

Manisa Celal Bayar University

Comp. Eng. Depart. CSE3134
Microprocessor and Embedded Systems

Simulation with 8085 Microprocessor

Project: The program that will read customer numbers from memory as HEX, find the highest number and write decimally on the screen.

Mehmet ÇETİN - 160315001

Mert DUMANLI - 160315002

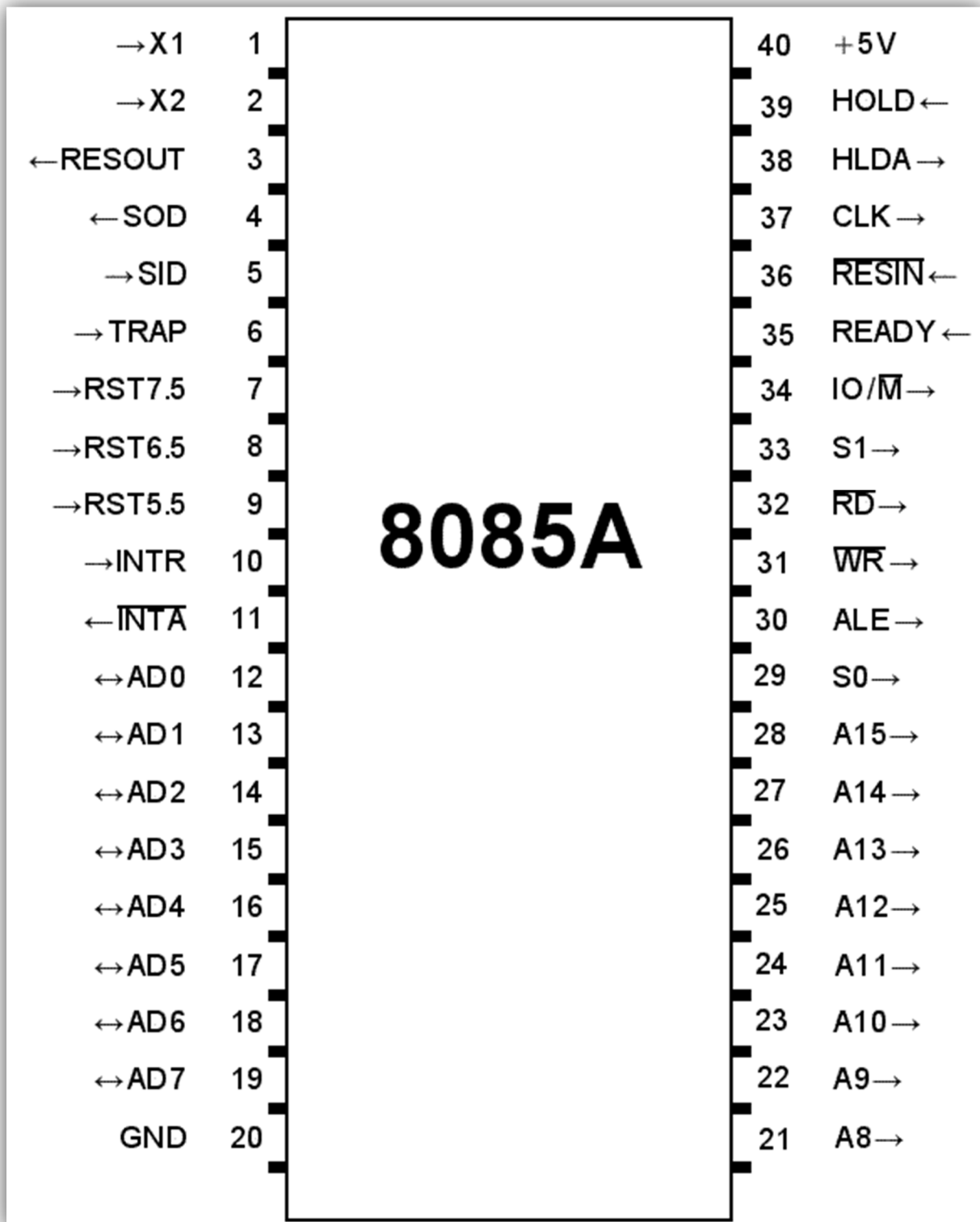
Süleyman DOĞAN - 160315003

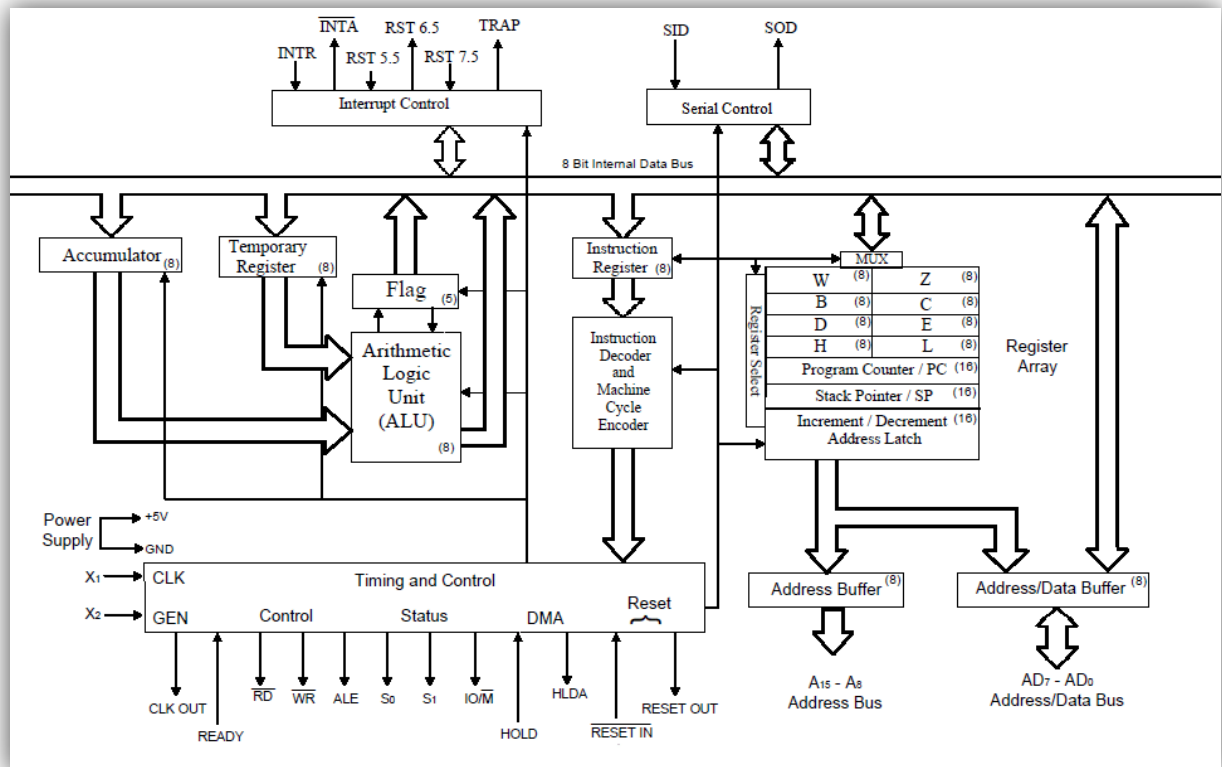
Ulaş Deniz İLHAN – 150316015

Table of Contents

What is 8085 Microprocessor?	3
Explanation of Codes.....	5
Testing.....	10
References	11

What is 8085 Microprocessor?





The Intel 8085 ("eighty-eighty-five") is an 8-bit microprocessor produced by Intel and introduced in March 1976. It is a software-binary compatible with the more-famous Intel 8080 with only two minor instructions added to support its added interrupt and serial input/output features. However, it requires less support circuitry, allowing simpler and less expensive microcomputer systems to be built.

The "5" in the part number highlighted the fact that the 8085 uses a single +5-volt (V) power supply by using depletion-mode transistors, rather than requiring the +5 V, -5 V and +12 V supplies needed by the 8080. This capability matched that of the competing Z80, a popular 8080-derived CPU introduced the year before. These processors could be used in computers running the CP/M operating system.

The 8085 is supplied in a 40-pin DIP package. To maximise the functions on the available pins, the 8085 uses a multiplexed address/data ($\text{AD}^0\text{-AD}^7$) bus. However, an 8085 circuit requires an 8-bit address latch, so Intel manufactured several support chips with an address latch built in. These include the 8755, with an address latch, 2 KB of EPROM and 16 I/O pins, and the 8155 with 256 bytes of RAM, 22 I/O pins and a 14-bit programmable timer/counter. The multiplexed address/data bus reduced the number of PCB tracks between the 8085 and such memory and I/O chips.

Both the 8080 and the 8085 were eclipsed by the Zilog Z80 for desktop computers, which took over most of the CP/M computer market, as well as a share of the booming home-computer market in the early-to-mid-1980s.

The 8085 had a long life as a controller, no doubt thanks to its built-in serial I/O and 5 prioritized interrupts, arguably microcontroller-like features that the Z80 CPU did not have. Once designed into such products as the DECtape II controller and the VT102 video terminal in the late 1970s, the 8085 served for new production throughout the lifetime of those products. This was typically longer than the product life of desktop computers.

Explanation of Codes:

We used the Sim8085 site to run this program. Site link: <https://www.sim8085.com>

```
1 ;0-7F
2 ;80-FF
3
4 MVI A, 0
5 MVI E, 0
6 ; E: If we find a number greater than 7F,
7 ; we increase E by 1. In this way,
8 ; we will not control values less than 80.
9 STA 0000H ; We reset at the beginning.
10 LXI H, 0001H
11 MVI D, 80H ; FF-80 = 0111 1111
12 MOV A, M
```

First, we reset the register where we keep the maximum number and the E register. We will use the E register when numbers larger than 7F (HEX) will be processed. We have referenced the address 0001H (HEX) to the M (HL register pair) registers. Each time the loop is back, we increase the HL register and access the written data in memory. The PRINT2 method is running automatically after the start codes run. We consider the first number to arrive to be the largest number for the moment.

```

18 PRINT2: MVI B, 0
19 ADD B ; Or SUB B
20 JZ END ; If the value entered is 0, it ends.
21 SUB D ; A = A - D
22 JP BIGTHAN7F ; If the sign bit is not 1, it continues
23 ADD D ; If the number is less than 80.
24 MOV C, A
25 MOV A, E
26 ADD B ; Or SUB B
27 JNZ PRINT ; If not 0, go.
28 ; If a number greater than 7F has been previously
29 ; processed, subsequent numbers
30 ; less than 80 are not processed.
31 LDA 0000H
32 SUB C ; A = A - C
33 JP PRINT ; If the sign bit is not 1.
34 MOV A, M
35 STA 0000H
36 JMP PRINT

```

Here we first check whether the number equals 0. Because the end condition of our program is the number to be read is 0. If the number is 0, the program ends. We then check if the number is greater than 80 (HEX) or smaller. The purpose of making this check:

If our result is a value greater than 80 (HEX) or greater than 80 (HEX), the sign bit flag becomes 1, so our results are erroneous. For example; While it is expected to be 84H - 3H = 81 (HEX), the result gives us the number of 1000 0001 (BINARY) which corresponds to 81H in binary base, making the sign bit 1, resulting in -81H. In order to avoid such problems, we have divided our program into 7F (HEX) and smaller with 80 (HEX) and larger.

If the number is greater than 7F (HEX), the BIGTHAN7F method starts working. If it is small, the PRINT2 method continues to run. If a number larger than 7F (HEX) was previously processed, the remaining operations are not performed and skipped to the PRINT method, since our number will be smaller in any case.

```

15 PRINT: INX H
16 MOV A, M

```

- The PRINT method allows us to read the next value by increasing the register index by 1.

```
38 BIGTHAN7F: ADD D ; Values greater than 7F.
39 CMA ; Take the complement.
40 MOV C, A
41 MOV A, E
42 ADD B
43 JZ CHANGE
44 LDA 0000H
45 CMA
46 SUB C ; A = A - C
47 JM PRINT ; If the sign bit is 1, return to top.
48
49 CHANGE: CMA
50 MOV A, M
51 STA 0000H
52 INR E ; It is enough to increase 1.
53 JMP PRINT
```

We add 80 to the number we subtracted before, and return it to its previous value by adding 80 here. If this is the first number greater than 7F (HEX) we have processed, the CHANGE method is called. The CHANGE method increases E by 1, allowing us to understand that we are trading with a number greater than 7F (HEX).

```
55 END: MVI H, 10D ; Convert to decimal.
56 MVI L, 0
57 MVI B, 0
58 MVI C, 0
59 MOV A, E
60 SUB B
61 JNZ BIG2 ; If the result is greater than 7F.
62 JMP END2
63
64 BIG2: LDA 0000H
65 SUB D ; Subtracts 128 from the maximum
66 ; value in decimal base.
67 STA 0000H
```

With the END method, we begin to convert the largest number we find to decimal base. If our number is a number greater than 7F (HEX), the BIG2 method is called and 128 in the BIG2 method is subtracted from the large number in a decimal base. If our number is not higher than 7F (HEX), END2 method is called.

```
69 END2: LDA 0000H ; Set the digits of the number.
70 MVI D, 0
71 SUB H
72 JM FINISH
73 STA 0000H
74 MOV A, B
75 INR A
76 MOV B, A
77 JMP END2
78
79 FINISH: ADD H
80 MOV C, A
81 ; The units digit of the number has been determined.
82 MOV A, B
83
84 HUND: SUB H
85 JM INTERVAL
86 MOV B, A; Tens digit
87 INR D ; Hundreds digit
88 JMP HUND
```

We adjust the digits of the maximum number with END2, FINISH, TOUCH methods.

```
90 INTERVAL: MOV A, E
91 SUB L
92 JNZ INCREASE
93 JMP LAST ; If the maximum number is less than 7F,
94 ; the program ends.
```

We check the range of the maximum number with the INTERVAL method. If it is a number less than 7F (HEX), the program ends. Otherwise, some additional operations are going

on. The first one is the INCREASE method. If the number is not less than 7F (HEX), the INCREASE method is called.

```
96 INCREASE: MOV A, C
97 ADI 8
98 MOV C, A
99 MOV A, B
100 ADI 2
101 MOV B, A
102 MOV A, D
103 ADI 1
104 MOV D, A
```


We converted the numbers larger than 7F (HEX) into decimal base, subtracting 128. With the help of the INCREASE method, we are adding the number of 128 again.

```
111 RETOUCH: MOV A, C
112 SUB H
113 JM RETOUCH2
114 MOV C, A
115 MOV A, B
116 INR A
117 MOV B, A
118 JMP RETOUCH
119
120 RETOUCH2: ADD H
121 MOV C, A
122 MOV A, B
123
124 RETOUCH3: SUB H
125 JM LAST
126 MOV B, A
127 INR D
128 JMP RETOUCH3
```

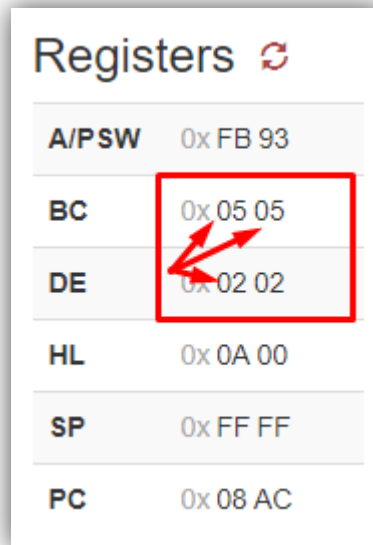
If the maximum value found is greater than 7F (HEX), when 128 is added to the result on a decimal basis, it may be possible to have a hand (accumulation). We correct this with the RETOUCH, RETOUCH2 and RETOUCH3 methods.

Testing:

Memory View																	
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	00	03	09	DA	80	FF	00	00	00	00	00	00	00	00	00	00	00
001	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
002	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
003	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
005	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
006	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
007	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
008	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
009	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Start Address at: 0x 

We entered 5 numbers in the addresses we accepted numbers.



A/PSW	0x FB 93
BC	0x 05 05
DE	0x 02 02
HL	0x 0A 00
SP	0x FF FF
PC	0x 08 AC

We can see that the number of 255, which is the decimal base of FF (HEX) number, is written in the registers we have determined previously.

References

- https://en.wikipedia.org/wiki/Intel_8085
- <https://electronicslesson.com/microprocessor/8085-microprocessor-architecture/>
- <https://www.youtube.com/watch?v=5puDuMhaRjM>