# Lecture 2

## Machine language

Let my machine talk to me...

*CS 241: Foundations of Sequential Programs*
Fall 2014

Troy Vasiga et al
University of Waterloo

## In Linux (and all operating systems)

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
- install your own linux (Ubuntu is well supported and easy to use)
- putty

# Grouping of Bits

- most common grouping is a byte: 8 bits
- Integers: unsigned $0 \ldots 2^8 \Leftrightarrow 0 \ldots 255$
  2's comp $\Rightarrow -128 \ldots 127$
- Characters:
  - ASCII: $26+26+10+10+\sqcup+\{+\ldots = < 128$

## Files

cat → output as ASCII characters
xxd → every bit
Interpretation is in the eye of the beholder.

- a sequence of bytes
- `cat` tells some of the story, but not the whole story
- `xxd` tells the whole story

## Larger Groupings of Bits

"Standard ASCII" ⇒ 128 characters

"extended ASCII" ⇒ 256 characters

- → typically £ ¥ ÷ ¿
- → non-standard

Unicode (UTF-8)

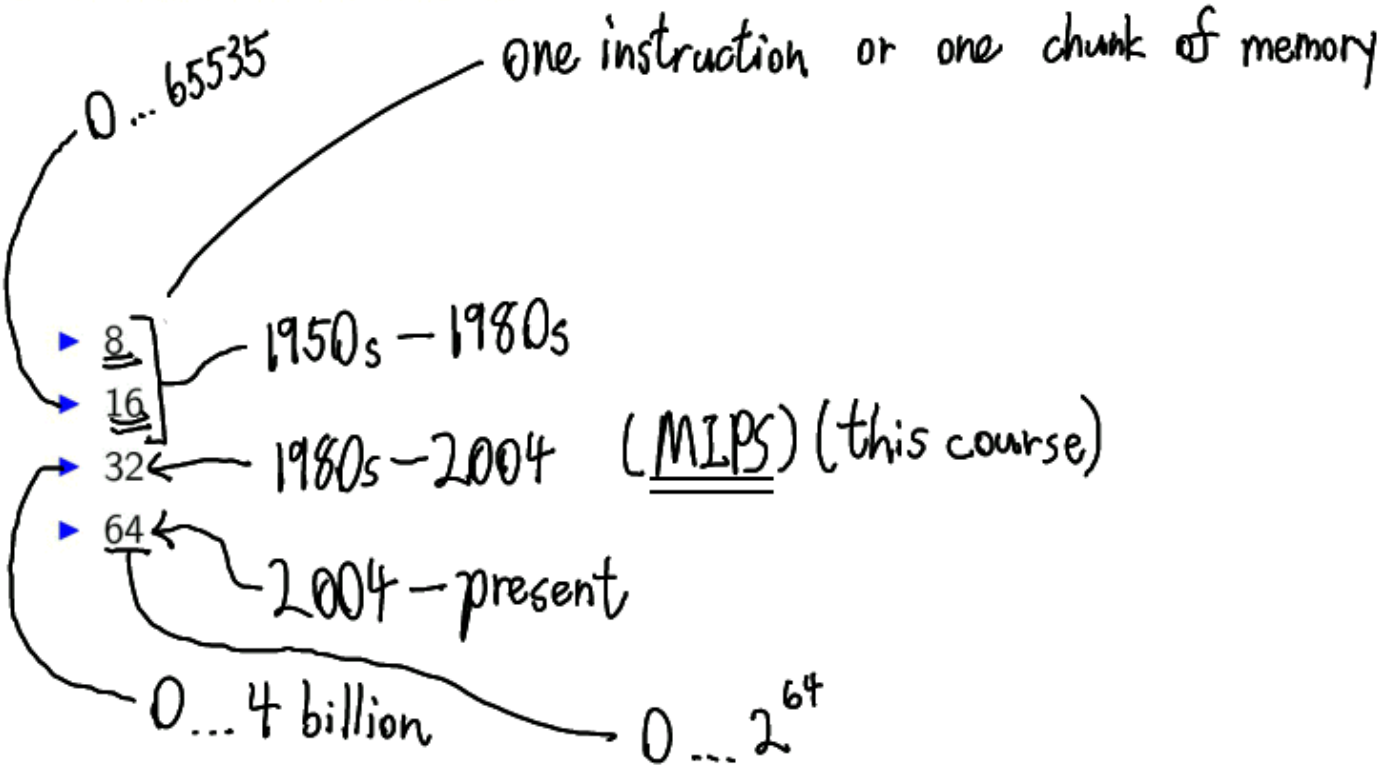
- → variable length
- → length part: e.g. 6 ⇒ next 6 bytes: character

$(2^8)^6 = 2^{48}$

$= (2^{10})^4 \cdot 2^8$

$\approx 256\,000\,000\,000\,000$

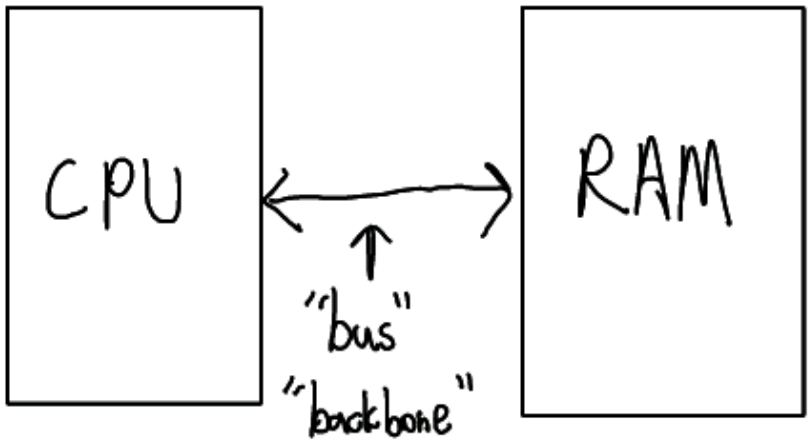
## A word about words



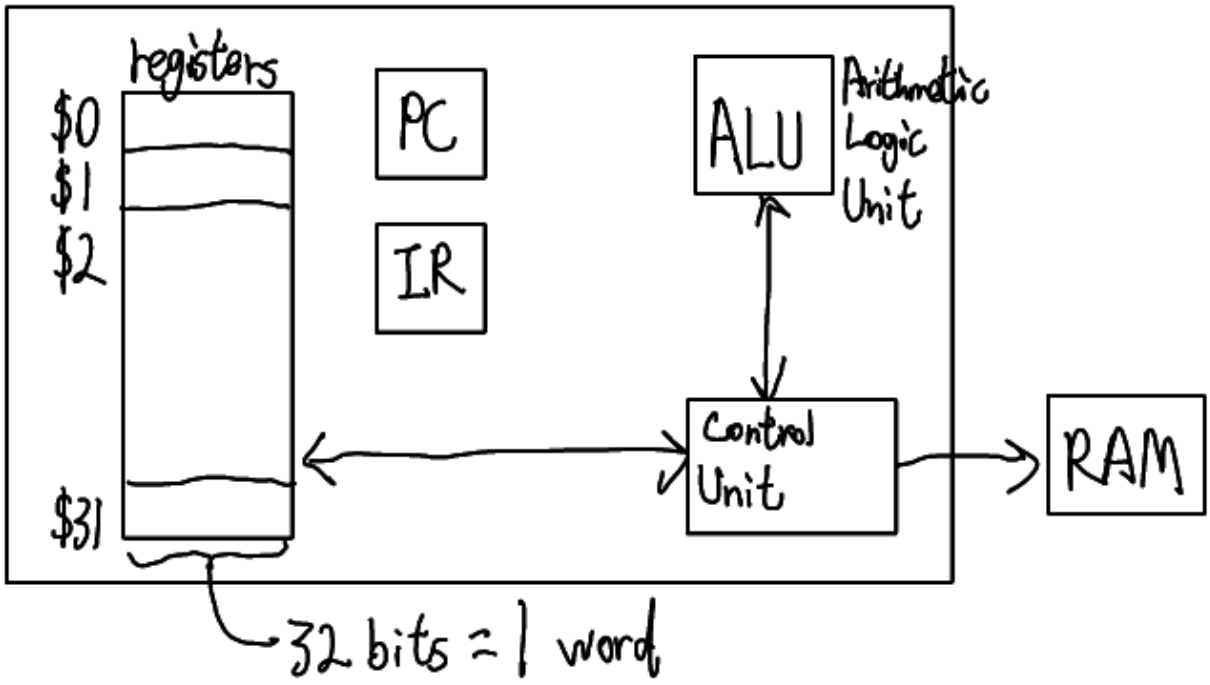
## Demo

- `beep`
- `hasY`
- `babe.jpg`

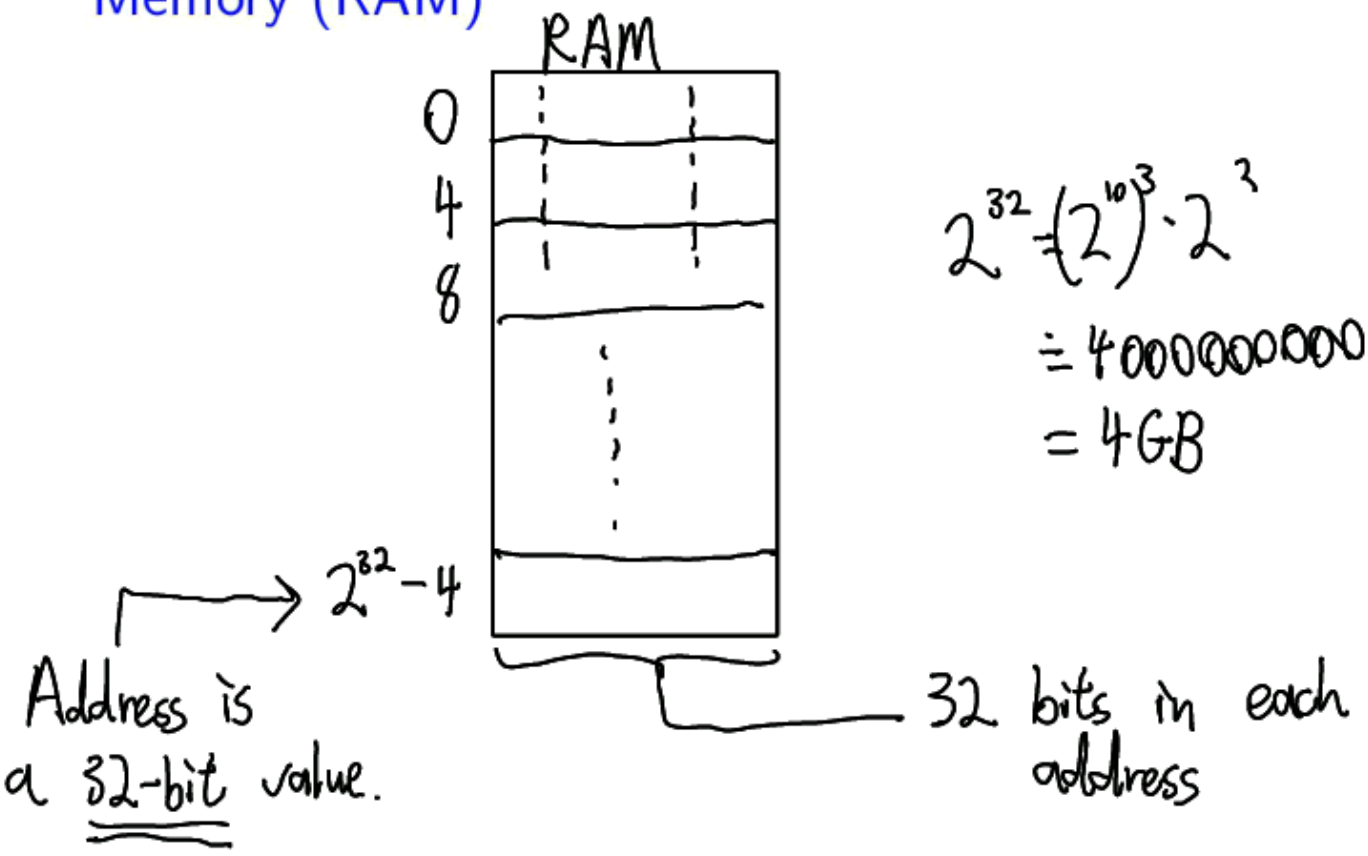
## Our Machine: A Stored Program Computer



## CPU


registers
$0
$1
$2
$31
PC
IR
ALU
Arithmetic
Logic
Unit
Control
Unit
RAM
32 bits = 1 word

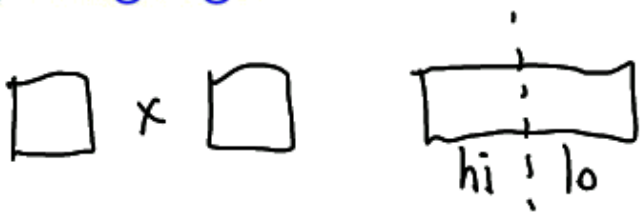
## Memory (RAM)


RAM
0
4
8
$2^{32}-4$
$2^{32} = (2^{10})^3 \cdot 2^2$
$\approx 4000000000$
$= 4GB$
Address is a 32-bit value.
32 bits in each address

# 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

  add the contents of $2 and $3 and place into $1
- computer meaning

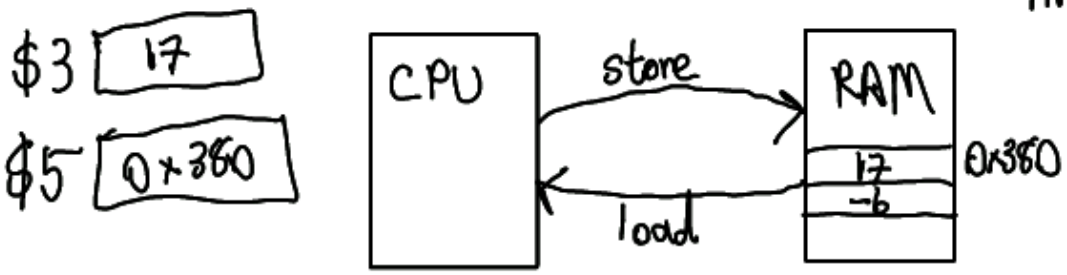
  0000 0000 0100 0011 0000 1000 0010 0000

  $2 $3 $1

  0x00430820

## Communication between CPU and RAM

IN $3, 0($5)




- load → "put one 32 bit from a ram location chunk into a register"
- store
  → "put a register into a ram location"

# Examples

See website.

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

## Machine Cycle

Program Counter:
Address of the next instruction to execute.

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