8085 SIMULATOR

AIM AND OBJECTIVES

Our aim is to create a user friendly and intuitive Simulator under Windows, for the 8085 processor. It offers complete transparency at the Register, Flag and Stack level. Easy coding through a Notepad with complete input validation. User with no extensive knowledge of 8085 can effectively use this tool. The software is a very flexible, reliable and easy to use pedagogical tool.

**Our** **objective** is to create an easy to use simulator by implementing:

* Absolute Transparency in functioning of the Microprocessor at Register and Flag levels.
* Constantly displays the contents of all the memory locations that are relevant to a program.
* Quick assembly language programming saving great amount of time and efforts.
* Indispensably useful for Learning and Teaching purposes
* The user is allowed to choose his own Start Address and Stack location as per his requirement.
* Total input validation - The beginner is guided all along by this well designed and intelligent Simulator.

8085  INSTRUCTION SET

**DATA TRANSFER GROUPS**

These groups include the move, exchange, load, and store operations. Data transfer instructions are among the most widely used of all microprocessor instructions. This group of instructions transfers data to and from registers and memory. None of the instructions of this group are the flag affecting instructions. The instructions included in this group are:

***MOV, MVI, LXI, LDA, STA and XCHG***

For e.g.: MOV (A, B) will move the contents of register B to register A.

**ARITHEMATIC AND LOGICAL GROUPS**

 This group includes the add, add with carry, subtract, subtract with borrow, increment, decrement, and decimal adjust accumulator operations. This group of instructions performs arithmetic and logical operations on data in registers and memory. Unless indicated otherwise all the instructions are flag affecting instructions. All subtraction operations are performed via 2s complement arithmetic and set the carry flag to 1 to indicate a borrow and clear it to indicate no borrow. The instructions included in this group are:

 ADD, INR, INX, DCR, DCX AND CMP

For e.g.: ADD B will add the contents of register B to the contents of register A and store the result in A.

**BRANCHING GROUPS**

This includes jump, call, return and restart instructions. This group of instructions alters the normal sequential program flow. The two types of branch instructions are:-Unconditional-Conditional

JNZ, JZ, JNC, JC

For e.g.: JZ 4000 will transfer the program flow to the memory location 4000 if zero flag is set.

**STACK, I/O AND MACHINE CONTROL INSTRUCTIONS**

This include push and pop, input and output, exchange, interrupt enables and disables, no operation and halts, and multi – purpose read and set interrupt mask instructions. This group of instructions performs inputs and outputs, manipulates the stack and alters the internal control flags. The instructions included in the group are:

Stack operations

* PUSH, POP, XTHL, and SPHL

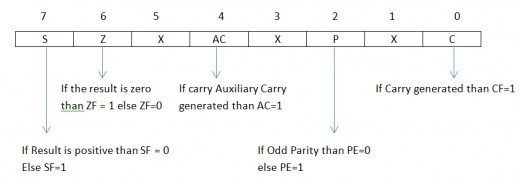
Control operations.

* HLT

For e.g.: PUSH B will push the contents the register B to the stack which is initialised previously.

8085 FLAG REGISTER

There are five flags in 8085, which are **sign flag (S), zero flag (Z), auxiliary carry flag (AC), parity flag (P)** and **carry flag (CY)**. The bit positions reserved for these flags in the flag registerare shown in figure below.



After an ALU operation, if the most significant bit of the result is 1, then **sign flag** is set.

The **zero flag** is set, if the ALU operation results in zero and it is reset if the result is non-zero.

In an arithmetic operation, when a carry is generated by the lower nibble, the auxiliary carry flag is set.

After an arithmetic or logical operation, if the result has an even number of 1's the **parity flag** is set, otherwise it is reset.

If an arithmetic operation results in a carry, the **carry flag** is set otherwise it is reset.

Among the five flags, the AC flag is used internally for BCD arithmetic and other four flags can be used by the programmer to check the conditions of the result of an operation.

EXECUTION OF THE CODE

GIVEN THE MEMORY ADDRESSES WITH ITS CONTENT AS FOLLOWS:

2000H 6

2001H 11

2002H 32

2003H 9

2004H 16

2005H 100

2006H 19

201AH 20

TEST PROGRAM FOR ADDITION OF TWO 8 BIT NUMBERS:

LXI H,2000H

MOV A,M

INX H

ADD M

STA 2006H

HLT

OUTPUT:

