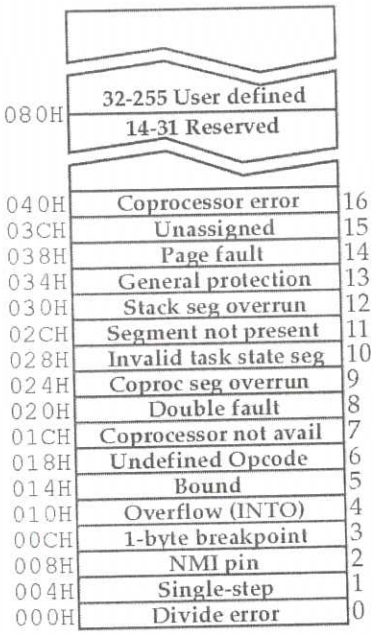
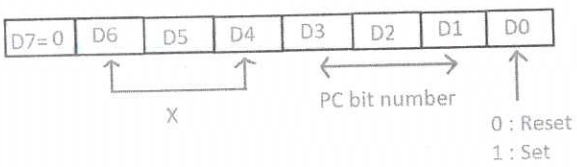
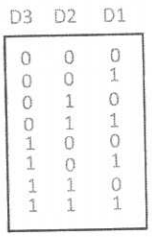
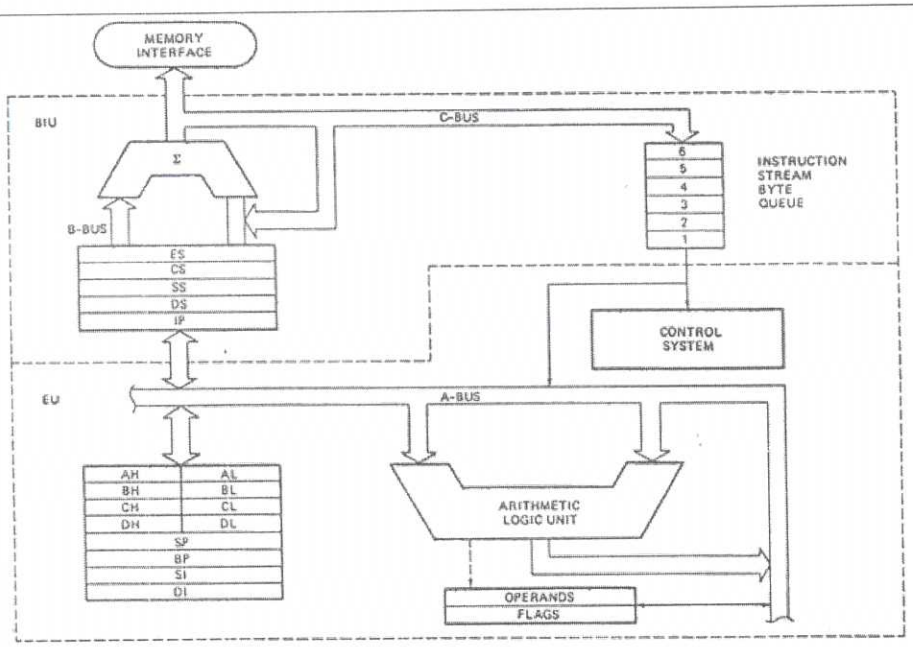
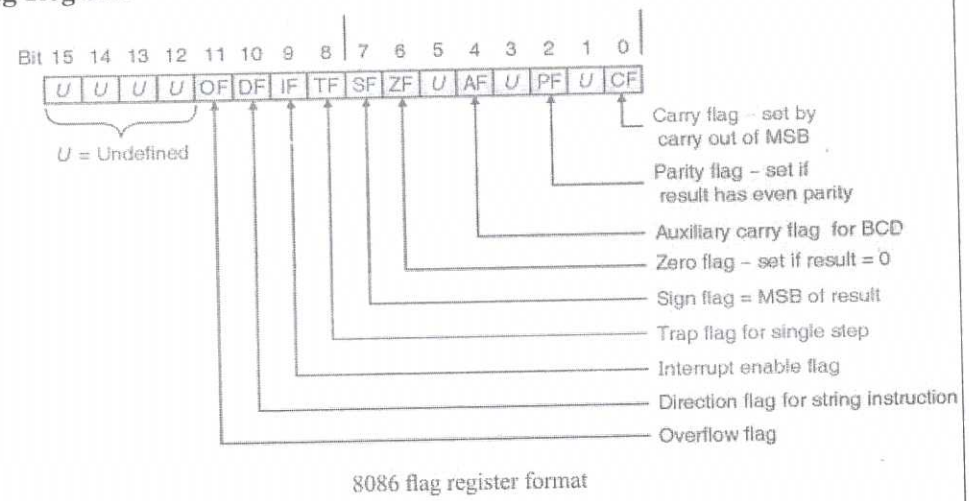
	Microprocessor and Interfacing - Rev 2015 TED 5131 ANSWER KEY		
Q no	Scoring Indicators	split score	Total score
I	**PART A**		
I.1	Keyboard and Display controller	2 x1	2
I.2	Instructions to the assembler, eg: DB,EQU,DD, Data, Segment etc	1+1	2
I.3	EAX, EBX, ECX, EDX	4 x 0.5	2
I.4	Minimum mode and maximum mode	2 x1	2
I.5	64 KB or 16 bit	2x 1	2
II			
II.1	PA = Segment base x 10H +Offset =2000 x 10 +2345 22345H	6	6
II.2	AL = 06H, CF =1, ZF =0, SF =0	6	6
II.3	Register addressing mode: Both operand are registers, eg: ADD AX, BX Direct Addressing: one operands address is given directly, eg: ADD AX,[2345]	3+3	6
II.4	ROL Instruction – Rotate left ROR Instruction – rotate Right RCL and RCR Instructions – rotate through carry Explain with examples	Any three 3 x 2	6
II.5	4 data types Packed byte - Eight bytes packed into one 64-bit quantity. Packed word - Four words packed into one 64-bit quantity. Packed doubleword - Two doublewords packed into one 64-bit quantity. Quadword - One 64-bit quantity.	4 x 1.5	6

II.6	 The interrupt vector table is locatec the first 1024 bytes of memory at addresses 000000H through 0003F. There are 256 4-byte entries (segn and offset in real mode). Seg high (Byte 3) | Seg low (Byte 2) | Offset high (Byte 1) | Offset l Byte Interrupt vector table on 8086 is a vector that consists of 256 total interrupts placed at first 1 kb of memory from 0000h to 03ffh, where each vector consists of segment and offset as a lookup or jump table to memory address of interrupt service routine. The size for each interrupt vector is 4 bytes (2 word in 16 bit), where 2 bytes (1 word) for segment and 2 bytes for offset of interrupt service routine address.	Explain with or without figure	6
II.7	**Bit set reset (BSR) mode –** This mode is used to set or reset the bits of port C only, and selected when the most significant bit (D7) in the control register is 0. This mode affects only one bit of port C at a time : Control Register is as follows:   D3,D2 and D1 are used for PORTC bit selection D0 – used set or reset the selected PORTC bit	6	6
	PART C		

III	 **Bus Interface Unit (BIU)** The Bus Interface Unit (BIU) generates the 20-bit physical memory address and provides the interface with external memory (ROM/RAM). To speed up the execution, 6-bytes of instruction are fetched in advance and kept in a 6-byte Instruction Queue while other instructions are being executed in the Execution Unit (EU). 8086's BIU produces the 20-bit physical memory address by combining a 16-bit segment address with a 16-bit offset address. There are four 16-bit segment registers, viz., the code segment (CS), the stack segment (SS), the extra segment (ES), and the data segment (DS). These segment registers hold the corresponding 16-bit segment addresses. **Execution Unit** Execution unit gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU. EU has no direct connection with system buses as shown in the above figure, it performs operations over data through BIU. It contains 1. ALU for arithmetic and logic operations 2. Flag registers for indicating the status and controls the CPU operations 3. General purpose registers : AX,BX,CX and DX . 4. Pointer registers SP, BP, SI , DI and instruction pointer.	Fig 9 exp 6	15

3/7

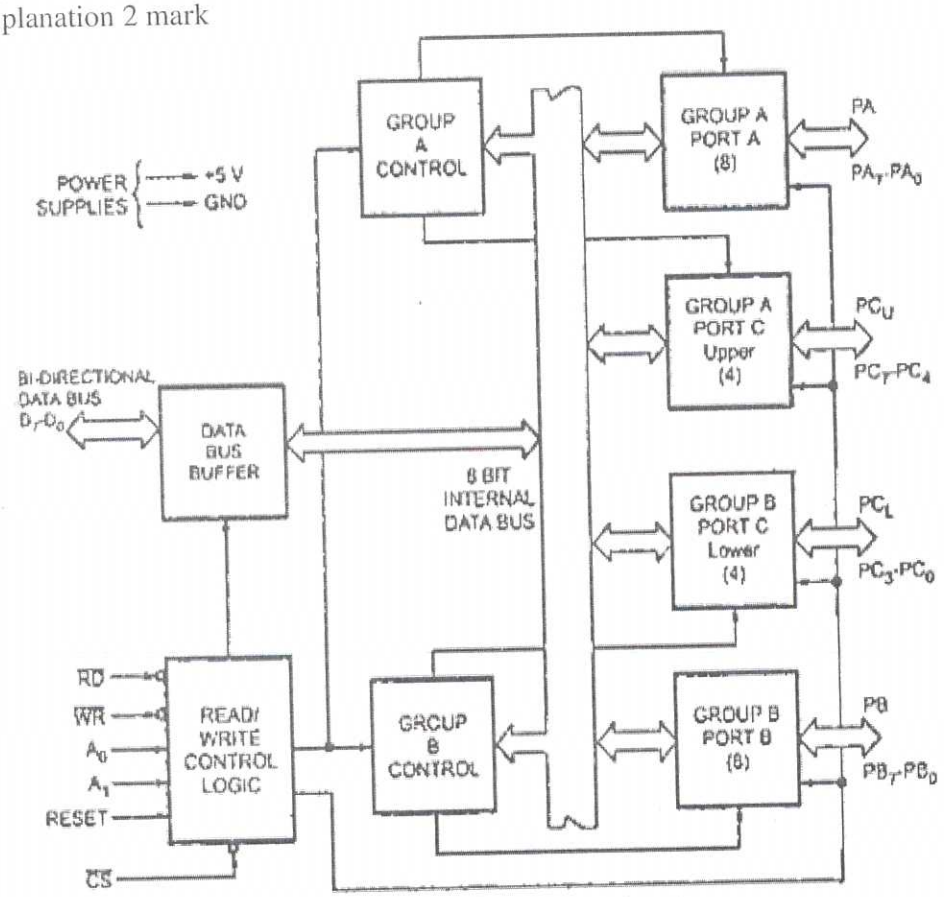
IV	**Flag Register**  8086 flag register format	figure 6 marks. Explnation 1 mark for each flag	15
V (a)	Data Transfer Instructions These instructions are used to transfer the data from the source operand to the destination operand. Following are the list of instructions under this group – MOV, PUSH, POP, XCHG, XLAT, In, OUT, LEA, LDS, LES, LAHF, SAHF, PUSHF, POPF	Explain any four 2 marks each	8

V. (b)	Sample program
	.data
	Number1 DB 10h
	Number2 DB 20 h
	SUM DB 00h
	.code
	MOV AL, [Number1]
	ADD AL, [Number2]
	MOV [SUM], AL
	
	Note: program coding style may be different with different assemblers
		Data definition – 2 , data moving 2, calculation 2, data saving 1	7

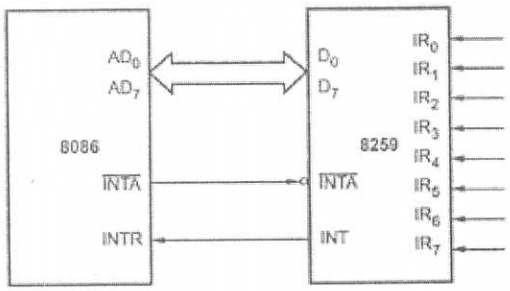
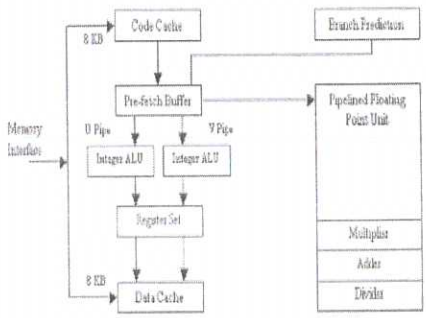
VI (a)	**NOT**– Used to invert each bit of a byte or word. **AND**– Used for adding each bit in a byte/word with the corresponding bit in another byte/word. **OR** – Used to multiply each bit in a byte/word with the corresponding bit in another byte/word. **XOR**– Used to perform Exclusive-OR operation over each bit in a byte/word with the corresponding bit in another byte/word. **TEST**– Used to add operands to update flags, without affecting operands.	Any four 4 x2	8

4/7

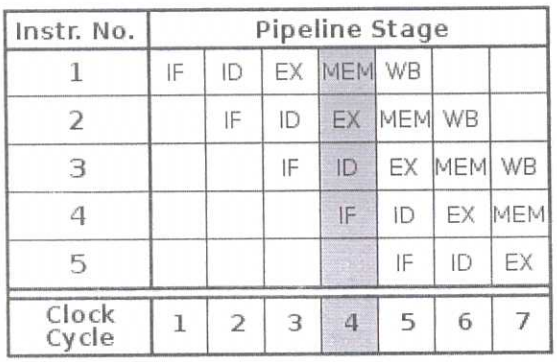
VI. (b)	.data
	Block1 DB 0123456789
	Block2 DB 0000000000
	count equ 10
	.code
	MOV CX, Count
	MOV SI, Block1
	MOV DI,Block2
	Repeat:MOVSB
	LOOP Repeat
	Note: program coding style may be different with different assemblers.
	7 marks with string handling
	5 marks without string handling
		Data definition 2, initialization of CX, SI ,DI – 2 , Data transfer part 3	7

VII. (a)	Figure 6 marks explanation 2 mark  Block Diagram of 8255	6+2	8
VII b.	Explain IO mode (mode 0,1,2) – 5 marks Explain BSR mode -2 marks **Mode 0 – Simple or basic I/O mode:** Port A, B and C can work either as input function or as output function. The outputs are latched but the inputs are not latched. It has interrupt handling capability. **Mode 1 – Handshake or strobbed I/O:**	5+2	7

5/7

	In this either port A or B can work and port C bits are used to provide handshaking. The outputs as well as inputs are latched. It has interrupt handling capability. **Mode 2 – Bidirectional I/O:** In this mode only port A will work, port B can either is in mode 0 or 1 and port C bits are used as handshake signal. The outputs as well as inputs are latched. It has interrupt handling capability. **Bit set reset (BSR) mode –** This mode is used to set or reset the bits of port C only, and selected when the most significant bit (D7) in the control register is 0.		
VIII (a)	FIGURE 6 EXPLANATION2  Connection between 8086 and 8259	6+2	8
VIII (b)	EXPLAIN PRIORITY OF INTERRUPT 7 internal –>NMI - →>Software -- →>External (32-255) With in a group, the lower type number has highest priority	7	7
IX a	FIGURE 6 EXPLANATION2  Architecture of Pentium	6+2	8
b)	Different modes of operation 80386 1. real mode – works like a faster 8086, address limited to 1MB space, 32 bit instructions and registers are used,. On reset the cpu works in real mode. Tis mode is used for initialization of several memory tables and flags. 2. protected mode - All the capabilities of 80386 are available for utilization in its protected mode of operation. The 80386 in protected mode support all the software written for 80286 and 8086 to be executed under the control of memory management and protection abilities of 80386. The protected mode allows the use of additional instruction, addressing modes and capabilities of 80386.	2.5 +2.5+2	7

6/7

	3. virtual 8086 mode – This mode allows the system to create one or more virtual 8086 tasks. This feature allows a 386 based computer to run multiple OS, each one located in its own 8086 environment		
X a	**FIGURE 6EXPLANATION 2**  	6+2	8
X b	**Pipe line Hazards** Pipeline hazards are situations that prevent the next instruction in the instruction stream from executing during its designated clock cycles. Any condition that causes a stall in the pipeline operations can be called a hazard. There are primarily three types of hazards: i. Data Hazards ii. Control Hazards or instruction Hazards iii. Structural Hazards. i. Data Hazards: A data hazard is any condition in which either the source or the destination operands of an instruction are not available at the time expected in the pipeline. As a result of which some operation has to be delayed and the pipeline stalls **ii. Structural Hazards:** This situation arises mainly when two instructions require a given hardware resourceat the same time and hence for one of the instructions the pipeline needs to be stalled. The most common case is when memory is accessed at the same time by two instructions. **iii. Control hazards:** The instruction fetch unit of the CPU is responsible for providing a stream of instructions to the execution unit. The instructions fetched by the fetch unit are in consecutive memory locations and they are executed. However the problem arises when one of the instructions is a branching instruction to some other memory location. Thus all the instruction fetched in the pipeline from consecutive memory locations are invalid now and need to removed(also called flushing of the pipeline).This induces a stall till new instructions are again fetched from the memory address specified in the branch instruction.	Listing 1, explanation of each 2 1+2+2+2	7

7/7