MIPS Instruction Formats

- All MIPS instructions are encoded in binary.
- All MIPS instructions are 32 bits long.

  (Note: some assembly langs do not have uniform length for all instructions)

  Examples:

  ```
  001000 10011010100000000000000100
  000010 00000000000000000100000001
  000000 10001100101000000000100000
  100011 10011010000000000000100000
  000100 01000000000000000000000101
  ```

- All instructions have an **opcode** (or **op**) that specifies the operation (first 6 bits).
- There are 32 registers. *(Need 5 bits to uniquely identify all 32.)*
- There are three instruction categories: I-format, J-format, and R-format (most common).

I-Format:

op	rs	rt	immediate	
001000	10011	01010	0000000000000100	**Example:** `addi $t2, $s3, 4`

J-Format:

op	address	
000010	00000000000000000100000001	**Example:** `j LOOP` (or `j 1028`)

R-Format:

op	rs	rt	rd	shamt	funct	
000000	10001	10010	10000	00000	100000	**Example:** `add $s0, $s1, $s2`

Typical Instruction Formats

Basic I-format Instructions

- Have 2 registers and a constant value immediately present in the instruction.
    - rs: source register (5 bits)
    - rt: destination register (5 bits)
    - immediate value (16 bits)

Instructions that follow typical format:

Name	Format	Layout						Example
		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
		op	rs	rt	immediate			
addi	I	8	2	1	100			addi $1, $2, 100
addiu	I	9	2	1	100			addiu $1, $2, 100
andi	I	12	2	1	100			andi $1, $2, 100
ori	I	13	2	1	100			ori $1, $2, 100
slti	I	10	2	1	100			slti $1, $2, 100
sltiu	I	11	2	1	100			sltiu $1, $2, 100

NOTE: *Order of components in machine code is different from assembly code. Assembly code order is similar to C, destination first. Machine code has source register, then destination.*

Example

```
addi $t2, $s3, 4        (registers 10 and 19)
```

op	rs	rt	immediate
8	19	10	4
001000	10011	01010	0000000000000100

- **Exceptions:** `beq`, `bne`, `lw`, `sw`, `lui` (See format exceptions)

J-format Instructions

- Have an address (part of one, actually) in the instruction.

Name	Format	Layout						Example
		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
		op	address					
j	J	2	2500					j 10000
jal	J	3	2500					jal 10000

Example

```
j LOOP
```

The address stored in a `j` instruction is 26 bits of the address associated with the specified label. The 26 bits are achieved by dropping the high-order 4 bits of the address and the low-order 2 bits (which would always be 00, since addresses are always divisible by 4).

```
address = low-order 26 bits of (addrFromLabelTable/4)
```

In the example above, if `LOOP` is at address `1028`, then the value stored in the machine instruction would be `257` (`257 = 1028/4`).

op	address
2	257
000010	00000000000000000100000001

Basic R-format Instructions

- Have op `0`. (all of them!)
- Also have:
  - rs: 1st register operand (register source) (5 bits)
  - rt: 2nd register operand (5 bits)
  - rd: register destination (5 bits)
  - shamt: shift amount (0 when not applicable) (5 bits)
  - funct: function code (identifies the specific R-format instruction) (6 bits)

Name	Format	Layout	Example

		6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
		op	rs	rt	rd	shamt	funct	
add	R	0	2	3	1	0	32	add $1, $2, $3
addu	R	0	2	3	1	0	33	addu $1, $2, $3
sub	R	0	2	3	1	0	34	sub $1, $2, $3
subu	R	0	2	3	1	0	35	subu $1, $2, $3
and	R	0	2	3	1	0	36	and $1, $2, $3
or	R	0	2	3	1	0	37	or $1, $2, $3
nor	R	0	2	3	1	0	39	nor $1, $2, $3
slt	R	0	2	3	1	0	42	slt $1, $2, $3
sltu	R	0	2	3	1	0	43	sltu $1, $2, $3

NOTE: op is 0, so funct disambiguates

Example

```
add $s0, $s1, $s2       (registers 16, 17, 18)
```

op	rs	rt	rd	shamt	funct
0	17	18	16	0	32
000000	10001	10010	10000	00000	100000

NOTE: Order of components in machine code is different from assembly code. Assembly code order is similar to C, destination first. Machine code has destination last.

C:	`a = b + c`
assembly code:	`add $s0, $s1, $s2    # add` **`rd`**`, rs, rt`
machine code:	`000000 10001 10010 10000 0000 100000`
	(`op rs rt` **`rd`** `shamt funct`)

[Previous Slide](#) *Alyce Brady, Kalamazoo College* [Format Exceptions](#)

[All Mappings](#)