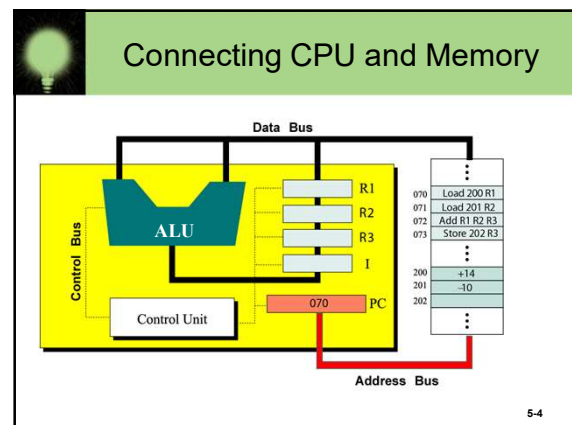


Chapter Goals

- Describe the **fetch-decode-execute cycle** of the von Neumann machine
- Low level programming languages (面向机器的语言)
 - List the **operations** that a computer can perform
 - Distinguish between **immediate mode addressing** and **direct addressing**
 - Distinguish between **machine language** and **assembly language**
 - Describe the steps in creating and running an assembly-language program

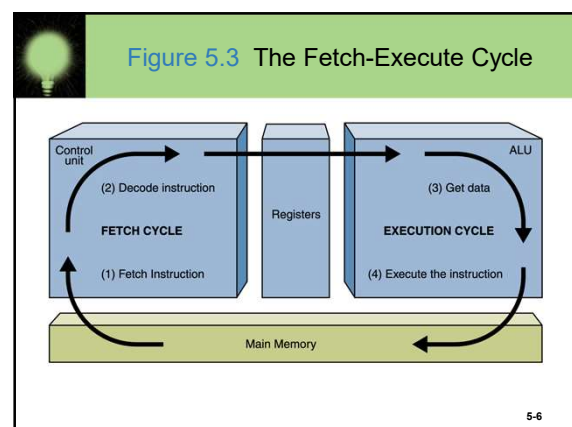
7-3



The Fetch-Execute Cycle

- Fetch the next instruction
- Decode the instruction
- Get data **if needed**
- Execute the instruction

5-5



How does CPU work ?

- Demo

<http://www.science.smith.edu/~jcardell/Courses/CSC103/CPUsim/cpusim.html>

5-7

Computer Operations

- A computer is a programmable electronic device that can **store**, **retrieve**, and **process data**
- **Data** and **instructions** to manipulate the data are **logically the same** and can be stored in the **same place**
- **Store**, **retrieve**, and **process** are actions that the computer can perform on data

7-8

Machine Language

- **Machine language** (机器语言) The instructions **built into the hardware** of a particular computer (计算机硬件可识别的语言)
- Initially, humans had no choice but to write programs in machine language because other programming languages had not yet been invented
 - (用0,1编写程序?)

7-9

Machine Language

- **Every** processor type **has its own** set of specific machine instructions
- The relationship between the processor and the instructions it can carry out is **completely integrated**
- Each machine-language instruction does **only one very low-level task**

7-10

PIPPIN Machine: A Virtual Computer

- **Virtual computer** A hypothetical machine designed to contain the important features of real computers that we want to illustrate
- Features in PIPPIN
 - The memory is made up of 256 bytes. A half store data and other store instruction
 - has 18 machine-language instructions
 - Has IR, PC, ACC(累加器) registers in CPU
 - A 8bit ALU (8位的CPU)
- We are only going to examine a few of these instructions

7-11

Some Sample Instructions

Opcode Operand		Assembly Instruction	Description
Binary	Hex		
00000000 bbbbbb	00 XX	ADD X	Add contents of referenced memory address to contents of accumulator. <i>Address Mode = Direct</i> Example: Add value stored at memory address 128 (1C) to contents of accumulator. 00000000 10000000 00 10
00010000 bbbbbb	10 XX	ADD #n	Add immediate value to contents of accumulator. <i>Address Mode = Immediate</i> Example: Add the number 45 (00101101 binary, 2D hex) to accumulator. 00010000 00101101 10 2D
00000001 bbbbbb	01 XX	SUB X	Subtract contents of referenced memory address from contents of accumulator. <i>Address Mode = Direct</i> Example: Subtract value stored at memory address 129 (X) from accumulator. 00000001 10000001 01 11

<http://cs.smith.edu/~jcardell/courses/CSC103/PIPPINGuide.html>

7-12

Instruction Format

- The instruction specifier is made up of several sections
 - The **operation code**
 - The **register specifier**
 - The **addressing-mode** specifier

7-13

Instruction Format

- There are two parts to an instruction
 - The 8-bit instruction specifier (命令指示)
 - And optionally, the 8-bit operand specifier (操作数)

0 0 0 X Z Z Z Z

b b b b b b b b

instruction specifier: **ZZZZ**: 操作码
 X: 寻址模式
 1 表示操作数是数值
 0 表示操作数是该地址的内容

operand specifier:
 一个数值, 或者
 一个内存地址

7-14

A Program Example

Address	Instruction	Accum Pass 1	Accum Pass 2
00	LOD X	3	2
02	SUB #1	2	1
04	JMZ 10	2	1
06	STO X	2	1
08	JMP 0	2	1
10	HLT	***	***
Initial Memory Values		Memory Pass 1	Memory Pass 2
W (128)	???	???	???
X (129)	3	2	1

7-15

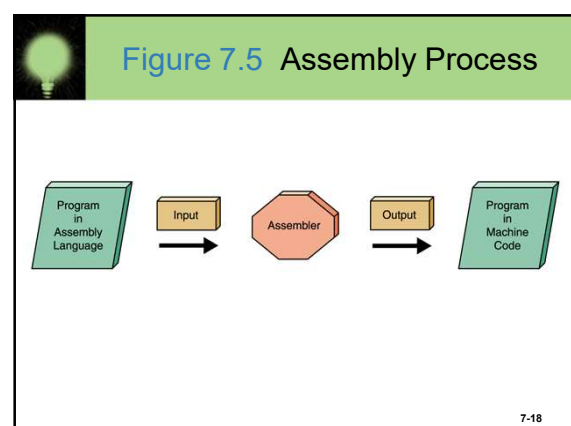
Program Execution Demo

7-16

Assembly Language

- Assembly languages** (汇编语言) A language that uses mnemonic codes (助记符号) to represent machine-language instructions
 - The programmer uses these alphanumeric codes **in place of binary digits**
 - A program called an assembler reads each of the instructions in mnemonic form and translates it into the machine-language equivalent (翻译成对应的机器语言)

7-17



A New Program

```

//sum (1..n)
set sum to 0
for count from 1 to n
    set sum to sum + count
end for
output sum

```

```

//W store n
//set sum to 0
LOD #0
STO Y
//for count from 1 to n
LOD #1 // i=1
STO X
JMP sum-end(16)
sum-next: LOD X // i++
ADD #1
STO X
sum-end: SUB W
JMZ end(28)
//set sum to sum + count
LOD Y
ADD X
STO Y
JMP sum-next(10)
//end for
end: HLT
//output sum

```

让我们把这个算法，翻译成对应的程序？

High level programming language

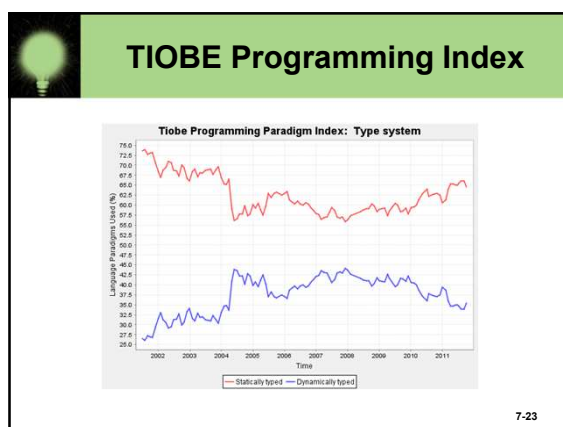
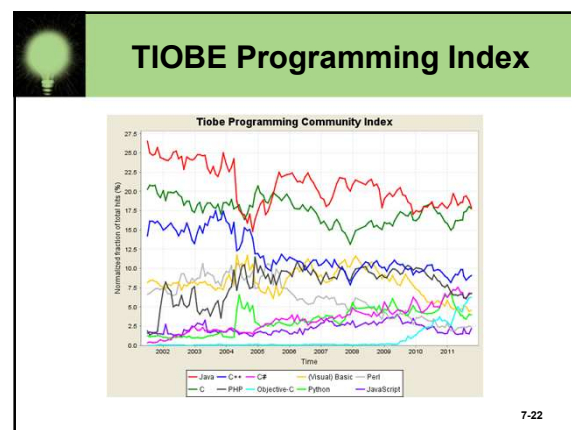
- High level programming language
– ?
- Examples
 - C, C++, Java, and Visual Basic
 - Ada, Lisp, C#
 - PHP, Python
 - more

<http://images.china-pub.com/ebook195001-200000/199000/ch01.pdf>

TIOBE Programming Index

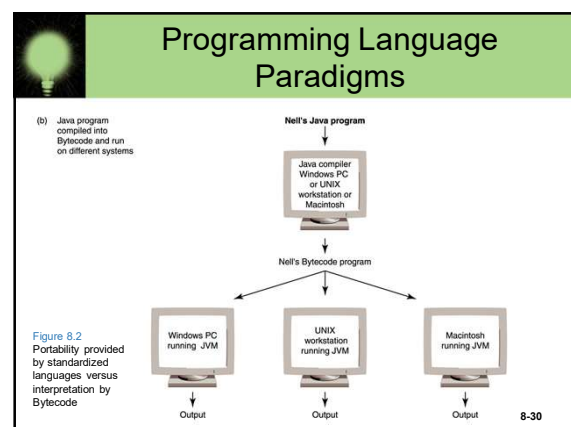
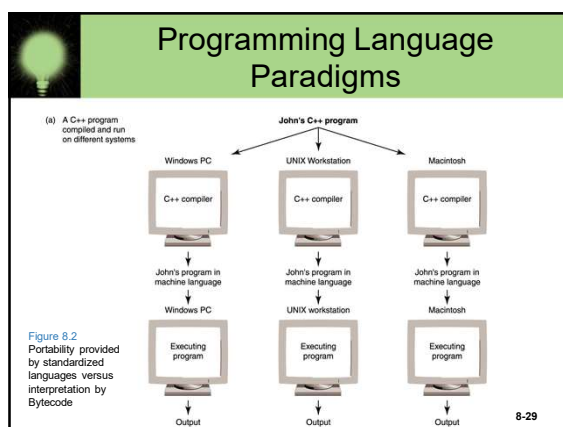
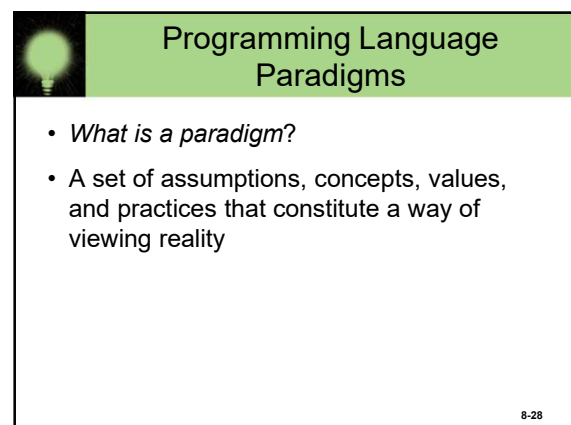
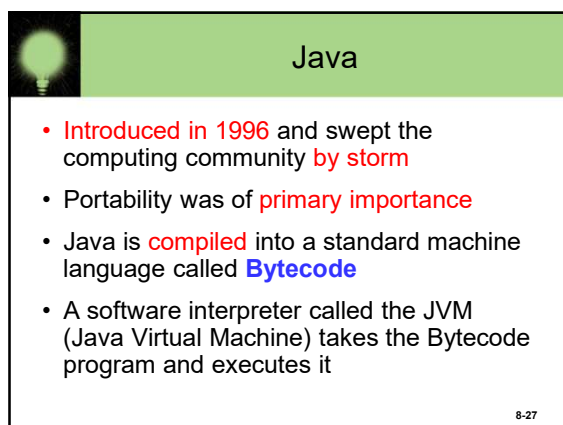
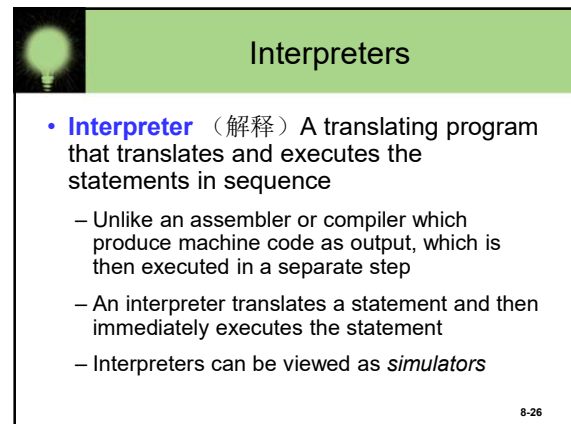
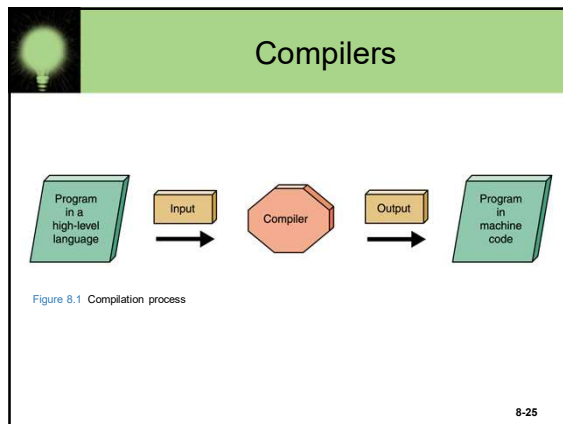
Position Oct 2011	Position Oct 2010	Delta in Position	Programming Language	Ratings Oct 2011	Delta Oct 2010	Status
1	1	—	Java	17.913%	-0.25%	A
2	2	—	C	17.707%	+0.53%	A
3	3	—	C++	9.072%	-0.73%	A
4	4	—	PHP	6.818%	-1.51%	A
5	6	↑	C#	6.723%	+1.76%	A
6	8	↑↑	Objective-C	6.245%	+2.54%	A
7	5	↓	(Visual) Basic	4.549%	-1.10%	A
8	7	↓	Python	3.944%	-0.92%	A
9	9	—	Perl	2.432%	+0.12%	A
10	11	↑	JavaScript	2.191%	+0.53%	A
11	10	↓	Ruby	1.526%	-0.41%	A
12	12	—	Delphi/Object Pascal	1.104%	-0.45%	A
13	13	—	Lisp	1.031%	-0.05%	A


<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html> ,2011.10



Compilers

- **Compiler** (编译) A program that translates a **high-level language program** into **machine code**
- High-level languages provide a richer set of instructions that makes the programmer's life even easier






Programming Language Paradigms

- Imperative or procedural model
 - FORTRAN, COBOL, BASIC, C, Pascal, Ada, and C++
- Functional model
 - LISP, Scheme (a derivative of LISP), and ML


8-31



Programming Language Paradigms

- Logic programming
 - PROLOG
- Object-oriented paradigm
 - SIMULA and Smalltalk
 - C++ is as an imperative language with some object-oriented features
 - Java is an object-oriented language with some imperative features


8-32



Programming in Python

```
>>> # Fibonacci series:
...     # the sum of two elements defines the next
...     a, b = 0, 1
>>> while b < 10:
...     print b
...     a, b = b, a+b
... 
```

7-33



作业 (1/1)

- 1、Program with machine language according to the following c.


```
int 8 a = 1;
int 8 c = a + 3;
```

 - 1) Write your assembly code & machine code
 - 2) Explain machine code execution with the fetch-decode-execute cycle
 - 3) Explain functions about IR, PC, ACC registers in a CPU
 - 4) Explain physical meaning about vars a & c in a machine
- 2、简答题
 - 1) What are stored in memory?
 - 2) Can a data or a instruction stored in the same place?
 - 3) Explain Instruction Format with example instructions.
- 3、解释以下词汇
 - 1) 汇编语言 (Assembly Language)
 - 2) 编译 (Compiler)
 - 3) 命令式语言 (Imperative programming)
 - 4) 函数编程语言 (Functional programming)
 - 5) 过程式编程 (Procedural programming)

7-34