August 26

- TA: Angela Van Osdol in 036
- Questions?

What is a computer?

- Tape drives?
- Big box with lots of lights?
- Display with huge letters?
- Little box with no lights?
- Lump in the cable?

5 Classic Computer Components

- Display

- LCD

- Mouse
Keyboard

Inside the case

b. Processor
c. PCI slots
e. Memory slots

Motherboard

Memory

• RAM ➔ Random Access Memory
• DRAM ➔ Dynamic Random Access Memory
• SRAM ➔ Static RAM
• ROM ➔ Read-only Memory
• Volatile / Non-Volatile ➔ needs power or not
• Magnetic ➔ stores bits as magnetized regions

Processor

Pentium III Xeon

You only need switches and wires!

• Relays
• Vacuum tubes
• Transistors
• Integrated Circuits
• VLSI
• Nanotubes?
• Quantum Effect Devices?
Abstraction: C to ASM

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

```
Abstraction: C to ASM

<table>
<thead>
<tr>
<th>C compiler</th>
<th>Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swap:</td>
<td></td>
</tr>
<tr>
<td>multi $2$, $5$, 4</td>
<td></td>
</tr>
<tr>
<td>add $2$, $4$, $2$</td>
<td></td>
</tr>
<tr>
<td>lw $15$, 0($2$)</td>
<td></td>
</tr>
<tr>
<td>sw $16$, 4($2$)</td>
<td></td>
</tr>
<tr>
<td>sw $15$, 4($2$)</td>
<td></td>
</tr>
<tr>
<td>jr $31$</td>
<td></td>
</tr>
</tbody>
</table>
```

Abstraction: ASM to Binary

```
Abstraction: ASM to Binary

<table>
<thead>
<tr>
<th>Assembly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Swap:</td>
</tr>
<tr>
<td>multi $2$, $5$, 4</td>
</tr>
<tr>
<td>add $2$, $4$, $2$</td>
</tr>
<tr>
<td>lw $15$, 0($2$)</td>
</tr>
<tr>
<td>sw $16$, 4($2$)</td>
</tr>
<tr>
<td>sw $15$, 4($2$)</td>
</tr>
<tr>
<td>jr $31$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```
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, Blaauw, and Brooks, 1964

- 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/Itl, 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?

Where we are headed

- Overview of C
- Performance issues (Chapter 2)  \ vocabulary and motivation
- A specific instruction set architecture (Chapter 3)
  Why MIPS? Why not Intel?
- Arithmetic and how to build an ALU (Chapter 4)
- Pipelining to improve performance (Chapter 6)
  briefly
- Memory: caches and virtual memory (Chapter 7)
- Key to a good grade: reading the book!