how computers work (2)
what goes on inside

inside the box: the motherboard
contains:
- memory to store data and instructions
- processor to execute instructions, and
- I/O ports to communicate with peripherals
- control circuitry to coordinate everything

the motherboard abstracted
memory or RAM (random access memory): where program instructions and associated data are stored when running on the computer
I/O ports: communication with peripherals
processor or CPU: where program instructions are executed; contains ALU (arithmetic and logic unit) and control unit

fetch/execute cycle
the control unit of the processor coordinates the fetch/execute cycle:

- an instruction and its data are fetched from memory and put in the processor
- the ALU of the processor executes the instruction
- the result is returned to memory or output ports
memory organization (32 bits)

- memory is a giant array of **words**
  - each word stores 4 bytes
  - each byte stores 8 bits
  - each bit is 0 or 1
  - the location of each byte has an address
- memory stores both data and instructions

instructions in memory

- the processor can execute simple instructions, which comprise the **machine language** of the processor
- a machine language program is stored in **instruction memory** in RAM
- each instruction is a binary string, say, 32 bits long

processor organization

- data is stored in registers
- arithmetic and logic unit
- program counter
- control unit (coordinator)

machine language instructions

machine instructions are a lot more primitive than, say, Java instructions

examples:

- **add**, **subtract**, **multiply** data stored in the processor
- **load** and **store** instructions to get data from memory to registers and back
- **branch** and **jump** instructions for control flow
- instructions for communication with I/O devices

example: add instruction

**add 3, 4, 3:**
add the contents of registers 3 and 4, and store the answer in register 3

<table>
<thead>
<tr>
<th>6 bits</th>
<th>5 bits</th>
<th>5 bits</th>
<th>5 bits</th>
<th>5 bits</th>
<th>6 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
<td>add</td>
</tr>
</tbody>
</table>

| 000000  | 00011   | 00100   | 00011   | 000000  | 100000 |

example: load instruction

**load 3, 7000:**
load into register 3 the contents of memory location 7000
example: branch instruction

Branch 2, 5:
if content of register 2 is 0, then move ahead by 5 instructions, otherwise continue to the next instruction

example: jmp instruction

Jmp 7:
move ahead by 7 instructions

processor organization

contains address of next instruction

program counter

control unit (coordinator)

register

register

register

the program counter

• when the execution of an instruction is complete, the control unit retrieves the next instruction from memory

• the program counter contains the memory address of the next instruction

• the program counter must be updated after each instruction is executed

fetch/execute cycle (in detail)

1. fetch instruction specified by program counter
2. decode instruction
3. fetch data from memory and store in registers
4. perform operation and send result to data memory, a register, or program counter
5. update data memory
6. update program counter
putting it all together

trace out the fetch/execute cycle on the following instructions:

- **load 3, 7000** (load into register 3 the contents of memory location 7000)
- **load 4, 7001** (load into register 4 the contents of memory location 7001)
- **add 3, 4, 3** (add the contents of registers 3 and 4 and store in register 3)
- **store 3, 7002** (store the contents of register 3 in memory location 7002)
- **jmp 5** (move ahead 5 instructions)
after 5th instruction

clocks
- a clock controls the timing of each step in the instruction cycle
- when it is time for a signal to be sent, the wire containing that signal is enabled
- at other times, the signal is disabled
- modern processors may have clock speeds of more than 1 gigahertz, which means 1 cycle per nanosecond, or $10^9$ instructions per second

“remember your nanoseconds!”

“The ‘nanoseconds’ that Grace Hopper handed out were lengths of wire, cut to not quite 12 inches in length, equal to the distance traveled by an electron along the wire in the space of a nanosecond - one billionth of a second. In teaching efficient programming methods, Admiral Hopper wanted to make sure her students would not waste nanoseconds.”

- From “Tribute to Grace Murray Hopper” by Merry Malsal
http://www.sdsc.edu/Hopper/GHC_INFO/hopper.html

making computations faster
- much effort has been expended in increasing processing speed:
  - multiple processors
  - caches (stores of data on the processor)
  - pipelining (starting one instruction before the last one finishes)
- processing speed has increased rapidly over the past 30 years

resources
- see chapter 9 of the text, up to page 261