

A decorative graphic element consisting of a blue gradient shape that starts as a thin arc on the left and curves downwards and to the right, ending as a solid blue area in the bottom right corner.

Week 1

Chapter 1: Basic Concepts

Class 1

Chapter Overview

- **Welcome to Assembly Language**
- Virtual Machine Concept
- Data Representation
- Boolean Operations

Welcome to Assembly Language

- Some Good Questions to Ask
- Assembly Language Applications

Questions to Ask

- Why am I learning Assembly Language?
- What background should I have?
- What is an assembler?
- What hardware/software do I need?
- What types of programs will I create?
- What do I get with this book?
- What will I learn?

Welcome to Assembly Language *(cont)*

- How does assembly language (AL) relate to machine language?
- How do C++ and Java relate to AL?
- Is AL portable?
- Why learn AL?

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

Comparing ASM to High-Level Languages

Type of Application	High-Level Languages	Assembly Language
Commercial or scientific application, 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.	The language may not provide for direct hardware access. Even if it does, awkward coding techniques may be required, resulting in maintenance difficulties.	Hardware access is straightforward and simple. Easy to maintain when programs are short and well documented.
Commercial or scientific application written for multiple platforms (different operating systems).	Usually portable. The source code can be recompiled on each target operating system with minimal changes.	Must be recoded separately for each platform, using an assembler with a different syntax. Difficult to maintain.
Embedded systems and computer games requiring direct hardware access.	May produce large executable files that exceed the memory capacity of the device.	Ideal, because the executable code is small and runs quickly.

What's Next

- Welcome to Assembly Language
- **Virtual Machine Concept**
- Data Representation
- Boolean Operations

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);`

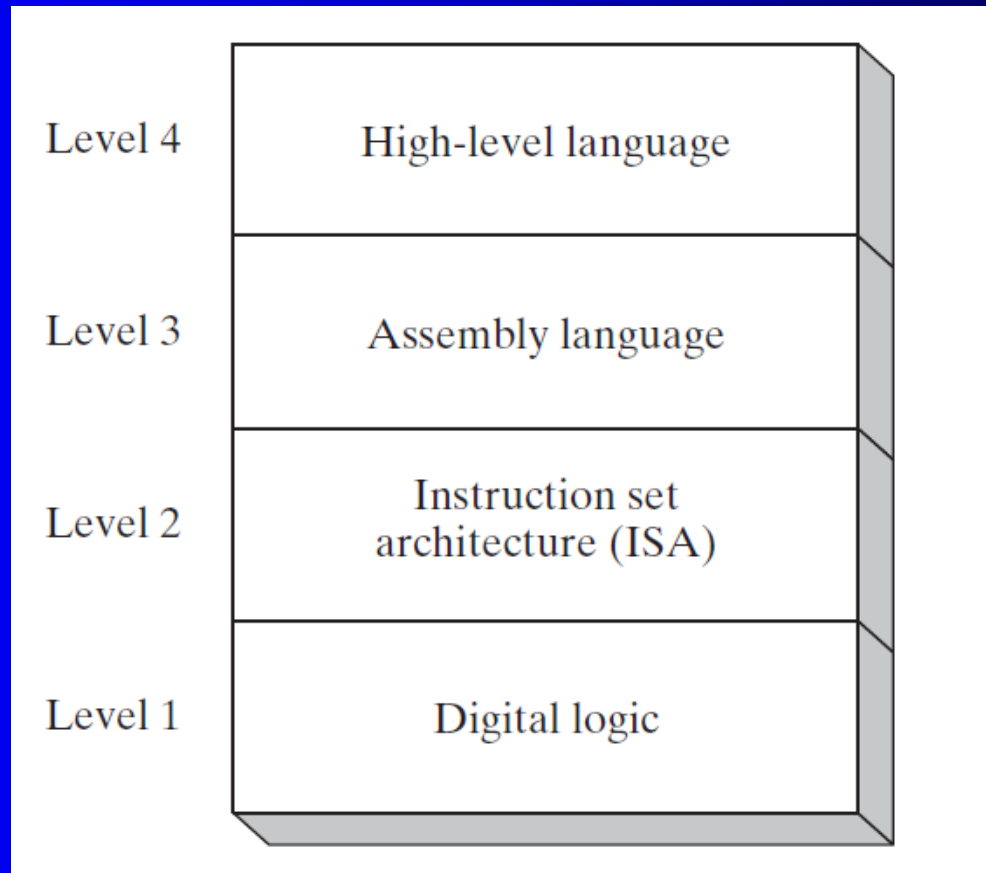
Assembly Language:

```
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



Level 4 - High-Level Language

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

Level 3 - Assembly Language

- Instruction mnemonics that have a one-to-one correspondence to machine language
- Programs are translated into Instruction Set Architecture Level - machine language (Level 2)

Level 2 - Instruction Set Architecture (ISA)

(see Book's Page 8 for details)

- Also known as conventional machine language
- Executed by Level 1 (Digital Logic)

Level 1 - Digital Logic

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

What's Next

- Welcome to Assembly Language
- Virtual Machine Concept
- **Data Representation**
- Boolean Operations

Data Representation

- Binary Numbers
 - Translating between binary and decimal
- Binary Addition
- Integer Storage Sizes
- Hexadecimal Integers
 - Translating between decimal and hexadecimal
 - Hexadecimal subtraction
- Signed Integers
 - Binary subtraction
- Character Storage

Binary Numbers

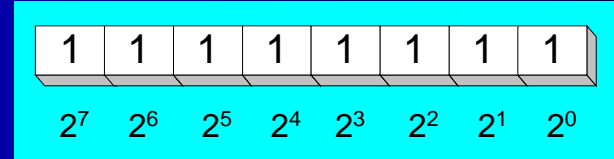
- Digits are 1 and 0
 - 1 = true
 - 0 = false
- MSB – most significant bit
- LSB – least significant bit

- Bit numbering:

MSB		LSB
	1 0 1 1 0 0 1 0 1 0 0 1 1 1 0 0	
15		0

Binary Numbers

- Each digit (bit) is either 1 or 0
- Each bit represents a power of 2:



Every binary number is a sum of powers of 2

Table 1-3 Binary Bit Position Values.

2^n	Decimal Value	2^n	Decimal Value
2^0	1	2^8	256
2^1	2	2^9	512
2^2	4	2^{10}	1024
2^3	8	2^{11}	2048
2^4	16	2^{12}	4096
2^5	32	2^{13}	8192
2^6	64	2^{14}	16384
2^7	128	2^{15}	32768