

COMP3211/COMP9211 Computer Architecture

Lecturer: Dr. Hui Annie Guo
huig@cse.unsw.edu.au
K17-501F (ext. 57136)

Overview

- Course overview
- Introduction to ISA

COMP3211/9211

2011S1 wk1_1 P2

Course overview

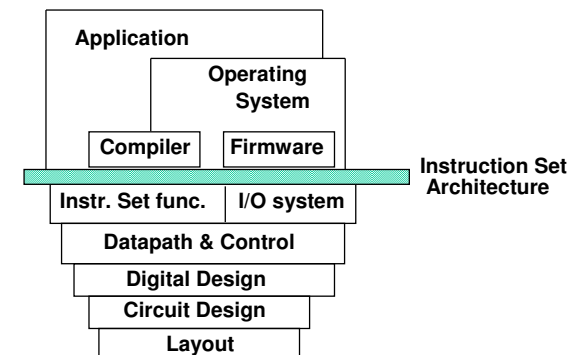
- What is the course about?
- Why is it important to learn this course?
- Aims of the course
- How to achieve them?
- Course organization
- Expectations
- Assessments
- etc

COMP3211/9211

2011S1 wk1_1 P3

What is this course about?

- This course is about
 - how computers work
 - how computers are designed
- Computer system abstraction Hierarchy



COMP3211/9211

2011S1 wk1_1 P4

What is computer architecture?

- **Computer architecture includes**
 - ISA (instruction set architecture)
 - Machine organization
- **ISA is the interface between hw and sw of a computer system.** It provides attributes visible to the programmer and specifications of functions to be implemented by hardware
 - e.g. Is there a multiply instruction?
- **Machine organization is how features are implemented**
 - e.g. Is there a hardware multiply unit or is it done by repeated addition?

COMP3211/9211

2011S1 wk1_1 P5

ISA & Organization

- **A family of machine can often share the basic architecture**
 - Provide code compatibility
 - E.g. Intel x86 family
- **However, organization varies significantly between versions**
- **Modern instruction set architectures:**
 - IA-32, PowerPC, MIPS, SPARC, ARM, and others

COMP3211/9211

2011S1 wk1_1 P6

Why important to learn this course

- **Computer architecture is a critical design level in computer systems, having significant impacts on**
 - Performance
 - Cost
 - Power consumption
 - And ...

COMP3211/9211

2011S1 wk1_1 P7

The aims of this course

- **Develop a deeper understanding of computer systems**
 - as a foundation for lifelong study of computer systems design
- **Know how to verify and evaluate a design**
 - Appreciate the importance of simulation as the primary means of validating designs and assessing performance
- **Practice professional skills in design and analysis, project management, and presentation**

COMP3211/9211

2011S1 wk1_1 P8

How to achieve these aims

- **Develop a deeper understanding of computer systems design**
 - Learn general principles and historical perspectives
 - Based on a popular RISC architecture
 - Hands-on exercises
 - Experiencing typical design phases
- **Know how a design can be verified and evaluated**
 - Appreciate the importance of simulation as the primary means of validating designs and assessing performance
 - Xilinx/Modelsim simulation tools
- **Practice professional skills in design and analysis, project management, and presentation**
 - Group project work
 - Discussions in tute classes

COMP3211/9211

2011S1 wk1_1 P9

How is the course organized

- **Lectures**
 - ISA design
 - Pipelined processor
 - Memory hierarchy
 - Bus system
 - Advanced design topics
- **Tutorials/Labs**
 - Discussion, project work, presentation

COMP3211/9211

2011S1 wk1_1 P10

Tutorials & Labs

- **One hour each per week**
- **Start Week 2**
- **Participation is essential**
 - To enhance your understanding of course materials
 - To effectively communicate with your project partners as well as other fellow students
 - To complete the project work; and
 - To present your project work

COMP3211/9211

2011S1 wk1_1 P11

Project

- **Two compulsory tasks**
 - Done in a group of 3 people
 - Formed by week 2.
 - Register your group with your tutor
- **One optional task**
 - Done individually
 - Bonus marks
- **Project will involve presentations**

COMP3211/9211

2011S1 wk1_1 P12

Quiz

- There will be a 50 minute quiz held during Wednesday lecture period in Week 6
- The quiz will cover all lecture, tutorial, and lab materials completed to the end of Week 5

COMP3211/9211

2011S1 wk1_1 P13

Expectations

- **Lecture**
 - Typical architecture types
 - What are they?
 - Design principles
 - What are they?
 - Why so?
 - Performance
 - Concepts and evaluation approaches
 - Single-cycle, multiple-cycle, pipelined datapath design
 - Related design issues
 - How are they designed?

COMP3211/9211

2011S1 wk1_1 P14

Expectations

- **Lecture (cont.)**
 - Memory hierarchy
 - Why do we need it?
 - The typical design issues and related solutions
 - Bus system
 - How does a bus system affect the overall performance?
 - Typical bus structures

COMP3211/9211

2011S1 wk1_1 P15

Expectations

- **Lab**
 - Lab equipment and tools
 - VHDL modeling and simulation
 - Using Xilinx WebPACK and ModelSIM
 - Completion of all labs
 - Prepare before lab
 - Understand the problems
 - Get familiar with the tools
 - Come out solutions
 - Finish tasks

COMP3211/9211

2011S1 wk1_1 P16

Expectations

- **Tutorial**
 - Attempt all questions before tutorial
 - Participate in the tute class
 - Discuss and understand solutions

COMP3211/9211

2011S1 wk1_1 P17

Expectations

- **Project**
 - Be responsible for your task in the project
 - Be responsible for the progress of whole group project
 - Try to improve your presentation skills and teamwork skills through project work

COMP3211/9211

2011S1 wk1_1 P18

Assessments

- **Participation of Tutes & Labs and project presentations**
 - 5%
- **Project**
 - 35% total
 - **Compulsory**
 - Task1 presentation, 8%
 - Task 2 presentation, 8%
 - Teamwork performance, 5%
 - Tasks 1&2 project report, 14%
 - **Optional***
 - Task 3, bonus (5%)

COMP3211/9211

2011S1 wk1_1 P19

Assessments

- **Quiz**
 - 20%
- **Final exam**
 - 2 hours
 - 40%
- **Final mark = Participation + Project + Quiz + Exam + bonus**
 - To pass the course, you must have (final mark ≥ 50) && (final exam ≥ 40)

COMP3211/9211

2011S1 wk1_1 P20

Staff

- **Tutor**
 - Mr Ji Gu
 - K17-510-04, Ext 54869, jigu@cse.unsw.edu.au
 - Ms Mei Hong
 - K17-510-03, Ext 57204, meihong@cse.unsw.edu.au
- **Lecturer**
 - Annie Guo
 - K17-501F, Ext 57136, huig@cse.unsw.edu.au
 - Consultation: Thursdays, 3-4pm

COMP3211/9211

2011S1 wk1_1 P21

Textbook and References

- **Textbook**
 - Computer Organization and Design: The Hardware/Software Interface, D.A. Patterson and J.L. Hennessy, 4th Ed., Morgan Kaufmann, 2009.
- **Reference books and docs**
 - See the course website for a list of references and downloadable software
- **Lecture notes**
 - Posted each week before the lecture
 - The lecture slides may not be the same as the lecture notes.
You are encouraged to take notes.

COMP3211/9211

2011S1 wk1_1 P22

Support

- **Course website**
 - <http://www.cse.unsw.edu.au/~cs3211>
- **Course staff**
 - The staff contact information can be found on course website
 - Note: email from your non-CSE account may not be answered.

COMP3211/9211

2011S1 wk1_1 P23

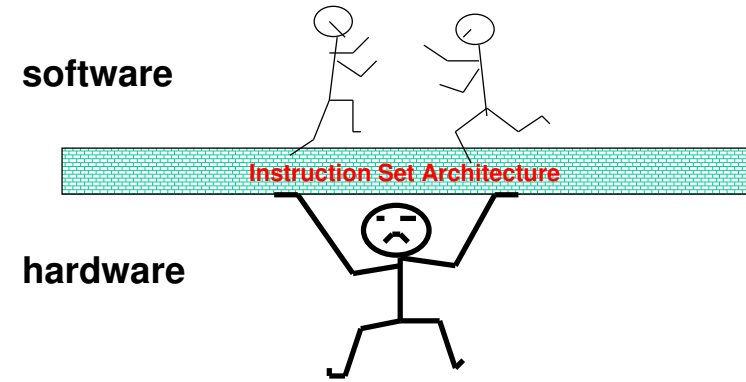
Introduction to Instruction Set Architecture and Design

Lecturer: Dr. Hui Annie Guo
huig@cse.unsw.edu.au
K17-501F (ext. 57136)

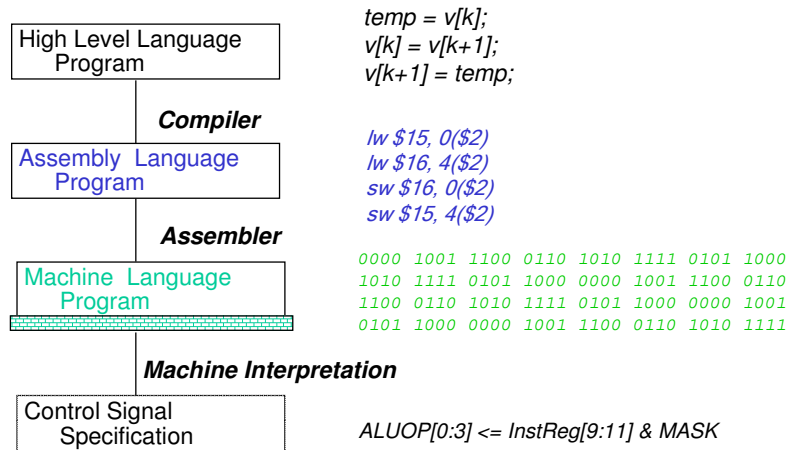
Introduction to ISA

- **What is instruction set architecture?**
 - Interface between hw/sw
 - Four classes of ISA
- **Four instruction set design principles**
- **Typical design issues and guidelines**
 - Registers
 - Addressing mode
 - Instruction format
 - Operations
 - Data

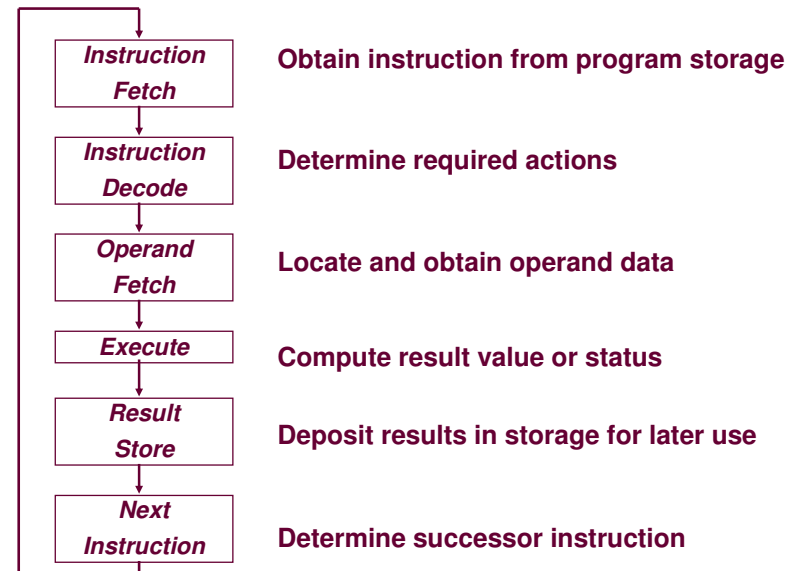
ISA - Interface between HW/SW



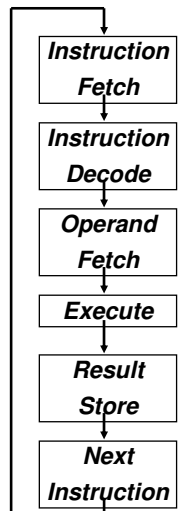
Levels of Representation



Execution Cycle



Instruction Set Architecture: What Must be Specified?



- Instruction Format or Encoding
 - how is it decoded?
- Location of operands and result
 - where other than memory?
 - how many explicit operands?
 - how are memory operands located?
 - which can or cannot be in memory?
- Data type and Size
- Operations
 - which are supported
- Successor instruction
 - jumps, conditions, branches
 - *fetch-decode-execute is implicit!*

Basic ISA Classes

- **Accumulator (1 register)**
 - 1 address add A acc acc + mem[A]
 - 1+x address addx A acc acc + mem[A + x]
- **Stack:**
 - 0 address add tos tos + next
- **General Purpose Register: (Register/Memory)**
 - 2 address add A B EA(A) EA(A) + EA(B)
 - 3 address add A B C EA(A) EA(B) + EA(C)
- **General Purpose Register: (Load/Store):**
 - 3 address add Ra Rb Rc Ra Rb + Rc
 - 2 address load Ra Rb Ra mem[Rb]
 - store Ra Rb mem[Rb] Ra

Programming Examples for four ISA Classes

- **Code sequence for $C = A + B$ for four classes of instruction sets:**

Stack	Accumulator	Register (register-memory)	Register (load-store)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R1,B	Load R2,B
Add	Store C	Store R1,C	Add R3,R1,R2
Pop C			Store R3, C

Instruction Set Architecture Design

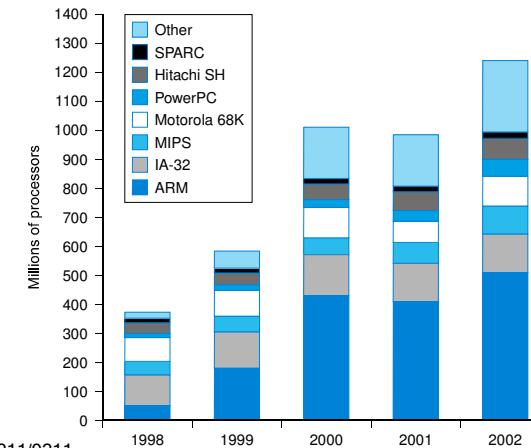
- **Goal:**
 - The instruction set should be easy to implement, it should give good performance (possibly more), and it should provide a ready target for high-level programs
- **Four (instruction set) design principles:**
 - Smaller is faster
 - Simplicity favors regularity
 - Make the common case fast
 - Good design demands a compromise

Example of Four Design Principles: MIPS

- **Simple (Simplicity favors regularity)**
 - MIPS instructions are all 32-bits large
 - Arithmetic instructions always have three operands
 - Operands of arithmetic instructions are in registers
- **Small (Smaller is faster)**
 - MIPS has a small register file of only 32 registers, each with 32 bits
- **Compromise (Good design demands a compromise)**
 - MIPS has three instruction formats
- **Optimizing of common case (making a common occurrence fast)**
 - Immediate values are provided in I-type instructions

Typical Design Issues and Guidelines

- There are many different ISAs
- We focus general-purpose register machines in this course



General Purpose Registers

- Since mid 1970's all machines use general purpose registers
- **Advantages**
 - registers are faster than memory
 - registers are easier for a compiler to use
 - e.g., $(A*B) - (C*D) - (E*F)$ can do multiplies in any order, whereas, need to take care with stack
 - registers can hold variables
 - memory traffic is reduced, so program is sped up, (since registers are faster than memory)
 - increase throughput since one instruction can manipulate multiple registers (c.f. add and lw)
 - code density improves (since register named with fewer bits than memory location)

MIPS Integer Registers

0	zero	constant 0	16	s0	callee saves
1	at	reserved for assembler	... (caller can clobber)		
2	v0	expression evaluation &	23	s7	
3	v1	function results	24	t8	temporary (cont'd)
4	a0	arguments	25	t9	
5	a1		26	k0	reserved for OS kernel
6	a2		27	k1	
7	a3		28	gp	Pointer to global area
8	t0	temporary: caller saves	29	sp	Stack pointer
...		(callee can clobber)	30	fp	frame pointer
15	t7		31	ra	Return Address (HW)

Memory Addressing

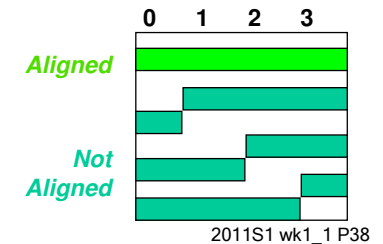
- Since 1980 almost every machine uses byte addresses
- Two issues of multi-byte objects stored in memory:
 - Endianness
 - The order of the multiple bytes stored in the memory
 - Alignment
 - The boundary of the multi-byte object in the memory

COMP3211/9211

2011S1 wk1_1 P37

Addressing Objects: Endianness and Alignment

- **Big Endian:**
 - address of most significant byte = word address
 - Most significant byte stored first
 - IBM 360/370, Motorola 68k, MIPS, Sparc, HP PA
- **Little Endian:**
 - address of least significant byte = word address
 - Least significant byte stored first
 - Intel 80x86, DEC Vax, DEC Alpha (Windows NT)
- **Alignment:**
 - objects fall on address that is multiple of their size.



COMP3211/9211

2011S1 wk1_1 P38

Possible Addressing Modes

Addressing mode	Example	Meaning
Immediate	Add R4,#3	$R4 \leftarrow R4+3$
Register	Add R4,R3	$R4 \leftarrow R4+R3$
Direct or absolute	Add R1,(1001)	$R1 \leftarrow R1+\text{Mem}[1001]$
Register indirect	Add R4,(R1)	$R4 \leftarrow R4+\text{Mem}[R1]$
Displacement	Add R4,100(R1)	$R4 \leftarrow R4+\text{Mem}[100+R1]$
Indexed / Base	Add R3,(R1+R2)	$R3 \leftarrow R3+\text{Mem}[R1+R2]$
Auto-increment	Add R1,(R2)+	$R1 \leftarrow R1+\text{Mem}[R2]; R2 \leftarrow R2+d$
Auto-decrement	Add R1,-(R2)	$R2 \leftarrow R2-d; R1 \leftarrow R1+\text{Mem}[R2]$
Scaled	Add R1,100(R2)[R3]	$R1 \leftarrow R1+\text{Mem}[100+R2+R3*d]$
Memory indirect	Add R1,@(R3)	$R1 \leftarrow R1+\text{Mem}[\text{Mem}[R3]]$

COMP3211/9211

2011S1 wk1_1 P39

Addressing Mode Usage? (ignore register mode)

3 programs measured on machine with all address modes (VAX)

--- Displacement:	42% avg, 32% to 55%	75%
--- Immediate:	33% avg, 17% to 43%	
--- Register deferred (indirect):	13% avg, 3% to 24%	88%
--- Scaled:	7% avg, 0% to 16%	
--- Memory indirect:	3% avg, 1% to 6%	
--- Misc:	2% avg, 0% to 3%	

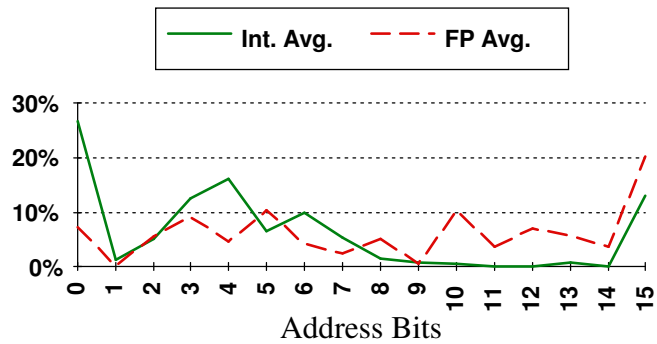
75% displacement & immediate
88% displacement, immediate & register indirect

What displacement values need to be supported?
What immediate values can be expected?

COMP3211/9211

2011S1 wk1_1 P40

Displacement Address Size?



- Avg. of 5 SPECint92 programs v. avg. 5 SPECfp92 programs
- X-axis is in powers of 2: 4 => addresses $> 2^3$ (8) and $\leq 2^4$ (16)
- 1% of addresses > 16 -bits
- 12 - 16 bits of displacement needed

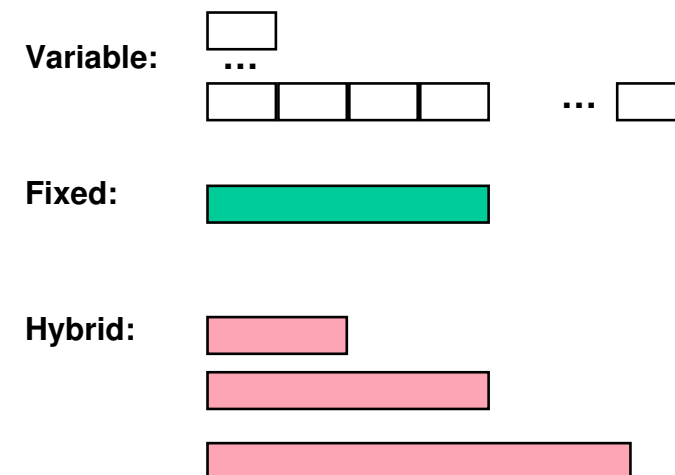
Immediate Size?

- 50% to 60% fit within 8 bits
- 75% to 80% fit within 16 bits

Addressing Modes Guideline Adopted by MIPS

- **Displacement**
 - Displacement size should be 12 to 16 bits
- **Immediate**
 - Immediate size should be 8 to 16 bits
- **Register Indirect**

Generic Examples of Instruction Format Widths



Some Instruction Format Design Strategies

- If code size is most important (as in some embedded apps), use variable length instructions
- If performance is most important, use fixed length instructions as MIPS does
- Embedded machines (ARM, MIPS) added optional mode to execute subset of 16-bit wide instructions (Thumb, MIPS16)
 - decide performance or density for each procedure
- If you have many memory operands per instruction and many addressing modes
 - use an Address Specifier per operand (VAX)
- If you have load-store machine with at most 1 address per instruction and only one or two addressing modes
 - encode addressing mode in the opcode (MIPS)

COMP3211/9211

2011S1 wk1_1 P45

MIPS Instruction Set Architecture

- We'll be working with the MIPS instruction set architecture
 - typical of architectures developed since the 1980's
 - almost 100 million MIPS processors manufactured in 2002
 - used by ATI Technologies, Broadcom, NEC, Nintendo, Cisco, Silicon Graphics, Sony, ...

COMP3211/9211

2011S1 wk1_1 P46

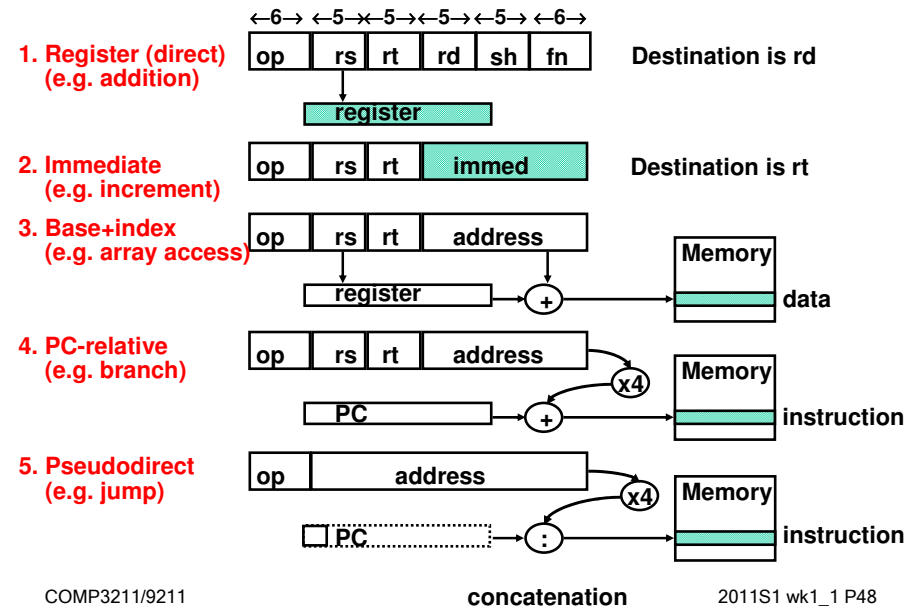
Features of MIPS ISA

- 32-bit fixed format inst (3 formats)
- 32 x 32-bit GPR (R0 contains zero) and 32 FP registers (and HI LO)
 - partitioned by software convention
- 3-address, reg-reg arithmetic instructions
 - AL operations are always performed on registers
- Single address mode for load/store: base+displacement
 - no indirection, scaled
 - 16-bit immediate plus LUI
- Simple branch conditions
 - compare against zero or two registers
 - no integer condition codes
- Delayed branch
 - execute instruction after the branch (or jump) even if
 - the branch is taken (Compiler can fill a delayed branch)

COMP3211/9211

2011S1 wk1_1 P47

5 MIPS Addressing Modes/3 Instruction Formats



COMP3211/9211

2011S1 wk1_1 P48

MIPS Arithmetic Instructions

<u>Instruction</u>	<u>Example</u>	<u>Meaning</u>	<u>Comments</u>
add	add \$1,\$2,\$3	$\$1 = \$2 + \$3$	3 operands;
subtract	sub \$1,\$2,\$3	$\$1 = \$2 - \$3$	3 operands;
add immediate	addi \$1,\$2,100	$\$1 = \$2 + 100$	+ constant;
add unsigned	addu \$1,\$2,\$3	$\$1 = \$2 + \$3$	3 operands;
subtract unsign.	subu \$1,\$2,\$3	$\$1 = \$2 - \$3$	3 operands;
add imm. unsign.	addiu \$1,\$2,10	$\$1 = \$2 + 10$	+ constant;
multiply	mult \$2,\$3	Hi, Lo = $\$2 \times \3	64-bit signed product
multiply unsign.	multu \$2,\$3	Hi, Lo = $\$2 \times \3	64-bit unsigned product
divide	div \$2,\$3	Lo = $\$2 \div \3 , Hi = $\$2 \bmod \3	Lo = quotient, Hi = remainder
divide unsign.	divu \$2,\$3	Lo = $\$2 \div \3 , Hi = $\$2 \bmod \3	Unsigned quotient & remainder
Move from Hi	mfhi \$1	$\$1 = \text{Hi}$	Used to get copy of Hi
Move from Lo	mflo \$1	$\$1 = \text{Lo}$	Used to get copy of Lo

COMP3211/9211

2011S1 wk1_1 P49

MIPS Logical Instructions

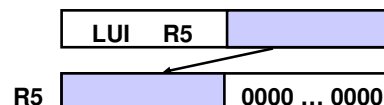
<u>Instruction</u>	<u>Example</u>	<u>Meaning</u>	<u>Comment</u>
and	and \$1,\$2,\$3	$\$1 = \$2 \& \$3$	3 reg. operands; Logical AND
or	or \$1,\$2,\$3	$\$1 = \$2 \$3$	3 reg. operands; Logical OR
xor	xor \$1,\$2,\$3	$\$1 = \$2 \oplus \$3$	3 reg. operands; Logical XOR
nor	nor \$1,\$2,\$3	$\$1 = \sim(\$2 \$3)$	3 reg. operands; Logical NOR
and immediate	andi \$1,\$2,10	$\$1 = \$2 \& 10$	Logical AND reg, constant
or immediate	ori \$1,\$2,10	$\$1 = \$2 10$	Logical OR reg, constant
xor immediate	xori \$1, \$2,10	$\$1 = \sim\$2 \& \sim 10$	Logical XOR reg, constant
shift left logical	sll \$1,\$2,10	$\$1 = \$2 \ll 10$	Shift left by constant
shift right logical	srl \$1,\$2,10	$\$1 = \$2 \gg 10$	Shift right by constant
shift right arithm.	sra \$1,\$2,10	$\$1 = \$2 \gg 10$	Shift right (sign extend)
shift left logical	sllv \$1,\$2,\$3	$\$1 = \$2 \ll \$3$	Shift left by variable
shift right logical	srlv \$1,\$2, \$3	$\$1 = \$2 \gg \$3$	Shift right by variable
shift right arithm.	srav \$1,\$2, \$3	$\$1 = \$2 \gg \$3$	Shift right arith. by variable

COMP3211/9211

2011S1 wk1_1 P50

MIPS Data Transfer Instructions

<u>Instruction</u>	<u>Comment</u>	<u>Meaning</u>
SW R3, 500(R4)	Store word	Mem[R4 + 500] ← R3
SH R3, 502(R2)	Store half	
SB R2, 41(R3)	Store byte	
LW R1, 30(R2)	Load word	R1 ← Mem[R2 + 30]
LH R1, 40(R3)	Load halfword	
LHU R1, 40(R3)	Load halfword unsigned	
LB R1, 40(R3)	Load byte	
LBU R1, 40(R3)	Load byte unsigned	
LUI R1, 40	Load Upper Immediate (16 bits shifted left by 16)	



COMP3211/9211

2011S1 wk1_1 P51

MIPS jump, branch, compare Instructions

<u>Instruction</u>	<u>Example</u>	<u>Meaning</u>
branch on equal	beq \$1,\$2,100	if ($\$1 == \2) go to PC+4+100 <i>Equal test; PC relative branch</i>
branch on not eq.	bne \$1,\$2,100	if ($\$1 \neq \2) go to PC+4+100 <i>Not equal test; PC relative</i>
set on less than	slt \$1,\$2,\$3	if ($\$2 < \3) $\$1=1$; else $\$1=0$ <i>Compare less than; 2's comp.</i>
set less than imm.	slti \$1,\$2,100	if ($\$2 < 100$) $\$1=1$; else $\$1=0$ <i>Compare < constant; 2's comp.</i>
set less than uns.	sltu \$1,\$2,\$3	if ($\$2 < \3) $\$1=1$; else $\$1=0$ <i>Compare less than; natural numbers</i>
set l. t. imm. uns.	sltiu \$1,\$2,100	if ($\$2 < 100$) $\$1=1$; else $\$1=0$ <i>Compare < constant; natural numbers</i>
jump	j 10000	go to 10000 <i>Jump to target address</i>
jump register	jr \$31	go to \$31 <i>For switch, procedure return</i>
jump and link	jal 10000	$\$31 = \text{PC} + 4$; go to 10000 <i>For procedure call</i>

COMP3211/9211

2011S1 wk1_1 P52

MIPS Instruction Set

- Refer to inside jacket of textbook, and Ch 2

Homework

- Browse the course website
- Skim through textbook
- Find your project partners