ALU 8086

Submitted to : **Mr. Kundan Kumar**

Team members :

→ **Prince Ranjan (21104074)**

→ **Navjot Singh Kardam (21104063)**

ALU 8086: An Introduction

The ALU, or Arithmetic Logic Unit, is the key component of the 8086 microprocessor that performs arithmetic and logical operations. It is responsible for executing instructions and manipulating data to carry out the desired computations in a computer system.

Features and Functionalities of ALU

1 Arithmetic Operations

The ALU can perform basic arithmetic operations such as addition, subtraction, multiplication, and division.

2 Logical Operations

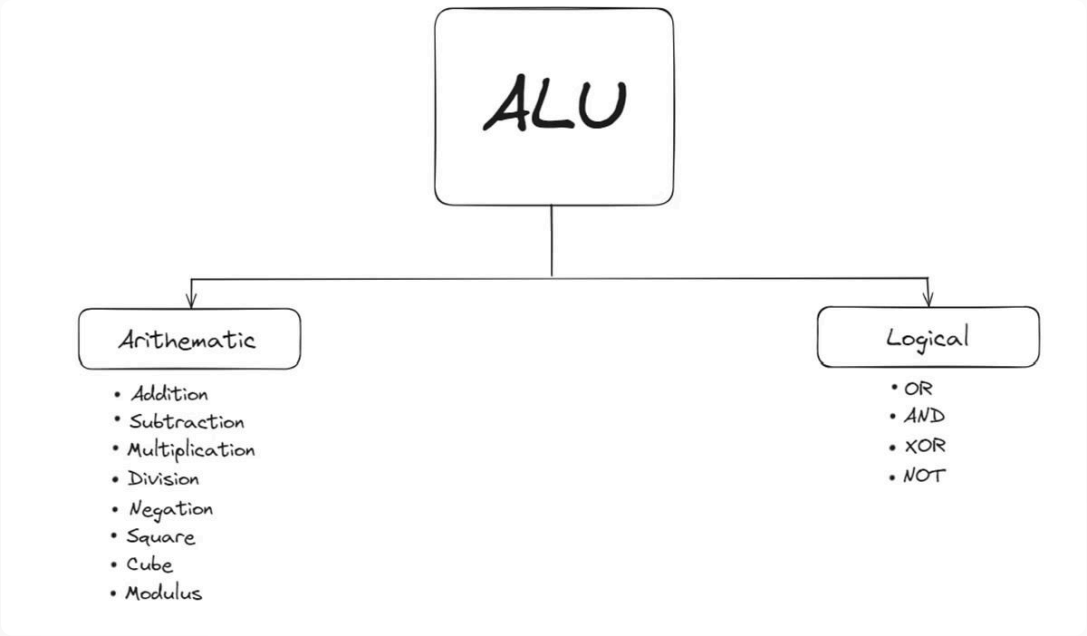
It can also execute logical operations like AND, OR, NOT, and XOR on binary data.

3 Comparison Operations

The ALU can compare values and set flags based on the results, enabling conditional branching in programs.

4 Data Manipulation

The ALU can perform shift and rotate operations on data, allowing for efficient bit-level manipulations.



1 Arithmetic Operations

The ALU performs various arithmetic operations like addition, subtraction, multiplication, and division.

2 Logical Operations

The ALU also handles logical operations such as AND, OR, and NOT.

ALU using Emu8086 Software

Emu8086 Simulator

The Emu8086 software is a powerful tool for simulating and understanding the inner workings of the 8086 microprocessor, including its ALU.

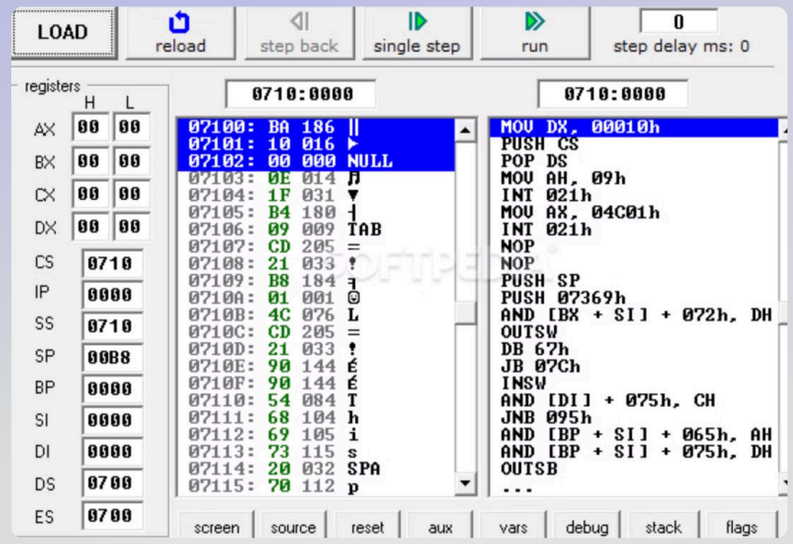
Hands-on Experience

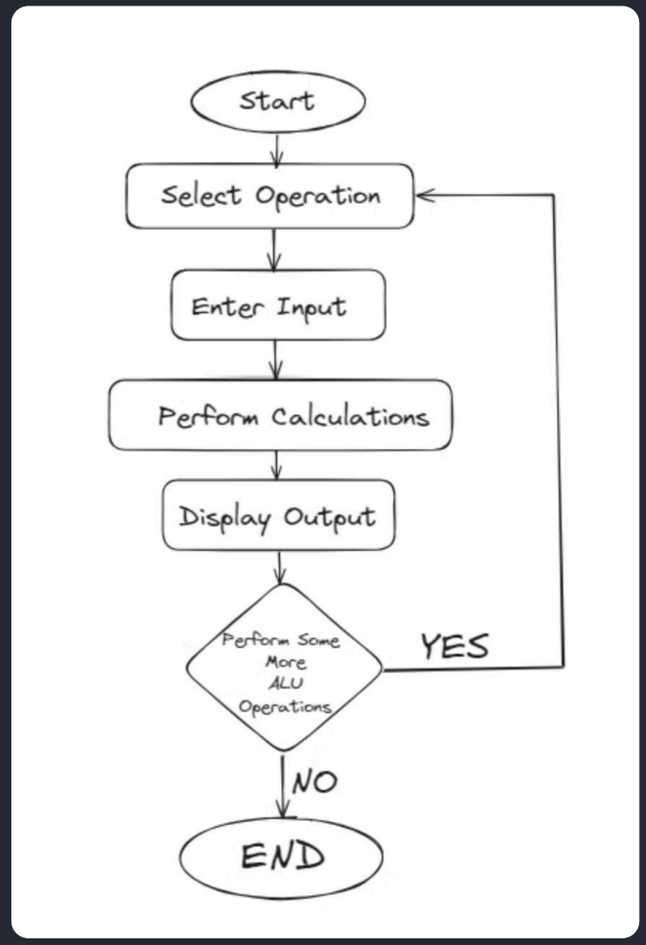
By using Emu8086, developers can write and test 8086 assembly language programs, gaining practical experience with the ALU's functionality.

Debugging and Troubleshooting

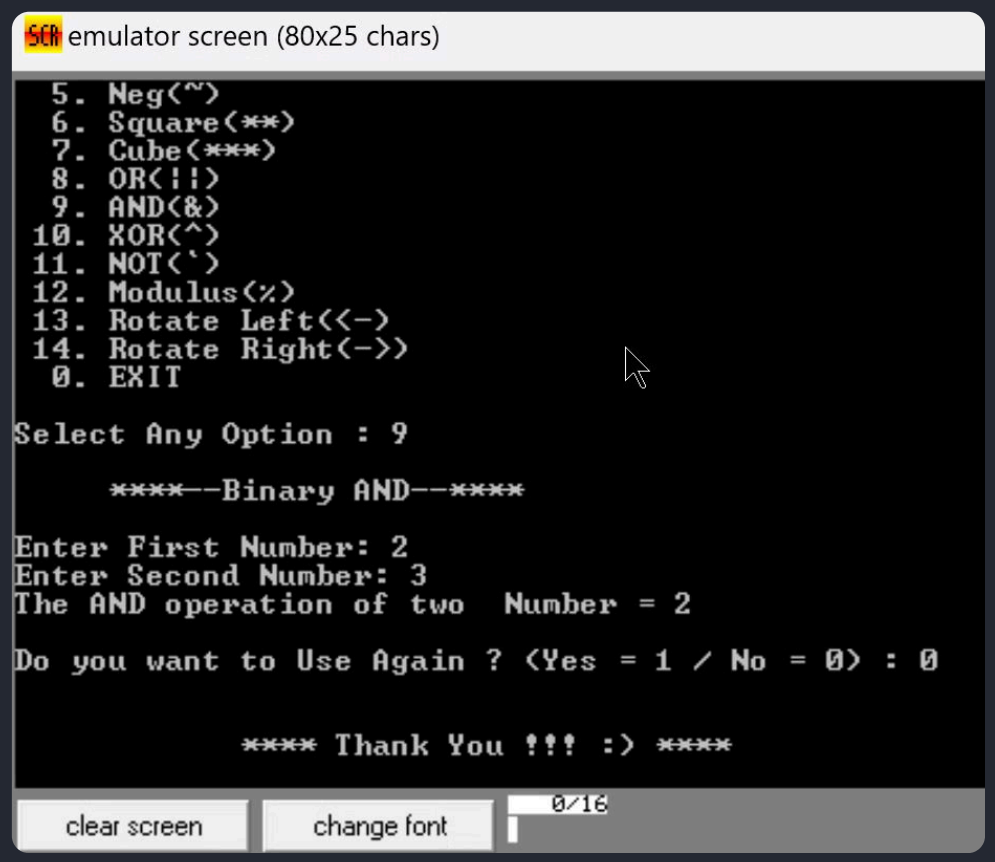
The software provides debugging capabilities to help identify and resolve issues in ALU-related code, making it a valuable tool for learning and development.

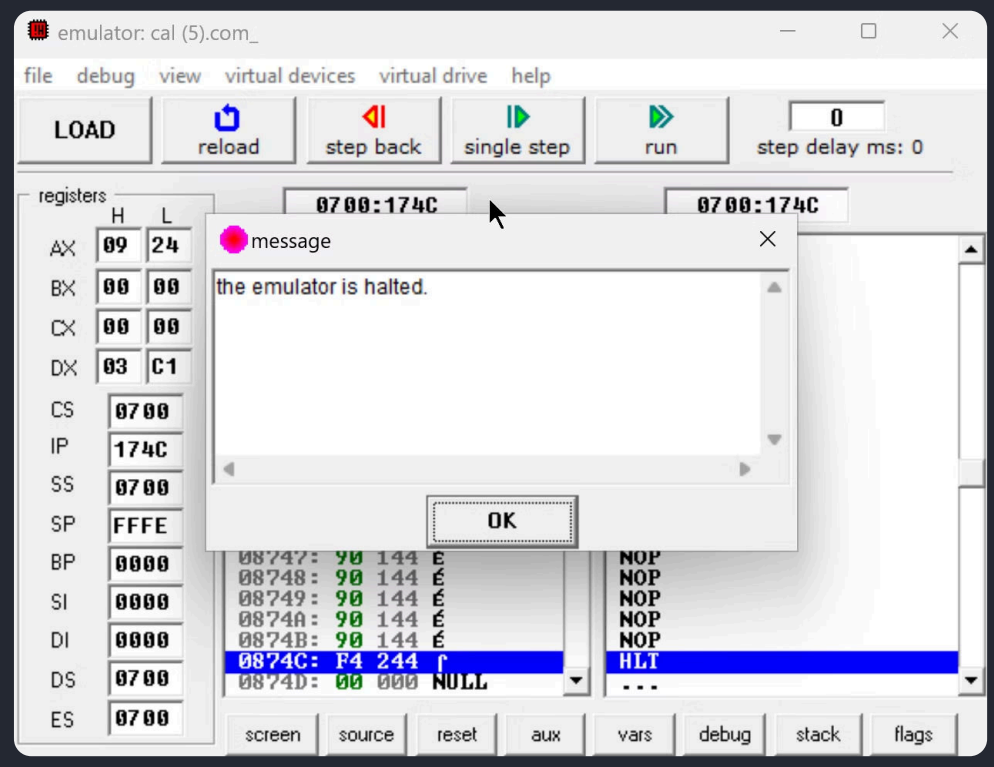
Flow Chart : Visualizing the ALU

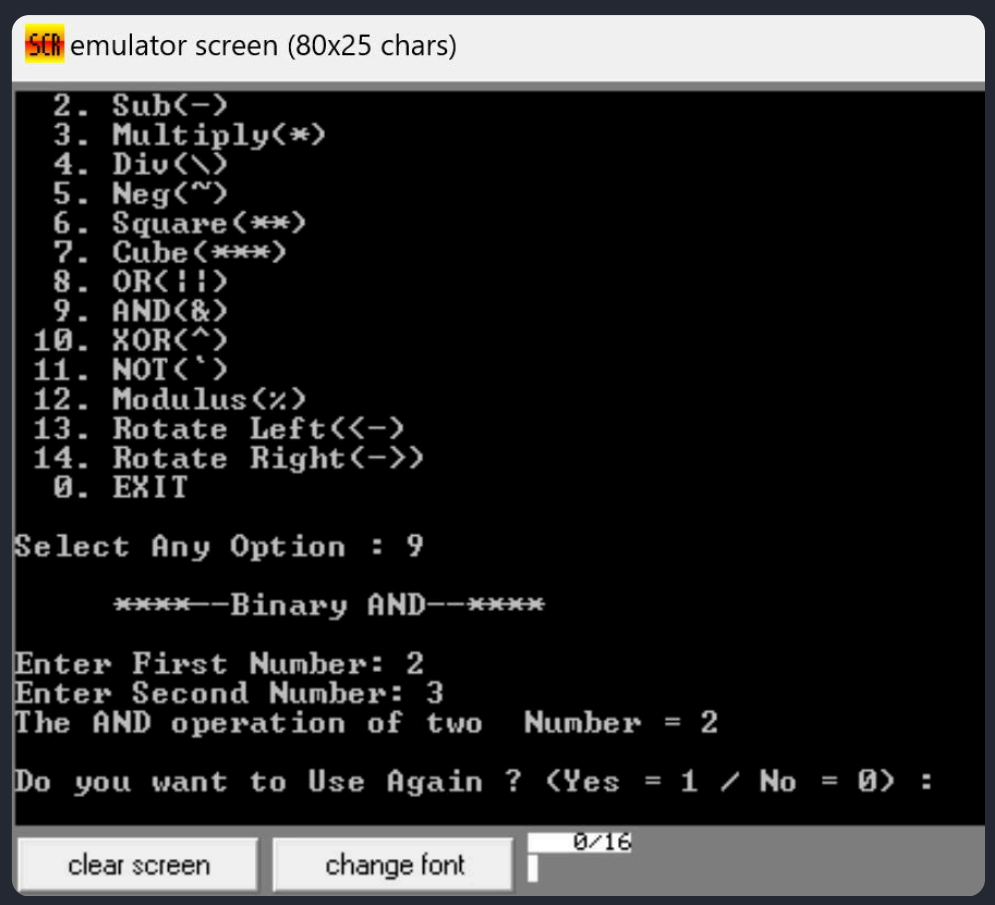


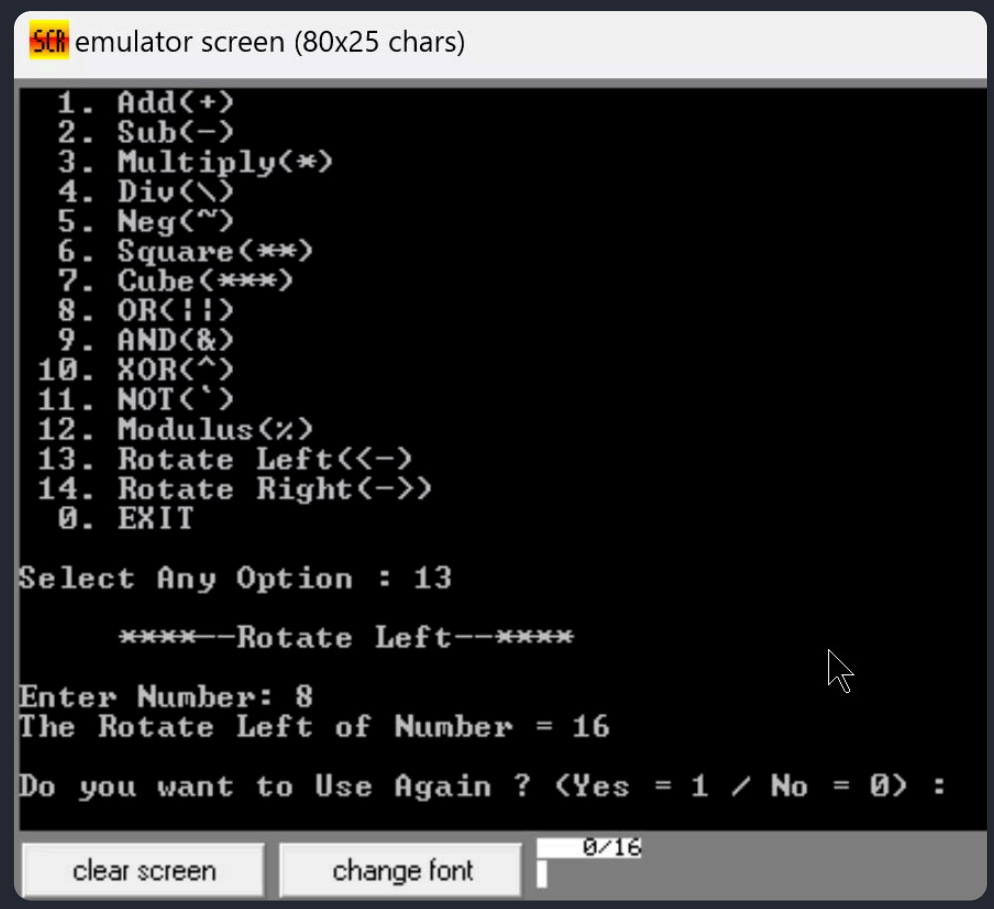


Outputs :









emulator screen (79x29 chars)

```
  2. Sub(-)
  3. Multiply(*)
  4. Div(\)
  5. Neg(~)
  6. Square(**)
  7. Cube(***)
  8. OR(|)
  9. AND(&)
 10. XOR(^)
 11. NOT(`)
 12. Modulus(%)
 13. Rotate Left(<-)
 14. Rotate Right(->)
  0. EXIT

Select Any Option : 1

     ****--Addition--****

Enter First Number: 2
Enter Second Number: 3
The SUM of two Numbers = 5

Do you want to Use Again ? (Yes = 1 / No = 0) : _
```

clear screen | change font | 0/16

emulator screen (95x37 chars)

```
                **** CALCULATOR ****

  1. Add(+)
  2. Sub(-)
  3. Multiply(*)
  4. Div(\)
  5. Neg(~)
  6. Square(**)
  7. Cube(***)
  8. OR(|)
  9. AND(&)
 10. XOR(^)
 11. NOT(`)
 12. Modulus(%)
 13. Rotate Left(<-)
 14. Rotate Right(->)
  0. EXIT

Select Any Option :
```

emulator: cal (5).com_

file debug view virtual devices virtual drive help

LOAD | reload | step back | single step | run | step delay ms: 0

registers

	H	L
AX	00	00
BX	00	00
CX	16	38
DX	00	00
CS	0700	
IP	0100	
SS	0700	
SP	FFFE	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

0700:0100 | 0700:0100

```
07100: E9 233 Θ   JMP 003F1h
07101: EE 238 ∈   AND [BX + SI], AH
07102: 02 002 ☺   XOR [04120h], BP
07103: 20 032 SPA DB 64h
07104: 20 032 SPA DB 64h
07105: 31 049 1   SUB [BP + DI], CH
07106: 2E 046 .   SUB [DI], CX
07107: 20 032 SPA OR AH, [BX + SI]
07108: 41 065 A   AND [BP + SI], DH
07109: 64 100 d   CS:
0710A: 64 100 d   AND [BP + DI] + 075h, DL
0710B: 28 040 (   DB 62h
0710C: 2B 043 +   SUB [DI], CH
0710D: 29 041 )   SUB [DI], CX
0710E: 0D 013 CRET OR AH, [BX + SI]
0710F: 0A 010 NEWL AND [BP + DI], DH
07110: 20 032 SPA CS:
07111: 20 032 SPA AND [DI] + 075h, CL
07112: 32 050 2   INSB
07113: 2E 046 .   JZ 0190h
07114: 20 032 SPA JO 0195h
07115: 53 083 S   ...
```

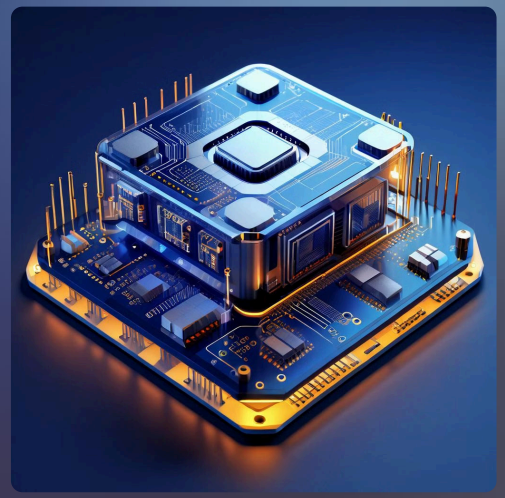
screen | source | reset | aux | vars | debug | stack | flags

edit: C:\Users\Prince\Downloads\cal (5).asm

file edit bookmarks assembler help

NEW | OPEN | SAVE | ASSEMBLE | RUN

```
001
002 ; You may customize this and other start-up templates;
003 ; The location of this template is c:\emu8086\inc\0_com_template.txt
004
005 INCLUDE EMU8086.INC     ;Including Library
006
007 org 100h
008
009 .data        ;Data Segment
010
011 msg_intro db '  1. Add(+)', 0dh,0ah, '  2. Sub(-)', 0dh,0ah, '  3. Mult
012
013 msg_A db 'The SUM of two Numbers = $', 0dh,0ah        ;Printing Strings
014 msg_S db 'The SUBTRACTION of two Numbers = $', 0dh,0ah
015 msg_M db 'The MULTIPLICATION of two Numbers = $', 0dh,0ah
016 msg_D db 'The DIVISION of two Numbers = $', 0dh,0ah
017 msg_N db 'The NEG value of Number = $', 0dh,0ah
018 msg_SQ db 'The SQUARE of Number = $', 0dh,0ah
019 msg_CB db 'The CUBE of Number = $', 0dh,0ah
020 msg_OR db 'The OR operation of two Number = $', 0dh,0ah
021 msg_AND db 'The AND operation of two  Number = $', 0dh,0ah
022 msg_XOR db 'The XOR operation of two Number = $', 0dh,0ah
023 msg_NOT db 'The NOT of Number = $', 0dh,0ah
024 msg_MD db 'The MODULUS of Two Numbers = $', 0dh,0ah
025 msg_RL db 'The Rotate Left of Number = $', 0dh,0ah
026 msg_RR db 'The Rotate Right of Number = $', 0dh,0ah
027
028 cont db 10,13,'Do you want to Use Again ? $'
029 bye db '              **** Thank You !!!  :) **** $'
030
031 val1 dw ?        ;Uninitialize
032 val2 dw ?        ;Uninitialize
033 res dw ?
034 agn dw ?
035
036 .code        ;Code Segment
037 MAIN PROC
038
```



Conclusion and Takeaways

Crucial Component

The ALU is a fundamental building block of the 8086 microprocessor, responsible for executing the core computational tasks.

Versatile Functionality

The ALU supports a wide range of arithmetic, logical, and comparison operations, enabling diverse and complex computations.

Understanding the ALU

Mastering the concepts and functionality of the ALU is crucial for understanding the inner workings of the 8086 processor and modern computer architecture.

THANK YOU !!!