Introduction

1 Programming Languages

- High-level language such as C++ and Java
  - faster program development,
    - easier program maintenance,
    - portability as very little knowledge about the hardware is required.
- Assembly Language
  - a low-level programming language that is directly influenced by the instruction set and architecture of a processor
    - human understandable code requiring some understanding of microprocessor implementation
    - offers higher efficiency and accessibility to system hardware
- Machine Language: programs consists of binary or hexdecimal code.

2 Computer Systems

The basic blocks of a computer are

- the central processing unit (CPU) [Microprocessors] [8086 through Core i7],
  - is contained in a single chip and must be interfaced with peripheral support chips in order to function.
  - Microprocessors can be a general-purpose microprocessors (PC) or a dedicated microprocessors (mobile phone, DSP, ..)

Figure 1: Basic Computer Architecture

  - Typically, the CPU contains several registers (memory elements), the ALU, and the control unit.
  - An arithmetic logic unit (ALU) is a digital circuit which performs arithmetic and logic operations on two n-bit digital words.
  - The computer microprocessor is the main entity that determines the capability of different computerized systems

- the memory
  - Read-only memory (ROM) is a storage medium whose contents cannot normally be altered once programmed.
  - Random access memory (RAM) is also a storage medium for groups of bits or words whose contents can not only be read but also altered at specific addresses.
  - Cache memory is a high speed memory located on the processor.
  - A register can be considered as volatile storage for a number of bits that is part of the processor

- the input/output (I/O) system
  - Printer, Serial, Hard Drive, Mouse, optical drive, Keyboard, Monitor, Scanner
  - Many of these devices operation is based on interrupts (will be discussed later)

2.1 System Buses

- The bus refers to a number of conductors (wires) organized to provide a means of communication among different elements in a microcomputer system. Different bus types are
  - an address bus
    * An address is a pattern of 0's and 1's that represents a specific location of memory or a particular I/O device.
    * the address bus typically determines the max supported memory size
    * For I/O, a value between 0000H and FFFFH is issued.
  - a data bus, and
  - a control bus
    * at least four signals are required to determine the combination of memory or I/O operation and read or write operation
    * additional lines may exist depending on the supported functions.

- Common Bus Standards
  - ISA (Industry Standard Architecture): 8 MHz [8-bit (8086/8088), 16-bit (80286-Pentium)]
  - EISA: 8 MHz [32-bit (older 386 and 486 machines)]

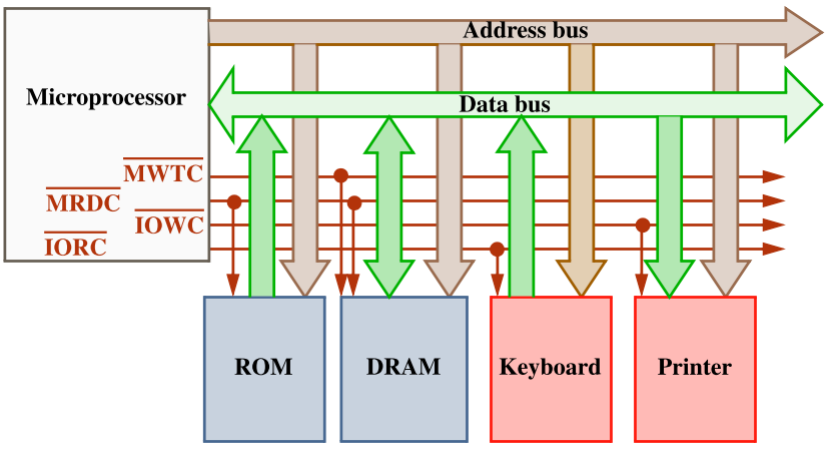


Figure 2: Bus Architecture

  - PCI (Peripheral Component Interconnect): 33 MHz[32-bit or 64-bit (Pentiums)]
  - PCI Express and PCI-X 533 MTS
  - VESA (Video Electronic Standards Association): Runs at processor speed. [32-bit or 64-bit (Pentiums)]
  - USB (Universal Serial Bus): 1.5 Mbps,12 Mbps and now 480 Mbps.
    * Serial connection to microprocessor and commonly used for keyboards, the mouse, modems and sound cards.
  - AGP (Advanced Graphics Port): 66MHz
    * Fast parallel connection: Across 64-bits for 533MB/sec.
    * For video cards.

- The presented components defines the capability of different computer systems.
- The computer can be programmed to perform a functions if the system hardware is capable of doing it. For example, we may not be able to perform decimal point operations unless the ALU is capable of doing so.
- The capabilities of any microprocessor is determined by its hardware and the supported ***instructions***.
- On studying an architecture, one should understand
  - the components of the microprocessor such as registers, flags, and ALU capabilities.
  - the supported instruction set required for writing programs
- In this course, we use 8086 microprocessor, which is considered a general purpose processor.
- On using any other processor in your system, one probably will follow the same footsteps by learning the processor internal architecture and supported instruction set.

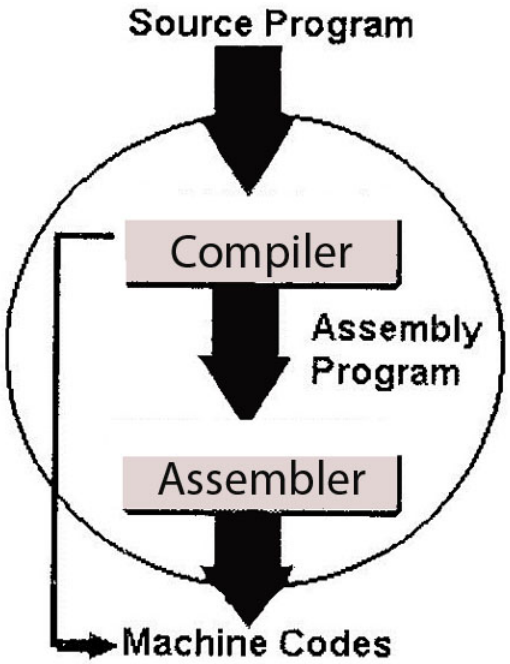


Figure 3: Computer Program Development

3 Data Representation

In this part, we present some of the fundamental concepts needed to implement and use a computer effectively, specifically we present the basics of number systems and codes used to represent the data to be processed by computers.

3.1 Number Systems

The microcomputer carries out all the arithmetic and logic operations internally using binary numbers. Because binary numbers are long, a more compact form using some other number system is preferable to represent them. The computer user finds it convenient to work with this compact form. Hence, it is important to understand the various number systems used with computers. Here in we consider decimal, Octal, and Hex-decimal *number representation*s.

3.1.1 Unsigned Number Repersentation

Decimal Number representation $125_{10} = 1 * 10^2 + 2 * 10^1 + 5 * 10^0$
$0.52 = 5 * 10^{-1} + 2 * 10^{-2}$

Binary Number Representation $101.01 = 1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2} = 5.25_{10}$

Octal Number Representation $25.32_8 = 2 * 8^1 + 5 * 8^0 + 3 * 8^{-1} + 2 * 8^{-2} = 21.40625_{10}$

Hexdecimal number representation

- In this representation, each digit may take 0-9 or A-F
- $23A_{16} = 2 * 16^2 + 3 * 16^1 + A * 16^0$

General Number System Repersentation Let

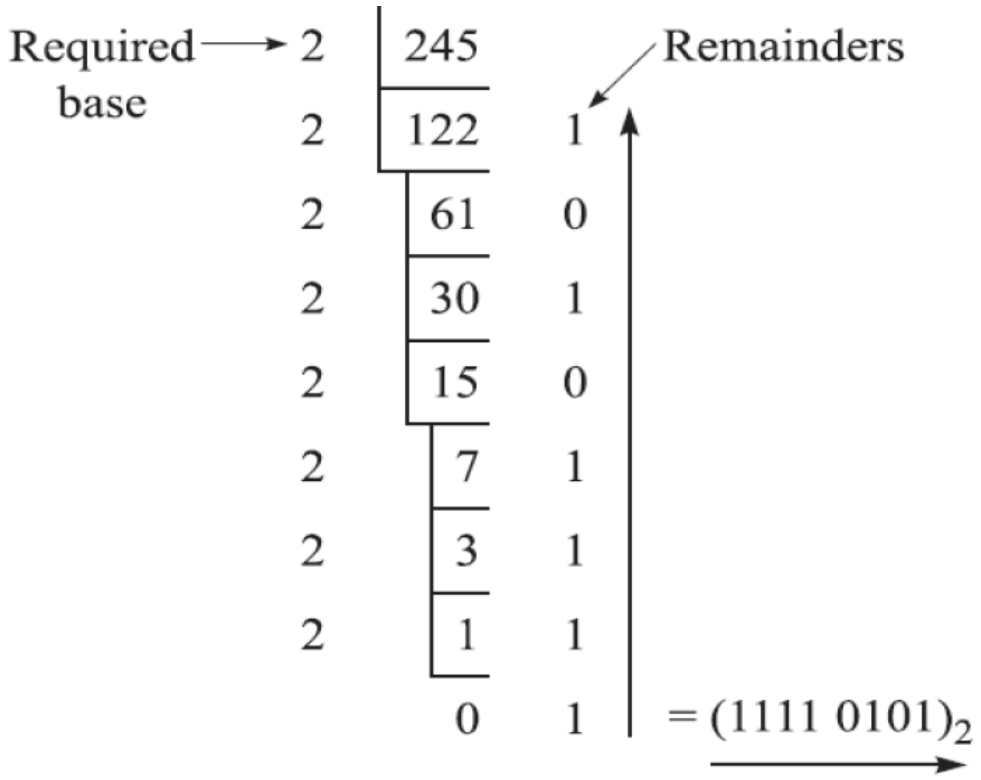
- b is the base or radix of the number system,
- p is the number of integer digits, and
- q is the number of fractional digits.

the number can be represented as $N = d_{p-1} * b^{p-1} + ... + d_0 * b^0 + d_{-1} * b^{-1} + ... + d_{-q} * b^{-q}$

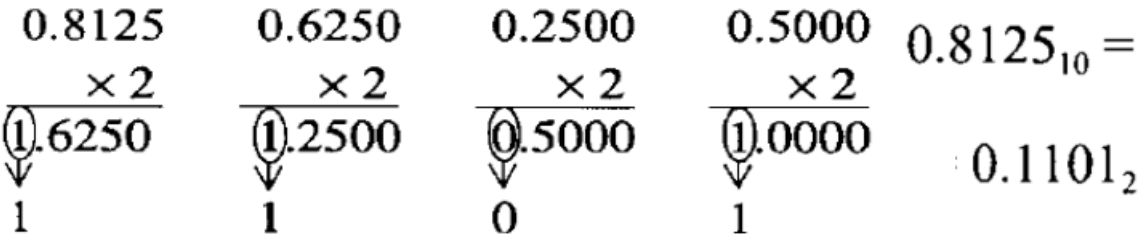
3.1.2 Number System Conversion

Converting Decimal representation to binary

- Convert 245 to its decimal equivalent.



- Considering Fraction conversion



- The conversion may not be exact due to limited word size

0.3615	0.7230	0.4460	0.8920	0.7840
$\times 2$	$\times 2$	$\times 2$	$\times 2$	$\times 2$
0.7230	1.4460	0.8920	1.7840	1.5680
0	1	0	1	1

$$0.01011_2 = 0 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5}$$

$$= 0.34375$$

Converting Decimal representation to Octal Convert 245 to its octal representaiton

8 | 245		
8 | 30	5	↑
8 | 3	6	
0	3	$= (365)_8$

Conversion between binary, octal, and hexdecimal systems

- The conversion between these systems is straight forward. Remember that octal and hexdecimal are compact forms of binary numbers.

$$1001_2 \Leftrightarrow 11_8 \Leftrightarrow 9_{16}$$

- To convert from binary, one simply group a suitable number of bits to the target representation and write down the equivalent number for each group of bits.

3.1.3 Signed Number Representations

In computing, signed number representations are required to encode negative numbers in binary number systems. The most two common representations are

- sign/magnitude
- two's complement

Sign/magnitude

- most significant bit (MSB) is used as a sign bit, N-1 bits for magnitude
- 0 in sign bit $\rightarrow$ +ve number
- $5_{10} = 0101_2$ and $-5_{10} = 1101_2$ [N=4]
- Diadv: does not work for ordinary binary addition
- $0101_2 + 1101_2 = 10010_2$!!
- Range $[-2^{N-1} + 1,\ 2^{N-1} - 1]$
- Some early binary computers (e.g., IBM 7090) used this representation

Two's Complement

- taking the two's complement:
  - Method #1: inverting all of the bits in the number, then adding 1
  - Method #2: Starting from the right, find the first '1' and Invert all of the bits to the left of that one
- MSB = 0 $\rightarrow$ +ve

00000000	0
00000001	1
...	...
01111110	126
01111111	127
10000000	−128
10000001	−127
10000010	−126
...	...
11111110	−2
11111111	−1

- The most positive number $01...111_2 = 2^{N-1} - 1$
- The most negative number $10...000_2 = -2^{N-1}$ (N=8 $\rightarrow -128$)

- Addition works properly for both positive and negative numbers
- $5 + (-5) = 0101_2 + 1011_2 = 10000_2$
- Zero is represented by all zeros and is considered positive because its sign bit is 0.
- overflow may occur if two numbers being added have the same sign bit and the result has the opposite sign bit. (try 4+5 assume N=4)

References

M. RAFIQUZZAMAN, "Fundamentals of Digital Logic and Microcomputer Design," Fifth Edition.