F.Y. B.Sc.(IT) : Sem. II

Microprocessor Architecture

Time : 2½ Hrs.] Prelim Question Paper Solution **[Marks : 75**

Q.1 Attempt the following (any THREE) [15]

Q.1(a) Explain following pins of 8085. [5]

INTA, RESET IN, RESET OUT, READY, HOLD and HLDA

Ans.:

- **INTA :** It is an interrupt acknowledgment signal.
- **RESET IN :** This signal is used to reset the microprocessor by setting the program counter to zero.
- **RESET OUT :** This signal is used to reset all the connected devices when the microprocessor is reset.
- **READY :** This signal indicates that the device is ready to send or receive data. If READY is low, then the CPU has to wait for READY to go high.
- **HOLD :** This signal indicates that another master is requesting the use of the address and data buses.
- **HLDA (HOLD Acknowledge) :** It indicates that the CPU has received the HOLD request and it will relinquish the bus in the next clock cycle. HLDA is set to low after the HOLD signal is removed.

Q.1(b) Explain Flag Flip Flop of 8085. [5]

Ans.: **Flag register :** It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator. These are the set of 5 flip-flops :

1. Sign (S)
2. Zero (Z)
3. Auxiliary Carry (AC)
4. Parity (P)
5. Carry (C)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

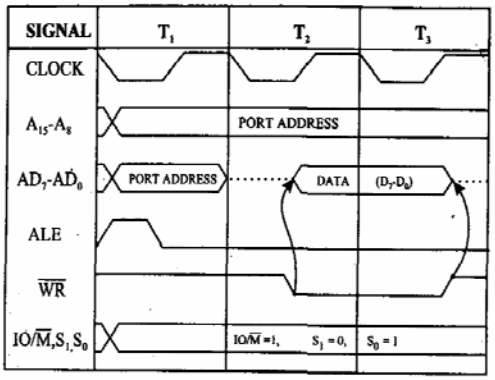
Fig.: Bit positions of various flags in the flag register of 8085.

Its bit position is shown in the following table :

Carry flag	This flag indicates an overflow condition for arithmetic operations.
Auxiliary flag	When an operation is performed at ALU, it results in a carry/barrow from lower nibble (i.e. D_0-D_3) to upper nibble (i.e. D_4-D_7), then this flag is set, i.e. carry given by D_3 bit to D_4 is AF flag. The processor uses this flag to perform binary to BCD conversion.
Parity flag	This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1's, then the Parity Flag is set. For odd number of 1's, the Parity Flag is reset.
Zero flag	This flag is set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.
Sign flag	This flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.

Q.1(c) Draw memory write cycle. [5]

Ans.:



Q.1(d) Write short notes on : [5]

1. Program Counter 2. Stack Pointer 3. Accumulator

Ans.: 1. Program Counter :

- It is a 16 bit register.
- Used by microprocessor.
- It holds the address of next instruction to be executed. For example, after execution MOV A,B instruction program counter will be automatically incremented and points next location of memory where another data to stored.

2. Stack Pointer :

- It is used when interrupt is generated by microprocessor interrupt is a command which stop the main program counter location will be stored in a stack and program counter will be loaded with a new address through which sub routine is called.
- Sub routine is a small program which is used many times by main program. For example, Generating delay in main program after execution of each instruction.

3. Accumulator :

- It is one of the general purpose register of microprocessor also called as A register. The **accumulator** is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the **accumulator**.

Q.1(e) What are the Features of Microprocessor 8085? [5]

Ans.:

- It was introduced in 1977.
- It is 8-bit microprocessor.
- Its actual name is 8085 A.
- It is single NMOS device.
- It contains 6200 transistors approx.
- Its dimensions are 164 mm x 222 mm.
- It is having 40 pins Dual-Inline-Package (DIP).
- It has three advanced versions:
  - 8085 AH
  - 8085 AH2
  - 8085 AH1
- These advanced versions are designed using HMOS technology.
- The advanced versions consume 20% less power supply.
- The microprocessor is one of the most important components of a digital computer.
- It acts as the brain of the computer system.
- As technology has progressed, microprocessors have become faster, smaller and capable of doing more work per clock cycle.
- Sometimes, microprocessor is written as μP. (μ is pronounced as Mu).

Q.1(f) What is word, nibble, assembly language and machine language. [5]

Ans.: Word :

Byte: The byte is a unit of digital information that most commonly consists of eight bits. Historically, the byte was the number of bits used to encode a single character of text in a computer and for this reason it is the smallest addressable unit of memory in many computer architectures.

Nibble: In computing, a nibble (often nybble or nyble to match the spelling of byte) is a four-bit aggregation, or half an octet.

Machine Language: Programs written in high-level *languages* are translated into assembly *language* or *machine language* by a compiler. Assembly *language* programs are translated into *machine language* by a program called an assembler. Every CPU has its own unique *machine language*.

Assembly Language: An ***assembly*** (or ***assembler***) ***language***, often abbreviated asm, is a low-level programming ***language*** for a computer, or other programmable device, in which there is a very strong (but often not one-to-one) correspondence between the ***language*** and the architecture's machine code instructions.

Q.2 Attempt the following (any THREE) [15]

Q.2(a) WAP to perform addition of two 8 bit numbers using 8085. [5]

Ans.: Algorithm :

1) Start the program by loading the first data into Accumulator.
2) Move the data to a register (B register).
3) Get the second data and load into Accumulator.
4) Add the two register contents.
5) Check for carry.
6) Store the value of sum and carry in memory location.
7) Terminate the program.

Program :

```
MVI C, 00          'Initialize C register to 00
LDA 4150           'Load the value to Accumulator.
MOV B, A           'Move the content of Accumulator to B register.
LDA 4151           'Load the value to Accumulator.
ADD B              'Add the value of register B to A
JNC LOOP           'Jump on no carry.
INR C              'Increment value of register C
LOOP: STA 4152     'Store the value of Accumulator (SUM).
MOV A, C           'Move content of register C to Acc.
STA 4153           'Store the value of Accumulator (CARRY)
HLT                'Halt the program.
```

Observation :

```
Input  :  80 (4150)
          80 (4251)
Output :  00 (4152)
          01 (4153)
```

Q.2(b) List different addressing modes used by 8085 microprocessor. Write any one 1byte and any one 2 byte instruction to perform arithmetic operaiton using 8085 microprocessor. [5]

Ans.: Addressing Modes in 8085 :

These are the instructions used to transfer the data from one register to another register, from the memory to the register, and from the register to the memory without any alteration in the content. Addressing modes in 8085 is classified into 5 groups :

- **Immediate Addressing Mode :** In this mode, the 8/16-bit data is specified in the instruction itself as one of its operand. For example : MVI K, 20F: means 20F is copied into register K.
- **Register addressing mode :** In this mode, the data is copied from one register to another. For example : MOV K, B: means data in register B is copied to register K.

- **Direct addressing mode :** In this mode, the data is directly copied from the given address to the register. For example : LDB 5000K: means the data at address 5000K is copied to register B.
- **Indirect addressing mode :** In this mode, the data is transferred from one register to another by using the address pointed by the register. For example : MOV K, B: means data is transferred from the memory address pointed by the register to the register K.
- **Implied addressing mode :** This mode doesn't require any operand; the data is specified by the opcode itself. For example : CMP.

Q.2(c) What is instruction and what are the type of instruction? [5]

Ans.: **Instruction :** An instruction set is a group of commands for a CPU in machine language. The term can refer to all possible instructions for a CPU or a subset of instructions to enhance its performance in certain situations.

Instruction word size: A **word** is a fixed-**sized** piece of data handled as a unit by the **instruction** set or the hardware of the processor. Modern processors, including embedded systems, usually have a **word size** of 8, 16, 24, 32, or 64 bits, while modern general purpose computers usually use 32 or 64 bits.

Type of instruction based on size :

Depending upon the size of machine codes, the 8085 instructions are classified into three types :

(a) One-byte instructions : A 1 byte instruction include the opcode and the operand in the 8 bits only which is one byte.

Example : 1. MOV C, A Hex code = 4FH (one byte)
2. ADD B Hex code = 80H (one byte)
3. CMA Hex code = 2FH (one byte)

(b) Two-byte instructions : The two byte instruction is one which contains an 8-bit op-code and 8-bit operand (Data).

Example : 1. MVI A, 09 Hex code = 3E, 09 (two bytes)
2. ADD B, 07 Hex code = 80, 07 (two bytes)
3. SUB A, 05 Hex code = 97, 05 (two bytes)

(c) Three-byte instructions : In a three byte instruction the first byte is opcode and second and third bytes are operands i.e. 16-bit data or 16-bit address.

Example : 1. LDA 8509 Hex code = 3A, 09, 85 (Three bytes)
2. LXI 2500 Hex code = 21, 00, 25 (Three bytes)
3. STA 2600 Hex code = 32, 00, 26 (Three bytes)

Q.2(d) List down Arithmetic Instruction. [5]

Ans.: Following is the table showing the list of arithmetic instructions:

OPCODE	OPERAND	EXPLANATION	EXAMPLE
ADD	R	A = A + R	ADD B
ADD	M	A = A + Mc	ADD 2050
ADI	8–bit data	A = A + 8–bit data	ADD 50
ADC	R	A = A + R + Prev. carry	ADC B
ADC	M	A = A + Mc + Prev. carry	ADC 2050
ACI	8–bit data	A = A + 8–bit data + Prev. carry	ACI 50
SUB	R	A = A – R	SUB B
SUB	M	A = A – Mc	SUB 2060
SUI	8–bit data	A = A – 8–bit data	SUI 50
SBB	R	A = A – R – Prev. carry	SBB B
SBB	M	A = A – MC – Prev. carry	SBB 2050

SBI	8–bit data	A = A – 8–bit data – Prev. carry	SBI 50
INR	R	R = R + 1	INR B
INR	M	M = Mc + 1	INR 2050
INX	r. p.	r. p = r. p + 1	INX H
DCR	R	R = R – 1	DCR B
DCR	M	M = Mc – 1	DCR 2050
DCX	r. p.	r. p. = r. p. – 1	DCX H
DAD	r. p.	HL = HL + r. p.	DAD H

In the table,
R stands for register
M stands for memory
Mc stands for memory contents
r.p. stands for register pair

Q.2(e) Compare memory mapped I/O and I/O mapped I/O. [5]

Ans.:

Memory mapped I/O	I/O mapped I/O
1. In this device address is 16–bit. Thus A0 to A15 lines are used to generate the device address.	1. In this device address is 8–bit. Thus to A7 or A8 to A15 lines are used to generate device address.
2. MEMR and MEMW control signals are used to control read and write I/O operations.	2. IOR and IOW control signals are us to control read and write I/O operation.
3. Instructions available are LDA, STA, MOV R. M. ADD M etc.	3. Instructions available are IN and OUT.
4. Data transfer is between any register and I/O device.	4. Data transfer is between accumulator and I/O device.
5. Decoding 16–bit address may require more hardware.	5. Decoding 8–bit address will require less hardware.

Q.2(f) Short note on CALL instruction. [5]

Ans.: The call instruction transfers the program sequence to the memory address given in the operand. Before transferring, the address of the next instruction after CALL is pushed onto the stack. Call instructions are 2 types: Unconditional Call Instructions and Conditional Call Instructions.

(a) **Unconditional Call Instructions:** It transfers the program sequence to the memory address given in the operand.

OPCODE	OPERAND	Explanation	Example
CALL	address	Unconditionally calls	CALL 2050

(b) **Conditional Call Instructions:** Only if the condition is satisfied, the instructions executes.

OPCODE	OPERAND	Explanation	Example
CC	address	Call if carry flag is 1	CC 2050
CNC	address	Call if zero flag is 0	CNC 2050
CZ	address	Calls if zero flag is 1	CZ 2050
CNZ	address	Calls if zero flag is 0	CNZ 2050
CPE	address	Calls if carry flag is 1	CPE 2050
CPO	address	Calls if carry flag is 0	CPO 2050
CM	address	Calls if sign flag is 1	CM 2050
CP	address	Calls if zero flag is 0	CP 2050

Q.3 Attempt the following (any THREE) : [15]

Q.3(a) Explain how 8085 microprocessor performs logical operation of comparing two data. [5]

Ans.: 1. **CMP (compare register or memory with accumulator):** The contents of the operand register or memory are M compared with the contents of the accumulator. Both contents are preserved . The result of the comparison is shown by setting the flags of the PSW as follows:
if (A) <reg/mem: carry flag is set.
if (A) = reg/mem:
zero flag is set.
if (A) >reg/mem: carry and zero flags are reset.
e.g.: CMP B
CMP M

2. **CPI (compare immediate with accumulator):** The second byte (8-bit data) is compared with the contents of the accumulator. The values being compared remain unchanged. The result of the comparison is shown by setting the flags of the PSW as follows:
if (A) < data: carry flag is set
if (A) = data: zero flag is set
if (A) > data: carry and zero flags are reset
e.g., CPI 89H

Q.3(b) What is time delay? Why is it needed? Explain how time delay can be generated using a register pair. [5]

Ans.: **Time delay** : A delay used to separate the occurrence of two events, especially in a mechanical or electronic device.

Time delay can be generated using a register pair
LXI B, count : 16 - bit count
BACK: DCX B : Decrement count
MOV A, C
ORA B : Logically OR Band C
JNZ BACK : If result is not zero repeat

Q.3(c) WAP to generate a delay of 0.4 seconds. [5]

Ans.: Calculation : In 8085, the operating frequency is half of the crystal frequency,
i.e., Operating frequency = 5/2 = 2.5 MHz
Time for one T -state = Number of T-states required
= 1 x 106
Source Program:
LXI B, count : 16 - bit count
BACK: DCX B : Decrement count
MOV A, C
ORA B : Logically OR Band C
JNZ BACK : If result is not zero repeat

Q.3(d) Explain the following concepts for subroutine program [5]
(i) Nesting (ii) Multiple Ending Subroutine

Ans.: **(i) Nesting:** In computer programming, a **nested function** (or **nested procedure** or **subroutine**) is a function which is defined within another function, the *enclosing function*. Due to simple recursive scope rules, a nested function is itself invisible outside of its immediately enclosing function, but can see (access) all local objects (data,

functions, types, etc.) of its immediately enclosing function as well as of any function(s) which, in turn, encloses that function. The nesting is theoretically possible to unlimited depth, although only a few levels are normally used in practical programs.

(ii) Multiple Ending Subroutine : *MULTIPLE ENDING SUBROUTINE"* is a subroutine that because of conditions and test has more that one return point:

```
code...
...
JNC OTHER
...
RET
OTHER: ...
...
RET
```

Q.3(e) WAP of 8085 to multiply two 8 bit numbers using logical instructions. [5]

Ans.: Algorithm :

1. Assign the value 05 to register B
2. Assign the value 04 to register C
3. Move the content of B in A
4. Rotate accumulator left without carry
5. Rotate accumulator left without carry
6. Store the content of accumulator at memory address 3050
7. Halt of the program

Program :

MEMORY ADDRESS	MNEMONICS	COMMENTS
2000	MVI B 05	B <– 05
2002	MVI C 04	C <– 04
2004	MOV A, B	A <– B
2005	RLC	rotate the content of A without carry
2006	RLC	rotate the content of A without carry
2007	STA 3050	3050 <– A
200A	HLT	End of the program

Explanation :

1. **MVI B 05:** assign the value 05 to B register.
2. **MVI C 04:** assign the value 04 to C register.
3. **MOV A, B:** move the content of register B to register A.
4. **RLC:** rotate the content of accumulator left without carry.
5. **RLC:** rotate the content of accumulator left without carry.
6. **STA 3050:** store the content of register A at memory location 3050
7. **HLT:** stops the execution of the program.

Q.3(f) Short note on stack. [5]

Ans.: In computer science, a stack is an abstract data type that serves as a collection of elements, with two principal operations:

- **push**, which adds an element to the collection, and
- **pop**, which removes the most recently added element that was not yet removed.

The order in which elements come off a stack gives rise to its alternative name, **LIFO** (**last in, first out**). Additionally, a peek operation may give access to the top without modifying the stack.

The name "stack" for this type of structure comes from the analogy to a set of physical items stacked on top of each other, which makes it easy to take an item off the top of the stack, while getting to an item deeper in the stack may require taking off multiple other items first.

Considered as a linear data structure, or more abstractly a sequential collection, the push and pop operations occur only at one end of the structure, referred to as the top of the stack. This makes it possible to implement a stack as a singly linked list and a pointer to the top element.

A stack may be implemented to have a bounded capacity. If the stack is full and does not contain enough space to accept an entity to be pushed, the stack is then considered to be in an overflow state. The pop operation removes an item from the top of the stack.

Q.4 Attempt the following (any THREE) **[15]**

Q.4(a) What do you mean by vectored interrupts? Discuss each of 8085 vectored interrupt in brief. **[5]**

Ans.: Vectored Interrupt : In vectored interrupts, the processor automatically branches to the specific address in response to an interrupt.
The vector addresses of software interrupts are given in table below.

Interrupt	Vector address
RST 0 RST 1	0000_H 0008_H
RST 2 RST 3	0010_H 0018_H
RST 4 RST 5	0020_H 0028_H
RST 6 RST 7	0030_H 0038_H

Interrupt	Vector address
RST 7.5	$003C_H$
RST 6.5	0034_H
RST 5.5	$002C_H$
TRAP	0024_H

Q.4(b) What is cross assembler and loader? **[5]**

Ans.: Cross Assembler :

An **assembler** is a program that converts assembly language ("human readable" text - if you are a nerd) into the actual binary processor specific machine code (non-human readable binary code - unless you are a nerd). Normally the machine code generated is for the processor used in the machine it is run on.

A **cross assembler** takes this conversion process a step further by allowing you to generate Machine code for a *different* processor than the one the compiler is run on.

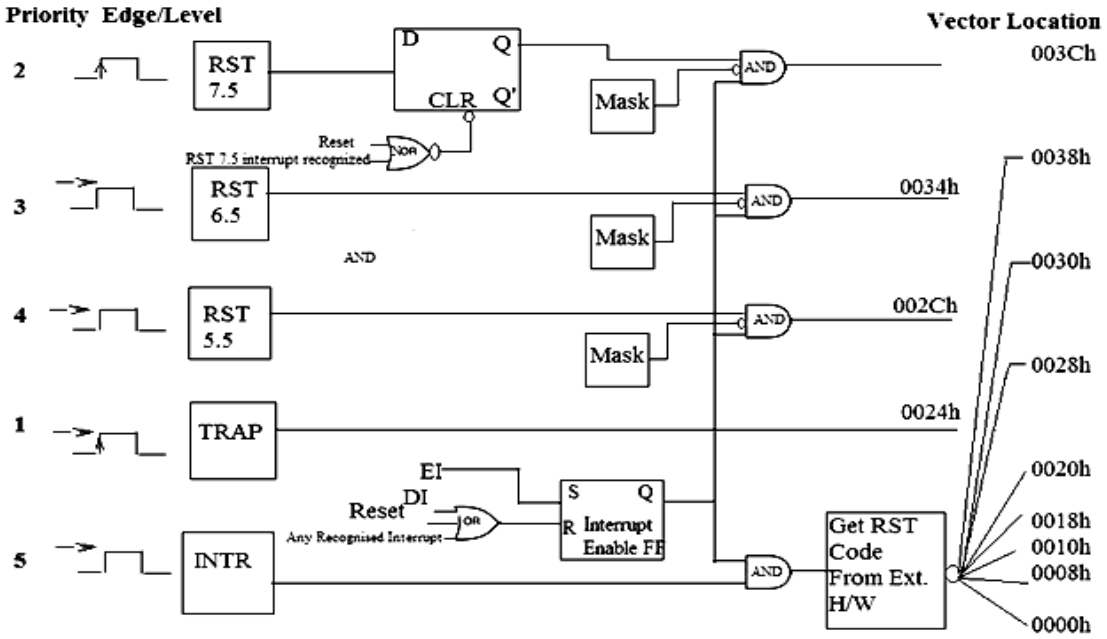
Cross assemblers are generally used to develop programs which are supposed to run on game consoles, appliances and other specialized small electronics systems which are not able to run a development environment. They can also be used to speed up development for low powered system, for example XAsm enables development on a PC based system for a Z80 powered MSX computer. Even though the MSX system is capable of running an assembler, having the additional memory, processor speed and storage capabilities like a harddisk significantly speeds up development efforts.

Loader :

In computing, a ***loader*** is the part of an operating system that is responsible for loading programs and libraries. It is one of the essential stages in the process of starting a program, as it places programs into memory and prepares them for execution.

Q.4(c) Short note on Interrupt structure of 8085. **[5]**

Ans.:



- An external device initiates the hardware interrupts and placing an appropriate signal at the interrupt pin of the processor.
- If the interrupt is accepted then the processor executes an interrupt service routine.

Five Hardware Interrupts in 8085 :

1. TRAP 2. RST 7.5 3. RST 6.5 4. RST 5.5 5. INTR

1. **TRAP** : Pin 6 (Input)
   - It is an non-maskable interrupt.
   - It has the highest priority.
   - It cannot be disabled.
   - It is both edge and level triggered.
   - It means TRAP signal must go from low to high.
   - And must remain high for a certain period of time.
   - TRAP is usually used for power failure and emergency shutoff.
2. **RST 7.5** : Pin 7 (Input)
   - It is a maskable interrupt.
   - It has the second highest priority.
   - It is positive edge triggered only.
   - The internal flip-flop is triggered by the rising edge.
   - The flip-flop remains high until it is cleared by RESET IN.
3. **RST 6.5** : Pin 8 (Input)
   - It is a maskable interrupt.
   - It has the third highest priority.
   - It is level triggered only.
   - The pin has to be held high for a specific period of time.
   - RST 6.5 can be enabled by EI instruction.
   - It can be disabled by DI instruction.
4. **RST 5.5** : Pin 9 (Input)
   - It is a maskable interrupt.
   - It has the fourth highest priority.
   - It is also level triggered.
   - The pin has to be held high for a specific period of time.

- This interrupt is very similar to RST 6.5.

5. **INTR** : Pin 10 (Input)
   - It is a maskable interrupt.
   - It has the lowest priority.
   - It is also level triggered.
   - It is a general purpose interrupt.
   - By general purpose we mean that it can be used to vector microprocessor to any specific subroutine having any address.

Q.4(d) WAP for BCD addition program. **[5]**

Ans.: Algorithm :

1. Load 00H in a register (for carry)
2. Load content from memory into register pair
3. Move content from L register to accumulator
4. Add content of H register with accumulator
5. Add 06H if sum is greater than 9 or Auxillary Carry is not zero
6. If carry flag is not equal to 1, go to step 8
7. Increment carry register by 1
8. Store content of accumulator into memory
9. Move content from carry register to accumulator
10. Store content of accumulator into memory
11. Stop

Program :

MEMORY	MNEMONICS	OPERANDS	COMMENT
2000	MVI	C, 00H	[C] <– 00H, carry
2002	LHLD	[2500]	[H–L] <– [2500]
2005	MOV	A, L	[A] <– [L]
2006	ADD	H	[A] <– [A] + [H]
2007	DAA		Add 06 if sm > 9 or AC = 1
2008	JNC	200C	Jump if no carry
200B	INR	C	[C] <– [C] + 1
200C	STA	[2502]	[A] –> [2502], Sum
200F	MOV	A, c	[A] <– [C]
2010	STA	[2503]	[A] –> [2503], Carry
2013	HLT		Stop

Explanation : Registers A, C, H, L are used for general purpose

1. **MVI** is used to move data immediately into any of registers (2 Byte)
2. **LHLD** is used to load register pair direct using 16-bit address (3 Byte instruction)
3. **MOV** is used to transfer the data from memory to accumulator (1 Byte)
4. **ADD** is used to add accumulator with any of register (1 Byte instruction)
5. **STA** is used to store data from accumulator into memory address (3 Byte instruction)
6. **DAA** is used to check if sum > 9 or AC = 1 add 06 (1 Byte instruction)
7. **JNC** is used jump if no carry to given memory location (3 Byte instruction)
8. **INR** is used to increase given register by 1 (1 Byte instruction)
9. **HLT** is used to halt the program

Q.4(e) WAP for addition of two 16 bits number. **[5]**

Ans.: Algorithm :

1. Load both the lower and the higher bits of first number at once
2. Copy the first number to another register pair

3. Load both the lower and the higher bits of second number at once
4. Add both the register pairs and store the result in a memory location

Memory address	Mnemonics	Comments
2000	LHLD 2050	A ← 2050
2003	XCHG	D ← H & E ← L
2004	LHLD 2052	A ← 2052
2007	DADD	H ← H + D & L ← L + E
2008	SHLD 3050	A → 3050
200B	hlt	Stops execution

Explanation :

1. **LHLD 2050** loads the value at 2050 in L register and that in 2051 in H register (first number)
2. **XCHG** copies the content of H to D register and L to S register
3. **LHLD 2052** loads the value at 2052 in L register and that in 2053 in H register (second number)
4. **DAD D** adds the value of H with D and L with E and stores the result in H and L
5. **SHLD 3050** stores the result at memory location 3050
6. **HLT** stops execution

Q.4(f) Explain LHLD, SHLD, XCHG and XTHL. [5]

Ans.: **LHLD: LHLD(Load H and L register direct): -** this instruction loads the contents of the 16- bit memory location into the HL register pair.
Eg: LHLD 3000H (the content of location 3000h is copied into the HL reg pair)

SHLD: SHLD(store H and L register direct): - The contents of register L are stored into the memory location specified by the 16-bit address in the operand and the contents of H register are stored into the next memory location by incrementing the operand. The contents of registers HL are not altered. This is a 3-byte instruction, the second byte specifies the low-order address and the third byte specifies the high-order address.
Eg: SHLD 3000H

XCHG: Exchange H and L with D and E. The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.
Eg: XCHG

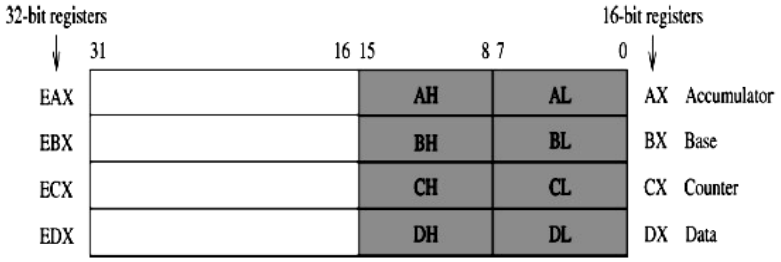
XTHL: This instruction exchanges H and L with top of stack. The contents of the L register are exchanged with the stack location pointed out by the contents of the stack pointer register. The contents of the H register are exchanged with the next stack location (SP+1); however, the contents of the stack pointer register are not altered.
Eg: XTHL

Q.5Attempt the following (any THREE) [15]

Q.5(a) What are different types of special Pentium registers? Describe them in brief. [5]

Ans.: **Pentium Registers :**

- Four 32–bit registers can be used as
- Four 32–bit register (EAX, EBX, ECX, EDX)
- Four 16–bit register (AX, BX, CX, DX)
- Eight 18–bit register (AH, AL, BH, BL, CH, CL, DH, DL)
- Some registers have special use
- ECX for count in loop instructions



Q.5(b) Compare i3, i5,i7. **[5]**

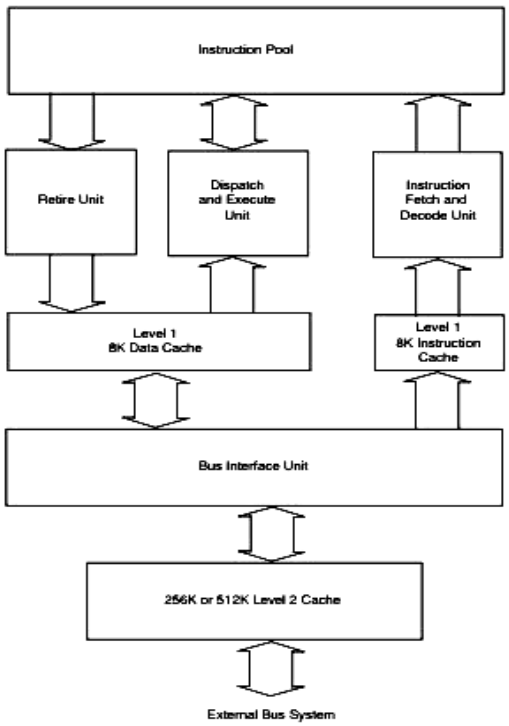
Ans.:

Sr. No.	Core i3	Core i5	Core i7
(i)	Entry level processor.	Mid range processor.	High end processor.
(ii)	2-4 Cores	2-4 Cores	4 Cores
(iii)	4 Threads	4 Threads	8 Threads
(iv)	Hyper-Threading (efficient use of processor resources)	Hyper-Threading (efficient use of processor resources)	Hyper-Threading (efficient use of processor resources)
(v)	3-4 MB Cache	3-8 MB Cache	4-8 MB Cache
(vi)	32 nm Silicon (less heat and energy)	32-45 nm Silicon (less heat and energy)	32-45 nm Silicon (less heat and energy)
(vii)		Turbo Mode (turn off core if not used)	Turbo Mode (turn off core if not used)

Q.5(c) Short note on Pentium pro processor.

Ans.:

FIGURE 18–13 The internal structure of the Pentium Pro microprocessor.



The Pentium Pro is structured differently than earlier microprocessors. Early microprocessors contained an execution unit and a bus interface unit with a small cache buffering the execution unit for the bus interface unit. This structure was modified in later microprocessors, but the modifications were just additional stages within the microprocessors. The Pentium architecture is also a modification, but more significant than

earlier microprocessors. Figure 18–13 shows a block diagram of the internal structure of the Pentium Pro microprocessor.

The system buses, which communicate to the memory and I/O, connect to an internal level 2 cache that is often on the main board in most other microprocessor systems. The level 2 cache in the Pentium Pro is either 256K bytes or 512K bytes. The integration of the level 2 cache speeds processing and reduces the number of components in a system.

The bus interface unit (BIU) controls the access to the system buses through the level 2 cache, as it does in most other microprocessors. Again, the difference is that the level 2 cache is integrated. The BIU generates the memory address and control signals, and passes and fetches data or instructions to either a level 1 data cache or a level 1 instruction cache. Each cache is 8K bytes in size at present and may be made larger in future versions of the microprocessor. Earlier versions of the Intel microprocessor contained a unified cache that held both instructions and data. The implementation of separate caches improves performance because data-intensive pro- grams no longer fill the cache with data.

Q.5(d) List some instruction of Pentium processor. [5]

Ans.:

Data Transfer Instructions

MOV	Move
CMOVE/CMOVZ	Conditional move if equal/Conditional move if zero
CMOVNE/CMOVNZ	Conditional move if not equal/Conditional move if not zero
CMOVA/CMOVNBE	Conditional move if above/Conditional move if not below or equal
CMOVAE/CMOVNB	Conditional move if above or equal/Conditional move if not below
CMOVB/CMOVNAE	Conditional move if below/Conditional move if not above or equal

Binary Arithmetic Instructions

ADD	Integer add
ADC	Add with carry
SUB	Subtract
SBB	Subtract with borrow
IMUL	Signed multiply
MUL	Unsigned multiply
IDIV	Signed divide
DIV	Unsigned divide
INC	Increment
DEC	Decrement
NEG	Negate
CMP	Compare

Decimal Arithmetic

DAA	Decimal adjust after addition
DAS	Decimal adjust after subtraction
AAA	ASCII adjust after addition
AAS	ASCII adjust after subtraction
AAM	ASCII adjust after multiplication
AAD	ASCII adjust before division

Q.5(e) Discuss the SYSENTER and SYSEXIT instructions of Pentium II Processor. [5]

Ans.: The SYSENTER instruction is part of the "Fast System Call" facility introduced on the Pentium® II processor. The SYSENTER instruction is optimized to provide the maximum performance for transitions to protection ring 0 (CPL = 0). The SYSENTER instruction sets the following registers according to values specified by the operating system in certain model-specific registers.

- CS register set to the value of (SYSENTER_CS_MSR)
- EIP register set to the value of (SYSENTER_EIP_MSR)
- SS register set to the sum of (8 plus the value in SYSENTER_CS_MSR)
- ESP register set to the value of (SYSENTER_ESP_MSR)

Looks like processor is trying to help us. Let's look at SYSEXIT also very quickly:

The SYSEXIT instruction is part of the "Fast System Call" facility introduced on the Pentium® II processor. The SYSEXIT instruction is optimized to provide the maximum performance for transitions to protection ring 3 (CPL = 3) from protection ring 0 (CPL = 0). The SYSEXIT instruction sets the following registers according to values specified by the operating system in certain model-specific or general purpose registers.

- CS register set to the sum of (16 plus the value in SYSENTER_CS_MSR)
- EIP register set to the value contained in the EDX register
- SS register set to the sum of (24 plus the value in SYSENTER_CS_MSR)
- ESP register set to the value contained in the ECX register

Q.5(f) Describe the memory management in Pentium and Pentium pro- processors. [5]

Ans.: The memory-management unit within the Pentium is upward-compatible with the 80386 and 80486 microprocessors. Many of the features of these earlier microprocessors are basically unchanged in the Pentium. The main change is in the paging unit and a new system memory-management mode.

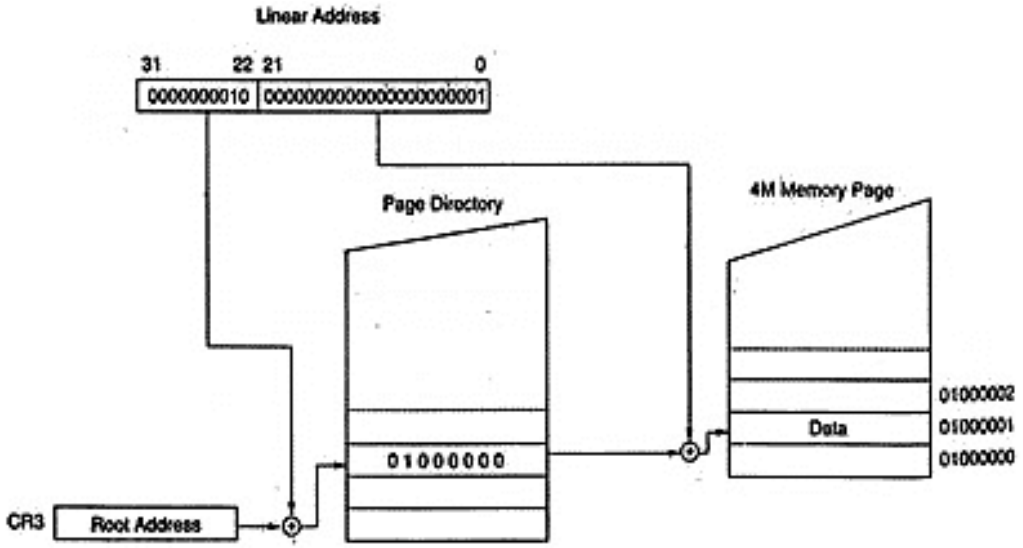
Paging Unit :

The paging mechanism functions with 4K-byte memory pages or with a new extension available to the Pentium with 4M byte-memory pages. The size of the paging table structure can become large in a system that contains a large memory. Recall that to fully repage 4G bytes of memory, the microprocessor requires slightly over 4M bytes of memory just for the page tables. In the Pentium, with the new 4M-byte paging feature, this is dramatically reduced to just a single page table. The new 4M-byte page sizes are selected by the PSE bit in control register 0.

The main difference between 4K paging and 4M paging is that in the 4M paging scheme there is no page table entry in the linear address. Pay close attention to the way the linear address is used with this scheme. Notice that the leftmost 10 bits of the linear address select an entry in the page directory (just as with 4K pages). Unlike 4K pages, there are no page tables; instead, the page directory addresses a 4M-byte memory page.

Memory-Management Mode :

The system memory-management mode (SMM) is on the same level as protected mode, real mode, and virtual mode, but it is provided to function as a manager. The SMM is not intended to be used as an application or a system-level feature. It is intended for high-level system functions such as power management and security, which most Pentiums use during operation.



❑ ❑ ❑ ❑ ❑