Welcome to the 8085 Microprocessor Simulator!

This Web-browser tool helps you to understand and experiment with the Intel 8085 microprocessor, It's like a virtual playground where you can experiment and discover how this old-school tech works.

Navigation Bar

- ***Reset All:*** *Click this button to reset the simulator to its initial state.*
- ***Help:*** *Access helpful documentation or instructions.*
- ***Info:*** *Obtain information about the simulator.*

Internal Memory Panel

This section displays the internal registers and flags of the 8085 microprocessor:

Register Values

- ***A:*** *Accumulator register.*
- ***BC:*** *B and C registers.*
- ***DE:*** *D and E registers.*
- ***HL:*** *H and L registers.*
- ***PSW:*** *Program Status Word.*
- ***PC:*** *Program Counter.*
- ***SP:*** *Stack Pointer.*
- ***Int-Reg:*** *Interrupt Register.*

Flags

- ***S:*** *Sign Flag.*
- ***Z:*** *Zero Flag.*
- ***AC:*** *Auxiliary Carry Flag.*
- ***P:*** *Parity Flag.*
- ***C:*** *Carry Flag.*

Decimal - Hex Conversion

Convert decimal to hexadecimal and vice versa.

Enter values in the "Decimal" or "Hex" fields and use the corresponding buttons to perform conversions.

Memory - IO Update

Select whether you want to work with "Memory" or "I/O Ports" using the dropdown menu.

Enter the address and data values, then click "Update Value" to modify memory or I/O content.

Code Editor

The central part of the interface is the code editor.

You can write and edit 8085 assembly code here.

Control buttons include:

- ***Run:*** *Execute the code.*
- ***Compile:*** *Check the code for errors.*
- ***Zoom In/Out:*** *Adjust the font size in the code editor.*

External Memory Panel

This section allows you to view and manipulate memory and I/O ports.

Toggle between "Memory" and "I/O Ports" using the buttons.

The display shows address (hex), address (decimal), and data values.

Instruction: MOV (Move Data)

The "MOV" instruction in the 8085 microprocessor is used to move data from one register or memory location to another register or memory location. It is a fundamental data transfer operation.

Usage:

```
MOV destination, source
```

- `destination`*: The register or memory location where the data will be moved to.*
- `source`*: The register or memory location from which the data will be moved.*

Operation:

The "MOV" instruction copies the contents of the `source` *to the* `destination`*. Both* `source` *and* `destination` *can be registers (A, B, C, D, E, H, or L) or memory locations (indicated by "M").*

Example:

```
    MOV A, B
```

This code instructs the microprocessor to copy the contents of the B register into the A register. If, initially, A contains 0x0A and B contains 0x25, after executing this instruction, the A register will contain 0x25.

```
    MOV M, A
```

Will copy the contents of the A register to the memory location pointed to by HL register pair. If HL contains 0x3020 (hexadecimal address), the value at memory location 0x3020 will be updated to whatever A contains.

Notes:

- *The "MOV" instruction does not affect any flags; it is a pure data transfer operation.*
- *Ensure that the registers or memory locations you specify as* `destination` *and* `source` *are valid and accessible in your program.*

h2>Instruction: LXI (Load Immediate Data into Register Pair)

The "LXI" instruction in the 8085 microprocessor is used to load an immediate 16-bit value into a register pair. It is often used to initialize the stack pointer or other register pairs.

Usage:

```
LXI RP, 16-bit value
```

- ***RP:*** *The register pair (BC, DE, or HL) that will be loaded with the immediate value.*
- ***16-bit value:*** *The 16-bit hexadecimal value to load into the specified register pair.*

Operation:

The "LXI" instruction loads the specified register pair (BC, DE, or HL) with the 16-bit immediate value. This is useful for setting up addresses or initializing counters.

Example:

```
LXI BC, 0A23h
```

This code loads the BC register pair with the value 0A23h (BC = 0A23h).

Instruction: ADD (Add Accumulator and Register/Memory)

The "ADD" instruction in the 8085 microprocessor is used to add the contents of a specified register or memory location to the accumulator (A).

Usage:

```
ADD Register/Memory
```

- ***Register/Memory:*** *The register or memory location whose value will be added to the accumulator.*

Operation:

The "ADD" instruction performs addition by adding the value stored in the specified register or memory location to the current value of the accumulator (A). The result is stored back in the accumulator.

Flags Affected:

- ***Z (Zero Flag):*** *Set if the result of addition is zero.*
- ***S (Sign Flag):*** *Set if the result is negative (most significant bit is 1).*
- ***P (Parity Flag):*** *Set if the number of set bits in the result is even.*
- ***C (Carry Flag):*** *Set if there is a carry out from the most significant bit during addition.*
- ***AC (Auxiliary Carry Flag):*** *Set if there is a carry out from bit 3 to bit 4 during addition, used for BCD arithmetic.*

Example:

```
ADD B ; Add the value of register B to the accumulator
ADD M ; Add the value at the memory location pointed to by HL to the accumulator
```

The above code examples demonstrate adding the contents of register B and memory location pointed to by HL to the accumulator, respectively.

Instruction: INX (Increment Register Pair)

The "INX" instruction in the 8085 microprocessor is used to increment the value of a register pair by 1.

Usage:

```
INX Register Pair
```

- ***Register Pair:*** *The register pair (BC, DE, or HL) that will be incremented.*

Operation:

The "INX" instruction increments the 16-bit value stored in the specified register pair (BC, DE, or HL) by 1. This instruction is commonly used for pointer manipulation, such as incrementing a memory address.

Flags Affected:

The "INX" instruction does not affect any of the flags in the flag register.

Example:

```
INX BC ; Increment the BC register pair by 1
INX HL ; Increment the HL register pair by 1
```

The above code examples demonstrate incrementing the values of register pairs BC and HL by 1, respectively.

Instruction: LDA (Load Accumulator Directly)

The "LDA" instruction in the 8085 microprocessor is used to load the accumulator (A) with the contents of a specific memory address.

Usage:

```
LDA 16-bit memory address
```

- ***16-bit memory address:*** *The 16-bit hexadecimal memory address from which the accumulator will be loaded.*

Operation:

The "LDA" instruction retrieves the data from the specified memory address (16-bit) and loads it directly into the accumulator (A). This is commonly used to fetch data from memory into the accumulator for further processing.

Flags Affected:

The "LDA" instruction does not affect any of the flags in the flag register.

Example:

```
LDA 2050h ; Load the accumulator with the data stored at memory address 2050h
LDA 0A23h ; Load the accumulator with the data stored at memory address 0A23h
```

The above code examples demonstrate loading the accumulator (A) with data from different memory addresses.

Instruction: STA (Store Accumulator Directly)

The "STA" instruction in the 8085 microprocessor is used to store the contents of the accumulator (A) into a specific memory address.

Usage:

```
STA 16-bit memory address
```

- ***16-bit memory address:*** *The 16-bit hexadecimal memory address where the content of the accumulator will be stored.*

Operation:

The "STA" instruction takes the data from the accumulator (A) and stores it directly into the specified memory address (16-bit). This instruction is commonly used to write data from the accumulator into a specific memory location.

Flags Affected:

The "STA" instruction does not affect any of the flags in the flag register.

Example:

```
STA 2050h ; Store the contents of the accumulator into memory address 2050h
STA 0A23h ; Store the contents of the accumulator into memory address 0A23h
```

The above code examples demonstrate storing the accumulator (A) data into different memory addresses.

Instruction: MVI (Move Immediate Value into Register or Memory)

The "MVI" instruction in the 8085 microprocessor is used to move an immediate 8-bit value into a specified register or memory location.

Usage:

```
MVI R, 8-bit value

MVI M, 8-bit value
```

- ***R:*** *The destination register (A, B, C, D, E, H, or L) where the immediate value will be moved.*
- ***M:*** *Memory address specified by the HL register pair where the immediate value will be stored.*
- ***8-bit value:*** *The 8-bit hexadecimal value to move into the specified register or memory location.*

Operation:

The "MVI" instruction copies the immediate 8-bit value into either the specified register (R) or the memory location pointed to by the HL register pair (M). This instruction is commonly used to load data into registers or memory for further processing.

Flags Affected:

The "MVI" instruction does not affect any of the flags in the flag register.

Examples:

```
MVI A, 42h   ; Load the accumulator (A) with the value 42h
MVI B, 0Fh   ; Load register B with the value 0Fh
MVI M, 78h   ; Store the value 78h into the memory location pointed by HL
```

The above code examples demonstrate how to use the "MVI" instruction to move immediate values into registers (A, B) and memory (M).

Instruction: DCX (Decrement Register Pair)

The "DCX" instruction in the 8085 microprocessor is used to decrement the value of a specified register pair by 1.

Usage:

```
DCX RP
```

- ***RP:*** *The register pair (BC, DE, HL, or SP) to be decremented.*

Operation:

The "DCX" instruction decrements the value of the specified register pair by 1. This instruction is commonly used for operations like decreasing a memory address or stack pointer.

Flags Affected:

The "DCX" instruction does not affect any of the flags in the flag register.

Examples:

```
DCX B     ; Decrement the BC register pair by 1
DCX HL    ; Decrement the HL register pair by 1
DCX SP    ; Decrement the Stack Pointer (SP) by 1
```

The above code examples demonstrate how to use the "DCX" instruction to decrement various register pairs, including BC, HL, and the Stack Pointer (SP).

Instruction: SUB (Subtract Accumulator from Register/Memory)

The "SUB" instruction in the 8085 microprocessor is used to subtract the value of a specified register or memory location from the contents of the accumulator (A).

Usage:

```
SUB R/M
```

- ***R/M:*** *The register or memory location whose value will be subtracted from the accumulator.*

Operation:

The "SUB" instruction performs subtraction by complementing (inverting) the bits of the specified operand and then adding it to the accumulator. The result is stored back in the accumulator.

Flags Affected:

The "SUB" instruction affects the following flags:

- ***Z (Zero Flag):*** *Set if the result in the accumulator is zero.*
- ***S (Sign Flag):*** *Set if the most significant bit (MSB) of the result is 1 (indicating a negative result).*
- ***P (Parity Flag):*** *Set if the parity of the result is even.*
- ***C (Carry Flag):*** *Set if there is a borrow from the most significant bit during subtraction (indicating a negative result).*
- ***AC (Auxiliary Carry Flag):*** *Set if there is a borrow from bit 3 to bit 4 during subtraction (used for BCD arithmetic).*

Examples:

```
SUB B     ; Subtract the value in register B from A
SUB M     ; Subtract the value at the memory location pointed to by HL from A
```

The above code examples demonstrate how to use the "SUB" instruction to subtract values from the accumulator. It can subtract values from registers (like B) or memory locations (using HL as a pointer).