

**NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY**

School of Electrical Engineering and Computer Sciences

**Computer Organization & Assembly Language**

**ASSIGNMENT NO. 1,2,3**

**Title: Creating an 8086 Simulator**

**SUBMITTED TO: Mr. Imran Abeel**

**SUBMITTED BY: Taha Ahmed, Qazi Shujaullah, Prithvi Raj, Usman, Taha Ahmad**

**CLASS: BSCS-11**

**SECTION: A**

**Due Date: 4-12-2022**

# 8086 SIMULATOR

# LANGUAGES AND PLATFORMS USED:

1. **C++:** For back-end and linking functions
2. **NET:** For GUI development

# REGISTERS USED

|  |  |  |  |
| --- | --- | --- | --- |
| **AX** | **BX** | **CX** | **DX** |
| **SP** | **SI** | **DI** | **BP** |

**MEMORY LOCATIONS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **[00000]** | **[00001]** | **[00002]** | **[00003]** |
| **[00004]** | **[00005]** | **[00006]** | **[00007]** |
| **[00008]** | **[00009]** | **[0000A]** | **[0000B]** |
| **[0000C]** | **[0000D]** | **[0000E]** | **[0000F]** |

**INSTRUCTION SET:**

This instruction set consists of following 15 instruction and their different modes.

|  |  |
| --- | --- |
| **Operations:** | **Opcode:** |
| **MOV** | 000000 |
| **ADD** | 000001 |
| **ROL** | 000010 |
| **INC** | 000011 |
| **ROR** | 000100 |
| **NOP** | 000101 |
| **AND** | 000110 |
| **OR** | 000111 |
| **XOR** | 001000 |
| **NOT** | 001001 |
| **SHR** | 001010 |
| **SHL** | 001011 |
| **HLT** | 001100 |
| **NEG** | 001101 |
| **CLC** | 001110 |

**INSTRUCTIONS IMPLEMENTED**

1. **MOV**
   1. **Validity Checks:**
      * + - Value being transferred is in destination’s limit
          - Registers being invoked are included in this Simulator
          - Storage in Little Endian format
          - Handle hexadecimal Immediate
   2. **Variations**
      * Mov Reg, Reg
      * Mov Reg, Immediate
      * Mov Reg, [XXXXX]
      * Mov [Reg], Reg
      * Mov [Reg +8bit disp], Reg
      * Mov [Reg +16bit disp], Reg
2. **ADD**
   1. **Validity Checks:**
      * Size of registers involved is comparable
   2. **Variation:**
      * Add Reg, Reg
      * Add Reg, Immediate
      * Add Reg, [XXXXX]
      * Add [Reg], Reg
      * Add [Reg +8bit disp], Reg
      * Add [Reg +16bit disp], Reg

1. **INC**
   1. **Validation Checks:**
      * Cannot increment if value stored in the register is the maximum value a register can hold
   2. **Variation:**
      * Inc Reg
      * Inc [XXXXX]
      * Inc [Reg]
      * Inc [Reg +8bit disp]
      * Inc [Reg +16bit disp]
2. **NEG**
   1. **Validation Checks:**
      * Destination register is present in our map
   2. **Variation:**
      * Neg Reg
      * Neg [XXXXX]
      * Neg [Reg]
      * Neg [Reg +8bit disp]
      * Neg [Reg +16bit disp]
3. **AND**
   1. **Validation Checks:**
      * Value being transferred is in destination’s limit
      * AND between same sized registers
      * Registers being invoked are included in this particular Simulator
   2. **Logic Used For Implementation:**

Using a FOR LOOP, same position bits - string indexing - are compared, if both are 1, at the corresponding index 1 is stored elsewise 0 is stored.

* 1. **Variation:**
     + And Reg, Reg
     + And Reg, Immediate
     + And Reg, [XXXXX]
     + And [Reg], Reg
     + And [Reg +8bit disp], Reg
     + And [Reg +16bit disp], Reg

1. **OR**
   1. **Validation Checks:**
      * Value being transferred is in destination’s limit
      * OR between same sized registers
      * Registers being invoked are included in this particular Simulator
      * Size of value stored is equivalent to destination size (append zeros to fill remaing)
   2. **Logic Used For Implementation**

Using a FOR LOOP, same position bits - string indexing - are compared, if any is 1, at the corresponding index 1 is stored elsewise 0 is stored.

* 1. **Variation:**
     + Or Reg, Reg
     + Or Reg, Immediate
     + Or Reg, [XXXXX]
     + Or [Reg], Reg
     + Or [Reg +8bit disp], Reg
     + Or [Reg +16bit disp], Reg

1. **NOT**
   1. **Validation Checks:**
      * Caters to invert MSB zeros
      * Value being transferred is in destination’s limit
   2. **Logic Used For Implementation**

Using a FOR LOOP, store inverted value of at corresponding indexes.

* 1. **Variation:**
     + Not Reg
     + Not [XXXXX]
     + Not [Reg]
     + Not [Reg +8bit disp]
     + Not [Reg +16bit disp]

1. **XOR** 
   1. **Validation Checks:**
      * Value being transferred is in destination’s limit
      * OR between same sized registers
      * Registers being invoked are included in this particular Simulator
   2. **Logic Used For Implementation**

Using a FOR LOOP, same position bits - string indexing - are compared, if both are not equal, store 1 in corresponding index, else store 0.

* 1. **Variation:**
     + Xor Reg, Reg
     + Xor Reg, Immediate
     + Xor Reg, [XXXXX]
     + Xor [Reg], Reg
     + Xor [Reg +8bit disp], Reg
     + Xor [Reg +16bit disp], Reg

1. **SHL**

* 1. **Validation Checks:**
     + Shift should not be greater than register size
     + Set size by appending zeros by appending zeros to MSB
  2. **Variation:**
* SHL Reg, Immediate
* SHL Mem, Immediate
* SHL Reg, CL
* SH Mem, CL

1. **SHR**
   1. **Validation Checks:**
      * Shift should not be greater than register size
      * Set size by appending zeros by appending zeros to MSB
   2. **Variation:**

* SHR Reg, Immediate
* SHR Mem, Immediate
* SHR Reg, CL
* SHR Mem, CL

1. **NOP**

It is an instruction with no operand that does nothing.

1. **HLT**

It is an instruction with no operand that interrupt the signal.

1. **ROL**

**Variation:**

* ROL Reg, Immediate
* ROL Mem, Immediate
* ROL Reg, CL
* ROL Mem, CL

1. **ROR**

**Variation:**

* ROL Reg, Immediate
* ROL Mem, Immediate
* ROL Reg, CL
* ROL Mem, CL

1. **CLC**

Sets carry flag to zero.