PVP 23

Code: 23EE3602

III B.Tech - II Semester - Regular Examinations – APRIL 2026

MICROPROCESSORS AND MICROCONTROLLERS
(ELECTRICAL & ELECTRONICS ENGINEERING)

Duration: 3 hours Max. Marks: 70

Note: 1. This question paper contains two Parts A and B.
2. Part-A contains 10 short answer questions. Each Question carries 2 Marks.
3. Part-B contains 5 essay questions with an internal choice from each unit. Each Question carries 10 marks.
4. All parts of Question paper must be answered in one place.

BL – Blooms Level CO – Course Outcome

PART – A

		BL	CO
1.a)	Write the uses of memory segmentation in 8086.	L2	CO1
b)	Describe the role of the Bus Interface Unit (BIU) in 8086.	L2	CO1
c)	Define and list out various addressing modes of 8086.	L2	CO2
d)	Explain the push and pop operation of 8086.	L2	CO2
e)	Write any three silent features of mode 2 of 8255.	L2	CO3
f)	Draw the control word format of I/O modes of 8255 PPI.	L2	CO3
g)	What is the function of the TMOD and TCON registers?	L2	CO1
h)	List the five interrupt sources of the 8051.	L2	CO1
i)	List out the various features of PIC 18.	L2	CO4
j)	What are the three main registers used for I/O port operations in the PIC18?	L2	CO4

PART – B

			BL	CO	Max. Marks
UNIT-I					
2	a)	Summarize the architectural advancements of 80486 over 8086.	L2	CO1	5 M
	b)	Draw the functional diagram of 8086 Microprocessor and explain the function of each block in detail.	L2	CO1	5 M
OR					
3	a)	Briefly explain register organization in 8086 microprocessor.	L2	CO1	5 M
	b)	Describe the memory segmentation and instruction queue.	L2	CO1	5 M
UNIT-II					
4	a)	Draw and discuss the read and write cycle timing diagrams of 8086 in minimum mode.	L2	CO1	5 M
	b)	Write an Assembly language program to multiply two 16-bit Hexa decimal numbers in 8086.	L4	CO2	5 M
OR					
5	a)	List out the shift and rotate instructions of 8086 microprocessor with examples.	L2	CO2	5 M
	b)	Write an Assembly language program to find the sum of squares of first ten numbers.	L4	CO2	5 M
UNIT-III					
6	a)	With a neat diagram explain the architecture of 8255.	L2	CO3	5 M

	b)	Differentiate between BSR and I/O modes of 8255 PPI.	L2	CO3	5 M
OR					
7	a)	Draw and explain the interfacing of analog to digital converter with 8086 microprocessor.	L2	CO3	5 M
	b)	Draw the architecture of 8257 DMA controller and explain its working.	L3	CO3	5 M
UNIT-IV					
8	a)	Describe the operation of timers present in 8051 microcontroller.	L2	CO2	5 M
	b)	List the various registers present in 8051 microcontroller and explain with an example.	L2	CO1	5 M
OR					
9	a)	Explain the different Instruction set of 8051 in detail.	L2	CO2	6 M
	b)	Explain how interrupts are prioritized.	L2	CO2	4 M
UNIT-V					
10	a)	Explain status register of PIC 18 microcontroller.	L2	CO4	5 M
	b)	Draw and explain the block diagram of PIC 18 microcontroller.	L3	CO4	5 M
OR					
11	a)	Explain different data types and operators used in PIC C programming.	L2	CO4	5 M
	b)	Write a C program to toggle PORTB bits with a delay of 200ms.	L4	CO4	5 M

III B. Tech – II Semester – Regular Examinations – APRIL 2026
MICROPROCESSORS AND MICROCONTROLLERS 23 EE3602
(ELECTRICAL & ELECTRONJCS ENGINEERING)
SCHEME OF VALUATION

PART A

1a) uses – 2M
b) BIU - 2M
c) Definition – 1M, List - 1M
d) PUSH - 1M POP - 1M
e) Salient features - 2M
f) Format - 2M
g) TMOD - 1M TCON - 1M
h) Interrupts - 2M
i) Any four features - 2M

i IIoRegisters of PIC 18 – 2M

PART B

2a) 8086 Vs 80486 - 5M
2b) Diagram - 3M
Explanation - 2M
3a) Registers - 5M
3b) Memory segmentation - 3M
Instruction Queue - 2M
4a) Read Cycle Diagram - 2M
Write Cycle Diagram - 2M
Explanation - 1M
4b) Program - 5M
5a) List - 2M
Any two instructions - 3M
5b) Program - 5M
6a) Diagram - 3M
Explanation - 2M
6b) BSR Mode - 2M
I/O Mode - 3M
7a) Diagram - 3M
Explanation - 2M
7b) Diagram - 3M
Explanation - 2M
8a) Modes of timers - 5M
8b) List of registers - 2M
Explanation - 3M
9a) List - 3M
Any two instructions - 2M
9b) IP format - 3M
Explanation - 2M
10a) Format - 3M
Explanation - 2M
10b) Diagram - 3M
Explanation - 2M
11a) Data types - 3M
Operators - 2M
11b) Program -5M

Code: 23EE3602

III B. Tech – II Semester – Regular Examinations – APRIL 2026
MICROPROCESSORS AND MICROCONTROLLERS
(ELECTRICAL & ELECTRONJCS ENGINEERING)
DETAILED KEY
PART A

1a) 1. Segmentation allows the 8086 to access up to **1 MB memory** using 16-bit registers by combining segment and offset addresses. Without segmentation, addressing would be limited to 64 KB.

1b) Instructions fetch, Instruction queuing, Operand fetch and storage, Address relocation and Bus control.

1c) Addressing mode indicates a way of locating data or operands.

1) Immediate Addressing mode
2) Register Addressing mode
3) Direct Addressing mode
4) Register Indirect Addressing mode
5) Indexed Addressing mode
6) Register Relative Addressing mode
7) Base Indexed Addressing mode
8) Relative Based Indexed Addressing mode

1d) **PUSH: Push to Stack**

PUSH SOURCE

This instruction pushes the contents of the specified register/memory location on to the stack.

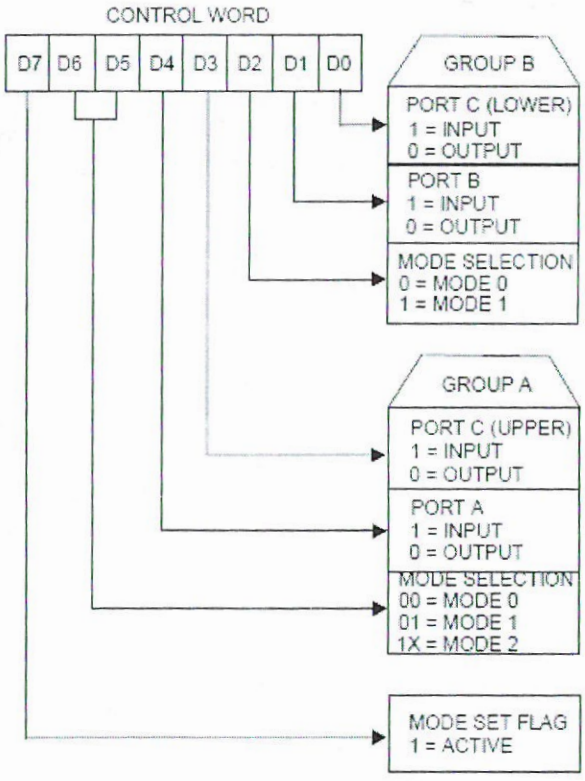
POP: Pop from Sack

POP DESTINATION

This instruction when executed, loads the specified register/memory.

1e) Bidirectional I/O
Handshaking Signals
Data transfer control

1f)



1g) In the 8051 microcontroller, the TMOD register configures how timers operate (mode, timer/counter selection, gate control), while the TCON register controls their start/stop, overflow flags, and external interrupts. Together, they define and manage the behavior of Timer 0 and Timer 1.

1h) External Interrupt 0 (INT0)
Timer 0 Overflow Interrupt (TF0)
External Interrupt 1 (INT1)
Timer 1 Overflow Interrupt (TF1)
Serial Communication Interrupt (RI/TI)

1i) **Features of PIC18**

8-bit CPU
RISC (Reduced instruction set computer) Architecture
2 MB program memory space
256 bytes to IKB of data EEPROM
Up to 3968 bytes of on-chip SRAM
4 KB to 128KB flash program memory
Sophisticated timer functions that include: input capture, output compare, PWM, real-time interrupt, and watchdog timer
Serial communication interfaces: SCI, SPI, I2C, and CAN Background debug mode (BDM)
10-bit A/D converter
Memory protection capability
Instruction pipelining
Operates at up to 40 MHz crystal oscillator

1j) TRISx (Tri-State Register)
PORTx (Port Register)
LATx (Latch Register)

PART B

2a)

Feature	8086	80486
Data Bus Width	16-bit	32-bit
Address Bus Width	20-bit (1 MB memory)	32-bit (4 GB memory)
Clock Speed	5–10 MHz	16–100 MHz
Instruction Execution	No pipelining, multi-cycle	5-stage pipeline, many instructions in 1 cycle
Floating Point Unit (FPU)	External 8087 coprocessor	Integrated on-chip FPU
Cache Memory	None	8 KB–16 KB on-chip L1 cache
Transistor Count	~29,000	1.2–1.6 million
Performance	~0.33 MIPS	Up to 50 MIPS

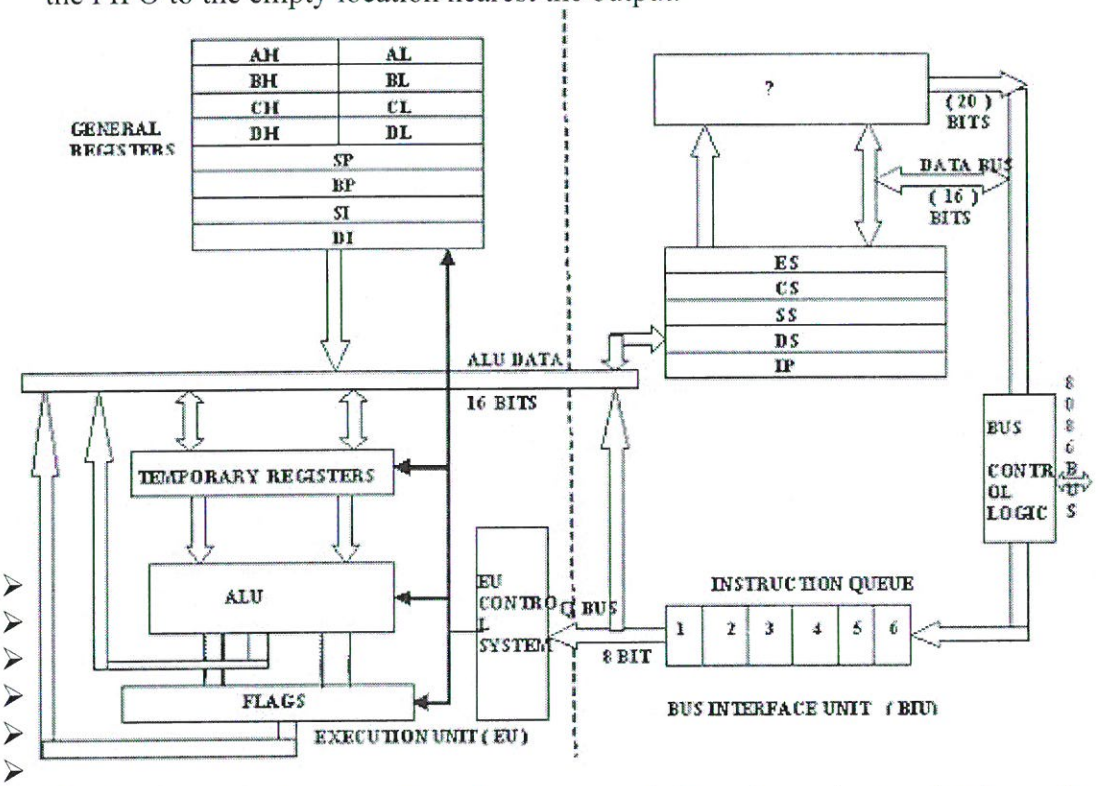
2b) Architecture of 8086 microprocessor

- 8086 has two blocks BIU and EU.
- The BIU performs all bus operations such as instruction fetching, reading and writing operands for memory and calculating the addresses of the memory operands. The instruction bytes are transferred to the instruction queue.
- EU executes instructions from the instruction system byte queue.
- Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining. This results in efficient use of the system bus and system performance.
- BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder.

- EU contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register.
- **BUS INTERFACR UNIT:**
- It provides a full 16 bit bidirectional data bus and 20 bit address bus.
- The bus interface unit is responsible for performing all external bus operations.

Specifically it has the following functions:

- Instructions fetch, Instruction queuing, Operand fetch and storage, Address relocation and Bus control.
- The BIU uses a mechanism known as an instruction stream queue to implement pipeline *architecture.*
- This queue permits prefetch of up to six bytes of instruction code. Whenever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by prefetching the next sequential instruction.
- These prefetching instructions are held in its FIFO queue. With its 16 bit data bus, the BIU fetches two instruction bytes in a single memory cycle.
- After a byte is loaded at the input end of the queue, it automatically shifts up through the FIFO to the empty location nearest the output.



- The EU accesses the queue from the output end. It reads one instruction byte after the other from the output of the queue. If the queue is full and the EU is not requesting access to operand in memory.
- These intervals of no bus activity, which may occur between bus cycles are known as ***Idle state.***
- If the BIU is already in the process of fetching an instruction when the EU request it to read or write operands from memory or I/O, the BIU first completes the instruction fetch bus cycle before initiating the operand read / write cycle.
- The BIU also contains a dedicated adder which is used to generate the 20bit physical address that is output on the address bus. This address is formed by combining the current contents of the code segment CS register and the current contents of the

instruction pointer IP register.

- The BIU is also responsible for generating bus control signals such as those for memory read or write and I/O read or write.

EXECUTION UNIT

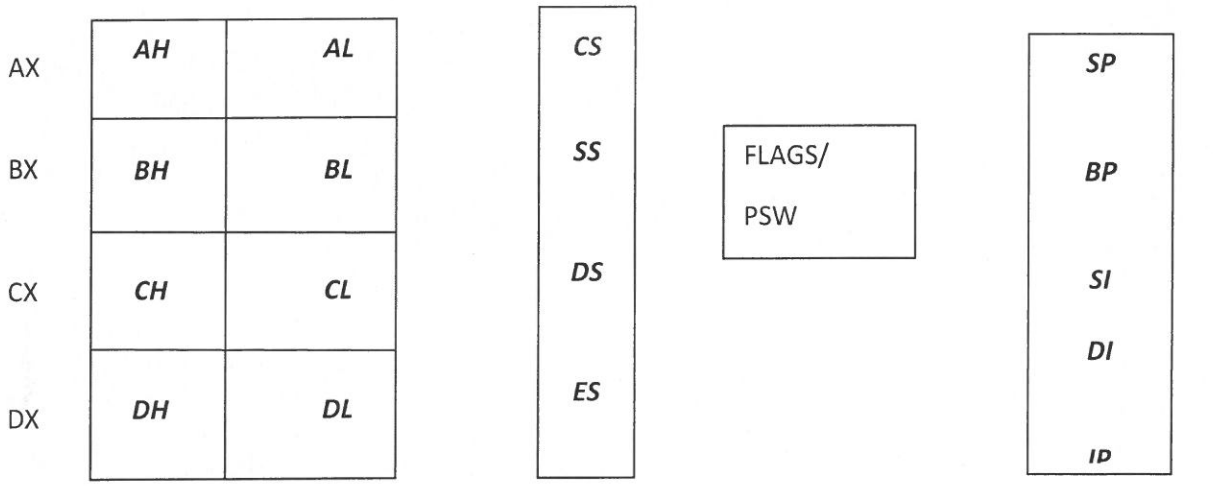
The Execution unit is responsible for decoding and executing all instructions.

- The EU extracts instructions from the top of the queue in the BIU, decodes them, generates operands if necessary, passes them to the BIU and requests it to perform the read or write bys cycles to memory or I/O and perform the operation specified by the instruction on the operands.
- During the execution of the instruction, the EU tests the status and control flags and updates them based on the results of executing the instruction.
- If the queue is empty, the EU waits for the next instruction byte to be fetched and shifted to top of the queue.
- When the EU executes a branch or jump instruction, it transfers control to a location corresponding to another set of sequential instructions.
- Whenever this happens, the BIU automatically resets the queue and then begins to fetch instructions from this new location to refill the queue.

3a) Register organization of 8086

All the registers of 8086 are 16-bit registers.

The register set of 8086 can be categorized into 4 different groups. The register organization of 8086 is shown in the figure.

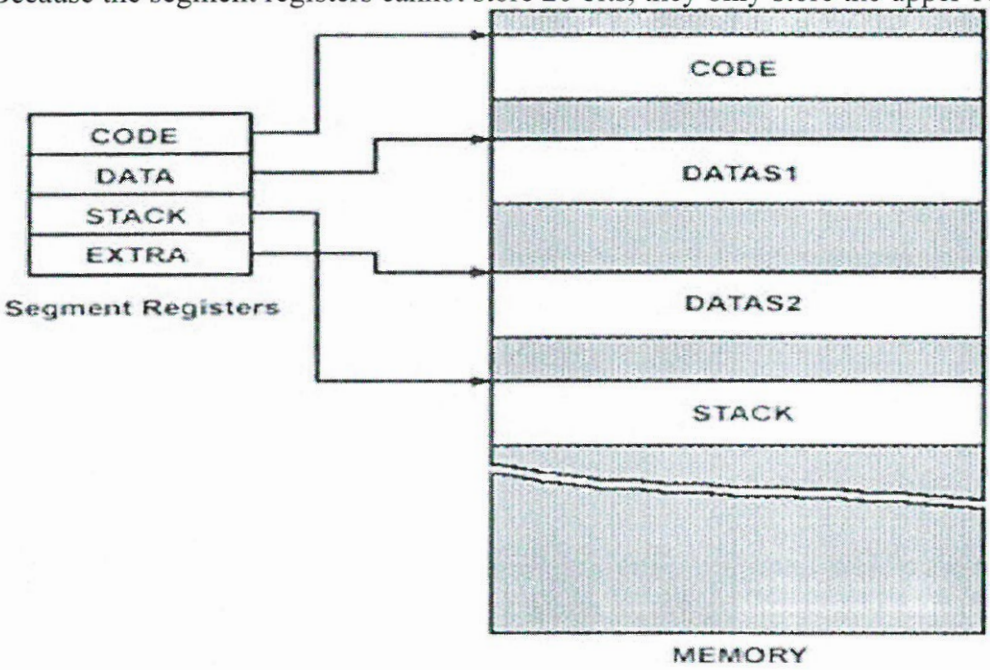


General data registers and index Registers | Segment Registers | Pointers

3b) Memory segmentation of 8086

- A. The total memory size is divided into segments of various sizes. A segment is just an area in memory. The process of dividing memory this way is called Segmentation.
- In memory, data is stored as bytes. Each byte has a specific address. Intel 8086 has 20 lines address bus. With 20 address lines, the memory that can be addressed is 2^{20}bytes.
- 2^{20} = 1,048,576 bytes (1 MB).
- 8086 can access memory with address ranging from 00000 H to FFFFF H.
- 20-bit address of 1MB memory cannot be stored in the registers available in 8086 because the size of each register is 16-bit.
- So the total 1MB memory is divided into 16 segments. The size of each segment is 64KB.

- Even though the 1MB memory is divided into 16 segments, 8086up can access only four segments at a time.
  These are:
  - Code Segment
  - Data Segment
  - Stack Segment
  - Extra Segment
- Each of these segments are addressed by an address stored in corresponding segment register.
- These registers are 16-bit in size.
- Each register stores the base address (starting address) of the corresponding segment.
- Because the segment registers cannot store 20 bits, they only store the upper 16 bits.



Instruction queue in 8086

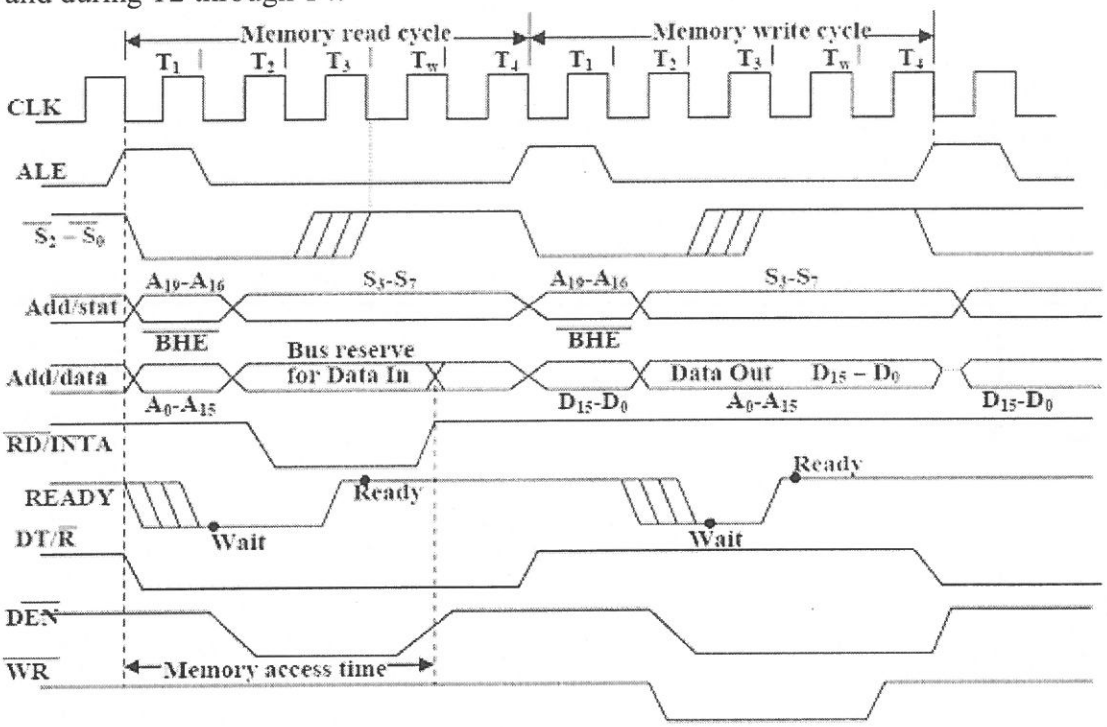
The length of instruction queue is ***six bytes***. The BIU uses a mechanism known as an instruction stream queue to implement pipeline architecture. Due to this pipe line processing speed of the processor increases.

This queue permits prefetch of up to six bytes of instruction code. Whenever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by prefetching the next sequential instruction.

- These prefetching instructions are held in its FIFO queue. With its 16 bit data bus, the BIU fetches two instruction bytes in a single memory cycle.
- After a byte is loaded at the input end of the queue, it automatically shifts up through the FIFO to the empty location nearest the output.
- The EU accesses the queue from the output end. It reads one instruction byte after the other from the output of the queue.

4a) The 8086 has a combined address and data bus commonly referred as a time multiplexed address and data bus. The main reason behind multiplexing address and data over the same pins is the maximum utilization of processor pins and it facilitates the use of 40 pin standard DIP package. The bus can be de-multiplexed using a few latches and transreceivers, whenever required.

Basically, all the processor bus cycles consist of at least four clock cycles. These are referred to as T1, T2, T3, T4. The address is transmitted by the processor during T1, it is present on the bus only for one cycle. The negative edge of this ALE pulse is used to separate the address and the data or status information. In maximum mode, the status lines S0, S1 and S2 are used to indicate the type of operation. Status bits S3 to S7 are multiplexed with higher order address bits and the BHE signal. Address is valid during T1 while status bits S3 to S7 are valid during T2 through T4.



4b)

```
MOV AX,NUM1
MOV BX,NUM2
MUL BX
INT 03
```

Note: Students may choose any numbers and in place of BX they can select CX also. Award marks accordingly.

5a) Shift instructions

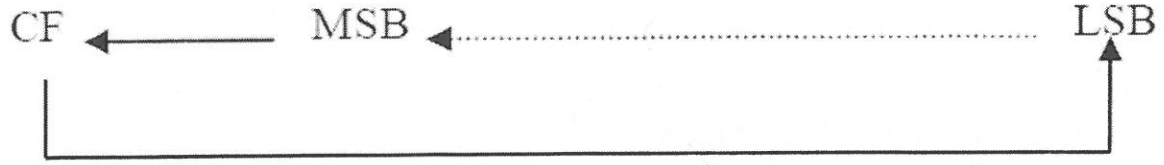
SAL, SAR, SHL

Rotate instructions

ROL, RCL, ROR, RCR

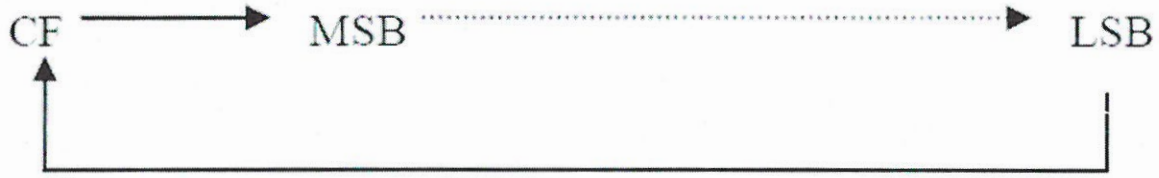
RCL – RCL Destination, Count

This instruction rotates all the bits in a specified word or byte some number of bit positions to the left. The operation circular because the MSB of the operand is rotated into the carry flag and the bit in the carry flag is rotated around into LSB of the operand.



RCR – RCR Destination, Count

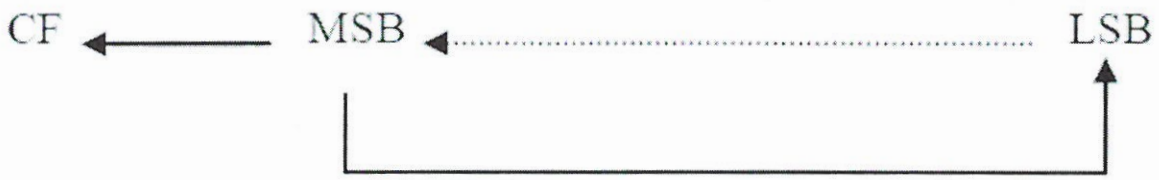
This instruction rotates all the bits in a specified word or byte some number of bit positions to the right. The operation circular because the LSB of the operand is rotated into the carry flag and the bit in the carry flag is rotate around into MSB of the operand.



For multi-bit rotate, CF will contain the bit most recently rotated out of the LSB.

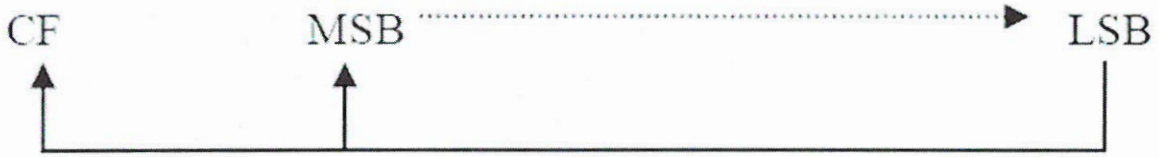
ROL – ROL Destination, Count

This instruction rotates all the bits in a specified word or byte to the left some number of bit positions. The data bit rotated out of MSB is circled back into the LSB. It is also copied into CF. In the case of multiple-bit rotate, CF will contain a copy of the bit most recently moved out of the MSB.



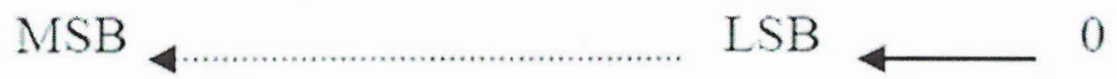
ROR – ROR Destination, Count

This instruction rotates all the bits in a specified word or byte some number of bit positions to right. The operation is desired as a rotate rather than shift, because the bit moved out of the LSB is rotated around into the MSB. The data bit moved out of the LSB is also copied into CF. In the case of multiple bit rotates, CF will contain a copy of the bit most recently moved out of the LSB.



SHL – SHL Destination, Count

SAL and SHL are two mnemonics for the same instruction. This instruction shifts each bit in the specified destination some number of bit positions to the left. As a bit is shifted out of the LSB operation, a 0 is put in the LSB position.



SAR – SAR Destination, Count

This instruction shifts each bit in the specified destination some number of bit positions to the right. As a bit is shifted out of the MSB position, a copy of the old MSB is put in the MSB position. In other words, the sign bit is copied into the MSB.

SHR – SHR Destination, Count

This instruction shifts each bit in the specified destination some number of bit positions to the right. As a bit is shifted out of the MSB position, a 0 is put in its place.

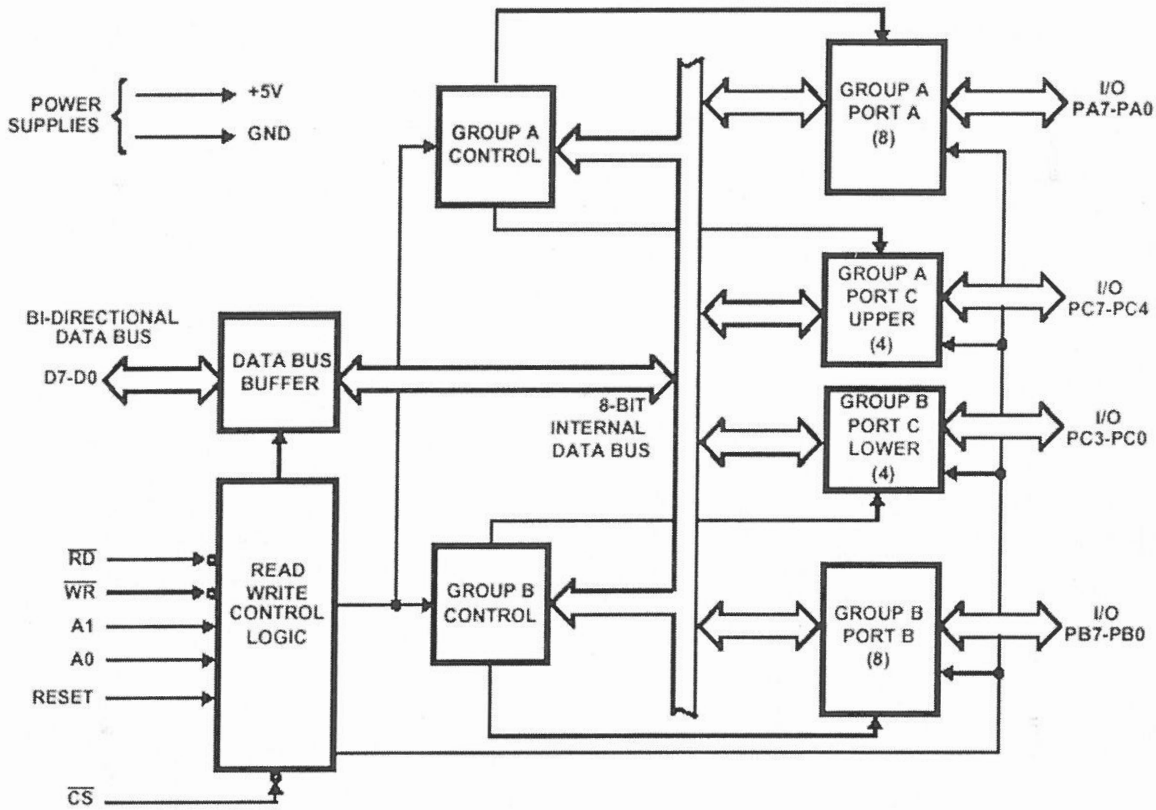
0 ——→ MSB ....→ LSB

5b)

```
        MOV CX, 000A
        MOV BX,0000
BACK:   MOV AX,CX
        MUL AX
        ADD BX,AX
        LOOP BACK
        INT 03
```

Note: Students may handle instructions in a different way. Award marks accordingly.

6a) BLOCK DIAGRAM OF 8255 PPI



BLOCK DIAGRAM OF 8255

The parallel input-output port chip 8255 is also known as Programmable peripheral parallel input-output port. It has 24 input/output lines which may be individually programmed into two groups of twelve lines each, or three groups of eight lines. The two groups of I/O pins are named as Group A and Group B. Each of these two groups contain a subgroup of eight I/O lines called as 8-bit port and another subgroup of four I/o lines or a 4-bit port. Thus Group A contains an 8-bit port A along with a 4-bit port C upper. The port A lines are identified by symbols PA_0-PA_7 while the port C lines are identified as PC_4-PC_7. Similarly Group B contains an 8-bit port B, containing lines PB_0-PB_7 and a 4-bit port C with lower bits PC_0-PC_3. The port C upper and port C lower can be used in combination as am 8-bit port C. Both the port Cs are assigned the same address. Thus one may have either three 8-bit I/O ports or two 8-bit and two 4-bit I/O ports from 8255. All of these ports can function independently either as input or as output ports. This can be achieved by programming the bits of an internal register of 8255 called as Control Word Register(CWR).

6b) 8255 PPI operates in two basic modes

1. Bit Set Reset mode(BSR)
2. I/O mode

To set the mode of operation CWR has to be written

BSR Mode:

Any one of the port C lines can be set or reset using BSR mode.

The format of CWR is

D7	D6	D5	D4	D3	D2	D1	D0
0	X	X	X	B2	B1	B0	0/1

D0 – 0/1 – RESET/SET any one of the port c lines depending upon D1, D2, D3 of CWR.

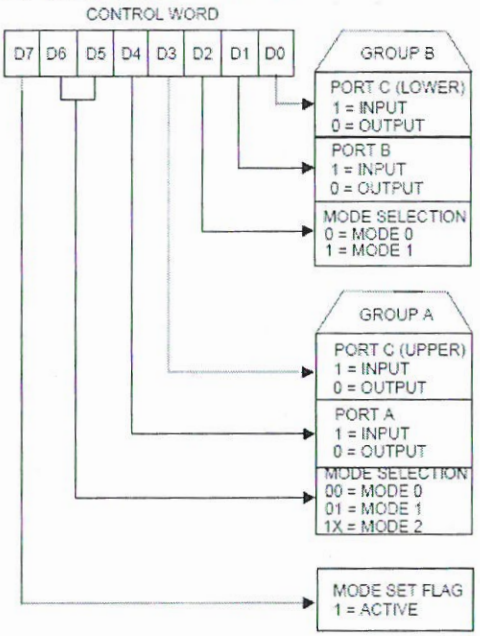
D7 – 0 – BSR MODE

1 – I/O MODE

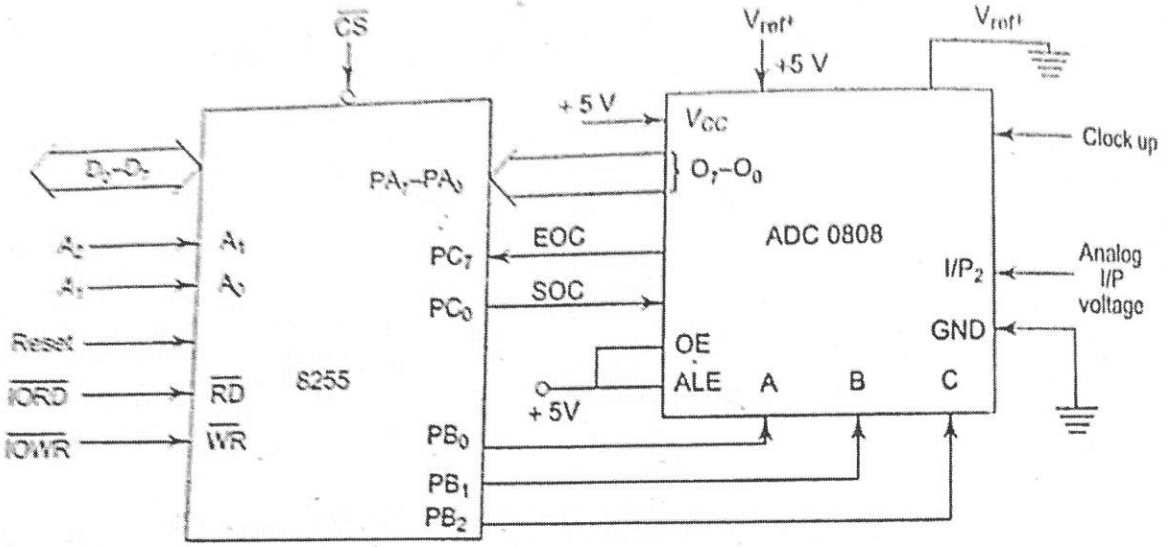
B2	B1	B0	PORTC LINES
0	0	0	PC0
0	0	1	PC1
0	1	0	PC2
0	1	1	PC3
1	0	0	PC4
1	0	1	PC5
1	1	0	PC6
1	1	1	PC7

I/O MODE

The format of CWR to operate 8255 in I/O mode is



7a)



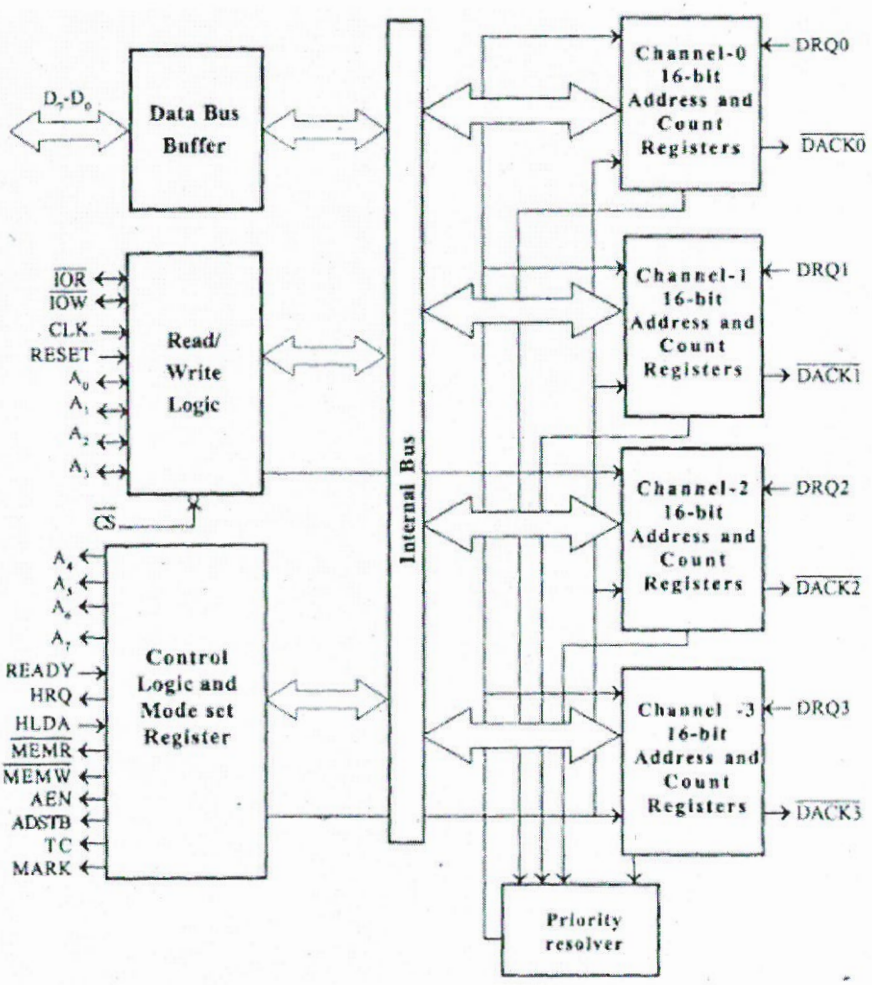
The analog to digital converter is treated as an input device by the microprocessor, that sends an initializing signal to the ADC to start the analog to digital data convertsion process. The start of conversion signal is a pulse of a specific duration. The process of analog to digital conversion is slow process, and the microprocessor has to wait for the digital data till the conversion is over. After the conversion is over, the ADC sends end of conversion(EOC) signal to inform the microprocessor about it and the result is redy at the output buffer of the ADC.

A general algorithm for ADC interfacing contains the following steps.

1. Ensure the stability of analog input, applied to the ADC.
2. Issue start of conversion(SOC) pulse to ADC
3. Read end of conversion (EOC) signal to mark the end of conversion process.
4. Read digital data output of the ADC as equivalent digital output.

7b) Functional Block Diagram of 8257:

The functional block diagram of 8257 is shown in fig. The functional blocks of 8257 are data bus buffer, read/write logic, control logic, priority resolver and four numbers of DMA channels. Each channel has two programmable 16-bit registers named as address register and count register.



Address register is used to store the starting address of memory location for DMA data transfer. The address in the address register is automatically incremented after every read/write/verify transfer. The count register is used to count the number of byte or word transferred by DMA

8a)

Mode	Bits (M1,M0)	Function
Mode 0	00	13-bit timer/counter (rarely used)
Mode 1	01	16-bit timer/counter (most common for delays)
Mode 2	10	8-bit auto-reload (ideal for periodic interrupts)
Mode 3	11	Split Timer 0 into two 8-bit timers; Timer 1 remains unchanged

8b) Registers of 8051 microcontroller are

1. General Purpose or Working Registers
2. Stack Pointer
3. Program Counter
4. Special Function Registers
5. Program Status Word (PSW)
6. Data Pointer Temporary Reg. (DPTR)

General Purpose or Working Registers:

Accumulator:

- Also referred as A-Register
- Used by all the arithmetic and logical instructions.

Special importance

- One operands is stored in it before the execution of an instruction and it also stores the result after the execution of an instruction.

B – Register:

- B-register is an 8-bit wide register.
- While multiplying, it holds one of the 8-bit operands and after the execution of the multiplication instruction, it stores the higher byte of the result.
- While dividing, it holds an 8-bit divisor and after the execution of division instruction, the remainder is stored in B-register.

Registers R0 through R7:

- There are 4 register banks each containing R0 through R7 located in ON chip RAM.
- Each of these registers is 8-bit wide.
- At a time only one bank can be selected by appropriate setting of bits in the PSW.

Stack Pointer:

- Stack Pointer of 8051 is 8-bit wide.
- It is incremented during PUSH and CALL operations and is decremented during POP and RET instructions.
- It may be initialized anywhere in the available on chip data RAM.
- After RESET, the stack pointer is initialized to 07H, causing stack to begin at 08H.

Program Counter (PC):

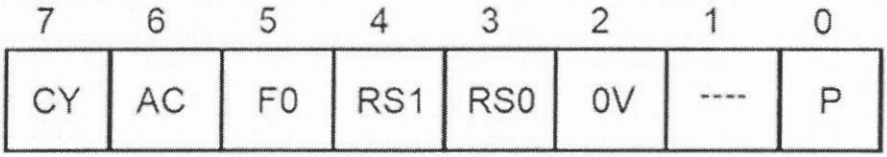
- Program counter in 8051 is 16-bit wide, and it can address 64K code bytes.
- Instruction opcode bytes are fetched from the program memory locations addressed by the program counter.
- PC always points to the instruction to be fetched and is automatically incremented after fetching instruction.
- PC is affected by call and jump instructions.

Special Function Registers (SFRs):

- The 128 bytes of on-chip additional RAM locations from 80H to 0FFH are reserved for the special functions and therefore these are called as special function registers (SFRs).
- All SFRs are directly addressable and can be read or written.

Program Status Word:

- Program status word is an 8-bit register.



Program Status Word(PSW)

Data Pointer (DPTR):

- DPTR is a 16-bit register consisting of two bytes.
  - DPH
  - DPL
  - With 16-bit pointer DPTR, a maximum of 64K of off-chip program and data memory can be addressed.

9a)

1. Data Transfer instructions
2. Arithmetic instructions

3. Logical instructions
4. Bit Manipulation instructions
5. Branching and Looping instructions

9b)

Priority can be changed by setting the bits in Interrupt Priority Register. **Interrupt Priority Register (IP)**

Size: 8-bit, bit-addressable.

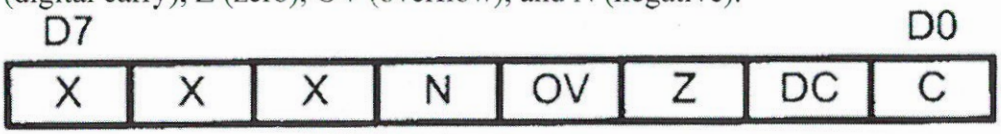
Function: Determines whether an interrupt is **high priority (1)** or **low priority (0)**.

Bit Structure of IP:

Bit	Name	Interrupt Source	Function
7	—	—	Not used
6	—	—	Not used
5	—	—	Not used
4	PS	Serial interrupt	Priority control
3	PT1	Timer 1 interrupt	Priority control
2	PX1	External interrupt 1	Priority control
1	PT0	Timer 0 interrupt	Priority control
0	PX0	External interrupt 0	Priority control

10a)

The status register is an 8-bit register. It is also referred to as the flag register. Although the status register is 8 bits wide, only 5 bits of it are used by the PIC18. The three unused bits are unimplemented and read as 0. The five flags are called conditional flags, meaning that they indicate some conditions that result after an instruction is executed. These five flags are C (carry), DC (digital carry), Z (zero), OV (overflow), and N (negative).



C – Carry flag

DC – Digital Carry flag

Z – Zero flag

OV – Overflow flag

N – Negative flag

X – D5, D6, and D7 are not implemented, and reserved for future use.

C, the carry flag

This flag is set whenever there is a carry out from the D7 bit. This flag bit is affected after an 8-bit addition or subtraction. Chapter 5 shows how the carry flag is used.

DC, the digital carry flag

If there is a carry from D3 to D4 during an ADD or SUB operation, this bit is set; otherwise, it is cleared. This flag bit is used by instructions that perform BCD (binary coded decimal) arithmetic. In some microprocessors this is called the AC flag (Auxiliary Carry flag).

Z, the zero flag

The zero flag reflects the result of an arithmetic or logic operation. If the result is zero, then Z = 1. Therefore, Z = 0 if the result is not zero. See Chapter 3 to see how we use the Z flag for looping.

OV, the overflow flag

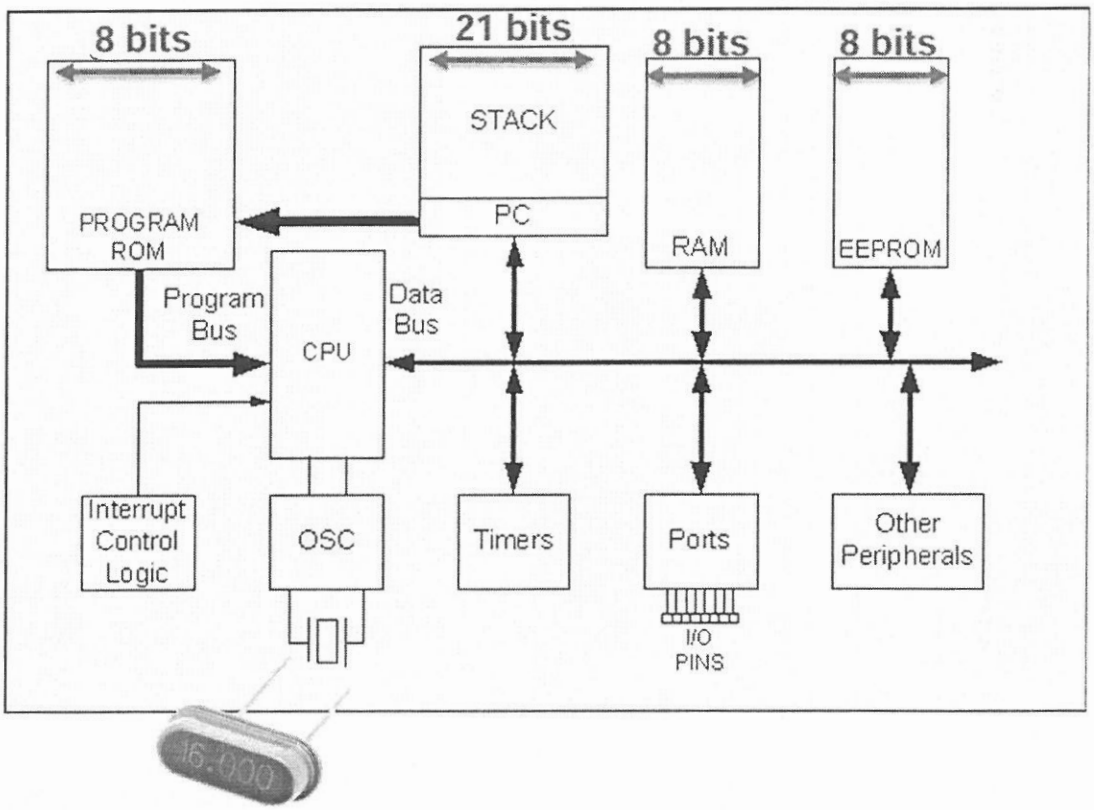
This flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into the sign bit. In general, the carry flag is used to detect errors in unsigned arithmetic operations while the overflow flag is used to detect errors in signed arithmetic operations.

N, the negative flag

Binary representation of signed numbers uses D7 as the sign bit. The negative flag reflects the result of an arithmetic operation. If the D7 bit of the result is zero, then N = 0 and the result is positive. If the D7 bit is one, then N = 1 and the result is negative. The negative and OV flag bits are used for the signed number arithmetic operations.

10b)

Basic Block Diagram of PIC18



PIC Microcontroller architecture is based on Harvard architecture and supports RISC architecture (Reduced Instruction Set Computer). PIC microcontroller architecture consists of memory organization (ram, rom, stack), CPU, timers, counter, ADC, DAC, serial communication, CCP(Captur/Compare/PWM) module and I/O ports. PIC microcontroller also supports the protocols like CAN(Controller Area Network), SPI(Serial Peripheral Interface), UART(Universal Asynchronous Reception and Transmission) for interfacing with other peripherals.

Architectural Features

8-bit CPU (Central Processing Unit)

Processes 8-bit data and instructions.

Arithmetic Logic Unit (ALU) and registers are 8-bit wide.

Modified Harvard Architecture

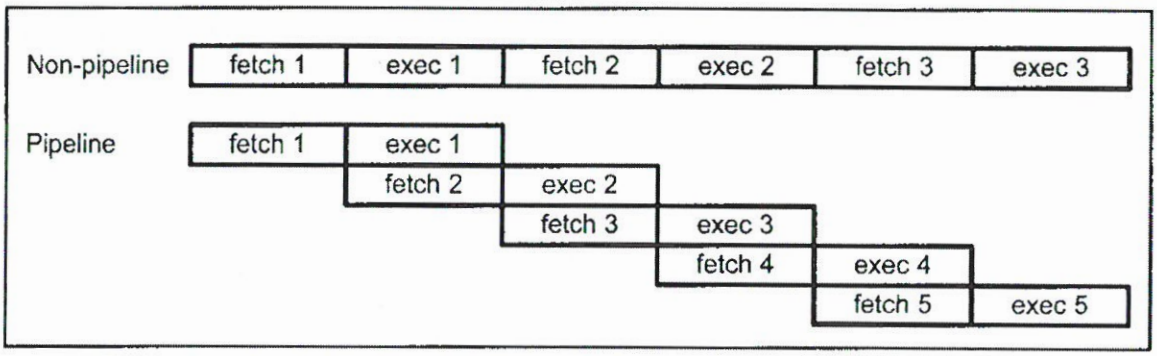
Separate program memory (ROM/Flash) and data memory (RAM).
– Separate buses for program and data → allows simultaneous instruction fetch and data access.

RISC Instruction Set

Most instructions are single-word (16 bits) and single-cycle.
– About 75 instructions total, optimized for speed.

Pipelined Execution

– Two-stage pipeline: fetch and execute.
– Next instruction is fetched while the current one executes, yielding one instruction per cycle (except for branches).



11a)

One of the goals of C18 programmers is to create smaller hex files, so it is worthwhile to re-examine C data types for C18. In other words, a good under-standing of C data types for the C18 can help programmers to create smaller hex files. In this section we focus on the specific C data types that are most useful and widely used for the PIC18 microcontroller. Table 7-1 shows data types and sizes.

Table 7-1: Some Data Types Widely Used by C18

Data Type	Size in Bits	Data Range/Usage
unsigned char	8-bit	0 to 255
char	8-bit	–128 to +127
unsigned int	16-bit	0 to 65,535
int	16-bit	–32,768 to +32,767
unsigned short	16-bit	0 to 65,535
short	16-bit	–32,768 to +32,767
unsigned short long	24-bit	0 to 16,777,215
short long	24-bit	–8,388,608 to +8,388,607
unsigned long	32-bit	0 to 4,294,967,295
long	32-bit	–2,147,483,648 to +2,147,483,648

Operators

1. Arithmetic Operators
   + (addition), - (subtraction), * (multiplication), / (division), % (modulus)
2. Relational Operators
   ==, !=, <, >, <=, >=
3. Logical Operators
   && (AND), || (OR), ! (NOT)
4. Bitwise Operators

& (AND), | (OR), ^ (XOR), ~ (NOT), << (left shift), >> (right shift)

5. Assignment Operators

=, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=

6. Increment/Decrement

++ (increment), -- (decrement)

7. Conditional / Ternary

?: → shorthand for if-else

8. Special Operators

sizeof → returns size of a data type

& → address-of operator

* → pointer dereference

11b)

```
#include <P18F458.h>
void MSDelay (unsigned int);
void main(void)
{
TRISB = 0;
PORTB = 0x55;
while (1)
{
PORTB = ~PORTB;
MSDelay (250);
}
}
void MSDelay (unsigned int itime)
{
unsigned int i;
unsigned char j;
for(i=0;i<itime;i++)
for(j=0;j<120;j++);
}
```