

Microprocessor & Microcontroller Systems (1333202) - Winter 2023 Solution

Milav Dabgar

January 16, 2024

Question 1(a) [3 marks]

Define Microprocessor.

Solution

A microprocessor is a single-chip CPU that contains all the arithmetic, logic, and control circuitry required to perform the functions of a digital computer's central processing unit.

Table 1. Microprocessor Key Features

Feature	Description
Single Chip	Complete CPU on one integrated circuit
Processing Unit	Executes instructions and performs calculations
Control Logic	Manages system operations and data flow

- **Central Processing Unit:** Core component that executes instructions
- **Integrated Circuit:** All functions combined on single silicon chip
- **Programmable Device:** Can execute different programs based on stored instructions

Mnemonic

“Single Chip CPU = Smart Computer Processor Unit”

Question 1(b) [4 marks]

Explain Flag register of microprocessor.

Solution

The Flag register stores status information about the result of arithmetic and logical operations performed by the ALU.

Table 2. 8085 Flag Register Bits

Flag	Position	Purpose
S (Sign)	Bit 7	Indicates sign of result (1=negative, 0=positive)
Z (Zero)	Bit 6	Set when result is zero
AC (Auxiliary Carry)	Bit 4	Carry from bit 3 to bit 4
P (Parity)	Bit 2	Even parity flag
CY (Carry)	Bit 0	Carry from MSB

- **Status Indicator:** Shows condition of last operation result

- **Conditional Instructions:** Used for branching and decision making
- **5 Active Flags:** Sign, Zero, Auxiliary Carry, Parity, and Carry flags

Mnemonic

“Flags Show Zero, Sign, Parity, Auxiliary, Carry”

Question 1(c) [7 marks]

Explain format of instruction of microprocessor with example.

Solution

Microprocessor instructions consist of opcode and operand fields that specify the operation and data locations.

Table 3. 8085 Instruction Format Types

Format	Size	Structure	Example
1-Byte	8 bits	Opcode only	MOV A,B
2-Byte	16 bits	Opcode + 8-bit data	MVI A,05H
3-Byte	24 bits	Opcode + 16-bit address	LDA 2000H

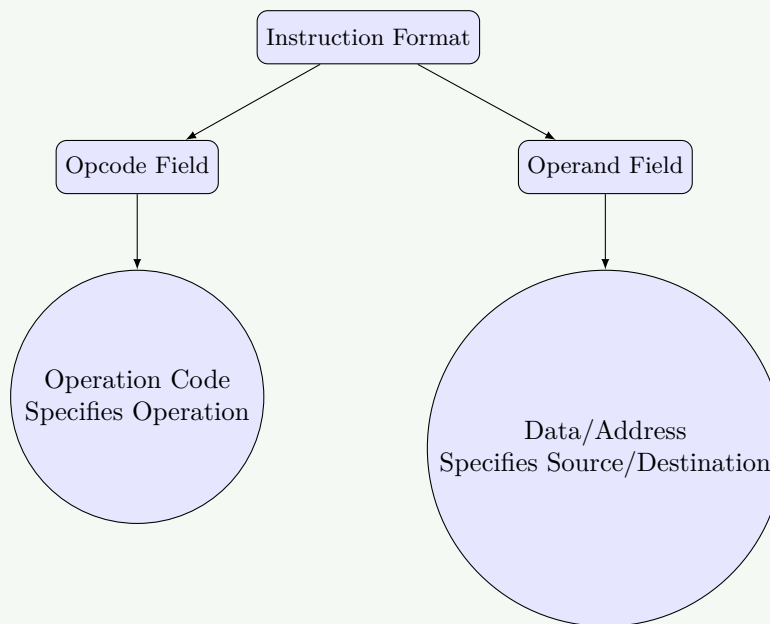


Figure 1. Instruction Format Structure

- **Opcode Field:** Defines the operation to be performed (ADD, MOV, JMP)
- **Operand Field:** Contains data, register, or memory address information
- **Variable Length:** Instructions can be 1, 2, or 3 bytes long
- **Addressing Modes:** Different ways to specify operand location

Mnemonic

“Opcode Operations + Operand Objects = Complete Commands”

Question 1(c OR) [7 marks]

Explain function of ALU, Control Unit and CPU of Microprocessor.

Solution

The CPU consists of three main functional units that work together to execute instructions.

Table 4. CPU Components and Functions

Component	Primary Function	Key Operations
ALU	Arithmetic & Logic Operations	ADD, SUB, AND, OR, XOR
Control Unit	Instruction Control	Fetch, Decode, Execute
CPU	Overall Processing	Coordinate all operations

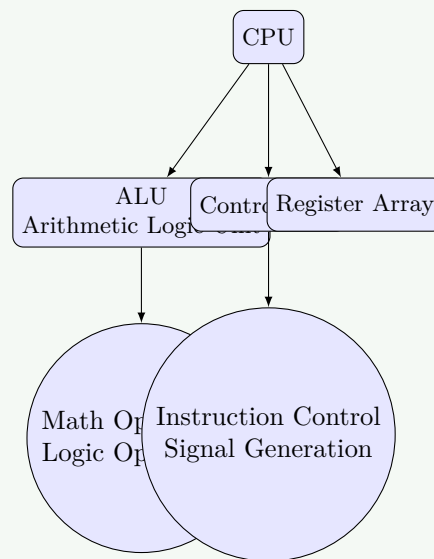


Figure 2. CPU Components

- **ALU Functions:** Performs all arithmetic calculations and logical operations
- **Control Unit Tasks:** Manages instruction execution cycle and generates control signals
- **CPU Coordination:** Integrates ALU and Control Unit for complete processing

Mnemonic

“ALU Adds, Control Commands, CPU Coordinates”

Question 2(a) [3 marks]

Explain function of ALE signal with diagram.

Solution

ALE (Address Latch Enable) signal is used to demultiplex the lower-order address and data lines.

Table 5. ALE Signal Functions

Function	Description
Address Latching	Captures lower 8-bit address
Demultiplexing	Separates address from data
Timing Control	Provides timing reference

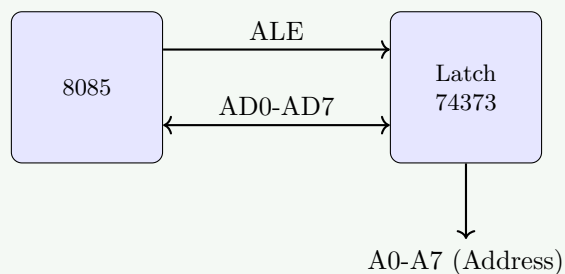


Figure 3. ALE and Demultiplexing

- **Active High Signal:** ALE goes high during T1 state
- **External Latching:** Used with 74373 latch to hold address
- **System Timing:** Provides reference for external devices

Mnemonic

“ALE Always Latches External Addresses”

Question 2(b) [4 marks]

Compare microprocessor and microcontroller

Solution

Table 6. Microprocessor vs Microcontroller Comparison

Parameter	Microprocessor	Microcontroller
Design	General purpose	Application specific
Memory	External RAM/ROM	Internal RAM/ROM
I/O Ports	External interface	Built-in I/O ports
Timers	External	Built-in timers
Cost	Higher system cost	Lower system cost
Power	Higher consumption	Lower consumption

- **Integration Level:** Microcontroller has more integrated components
- **Application Focus:** Microprocessor for computing, microcontroller for control
- **System Complexity:** Microprocessor needs more external components
- **Design Flexibility:** Microprocessor offers more expandability

Mnemonic

“Microprocessor = More Power, Microcontroller = More Control”

Question 2(c) [7 marks]

Draw & explain block diagram of microprocessor.

Solution

The 8085 microprocessor consists of several functional blocks that work together.

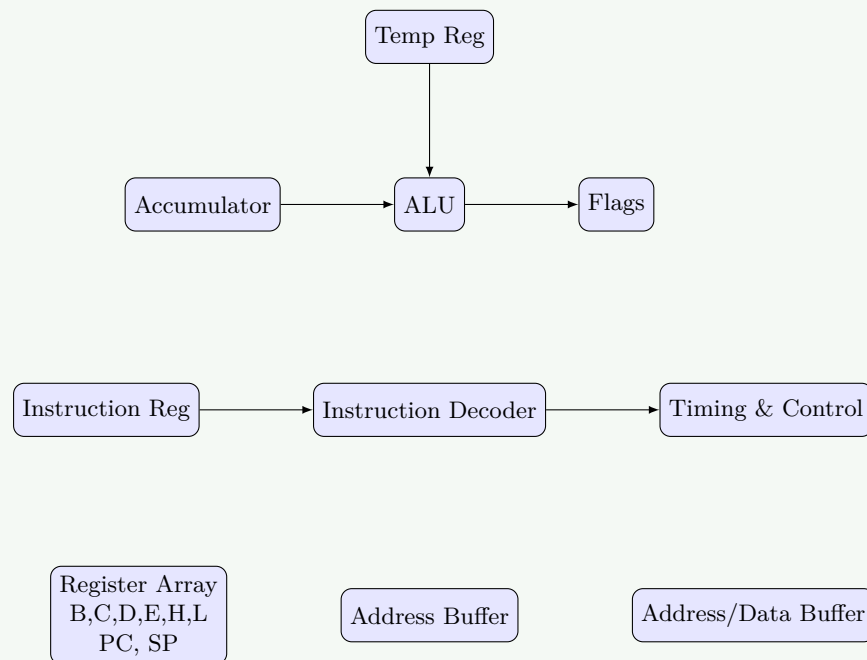


Figure 4. 8085 Architecture Block Diagram

Table 7. Block Functions

Block	Function
ALU	Arithmetic and logical operations
Register Array	Temporary data storage (B,C,D,E,H,L)
Control Unit	Instruction execution control
Address Buffer	Drive address bus lines

- **Data Path:** Information flows between registers through internal bus
- **Control Signals:** Generated by timing and control unit
- **Bus Interface:** Connects to external memory and I/O devices
- **Register Operations:** Temporary storage for operands and results

Mnemonic

“Blocks Build Better Processing Systems”

Question 2(a OR) [3 marks]

Explain 16 bits registers of microprocessor.

Solution

The 8085 has three 16-bit registers formed by combining 8-bit register pairs.

Table 8. 16-bit Registers

Register	Formation	Purpose
PC	Single 16-bit	Program Counter - next instruction address
SP	Single 16-bit	Stack Pointer - top of stack address
HL	H + L registers	Memory pointer - data address

- **Program Counter:** Automatically increments to next instruction
- **Stack Pointer:** Points to last pushed data on stack
- **HL Pair:** Most frequently used for memory addressing

Mnemonic

“PC Points Program, SP Stacks Properly, HL Holds Location”

Question 2(b OR) [4 marks]

Explain de-multiplexing lower order address and data lines with diagram of microprocessor.

Solution

The 8085 multiplexes lower 8-bit address with data lines to reduce pin count.

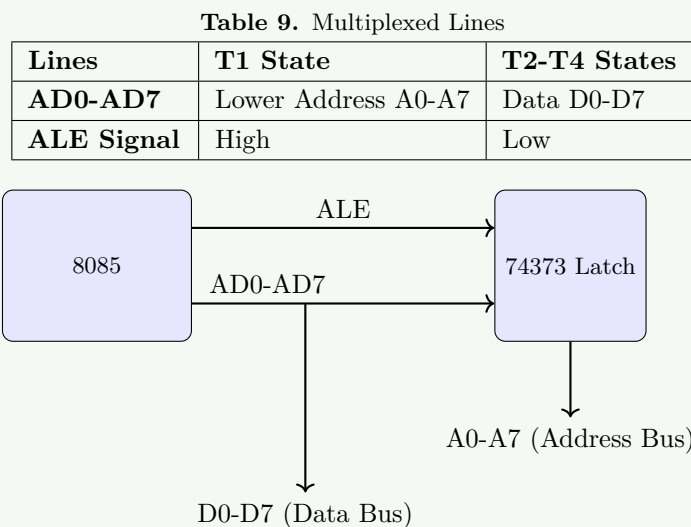


Figure 5. De-multiplexing Circuit

- **Time Division:** Same lines carry address then data
- **External Latch:** 74373 captures address when ALE is high
- **Signal Separation:** Creates separate address and data buses

Mnemonic

“ALE Always Latches External Address Elegantly”

Question 2(c OR) [7 marks]

Draw and explain pin diagram of 8085.

Solution

The 8085 is a 40-pin microprocessor with multiplexed address/data bus.

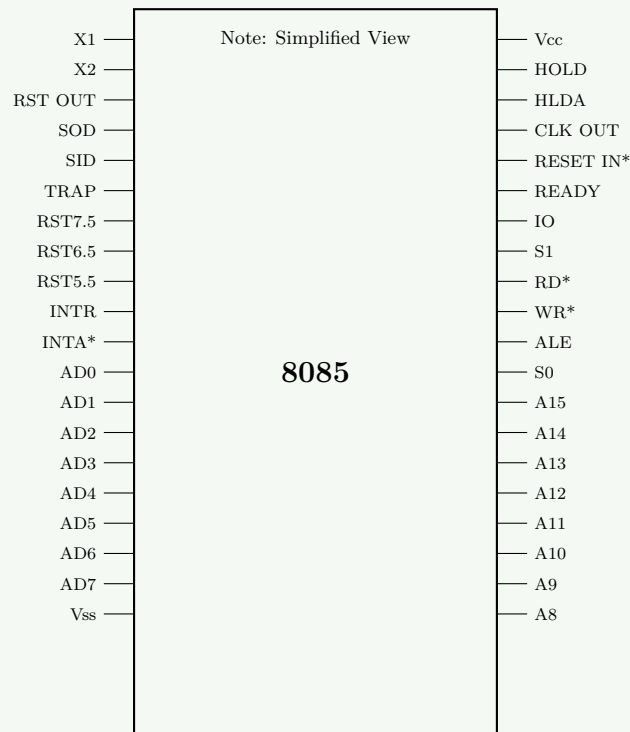


Figure 6. 8085 Pin Diagram

Table 10. Pin Groups

Group	Pins	Function
Address/Data	AD0-AD7, A8-A15	Memory addressing and data transfer
Control	ALE, RD*, WR*, IO/M*	Bus control signals
Interrupts	INTR, RST7-RST5, TRAP	Interrupt handling
Power	Vcc, Vss	Power supply connections

- **Multiplexed Bus:** AD0-AD7 carry both address and data
- **Active Low Signals:** Signals with * are active low
- **Crystal Connections:** X1, X2 for clock generation

Mnemonic

“Forty Pins Provide Perfect Processing Power”

Question 3(a) [3 marks]

Draw clock and reset circuit of microcontroller

Solution

The 8051 requires external clock and reset circuits for proper operation.

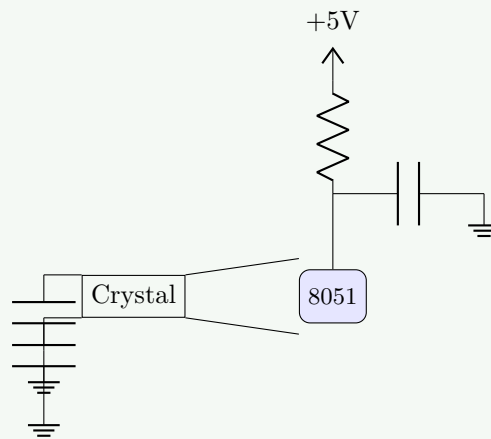


Figure 7. Clock and Reset Circuits

Table 11. Circuit Components

Component	Value	Purpose
Crystal	11.0592 MHz	Clock generation
Capacitors	30pF each	Crystal stabilization
Reset Resistor	10KΩ	Pull-up for reset
Reset Capacitor	10μF	Power-on reset delay

- **Clock Frequency:** Commonly 11.0592 MHz for serial communication
- **Reset Duration:** Must be high for at least 2 machine cycles
- **Power-on Reset:** Automatic reset when power is applied

Mnemonic

“Crystals Create Clock, Resistors Reset Reliably”

Question 3(b) [4 marks]

Explain internal RAM of 8051.

Solution

The 8051 contains 256 bytes of internal RAM organized in different sections.

Table 12. Internal RAM Organization

Address Range	Size	Purpose
00H-1FH	32 bytes	Register Banks (4 banks × 8 registers)
20H-2FH	16 bytes	Bit-addressable area
30H-7FH	80 bytes	General purpose RAM
80H-FFH	128 bytes	Special Function Registers (SFRs)

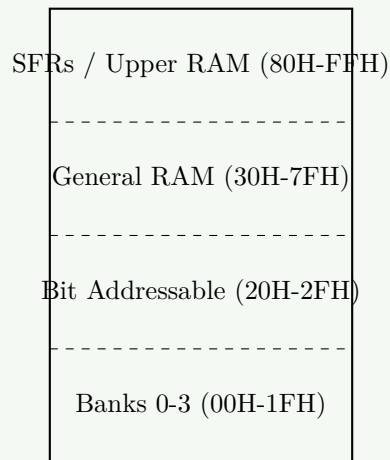


Figure 8. Internal RAM Map

- **Register Banks:** Four banks of 8 registers each (R0-R7)
- **Bit Addressing:** Individual bits can be addressed in 20H-2FH area
- **Stack Area:** Usually located in general purpose RAM area
- **Direct Access:** All locations accessible through direct addressing

Mnemonic

“RAM Registers, Bits, General, Special Functions”

Question 3(c) [7 marks]

Explain block diagram of 8051.

Solution

The 8051 microcontroller integrates CPU, memory, and I/O on a single chip.

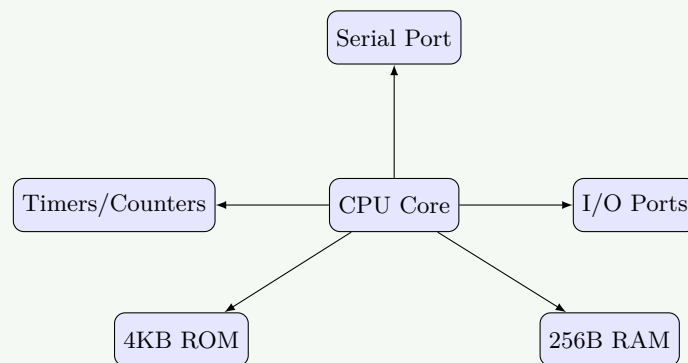


Figure 9. 8051 Block Diagram

Table 13. Major Blocks

Block	Function
CPU	Instruction execution and control
Memory	4KB ROM + 256B RAM
Timers	Two 16-bit timer/counters
I/O Ports	Four 8-bit bidirectional ports
Serial Port	Full-duplex UART
Interrupts	5-source interrupt system

- **Harvard Architecture:** Separate program and data memory spaces
- **Built-in Peripherals:** Timers, serial port, interrupts integrated
- **Expandable:** External memory and I/O can be added
- **Control Applications:** Optimized for embedded control tasks

Mnemonic

“Complete Control Chip Contains CPU, Memory, I/O”

Question 3(a OR) [3 marks]

Explain function of DPTR and PC.

Solution

DPTR and PC are important 16-bit registers in 8051 for memory addressing.

Table 14. DPTR and PC Functions

Register	Full Form	Function
DPTR	Data Pointer	Points to external data memory
PC	Program Counter	Points to next instruction address

- **DPTR Usage:** Accessing external RAM and lookup tables
- **PC Function:** Automatically increments after instruction fetch
- **16-bit Addressing:** Both can address 64KB memory space

Mnemonic

“DPTR Data Pointer, PC Program Counter”

Question 3(b OR) [4 marks]

Explain different timer modes of microcontroller.

Solution

The 8051 has two timers with four different operating modes.

Table 15. Timer Modes

Mode	Configuration	Purpose
Mode 0	13-bit timer	Compatible with 8048
Mode 1	16-bit timer	Maximum count capability
Mode 2	8-bit auto-reload	Constant time intervals
Mode 3	Two 8-bit timers	Timer 0 split operation

- **Mode Selection:** Controlled by TMOD register bits
- **Timer 0/1:** Both timers support modes 0, 1, 2
- **Mode 3 Special:** Only Timer 0 can operate in mode 3
- **Applications:** Delays, baud rate generation, event counting

Mnemonic

“Modes Make Timers Tremendously Versatile”

Question 3(c OR) [7 marks]

Explain interrupts of microcontroller.

Solution

The 8051 has a 5-source interrupt system for handling external events.

Table 16. 8051 Interrupt Sources

Interrupt	Vector Address	Priority	Trigger
Reset	0000H	Highest	Power-on/External
External 0	0003H	High	INT0 pin
Timer 0	000BH	Medium	Timer 0 overflow
External 1	0013H	Medium	INT1 pin
Timer 1	001BH	Low	Timer 1 overflow
Serial	0023H	Lowest	Serial communication

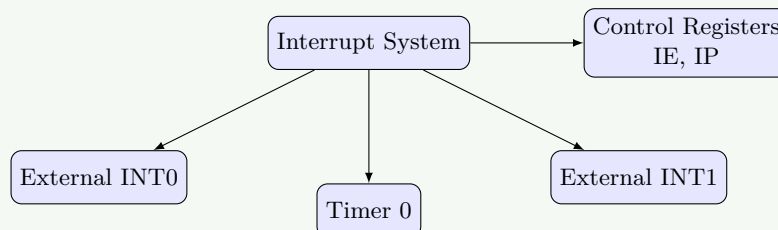


Figure 10. Interrupt System Structure

- **Interrupt Enable:** IE register controls individual interrupt enables
- **Priority Control:** IP register sets interrupt priorities
- **Vector Addresses:** Each interrupt has fixed vector location
- **Nested Interrupts:** Higher priority can interrupt lower priority

Mnemonic

“Five Interrupt Sources Serve System Efficiently”

Question 4(a) [3 marks]

Explain data transfer instruction with example for 8051.

Solution

Data transfer instructions move data between registers, memory, and I/O ports.

Table 17. Data Transfer Instructions

Instruction	Example	Function
MOV	MOV A, #55H	Move immediate data to accumulator
MOVX	MOVX A, @DPTR	Move external RAM to accumulator
MOVC	MOVC A, @A+PC	Move code memory to accumulator

- **MOV Variants:** Register to register, immediate to register
- **External Access:** MOVX for external RAM operations
- **Code Access:** MOVC for reading program memory tables

Mnemonic

“MOV Moves data, MOVX eXternal, MOVC Code”

Question 4(b) [4 marks]

List and explain different addressing modes of microcontroller.

Solution

The 8051 supports several addressing modes for flexible data access.

Table 18. 8051 Addressing Modes

Mode	Example	Description
Immediate	MOV A, #55H	Data specified in instruction
Register	MOV A, R0	Use register contents
Direct	MOV A, 30H	Direct memory address
Indirect	MOV A, @R0	Address stored in register
Indexed	MOVC A, @A+DPTR	Base address plus offset

- **Immediate Mode:** Constant data included in instruction
- **Register Mode:** Fastest execution using register file
- **Direct Mode:** Access any internal RAM location
- **Indexed Mode:** Table lookup and array access

Mnemonic

“Immediate, Register, Direct, Indirect, Indexed Addressing”

Question 4(c) [7 marks]

Write a program to copy block of 8 data starting from location 100h to 200h.

Solution

Listing 1. Block Transfer Program

```

1  ORG 0000H      ; Start address
2  MOV R0,#100H   ; Source address pointer
3  MOV R1,#200H   ; Destination address pointer
4  MOV R2,#08H    ; Counter for 8 bytes
5
6  LOOP:
7  MOV A,@R0      ; Read data from source
8  MOV @R1,A      ; Write data to destination
9  INC R0         ; Increment source pointer
10 INC R1         ; Increment destination pointer
11 DJNZ R2,LOOP   ; Decrement counter and jump if not zero
12
13 END            ; End of program

```

Table 19. Register Usage

Register	Purpose
R0	Source address pointer (100H)
R1	Destination address pointer (200H)
R2	Loop counter (8 bytes)
A	Temporary data storage

- **Indirect Addressing:** @R0 and @R1 for memory access
- **Loop Control:** DJNZ instruction decrements and tests
- **Block Transfer:** Copies 8 consecutive bytes efficiently

Mnemonic

“Read, Write, Increment, Decrement, Jump Loop”

Question 4(a OR) [3 marks]

Write a program to add two bytes of data and store result in R0 register.

Solution

Listing 2. Simple Addition Program

```

1  ORG 0000H      ; Start address
2  MOV A,#25H     ; Load first byte
3  ADD A,#35H     ; Add second byte
4  MOV R0,A       ; Store result in R0
5  END           ; End program

```

Table 20. Operation Steps

Step	Instruction	Result
1	MOV A,#25H	A = 25H
2	ADD A,#35H	A = 5AH
3	MOV R0,A	R0 = 5AH

- **Addition Result:** $25H + 35H = 5AH$
- **Flag Effects:** Carry flag set if result $> FFH$

Mnemonic

“Move, Add, Move = Simple Addition”

Question 4(b OR) [4 marks]

Explain indexed addressing mode with example.

Solution

Indexed addressing uses a base address plus an offset for memory access.

Table 21. Indexed Addressing Details

Component	Description	Example
Base Address	DPTR or PC register	DPTR = 1000H
Index	Accumulator contents	A = 05H
Effective Address	Base + Index	$1000H + 05H = 1005H$

Listing 3. Indexed Addressing Example

```

1 MOV DPTR,#1000H    ; Base address
2 MOV A,#05H         ; Index value
3 MOVC A,@A+DPTR     ; Read from address 1005H

```

- **Table Access:** Ideal for lookup tables and arrays
- **Program Memory:** MOVC reads from code memory only
- **Dynamic Indexing:** Index can change during execution

Mnemonic

“Base + Index = Dynamic Access”

Question 4(c OR) [7 marks]

Explain stack operation of microcontroller, PUSH and POP instruction.

Solution

The stack is a LIFO memory structure used for temporary data storage.

Table 22. Stack Operations

Operation	Instruction	Function
PUSH	PUSH 30H	Store data on stack
POP	POP 30H	Retrieve data from stack
Stack Pointer	SP register	Points to top of stack

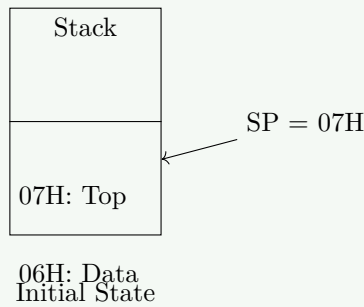


Figure 11. Stack Structure

Listing 4. Stack Operation Example

```

1  MOV SP,#30H      ; Initialize stack pointer
2  PUSH ACC         ; Save accumulator
3  PUSH B           ; Save B register
4  POP B            ; Restore B register
5  POP ACC          ; Restore accumulator

```

- **LIFO Structure:** Last In, First Out data organization
- **SP Auto-increment:** Stack pointer automatically adjusts
- **Subroutine Calls:** Stack saves return addresses
- **Register Preservation:** Save/restore register contents

Mnemonic

“PUSH Puts Up, Stack Holds, POP Pulls Out”

Question 5(a) [3 marks]

Explain branching instruction with example.

Solution

Branching instructions alter program flow based on conditions or unconditionally.

Table 23. Branching Instructions

Type	Instruction	Example
Unconditional	LJMP address	LJMP 2000H
Conditional	JZ address	JZ ZERO_LABEL
Call/Return	LCALL address	LCALL SUBROUTINE

Listing 5. Branching Example

```

1  MOV A,#00H      ; Load zero
2  JZ ZERO_FOUND   ; Jump if A is zero
3  LJMP CONTINUE   ; Jump to continue
4  ZERO_FOUND:
5      MOV RO,#01H  ; Set flag
6  CONTINUE:
7      NOP          ; Continue execution

```

- **Program Control:** Changes execution sequence

- **Conditional Jumps:** Based on flag register status
- **Address Range:** Can jump to any program memory location

Mnemonic

“Jump Changes Control Flow”

Question 5(b) [4 marks]

Interface 8 leds with microcontroller and write a program to turn on and off.

Solution

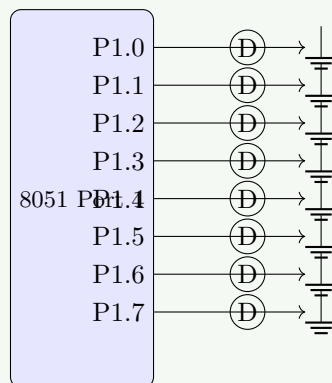


Figure 12. LED Interfacing Circuit

Listing 6. LED Blink Program

```

1  ORG 0000H
2  MAIN:
3      MOV P1,#0FFH      ; Turn ON all LEDs
4      CALL DELAY        ; Wait
5      MOV P1,#00H      ; Turn OFF all LEDs
6      CALL DELAY        ; Wait
7      SJMP MAIN         ; Repeat
8
9  DELAY:
10     MOV R0,#0FFH      ; Outer loop counter
11     LOOP1:
12         MOV R1,#0FFH  ; Inner loop counter
13     LOOP2:
14         DJNZ R1,LOOP2  ; Inner delay loop
15         DJNZ R0,LOOP1  ; Outer delay loop
16         RET           ; Return
17     END

```

- **Current Limiting:** Resistors protect LEDs from high current
- **Logic High:** Writing 1 to port pin turns ON LED (if wired to ground)
- **Delay Loop:** Nested loops create visible blink delay

Question 5(c) [7 marks]

Interface LCD with microcontroller and write a program to display "welcome".

Solution

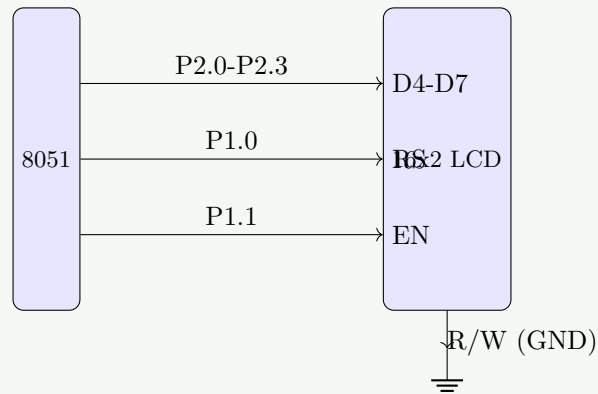


Figure 13. LCD Interface Connections

Listing 7. LCD Display Program

```

1  ORG 0000H
2      CALL LCD_INIT      ; Initialize LCD
3      CALL DISPLAY_MSG   ; Display message
4      SJMP $             ; Stop here
5
6  LCD_INIT:
7      MOV P2,#38H        ; Function set: 8-bit, 2-line
8      CALL COMMAND
9      MOV P2,#0EH        ; Display ON, Cursor ON
10     CALL COMMAND
11     MOV P2,#01H        ; Clear display
12     CALL COMMAND
13     MOV P2,#06H        ; Entry mode set
14     CALL COMMAND
15     RET
16
17  DISPLAY_MSG:
18     MOV DPTR,#MESSAGE   ; Point to message
19  NEXT_CHAR:
20     CLR A
21     MOVC A,@A+DPTR      ; Read character
22     JZ DONE             ; If zero, end of string
23     CALL SEND_CHAR      ; Send character to LCD
24     INC DPTR            ; Next character
25     SJMP NEXT_CHAR
26  DONE:
27     RET
28
29  COMMAND:
30     CLR P1.0            ; RS = 0 for command
31     SETB P1.1           ; EN = 1
32     CLR P1.1           ; EN = 0 (pulse)
33     CALL DELAY
34     RET
35
36  SEND_CHAR:
37     MOV P2,A            ; Put character on data lines
38     SETB P1.0           ; RS = 1 for data
39     SETB P1.1           ; EN = 1
40     CLR P1.1           ; EN = 0 (pulse)
41     CALL DELAY
42     RET
43

```

```

44 DELAY:
45     MOV R0,#50          ; Delay routine
46 DELAY_LOOP:
47     MOV R1,#255
48 DELAY_INNER:
49     DJNZ R1,DELAY_INNER
50     DJNZ R0,DELAY_LOOP
51     RET
52
53 MESSAGE:
54     DB "WELCOME",0      ; Message string with null terminator
55 END

```

Table 24. LCD Interface Pins

8051 Pin	LCD Pin	Function
P2.0-P2.3	D4-D7	4-bit data lines
P1.0	RS	Register select (0=command, 1=data)
P1.1	EN	Enable pulse
GND	R/W	Read/Write (tied to ground for write)

- **4-bit Mode:** Uses only upper 4 data lines to save pins
- **Control Signals:** RS selects command/data, EN provides timing pulse
- **Character Display:** Each character sent as ASCII code
- **Initialization:** Required command sequence for proper operation

Mnemonic

“LCD Displays Characters with Commands and Data”

Question 5(a OR) [3 marks]

Explain logical instruction with example.

Solution

Logical instructions perform bitwise operations on data, useful for manipulating individual bits.

Table 25. Logical Instructions

Instruction	Example	Function
ANL	ANL A,#0FH	Bitwise AND operation
ORL	ORL A,#F0H	Bitwise OR operation
XRL	XRL A,#FFH	Bitwise XOR operation

Listing 8. Logical Operations

```

1 MOV A,#55H          ; A = 01010101B
2 ANL A,#0FH          ; A = 00000101B (mask upper bits)
3 ORL A,#F0H          ; A = 11110101B (set upper bits)
4 XRL A,#FFH          ; A = 00001010B (complement all bits)

```

- **Bit Manipulation:** Used for setting, clearing, and testing bits
- **Masking Operations:** ANL clears unwanted bits
- **Flag Effects:** Updates parity flag based on result

Mnemonic

“AND Masks, OR Sets, XOR Toggles”

Question 5(b OR) [4 marks]

Interface 7 segment with microcontroller.

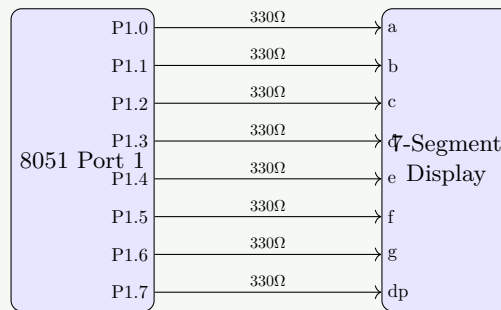
Solution

Figure 14. 7-Segment Interface

Listing 9. 0-9 Counter Program

```

1  ORG 0000H
2      MOV DPTR,#DIGIT_TABLE    ; Point to lookup table
3      MOV RO,#0                ; Start with digit 0
4
5  MAIN_LOOP:
6      MOV A,RO                 ; Get current digit
7      MOVC A,@A+DPTR           ; Get 7-segment code
8      MOV P1,A                 ; Display on 7-segment
9      CALL DELAY               ; Wait 1 second
10     INC RO                   ; Next digit
11     CJNE RO,#10,MAIN_LOOP    ; Check if reached 10
12     MOV RO,#0                ; Reset to 0
13     SJMP MAIN_LOOP           ; Repeat
14
15  DIGIT_TABLE:
16      DB 3FH, 06H, 5BH, 4FH, 66H    ; 0,1,2,3,4
17      DB 6DH, 7DH, 07H, 7FH, 6FH    ; 5,6,7,8,9
18  END

```

Table 26. 7-Segment Codes

Digit	Hex Code	Binary	Segments Lit
0	3FH	00111111	a,b,c,d,e,f
1	06H	00000110	b,c
2	5BH	01011011	a,b,g,e,d

- **Common Cathode:** Segments light when port pin is high
- **Current Limiting:** Resistors prevent segment damage
- **Lookup Table:** Efficient storage of segment patterns

Mnemonic

“Seven Segments Show Digits Clearly”

Question 5(c OR) [7 marks]

Interface LM 35 with microcontroller and explain block diagram of temperature controller.

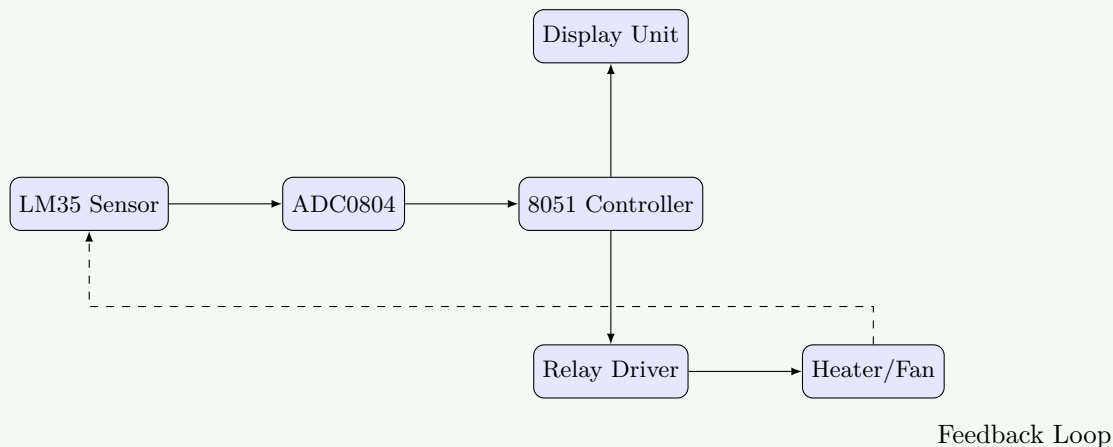
Solution

Figure 15. Temperature Controller Block Diagram

Listing 10. Temperature Control Program

```

1  ORG 0000H
2  MAIN:
3      CALL READ_TEMP      ; Read temperature from ADC
4      CALL DISPLAY_TEMP   ; Show temperature on display
5      CALL TEMP_CONTROL   ; Control heating/cooling
6      CALL DELAY          ; Wait before next reading
7      SJMP MAIN
8
9  READ_TEMP:
10     CLR P2.0             ; Start ADC conversion
11     SETB P2.0            ; Pulse to start
12     JNB P2.1,$           ; Wait for conversion complete
13     MOV A,P1             ; Read temperature data
14     RET
15
16  TEMP_CONTROL:
17     CJNE A,#30,CHECK_HIGH ; Compare with setpoint (30C)
18  CHECK_HIGH:
19     JC TEMP_LOW          ; If A < 30, temperature is low
20     SETB P3.0            ; Turn ON cooling (fan)
21     CLR P3.1             ; Turn OFF heating
22     RET
23  TEMP_LOW:
24     CLR P3.0             ; Turn OFF cooling
25     SETB P3.1            ; Turn ON heating
26     RET
27  END
  
```

Table 27. System Components

Component	Function
LM35	Temperature sensor (10mV/°C)
ADC0804	Analog to digital converter
8051	Main controller
Relay	Switch high power loads
Display	Show current temperature

- **Temperature Sensing:** LM35 provides 10mV per degree Celsius
- **ADC Conversion:** Converts analog voltage to digital value
- **Control Logic:** Compares with setpoint and controls relay
- **Feedback System:** Continuous monitoring and adjustment

Mnemonic

“Sense, Convert, Compare, Control Temperature Automatically”