8085 Emulator C++

-Shantanu Chaudhari

Approach

The project is a 8085 emulator coded in C++. It mimics the working of various 8085 instructions listed below. The emulator runs assembly language instructions and produce outputs similar to the one obtained while working on the individual microprocessor. You can dump the memory to terminal by providing “-m” flag after the filename.

The following instructions are Implemented in the emulator:

- ADD
- ADI
- ANA
- CALL
- CC
- CM
- CMA
- CMC
- CMP
- CNC
- CNZ
- CP
- CPE
- CPI
- CPO
- CZ
- DAD
- DCR
- POP
- PUSH
- RC
- RET
- RM
- RNC
- RNZ
- RP
- RPE
- RPO
- RZ
- DCX
- INR
- INX
- JC
- JM
- JMP
- JNC
- JNZ
- JP
- JPE
- JPO
- JZ
- LDA
- LHLD
- LXI
- MOV
- MVI
- ORA
- ORI
- SUB
- SUI
- XCHG
- XRA
- XRI
- SHLD
- STA
- STAX
- STC
- SET

The initially checks all the instruction and the number of arguments for the instructions, if any incorrect instruction is found or the arguments don’t match, the program will exit. Then all the instructions are stored in the memory (Map), and program executes by calling the respective function for the opcode. The emulator doesn’t support tagging the instructions i.e. “LOOP: ADD A” is not valid, instead, you’ll have to provide the specific address of the instruction i.e. 1024 when the starting address is set to 1000. The emulator also supports stack

pointer, it is initially set to FFFF, but can be changed as per requirement. PSW isn't supported.

Subroutines and Recursion is fully supported with all the required instructions. HLT must be included in the program to stop, or else the program will continue endlessly.

Each instruction is implemented in a separate file, the individual implementation can be viewed from there.

Testing

The emulator was tested on various sample codes, the code and memory output can be found below

1. Swapping values of 2 memory locations:

```
1  SET 2100,05
2  SET 2101,07
3  LDA 2100
4  MOV B,A
5  LDA 2101
6  STA 2100
7  MOV A,B
8  STA 2101
9  HLT
```

```
Enter the starting address:
1000
Memory Dump Starts

ADDR Instruction
------------------------
1000 SET 2100,05
1003 SET 2101,07
1006 LDA 2100
1009 MOV B,A
100A LDA 2101
100D STA 2100
1010 MOV A,B
1011 STA 2101
1014 HLT
2100 07
2101 05
```

2. Multiplying 2 numbers

```
1  MVI A,00
2  MVI B,17
3  MVI C,09
4  ADD B
5  DCR C
6  JZ 100E
7  JMP 1006
8  STA 2100
9  MOV A,B
10 STA 2101
11 HLT
```

```
Enter the starting address:
1000
Memory Dump Starts

ADDR Instruction
------------------------
1000 MVI A,00
1002 MVI B,17
1004 MVI C,09
1006 ADD B
1007 DCR C
1008 JZ 100E
100B JMP 1006
100E STA 2100
1011 MOV A,B
1012 STA 2101
1015 HLT
2100 CF
2101 17
```

3. Find a factorial of a number

```
1  SET 2010,05
2  LXI H,2010
3  MOV B,M
4  MVI A,00
5  MOV D,B
6  DCR B
7  JZ 101B
8  MOV E,B
9  ADD D
10 DCR E
11 JNZ 100F
12 MOV D,A
13 MVI A,00
14 DCR B
15 JMP 100B
16 MOV A,D
17 STA 2011
18 HLT
```

```
Enter the starting address:
1000
Memory Dump Starts

ADDR Instruction
------------------------
1000 SET 2010,05
1003 LXI H,2010
1006 MOV B,M
1007 MVI A,00
1009 MOV D,B
100A DCR B
100B JZ 101B
100E MOV E,B
100F ADD D
1010 DCR E
1011 JNZ 100F
1014 MOV D,A
1015 MVI A,00
1017 DCR B
1018 JMP 100B
101B MOV A,D
101C STA 2011
101F HLT
2010 05
2011 78
```

4. Testing Subroutines

```
1  LXI B,AABB
2  LXI SP,4000
3  CALL 100F
4  MVI A,01
5  STA 2000
6  HLT
7  MVI A,02
8  STA 2001
9  RET
```

```
Enter the starting address:
1000
Memory Dump Starts

ADDR Instruction
------------------------
1000 LXI B,AABB
1003 LXI SP,4000
1006 CALL 100F
1009 MVI A,01
100B STA 2000
100E HLT
100F MVI A,02
1011 STA 2001
1014 RET
2000 01
2001 02
3FFE 09
3FFF 10
```