


COMPUTER ORGANIZATION AND DESIGN
The Hardware/Software Interface



EECS 2021

Computer Organization Fall 2015


*The slides are based on the publisher slides
and contribution from Profs Amir Asif and
Peter Lian*
*The slides will be modified, annotated,
explained on the board, and sometimes
corrected in the class*

Based on slides by the author and prof.
Mary Jane Irwin of PSU.

Representing Instructions

§2.5 Representing Instructions in the Computer

- Instructions are encoded in binary
 - Called machine code
- MIPS instructions
 - Encoded as 32-bit instruction words
 - Small number of formats encoding operation code (opcode), register numbers, ...
 - Regularity!
- Register numbers
 - \$t0 – \$t7 are reg's 8 – 15
 - \$t8 – \$t9 are reg's 24 – 25
 - \$s0 – \$s7 are reg's 16 – 23



Chapter 2 — Instructions: Language of the Computer — 2

MIPS R-format Instructions

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

■ Instruction fields

- op: operation code (opcode)
- rs: first source register number
- rt: second source register number
- rd: destination register number
- shamt: shift amount (00000 for now)
- funct: function code (extends opcode, selects the specific variant of the operation specified in the opcode field)



Chapter 2 — Instructions: Language of the Computer — 3

R-format Example

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

add \$t0, \$s1, \$s2

special	\$s1	\$s2	\$t0	0	add
0	17	18	8	0	32 _{ten}
000000	10001	10010	01000	00000	100000

00000010001100100100000000100000₂ = 02324020₁₆



Chapter 2 — Instructions: Language of the Computer — 4

Hexadecimal

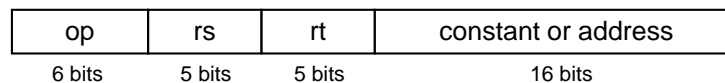
- Base 16
 - Compact representation of bit strings
 - 4 bits per hex digit

0	0000	4	0100	8	1000	c	1100
1	0001	5	0101	9	1001	d	1101
2	0010	6	0110	a	1010	e	1110
3	0011	7	0111	b	1011	f	1111

- Example: eca8 6420
 - 1110 1100 1010 1000 0110 0100 0010 0000



MIPS I-format Instructions



- Immediate arithmetic and load/store instructions
 - rt: destination -- rs source register number
 - Constant: -2^{15} to $+2^{15} - 1$
 - Address: offset added to base address in rs
- **Design Principle 4: Good design demands good compromises**
 - Different formats complicate decoding, but allow 32-bit instructions uniformly
 - Keep formats as similar as possible




MIPS I-format Instructions

op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

`addi $t0, $s1, 10`

addi	\$s1	\$t0	constant
8	17	8	10
001000	10001	01000	0000000000001010



Chapter 2 — Instructions: Language of the Computer — 7

Stored Program Computers

The BIG Picture

Processor

Memory

Accounting program (machine code)

Editor program (machine code)


C compiler (machine code)

Payroll data

Book text

Source code in C for editor program

- Instructions represented in binary, just like data
- Instructions and data stored in memory
- Programs can operate on programs
 - e.g., compilers, linkers, ...
- Binary compatibility allows compiled programs to work on different computers
 - Standardized ISAs



Chapter 2 — Instructions: Language of the Computer — 8

Logical Operations

- Instructions for bitwise manipulation

Operation	C	Java	MIPS
Shift left	<<	<<	sll
Shift right	>>	>>>	srl
Bitwise AND	&	&	and, andi
Bitwise OR			or, ori
Bitwise NOT	~	~	nor

- Useful for extracting and inserting groups of bits in a word



Shift Operations

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

- shamt: how many positions to shift
- Shift left logical
 - Shift left and fill with 0 bits
 - sll by i bits multiplies by 2^i
- Shift right logical
 - Shift right and fill with 0 bits
 - srl by i bits divides by 2^i (unsigned only)



AND Operations

- Useful to mask bits in a word
 - Select some bits, clear others to 0
- and \$t0, \$t1, \$t2

\$t2	0000 0000 0000 0000 0000 1101 1100 0000
\$t1	0000 0000 0000 0000 0011 1100 0000 0000
\$t0	0000 0000 0000 0000 0000 1100 0000 0000



Chapter 2 — Instructions: Language of the Computer — 11

OR Operations

- Useful to include bits in a word
 - Set some bits to 1, leave others unchanged
- or \$t0, \$t1, \$t2

\$t2	0000 0000 0000 0000 0000 1101 1100 0000
\$t1	0000 0000 0000 0000 0011 1100 0000 0000
\$t0	0000 0000 0000 0000 0011 1101 1100 0000



Chapter 2 — Instructions: Language of the Computer — 12

NOT Operations

- Useful to invert bits in a word
 - Change 0 to 1, and 1 to 0
- MIPS has NOR 3-operand instruction
 - $a \text{ NOR } b == \text{NOT} (a \text{ OR } b)$

`nor $t0, $t1, $zero` ←

Register 0: always
read as zero

\$zero	0000 0000 0000 0000 0000 0000 0000 0000
\$t1	0000 0000 0000 0000 0011 1100 0000 0000
\$t0	1111 1111 1111 1111 1100 0011 1111 1111

