## Introduction: What is Assembly Language and What is the Purpose of Using it?

Assembly language is a low-level programming language that is used to create instructions for a computer's processor. Assembly language can be translated into machine code or object code which can then be executed by the computer.

Assembly language is often used when the programmer wants to have direct control of the computer's processor and when they need to interact with hardware devices. It also makes the program more efficient and easier to understand for other programmers.

Assembly Language was developed in 1951 by John Mauchly and John Eckert, who were working on ENIAC at the time.

#### General Purpose Registers - scratch registers

* AX (AH, AL) Accumulator: Main arithmetic register
* BX (BH, BL) Base: Generally used as a memory base or offset
* CX (CH, CL) Counter: Generally used as a counter for loops
* DX (DH, DL) Data: General 16-bit storage, division remainder

#### Offset Registers

* IP Instruction pointer: Current instruction offset
* SP Stack pointer: Current stack offset
* BP Base pointer: Base for referencing values stored on stack
* SI Source index: General addressing, source offset in string ops
* DI Destination index: General addressing, destination in string ops

#### Segment Registers

* CS Code segment: Segment to which IP refers
* SS Stack segment: Segment to which SP refers
* DS Data segment: General addressing, usually for program's data area
* ES Extra segment: General addressing, destination segment in string ops

**Flags Register (Respectively bits 11,10,9,8,7,6,4,2,0)**

* OF Overflow flag: Indicates a signed arithmetic overflow occurred
* DF Direction flag: Controls incr. direction in string ops (0=ink, 1=dec)
* IF Interrupt flag: Controls whether interrupts are enabled
* TF Trap flag: Controls debug interrupt generation after instructions
* SF Sign flag: Indicates a negative result or comparison
* ZF Zero flag: Indicates a zero result or an equal comparison
* AF Auxiliary flag: Indicates adjustment is needed after BCD arithmetic
* PF Parity flag: Indicates an even number of 1 bit
* CF Carry flag: Indicates an arithmetic carry occurred4

## Project Theme: Flag of Pakistan

The flag of Pakistan was designed by Syed Amir-Uddin Kedwaii. The design is based on a classical Islamic art style, which incorporates calligraphy and Islamic geometric patterns.

The green represents Islam as well as hope for peace between different cultures and religions. The white symbolizes honesty and integrity while the red stands for bravery. The crescent moon represents progress while its star stands for light or knowledge.

### Symbolism

The Islamic green of the flag represents the Muslim-majority populace of Pakistan while the white stripe on the hoist-end represents its various religious minorities i.e., non-Muslims, such as Hindus, Christians, Sikhs, Zoroastrians, and others. The combined star and crescent serve as a symbol of Islam, with the crescent representing progress and the five-pointed star representing light and knowledge. The flag symbolizes Pakistan's commitment to both Islam as well as the rights of religious minorities.

In this project we compiled and executed the assembly language code x8086 to do the basic functions and control the pixels on screen. We simulated the program in emulator software in a microprocessor emulator with an integrated 8086 assembler. This is done by compiling 8086 Assembly code, assembling it, loading it on an emulator, executing it with input data to produce output data. The emulator can run programs on a Virtual Machine, and emulate real hardware including screen, memory, and input and output devices. It helps you program in assembly language. The source code is compiled by assembler and then executed on Emulator step-by-step, allowing you to watch registers, flags, and memory while your program runs.

#### EMULATOR SOFTWARE:

8086 Emulator is a program designed to run programs from the 8086 microprocessors. It is compatible with PC desktops and laptops and can compile programs from the 8086 assemblers. This tool is primarily designed to copy or emulate the hardware in a computer. This includes a program's memory, CPU, RAM, input, and output devices, and even the display screen.

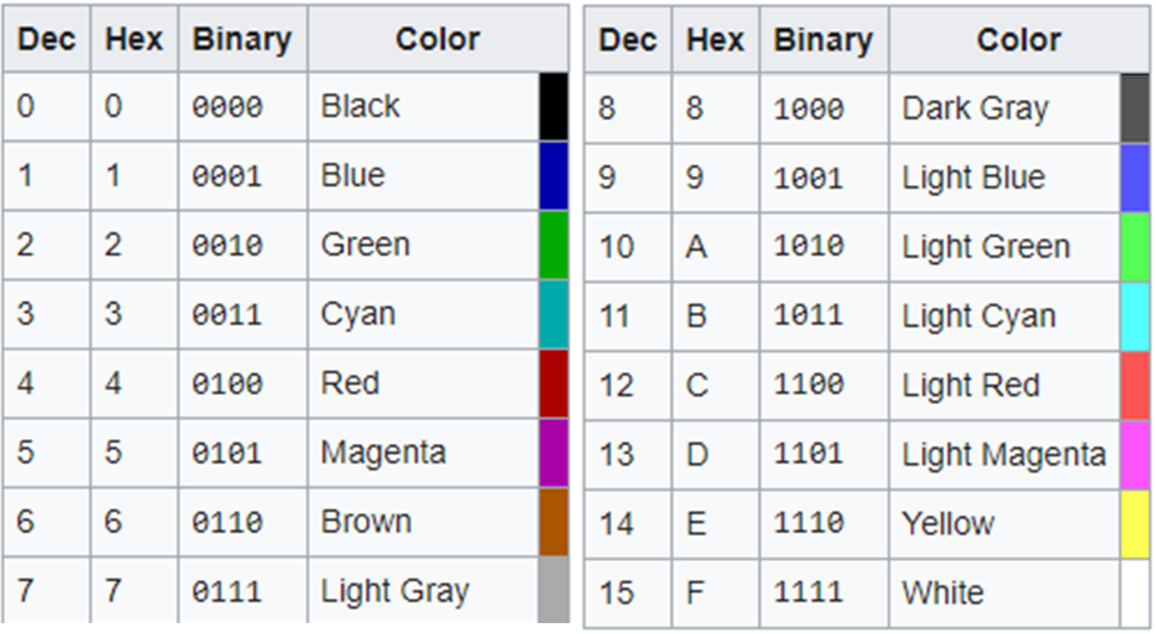
There are instructions to follow when it comes to this emulator. You can execute it in one of two ways: backwards or forwards. You have examples of assembly language source code as well as the ability to program certain things. It's a great way for a person who knows assembly language and wants a different perspective on the code.

The UI of an 8086 Microprocessor emulator is user-friendly and very easy to manage There are five main buttons with text and icons included which are "Load", "Reload", "Step Back", "Single Step" and "Run". Above the buttons are the menu options for tools like “File”, “View”, “Virtual Devices”, and more. Clicking on them will take you to different tasks they serve. There is also a helpful "Help" button in case you need assistance. Below the buttons is a series of numbers and codes. On the leftmost side there is a "Registers" area with either an “H” or “L” indicating if it is currently in pounds or pounds. The other side is divided into two, which enables users to manually reset, debug, flag.

## HOW CODE IS WORKING:

The code will start with a black screen which can be access with giving some values in AX register and using the interrupt command to execute instruction in AX. I have used AH=00 and AL=12h so I can access video mode with resolution of 700x400.

To Turn ON/OFF Pixel on screen we use Instruction AH=0CH and giving some color by placing value in AL Register. Some Color code are below.

****

when we are in video mode the column and row are the main key-roll to work on screen and the value of these two is control by CX and DX Registers. The value of Column is store in CX and Value of ROWs store on DX register. we can also set manually value in these registers.

Our main goal is to print Rectangle with fill of Green and a Rectangle with White color on screen. as we are now working in pixel/ video mode, so we must turn every pixel in that space Turn on to get these done. I have built a flowchart which showing a complete guide how it can be done easily using nested loop in my code. for controlling the loop counter I’m using the BX Roister as CX is using for the value of Column. We can use INC/DEC Command to change the value of any register or variable by One. and Using Labels to tell a loop to jump on a specific part of Code on a condition.

### FLOWCHART:

|  |
| --- |
| A screenshot of a computer  Description automatically generated with low confidence |
| Diagram  Description automatically generated |
| Diagram  Description automatically generated |

### CODE:

|  |  |
| --- | --- |
| A picture containing text  Description automatically generated | A picture containing text  Description automatically generated |
| A picture containing graphical user interface  Description automatically generated | A picture containing table  Description automatically generated |
| A picture containing table  Description automatically generated | A picture containing chart  Description automatically generated |

## Output:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

### CONCLUSION:

In this project, I was able to use the assembly language to program a complete front end of a program. I completed my project by using Assembly language to program a computer to produce a desired outcome. I successfully computed and printed the flag of Pakistan in pixel mode in x86. also, I successfully calculated the pixel row and column to set and print desired output. Assembly language is not a simple retail language, but computer engineering professionals should know it for the sake of knowledge. assembly language is a programming language at the lowest level. It allows you to work at the byte-level of binary code. The main thing about it is that it enables you to understand what's happening on the CPU, which is where most computers are happening.