

# Digital Design and Computer Architecture



David Money Harris & Sarah L. Harris



**Universidade Federal de Campina Grande**

**Departamento de Sistemas e Computação**

**Curso de Bacharelado em Ciência da Computação**

## **Organização e Arquitetura de Computadores I**

---

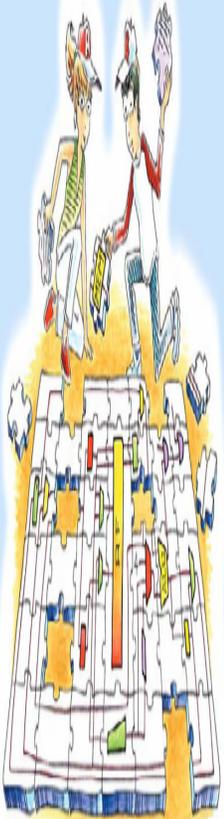
---

### **Organização e Arquitetura Básicas de Computadores (Parte I - Complementar)**

**Prof<sup>a</sup> Joseana Macêdo Fachine Régis de Araújo**

**[joseana@computacao.ufcg.edu.br](mailto:joseana@computacao.ufcg.edu.br)**

Carga Horária: 60 horas



# Tópicos

---

---

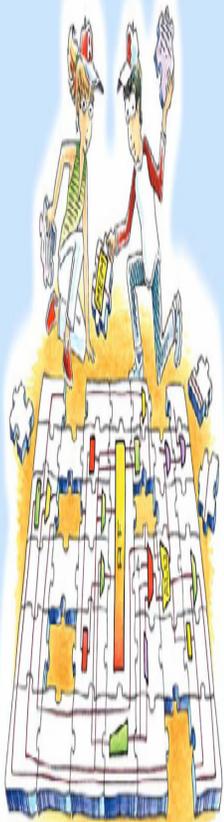
- Exemplo MIPS
  - Arquitetura do Conjunto de Instruções
    - Informações Complementares



# ISA - MIPS

Table B.1 Instructions, sorted by opcode

| Opcode                   | Name      | Description   | Operation   |
|--------------------------|-----------|---|---|
| 000000 (0)               | R-type    | all R-type instructions   | see Table B.2                                     |
| 000001 (1)<br>(rt = 0/1) | bltz/bgez | branch less than zero/<br>branch greater than or<br>equal to zero | if ([rs] < 0) PC = BTA/<br>if ([rs] ≥ 0) PC = BTA |
| 000010 (2)               | j         | jump  | PC = JTA  |
| 000011 (3)               | jal       | jump and link   | \$ra = PC+4, PC = JTA                             |
| 000100 (4)               | beq       | branch if equal   | if ([rs]==[rt]) PC = BTA                          |
| 000101 (5)               | bne       | branch if not equal   | if ([rs]!= [rt]) PC = BTA                         |
| 000110 (6)               | blez      | branch if less than or equal to zero                              | if ([rs] ≤ 0) PC = BTA                            |
| 000111 (7)               | bgtz      | branch if greater than zero                                       | if ([rs] > 0) PC = BTA                            |
| 001000 (8)               | addi      | add immediate   | [rt] = [rs] + SignImm                             |
| 001001 (9)               | addiu     | add immediate unsigned  | [rt] = [rs] + SignImm                             |
| 001010 (10)              | slti      | set less than immediate   | [rs] < SignImm ? [rt]=1 : [rt]=0                  |
| 001011 (11)              | sltiu     | set less than immediate unsigned                                  | [rs] < SignImm ? [rt]=1 : [rt]=0                  |



# ISA - MIPS

|             |                         |  |   |
|-------------|-------------------------|--|---|
| 001010 (10) | slti                    | set less than immediate                    | $[rs] < \text{SignImm} ? [rt]=1 : [rt]=0$               |
| 001011 (11) | sltiu                   | set less than immediate unsigned           | $[rs] < \text{SignImm} ? [rt]=1 : [rt]=0$               |
| 001100 (12) | andi                    | and immediate                              | $[rt] = [rs] \& \text{ZeroImm}$                         |
| 001101 (13) | ori                     | or immediate                               | $[rt] = [rs] \mid \text{ZeroImm}$                       |
| 001110 (14) | xori                    | xor immediate                              | $[rt] = [rs] \wedge \text{ZeroImm}$                     |
| 001111 (15) | lui                     | load upper immediate                       | $[rt] = \{\text{Imm}, 16'b0\}$                          |
| 010000 (16) | mfc0,<br>mtc0           | move from/to coprocessor 0                 | $[rt] = [rd] / [rd] = [rt]$<br>(rd is in coprocessor 0) |
| 010001 (17) | F-type                  | fop = 16/17: F-type instructions           | see Table B.3   |
| 010001 (17) | bc1f/bc1t<br>(rt = 0/1) | fop = 8: branch if fpcond is<br>FALSE/TRUE | if (fpcond == 0) PC = BTA/<br>if (fpcond == 1) PC = BTA |
| 100000 (32) | lb                      | load byte                                  | $[rt] = \text{SignExt}([Address]_{7:0})$                |
| 100001 (33) | lh                      | load halfword                              | $[rt] = \text{SignExt}([Address]_{15:0})$               |
| 100011 (35) | lw                      | load word                                  | $[rt] = [Address]$                                      |
| 100100 (36) | lbu                     | load byte unsigned                         | $[rt] = \text{ZeroExt}([Address]_{7:0})$                |
| 100101 (37) | lhu                     | load halfword unsigned                     | $[rt] = \text{ZeroExt}([Address]_{15:0})$               |
| 101000 (40) | sb                      | store byte                                 | $[Address]_{7:0} = [rt]_{7:0}$                          |



David Money Harris & Sarah L. Harris

# ISA - MIPS

**Table B.1** Instructions, sorted by opcode—Cont'd

| Opcode      | Name | Description                    | Operation                        |
|-------------|------|--------------------------------|----------------------------------|
| 101001 (41) | sh   | store halfword                 | $[Address]_{15:0} = [rt]_{15:0}$ |
| 101011 (43) | sw   | store word                     | $[Address] = [rt]$               |
| 110001 (49) | lwc1 | load word to FP coprocessor 1  | $[ft] = [Address]$               |
| 111001 (56) | swc1 | store word to FP coprocessor 1 | $[Address] = [ft]$               |

**Table B.2** R-type instructions, sorted by funct field

| Funct      | Name | Description                  | Operation   |
|------------|------|------------------------------|---|
| 000000 (0) | sll  | shift left logical           | $[rd] = [rt] \ll \text{shamt}$                            |
| 000010 (2) | srl  | shift right logical          | $[rd] = [rt] \gg \text{shamt}$                            |
| 000011 (3) | sra  | shift right arithmetic       | $[rd] = [rt] \ggg \text{shamt}$                           |
| 000100 (4) | sllv | shift left logical variable  | $[rd] = [rt] \ll [rs]_{4:0}$<br>assembly: sllv rd, rt, rs |
| 000110 (6) | srlv | shift right logical variable | $[rd] = [rt] \gg [rs]_{4:0}$<br>assembly: srlv rd, rt, rs |



# ISA - MIPS

|             |         |                                 |  |
|-------------|---------|---------------------------------|--|
| 000111 (7)  | srav    | shift right arithmetic variable | $[rd] = [rt] \ggg [rs]_{4:0}$<br>assembly: srav rd, rt, rs |
| 001000 (8)  | jr      | jump register                   | $PC = [rs]$  |
| 001001 (9)  | jalr    | jump and link register          | $\$ra = PC + 4, PC = [rs]$                                 |
| 001100 (12) | syscall | system call                     | system call exception                                      |
| 001101 (13) | break   | break                           | break exception  |
| 010000 (16) | mfhi    | move from hi                    | $[rd] = [hi]$  |
| 010001 (17) | mthi    | move to hi                      | $[hi] = [rs]$  |
| 010010 (18) | mflo    | move from lo                    | $[rd] = [lo]$  |
| 010011 (19) | mtlo    | move to lo                      | $[lo] = [rs]$  |
| 011000 (24) | mult    | multiply                        | $\{[hi], [lo]\} = [rs] \times [rt]$                        |
| 011001 (25) | multu   | multiply unsigned               | $\{[hi], [lo]\} = [rs] \times [rt]$                        |
| 011010 (26) | div     | divide                          | $[lo] = [rs]/[rt],$<br>$[hi] = [rs]\%[rt]$                 |



David Money Harris & Sarah L. Harris

# ISA - MIPS

Table B.2 R-type instructions, sorted by funct field—Cont'd

| Funct       | Name | Description            | Operation                                   |
|-------------|------|------------------------|---|
| 011011 (27) | divu | divide unsigned        | $[lo] = [rs]/[rt]$ ,<br>$[hi] = [rs]\%[rt]$ |
| 100000 (32) | add  | add                    | $[rd] = [rs] + [rt]$                        |
| 100001 (33) | addu | add unsigned           | $[rd] = [rs] + [rt]$                        |
| 100010 (34) | sub  | subtract               | $[rd] = [rs] - [rt]$                        |
| 100011 (35) | subu | subtract unsigned      | $[rd] = [rs] - [rt]$                        |
| 100100 (36) | and  | and                    | $[rd] = [rs] \& [rt]$                       |
| 100101 (37) | or   | or                     | $[rd] = [rs] \mid [rt]$                     |
| 100110 (38) | xor  | xor                    | $[rd] = [rs] \wedge [rt]$                   |
| 100111 (39) | nor  | nor                    | $[rd] = \sim([rs] \mid [rt])$               |
| 101010 (42) | slt  | set less than          | $[rs] < [rt] ? [rd] = 1 : [rd] = 0$         |
| 101011 (43) | sltu | set less than unsigned | $[rs] < [rt] ? [rd] = 1 : [rd] = 0$         |



# ISA - MIPS

Table B.3 F-type instructions (fop = 16/17)

| Funct       | Name            | Description                      | Operation                          |
|-------------|-----------------|----------------------------------|------------------------------------|
| 000000 (0)  | add.s/add.d     | FP add                           | $[fd] = [fs] + [ft]$               |
| 000001 (1)  | sub.s/sub.d     | FP subtract                      | $[fd] = [fs] - [ft]$               |
| 000010 (2)  | mul.s/mul.d     | FP multiply                      | $[fd] = [fs] * [ft]$               |
| 000011 (3)  | div.s/div.d     | FP divide                        | $[fd] = [fs]/[ft]$                 |
| 000101 (5)  | abs.s/abs.d     | FP absolute value                | $[fd] = ([fs] < 0) ? [-fs] : [fs]$ |
| 000111 (7)  | neg.s/neg.d     | FP negation                      | $[fd] = [-fs]$                     |
| 111010 (58) | c.seq.s/c.seq.d | FP equality comparison           | fpcond = ( $[fs] == [ft]$ )        |
| 111100 (60) | c.lt.s/c.lt.d   | FP less than comparison          | fpcond = ( $[fs] < [ft]$ )         |
| 111110 (62) | c.le.s/c.le.d   | FP less than or equal comparison | fpcond = ( $[fs] \leq [ft]$ )      |