Assembly Language for Intel-Based Computers, 4th Edition

Kip R. Irvine

Chapter 1: Basic Concepts

Slides prepared by Kip R. Irvine

Revision date: 09/15/2002

Modified by Nikolay Metodiev Sirakov – 01.24.2012

- Chapter corrections (Web) Assembly language sources (Web)
- Printing a slide show

Assembly Language Applications

- Some representative types of applications:
  - Business application for single platform
  - Hardware device driver
  - Business application for multiple platforms
  - Embedded systems & computer games

(see next panel)

Comparing ASM to High-Level Languages

Type of Application	High-Level Languages	Assembly Language
Business application software, written for single platform, medium to large size.	Formal structures make it easy to organize and maintain large sections of code.	Minimal formal structure, so one must be imposed by programmers who have varying levels of experience. This leads to difficulties maintaining existing code.
Hardware device driver.	Language may not provide for direct hardware access. Even if it does, awkward coding techniques must often be used, resulting in maintenance difficulties.	Hardware access is straightforward and simple. Easy to maintain when programs are short and well documented.
Business application written for multiple platforms (different operating systems).	Usually very portable. The source code can be recompiled on each target operating system with minimal changes.	Must be recoded separately for each platform, often using an assembler with a different syntax. Difficult to maintain.
Embedded systems and computer games requiring direct hardware access.	Produces too much executable code, and may not run efficiently.	Ideal, because the executable code is small and runs quickly.

Irvine, Kip R. Assembly Language for Intel-Based Computers, 2003. Web site Examples

Virtual Machine Concept

- Virtual Machines
- Specific Machine Levels

Virtual Machines

- Tanenbaum: Virtual machine concept
- Programming Language analogy:
  - Each computer has a native machine language (language L0) that runs directly on its hardware
  - A more human-friendly language is usually constructed above machine language, called Language L1
- Programs written in L1 can run two different ways:
  - Interpretation – L0 program interprets and executes L1 instructions one by one
  - Translation – L1 program is completely translated into an L0 program, which then runs on the computer hardware

Translating Languages

English: Display the sum of A times B plus C.

C++: cout << (A * B + C);

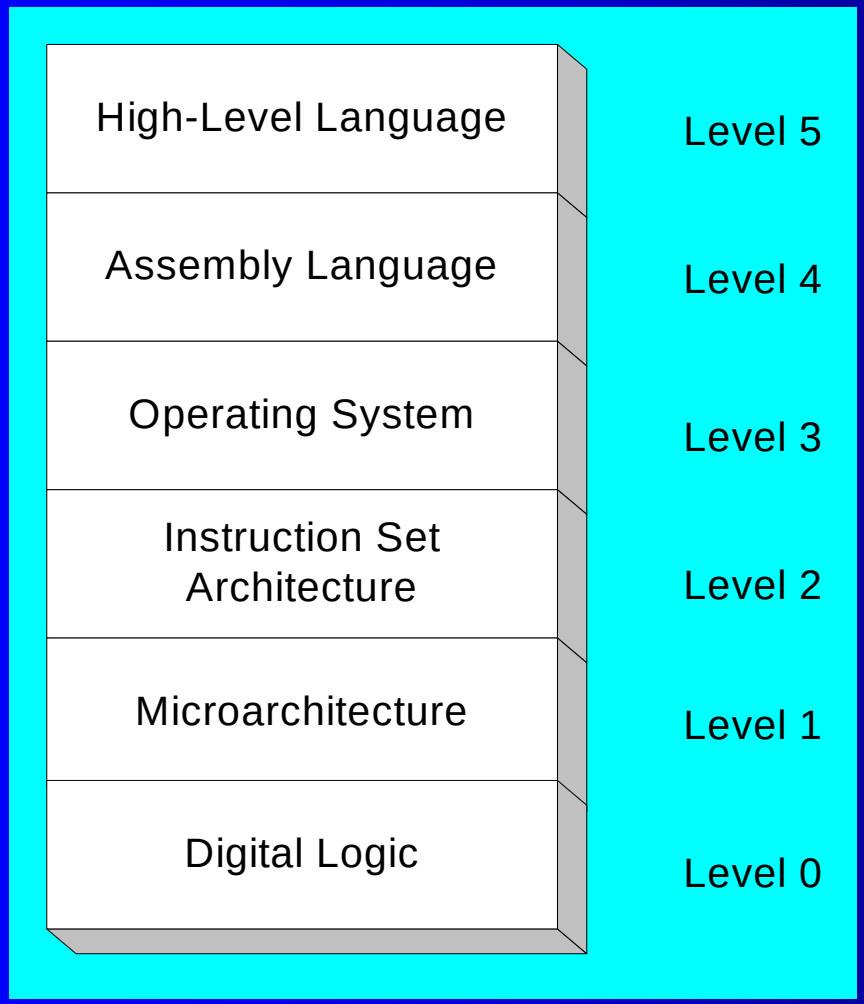
Assembly Language:

```
mov eax,A
mul B
add eax,C
call WriteInt
```

Intel Machine Language:

```
A1 00000000
F7 25 00000004
03 05 00000008
E8 00500000
```

Specific Machine Levels



The 6th Edition does not consider Level1, but we will keep this level.

Also, we consider Level 6, which may include Logical Symbolic Programming: Prolog, Lisp, Mathematica, MatLab.

Digital Logic

- Level 0
- CPU, constructed from digital logic gates
- System bus
- Memory
- Implemented using bipolar transistors

next: Data Representation

Microarchitecture

- Level 1
- Interprets conventional machine instructions (Level 2)
- Executed by digital hardware (Level 0)

Instruction Set Architecture

- Level 2
- Also known as conventional machine language
- Executed by Level 1 (microarchitecture) program

Operating System

- Level 3
- Provides services to Level 4 programs
- Translated and run at the instruction set architecture level (Level 2)

Assembly Language

- Level 4
- Instruction mnemonics that have a one-to-one correspondence to machine language
- Calls functions written at the operating system level (Level 3)
- Programs are translated into machine language (Level 2)

Irvine, Kip R. Assembly Language for Intel-Based Computers, 2003. Web site Examples

High-Level Language

- Level 5
- Application-oriented languages
  - C++, Java, Pascal, Visual Basic . . .
- Programs compile into assembly language (Level 4)

HPP, P8, P12, Sections Review