Lecture 2
Machine language
Let my machine talk to me...

CS 241: Foundations of Sequential Programs
Spring 2017

Troy Vasiga et al. (where Dan Holtby ∈ al)
University of Waterloo
Binary and Hex

Dan: Maybe some more example? I dunno, wing it.
Bit sequences matter!
Lots of ways of Linux-ifing your Windows machine:

- cygwin (and use
  `ssh -Y yourname@linux.student.cs.uwaterloo.ca`)
- Linux Live CD / USB
- install your own linux (Ubuntu is well supported and easy to use)
- PuTTY (linux.student.cs again)
- Windows 10 has bash now
Files

- a sequence of bytes
- `cat` tells some of the story, but not the whole story
- `xxd` tells the whole story
  - `-c 4` for one word per line
  - `-b` for binary instead of hex
Demo

- C code
- Machine Code (x64)
- JPEG
Thursday Dan, draw the diagram again!
Regards, Wednesday Dan
Machine Language

- Instructions to the machine
- MIPS: 18 different 32-bit instructions encoded in two basic instruction formats
MIPS as a Programming Language

- the language that the CPU speaks
- example: add $1, $2, $3
- human meaning
- computer meaning
Communication between CPU and RAM

- load
- store
Machine Cycle

- PC ← 0
- loop
  - fetch word from RAM whose address is in the PC
  - place that word in IR
  - PC ← PC + 4
  - decode and execute the instruction that is in IR
Examples

See website. In particular:

- add example
- add and lis example
- slt example
- beq example

Note: you are writing *subprograms* (for the most part) in this course, and thus you should always *return*. 
Functions

How do I function?

▶ jr to the first instruction of the function
▶ return using jr $31.
  ▶ need to set $31 so you can return
  ▶ how?? use jalr instead
Need to save $31 (and possibly other registers) somewhere.

- C / C++ have the *call stack*
- Can we just do that?
- Yes: $30 is initialized with the end of memory
Seems terrible to count instructions to change function calls each time you edit the file!

- It is
- Would be nice if we could make the computer count for us
- Labeled instructions