Below the Program

- High-level language program (in C)
  ```c
  swap int v[], int k){
    int temp;
    temp = v[k];
    v[k] = v[k+1];
    v[k+1] = temp;
  }
  ```
- Assembly language program (for MIPS)
  ```assembly
  swap: sll $2, $5, 2
  add $2, $4, $2
  lw $15, 0($2)
  lw $16, 4($2)
  sw $16, 0($2)
  sw $15, 4($2)
  jr $31
  ```
- Machine (object) code (for MIPS)
  ```
  000000 00000 00101 000100001000000
  00000 00100 00010 000100001000000 . . .
  ```

Logic Design

- Next 2 weeks: we’ll study how a modern processor is built starting with basic logic elements as building blocks.
- Why study logic design?
  - Understand what processors can do fast and what they can’t do fast (avoid slow things if you want your code to run fast!)
  - Background for more detailed hardware courses (CS 150, CS 152)

Logic Gates

- Basic building blocks are logic **gates**
  - In the beginning, did ad hoc designs, and then saw patterns repeated, gave names
  - Can build gates with transistors and resistors
- Then found theoretical basis for design
  - Can represent and reason about gates with truth tables and Boolean algebra
  - Assume know truth tables and Boolean algebra from a math or circuits course.
  - Section B.2 in the textbook has a review

Physical Hardware

Let’s look closer…
Gate-level view vs. Block diagram

Signals and Waveforms: Clocks

Signals and Waveforms: Adders

Signals and Waveforms: Grouping

Signals and Waveforms: Circuit Delay

Combinational Logic

- Complex logic blocks are built from basic AND, OR, NOT building blocks we’ll see shortly.

- A *combinational* logic block is one in which the output is a function only of its current input.

- Combinational logic cannot have memory (e.g., a register is not a combinational unit).
Circuits with STATE (e.g., register)

CLK \rightarrow \text{REGISTER} \\
\text{m_input} \rightarrow f_{\text{output}}

Administrivia

- Midterm coming up on Monday @ 7pm in 1 Pimentel. Heard this enough yet?

Peer Instruction

A. SW can \textit{peek} at HW (past ISA abstraction boundary) for optimizations
B. SW can \textit{depend} on particular HW implementation of ISA
C. Timing diagrams serve as a \textit{critical} debugging tool in the EE toolkit

And in conclusion...

- ISA is very important abstraction layer
  - Contract between HW and SW
- Basic building blocks are logic \textit{gates}
- Clocks control pulse of our circuits
- Voltages are analog, quantized to 0/1
- Circuit delays are fact of life
- Two types
  - Stateless Combinational Logic (&,|,~)
  - State circuits (e.g., registers)