Lecture 1: Preliminaries

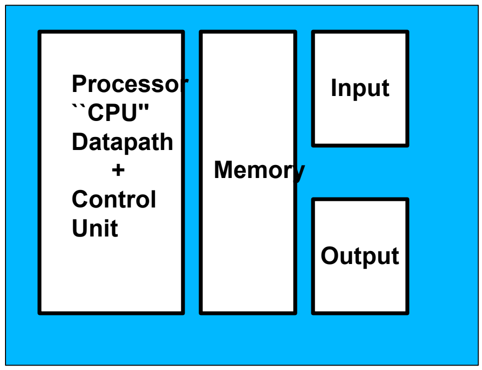
Dragan Mirkovic
Department of Computer Science
University of Houston

Introduction

- Announcements:
  - Course and lab web pages are available
  - Lab will probably start next week.
  - Questions?
- Today's lecture:
  - Computer Abstractions and Technology (Ch. 1 in Jones)
  - Hardware overview
  - Numbering systems, addressing, etc.

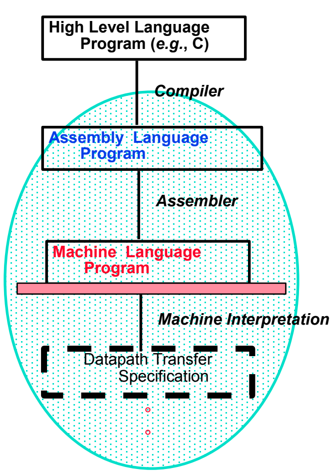
Main parts of the computer

- Since 1946 all computers have had 5 components



Interconnection Structures (buses)

What is below your program?



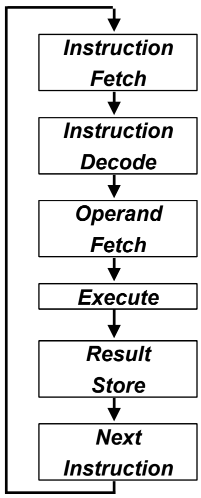

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

High level language
(C code)

```
Hello   PROC
        mov     ax, @data
        mov     ds, ax
        mov     dx, OFFSET Message
        mov     ah, 9h
        int     21h
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Execution Cycle



Obtain instruction from program storage

Determine required actions and instruction size

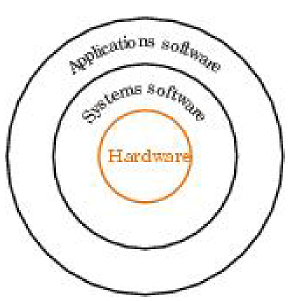
Locate and obtain operand data

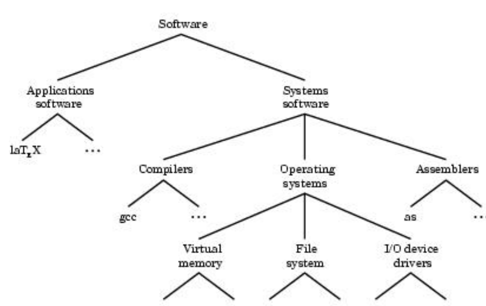
Compute result value or status

Deposit results in storage for later use

Determine successor instruction

Hardware and Software





Hardware Overview

- From the outside:
  - Monitor, keyboard, mouse, “box”, …

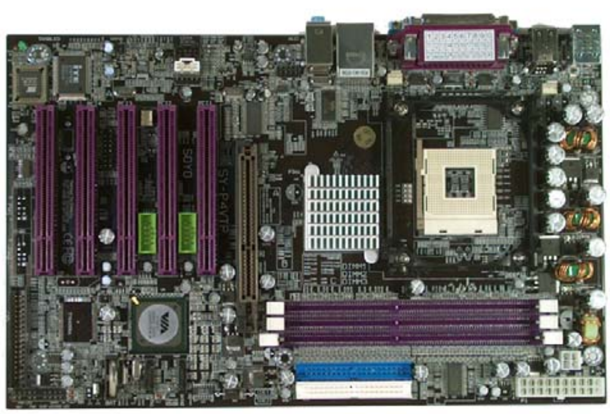
Hardware Overview (Cont.)

- What is inside the box?
  - Motherboard:
    - CPU
    - Memory
    - I/O Devices
  - External Drives:
    - Hard disk
    - Floppy disk
    - CD, DVD, …
  - Power, cables, …

Motherboard

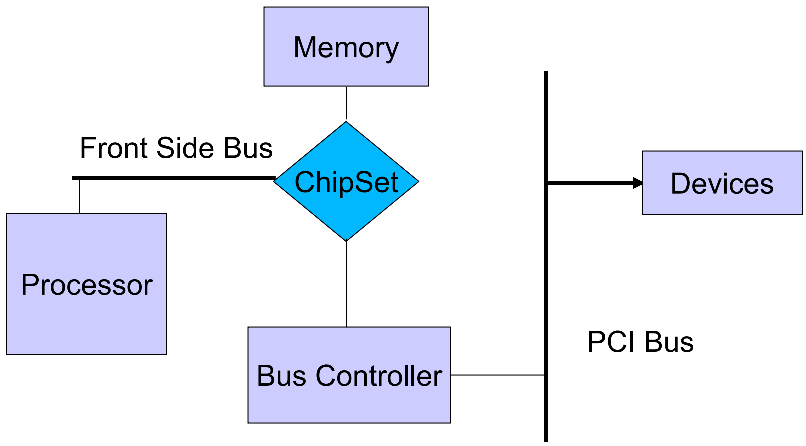
- Contains the most important parts of your computer
  - CPU
  - Memory
  - Chipset
  - Video (?), Audio (?), etc

Example: P4 MB



D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Computer system components



Computer system components

- The CPU
  - In most of the cases a single IC
  - Speed in MHz or GHz (10^6 or 10^9 Hz) machine cycles per second
- Memory
  - Fast (Cache) on/off dye: Expensive, small
  - Slow (main): Cheap, big
  - RAM v.s. ROM (software v.s. firmwere)
  - Collection of bits (binary digits: 0 or 1)
  - Arranged in groups of 8 as **bytes**
  - Size in MB, GB, TB, …

Example: Intel Architecture

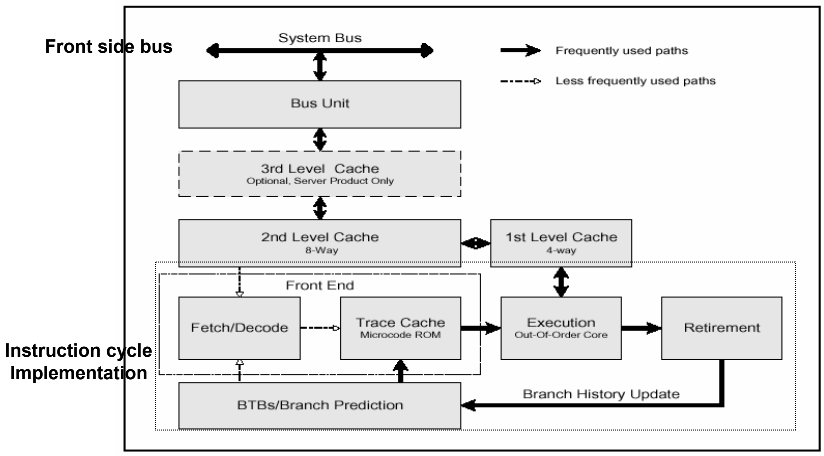
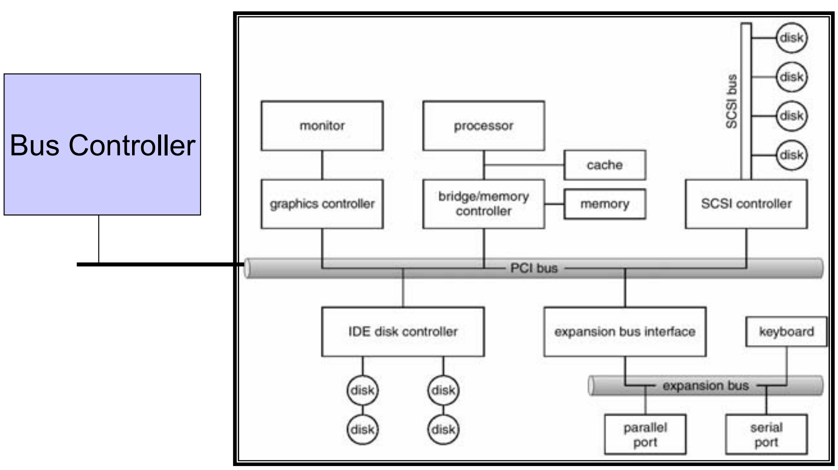


Figure 2-2. The Intel NetBurst Micro-Architecture

PCI Bus and Devices



CPU Structure

- Very complex:
  - Example: Intel P4 (IA-32 bit arch):
    - 8 32-bit registers (general purpose)
    - 6 16-bit (Segment registers)
    - 8 80-bit FPU registers
    - 8 64-bit MMX registers
    - 8 128-bit XMM registers (SSE, SSE2)
    - 3 16-bit registers (control, status, tag)
    - ALU (FPU, Integer, Logic, …)
    - Etc. (see vendor documentation for details

Simplified view

- ALU: performs integer computations
- FPU: performs floating-point computations
- Control unit: which operations to perform
- Registers:
  - temporary storage for ALU and FPU
  - General registers
  - Special purpose registers:
    - Instruction pointer (IP)
    - Stack pointer (SP)

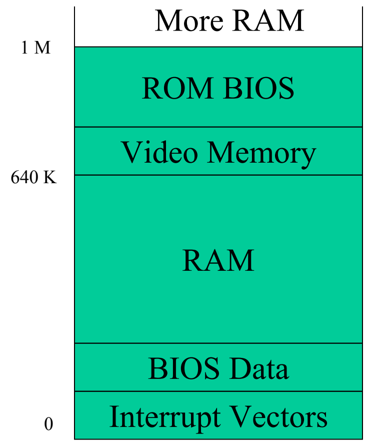
Instruction Set

- Two basic philosophies:
  - CISC (Complex Instruction Set Computing)
    - Each instruction does many things
    - One or less than one per cycle
    - All IBM PC CPUs
  - RISC (Reduced Instruction Set Computing)
    - Simple instructions,
    - One or more than one per cycle
    - SUN, HP, Motorola PPC, SGI

IBM PC Hardware

- Present day personal computers are based on either
  - Intel 80x86 processors and their clones (AMD, Cyrix, …)
  - Motorola PPC G3, G4, G5 (RISC)
- We will be concerned with Intel 80x86 family of processors
- Regardless of the model the 1st MB of RAM has the original organization.

IBM PC Memory organization



- Interrupt vectors
- BIOS Data
- RAM
- Video memory
- ROM BIOS
- More RAM

80x86 Memory Addressing

- On 80386 and up two types of addressing:
  - 16-bit addressing
    - Only 2^{16} (65536) bytes possible (0 – 65535)
    - 1 MB = 2^{20} = 1,048,576
  - 32-bit addressing
    - Up to 2^{32} (4 GB) bytes possible
    - Current limit for 32-bit architectures
    - Memory protection for multiple programs
    - Segmentation and paging

DOS Window

- Virtual 8086 mode in Windows
- 1 MB segment of memory
- DOS runs only in 16 bit mode
- We will use mostly the 16-bit mode for this course

Numbering systems

- Integer representation
  - Recursive definition 1 + successor
  - Which symbol to use for each of them?
  - Positional number systems
    - Position determines the order of the each digit
    - Uniquely defined with base b and digits 0,1, …, b-1 (normalized form)

$$a = a_n a_{n-1} \cdots a_0 = \sum_{i=0}^{n} a_i b^i = a_0 + a_1 b + \cdots + a_n b^n$$

  - Decimal number system (base = 10)
  - Binary number system (base = 2)
  - Hexadecimal number system (base = 16)

Numbering systems - Notation

- Binary numbers

  101011100110110b or 101011100110110B
- Hexadecimal numbers

  129EFF9H or 129EFF9h
- Other notations
  - $(10110111)_2$, $(10110111)_{16}$, $(10110111)_{10}$
  - In general:

$$(a_n a_{n-1} \cdots a_1 a_0)_b$$

Conversions

- Methods:
  - Direct
  - Recursive
- Binary to decimal (direct method) Ex.: 1011010B = 90

$$1011010 = 1\times 2^6 + 0\times 2^5 + 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0$$

$$64 + 16 + 8 + 2 = 90$$

- Decimal to binary, Horner's method (recursive)

$$1\times 2^6 + 0\times 2^5 + 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0 =$$
$$(((((1\times 2+0)\times 2+1)\times 2+1)\times 2+0)\times 2+1)\times 2+0 = 90$$

Conversions…

- Assume that we don't know the binary representation

$$(((((a_6 \times 2 + a_5) \times 2 + a_4) \times 2 + a_3) \times 2 + a_2) \times 2 + a_1) \times 2 + a_0 = 90$$

$$\Rightarrow a_0 = 0$$

- It follows that a_0 is the remainder of 90/2 = 45 and a_0 =0. We can continue in this fashion and calculate $a_1, \ldots, a_6$ by recursively dividing the decimal representation by 2.

$$((((a_6 \times 2 + a_5) \times 2 + a_4) \times 2 + a_3) \times 2 + a_2) \times 2 + a_1 = 45 \Rightarrow a_1 = 1$$

$$(((a_6 \times 2 + a_5) \times 2 + a_4) \times 2 + a_3) \times 2 + a_2 = 22 \Rightarrow a_2 = 0$$

$$((a_6 \times 2 + a_5) \times 2 + a_4) \times 2 + a_3 = 11 \Rightarrow a_3 = 1$$

$$(a_6 \times 2 + a_5) \times 2 + a_4 = 5 \Rightarrow a_4 = 1$$

$$a_6 \times 2 + a_5 = 2 \Rightarrow a_5 = 0, \quad a_6 = 1$$

Conversions…

- Octal and Hexadecimal numbering systems
- Conversions to and from binary is very simple by grouping of digits
- A possible confusion with hex numbers containing only the letters (ex. each)
- The rule is
  - A number is assembly language must start with a digit (0EACH is a number EACH is not)

Conversion Table

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Addition in Bin and Hex

- Same rules as in decimal system
- School boy method
- Binary: Hexadecimal:

$$\begin{array}{r} 1011 \\ +1110 \\ \hline 11001 \end{array} \qquad \begin{array}{r} D \\ +E \\ \hline 19 \end{array}$$

D. Mirkovic, COSC 2410: Computer Organization and Programming, Spring 2004

Negative numbers

- In any basis the minus sign is used to represent negative numbers
- In a computer one bit is reserved for the sign representation
- Conventions

  The left-most bit represents the sign

  0 represents +

  1 represents –

Sign-magnitude representation

- The most obvious way
- Example:

Sign bit

$$12 = 00001100B$$

$$-12 = 10001100B$$

- Problems:
  - Arithmetic is very complicated
  - 0 has two different representations

1's Complement Representation

- Negative numbers are represented as complements of 11111111B (2^8-1) of the corresponding positive number
- Example: $$12 = 00001100B$$
$$-12 = 11111111B - 00001100B$$
$$-12 = 11110011B$$
- Solves the arithmetic problem with **end-around carry**
- 0 is still not unique:
  +0 = 00000000B, -0 = 11111111B

2's Complement Representation

- This is the method actually used by the IBM PC and other modern computers
- 2's complement = 1's complement + 1

$$11111111B - X + 1 = 100000000B - X = 2^8 - X$$

- In general the *n*-bit 2's complement of X is $2^n - X$
- 2's complement is its own inverse: $2^n-(2^n-X)=X$
- Example: -12 = 11110100B

2's Complement …

- Quick 2's complement:
  - Starting from the right keep all zeros and the 1st one unchanged
  - Change all further digits
- Example:

$$
\begin{aligned}
-12 &= 2^8 - 12 = 244 \\
&= F4H \\
&= 100000000B - 00001100B \\
&= 11110100B
\end{aligned}
$$

Addition of 2's Complement Numbers

- Add two numbers as though they were unsigned
- Discard any carries on the left end
- Example: 76-12 in 2's complement form

$$76 - 12 = 01001100B + 11110100B = 01000000B = 64$$

- Example: -76+12 in 2's complement form

$$-76 + 12 = 10110100B + 00001100B = 11000000B = -64$$

Notes on 2's Complement

- Hex representation is more compact
- Hex numbers starting with 8-F are negative
- Range of signed and unsigned numbers

Length	Signed range	Unsigned range
8-bit	-128…127	0 … 255
16-bit	-32768…32767	0 … 65535
n-bit	$-2^{n-1} \ldots 2^{n-1} - 1$	$0 \ldots 2^{n} - 1$

- In complement arithmetic all operands **must** have the same length
- Changing lengths of 2'complement numbers is easy

Overflow

- If a finite number of bits is used for representation an overflow can occur
- Examples: 76+54 and –76-54

$$76 + 54 = 01001100B + 00110110B = 10000010B = -126$$
$$-76 - 54 = 10110100B + 11001010B = 01111110B = 126$$

- These answers are wrong!

  76+54 = 130 out of range for 8-bit

Representation Summary

- With n bits there are 2^n possible numbers that we can encode.
- How to encode positive and negative numbers together?
- Issues: ease of operations, uniqueness, simplicity
- Which one is the best? Why?

Unsigned	Sign-Magnitude	1's Complement	2's Complement
000	+0	+0	+0
001	+1	+1	+1
010	+2	+2	+2
011	+3	+3	+3
100	-0	-3	-4
101	-1	-2	-3
110	-2	-1	-2
111	-3	-0	-1

Homework

- Exercises 1.3, problems 1,2,5, and 6. pp.19 in Jones.
- Read Chapter 1 in Jones.