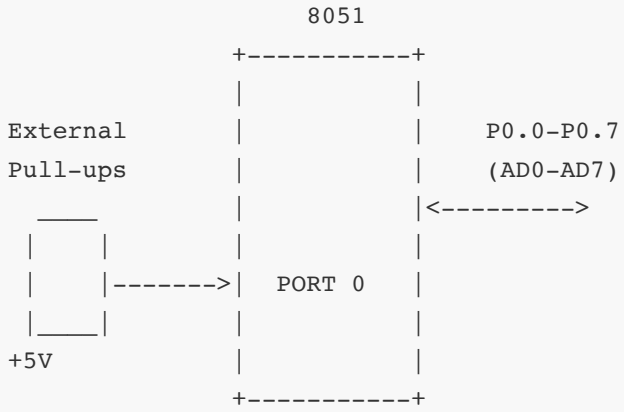
# Question 1(a) [3 marks]

**Describe any one Port Configuration of 8051 Microcontroller.**

**Answer**:

| Configuration | Description |
|---|---|
| **Port 0** | Dual-purpose port - 8-bit open drain bidirectional I/O port and multiplexed low address/data bus. External pull-up resistors required for I/O functions. |

**Diagram:**




**Mnemonic:** "PORT 0-PLAD" (Port 0 needs Pull-ups, works as Latch/Address/Data)

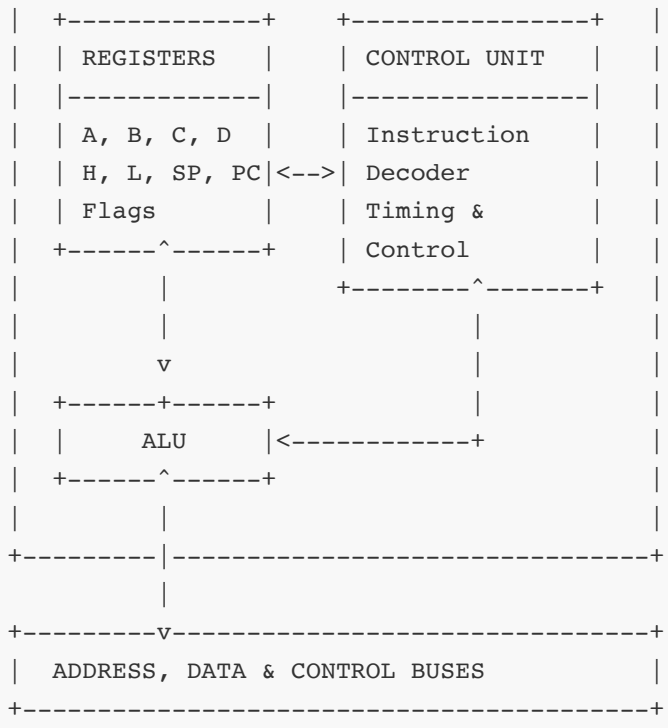
# Question 1(b) [4 marks]

**Illustrate Microprocessor Architecture.**

**Answer**:

| Component | Function |
|---|---|
| **ALU** | Performs arithmetic and logical operations |
| **Registers** | Temporary storage for data and addresses |
| **Control Unit** | Directs operation of processor and data flow |
| **Buses** | Pathways for data transfer (address, data, control) |

**Diagram:**




**Mnemonic:** "RABC" - "Registers, ALU, Buses, Control"

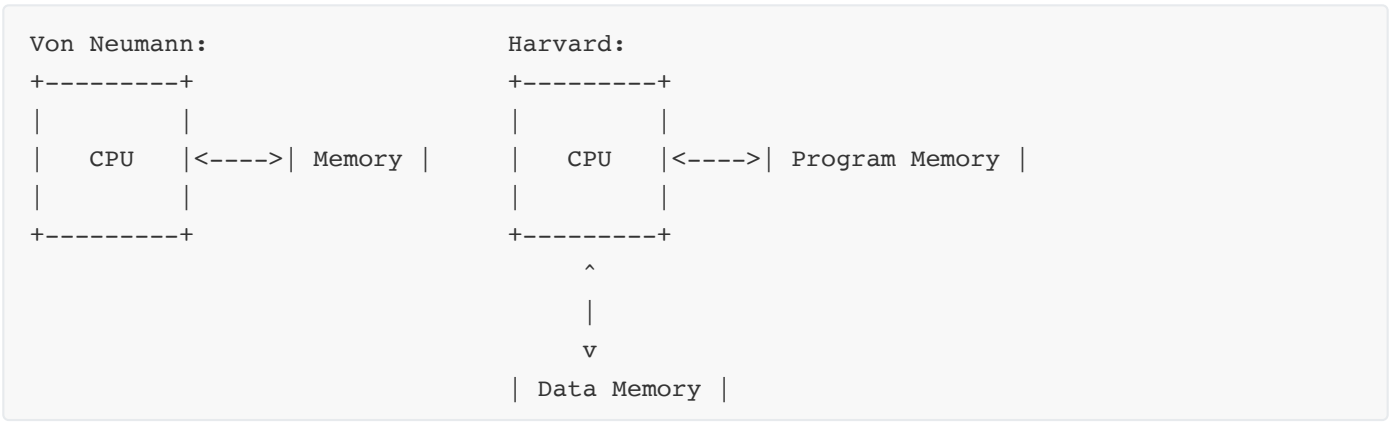
# Question 1(c) [7 marks]

**Compare Von Neumann & Harvard architecture.**

**Answer**:

| Feature | Von Neumann Architecture | Harvard Architecture |
|---|---|---|
| Memory Buses | Single memory bus for instructions and data | Separate buses for program and data memory |
| Execution | Sequential execution | Parallel fetch and execute possible |
| Speed | Slower due to bus bottleneck | Faster due to simultaneous access |
| Memory Access | Single memory space | Separate memory spaces |
| Complexity | Simpler design | More complex design |
| Applications | General-purpose computing | DSP, microcontrollers, embedded systems |
| Examples | Most PCs, 8085, 8086 | 8051, PIC, ARM Cortex-M |

**Diagram:**




**Mnemonic:** "Harvard Has Separate Streets" (Harvard Has Separate memory paths)

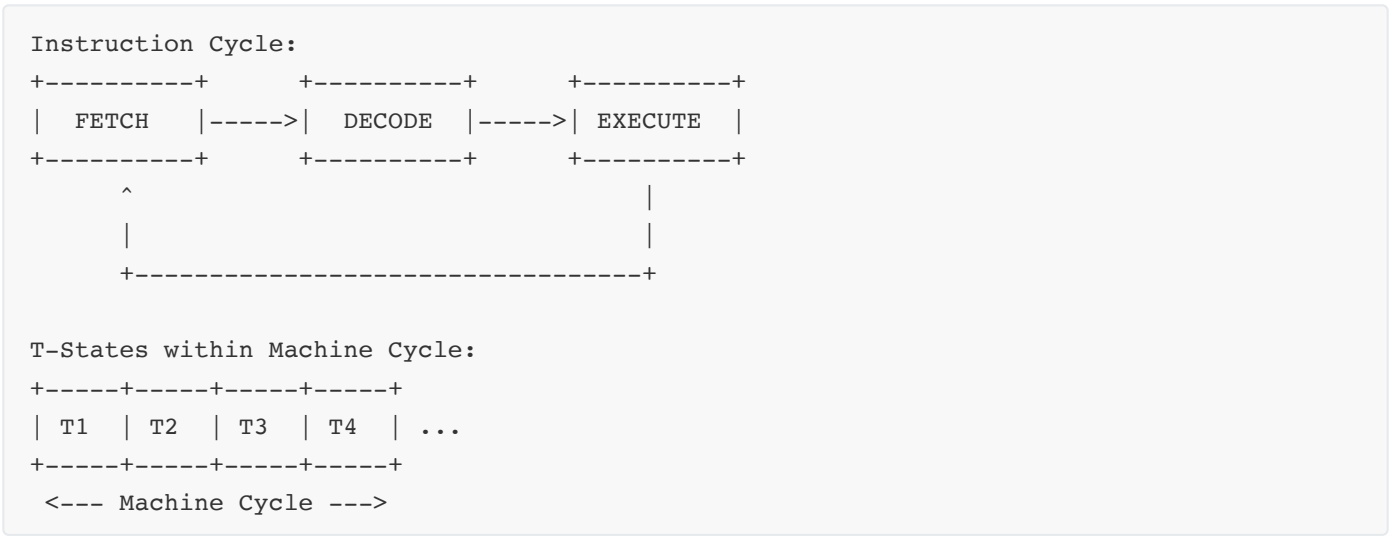
# Question 1(c OR) [7 marks]

**Define RISC, CISC, Opcode, Operand, Instruction Cycle, Machine Cycle, and T State.**

**Answer**:

| Term | Definition |
|---|---|
| **RISC** | Reduced Instruction Set Computer - architecture with simple instructions optimized for speed |
| **CISC** | Complex Instruction Set Computer - architecture with complex, powerful instructions |
| **Opcode** | Operation Code - part of instruction that specifies operation to be performed |
| **Operand** | Data value or address used in operation |
| **Instruction Cycle** | Complete process to fetch, decode and execute an instruction |
| **Machine Cycle** | Basic operation like memory read/write (subset of instruction cycle) |
| **T-State** | Time state - smallest unit of time in processor operation (clock period) |

**Diagram:**




**Mnemonic:** "RICO ITEM" (RISC, CISC, Opcode, Instruction cycle, T-state, Execute, Machine cycle)

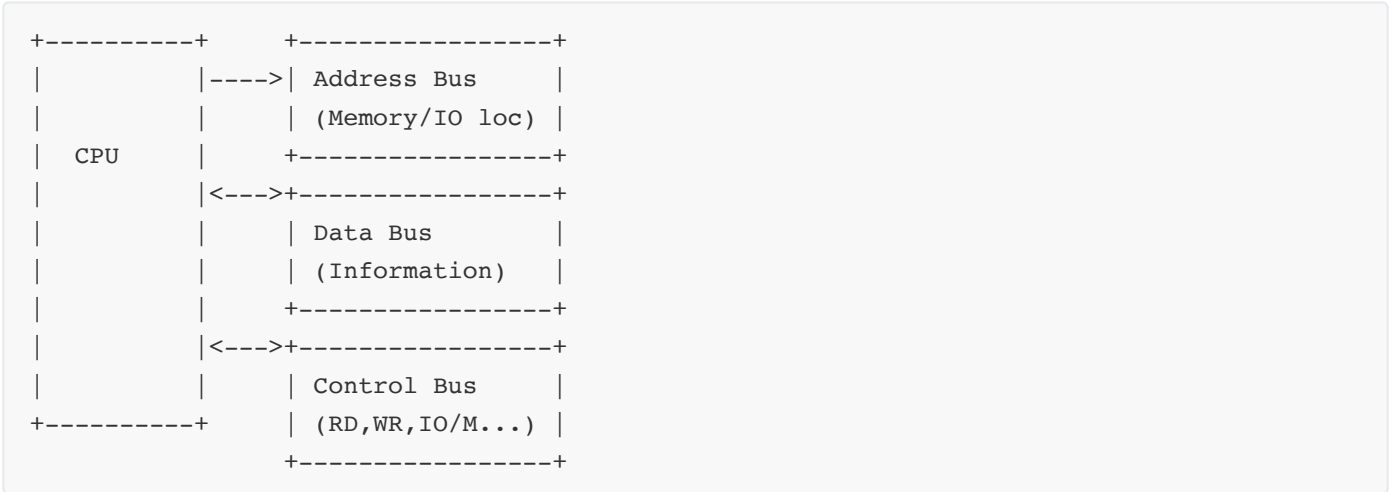
# Question 2(a) [3 marks]

**Define Data bus, Address bus and Control bus.**

**Answer**:

| Bus Type | Definition |
|---|---|
| **Data Bus** | Bidirectional pathway that transfers actual data between microprocessor and peripheral devices |
| **Address Bus** | Unidirectional pathway that carries memory/IO device locations to be accessed |
| **Control Bus** | Group of signal lines that coordinate and synchronize all system operations |

**Diagram:**




**Mnemonic:** "ADC" - "Address finds location, Data carries information, Control coordinates operations"

# Question 2(b) [4 marks]

**Compare Microprocessor and Microcontroller.**

**Answer**:

| Feature | Microprocessor | Microcontroller |
|---|---|---|
| Definition | CPU on a single chip | Complete computer system on a chip |
| Memory | External RAM/ROM needed | Built-in RAM/ROM |
| I/O Ports | Limited or none on-chip | Multiple I/O ports on-chip |
| Peripherals | External peripherals needed | Built-in peripherals (timers, ADC, etc.) |
| Applications | General computing, PCs | Embedded systems, IoT devices |
| Cost | Higher for complete system | Lower (all-in-one solution) |
| Power Consumption | Higher | Lower |

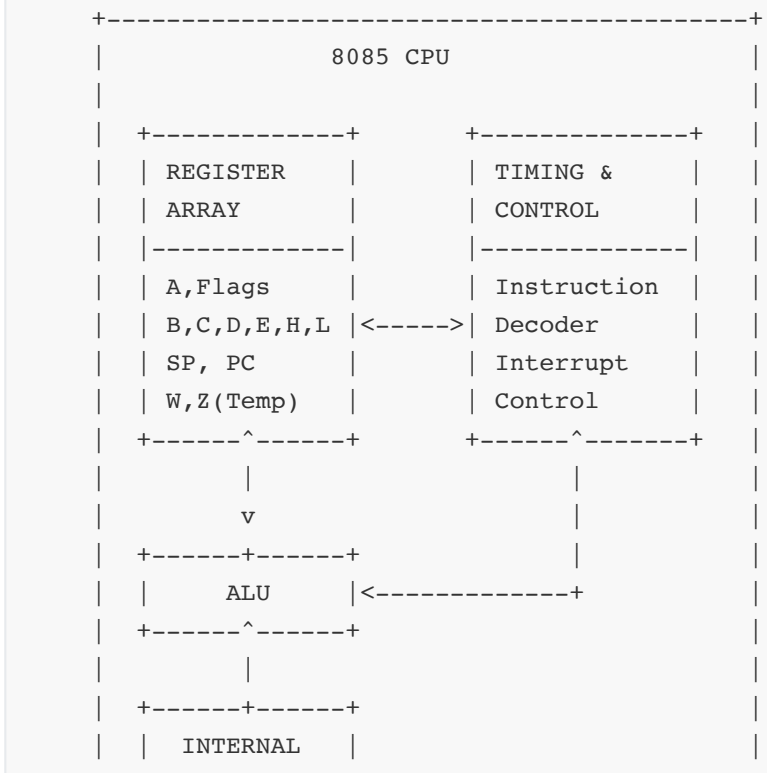
**Mnemonic:** "MEMI-CAP" (Memory external/internal, Cost, Applications, Peripherals)

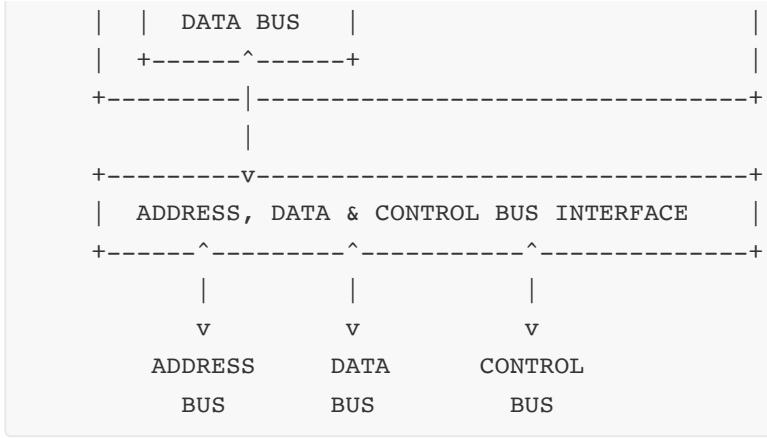
# Question 2(c) [7 marks]

**Sketch and explain 8085 block diagram.**

**Answer**:

**Diagram:**






**Main Components:**

- **Register Array**: A (Accumulator), Flags, B-L, SP, PC, temp registers
- **ALU**: Performs arithmetic and logical operations
- **Timing & Control**: Generates control signals, handles interrupts
- **Bus Interface**: Connects CPU to external devices
- **Internal Data Bus**: Links internal components

**Mnemonic:** "RATBI" - "Registers, ALU, Timing, Buses, Interface"

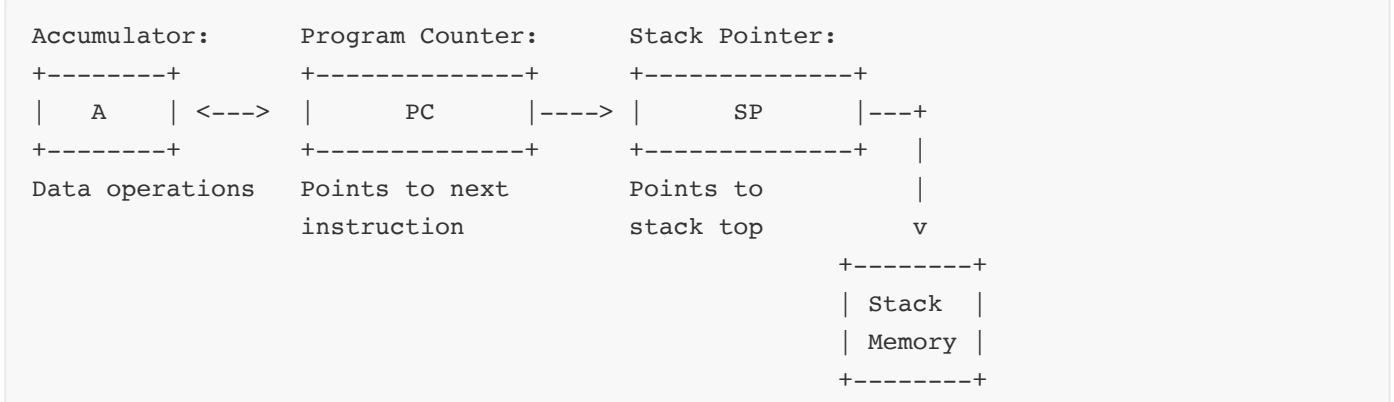
# Question 2(a OR) [3 marks]

**Explain Accumulator, Program Counter and Stack Pointer.**

**Answer**:

| Register | Function |
|---|---|
| **Accumulator (A)** | 8-bit register that stores results of arithmetic and logical operations |
| **Program Counter (PC)** | 16-bit register that holds address of next instruction to be executed |
| **Stack Pointer (SP)** | 16-bit register that points to current top of stack in memory |

**Diagram:**



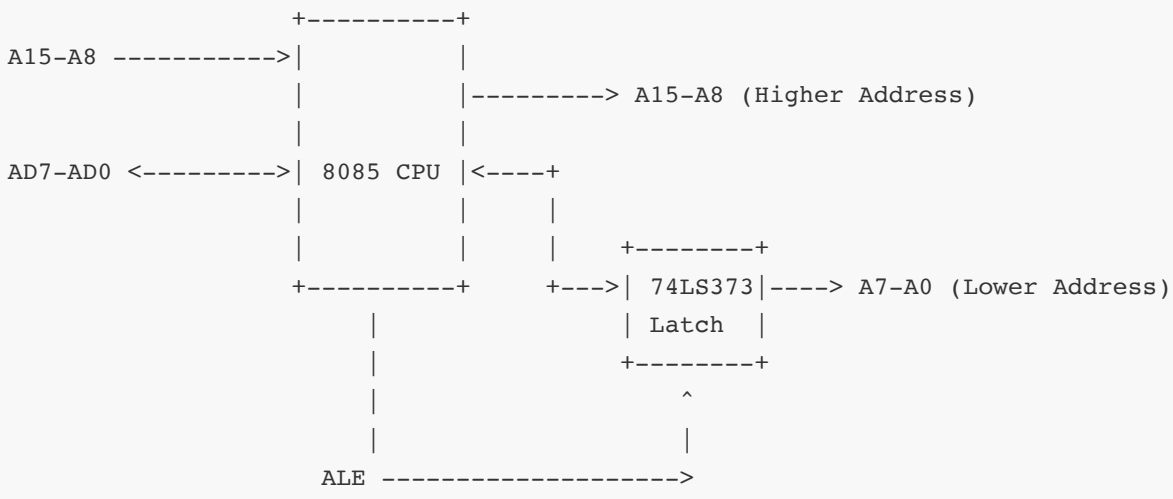
**Mnemonic:** "APS" - "Accumulator Processes, PC Predicts, SP Stacks"

# Question 2(b OR) [4 marks]

**Sketch and explain Demultiplexing of Address bus and data bus.**

**Answer**:

**Diagram:**




**Process**:

1. **Multiplexing**: AD0-AD7 pins share address and data signals to reduce pin count
2. **Demultiplexing Steps**:
   - CPU places address on AD0-AD7 pins
   - ALE (Address Latch Enable) signal goes HIGH
   - External latch (74LS373) captures lower address bits
   - ALE goes LOW, latching the address
   - AD0-AD7 pins now carry data

**Mnemonic:** "ALAD" - "ALE Active, Latch Address, After Data"

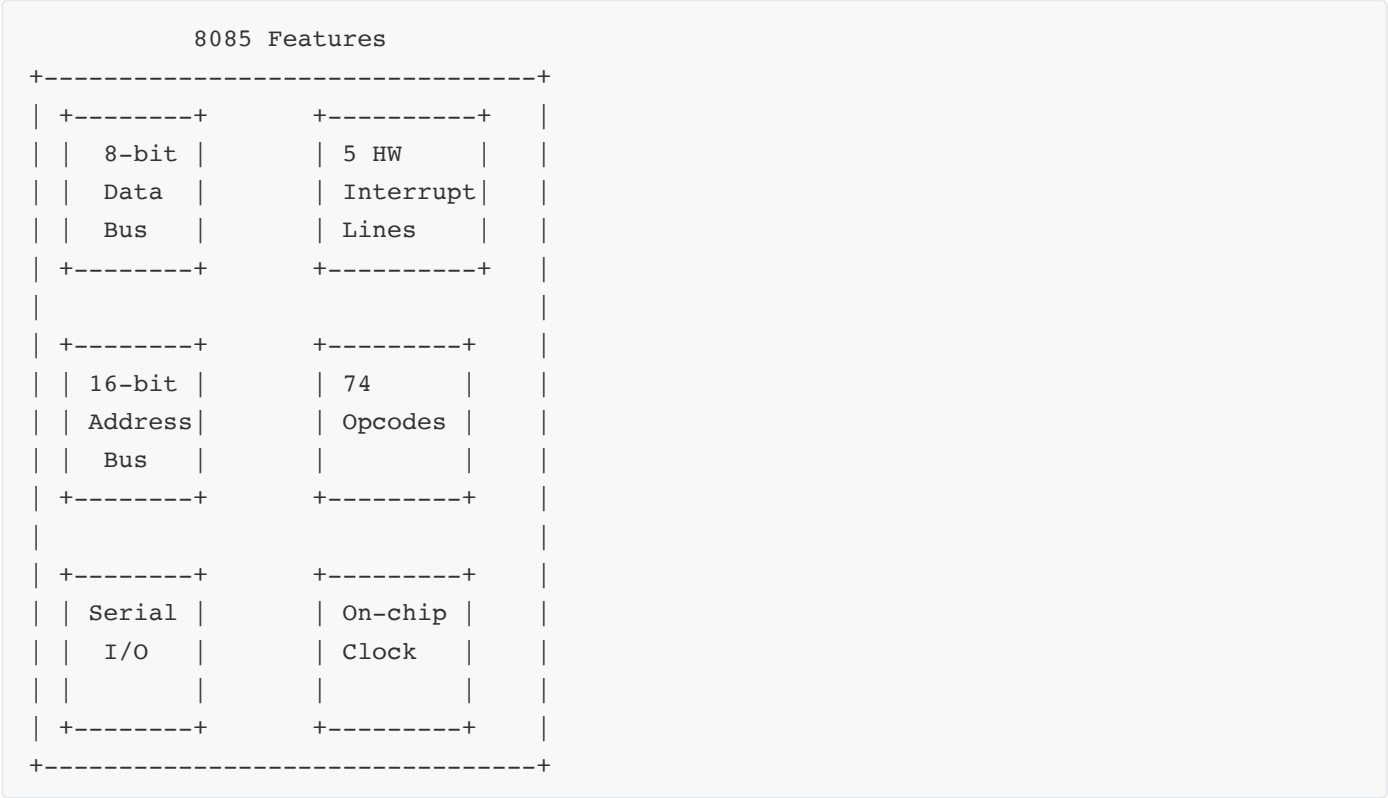
# Question 2(c OR) [7 marks]

**List any seven features of 8085.**

**Answer**:

| Feature | Description |
|---|---|
| **8-bit Data Bus** | Transfers 8 bits of data in parallel |
| **16-bit Address Bus** | Can address up to 64KB of memory (2^16) |
| **Hardware Interrupts** | 5 hardware interrupts (TRAP, RST 7.5, 6.5, 5.5, INTR) |
| **Serial I/O** | SID and SOD pins for serial communication |
| **Clock Generation** | On-chip clock generator with crystal |
| **Instruction Set** | 74 operation codes generating 246 instructions |
| **Register Set** | Six 8-bit registers (B,C,D,E,H,L), accumulator, flags, SP, PC |

**Diagram:**




**Mnemonic:** "CHAIRS" - "Clock, Hardware interrupts, Address bus, Instruction set, Registers, Serial I/O"

# Question 3(a) [3 marks]

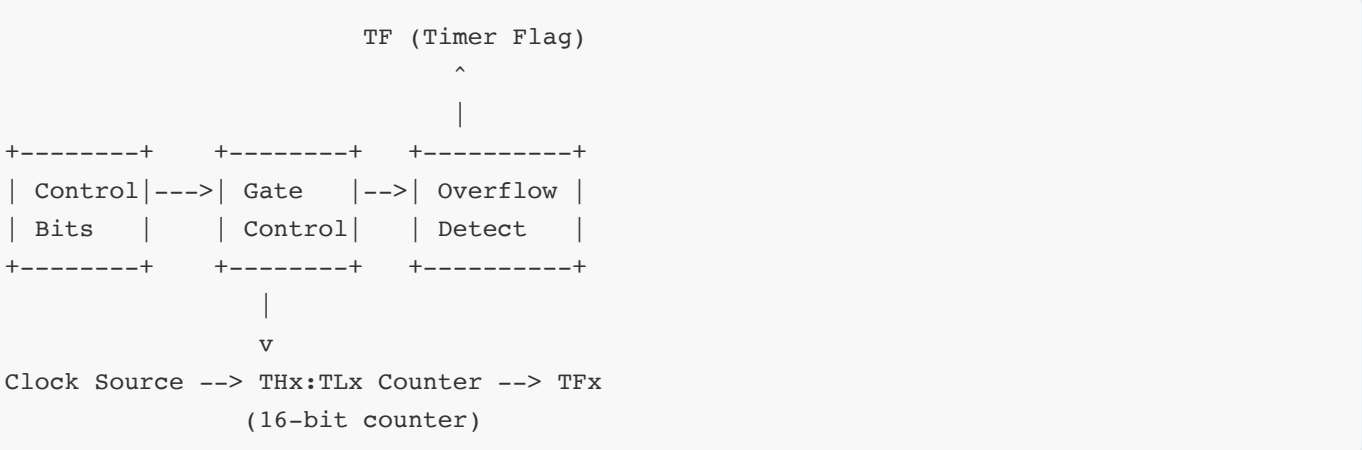
**Illustrate any one Timer Mode of 8051.**

**Answer**:

**Mode 1: 16-bit Timer/Counter**

| Feature | Description |
|---|---|
| **Timer Structure** | 16-bit timer using THx and TLx registers |
| **Operation** | Counts from 0000H to FFFFH, then sets TF flag |
| **Counter Size** | Full 16-bit counter (2^16 = 65,536 counts) |
| **Registers** | THx (high byte) and TLx (low byte) |

**Diagram:**




**Mnemonic:** "MOGC" - "Mode 1 uses Overflow detection, Gate control, Complete 16-bits"

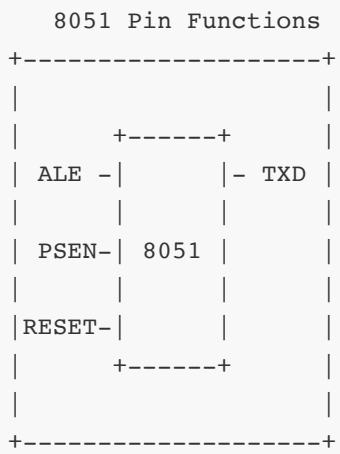
# Question 3(b) [4 marks]

**State function of ALE, PSEN, RESET and TXD pin for 8051.**

**Answer**:

| Pin | Function |
|---|---|
| **ALE** | Address Latch Enable - Provides control signal to latch low byte of address from port 0 |
| **PSEN** | Program Store Enable - Read strobe for external program memory access |
| **RESET** | Reset input - Forces CPU to initial state when held HIGH for 2 machine cycles |
| **TXD** | Transmit Data - Serial port output pin for serial data transmission |

**Diagram:**




**Mnemonic:** "APTR" - "Address latch, Program store, Total reset, tRansmit data"

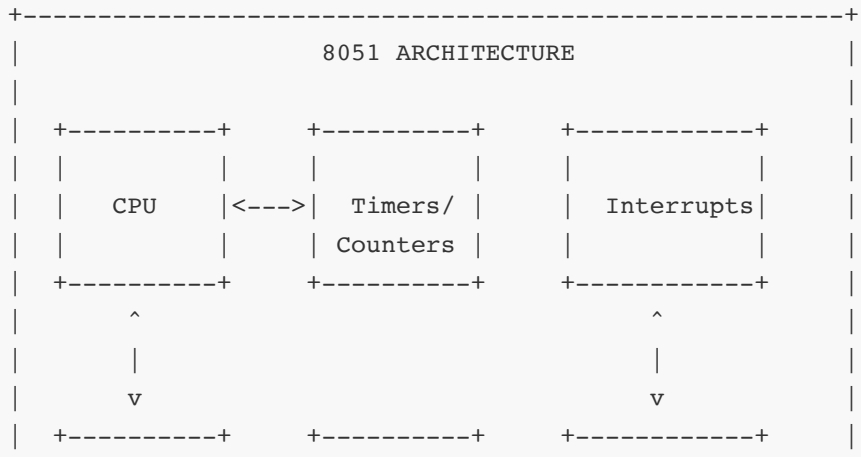
# Question 3(c) [7 marks]

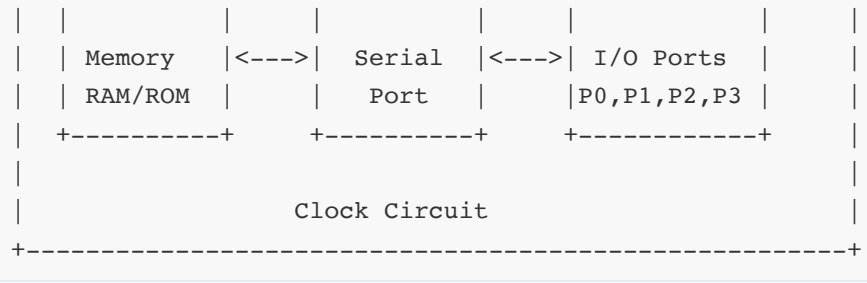
**Explain functions of each block of 8051 Microcontroller.**

**Answer**:

| Block | Function |
|---|---|
| **CPU** | 8-bit processor that fetches and executes instructions |
| **Memory** | 4KB internal ROM and 128 bytes of internal RAM |
| **I/O Ports** | Four 8-bit bidirectional I/O ports (P0-P3) |
| **Timers/Counters** | Two 16-bit timers/counters for timing and counting |
| **Serial Port** | Full-duplex UART for serial communication |
| **Interrupts** | Five interrupt sources with two priority levels |
| **Clock Circuit** | Provides timing for all operations |

**Diagram:**






**Mnemonic:** "CRIMSON" - "CPU, RAM/ROM, I/O, Memory, Serial port, Oscillator, iNterrupts"

# Question 3(a OR) [3 marks]

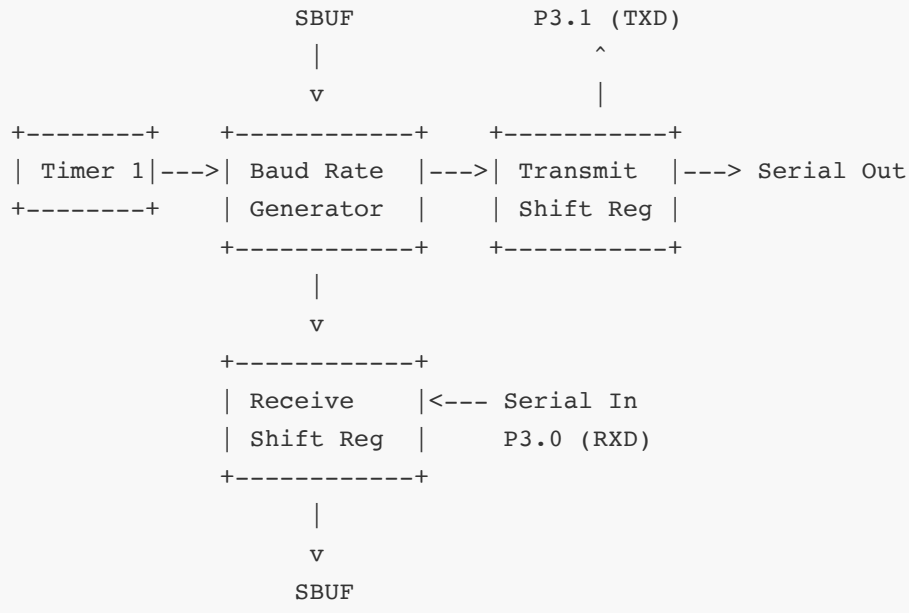
**Illustrate any one Serial Communication Mode of 8051.**

**Answer**:

**Mode 1: 8-bit UART**

| Feature | Description |
|---|---|
| **Format** | 10 bits (start bit, 8 data bits, stop bit) |
| **Baud Rate** | Variable, determined by Timer 1 |
| **Data Direction** | Full-duplex (simultaneous transmit and receive) |
| **Pins Used** | TXD (P3.1) for transmit, RXD (P3.0) for receive |

**Diagram:**




**Mnemonic:** "FADS" - "Format 10-bit, Auto baud from Timer 1, Duplex mode, Standard UART"

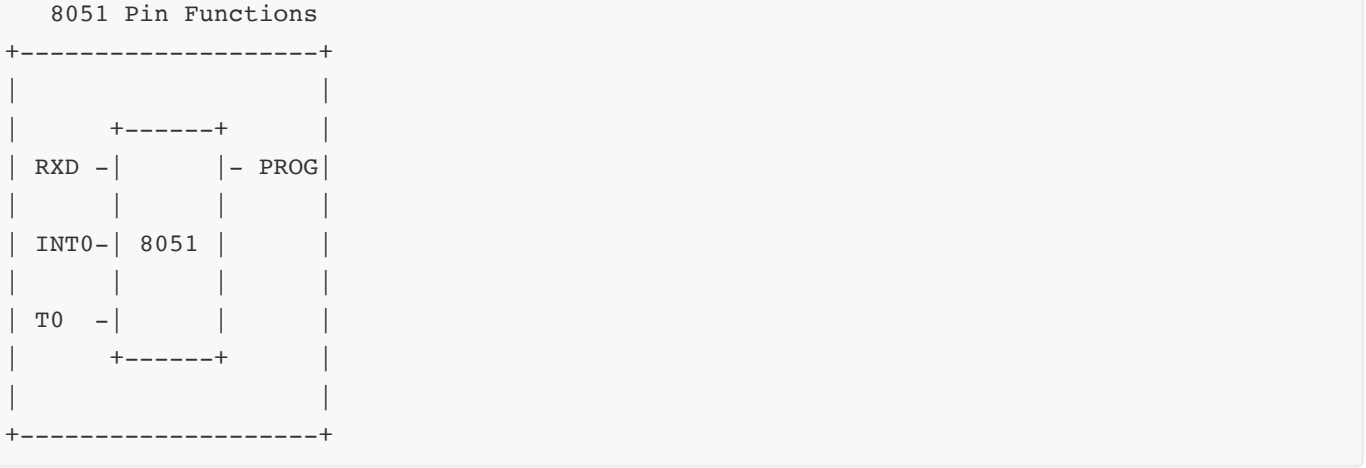
# Question 3(b OR) [4 marks]

**State function of RXD, INT0, T0 and PROG pin for 8051.**

**Answer**:

| Pin | Function |
|---|---|
| **RXD (P3.0)** | Receive Data - Serial port input pin for serial data reception |
| **INT0 (P3.2)** | External Interrupt 0 - Input that can trigger external interrupt |
| **T0 (P3.4)** | Timer 0 - External count input for Timer/Counter 0 |
| **PROG (EA)** | Program Enable - When LOW, forces CPU to fetch code from external memory |

**Diagram:**




**Mnemonic:** "RIPE" - "Receive data, Interrupt trigger, Pulse counting, External memory"

# Question 3(c OR) [7 marks]

**Describe ALU, PC, DPTR, RS0, RS1, Internal RAM and Internal ROM of 8051.**

**Answer**:

| Component | Description |
|---|---|
| **ALU** | Arithmetic Logic Unit - Performs math and logical operations |
| **PC** | Program Counter - 16-bit register that points to next instruction |
| **DPTR** | Data Pointer - 16-bit register (DPH+DPL) for external memory addressing |
| **RS0, RS1** | Register Bank Select bits in PSW - Select one of four register banks |
| **Internal RAM** | 128 bytes on-chip RAM (00H-7FH) for variables and stack |
| **Internal ROM** | 4KB on-chip ROM (0000H-0FFFH) for program storage |

**Diagram:**

```
8051 Memory Organization:
+-------------------+ 0FFFH
|                   |
|  Internal ROM     |
|  (4KB)            |
|                   |
+-------------------+ 0000H

Internal RAM:
+-------------------+ 7FH
|  Scratch Pad      |
+-------------------+ 30H
|  Bit-addressable  |
+-------------------+ 20H
|  Register Banks   |
|  (RS0,RS1 select) |
+-------------------+ 00H
```

**Mnemonic:** "APRID" - "ALU Processes, PC Remembers, Register bank select, Internal memory, DPTR points"

# Question 4(a) [3 marks]

**Develop an Assembly language program to divide 08H by 02H.**

**Answer**:

```
      MOV A, #08H    ; Load dividend 08H into accumulator
      MOV B, #02H    ; Load divisor 02H into B register
      DIV AB         ; Divide A by B (A=quotient, B=remainder)
      MOV R0, A      ; Store quotient in R0 (04H)
      MOV R1, B      ; Store remainder in R1 (00H)
```

**Diagram:**

```
Before DIV AB:       After DIV AB:
+--------+           +--------+
| A: 08H |           | A: 04H | (Quotient)
+--------+           +--------+
+--------+           +--------+
| B: 02H |           | B: 00H | (Remainder)
+--------+           +--------+
```

**Mnemonic:** "LDDS" - "Load dividend, Divisor in B, Divide, Store results"

# Question 4(b) [4 marks]

**Develop an Assembly language program to add 76H and 32H.**

**Answer**:

```
      MOV A, #76H    ; Load first number 76H into accumulator
      MOV R0, #32H   ; Load second number 32H into R0
      ADD A, R0      ; Add R0 to A (76H + 32H = A8H)
      MOV R1, A      ; Store result in R1 (A8H)
      JNC DONE       ; Jump if no carry
      MOV R2, #01H   ; If carry occurred, store 1 in R2
DONE: NOP            ; End program
```

**Diagram:**

```
+------+     +------+     +--------+
| 76H  | + ? | 32H  | = ? | A8H    | + Carry Flag
+------+     +------+     +--------+

Calculation:
   76H = 0111 0110
 + 32H = 0011 0010
-----------------
   A8H = 1010 1000
```

**Mnemonic:** "LASER" - "Load A, Store second number, Execute addition, Result stored"

# Question 4(c) [7 marks]

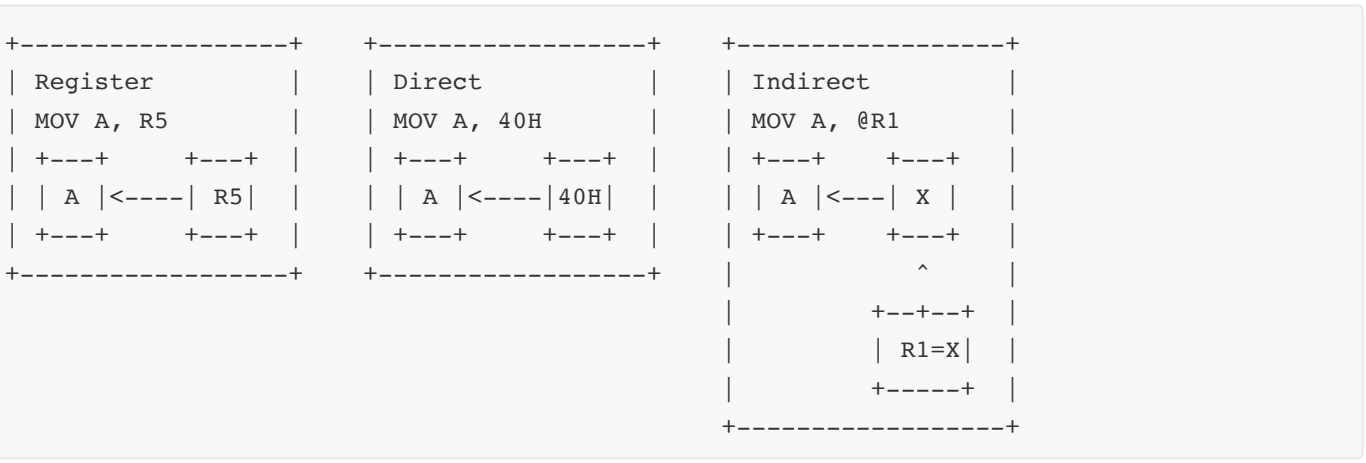
**What is Addressing mode? Classify it for 8051.**

**Answer**:

**Addressing Mode**: Method to specify the location of operand/data for an instruction.

| Addressing Mode | Description | Example |
|---|---|---|
| **Register** | Operand in register | `MOV A, R0` (Move R0 to A) |
| **Direct** | Operand at specific memory location | `MOV A, 30H` (Move data from 30H to A) |
| **Register Indirect** | Register contains address of operand | `MOV A, @R0` (Move data from address in R0 to A) |
| **Immediate** | Operand is part of instruction | `MOV A, #55H` (Load A with 55H) |
| **Indexed** | Base address + offset | `MOVC A, @A+DPTR` (Get code byte at A+DPTR) |
| **Bit** | Individual bit addressable | `SETB P1.0` (Set bit 0 of Port 1) |
| **Implied/Inherent** | Operand implied by instruction | `RRC A` (Rotate A right with carry) |

**Diagram:**




**Mnemonic:** "RIDDIB" - "Register, Immediate, Direct, Data indirect, Indexed, Bit"

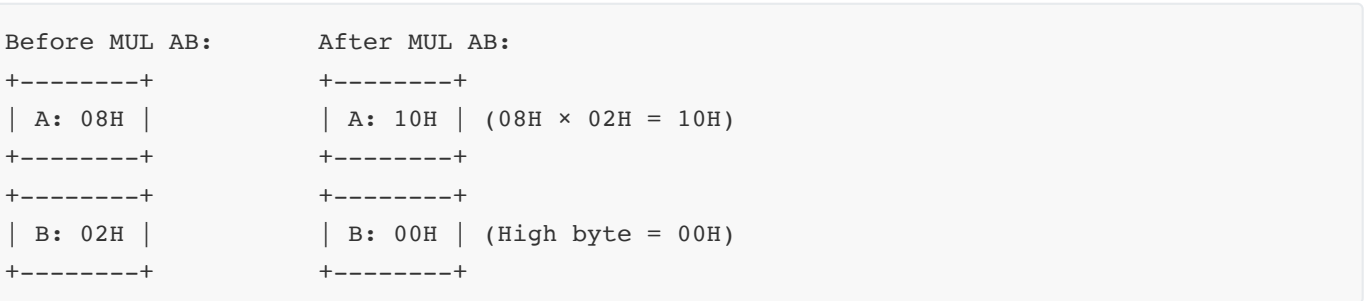
# Question 4(a OR) [3 marks]

**Develop an Assembly language program to multiply 08H and 02H.**

**Answer**:

```
      MOV A, #08H    ; Load first number 08H into accumulator
      MOV B, #02H    ; Load second number 02H into B register
      MUL AB         ; Multiply A and B (B:A = result)
      MOV R0, A      ; Store low-byte result in R0 (10H)
      MOV R1, B      ; Store high-byte result in R1 (00H)
```

**Diagram:**




**Mnemonic:** "LMSR" - "Load numbers, Multiply, Store Result"

# Question 4(b) [4 marks]

**Develop an Assembly language program to subtract 76H from 32H.**

**Answer**:

```
      MOV A, #32H    ; Load 32H into accumulator
      MOV R0, #76H   ; Load 76H into R0
      CLR C          ; Clear carry flag (borrow flag)
      SUBB A, R0     ; Subtract R0 from A with borrow (32H - 76H = BCH)
      MOV R1, A      ; Store result in R1 (BCH, which represents -44H)
```

**Diagram:**

```
+------+     +------+     +--------+
| 32H  | - ? | 76H  | = ? | BCH    | (represents -44H)
+------+     +------+     +--------+

Calculation:
   32H = 0011 0010
 - 76H = 0111 0110
-----------------
   BCH = 1011 1100 (two's complement of 44H)
```

**Mnemonic:** "LESS" - "Load first number, Enable borrow (CLR C), Subtract, Store"

# Question 4(c) [7 marks]

**List types of instruction set. Explain any three with one example.**

**Answer**:

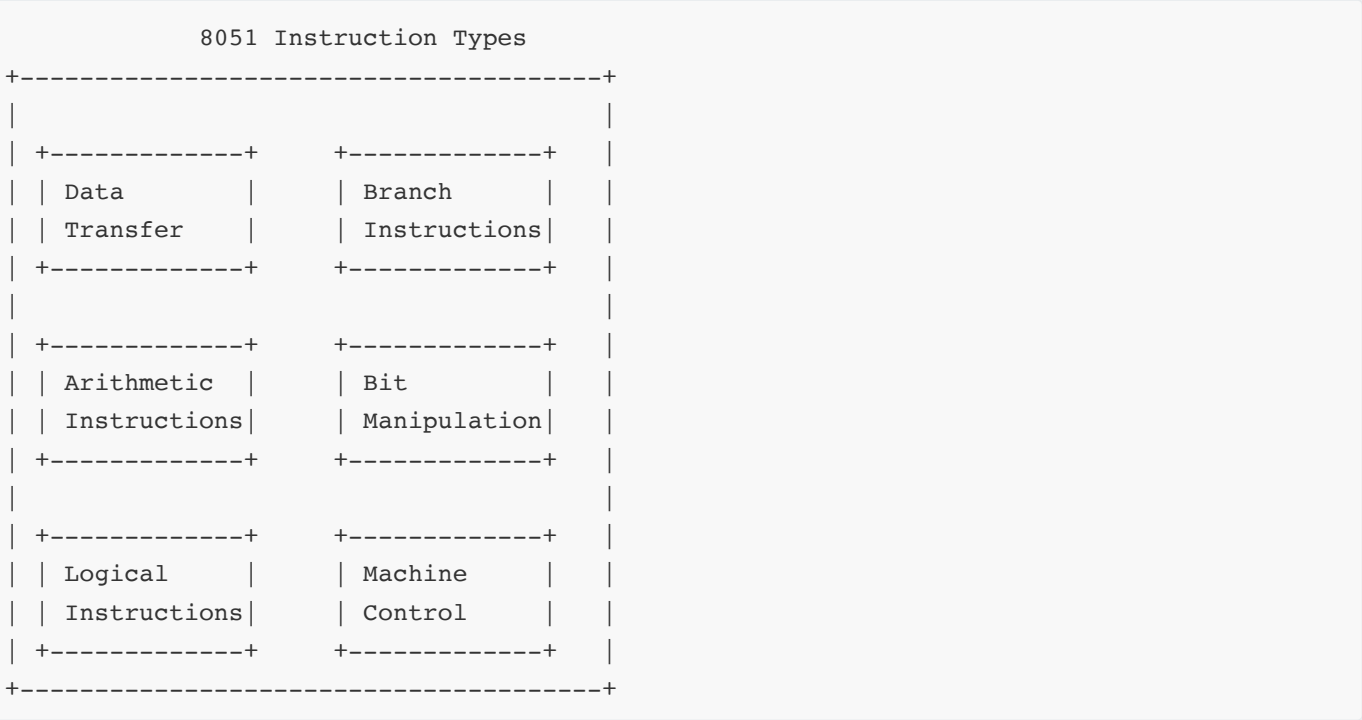
| Instruction Group | Description | Example |
|---|---|---|
| **Arithmetic** | Mathematical operations | `ADD A, R0` (Add R0 to A) |
| **Logical** | Logical operations | `ANL A, #0FH` (AND A with 0FH) |
| **Data Transfer** | Move data between locations | `MOV A, R7` (Move R7 to A) |
| **Branch** | Change program flow | `JNZ LOOP` (Jump if A not zero) |
| **Bit Manipulation** | Operate on individual bits | `SETB P1.0` (Set bit 0 of Port 1) |
| **Machine Control** | Control processor operation | `NOP` (No operation) |

**Explained Instructions**:

1. **Data Transfer Instructions**:
   - Move data between registers, memory, or I/O ports
   - Example: `MOV A, 30H` - Moves data from memory location 30H to accumulator
   - Operation: `A ← [30H]`
2. **Arithmetic Instructions**:
   - Perform mathematical operations like addition, subtraction, etc.
   - Example: `ADD A, R0` - Adds content of R0 to accumulator
   - Operation: `A ← A + R0`
3. **Logical Instructions**:
   - Perform logical operations like AND, OR, XOR, NOT
   - Example: `ANL A, #0FH` - Masks upper nibble (keeps only lower nibble)

- Operation: `A ← A AND 0FH`

**Diagram:**



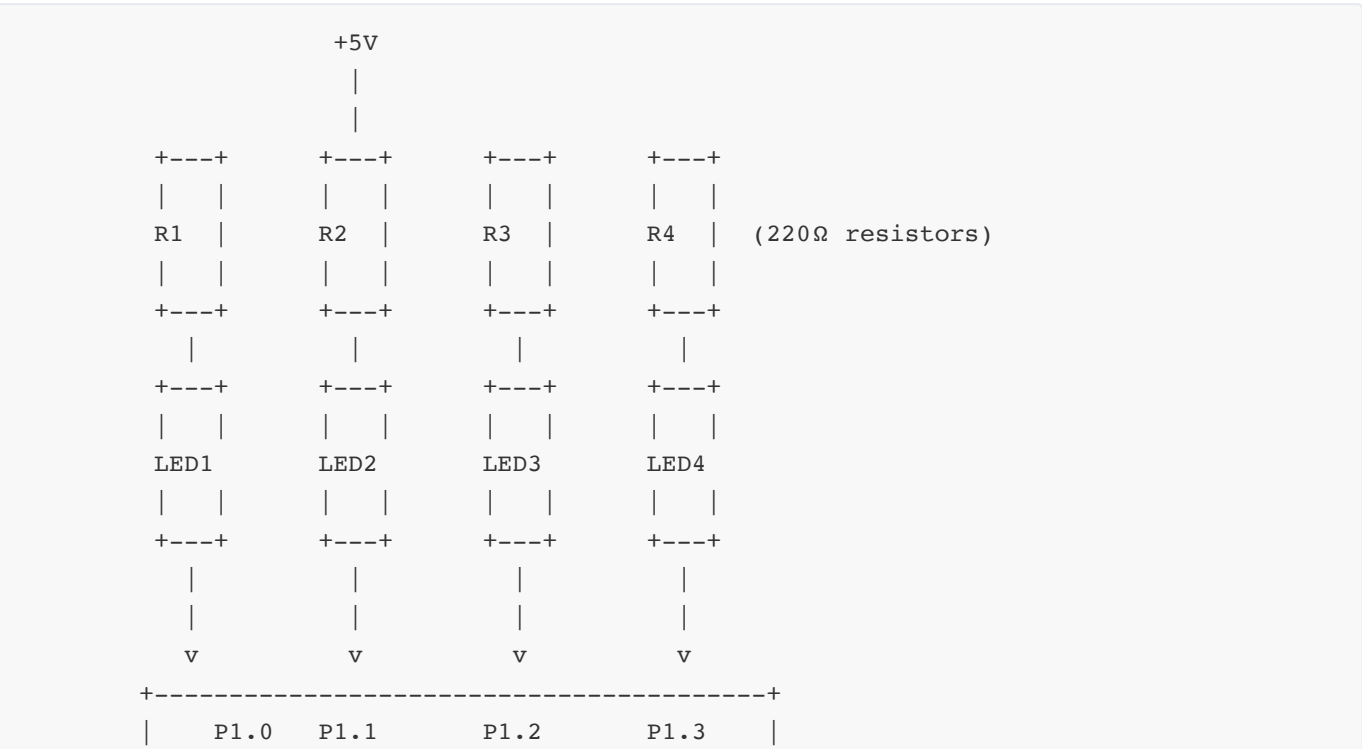

**Mnemonic:** "BALDM" - "Branch, Arithmetic, Logical, Data transfer, Machine control"

# Question 5(a) [3 marks]

**Sketch interfacing of four LEDs with 8051 Microcontroller.**

**Answer**:

**Diagram:**




**Components**:

- 8051 microcontroller
- Four LEDs
- Four current limiting resistors (220Ω)
- Power supply

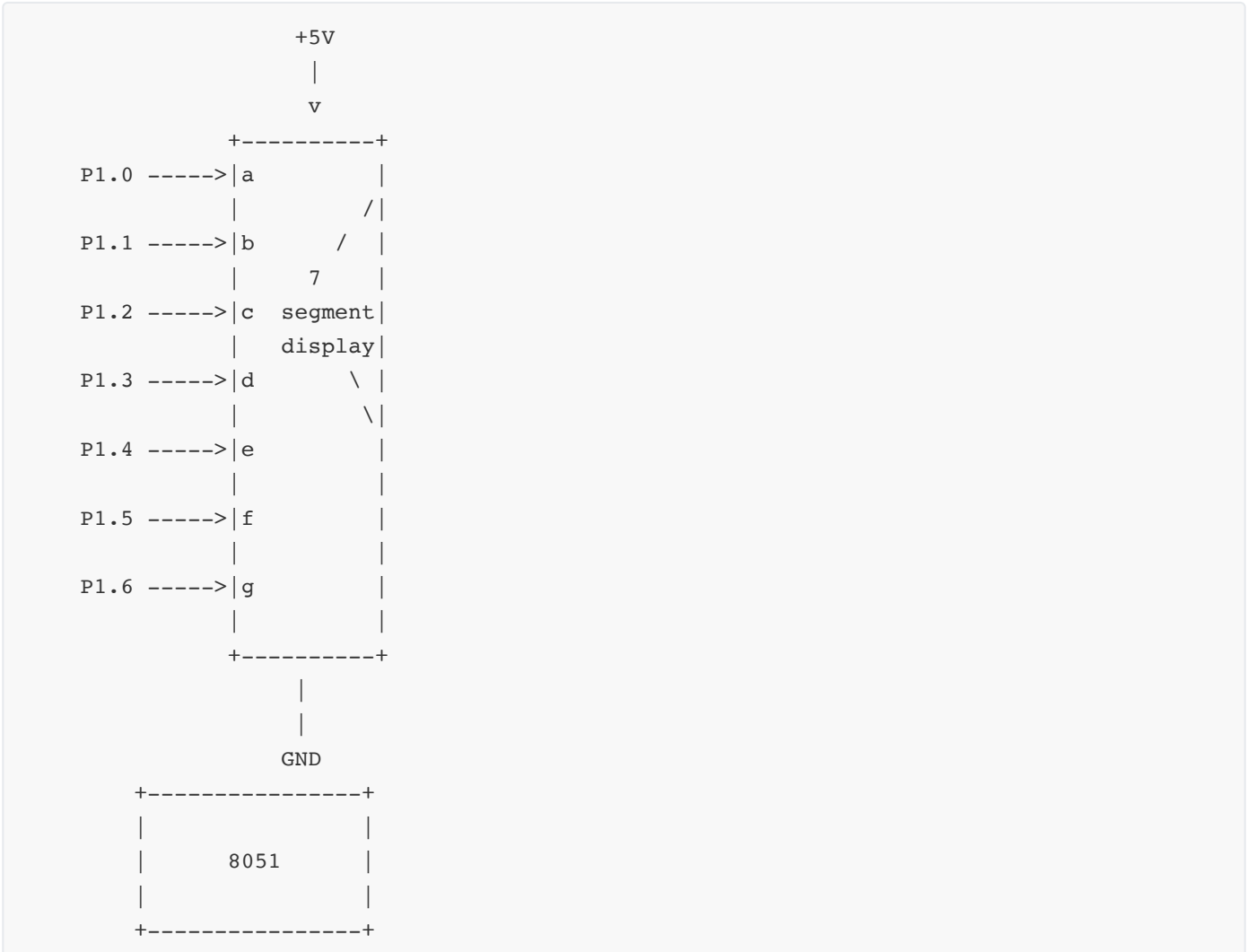
**Mnemonic:** "PALS" - "Port pins, Active-low control, LEDs, Simple circuit"

# Question 5(b) [4 marks]

**Sketch interfacing of 7 segment LED with 8051 Microcontroller.**

**Answer**:

**Diagram:**



**Components**:

- 8051 microcontroller
- 7-segment LED display (common cathode)
- Seven current limiting resistors (not shown)
- Power supply

**Code Example**:

```
; Define segment patterns for digits 0-9
DIGITS: DB 3FH, 06H, 5BH, 4FH, 66H, 6DH, 7DH, 07H, 7FH, 6FH

; Display digit 5
MOV A, #6DH      ; Segment pattern for 5
MOV P1, A        ; Send to port P1
```

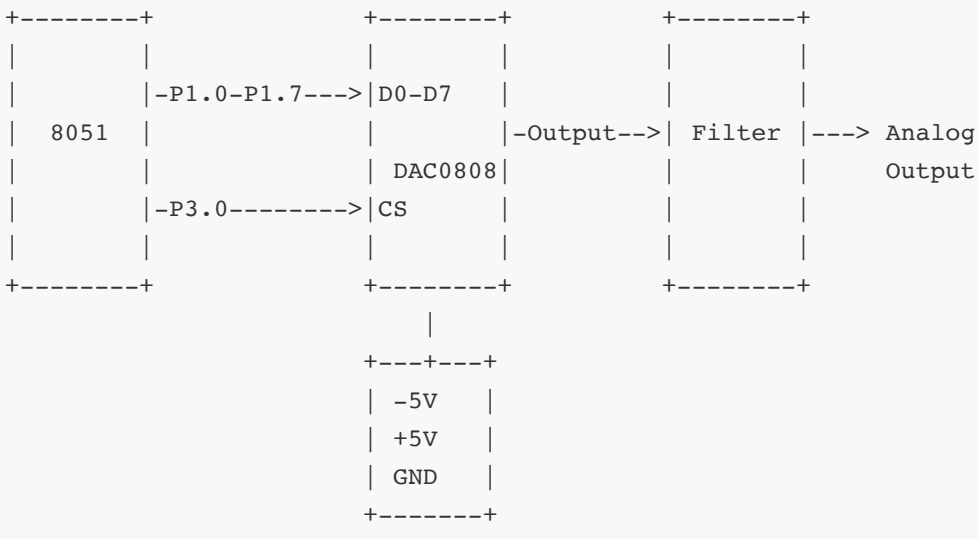
**Mnemonic:** "SPACE-7" - "Seven Pins, Active segments, Common ground, Easy display"

# Question 5(c) [7 marks]

**Explain interfacing of DAC with 8051 Microcontroller and write necessary program.**

**Answer**:

**Diagram:**




**Components**:

- 8051 microcontroller
- DAC0808 (8-bit digital-to-analog converter)
- Operational amplifier for output buffering
- RC filter for smoothing
- Power supply

**Connections**:

- P1.0-P1.7 → D0-D7 (8-bit digital input)
- P3.0 → CS (Chip Select)

**Program for generating a sawtooth wave**:

```
START:  MOV R0, #00H      ; Initialize R0 to 0
LOOP:   MOV P1, R0        ; Output value to DAC
        CALL DELAY        ; Wait for some time
        INC R0            ; Increment value
        SJMP LOOP         ; Repeat to create sawtooth wave

DELAY:  MOV R7, #50       ; Load delay counter
DELAY1: MOV R6, #255      ; Inner loop counter
DELAY2: DJNZ R6, DELAY2   ; Decrement R6 until zero
        DJNZ R7, DELAY1   ; Decrement R7 until zero
        RET               ; Return from subroutine
```

**Working Principle**:

1. Digital value is output on Port 1
2. DAC converts 8-bit digital value to proportional analog voltage
3. Filter smooths the output signal
4. Program creates a sawtooth wave by incrementing output value

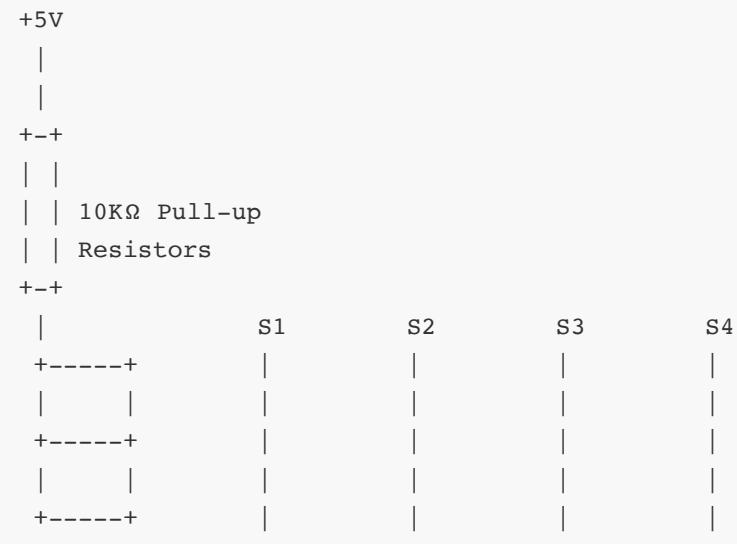
**Mnemonic:** "DICAF" - "Digital input, Increment, Convert to analog, Amplify, Filter"

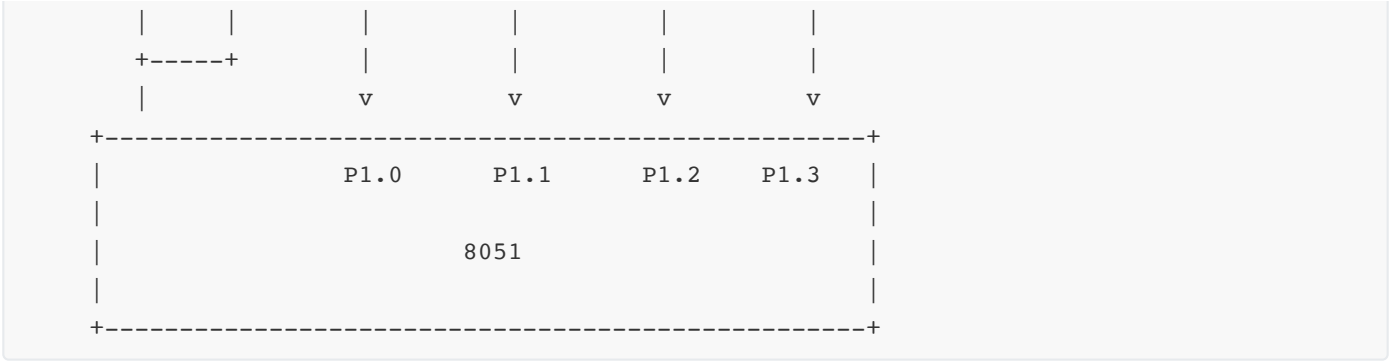
# Question 5(a OR) [3 marks]

**Sketch interfacing of four Switches with 8051 Microcontroller.**

**Answer**:

**Diagram:**






**Components**:

- 8051 microcontroller
- Four push buttons (normally open)
- Pull-up resistors (10KΩ)
- Power supply

**Working Principle**:

- Switches connect to ground when pressed
- Port pins read HIGH (1) when switch open
- Port pins read LOW (0) when switch pressed

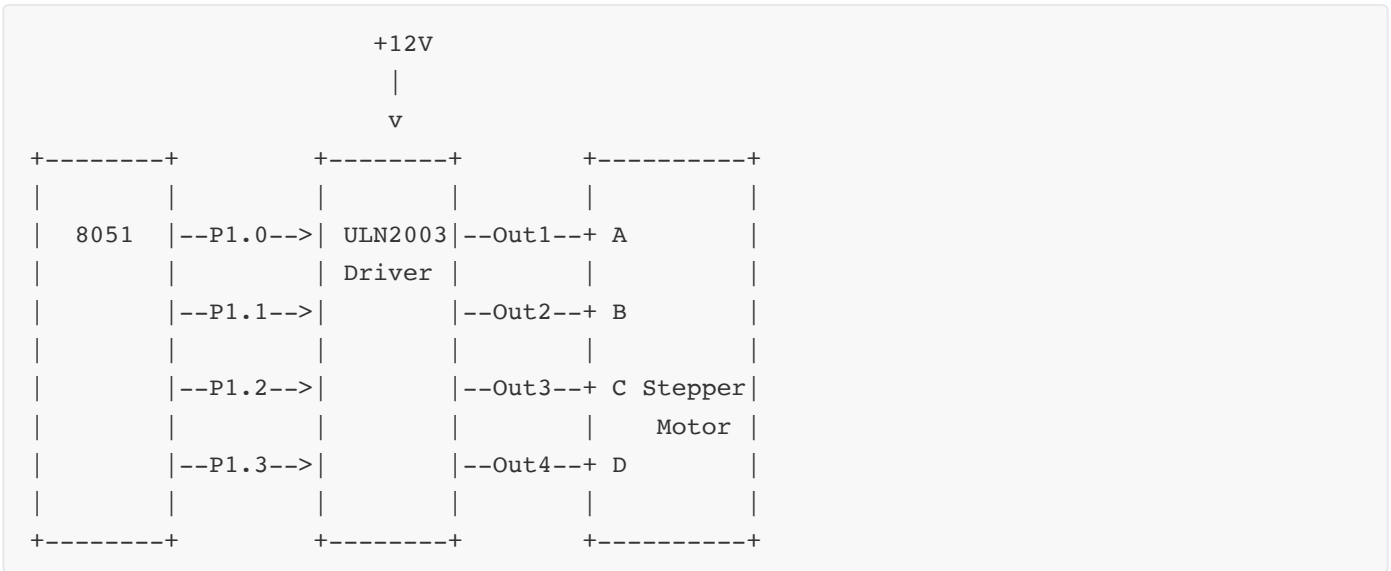
**Mnemonic:** "PIPS" - "Pull-ups, Input pins, Press for zero, Switches"

# Question 5(b) [4 marks]

**Sketch interfacing of Stepper motor with 8051 Microcontroller.**

**Answer**:

**Diagram:**




**Components**:

- 8051 microcontroller
- ULN2003 driver IC
- Stepper motor (4-phase)
- Power supply

**Excitation Sequence**:

| Step | P1.3 (D) | P1.2 (C) | P1.1 (B) | P1.0 (A) | Hex Value |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 01H |
| 2 | 0 | 0 | 1 | 0 | 02H |
| 3 | 0 | 1 | 0 | 0 | 04H |
| 4 | 1 | 0 | 0 | 0 | 08H |

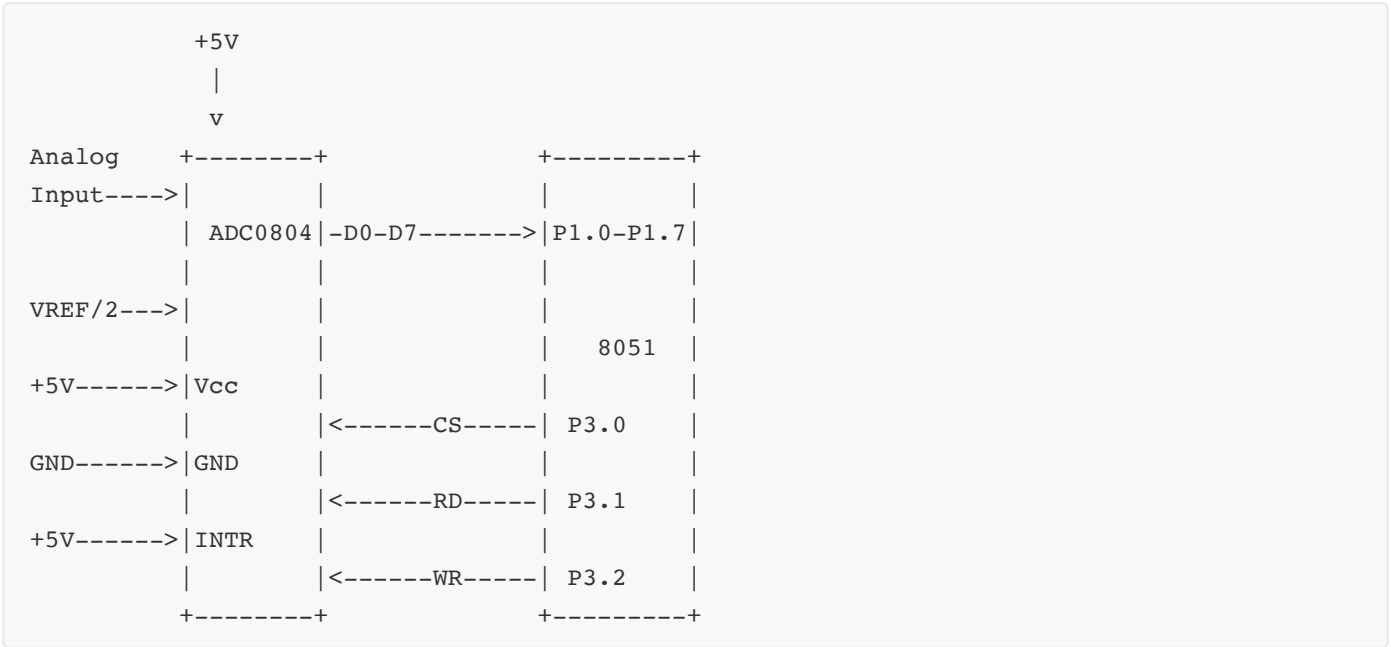
**Mnemonic:** "CUPS" - "Controller outputs sequence, ULN2003 amplifies, Phases energized, Stepping motion"

# Question 5(c) [7 marks]

**Explain interfacing of ADC with 8051 Microcontroller and write necessary program.**

**Answer**:

**Diagram:**




**Components**:

- 8051 microcontroller
- ADC0804 (8-bit analog-to-digital converter)

- Reference voltage source
- Input conditioning circuit (not shown)

**Connections**:

- P1.0-P1.7 ← D0-D7 (8-bit digital output from ADC)
- P3.0 → CS (Chip Select)
- P3.1 → RD (Read)
- P3.2 → WR (Write)

**Program for reading analog input**:

```
START:  MOV P1, #0FFH    ; Configure P1 as input port

READ:   CLR P3.0         ; Enable ADC (CS = 0)
        CLR P3.2         ; Start conversion (WR = 0)
        NOP              ; Small delay
        NOP
        SETB P3.2        ; WR = 1

WAIT:   JB P3.3, WAIT    ; Wait for conversion (INTR = 0)

        CLR P3.1         ; RD = 0 to read data
        MOV A, P1        ; Read converted value
        SETB P3.1        ; RD = 1
        SETB P3.0        ; Disable ADC (CS = 1)

PROCESS:                 ; Process the data as needed
        ; Example: Store in R0
        MOV R0, A

        SJMP READ        ; Repeat for continuous conversion
```

**Working Principle**:

1. Controller sends start conversion signal
2. ADC converts analog input to 8-bit digital value
3. Controller reads digital value after conversion complete
4. Program processes the digital value as required

**Mnemonic:** "CARSW" - "Convert Analog, Read Digital, Start conversion, Wait for completion"