ECE 251: Computer Architecture

Week 02 Notes - Instruction Set Architecture (ISA)

Prof Rob Marano

Spring 2026

1. The Language of the Computer

- **Instruction Set**: The vocabulary of commands understood by a computer.
- **Hardware Operations**: Every computer must perform basic arithmetic.

Design Principle 1:

Simplicity favors regularity.

- Fixed format instructions simplify decoding.
- MIPS arithmetic instructions **always** have exactly three operands:
  - `add $t0, $t1, $t2        ($t0 = $t1 + $t2)`
  - `sub $v0, $a0, $a1        ($v0 = $a0 - $a1)`

2. The Operands

- **Registers**: The hardware primitives used for actively storing variables.
- MIPS has precisely **32 registers**, each exactly **32-bits** wide (a *word*).

Design Principle 2:

Smaller is faster.

- A smaller number of registers allows for faster clock cycles than a massive, sprawling array.
- Complex data structures (like enormous Arrays or Structs) cannot fit in 32 registers, so they are kept in **Main Memory**.

3. Data Transfer & Immediates

- **Data Transfer Instructions**: Move data between Memory and Registers.
  - `lw` (load word): Copy heavily-used data *from* memory to register.
  - `sw` (store word): Copy heavily-used data *from* register back to memory.
- MIPS requires data alignment (words must start at addresses that are multiples of 4).
- **Immediate Operands**: Constants embedded directly into the instruction.
  - `addi $s1, $s2, 100`

Design Principle 3:

Make the common case fast.

- Embedding small, frequently used constants directly avoids slow memory loads.

4. Representing Instructions in MIPS

Instructions are compiled into **32-bit fixed-length** binary machine code.

- **R-type (Register)**: Used for pure arithmetic.
  - `[op, rs, rt, rd, shamt, funct]`
- **I-type (Immediate)**: Used for data transfer, constants, and branches.
  - `[op, rs, rt, constant/address]`
- **J-type (Jump)**: Used for unconditional jumps.
  - `[op, address]`

5. Logical Operations

Operations uniquely suited to manipulate boolean bits within valid words.

- **Shift Instructions**:
  - `sll` (shift left logical): Effectively multiples a value by identical powers of 2.
  - `srl` (shift right logical): Effectively divides unsigned values by identical powers of 2.
- **Bitwise Logic**:
  - `and`, `or`, `nor`, `xor`
  - Operate structurally bit-by-bit. Crucial for masking, testing flags, and setting isolated bits.