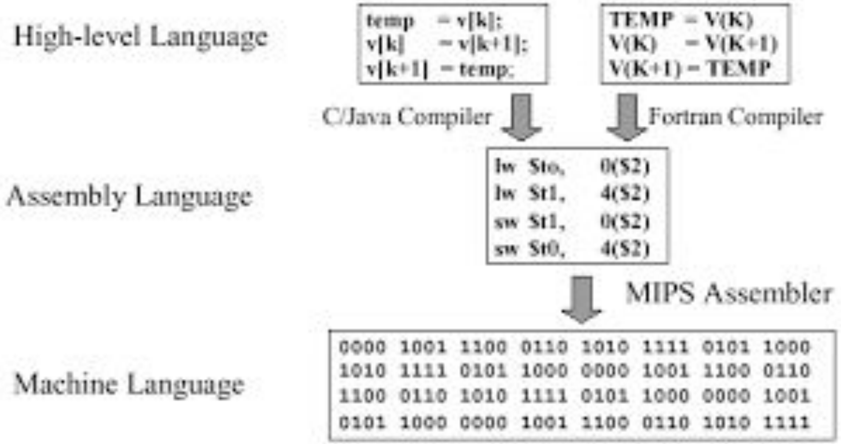
Fundamentals of IT

Programming Language

Programming Language

- It is a set of rules that provides a way of telling a computer what operations to perform
- It is a set of rules for communicating an algorithm
- It is a notational system for describing computation in machine-readable or human-readable form
- A programming language is a tool for developing executable models for a class of problem domain



High-level Language

```
temp    = v[k];
v[k]    = v[k+1];
v[k+1]  = temp;
```

```
TEMP   = V(K)
V(K)   = V(K+1)
V(K+1) = TEMP
```

C/Java Compiler

Fortran Compiler

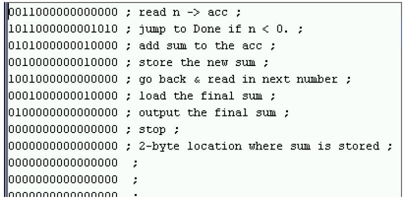
Assembly Language

```
lw  $to,   0($2)
lw  $t1,   4($2)
sw  $t1,   0($2)
sw  $t0,   4($2)
```

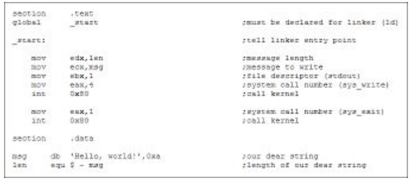
MIPS Assembler

Machine Language

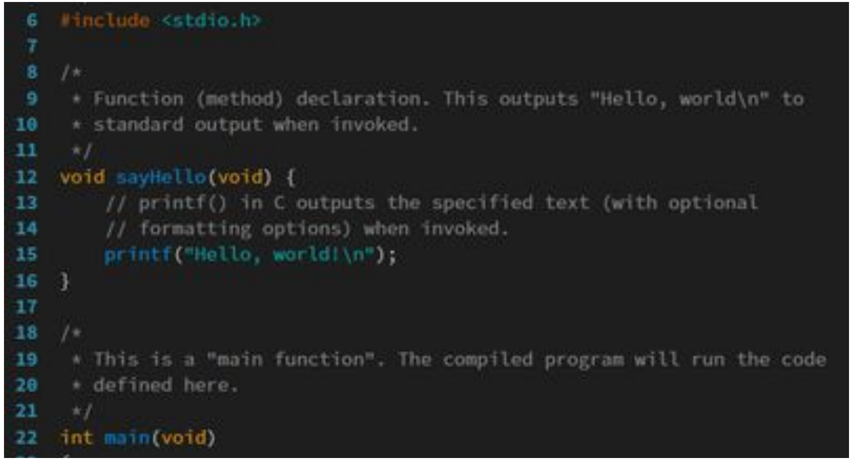
```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```



```
0011000000000000 ; read n -> acc ;
1011000000001010 ; jump to Done if n < 0. ;
0101000000010000 ; add sum to the acc ;
0010000000010000 ; store the new sum ;
1001000000000000 ; go back & read in next number ;
0001000000010000 ; load the final sum ;
0100000000000000 ; output the final sum ;
0000000000000000 ; stop ;
0000000000000000 ; 2-byte location where sum is stored ;
0000000000000000  ;
0000000000000000  ;
0000000000000000  ;
```



```
section     .text
global      _start                              ;must be declared for linker (ld)

_start:                                         ;tell linker entry point

    mov     edx,len                             ;message length
    mov     ecx,msg                             ;message to write
    mov     ebx,1                               ;file descriptor (stdout)
    mov     eax,4                               ;system call number (sys_write)
    int     0x80                                ;call kernel

    mov     eax,1                               ;system call number (sys_exit)
    int     0x80                                ;call kernel

section     .data

msg     db  'Hello, world!',0xa                 ;our dear string
len     equ $ - msg                             ;length of our dear string
```



```
#include <stdio.h>

/*
 * Function (method) declaration. This outputs "Hello, world\n" to
 * standard output when invoked.
 */
void sayHello(void) {
    // printf() in C outputs the specified text (with optional
    // formatting options) when invoked.
    printf("Hello, world!\n");
}

/*
 * This is a "main function". The compiled program will run the code
 * defined here.
 */
int main(void)
```

High Level vs Low Level Language

High Level Language	Low Level Language
It is programmer friendly	It is machine friendly
Less memory efficient	High memory efficient
Easy to debug	Complex to debug
It is portable	It is non portable
It is machine independent	It is machine dependent
It needs compiler or interpreter for translation	It needs assembler for translation

Compiled vs Interpreted

	Compiler	**Interpreter**
Input	Takes entire program as its input	Takes a single line of code or instruction as input
Output	Generate intermediate object code	Does not generate object code
Speed	Executes faster	Slower
Memory	Requires more memory to create object code	Requires less memory
Workload	Doesn’t need to compile every single time	Converts to machine level code at execution every time
Errors	Displays errors once the entire program is checked	Displays error when each instruction is run

Types of Programming language

1. First Generation Languages
2. Second Generation Languages
3. Third Generation Languages
4. Fourth Generation Languages
5. Fifth Generation Languages

First Generation Language (1GL)

- 1GL is a grouping of programming languages that are machine level languages used to program first generation computers
- Very efficient code but very difficult to write
- Operation code such as addition or substaction

Second Generation Language

- Assembly Languages
- Symbolic code replaced binary operation code
- Assembler is required to translate it into machine code
- Very efficient code and easy to write

Third Generation Language

- High level languages such as FORTRAN, COBOL
- Closer to English but included simple mathematical notations
- Programs written in source code which must be translated into object code
- Uses compiler/interpreter

Fourth Generation Language

- 4GL is grouping of programming languages that attempts to get closer to human language, form of thinking and conceptualization
- It requires fewer instructions than 3GL to accomplish a task
- 4GLs were designed to reduce overall time, effort and cost of a software development
- Examples: Python, Ruby, SQL etc

Fifth Generation Languages

- 5GL is any programming language based on problem solving using constraints given to the program, rather than using an algorithm
- Examples: OPS5, PROLOG
- Such languages are primarily developed for fields such as AI and technology

Other Programming Languages

- Structured Programming Languages
- Object Oriented Programming Languages
- Scripting Languages
- Command Languages
- Text Processing Languages
- Mark-Up Languages
- Query Languages
- Visual Programming Languages