The MIPS instruction set is covered in detail in the SPIM handout. In all instructions, Src2 can either be a register or a 16-bit immediate value. I left out the immediate forms because the MIPS assembler takes care of converting the general forms into the immediate forms when appropriate.

Logical operations and, or, xor, not.

Unary operations abs, neg, not.

Slide 6–1

Slide 6–2

Slide 6–3

MIPS Instructions

MIPS Instructions...

Arithmetic:
- add, sub, subu
- addi, slti, sltiu, sltu, srl, sra, sar, ori, lui, lui

Shift/rotate:
- sll, srl, sra, rol, ror

Multiplication:
- mul, mull, mulh, mulf, multi
  - multi: multiplies the contents of the two registers, putting the low-order word of the result in the register R0, and the high-order word in R1.
  - Muli: Multiplies R1 by R2 and stores the result in Rdest. These are synthetic instructions.
Load/Store Instructions ...

- The lw instruction, and its variants, load a value from memory into register Rdest. The memory address can be specified in one of the ways shown above.
- A load instruction affects the entire register, even if you only load a byte into it (for example). The high-order bytes are either
  1. cleared to zero (for an unsigned load), or
  2. sign-extended (for a signed load).

Load/Store Instructions ...

- **sw**
  Syntax: `lw Rsoc, address`
- Stores the value in register Rsoc into memory at the specified address.
- Note that if you store one byte, only that byte of memory is affected. Stores have different behavior from loads in that sense.

MIPS Instructions ...

Division

`div, divu. Syntax: op Rsoc1, Rsoc2.`
- Divides the contents of Rsoc1 by Rsoc2, and leaves the quotient in 'lo' and the remainder in 'hi'.
`div, divu. Syntax: op Rdest, Rsoc1, Rsoc2.`
- Divides the contents of Rsoc1 by Rsoc2, and leaves the quotient in Rdest.

Load/Store Instructions

- **lw** Syntax: `lw Rdest, address`
- address can be one of:
  1. (register)
  2. imm
  3. imm(register)
  4. symbol
  5. symbol±imm
  6. symbol±imm(register).
We'll cover addressing modes in more detail in a later topic.
Branch and Jump Instructions

- The syntax varies for different branch instructions:
  1. Some, like `J`, have a single label argument and always branch to that label.
  2. Others compare two registers and branch if the comparison is true.
  3. Some test a single register and branch if the test is true.

Misc. Instructions

- `li Rd, imm` loads the immediate value imm into Rd.
- `lui Rd, imm` loads the low-order halfword of imm into the high-order halfword of Rd. The low-order bits of Rd are set to zero.

Misc. Instructions... 

- `move Rddest, Rs, Rsrc` sets Rddest to Rs.
- `la Rsdest, address` loads the value of address into Rddest, not the contents of memory at that address.
- The following instructions are used:
  1. `fmihi Rddest`
  2. `fmllo Rddest`
  3. `fmihi Rsrc`
  4. `fmllo Rsrc`

Readings and References

- SPIM Manual.