MIPS Instruction Set

32-bit MIPS RISC CPU – 1er Laboratorio de EE604 O-P - Introducción a Microcontroladores – UNI-FIEE 2025-2

Register Arithmetic Operations (R-format)			31-26	25-21	20-16	15-11	10-6	5-0
add	**rd, rs, rt**	rd = rs + rt (signed)	000000	rs	rt	rd	00000	100000
addu	**rd, rs, rt**	rd = rs + rt (unsigned)	000000	rs	rt	rd	00000	100001
sub	**rd, rs, rt**	rd = rs – rt (signed)	000000	rs	rt	rd	00000	100010
subu	**rd, rs, rt**	rd = rs – rt (unsigned)	000000	rs	rt	rd	00000	100011
slt	**rd, rs, rt**	rd = (rs < rt) ? 1:0 (signed)	000000	rs	rt	rd	00000	101010
sltu	**rd, rs, rt**	rd = (rs < rt) ? 1:0 (unsigned)	000000	rs	rt	rd	00000	101011

Register Logic Operations (R-format)			31-26	25-21	20-16	15-11	10-6	5-0
and	**rd, rs, rt**	rd = rs & rt	000000	rs	rt	rd	00000	100100
or	**rd, rs, rt**	rd = rs | rt	000000	rs	rt	rd	00000	100101
xor	**rd, rs, rt**	rd = rs ⊕ rt	000000	rs	rt	rd	00000	100110
nor	**rd, rs, rt**	rd = ~(rs | rt)	000000	rs	rt	rd	00000	100111

Immediate Operations (I-format)			31-26	25-21	20-16	15-0
addi	**rt, rs, const16**	rt = rs + SignExt(const16)	001000	rs	rt	const16
addiu	**rt, rs, const16**	rt = rs + (0000_{16}::const16)	001001	rs	rt	const16
slti	**rt, rs, const16**	rt = rs < SignExt(const16) ? 1:0	001010	rs	rt	const16
sltiu	**rt, rs, const16**	rt = rs < (0000_{16}::const16) ? 1:0	001011	rs	rt	const16
andi	**rt, rs, const16**	rt = rs & (0000_{16}::const16)	001100	rs	rt	const16
ori	**rt, rs, const16**	rt = rs | (0000_{16}::const16)	001101	rs	rt	const16
xori	**rt, rs, const16**	rt = rs ⊕ (0000_{16}::const16)	001110	rs	rt	const16

Load & Store Operations (I-format)			31-26	25-21	20-16	15-0
load	**rt, off16(rs)**	rt = mem[rs + SignExt(off16)]	100011	rs	rt	off16
store	**rt, off16(rs)**	mem[rs + SignExt(off16)] = rt	101011	rs	rt	off16

Branch Operations (I-format)			31-26	25-21	20-16	15-0
beq	**rs, rt, off16**	if rs = rt, PC= PC + 4*SignExt(off16)	000100	rs	rt	off16

Jump Operations (J-format)			31-26	25-0
j	**addr26**	PC= PC[31:28]::addr26::0^2	000010	addr26
jal	**addr26**	r31= PC, PC= PC[31:28]::addr26::0^2	000011	addr26

Jump Operation (R-format)			31-26			25-0		
jr	**rs**	PC = rs	000000	rs	00000	00000	00000	001000

Assembler Specifics

32-bit MIPS RISC CPU – 1er Laboratorio de EE604 O-P Introducción a Microcontroladores – UNI-FIEE 2025-2

Organization:

1. Each line will contain at most 1 instruction as specified by the Instruction Set table.
2. All operands must be separated by a comma ",".
3. All lines containing an instruction must start with the operand mnemonic. Examples:
   a. YES: add r1, r2, r3
   b. NO: <space> add r1, r2, r3
   c. NO: <tab> add r1, r2, r3
   d. NO: / add r1, r2, r3
4. Comments can be on any line but must be preceded by "//". Examples:
   a. YES:
   ```
   addi  r3, r0, 0   //add r0 + 0
   beq   r3, r2, endloopa  //branch to endloopa
   ```
   b. YES:
   ```
   //add r0 + 0
   addi  r3, r0, 0
   //branch to endloopa
   beq   r3, r2, endloopa
   ```

Spacing:

1. All operand mnemonics must be followed by a space " " or tab "/t".
2. Additional spacing using " " and "/t" can be included between operands and after the instruction. Examples:
   a. YES: add <space></t>r1, r2, r3
   b. YES: add r1,<space> r2, r3
   c. YES: add r1, r2, </t></t><space> r3
   d. NO: <space> add r1, r2, r3
3. Blank lines in between instructions can be included. Example:
   a.
   ```
   add  r1, r2, r3

   and  r11, r4, r5
   load r4, 0(r6)
   ```

Branches and Jumps:

1. Labels must be in their own line and be terminated by a ":". Example:
   a.
   ```
   addi  r3, r0, 0
   beq   r3, r2, endloopa
   addi  r4, r0, 0
   endloopa:
   add   r3, r2, r4
   ```
2. Branch/Jump instructions will accept any of the following input formats for the offset/address (immediate values must be in decimal):
   a. YES: beq r1, r2, label
   b. YES: beq r1, r2, 46
   c. YES: beq r1, r2, -7
   d. YES: j label
   e. YES: j 46