# Arch: MIPS ISA

Alex Chi

*Update: March 24, 2020*

## Contents

# 1 MIPS Instruction Fields

```
| op (6) | rs (5) | rt (5) | rd (5) | shamt (5) | funct (6) |
```

# 2 MIPS Design Principles

- simplicity favors regularity
  - fixed-size instructions
  - small number of instruction formats
  - opcode always 6 bits
- smaller
  - limited instruction set
  - limited number of registers (but still more than CISC)
  - limited number of addressing modes
- make common case fast
  - arithmetic operands from register file
  - allow instructions to contain immediate operands (we hard-encode immediate into a field)
- only 3 instruction types
  - (RISC-V has 6)
  - (countless for x86)

# 3 MIPS-32 ISA

## 3.1 Instruction Categories

- computational
- load/store (no modify memory in 1 instruction)
- jump and branch
- floating point
- memory management
- special

## 3.2 Registers

- GP (general-purpose registers)
  - user can use them for whatever things
- PC
- HI/LO
  - e.g. when multiplying

## 3.3 3 Instruction Formats

- R format `|op|rs|rt|rd|sa|funct|`
- I format `|op|rs|rt| immediate |`
- J format `|op| jump target |`

## 3.4 MIPS Register File

- input: src1, src2, dst, write data, write control
- output: src1, src2
- how to run `mov r1, r2` in a cycle?
  - src1: r1
  - src2: 0
  - dst: r2
  - write data: src1
  - write control: 1

## 3.5 Register Convention

- #0 zero = 0
- #1 at, reserved for assembler
- #28 gp, global pointer
- #29 sp, stack pointer
- #30 fp, frame pointer
- #31, ra, return address, auto rewritten by hardware when calling "jal"

## 3.6 Arithmetic Instruction

- add dst, src1, src2 (R format)
- shift by constant and shift by register

## 3.7 Load Instruction

- `lw r1, 24(s3)`
- destination in rt field
- offset limited to 16bits ($-2^{15}$ - $2^{15}$)
- there are also `lw, lb`
- alignment restriction: access a word requires natural word boundary
- (compare to RISC-V, it uses `funct3` field for indicating load word/byte)

### 3.7.1 Endian

- big endian: MIPS, IBM 360/370
- little endian: Intel 80x86
- little endian: lower address stores least significant bits
- if only load a byte: zero extension
- store only touches given address

## 3.8 Immediate Instruction

- `addi \$sp, \$sp, 4`
- `slti \$t0, \$s2, 15`
- an extension is necessary for upper 16 bits
- immediate format limits value to $2^{15} - 1to - 2^{15}$

## 3.9 Control Flow Instruction

- `bne \$s0, \$s1, Lbl` go to Lbl if $s0 \neq s1$
- `beq \$s0, \$s1, Lbl` go to Lbl if eq
- next instruction is relative address $pc + offset + 4$ or $pc + 4$
- a side note: RISC-V use $pc + offset$ (instead of +4) for branch destination.

## 3.10 Set Less Than

- `slt \$to, \$s0, \$s1` if s0 < s1 set t0 to 1
- alternative version
  - `sltu` (slt unsigned)
  - `slti` (slt immediate)
  - `sltiu`
- this can be used to implement jump less than
- for MIPS, assembler will expand blt, bge to slt and beq, and `at` (`r1`) is used to save this value
- a side note: RISC-V has encoding for blt, bge, etc.

## 3.11 Unconditional Jump

- j format (26-bit address)
- `j label`
- 4 bit from upper PC, 26 bits, and 00 ($\times 4$)
- jump to absolute address
- branch far away: assembler will use unconditional jump
- or jump register