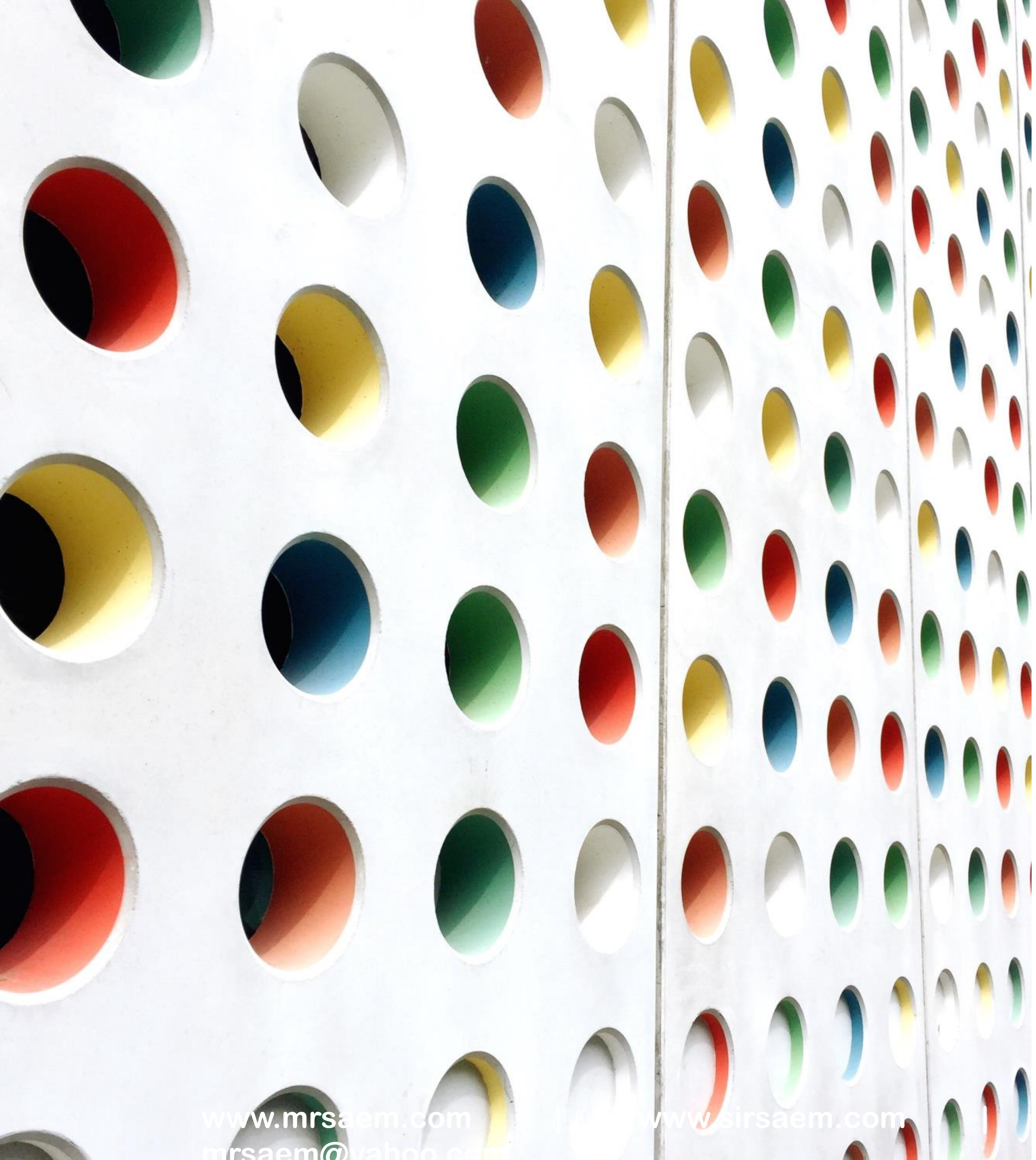


Chapter 6:

Assembly Language Part 01

CS With Mr Saem

Introduction to Assembly Language

Assembly Language is a low-level programming language. It helps in understanding the programming language to machine code.

In computer, there is assembler that helps in converting the assembly code into machine code executable.

Assembly language is designed to understand the instruction and provide to machine language for further processing.

It mainly depends on the architecture of the system whether it is the operating system or computer architecture.

Assembly Language mainly consists of mnemonic processor instructions or data, and other statements or instructions.

It is produced with the help of compiling the high-level language source code like C, C++. Assembly Language helps in fine-tuning the program.

A Level
CS With Mr Saem

Why is Assembly Language Useful?

Assembly language helps programmers to write the human-readable code that is almost similar to machine language.

Machine language is difficult to understand and read as it is just a series of numbers.

Assembly language helps in providing full control of what tasks a computer is performing.

Why you should learn Assembly Language?

The learning of assembly language is still important for programmers. It helps in taking complete control over the system and its resources.

By learning assembly language, the programmer is able to write the code to access registers and able to retrieve the memory address of pointers and values.

It mainly helps in speed optimization that increase efficiency and performance.

Assembly language learning helps in understanding the processor and memory functions. If the programmer is writing any program that needs to be a compiler that means the programmer should have a complete understanding of the processor.

Assembly language helps in understanding the work of processor and memory. It is cryptic and symbolic language.

Why you should learn Assembly Language?

Assembly Language helps in contacting the hardware directly. This language is mainly based on computer architecture and it recognizes the certain type of processor and its different for different CPUs.

Assembly language refers as transparent compared to other high-level languages. It has a small number of operations but it is helpful in understanding the algorithms and other flow of controls.

It makes the code less complex and easy debugging as well.

Features

The features of the assembly language are mentioned below:

1. It can use mnemonic than numeric operation code and it also provides the information of any error in the code.

2. This language helps in specifying the symbolic operand that means it does not need to specify the machine address of that operand. It can be represented in the form of a symbol.

3. Opcode – short for operation code, the part of a machine code instruction that identifies the action the CPU will perform.

4. Operand – the part of a machine code instruction that identifies the data to be used by the CPU.

Advantages of Assembly Language

1.It allows complex jobs to run in a simpler way.

2.It is memory efficient, as it requires less memory.

3.It is faster in speed, as its execution time is less.

4.It is mainly hardware oriented.

5.It requires less instruction to get the result.

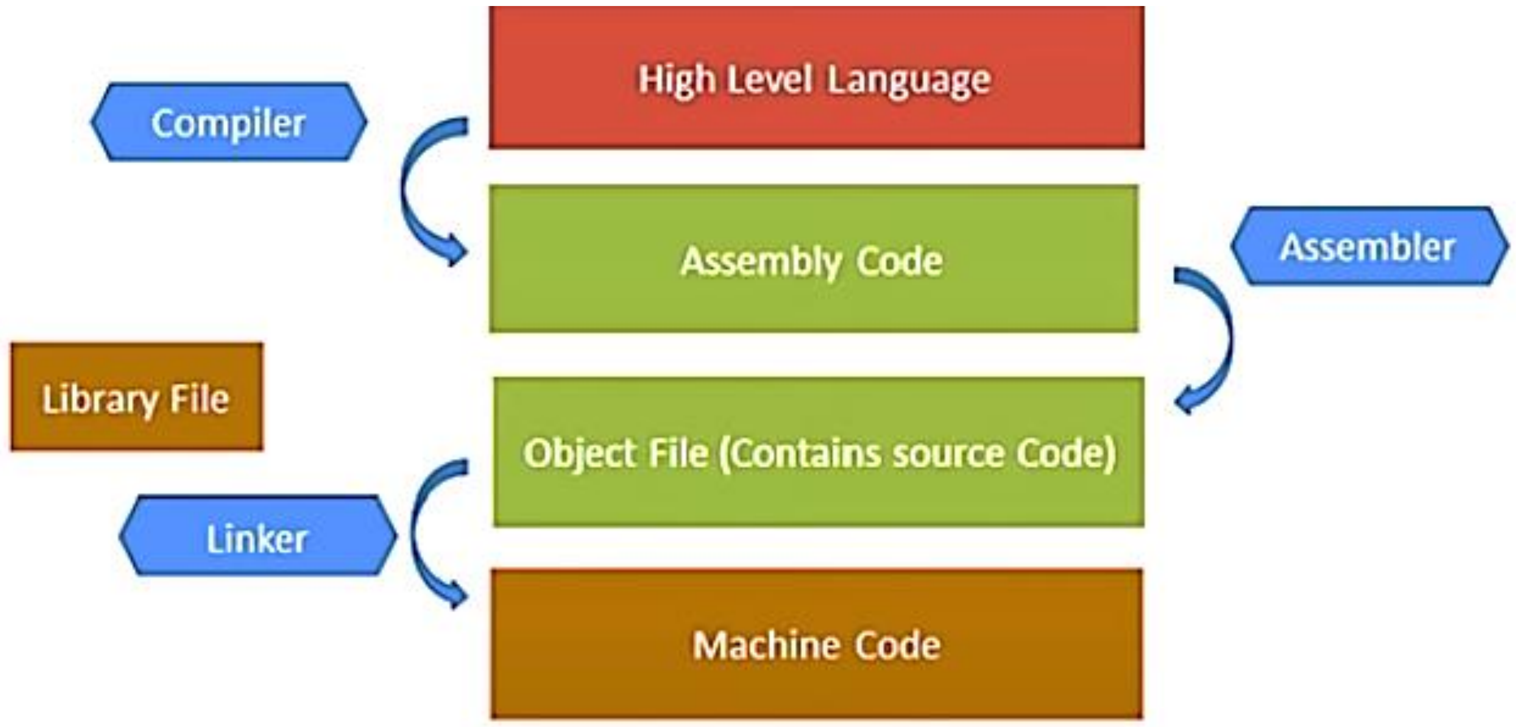
6.It is used for critical jobs.

7.It is not required to keep track of memory locations.

8.It is a low-level embedded system.

Disadvantages of Assembly Language

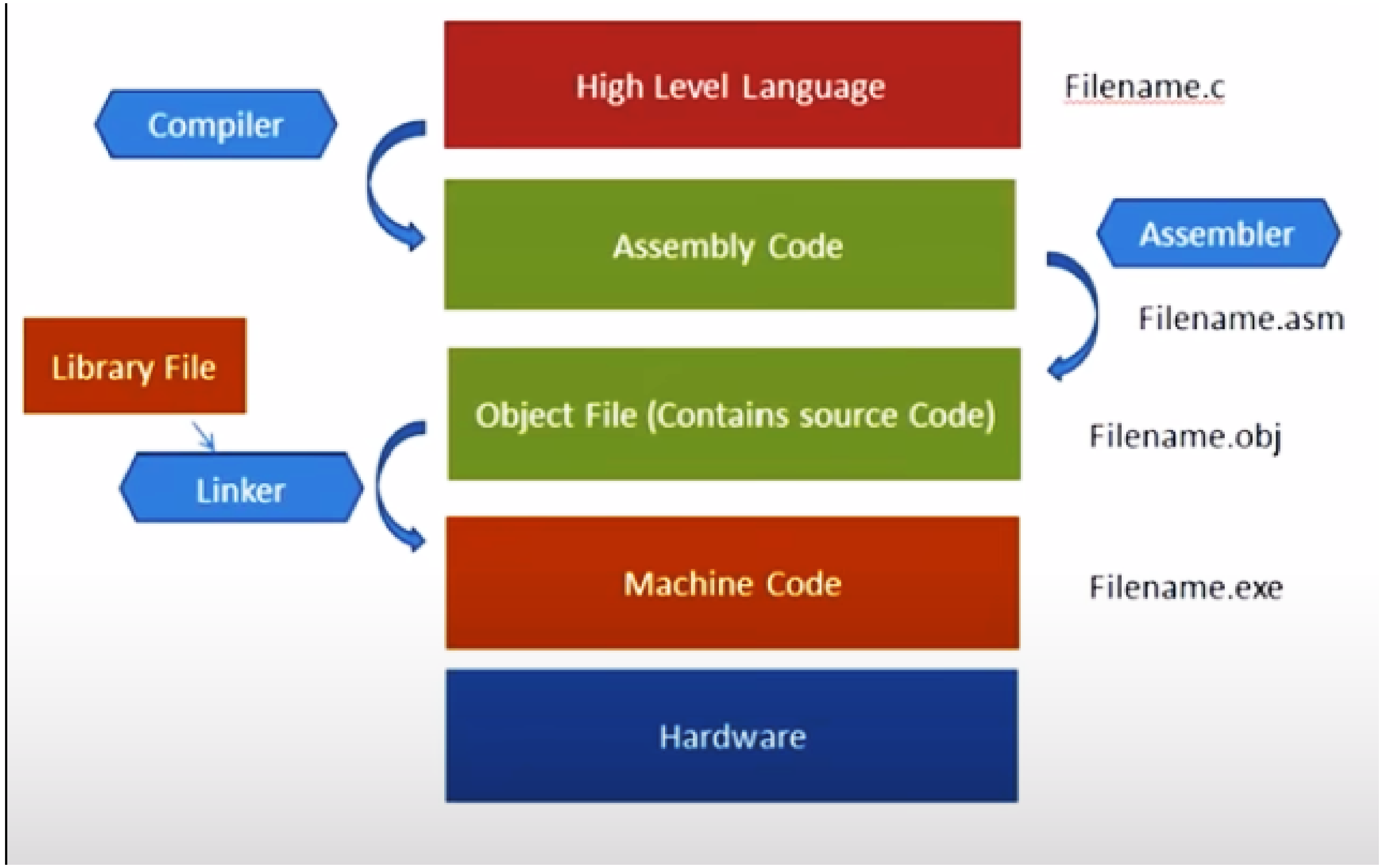
1. It takes a lot of time and effort to write the code for the same.
2. It is very complex and difficult to understand.
3. The syntax is difficult to remember.
4. It has a lack of portability of program between different computer architectures.
5. It needs more size or memory of the computer to run the long programs written in Assembly Language.



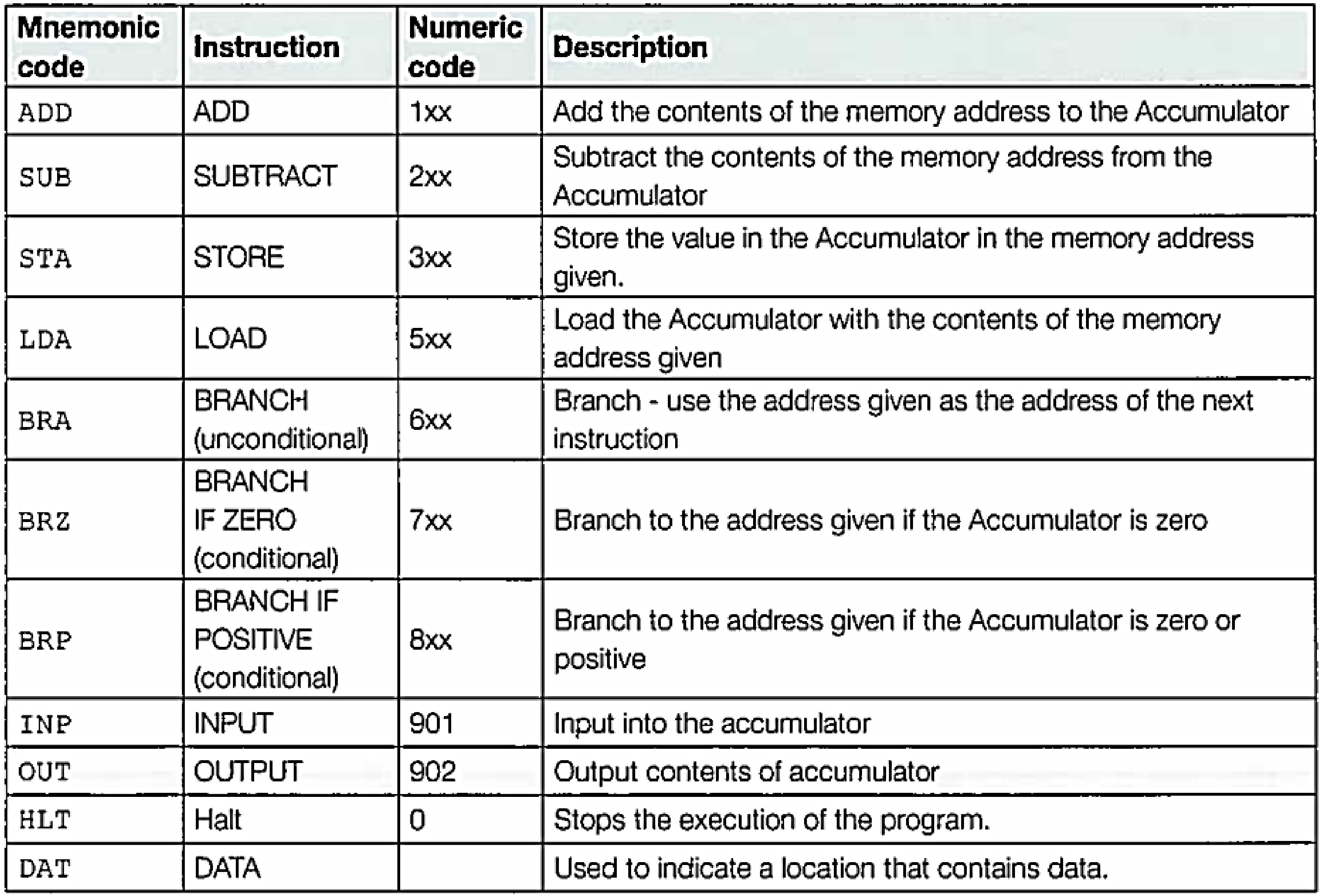
As you can see Assembly Language is closer to the Machine language so it can optimize the process of conversion instead of using High Level language. Thus less processing time required aswell.

Extension files

A Level CS With Mr Saem



Assembly Language Instructions



Mnemonic code	Instruction	Numeric code	Description
ADD	ADD	1xx	Add the contents of the memory address to the Accumulator
SUB	SUBTRACT	2xx	Subtract the contents of the memory address from the Accumulator
STA	STORE	3xx	Store the value in the Accumulator in the memory address given.
LDA	LOAD	5xx	Load the Accumulator with the contents of the memory address given
BRA	BRANCH (unconditional)	6xx	Branch - use the address given as the address of the next instruction
BRZ	BRANCH IF ZERO (conditional)	7xx	Branch to the address given if the Accumulator is zero
BRP	BRANCH IF POSITIVE (conditional)	8xx	Branch to the address given if the Accumulator is zero or positive
INP	INPUT	901	Input into the accumulator
OUT	OUTPUT	902	Output contents of accumulator
HLT	Halt	0	Stops the execution of the program.
DAT	DATA		Used to indicate a location that contains data.

Data transfer and arithmetic operations

Example 1

Input three numbers x, y and z. Calculate and output the value of x + y − z

```
          INP        ;Input y into accumulator ACC)
          STA y      ;Store the number in y
          INP        ;Input x into ACC
          STA z      ;Store the number in z
          INP        ;Input x into ACC
          ADD y      ;Add y to the number in ACC
          SUB z      ;subtract z from the number in ACC
          OUT        ;output the number in ACC
          HLT        ;halt
x         DAT
y         DAT
z         DAT
```

```
LDR    20
ADD    43
STR    20
SUB    41
STR    45
```

This code does the following:

- loads the accumulator with the contents of address 20
- adds the value held in memory location 43 to the value held in the accumulator
- stores the contents of the accumulator to address 20
- subtracts the value held in memory location 41 from the value held in the accumulator
- stores the contents of the accumulator to address 45.