8085 Microprocessor: Internal Architecture

Course Teacher:

Md. Obaidur Rahman, Ph.D.
Professor
Department of Computer Science and Engineering (CSE)
Dhaka University of Engineering & Technology (DUET), Gazipur.

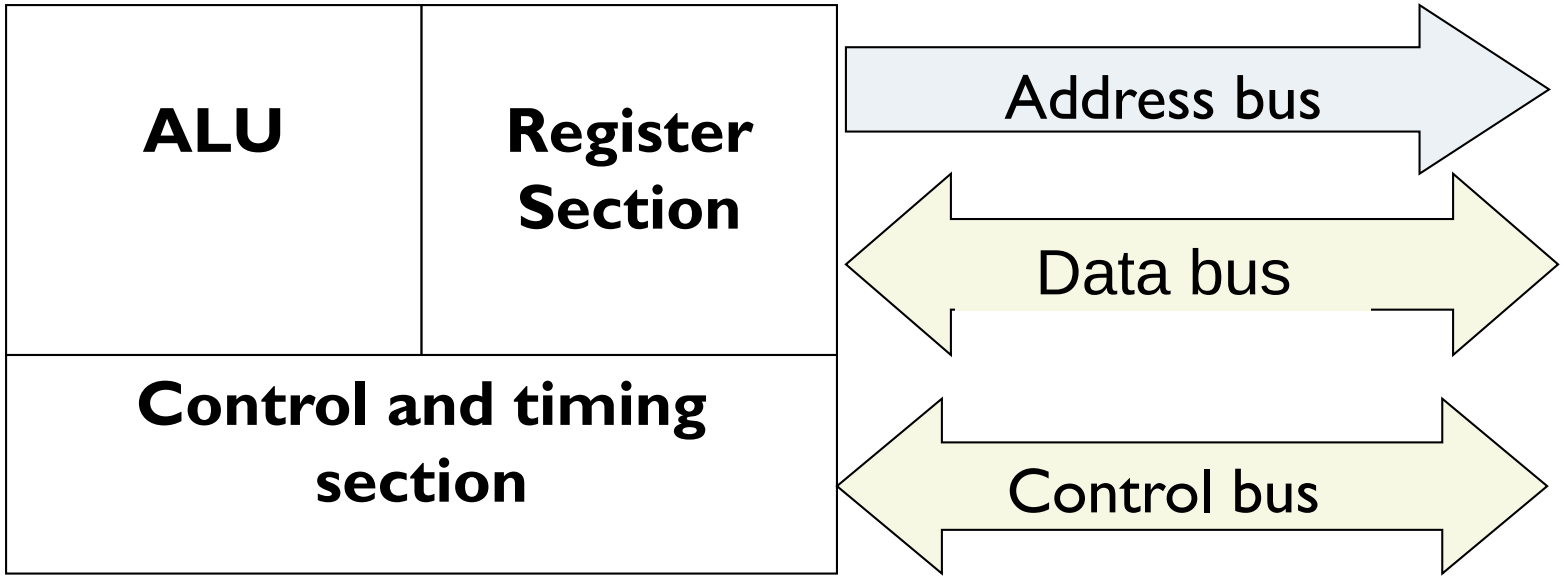
Course ID: CSE - 4503
Course Title: Microprocessors and Assembly Language
Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT), Gazipur.

Lecture References:

- **Book:**
  - *Microprocessor, architecture, programming & application with the 8085, Chapter # 2,* **Author:** Gaonkar

- **Lecture Materials:**
  - *IBM PC Organization,* CAP/IT221

Internal Structure of a Microprocessor

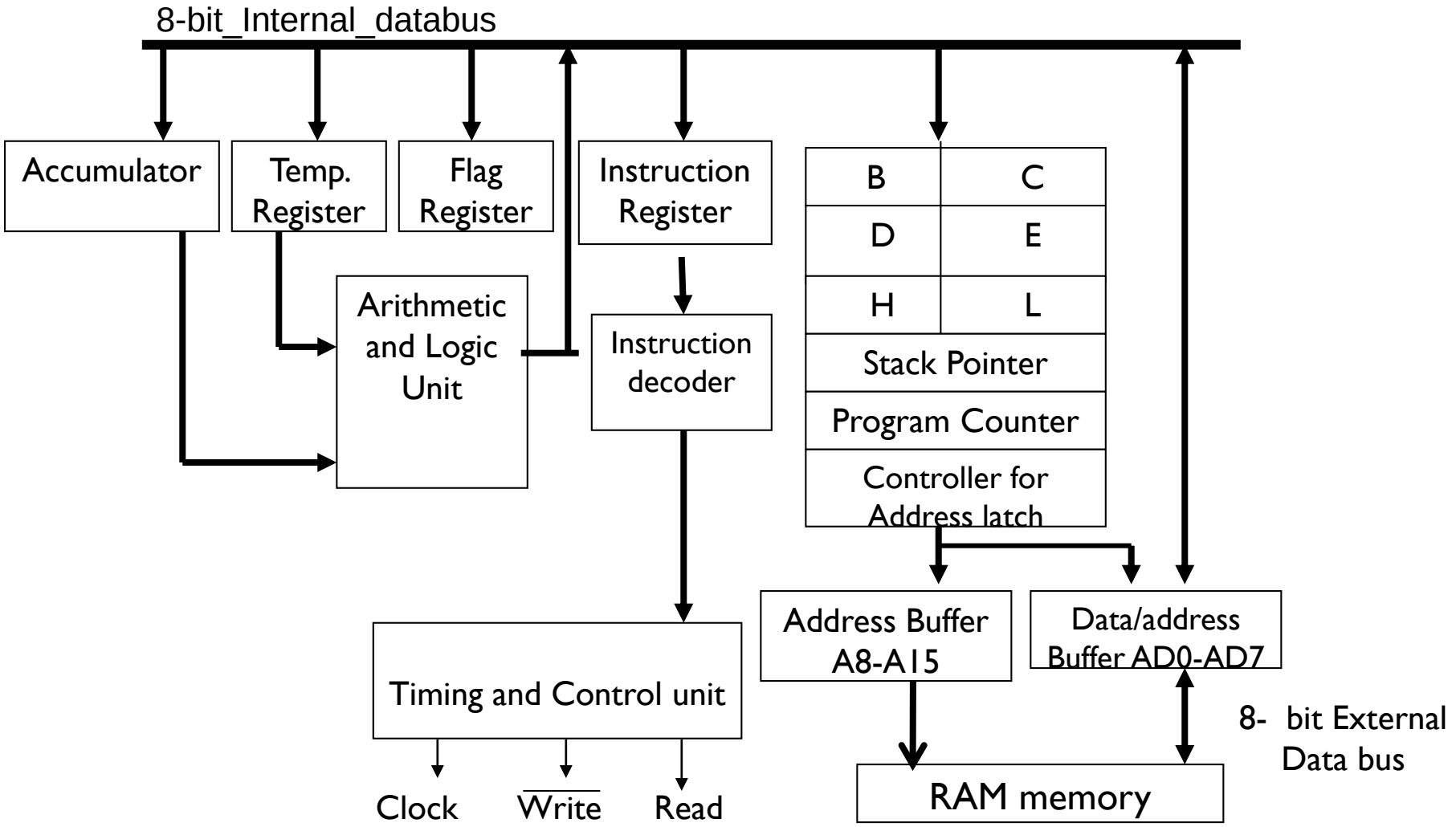


Block Diagram of a Microprocessor

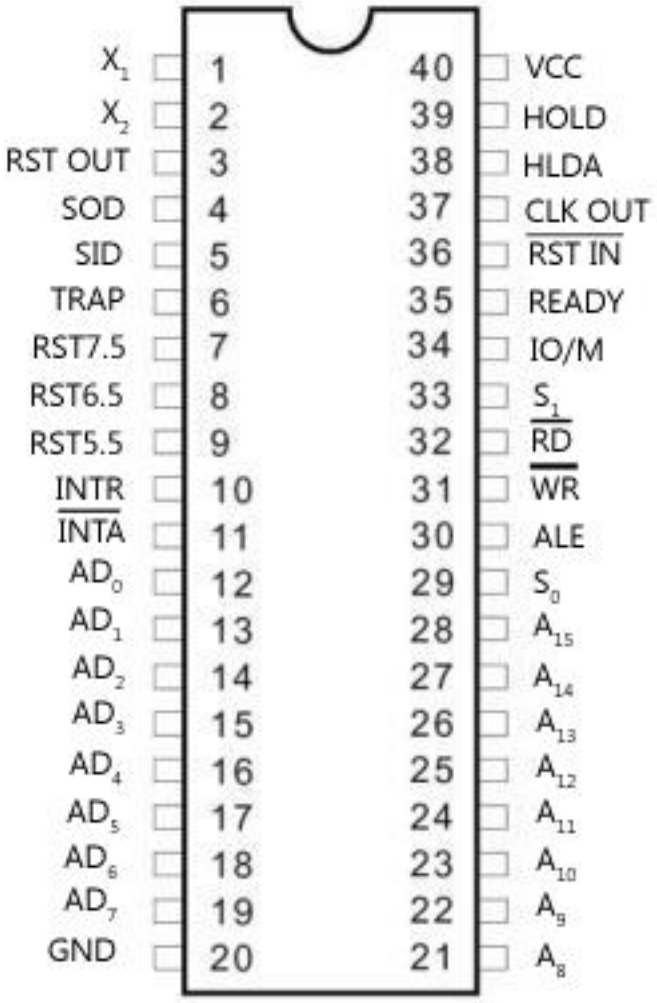
8085 Microprocessor

- **Intel 8085**
  - The 8085 microprocessor was introduced by Intel in the year 1976.
  - 8-bit microprocessor with 16-bit address bus and 8-bit data bus.
  - This microprocessor is an update of 8080 microprocessor.
  - It is an 8-bit microprocessor with a 40 pin Dual in Line Package (DIP)
  - Total 74 operation codes in assembly language and those can generate 246 instructions.

8-Bit 8085 Intel Processor Architecture

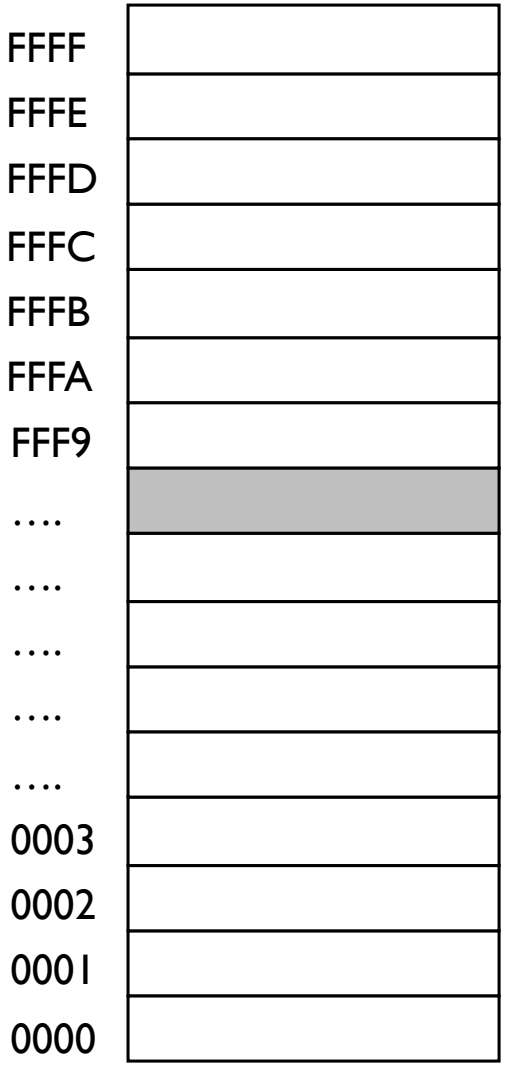


8-Bit 8085 Intel Processor (Pin Diagram)



8085 Registers and Memory

- **Registers:**
  - Total 11 registers and 1 temporary register
  - Information is stored in registers
  - Registers are classified according to the functions they perform
- 64 Kbytes of memory and 65536 memory locations.



8085 Register Categories

- **Accumulator** – 8 bit register which holds the latest result from ALU
- **B, C, D, E, H** and **L** are general purpose registers
- **HL** pair can be used for indirect addressing as well
- **Program counter** – 16 bit register which holds the address of the next instruction to be executed
- **Instruction Register** – It holds the instruction that is currently being processed.
- **Stack Pointer** is used during subroutine calling and execution.
- **Address Latch** – It increments/ decrements the address before sent to the address buffer

8085 Register Categories

- **Flag Register**

- **Sign Flag**: If the result of the latest arithmetic operation is having MSB (most- significant byte) '1' (meaning it is a negative number), then the sign flag is set to '1'. Otherwise, it is reset to '0' which means it is a positive number.

- **Zero Flag**: If the result of the latest operation is zero, then zero flag will be set to '1'; otherwise it be reset to '0'.

- **Auxiliary Carry Flag**: This flag is not accessible to programmer. This flag will be used by the system during BCD (binary-coded decimal) operations.

- **Parity Flag**: If the result of the latest operation is having even number of '1's, then this flag will be set to '1' Otherwise this will be reset to '0'. This is used for error checking.

- **Carry Flag**: If the result of the latest operations exceeds 8-bits then this flag will be set to '1'. Otherwise it be reset to '0'.

Simple Assembly Program in 8085

```
MVI   A, 32H
MVI   B , 48H
ADD   B
OUT 01H
HLT
```

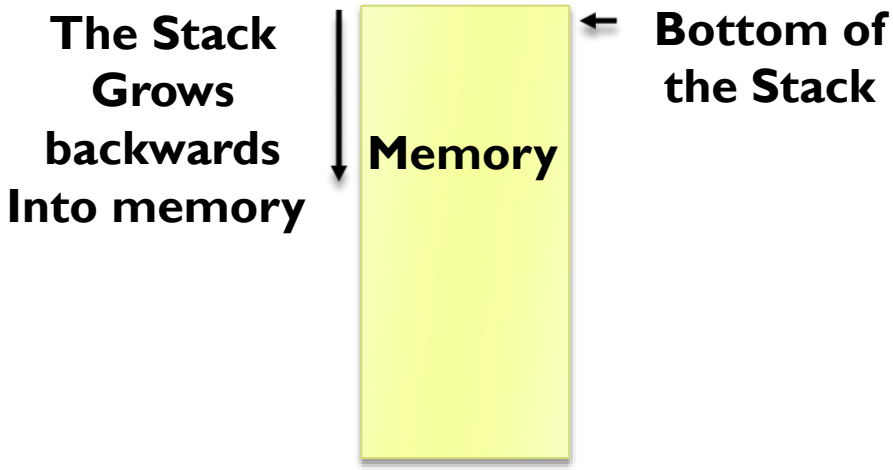
Task:

- Derive the flag values after the ADD instructions

Memory Address	Contents (Binary)	Contents (Hex)	Operation
2000h	0011 1110	3E	Load Reg. Acc.
2001h	0011 0010	32	Value is 32h
2002h	0000 0110	06	Load Reg. B
2003h	0100 1000	48	Value is 48h
2004h	1000 0000	80	Add B with A & Store in A
2005h	1101 0011	D3	Display
2006h	0000 0001	01	Port Id 01h
2007h	0111 1100	76	End

Stack Pointer and Stack Memory

- The stack is an area of memory identified by the programmer for temporary storage of information.
- The stack is a LIFO structure.
- The stack normally grows backwards into memory.
  - Programmer can defines the bottom of the stack (**SP**) and the stack grows up into reducing address range.



Stack Memory

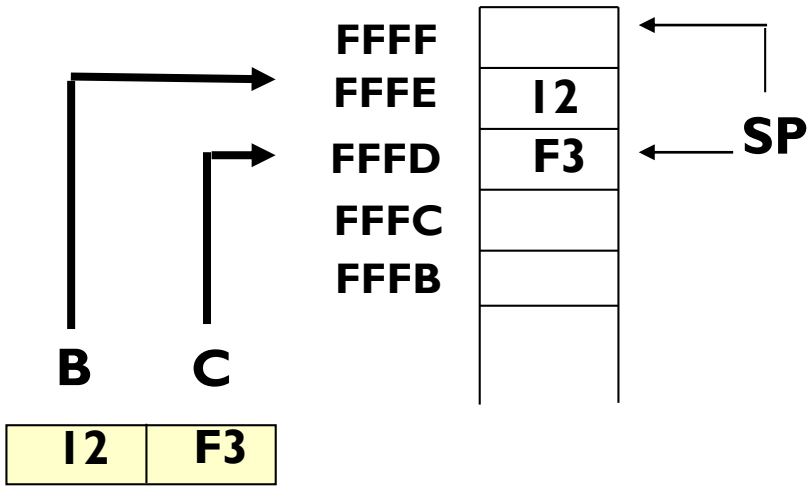
- Grows backwards into memory
- Better to place the bottom of the stack at the end of memory
- To keep it as far away from user programs as possible.
- Stack is defined by setting the SP (Stack Pointer) register.

LXI SP, FFFFh ; Load 16-bit number in a register

- This sets SP to location FFFFh (end of memory for 8085).

Saving Information in Stack

- Save information by PUSHing onto STACK
- Retrieved from STACK by POPing it off.
- PUSH and POP work with register pairs only.
- Example "PUSH B"
  - Decrement SP, Copy B to 0(SP)
  - Decrement SP, Copy C tp 0(SP)
- Example "POP B"
  - Copy SP to C, Increment SP
  - Copy SP to B, Increment SP



Thank You !!