

How Computers Work

Jun 4, 2007
KangKang Yin

Step 1: from text to binary

Represent information by symbols

- ▶ Telegraph: dots and dashes in morse code
- ▶ Braille: raised dots
- ▶ Genes: nucleotides (A,C,G,T) in our DNA
- ▶ Computer: 0 and 1

digitizing text, keyboard inputs

- ▶ letters, numbers, punctuation marks, spaces, and other symbols ☺
- ▶ "nonprintable" characters: new-line, tab
- ▶ more: backspace, function keys

ASCII standard

- ▶ American Standard Code for Information Interchange
- ▶ ASCII table: <http://www.ascii-code.com/>
- ▶ 8 bits == 1 byte → 256 symbols

Representing text in binary

symbol	bit representation
a	01100001
b	01100010
c	01100011
d	01100100

- ▶ "bad" is represented as "01100010 011000101100100"
- ▶ what does "011000110110000101100010" represent?

Representing decimal digits in ASCII

- ▶ Why?
 - decimal numbers embedded in text
 - vs.
 - numbers in computer arithmetic
- ▶ the CS department's phone number
 - "822-3061"
 - "00111000 00110010 00110010 00101101 00110011 00110000 00110110 00110001"

numbers in JavaScript

- ▶ Loosely typed
- ▶ suppose 'x' is declared as a variable:
 - `var x;`
- ▶ what is `x + 2 + 3`?

numbers in JavaScript

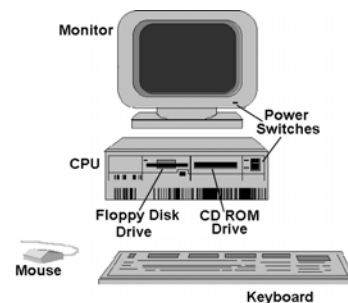
- ▶ `var x="1";`
 - `x + 2 + 3` is the string "123"
- ▶ `var x=1;`
 - `x + 2 + 3` is 6
- ▶ `var x="1";`
 - `parseInt(x) + 2 + 3` is 6

numbers in Java

- ▶ Java syntax is designed to avoid any confusion about the representation of numbers
- ▶ keywords are used in variable declarations, to declare right from the start which type the variable has
 - *`int width;`*
 - *`String userName;`*

Step 2: from binary to electrons

computer from the outside



Inside the box

- ▶ memory: to store data and instructions
- ▶ processor: to execute instructions
- ▶ expansion slots: to connect with peripheral cards
- ▶ I/O ports: to communicate with peripherals
- ▶ control circuitry and data channels: to coordinate everything

Let's look at pictures

<http://www.karbosguide.com/books/pcarchitecture/chapter04.htm>

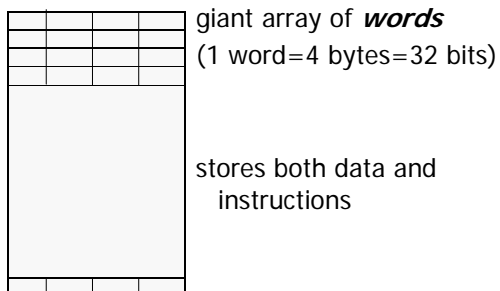
The parts that compute

- ▶ memory or RAM (random access memory) stores
 - program instructions
 - associated data
- ▶ processor or CPU executes program instructions
 - ALU (arithmetic and logic unit)
 - control unit

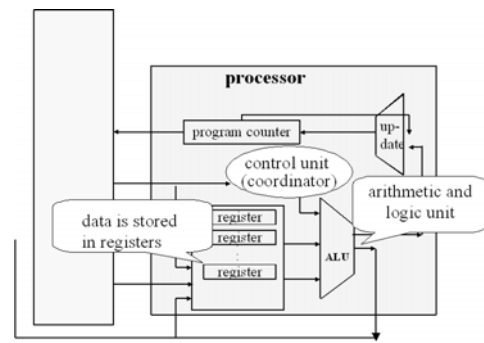
Fetch/Execute Cycle

- ▶ Instruction Fetch: from memory
- ▶ Instruction Decode: by CPU's control unit
- ▶ Data Fetch: from memory
- ▶ Instruction Execution: by ALU
- ▶ Result Return: to memory

memory organization



processor organization



Computer Languages

▶ (high-level) Programming language:

- Java, C++, JavaScript

Compile



▶ (low-level) Assembly language:

- Intel 80x86, Sun SPARC

Assemble



▶ (binary) Machine language:

....111000100100111....

Basic Assembly Instructions

▶ **add, subtract, multiply**: data stored in the processor

▶ **load, store**: get data from memory to registers and back

▶ **branch, jump**: instructions for control flow

example instructions

▶ **add 3, 4, 3**

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

▶ **load 3, 7000**

- load into register 3 the contents of memory location 7000

▶ **jmp 7:**

- move ahead by 7 instructions

To be continued