

CSCI 210: Computer Architecture  
Lecture 9: Computer Representation of MIPS  
instructions 2

Stephen Checkoway

Oberlin College

Mar. 9, 2022

Slides from Cynthia Taylor

# Announcements

- Problem Set 2 due Friday
- Lab 1 due Sunday
- Office Hours Friday 13:30 – 14:30

# Representing Instructions

- MIPS instructions
  - Encoded as 32-bit instruction words
  - Small number of formats encoding operation code (opcode), register numbers, ...
  - Regularity!

	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
R-type	<b>opcode</b>	<b>rs</b>	<b>rt</b>	<b>rd</b>	<b>sa</b>	<b>funct</b>
I-type	<b>opcode</b>	<b>rs</b>	<b>rt</b>	<b>immediate</b>		
J-type	<b>opcode</b>	<b>target</b>				

# MIPS Instruction Formats

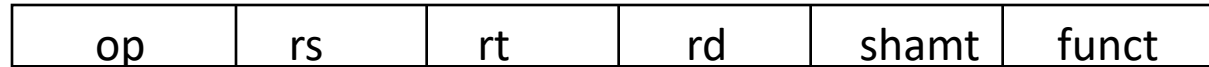
	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
R-type	<b>opcode</b>	<b>rs</b>	<b>rt</b>	<b>rd</b>	<b>sa</b>	<b>funct</b>
I-type	<b>opcode</b>	<b>rs</b>	<b>rt</b>	<b>immediate</b>		
J-type	<b>opcode</b>	<b>target</b>				

Which row contains correct examples of instructions with the given types?

	<b>R-type</b>	<b>I-type</b>
A	addi	sw
B	addi	sub
C	add	sw
D	add	sub
E	None of the above	

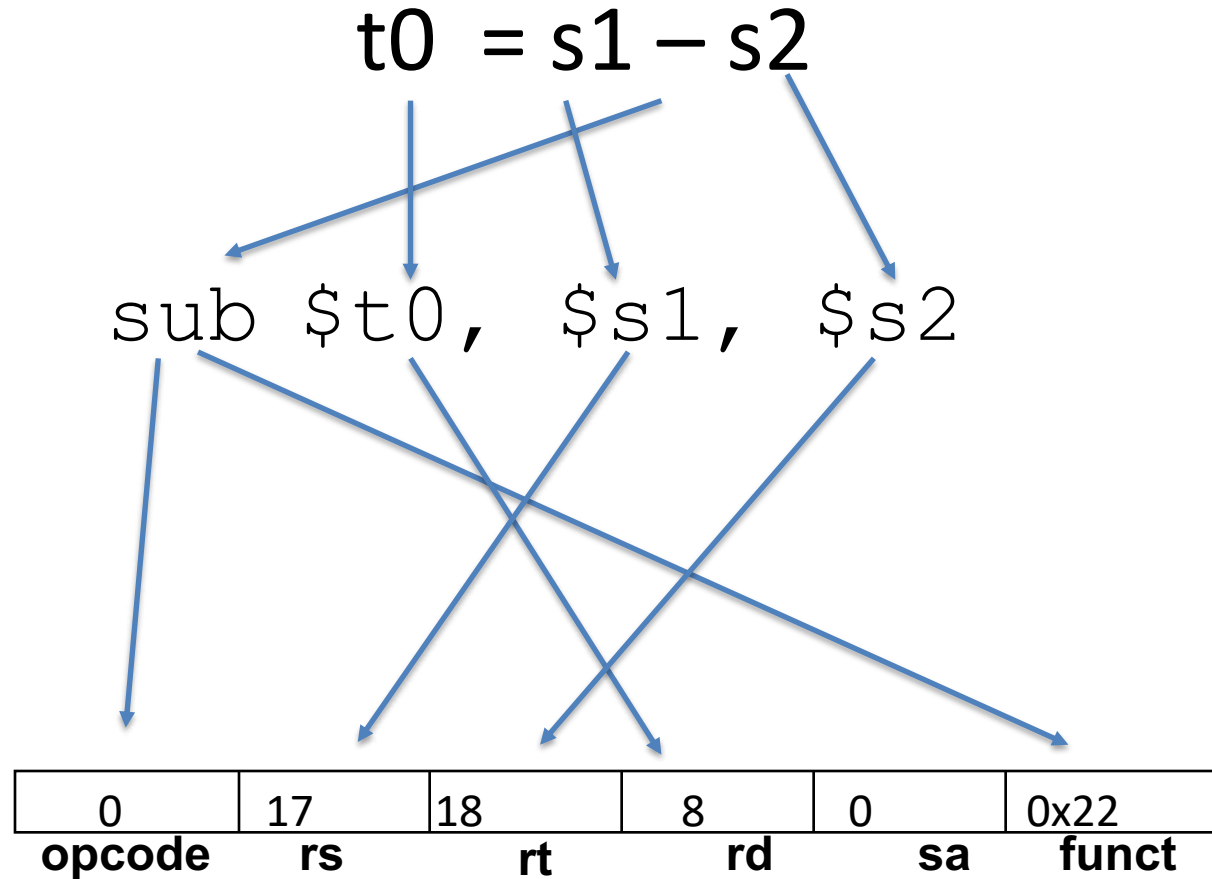
# MIPS Instruction Fields

MIPS fields are given names to make them easier to refer to



- op      6-bits opcode that specifies the operation
- rs      5-bits register file address of the first source operand
- rt      5-bits register file address of the second source operand
- rd      5-bits register file address of the result's destination
- shamt   5-bits shift amount (for shift instructions)
- funct   6-bits function code augmenting the opcode

# MIPS Arithmetic Instructions Format



# R-format Example



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

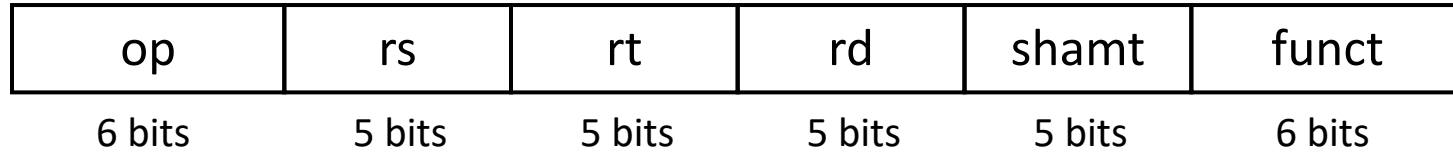
## CORE INSTRUCTION SET

NAME, MNEMONIC	FOR-MAT	OPERATION (in Verilog)	OPCODE / FUNCT (Hex)
Add	add	R R[rd] = R[rs] + R[rt]	(1) 0 / 20 <sub>hex</sub>
Add Immediate	addi	I R[rt] = R[rs] + SignExtImm	(1,2) 8 <sub>hex</sub>
Add Imm. Unsigned	addiu	I R[rt] = R[rs] + SignExtImm	(2) 9 <sub>hex</sub>
Add Unsigned	addu	R R[rd] = R[rs] + R[rt]	0 / 21 <sub>hex</sub>

NAME	NUMBER	USE
\$zero	0	The Constant Value 0
\$at	1	Assembler Temporary
\$v0-\$v1	2-3	Values for Function Results and Expression Evaluation
\$a0-\$a3	4-7	Arguments
\$t0-\$t7	8-15	Temporaries
\$s0-\$s7	16-23	Saved Temporaries
\$t8-\$t9	24-25	Temporaries
\$k0-\$k1	26-27	Reserved for OS Kernel
\$gp	28	Global Pointer
\$sp	29	Stack Pointer
\$fp	30	Frame Pointer
\$ra	31	Return Address

Convert this MIPS machine instruction to assembly:

000000 01110 10001 10010 00000 100010



Selection	Instruction
A	add \$s2, \$t7, \$s4
B	add \$s1, \$t6, \$s3
C	sub \$t6, \$s1, \$s2
D	sub \$s2, \$t6, \$s1
E	None of the above



# MIPS I-format Instructions



- Immediate arithmetic and load/store instructions
  - rt: destination or source register number
  - Constant:  $-2^{15}$  to  $+2^{15} - 1$  (or 0 to  $2^{16} - 1$  for some instructions)
  - offset: offset added to base address in rs

# Machine Language – I Format



- Load/Store Instruction Format:

`lw $t0, 24($s3)`

Load Linked	ll	I	R[rt] = M[R[rs]+SignExtImm]	(2,7)	30 <sub>hex</sub>
Load Upper Imm.	lui	I	R[rt] = {imm, 16'b0}		f <sub>hex</sub>
Load Word	lw	I	R[rt] = M[R[rs]+SignExtImm]	(2)	23 <sub>hex</sub>
Nor	nor	R	R[rd] = ~(R[rs]   R[rt])		0 / 27 <sub>hex</sub>

NAME	NUMBER	USE
\$zero	0	The Constant Value 0
\$at	1	Assembler Temporary
\$v0-\$v1	2-3	Values for Function Results and Expression Evaluation
\$a0-\$a3	4-7	Arguments
\$t0-\$t7	8-15	Temporaries
\$s0-\$s7	16-23	Saved Temporaries
\$t8-\$t9	24-25	Temporaries
\$k0-\$k1	26-27	Reserved for OS Kernel
\$gp	28	Global Pointer
\$sp	29	Stack Pointer
\$fp	30	Frame Pointer
\$ra	31	Return Address

# Machine Language – I Format



- Immediate Addition Instruction Format:

```
addi $t0, $s3, 26
```

## CORE INSTRUCTION SET

NAME, MNEMONIC	FOR-MAT	OPERATION (in Verilog)	OPCODE / FUNCT (Hex)
Add	add	R R[rd] = R[rs] + R[rt]	(1) 0 / 20 <sub>hex</sub>
Add Immediate	addi	I R[rt] = R[rs] + SignExtImm	(1,2) 8 <sub>hex</sub>
Add Imm. Unsigned	addiu	I R[rt] = R[rs] + SignExtImm	(2) 9 <sub>hex</sub>
Add Unsigned	addu	R R[rd] = R[rs] + R[rt]	0 / 21 <sub>hex</sub>

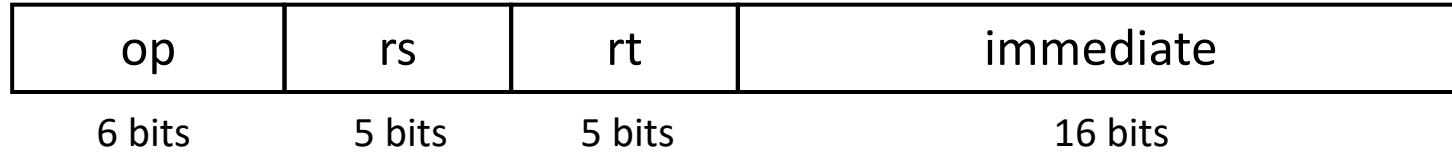
NAME	NUMBER	USE
\$zero	0	The Constant Value 0
\$at	1	Assembler Temporary
\$v0-\$v1	2-3	Values for Function Results and Expression Evaluation
\$a0-\$a3	4-7	Arguments
\$t0-\$t7	8-15	Temporaries
\$s0-\$s7	16-23	Saved Temporaries
\$t8-\$t9	24-25	Temporaries
\$k0-\$k1	26-27	Reserved for OS Kernel
\$gp	28	Global Pointer
\$sp	29	Stack Pointer
\$fp	30	Frame Pointer
\$ra	31	Return Address

Convert this MIPS assembly instruction to machine code

```
sw $t0, 32($s6)
```

Selection	Instruction
A	010101 11011 00100 0000 0000 0010 0000
B	101011 01000 10110 0000 0000 0010 0000
C	101011 10110 01000 0000 0000 0010 0000
D	000000 00010 00000 1010 1110 1100 1000
E	None of the above

# Sign-extend vs. zero-extend



- The immediate field of an I-format instruction is either sign-extended or zero-extended
  - sign extension: the sign bit (bit 15) is copied into bits 31–16
  - zero extension: 0 is placed into bits 31–16

- Opcode determines which occurs

Add Immediate	addi	I	$R[rt] = R[rs] + \text{SignExtImm}$	(1,2)	$8_{\text{hex}}$
Add Imm. Unsigned	addiu	I	$R[rt] = R[rs] + \text{SignExtImm}$	(2)	$9_{\text{hex}}$
Add Unsigned	addu	R	$R[rd] = R[rs] + R[rt]$		$0 / 21_{\text{hex}}$
And	and	R	$R[rd] = R[rs] \& R[rt]$		$0 / 24_{\text{hex}}$
And Immediate	andi	I	$R[rt] = R[rs] \& \text{ZeroExtImm}$	(3)	$c_{\text{hex}}$

# Disassemble the MIPS instruction 0xA20CFE0

Hint: Convert to binary, break into fields, and convert

- A. sb      \$s0, 65504(\$t4)
- B. sb      \$s0, -32(\$t4)
- C. sb      \$t4, 65504(\$s0)
- D. sb      \$t4, -32(\$s0)
- E. sb      \$t4(\$s0), 0xFFE0

What's the mnemonic for the MIPS instruction  
0x012A4025

- A. add
- B. addi
- C. or
- D. ori
- E. nor

# Reading

- Next lecture: Bit Level Operations
  - Section 2.6
- Problem Set 2 due Friday
- Lab 1 due Sunday