The Main Parts of a Computer

- CPU
- Bus
- Memory
- I/O Devices
I/O and Storage Devices

• (lots of devices, we'll discuss later)
• I/O devices talk to the CPU via an interface
  – transmits device status, operation to perform, data to be transferred
Buses

- A bus = wires connecting the parts together
- There are lots of buses in a computer system
Properties of Buses

• How many wires are dedicated to transferring data?
  - called the "width" of the data bus
• How fast the bus can transfer data?
• What control and status signals does the bus have?
• How many address lines does the bus have?
An Example: the PCI Bus

- Main high-speed bus standardized in PCs
- Speed up to 66 MHz
- Data width up to 64 bits
- Up to 200 wires total (data + address + control)
Main Memory

- Purpose is to store programs that are executing, and the data they need
- Speed: 1-5ns (fast memory), 25-100ns (slow memory)
- Compared with speed of a disk: a million times faster!
- But, about a thousand times more expensive
  - plus, data is lost when power is removed
Memory as Post Office
Mailboxes

- There are a fixed number of boxes
- Every box has a number, or address
- Every box has the same, fixed size
- Boxes have varying contents
The Unit of Addressability

- In x86 (and most other) architectures, each memory location stores one byte (8 bits)
Memory Addresses and Size

- A memory address is specified with \( n \) bits
  - this means can be at most \( 2^n \) memory locations
  - nowadays, 32-bit addresses are common

<table>
<thead>
<tr>
<th>( N )</th>
<th>( 2^N )</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>64K</td>
</tr>
<tr>
<td>24</td>
<td>16M</td>
</tr>
<tr>
<td>32</td>
<td>4G</td>
</tr>
<tr>
<td>48</td>
<td>1.8*10^{14}</td>
</tr>
<tr>
<td>64</td>
<td>2.8*10^{19}</td>
</tr>
</tbody>
</table>
The Memory Interface

```
+----------------+       +----------------+
|                |       |                |
|      CPU       |       |      Memory    |
|                |       |                |
| Read / Write   |       |                |
|                |       |                |
|    Data        |       |                |
|                |       |                |
|    Address     |       |                |
```
The Memory Read Operation

- The CPU outputs a memory address, then issues the Read command.
- The memory looks up the data at the requested location, while the CPU waits.
- The memory sends the data to the CPU.
The Memory Write Operation

- The CPU outputs a memory address, outputs the data, then issues the Write command.
- The memory receives the data, stores it in the specified location.
**Read Only Memory (ROM)**

- Can read the contents, but not change them (easily)
- Useful for storing boot-strap routines, constant-valued parameters
Words

- Most data types are bigger than one byte
- The "word size" of a computer = the size of general-purpose registers
  - a word consists of consecutive bytes in memory
  - "word address" = address of first byte of the word
- Operand sizes in the Pentium

<table>
<thead>
<tr>
<th># of bits</th>
<th>Called...</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>Byte</td>
</tr>
<tr>
<td>16</td>
<td>Word</td>
</tr>
<tr>
<td>32</td>
<td>Doubleword</td>
</tr>
<tr>
<td>64</td>
<td>quadword</td>
</tr>
</tbody>
</table>
Interpreting Memory Contents

• You can't look at the value and tell what it means; everything is a bit string!
  - it's up to the program to interpret the contents of memory

• Example:
  - 01101000 01100101 01111001 00100001
The Central Processing Unit, or Processor

- The "brain" of a computer
  - executes program instructions, controls all the parts of the computer

- Consists of registers, the ALU (arithmetic logic unit), and the control unit
The CPU (cont.)

- Registers are temporary storage locations
  - advantages: speed and efficiency
- The Arithmetic-Logic Unit (ALU) is where data is manipulated (add, AND, compare, multiply, etc.)
- The control unit determines the sequence and timing of everything the CPU does
How Does an Instruction Get Executed?

• A special register contains the address of the instruction

✔ The CPU "fetches" the instruction from memory at that address

✔ The CPU "decodes" the instruction to figure out what to do

✔ The CPU "fetches" any data (operands) needed by the instruction, from memory or registers

✔ The CPU "executes" the operation specified by the instruction on this data

✔ The CPU "stores" any results into a register or memory
The Instruction Cycle

Fetch Instruction

Decode Instruction

Fetch Operands

Execute Instruction

Store Results
How Does a Complete Program Get Executed?

• The "special register" is initialized to point to the first instruction of the program.

• As part of the instruction fetch cycle, the "special register" is updated to point to the next instruction.

• When one instruction is finished, the cycle is begun all over again.
  - the CPU never quits; it is always executing something.
Fetch Instruction and Update Special Register

Decode Instruction

Fetch Operands

Execute Instruction

Store Results
Clocks and Timing

• The actions of a CPU are synchronized by an external clock
• The speed with which the clock ticks determines how fast the CPU runs
• \( \text{MHz} = \text{one million clock ticks / second} \)
• \( \text{GHz} = \text{one billion clock ticks / second} \)