Intro To MIPS

The process or contains some amount of local storage known as the Register

- A process has registers and the ALU
- Registers are where you store values that are currently being worked on
- The values stored in the registers are sent to the alu to be added, subtracted, etc.. the result is stored back in the register

Memory

- Modeled in continuous space
- Every byte in the memory is associated with an index, called the address
- We can read and write in the memory
- Give the address to the memory hardware, we can read the content in that byte
- Given the address and a byte value, we can modify the content in the memory at that address

Program and Data

- consists of instructions and data, both stored in memory
- Instructions in binary

Machine Code

- It is binary!
- The instruction set constitutes the vocabulary of the machine. These are the words understood by the machine itself
- To work on the machine we need a translator
  To see more about translation visit Program Translation

Why learn Assembly Language?

- Knowing assembly illuminates concepts not only in computer organization but operating systems, compilers, parallel systems, etc.
- Understanding high-level constructs are implemented leads to more effective use of those structures.
  - Control constructs (if, do-while, etc)
  - Pointers
  - Parameter Passing

MIPS is a RISC (Reduced Instruction Set Computer) instruction set, meaning that it has simple and few instructions

- Originated in the early 1980's
- An acronym for Microprocessor without Interlocked Pipeline Stages

RISC Philosophy-

- fixed instruction lengths
- load-store instruction sets
- limited number of addressing modes
- limited number of operations

The Four ISA Design Principles

Simplicity favors regularity

- Consistent instruction size, instruction formats, data formats
- Erases implementation by simplifying hardware
  Smaller is Faster
- Fewer bits to access and modify
- Use the register file instead of slower memory
  Make the common case fast
- Small constants are common, thus small intermediate fields should be used
  Good design demands good compromises
- Compromise with special formats for important exceptions
- A long jump (beyond a small constant)