#4 : Encoding MIPS Instructions

Computer Architecture 2021/2022

Ricardo Rocha

Computer Science Department, Faculty of Sciences, University of Porto

Instruction Set

The words of a computer's language are called **instructions** and the vocabulary of commands understood by a given architecture is called an **instruction set**. Common groups of instructions are:

- Arithmetic instructions
- Logical instructions
- Data transfer instructions
- Conditional branch instructions
- Unconditional jump instructions

Registers

MIPS has **32 registers**, numbered from 0 to 31, each with **32 bits**.

To identify a register in MIPS we thus need **5 bits (2^5=32)**.

Name	Register number	Usage
$zero	0	The constant value 0
$v0–$v1	2–3	Values for results and expression evaluation
$a0–$a3	4–7	Arguments
$t0–$t7	8–15	Temporaries
$s0–$s7	16–23	Saved
$t8–$t9	24–25	More temporaries
$gp	28	Global pointer
$sp	29	Stack pointer
$fp	30	Frame pointer
$ra	31	Return address

Instruction Format

MIPS instructions are encoded in binary, as **32-bit instruction words**, called **machine code**. The layout of an instruction is called the **instruction format**. Only 3 different formats exist.

Name	Fields						Comments
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions are 32 bits long
R-format	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	op	rs	rt	address/immediate			Transfer, branch, imm. format
J-format	op	target address					Jump instruction format

Instruction Fields

Name	Fields						Comments
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions are 32 bits long
R-format	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	op	rs	rt	address/immediate			Transfer, branch, imm. format
J-format	op	target address					Jump instruction format

MIPS fields have names to make them easier to discuss:

- **op** – basic operation of the instruction, traditionally called the opcode
- **rs** – the first register source operand
- **rt** – the second register source operand
- **rd** – the register destination operand, which gets the result of the operation
- **shamt** – shift amount to be used in shift instructions, zero otherwise
- **funct** – often called the function code, selects the specific variant of the operation in the opcode field

Instruction Opcodes

op(31:26):								
28–26 31–29	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	Rfmt	Bltz/gez	j	jal	beq	bne	blez	bgtz
1(001)	addi	addiu	slti	sltiu	ANDi	ORi	xORi	lui
2(010)	TLB	FlPt						
3(011)								
4(100)	lb	lh	lwl	lw	lbu	lhu	lwr	
5(101)	sb	sh	swl	sw			swr	
6(110)	lwc0	lwc1						
7(111)	swc0	swc1						

R-format Function Codes

op(31:26)=000000 (R-format), funct(5:0)								
2–0 5–3	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	shift left logical		shift right logical	sra	sllv		srlv	srav
1(001)	jump register	jalr			syscall	break		
2(010)	mfhi	mthi	mflo	mtlo				
3(011)	mult	multu	div	divu				
4(100)	add	addu	subtract	subu	and	or	xor	not or (nor)
5(101)			set l.t.	set l.t. unsigned				
6(110)								
7(111)								

From Assembly to Machine Code

Consider the instruction: **add $t0, $s1, $s2**. Its decimal representation is:

- **op** = 0 (arithmetic)
- **rs** = 17 ($s1)
- **rt** = 18 ($s2)
- **rd** = 8 ($t0)
- **shamt** = 0 (not used)
- **funct** = 32 (add)

0	17	18	8	0	32

And the binary representation is:

000000	10001	10010	01000	00000	100000
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

From Assembly to Machine Code

Name	Fields						Comments
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions are 32 bits long
R-format	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	op	rs	rt	address/immediate			Transfer, branch, imm. format
J-format	op	target address					Jump instruction format

In branch (I-format) and jump (J-format) instructions, the **address** and **target address** fields need to be shifted left 2 bits to correctly represent a valid instruction address (32-bits aligned).

beq $s1, $s2, 25 **(25<<2 = 100)**

j 2500 **(2500<<2 = 10000)**