Machine Code, Number System, and C Variables and Operations

A. Sahu and P. Mitra
Dept of Comp. Sc. & Engg.
Indian Institute of Technology Guwahati

Outline

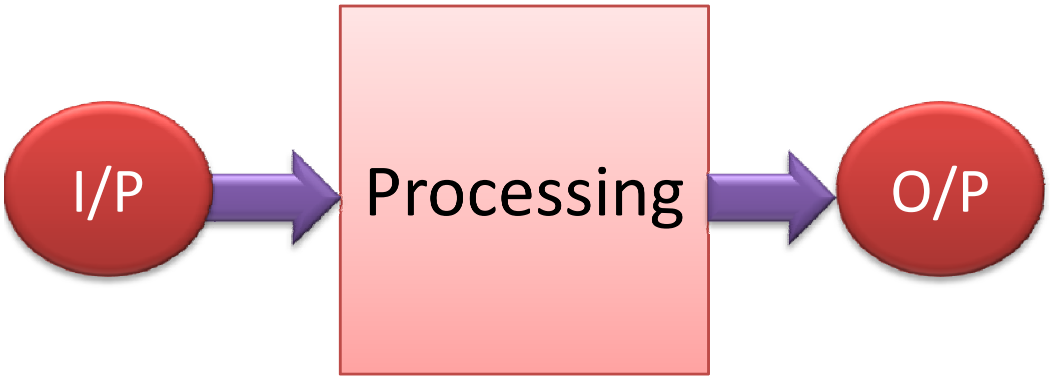
- Compiler Phases for C Programming
  - Compiler, Assembler, Linker, Loader
  - Source, Assembly, Object, Executable
- Number System
  - Binary, Octal and Hex
- Flow Charts
- C Programming: Variable, data type and operations

Quick Recap

What Is A Computer?

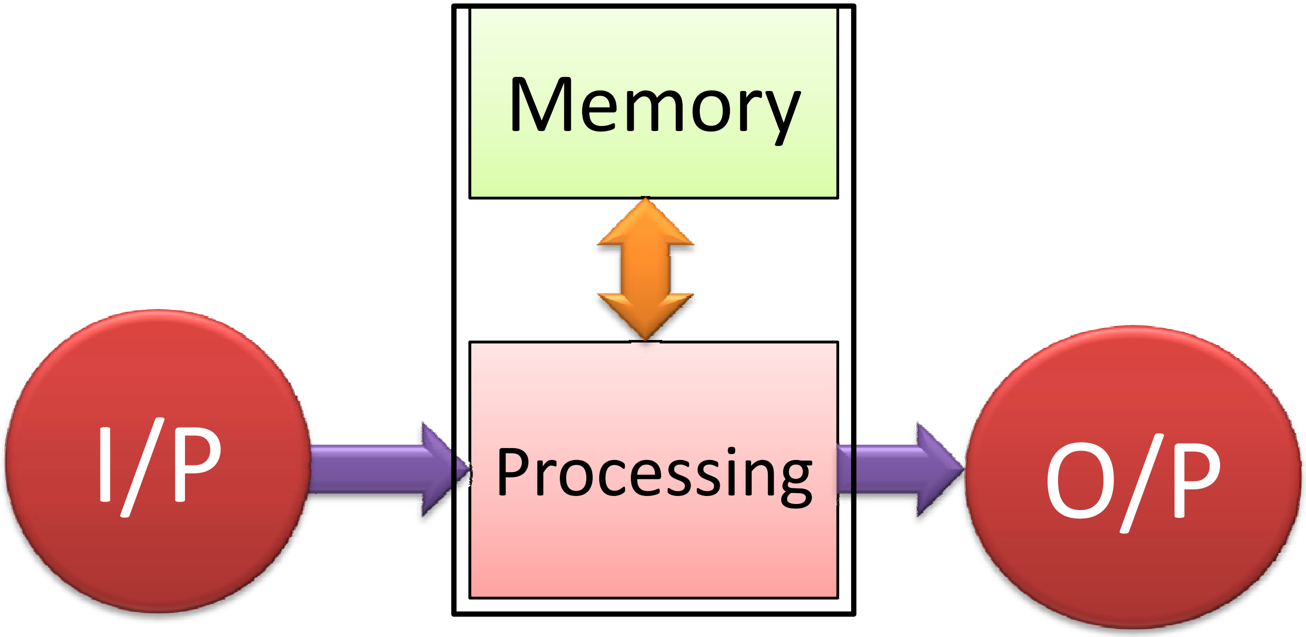
- An electronic device
- Operates under control of instructions (software)
- Stored in its own memory unit
- It can
  - Accept data (input),
  - Manipulate data (process),
  - Produce output from the processing.
- A collection of devices that function together as a system.

Computer System



- Keyboard, Mouse : Input
- Speaker, Monitor/Display : Output
- CPU Box : Processing

Computer System: Von Newman

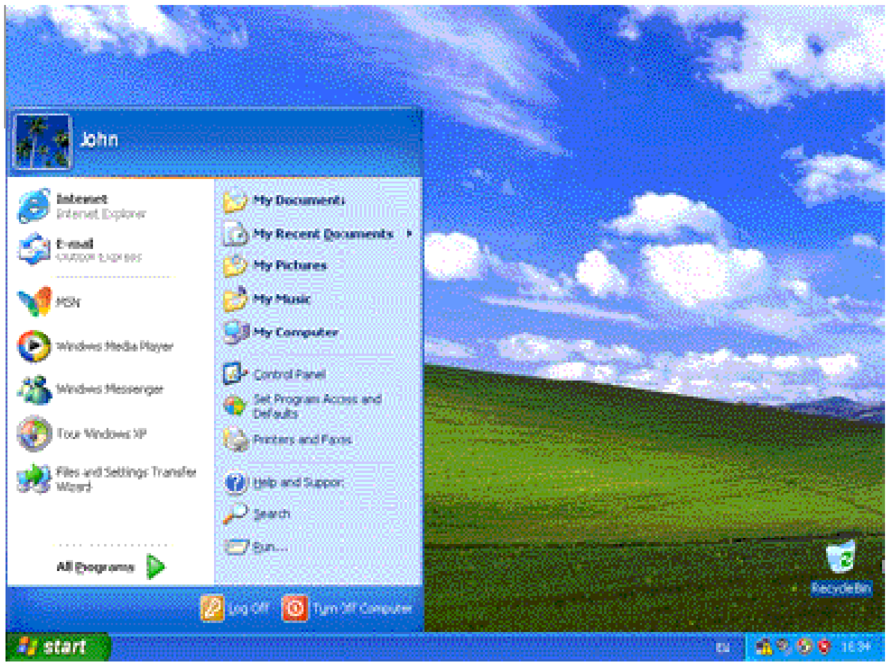


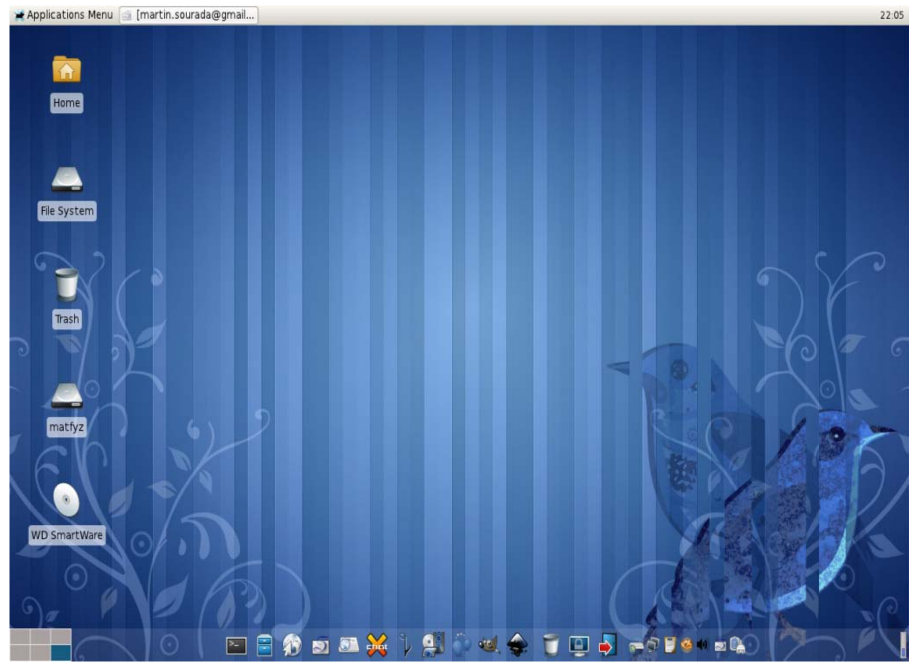
- Input and Output
- Processing
- Memory : Where it store
  - Instruction, Data, Intermediate compute

Computer System: When you Switch on

- Operating System boots from Hard disk
- OS : Give you an environment where you work
- Different OS
  - Window XP/Vista/7
  - Linux: Fedora, Ubuntu, Debian
- Application can be invoked by clicking some icon
- Application: Word, Excel, Internet Explorer, Mozilla, Media Player

<u>Screen shot : Window & Linux</u>





- Both are equally Good and Powerful
- Window: User friendly, prone to Virus, Commercial (Not Free: you have to Pay Money)
- Linux : Robust and Freely available

How to create your own Application?

- **Operating System** : Linux, Window
- **Applications:** Word/Excel, Media Player, Mozilla, Explorer, etc
  - We use it to do some tasks, **but don't know inside contents**
- Programming helps us to create our own application
  - Already solution approach is known
  - Draw the flow chart, write Pseudocode
  - Write code, compile, run and test

Programming: Purpose?

- **Programming**
  - Purpose : to create a program that performs specific operations or exhibits a certain desired behavior.
  - Computer Language (C , C++, Java, Fortran, Cobol)
  - Design our own application
  - Almost from the beginning
  - Understanding how software/application works

How to do programming

- Problem: Specification
  - Example: Compute sum of first N natural number
  - Define Input {N}, Output {SUM}
  - How to do : Flow chart
  - Write the C/C++/{*} Code in Note/Paper
- Program
  - Sequence of Instructions and Data
  - Can be run by
    - Compiling and running
    - Interpreting

How to do programming

- OS, Shell, IDE, Editor:
  - **Linux, Bash Shell**, **gedit/VI**/Pico
  - **Word Processor is not used to write program**
  - Integrated Development Environment: GUI Based
    - TurboC/VisualC++/Kdevelop/Dev GUI
- Use the Program (Method 1)
  - Compiling: **GCC**, TCC, VCC
  - Running: ./a.out
- Use the Program (Method 2)
  - Interpret the program and run

Interpreter Vs Compiler

- Interpreter
  - Examples: Shell/Command Prompts, ML, Perl, Python, Matlab
  - Read code line by line and execute, sequential
  - **Basic syntax Errors occur at run time**
- Compiler
  - Example: C, C++, Java
  - Read whole code together, make an executable and run the executable
  - **Basic syntax Errors don't occur at run time, only logical and runtime error occurs**

Interpreter Example

- Interpreter
  - Examples: basic command line calculator of Linux
  - $bc

    3*4

    12

    6+ (3+2)^2

    31
- Doing small computation easier
- Interpreter can read from file and execute line by line, example shell script

Writing and Compiling C Program under Linux

```
#include <stdio.h>

int main(){

  printf(“Hello world”);

  return 0;
}
```

Header file: Standard Input/Output

Starting of program

Printing message

End of program

Compiling program: test.c

- Compiling : $gcc test.c
- Listing : $ls

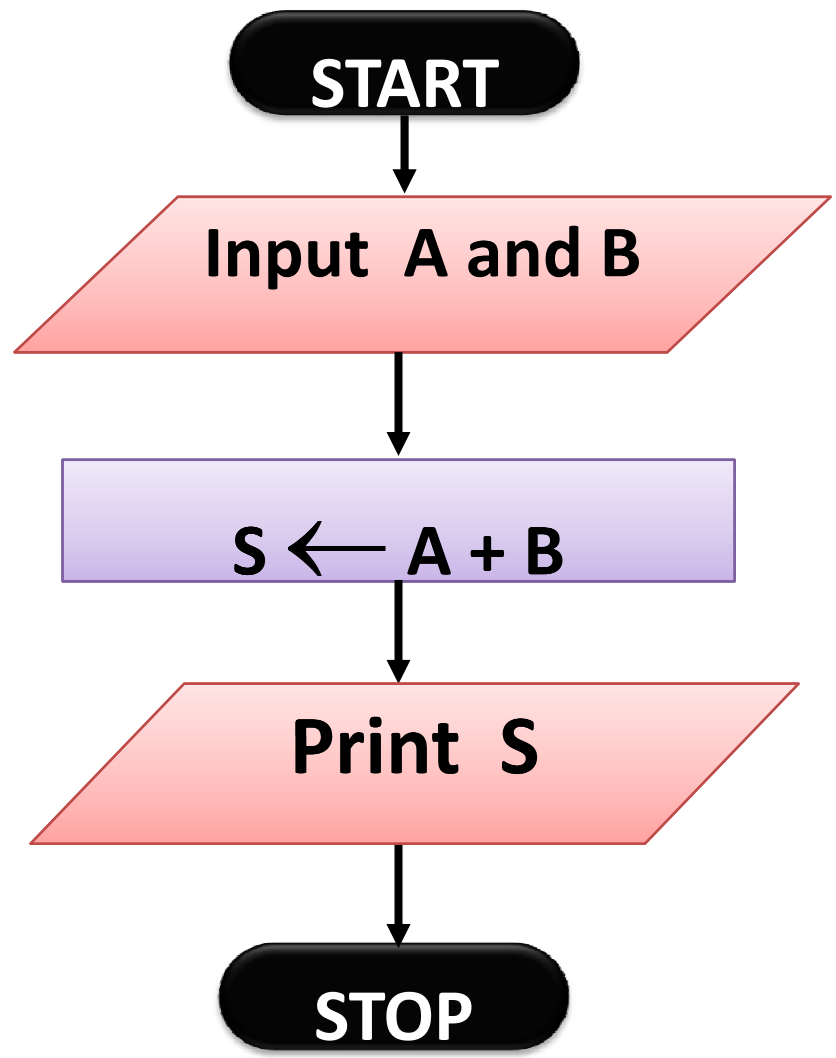
  test.c a.out
- Execute the program : $./a.out

  Hello world

Example 1: Adding two number

- Step 1: Input A and B
- Step 2: S $\leftarrow$ A + B
- Step 3: Print S

Flowchart



Sum A+B : Input and output

```
#include <stdio.h>

int main(){
 int A,B, S;
 printf("Enter two
          numbers ");
 scanf("%d %d",&A,&B);

 S=A+B;

 printf("Res=%d", S);
 return 0;
}
```

Header file:
Standard Input/Output

Printing message

Asking for inputs

Compute

Output Result

Compiling program: test.c

- Compiling : $gcc -Wall test.c

  It is advisable to use –Wall option to raise all warnings of the code

- Listing using $ls

  test.c a.out

- Executing : $./a.out

  Input two numbers 5 7

  Res=12

Compiling program: Object file, assembly file, exe file

$gcc –S test.c

Generate **assembly** language file **.s**

$ gcc test.s

$gcc –c test.c

Generate **object** file

$gcc test.o

Generate **executable** file

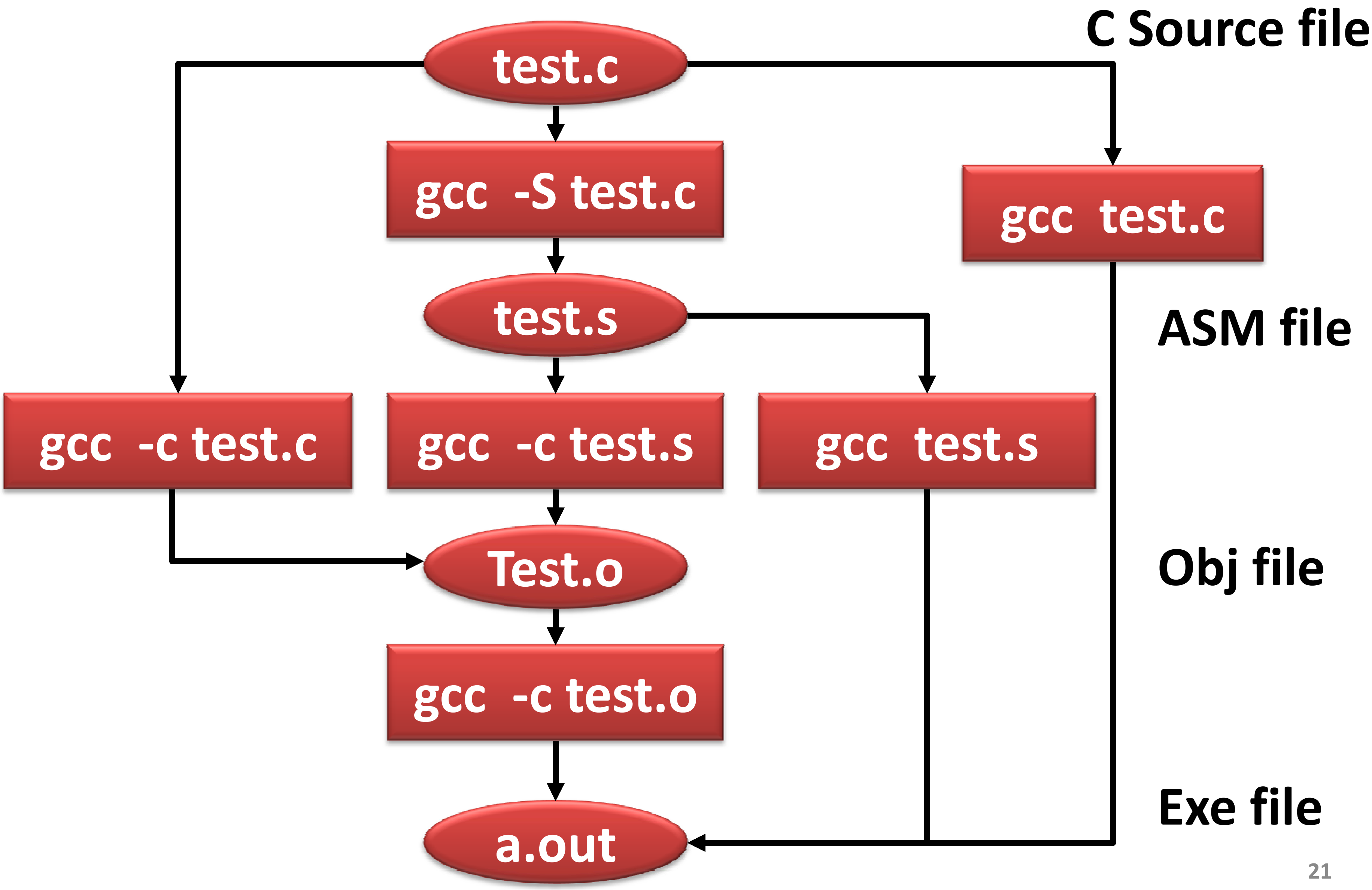
$./a.out

run the executable

Source, Assembly TXT

Object, EXE Binary

Compilation flow



Human-Readable Machine Language

- Computers like ones and zeros…

```
0001110010000110
```

- Humans like symbols…

```
ADD  R6,R2,R6; increment index reg.
```

- Assembler is a program that turns symbols into machine instructions.
  - ISA-specific: Close correspondence between symbols and instruction set
    - Mnemonics for opcodes
    - Labels for memory locations
  - additional operations for allocating storage and initializing data

Object File Format

- Object file contains
  - Starting address (location where program must be loaded), followed by Machine instructions
- Example
  - Beginning of “count character” object file looks like this:

```
0011000000000000   <- .ORIG x3000
0101010010100000   <- AND R2, R2, #0
0010011000010001   <- LD R3, PTR
1111000000100011   <- TRAP x23
```

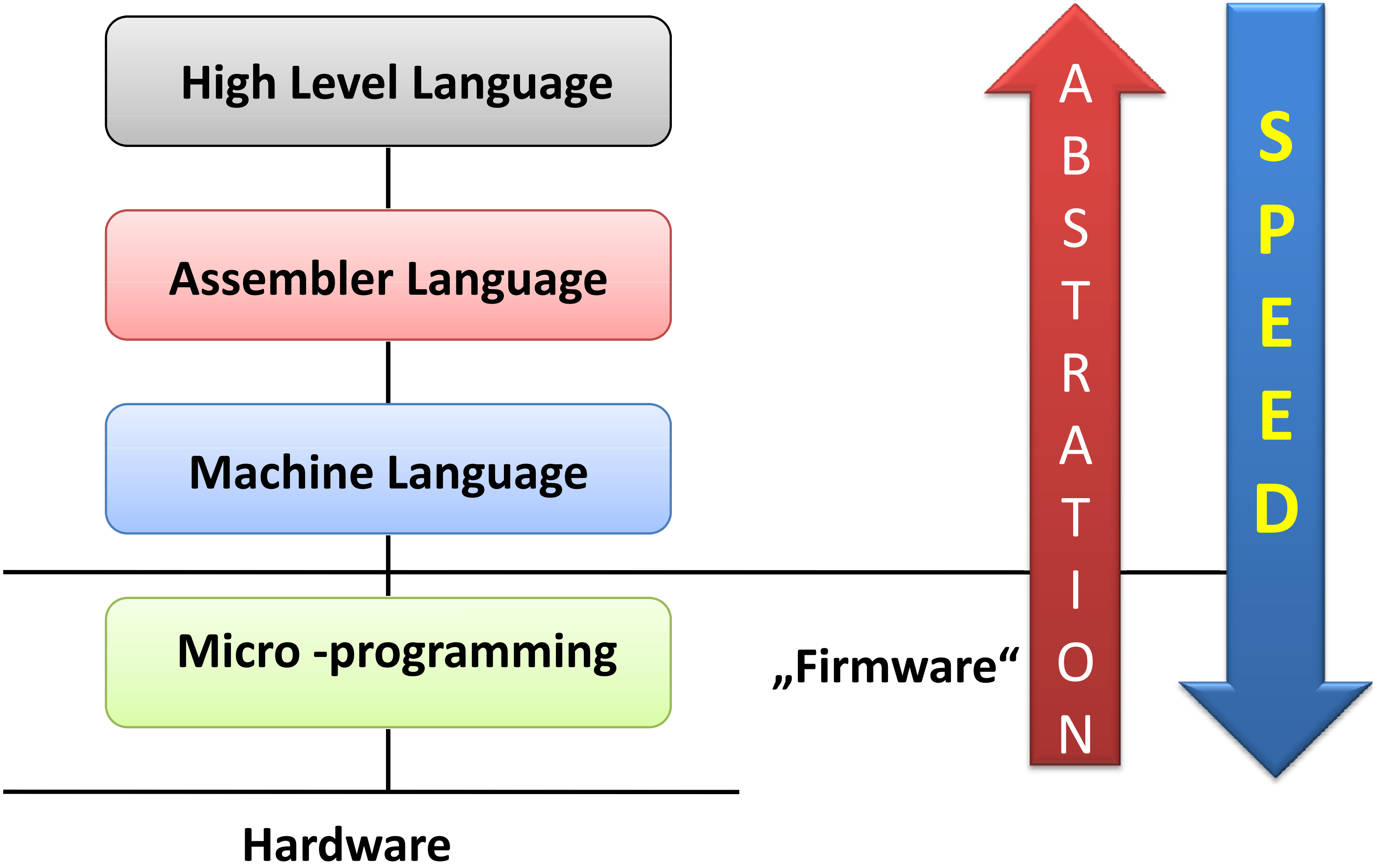
%

Assembly Language: Human Readable

```
include 'system.inc'
section .data
  hello db 'Hello, World!',
           0Ah hbytes equ $-hello
section .text
global _start
_start:
     push dword hbytes
     push dword hello
     push dword stdout
     sys.write
     push dword 0
sys.exit
```

```
$nasm -f elf hello.asm
$ld -s -o hello hello.o
$./hello
Hello, World!
```

Language Levels



High Level to Micro Code

- **High Level language**
  - Formulating program for certain application areas
  - Hardware independent
- **Assembler languages**
  - Machine oriented language
  - Programs orient on special hardware properties
  - More comfortable than machine code (e.g. by using symbolic notations)

High Level to Micro Code

- Machine code:
  - Set of commands directly executable via CPU
  - Commands in numeric code
  - Lowest semantic level
  - Generally 2 executing oportunities:
    - Interpretiv via micro code
    - Directly processing via hardware

High Level to Micro Code

- Micro programming:
  - Implementing of executing of machine commands (Control unit - controller)
  - Machine command executed/shown as sequence of micro code commands
  - Micro code commands:
    - Simpliest process controlling
      - Moving of data, Opening of grids, Tests

Number System (binary, octal, dec and hexa decimal)

Computer: Number System

- Computers like ones and zeros…

$$\boxed{\texttt{0001110010000110}}$$

- Humans like symbols…
- We need to know: binary number system
  - Also Octal and Hex number system
  - Type conversions

Famous Number System

- Decimal System: 0 -9
  - May evolves: because human have 10 finger
- Roman System
  - May evolves to make easy to look and feel
  - Pre/Post Concept: (IV, V & VI) is (5-1, 5 & 5+1)
- Binary System, Others (Oct, Hex)
  - One can cut an apple in to two

Significant Digits

Binary: 11101101

Most significant digit ***Least significant digit***

Decimal: 1063079

Most significant digit ***Least significant digit***

Decimal (base 10)

- Uses positional representation
- Each digit corresponds to a power of 10 based on its position in the number
- The powers of 10 increment from 0, 1, 2, etc. as you move right to left
  - $1,479 = 1 * 10^3 + 4 * 10^2 + 7 * 10^1 + 9 * 10^0$

Binary (base 2)

- Two digits: 0, 1
- To make the binary numbers more readable, the digits are often put in groups of 4
  - $1010 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0$
    $= 8 + 2$
    $= 10$
  - $1100\ \ 1001 = 1 * 2^7 + 1 * 2^6 + 1 * 2^3 + 1 * 2^0$
    $= 128 + 64 + 8 + 1$
    $= 201$

How to Encode Numbers: Binary Numbers

- Working with binary numbers
  - In base ten, helps to know powers of 10
    - One, Ten, Hundred, Thousand, ...
  - In base two, helps to know powers of 2
    - One, Two, Four, Eight, Sixteen, ..
    - Count up by powers of two

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1

Important Property of Binary Number

- Number of different number can be possible for a N bit binary number
  - 2^N, for 2 bit number it is 4 (00, 01,10 and 11)
- Summation two N bit binary number cannot exceed $2*2^N$
- $2^0+2^1+2^2+2^3+\ldots+2^N=2^{N+1}-1$ example
  - 1+2+4+8=15=16-1

Octal (base 8)

- Shorter & easier to read than binary
- 8 digits: 0, 1, 2, 3, 4, 5, 6, 7,
- **Octal numbers to Decimal**

$$136_8 = 1 * 8^2 + 3 * 8^1 + 6 * 8^0$$

$$= 1 * 64 + 3 * 8 + 6 * 1$$

$$= 94_{10}$$

Hexadecimal (base 16)

- Shorter & easier to read than binary
- 16 digits:
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, **A, B, C, D, E, F**
- "0x" often precedes hexadecimal numbers

$$
\begin{aligned}
0x123 &= 1 * 16^2 + 2 * 16^1 + 3 * 16^0 \\
&= 1 * 256 + 2 * 16 + 3 * 1 \\
&= 256 + 32 + 3 \\
&= 291
\end{aligned}
$$

Counting

Dec	Binary	Oct	Hex	Dec	Binary	Oct	Hex
0	00000	0	0	8	01000	10	8
1	00001	1	1	9	01001	11	9
2	00010	2	2	10	01010	12	A
3	00011	3	3	11	01011	13	B
4	00100	4	4	12	01100	14	C
5	00101	5	5	13	01101	15	D
6	00110	6	6	14	01110	16	E
7	00111	7	7	15	01111	17	F
8	01000	10	8	16	10000	20	10

Fractional Number

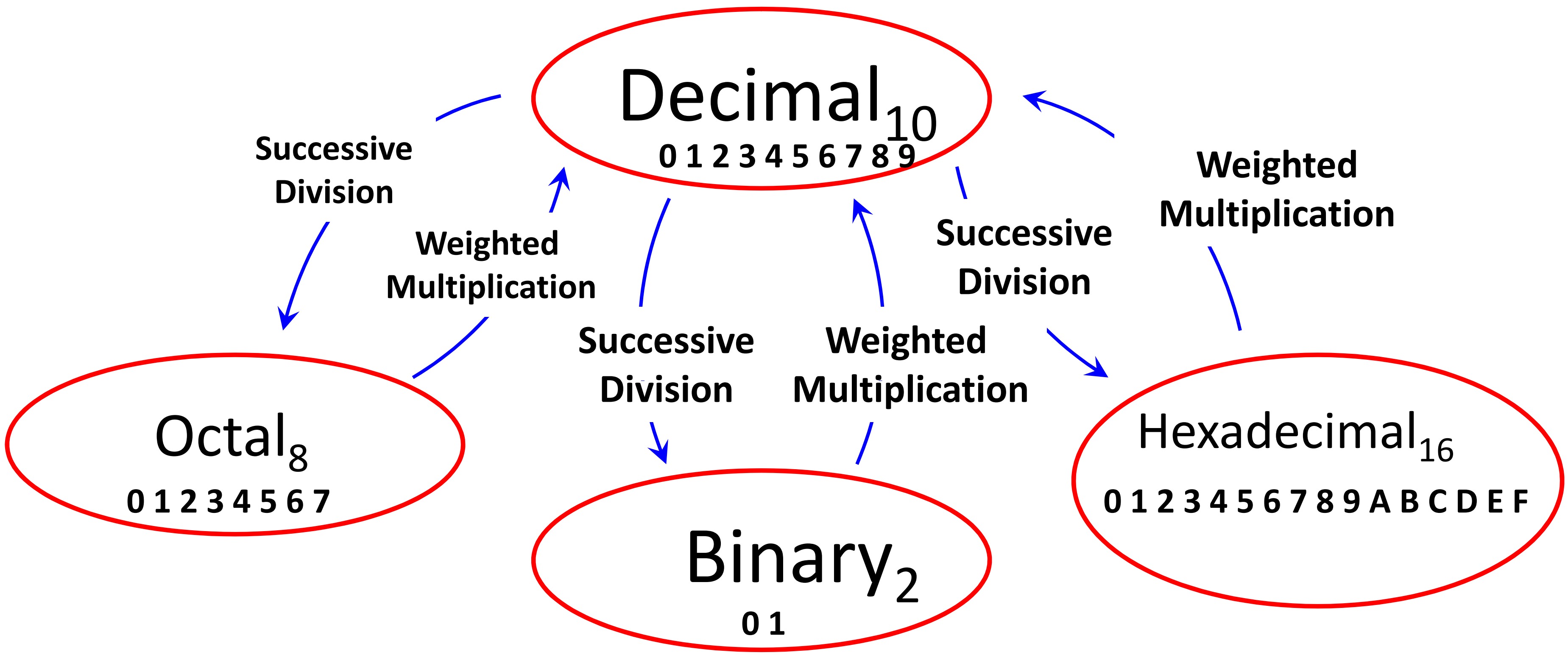
- Point: Decimal Point, Binary Point, Hexadecimal point
- Decimal

  $247.75 = 2x10^{2}+4x10^{1}+7x10^{0} + 7x10^{-1}+5x10^{-2}$
- Binary

  $10.101= 1x2^{1}+0x2^{0} + 1x2^{-1}+0x2^{-2}+1x2^{-3}$
- Hexadecimal

  $6A.7D=6x16^{1}+10x16^{0} + 7x16^{-1}+Dx16^{-2}$

Converting To and From Decimal



Decimal ↔ Binary

$Base_{10}$ DECIMAL → **Successive Division** → $Base_2$ BINARY

a) Divide the decimal number by **2**; the remainder is the LSB of the **binary** number.

b) If the quotation is zero, the conversion is complete. Otherwise repeat step (a) using the quotation as the decimal number. The new remainder is the next most significant bit of the **binary** number.

$Base_2$ BINARY → **Weighted Multiplication** → $Base_{10}$ DECIMAL

a) Multiply each bit of the **binary** number by its corresponding bit-weighting factor (i.e., Bit-0→2^0=1; Bit-1→2^1=2; Bit-2→2^2=4; etc).

b) Sum up all of the products in step (a) to get the decimal number.

Decimal to Binary : Division Method

- Divide decimal number by 2 and insert remainder into new binary number.
  - Continue dividing quotient by 2 until the quotient is 0.
- Example: Convert decimal number 12 to binary

12 div 2 = (Quo=6 , Rem=0) LSB

6 div 2 = (Quo=3, Rem=0)

3 div 2 = (Quo=1,Rem=1)

1 div 2 = (Quo=0, Rem=1) MSB

$$\boxed{12_{10} = 1\,1\,00_{2}}$$

Decimal to Octal Conversion

The Process: Successive Division

- Divide number by **8**; R is the LSB of the **octal** number
- While Q is 0
  - Using the Q as the decimal number.
  - New remainder is MSB of the **octal** number.

$$8\,\overline{)\,94}^{\,11} \qquad r = 6 \quad \leftarrow \quad \text{LSB}$$

$$8\,\overline{)\,11}^{\,1} \qquad r = 3$$

$$8\,\overline{)\,1}^{\,0} \qquad r = 1 \quad \leftarrow \quad \text{MSB}$$

$$94_{10} = 136_8$$

Decimal to Hexadecimal Conversion

The Process: Successive Division

- Divide number by **16**; R is the LSB of the **hex** number
- While Q is 0
  - Using the Q as the decimal number.
  - New remainder is MSB of the **hex** number.

$$16\overline{\smash{)}\,94}^{\;5} \qquad r = E \quad \leftarrow \text{LSB}$$

$$16\overline{\smash{)}\,5}^{\;0} \qquad r = 5 \quad \leftarrow \text{MSB}$$

$$\boxed{94_{10} = 5E_{16}}$$

Substitution Code

Convert $1110\ 0110\ 1010_2$ to hex using the 4-bit substitution code :

$$\overbrace{1110}^{E}\ \overbrace{0110}^{6}\ \overbrace{1010}^{A}$$

$$\boxed{E6A_{16}}$$

Thanks