UNIT-5 ASSEMBLY LANGUAGE PROGRAMMING

INTRODUCTION TO MICROPROCESSOR 8085:

The 8085 is pronounced as "eighty-eighty-five" microprocessor. It is an 8-bit microprocessor designed by Intel in 1977 using NMOS (N-channel metal-oxide semiconductor) technology.

It has the following configuration −

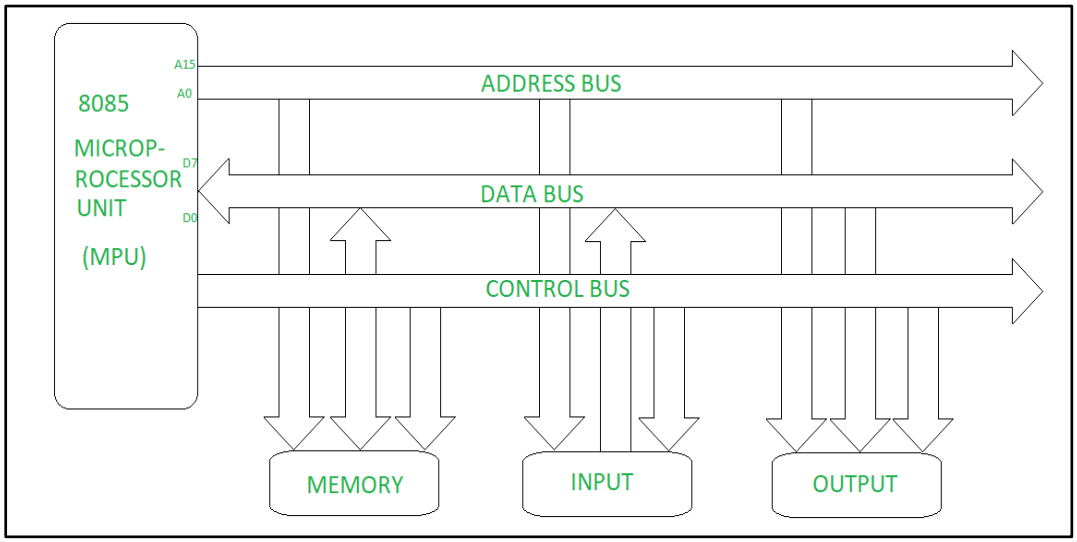
- 8-bit data bus and 16-bit address bus, which can address upto 64KB
- A 16-bit program counter and 16-bit stack pointer
- Six 8-bit registers arranged in pairs: BC, DE, HL
- Requires +5V supply to operate at 3.2 MHZ single phase clock

Here are some key features of the 8085 microprocessor:

1. **8-Bit Architecture:** The 8085 is an 8-bit microprocessor, which means it processes data in 8-bit chunks, making it suitable for handling relatively small data sizes.
2. **16-Bit Address Bus:** The 8085 has a 16-bit address bus, allowing it to address up to 64 KB (64 * 1024 bytes) of memory. This provides a significant addressable memory range for its time.
3. **Internal Registers:** The 8085 includes several internal registers:
   - Accumulator (A)
   - General-purpose registers (B, C, D, E, H, L)
   - Temporary register (WZ)
   - Special-purpose registers (Program Counter: PC, Stack Pointer: SP, Flag Register: FLAGS)
4. **Clock Speed:** The 8085 typically operates at clock speeds ranging from 2 to 3 MHz. This clock speed determines the execution rate of instructions.
5. **Instruction Set:** The 8085 has a compact instruction set with a total of 74 instructions. These instructions cover a wide range of data manipulation, logical, arithmetic, and control operations.
6. **Addressing Modes:** The 8085 supports various addressing modes, including immediate, direct, indirect, and register addressing. These modes provide flexibility in accessing memory and operands.
7. **Interrupts:** The 8085 supports five interrupt lines to handle external events and interrupt-driven operations. Interrupts enable the microprocessor to respond to time-critical tasks and events.
8. **Flag Register:** The 8085 has a flag register (FLAGS) that contains important status flags such as carry (CY), zero (Z), sign (S), parity (P), and auxiliary carry (AC). These flags are used to indicate the results of arithmetic and logic operations.
9. **Serial I/O:** The 8085 has serial input/output lines (SID and SOD), which enable serial communication with external devices.
10. **Power Supply:** The 8085 typically operates on a +5V power supply.
11. **Pin Configuration:** The standard 8085 microprocessor package has 40 pins, each serving specific functions, including data bus, address bus, control signals, clock input, and power supply connections.
12. **Backward Compatibility:** The 8085 is designed to be backward compatible with the earlier 8080 microprocessor, easing the transition for systems using the 8080.

BUS STRUCTURE OF 8085 MICROPROCESSOR:

The bus organization of 8085 microprocessor is the way in which the microprocessor communicates with other devices in a computer system. The 8085 microprocessor has a 16-bit address bus, an 8-bit data bus, and various control signals that are used to manage data transfer and other operations.



1. **Address Bus:**

   The **Address Bus** is used to specify the memory location or device with which the microprocessor wants to communicate. It is 16 bits wide, which allows the microprocessor to address up to 64K bytes of memory. The address bus is unidirectional, which means that data can only flow in one direction from the microprocessor to the addressed device.

   - **Unidirectional:** The Address Bus is a unidirectional bus, meaning it carries information only in one direction, from the microprocessor to the external memory or I/O devices.
   - **Size:** The 8085 has a 16-bit Address Bus, consisting of 16 address lines. These address lines allow the microprocessor to address a maximum of 64KB (2^{16} = 65536) of memory locations.
   - **Function:** The microprocessor uses the Address Bus to specify the memory location or I/O device with which it wants to read data from or write data to. The addresses on the Address Bus determine the specific location in the memory map that the microprocessor is currently accessing.

2. **Data Bus:**

   The **Data Bus** is used to transfer data between the microprocessor and other devices. It is 8 bits wide, which means that data can be transferred in bite-sized chunks. The data bus is bidirectional, which means that data can flow in either direction between the microprocessor and other devices.

   - **Bidirectional:** The Data Bus is a bidirectional bus, meaning it carries data in both directions, between the microprocessor and the external memory or I/O devices.
   - **Size:** The 8085 has an 8-bit Data Bus, consisting of 8 data lines. This means that the microprocessor can transfer 8 bits (1 byte) of data at a time between itself and the memory or I/O devices.
   - **Function:** The Data Bus is used to transfer actual data between the microprocessor and the memory or I/O devices. During read operations, data from the memory or I/O device is placed on the Data Bus for the microprocessor to read. During write operations, the microprocessor places data on the Data Bus to be written to the specified memory location or I/O device.

3. **Control Bus:**

   The **Control Bus** has various control signals that are used to manage data transfer and other operations. These control signals include the read (RD), write (WR), and hold (HLDA) signals. The RD and WR signals are used to control data transfer to and from memory or other devices, while the HLDA signal is used to indicate that microprocessor is in a hold state and cannot execute instructions.

   - **Bidirectional:** While some lines of the Control Bus are used to send control signals from the microprocessor to external devices, there are also lines for status signals from external devices back to the microprocessor.
   - **Function:** The Control Bus carries various control and status signals that coordinate the operations of the microprocessor and the external devices. These signals include:
   - **Read and Write Control Signals:** Indicate whether the current operation is a read (data is being fetched) or write (data is being stored) operation.
   - **Interrupt Request and Acknowledge Signals:** Used for interrupt handling, where external devices can interrupt the microprocessor to handle specific events.
   - **Clock Signals:** Control the timing of the microprocessor's operations.
   - **Reset Signals:** Used to reset the microprocessor and start execution from a known location.
   - **Control Signals for Specific Operations:** Signals that indicate the current operation being performed by the microprocessor, such as memory or I/O operations.

Advantages:

1. **Flexibility:** The bus organization used in the 8085 microprocessor allows it to communicate with a wide range of devices. This flexibility makes it well-suited for use in a variety of computer systems, including embedded systems, personal computers, and other devices.
2. **Modularity:** The bus organization makes it easy to add or remove devices from a computer system. This modularity allows system designers to customize the system to meet the needs of specific applications.
3. **Scalability:** The bus organization used in the 8085 microprocessor is scalable, which means that it can be used in systems of varying sizes and complexity. This scalability makes it well-suited for use in systems that require a wide range of performance levels.
4. **Low Cost:** The bus organization used in the 8085 microprocessor is relatively simple and inexpensive to implement. This makes it an attractive option for low-cost, embedded applications.

Disadvantages:

1. **Limited Bandwidth:** The bus organization used in the 8085 microprocessor has a limited bandwidth, which can limit the performance of the processor in high-performance applications.
2. **Latency:** The bus organization can introduce latency, which is the delay between the time a command is issued and the time the response is received. This latency can be a problem in real-time applications that require immediate responses.
3. **Data Integrity:** The bus organization used in the 8085 microprocessor is vulnerable to data corruption due to electromagnetic interference and other sources of noise. This can lead to errors in data transmission and processing.
4. **Complexity:** The bus organization used in the 8085 microprocessor can be complex to implement and troubleshoot, which can increase the cost and time required to develop and maintain computer systems.

Uses of Bus Organization in 8085 Microprocessor:

1. **Memory access:** The bus organization is used for accessing memory by transferring the address of the memory location through the address bus and the data to be stored or retrieved through the data bus. This enables the microprocessor to read and write data to and from memory, which is essential for executing instructions and storing data.
2. **I/O operations:** The bus organization is used for performing input/output (I/O) operations by transferring the input/output device address through the address bus and the data to be input or output through the data bus. This enables the microprocessor to communicate with peripheral devices such as keyboards, displays, and sensors.
3. **Interrupt handling:** The bus organization is used for interrupt handling, where the microprocessor uses the address bus to fetch the interrupt vector and the data bus to fetch the interrupt service routine. This enables the microprocessor to respond to external events and perform time-critical operations.
4. **DMA operations:** The bus organization is used for performing Direct Memory Access (DMA) operations, where the data transfer between the memory and I/O devices takes place without the intervention of the microprocessor. This enables high-speed data transfer between devices and reduces the load on the microprocessor.
5. **Control signal transfer:** The bus organization is used for transferring control signals between the microprocessor and other components of the system. This enables the microprocessor to control the operation of devices and coordinate the execution of instructions.

Issues of Bus organization in 8085 microprocessor:

1. **Limited data transfer rate:** The 8085 microprocessor has an 8-bit data bus, which means that it can transfer only 8 bits of data at a time. This limited data transfer rate can be a bottleneck in systems that require faster data transfer.
2. **Limited address range:** It has a 16-bit address bus, which limits the addressable memory to 64 KB. This can be a limitation in systems that require larger memory addressing.
3. **Bus contention:** Bus contention occurs when two or more devices try to use the bus at the same time. This can cause data corruption and other errors in the system.
4. **Timing issues:** The bus organization requires precise timing for the signals to be transmitted correctly. Any timing errors can cause data corruption or other errors in the system.
5. **Limited number of devices:** The bus organization of the 8085 microprocessor can support a limited number of devices due to its limited bus width and address range. This can be a limitation in systems that require more devices to be connected.
6. **Noise interference:** The signals on the bus can be affected by noise interference, which can cause errors in the system.
7. **Power consumption:** The bus organization can consume significant power, especially when many devices are connected to the bus. This can be a limitation in portable or low-power systems.

INSTRUCTION SETS OF 8085:

1. Data Transfer Instruction
2. Arithmetic Instruction
3. Logical Instruction
4. Control Transfer Instruction
5. Stack and I/O Instruction

DATA TRANSFER INSTRUCTIONS:

- MOV: Move data from source to destination.
- MVI: Move immediate data to a register or memory location.
- LDA: Load the accumulator with data from a memory address.
- STA: Store the contents of the accumulator into a memory address.
- LHLD: Load the HL register pair with data from a memory address.
- SHLD: Store the contents of the HL register pair into a memory address.
- LDAX: Load accumulator with data from a memory location pointed to by BC or DE.
- STAX: Store contents of accumulator into a memory location pointed to by BC or DE.

Mnemonic	Opcode	Description
MOV Rd, Rs	8x	Move data from source (Rs) to destination (Rd)
MVI Rd, data	3E	Move immediate data to destination (Rd)
LDA address	3A	Load accumulator with data from memory address
STA address	32	Store accumulator content into memory address
LHLD address	2A	Load HL register pair from memory address
SHLD address	22	Store HL register pair into memory address
LDAX Rp	0A (B), 1A (D)	Load accumulator from memory location pointed to by BC (B) or DE (D)
STAX Rp	02 (B), 12 (D)	Store accumulator into memory location pointed to by BC (B) or DE (D)

In the table:

- "Rd" and "Rs" represent destination and source registers, respectively. The actual register designations (A, B, C, D, E, H, L) are used.
- "data" represents an immediate data value.
- "address" represents a memory address.
- "Rp" represents the register pairs BC or DE.

ARITHMETIC INSTRUCTIONS:

- ADD: Add data to the accumulator.
- ADI: Add immediate data to the accumulator.
- ADC: Add data to the accumulator with carry.
- ACI: Add immediate data to the accumulator with carry.
- SUB: Subtract data from the accumulator.
- SUI: Subtract immediate data from the accumulator.
- SBB: Subtract data from the accumulator with borrow.
- SBI: Subtract immediate data from the accumulator with borrow.
- INR: Increment a register or memory location.
- DCR: Decrement a register or memory location.

Mnemonic	Opcode	Description
ADD R	80-87	Add the contents of register (R) to the accumulator
ADI data	C6	Add immediate data to the accumulator
ADC R	88-8F	Add the contents of register (R) and carry flag to the accumulator
ACI data	CE	Add immediate data and carry flag to the accumulator
SUB R	90-97	Subtract the contents of register (R) from the accumulator
SUI data	D6	Subtract immediate data from the accumulator
SBB R	98-9F	Subtract the contents of register (R) and borrow (carry) flag from the accumulator
SBI data	DE	Subtract immediate data and borrow (carry) flag from the accumulator
INR R	04-3D	Increment the specified register (R) by 1
DCR R	05-3D	Decrement the specified register (R) by 1

In the table:

- "R" represents the specific registers used in the instructions (A, B, C, D, E, H, L).
- "data" represents an immediate data value.
- The range of opcodes for each instruction indicates that there are multiple versions of these instructions, one for each possible register (e.g., ADD A, ADD B, ADD C, etc.).

LOGICAL INSTRUCTIONS:

- ANA: Perform bitwise AND between data and the accumulator.
- ANI: Perform bitwise AND between immediate data and the accumulator.
- XRA: Perform bitwise XOR between data and the accumulator.
- XRI: Perform bitwise XOR between immediate data and the accumulator.
- ORA: Perform bitwise OR between data and the accumulator.
- ORI: Perform bitwise OR between immediate data and the accumulator.
- CMA: Complement (bitwise NOT) the contents of the accumulator.
- CMP: Compare data with the accumulator.

Mnemonic	Opcode	Description
ANA R	A0-A7	Perform bitwise AND with register (R) and accumulator
ANI data	E6	Perform bitwise AND with immediate data and accumulator
XRA R	A8-AF	Perform bitwise XOR with register (R) and accumulator
XRI data	EE	Perform bitwise XOR with immediate data and accumulator
ORA R	B0-B7	Perform bitwise OR with register (R) and accumulator
ORI data	F6	Perform bitwise OR with immediate data and accumulator
CMA	2F	Complement (bitwise NOT) the contents of the accumulator
CMP R	B8-BF	Compare the contents of register (R) with the accumulator

In the table:

- "R" represents the specific registers used in the instructions (A, B, C, D, E, H, L).
- "data" represents an immediate data value.
- The range of opcodes for each instruction indicates that there are multiple versions of these instructions, one for each possible register (e.g., ANA A, ANA B, ANA C, etc.).

CONTROL TRANSFER INSTRUCTIONS:

- CALL: Call a subroutine at a specified memory address.
- RET: Return from a subroutine.
- JMP: Jump to a specified memory address.
- JNZ/JZ/JNC/JC/JPO/JPE/JP/JM: Conditional jump instructions based on the status of various flags.
- RIM: Read status of the interrupt system and special function registers.
- SIM: Set status of the interrupt system and special function registers.

Mnemonic	Opcode	Description
CALL address	CD	Call a subroutine at the specified memory address
RET	C9	Return from a subroutine
JMP address	C3	Unconditional jump to the specified memory address
JNZ address	C2	Jump if not zero (Z flag is not set)
JZ address	CA	Jump if zero (Z flag is set)
JNC address	D2	Jump if no carry (carry flag is not set)
JC address	DA	Jump if carry (carry flag is set)
JPO address	E2	Jump if parity odd (P flag is not set)
JPE address	EA	Jump if parity even (P flag is set)
JP address	F2	Jump if positive (S flag is not set)
JM address	FA	Jump if negative (S flag is set)
RIM	20	Read status of the interrupt system and special function registers
SIM	30	Set status of the interrupt system and special function registers

In the table:

- "address" represents a memory address.
- The opcodes provided are for illustrative purposes and represent the most common cases. The actual opcodes for these instructions depend on the specific addressing mode or additional flags.
- The mnemonic "CALL" is used for subroutine calls, "RET" is used for subroutine returns, "JMP" is an unconditional jump, and others (e.g., JNZ, JZ) are conditional jumps based on status of various flags.

STACK AND I/O INSTRUCTIONS:

- PUSH: Push data onto the stack.
- POP: Pop data from the stack.
- HLT: Halt the microprocessor.
- IN: Input data from an I/O port.
- OUT: Output data to an I/O port.

Mnemonic	Opcode	Description
PUSH Rp	C5 (B), D5 (D), E5 (H), F5 (PSW)	Push the contents of the specified register pair (Rp) onto the stack
POP Rp	C1 (B), D1 (D), E1 (H), F1 (PSW)	Pop the top of the stack and store the value in the specified register pair (Rp)
HLT	76	Halt the microprocessor
IN port	DB	Input data from the specified I/O port into the accumulator
OUT port	D3	Output data from the accumulator to the specified I/O port

In the table:

- "Rp" represents the register pairs: BC (B), DE (D), HL (H), and PSW.
- "port" represents a specific I/O port address.
- The PUSH instruction pushes the contents of a register pair onto the stack, while the POP instruction pops a value from the stack and stores it in a register pair. The HLT instruction halts the microprocessor, effectively stopping further execution.
- The IN instruction inputs data from a specified I/O port into the accumulator, and the OUT instruction outputs data from the accumulator to a specified I/O port.

INTRODUCTION TO ASSEMBLY LANGUAGE PROGRAMMING:

Assembly language programming for the 8085 microprocessor involves writing low-level instructions that the microprocessor can understand and execute. The 8085 is an 8-bit microprocessor with a relatively simple instruction set, making it a good platform for learning assembly programming concepts.

Here are some key points to understand about 8085 assembly language programming:

1. **Machine Code and Assembly Language:**
   - **Machine code** is the binary representation of instructions that the microprocessor executes directly. It consists of sequences of 1s and 0s.
   - **Assembly language** is a human-readable representation of machine code. It uses mnemonics to represent instructions and symbolic names for registers and memory locations.
2. **Registers and Memory:**
   - The 8085 microprocessor has a set of registers, including the accumulator (A), general-purpose registers (B, C, D, E, H, L), and special-purpose registers (e.g., Program Counter, Stack Pointer).
   - It can access up to 64KB of memory (addressable by 16-bit addresses).
   - Memory locations can hold data and program instructions.
3. **Instruction Format:**
   - Each 8085 instruction consists of an opcode (operation code) and operands (registers, memory addresses, immediate values).
   - Instructions may have zero, one, or two operands.
   - The size of operands (8-bit or 16-bit) depends on the specific instruction.
4. **Programming Concepts:**
   - Data transfer instructions move data between registers and memory locations.
   - Arithmetic and logical instructions perform operations on data in registers and memory.
   - Control transfer instructions alter the program flow, enabling loops, conditional statements, and subroutines.
   - Stack and I/O instructions handle stack operations and input/output operations.

5. **Assembling and Running:**
   - Assembly language programs are written in a text editor using mnemonics and symbolic names.
   - The assembly program is then assembled into machine code using an assembler.
   - The resulting machine code can be loaded onto the 8085 microprocessor using appropriate hardware or software (e.g., a simulator) to execute the program.
6. **Debugging and Optimization:**
   - Debugging assembly programs can be challenging due to the low-level nature of the code.
   - Debuggers and simulators can help in identifying errors and analyzing program behavior.
   - Optimization techniques, such as code size reduction and efficient algorithms, are important in assembly programming.

For Examples: Refer Microprocessor Practical File

Programming Techniques with Additional Instructions in 8085:

1. **Subroutine Calls (CALL and RET):**
   - The CALL instruction is used to call subroutines. It saves the return address (address after the CALL instruction) on the stack and transfers control to the specified memory address.
   - The RET instruction is used to return from a subroutine. It pops the return address from the stack and transfers control to that address.
   - Subroutines allow you to modularize your code and reuse specific functionality.
2. **Looping (Conditional Jumps):**
   - The conditional jump instructions (e.g., JNZ, JZ, JNC, JC) are used to perform conditional jumps based on the status of flags (e.g., Zero, Carry).
   - These instructions are useful for creating loops. You can check a condition using a conditional jump and keep jumping back to the start of the loop until the condition is satisfied.
3. **Counters and Iterations:**
   - You can use a combination of instructions, such as INR (increment) or DCR (decrement), to create counters for loops or iterations.
   - By incrementing or decrementing a register or memory location within a loop, you can create precise loops with a specified number of iterations.
4. **Stack Usage (PUSH and POP):**
   - The PUSH instruction is used to push data onto the stack, and the POP instruction is used to pop data from the stack.
   - The stack is often used to save and restore register values in subroutines. It's crucial for maintaining the program state.
5. **I/O Operations:** The IN and OUT instructions allow the microprocessor to communicate with input and output devices. We can use these instructions to interact with external devices. Proper I/O handling is essential for many real-world applications.

6. **Flags and Condition Testing:** We can use instructions like CMP (compare) to set condition flags without altering accumulator. These flags are useful for conditional execution & decision-making in our program.
7. **Special Function Registers (SFRs):** The 8085 has special function registers that can be used for specific tasks, such as controlling interrupts, enabling or disabling certain features, or managing I/O operations.
8. **Immediate Data Manipulation (MVI):** The MVI instruction allows you to load immediate data into a register or memory location. This is useful for setting up initial values or constants.
9. **Data Exchange (XCHG):** The XCHG instruction is used to exchange the contents of the HL register pair with the DE register pair. It's useful for data shuffling.

Counters and Time Delays:

Counters:

Counters are used to keep track of events or iterations. In 8085 assembly programming, you can create counters using registers or memory locations. Here's a basic example of using a register (B or C) as a counter:

```
; Initialize counter (B or C) to a specific value
MVI B, 0FFh  ; Initialize counter B to 255 (hexadecimal FF)

LOOP:
  ; Your loop body here

  ; Decrement the counter
  DCR B      ; Decrement the value of counter B

  ; Check if the counter has reached zero
  JNZ LOOP   ; Jump back to the LOOP label if counter B is not zero
```

Time Delays:

Time delays are essential when you need to create specific timing intervals in your program. You can create time delays by executing a series of instructions that take a known amount of time to execute. For this purpose, you can use loops and the NOP (No Operation) instruction, which takes one machine cycle to execute. The number of NOP instructions in the loop determines the delay duration.

```
; Time delay loop
DELAY:
  NOP       ; One NOP instruction (1 machine cycle delay)
  DCR B     ; Decrement a counter (using register B as the counter)
  JNZ DELAY ; Jump back to DELAY if the counter (B) is not zero
```

Stack and Subroutines in 8085:

1. Stack:

- The stack is a special region in memory used for temporary data storage. It operates as a "Last-In, First-Out" (LIFO) data structure, meaning that the most recently pushed item is the first to be popped.
- The 8085 microprocessor uses a dedicated stack pointer register (SP) to keep track of the top of the stack. The stack grows downward in memory, meaning that as items are pushed onto the stack, the stack pointer is decremented to point to the new top of the stack.
- **PUSH:** Pushes the contents of a register pair (e.g., BC, DE, HL) onto the stack. It decrements the stack pointer twice (for the high byte and low byte) and stores the data in memory.
- **POP:** Pops the top two bytes of the stack into a register pair. It retrieves the data from memory, increments the stack pointer twice, and loads the data into the specified registers.

2. Subroutines:

- Subroutines are reusable sections of code that can be called from different parts of a program. They are particularly useful for modular programming, reducing redundancy, and improving code maintainability. Subroutines can take parameters (inputs) and return results (outputs).
- **CALL:** The CALL instruction is used to call a subroutine. It pushes the return address (address of the instruction after CALL) onto the stack and transfers control to the specified memory address.
- **RET:** The RET (return) instruction is used within a subroutine to return to the main program. It pops the return address from the stack and transfers control to that address.