Document due to TAs, 11:59PM
T 9/13	Timing	PDF , PDF		
W 9/14				
Th 9/15	Performance	PDF	5.5, 5.7	
F 9/16				Lab 2: ModelSim Checkoff, 12-2PM or 3-5PM 125 Cory 150 Lab Lecture , 2-3 PM, 125 Cory

Week Four/Five: Lab 2 Ends, Begin Lab 3

W 9/21				
Th 9/22	Pipelining II	PDF	6.5-7	
F 9/23				Lab 2: Xilinx Checkoff, 12-2PM or 3-5PM, 125 Cory 150 Lab Lecture , 2-3 PM, 125 Cory
Sa 9/24				
Su 9/25				
M 9/26				Lab 2: Final Report due, 11:59 PM, via the submit program
T 9/27	Pipelining III	PDF	6.8-9	
W 9/28				
Th 9/29	Midterm Review Session in Class	PDF		HW 1 due in class Lab 3: Preliminary Design Document and Team Evaluations due to TAs, 11:59PM
F 9/30				Lab 3: Preliminary Design Document Review, 12-2PM or 3-5PM, 125 Cory

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**.



Text, Printing, Honor Code ...

* **Text:** “**Computer Organization and Design**”, 3rd Edition, David **Patterson** and John **Hennessy**.

* **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.

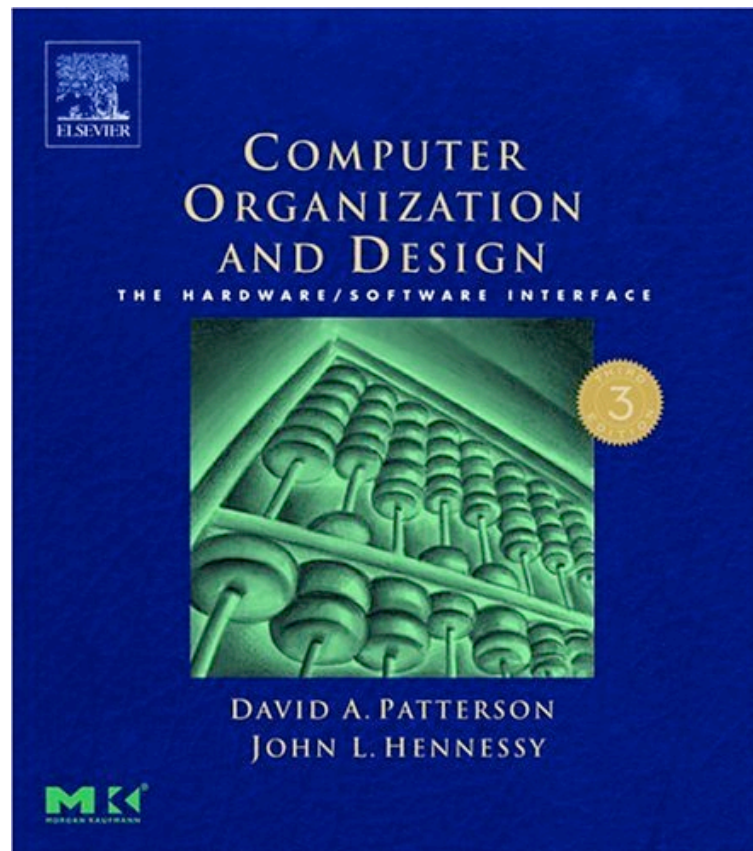


Lecture Topic	Notes	Reading
The MIPS ISA		Ch 2, 3.1-3, 3.8



MIPS Instruction Set

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



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



MIPS Instruction Set

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

MIPS
TECHNOLOGIES

**MIPS32™ Architecture For Programmers
Volume II: The MIPS32™ Instruction Set**

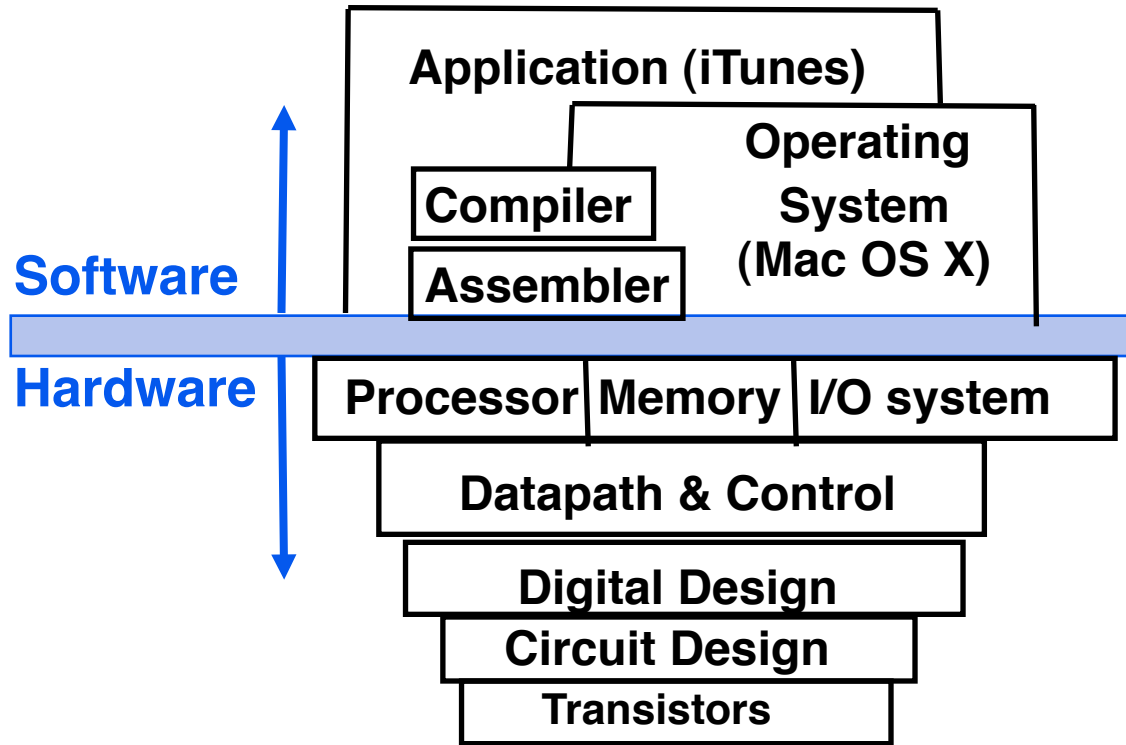
Document Number: MD00086
Revision 2.00
June 9, 2003

**MIPS ISA
document
available on
Resources
page on
class
website.**



Instruction Sets: A Thin Interface

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



Instruction Set Architecture

"R-Format"

Fieldsize: 6 bits 5 bits 5 bits 5 bits 5 bits 6 bits

Bitfield:

opcode	rs	rt	rd	shamt	funct
--------	----	----	----	-------	-------

Binary: 00000 01001 01010 01000 00000 100000

In Hexadecimal: 012A4020

Hardware implements semantics ...

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

**Instruction
Fetch**

Fetch next inst from memory: 012A4020

**Instruction
Decode**

opcode	rs	rt	rd	shamt	funct
--------	----	----	----	-------	-------

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

**Operand
Fetch**

"Retrieve" register values: \$9 \$10

Execute

Add \$9 to \$10

**Result
Store**

Place this sum in \$8

**Next
Instruction**

Prepare to fetch instruction that follows the ADD in the program.



Memory Instructions: LW \$1, 32(\$2)

Instruction
Fetch

Fetch the load inst from memory

Instruction
Decode



"I-Format"

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

Operand
Fetch

"Retrieve" register value: \$2

Execute

Compute memory address: $32 + \$2$

Result
Store

Load memory address contents into: \$1

Next
Instruction

Prepare to fetch instr that follows the 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: BEQ \$1, \$2, 25

Instruction
Fetch

Fetch branch inst from memory

Instruction
Decode

opcode	rs	rt	offset
--------	----	----	--------

 "I-Format"

Operand
Fetch

Decode fields to get: BEQ \$1, \$2, 25

Execute

"Retrieve" register values: \$1, \$2

Result
Store

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

Next
Instruction

ALWAYS prepare to fetch instr that follows the BEQ in the program ("delayed branch"). IF we take branch, the instr we fetch AFTER that instruction is $PC + 4 + 100$.

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**.

Example: Lab 3, Week 2: Hazard-Free Code

F 10/7				Lab 3: Initial Xilinx Checkoff, 12-2PM or 3-5PM, 125 Cory
-----------	--	--	--	---

3. On **Friday 10/7** in lab section, you will demonstrate the processor running on the Calix 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 expectations, 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.**

Type	Instructions
arithmetic	addu, subu, addiu
logical	and, andi, or, ori, xor, xori, lui
shift	sll, sra, srl
compare	slt, slti, sltu, sltui
control	beq, bne, bgez, bltz, j, jr, jal
data transfer	lw, sw
Other:	break



Week 3: Run TA's "Hard Tests" on Xilinx

F 10/14				Lab 3: Final Xilinx Checkoff, 12-2PM or 3-5PM, 125 Cory
Sa 10/15				
Su 10/16				
M 10/17				Lab 3: Final Report Due, 11:59 PM via the submit program

4. On **Friday 10/14** in lab section, you will demonstrate the processor running on the Calix 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**



CS 152: Testing against a published spec

Jump and Link (JAL).

Lab 3, final project.

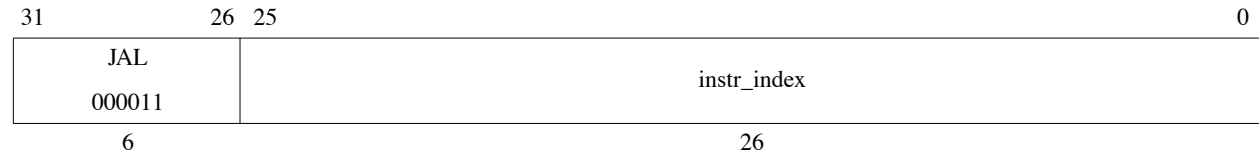
TA JAL test code

stresses

pipeline logic.

Jump and Link

JAL



Format: JAL target

MIPS32

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:

$$\begin{aligned} \mathbf{I}: \text{GPR}[31] &\leftarrow \text{PC} + 8 \\ \mathbf{I+1}: \text{PC} &\leftarrow \text{PC}_{\text{GPREN}-1..28} \parallel \text{instr_index} \parallel 0^2 \end{aligned}$$

Exceptions:

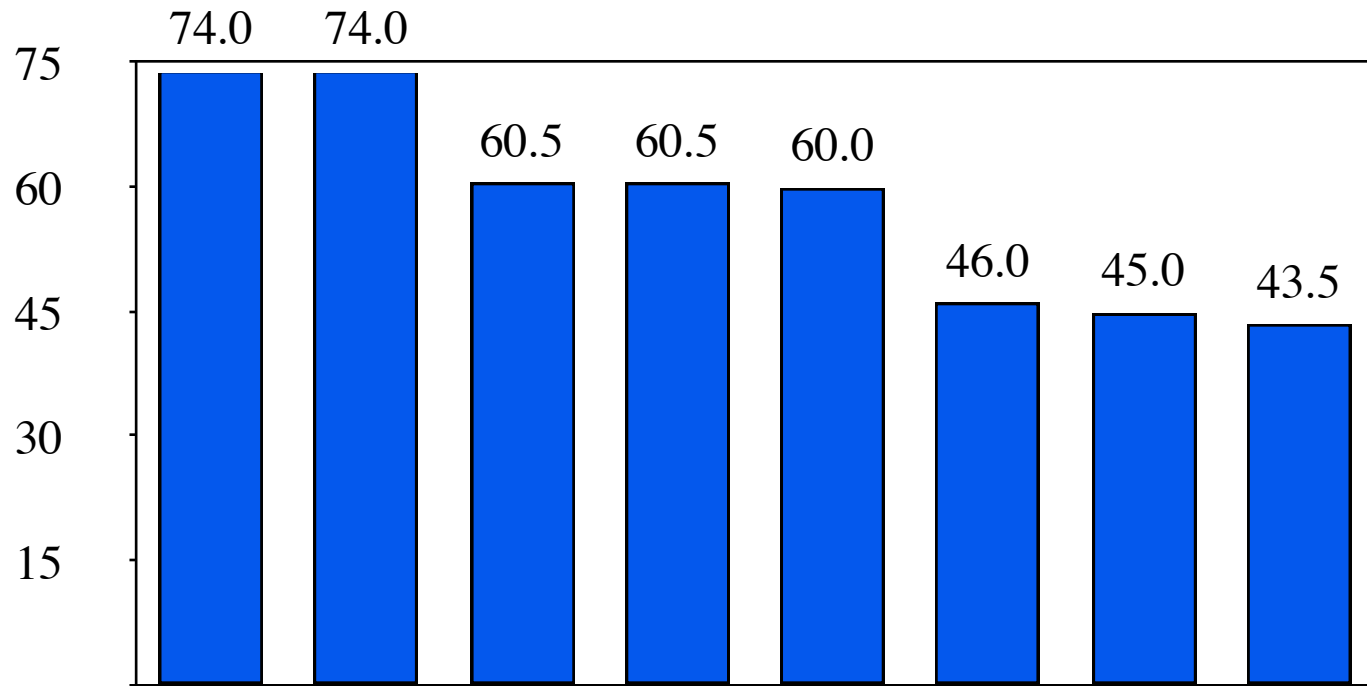
None

Programming Notes:

Forming the branch target address by concatenating 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.

Lab 1 grades, Spring 2005 ...



“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.”

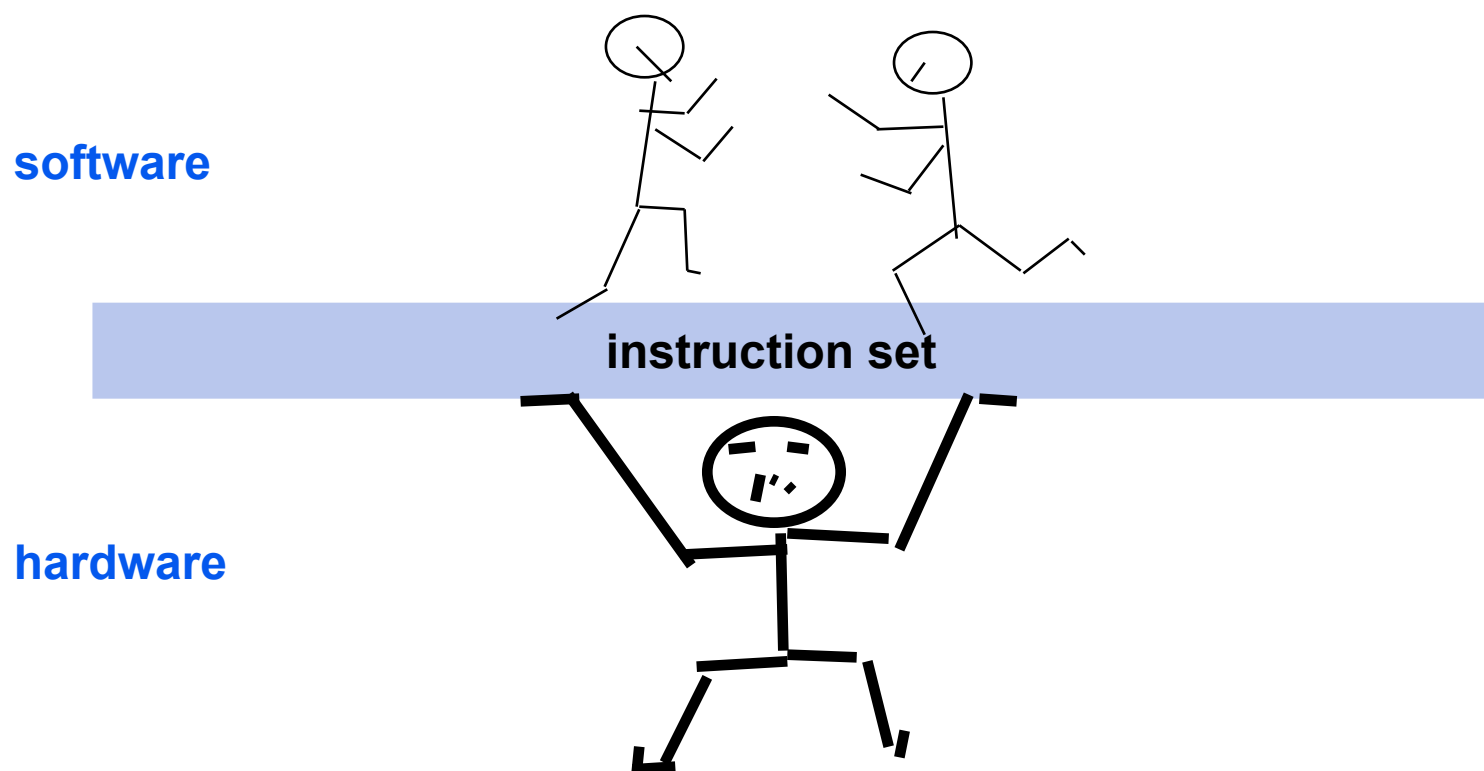
Lab 1 grades, Spring 2005 ...

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.