# January 16

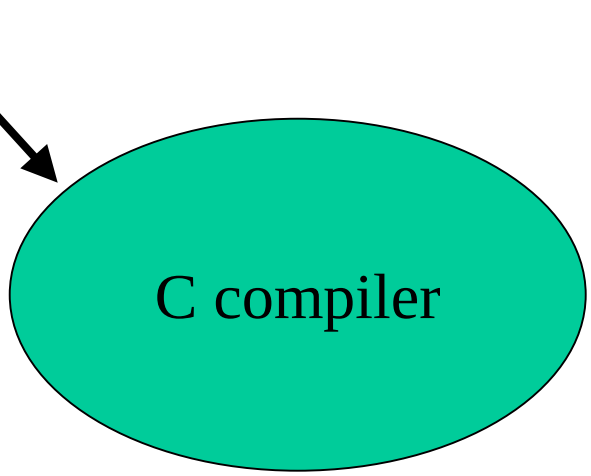
- The books are here.
- Assignment 1 now due Thursday 18 Jan.
- Questions on what we’re doing?
- Questions on anything at end of class.

# Abstractions

- What the user wanted.
- What the programmer designed.
- What the programmer thought about.
- What the language allowed.
- Assembly language.
- Binary.
- Function blocks.
- Gates
- Devices
- Physics

# Abstraction: C to ASM

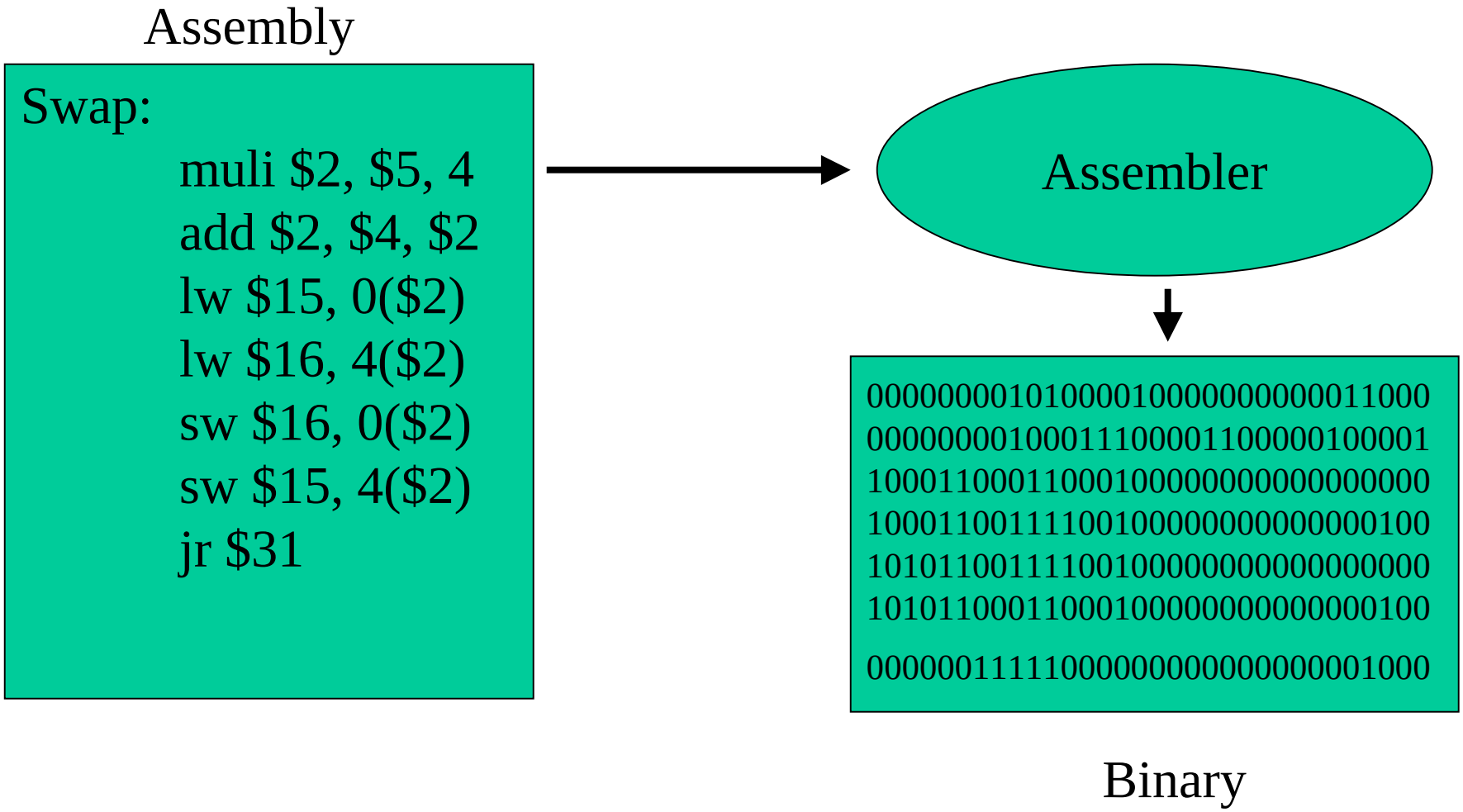
```
Swap(int v[], int k) {
  int temp;
  temp = v[k]; v[k] = v[k+1]; v[k+1] = temp;
}
```




Assembly

```
Swap:
        muli $2, $5, 4
        add $2, $4, $2
        lw $15, 0($2)
        lw $16, 4($2)
        sw $16, 0($2)
        sw $15, 4($2)
        jr $31
```

# Abstraction: ASM to Binary



# Instruction Set Architecture

**... the attributes of a [computing] system as seen by the programmer, *i.e.* the conceptual structure and functional behavior, as distinct from the organization of the data flows and controls, the logic design, and the physical implementation.**

**– Amdahl, Blaaw, and Brooks, 1964**

- An ABSTRACTION!
- interface between hardware and low-level software
- standardizes instructions, machine language bit patterns, etc.
- advantage: *different implementations of the same architecture*
- disadvantage: *sometimes prevents using new innovations*

Modern instruction set architectures:

– 80x86/Pentium/K6, PowerPC, DEC Alpha, MIPS, SPARC, HP

# CISC vs. RISC

- ISA’s originally for humans to use
- Small memory size was critical thus complex instructions
- High-level-language architectures (B5000)
- RISC says do a few things well; only supply what the compiler will use; rely on compiler to get it right.

# Why look at MIPS?

- Why not one that *matters* like Intel?

**Complexity…**

**Ugliness…**

**Horror…**

**Reality…**

# The Really Big Ideas

- Just **bits** for data and program
- Program is a sequence of instruction words
- Data-type determined by instruction
- Large linear “array” of memory
- Small number of “variables” (registers)

# Just Bits

- Program and data have the same representation
- Programs can manipulate programs
- Programs can manipulate themselves!
- Bits not the only way (Lisp)

# Data Types

- char byte short int pointer quad float double
- Instruction determines type of operands
  - Add (int), Add.s (float), Add.d (double)
- Free to reinterpret at will
- How big is a char?
- What’s a pointer?

# Memory

- Large (usually) linear array
- Only **read** with load instructions
  - lw $t5, 100($a3)  ($t5 = mem[100+$a3])
- Only **modified** with store instructions
  - sw $s0, 24($t3)  (mem[24+$t3] = $s0)
- CISC machines have lots of ways to read and write memory

# Memory

- Address is always in bytes
- Words on 4 byte boundary (how many 0’s?)
- Short only on 2 byte boundary
- Doubles only on 8 byte boundary
- CISC allowed them anywhere

Why?

**It’s an ABSTRACTION!**

# GP Registers

- Variables for our programs
- The ONLY operands for most instructions
- A very small number (32 in MIPS)

  Why?
- All 32 bits
- What about new 64 bit ISA's?