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)

```
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 0001000010000000
000000 00100 00010 0001000000100000 . .
```
What are “Machine Structures”?

Coordination of many levels of abstraction

ISA is an important abstraction level: contract between HW & SW

Synchronous Digital Systems

The hardware of a processor, such as the MIPS, is an example of a Synchronous Digital System

Synchronous:

• Means all operations are coordinated by a central clock. (It keeps the beat of the system).

Digital:

• Mean all values are represented by discrete values
• Electrical signals are treated as 1’s and 0’s and grouped together to form words.
Logic Design

° Next 4 weeks: we’ll study how a modern processor is built; starting with basic elements as building blocks.

° Why study hardware design?
  • Understand capabilities and limitations of hardware in general and processors in particular.
  • 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)
  • There is just so much you can do with processors. At some point you may need to design your own custom hardware.

PowerPC Die Photograph

Let’s look closer…
Transistor Circuit Rep. vs. Block diagram

- Chips is composed of nothing but transistors and wires.
- Small groups of transistors form useful building blocks.
- Blocks are organized in a hierarchy to build higher-level blocks: ex: adders.

The Clock Signal
Signals and Waveforms

Signals and Waveforms: Grouping
Type of Circuits

- **Synchronous Digital Systems** are made up of two basic types of circuits:
  - **Combinational Logic (CL) circuits**
    - Our previous adder circuit is an example.
    - Output is a function of the inputs only.
    - Similar to a pure function in mathematics, \( y = f(x) \). (No way to store information from one invocation to the next. No side effects)
  - **State Elements**: circuits that store information.
Circuits with STATE (e.g., register)

And in conclusion...

° ISA is very important abstraction layer
  • Contract between HW and SW

° Clocks control pulse of our circuits

° Voltages are analog, quantized to 0/1

° Circuit delays are fact of life

° Two types of circuits:
  • Stateless Combinational Logic (&, I, ~)
  • State circuits (e.g., registers)