CMR INSTITUTE OF TECHNOLOGY

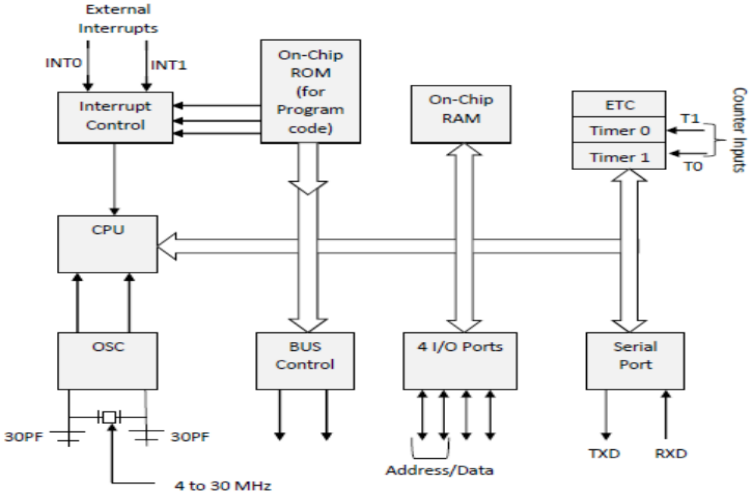
USN | | | | | | | | | | |

Internal Assesment Test - I

Sub:	Microcontroller							Code:	17EE52
Date:	06/09/2019	Duration:	90 mins	Max Marks:	50	Sem:	5	Branch:	EEE
Answer Any FIVE FULL Questions									

		Marks	OBE CO	OBE RBT
1	With a neat diagram, explain the architecture of 8051.	[10]	CO1	L1

Architecture of 8051



…….5

A brief explanation about each block…….5

2	a. Compare Microprocessor and Microcontroller. b. Calculate the time required for 2 machine cycle instruction. (i) 12MHz (ii) 11.0592 MHz	[06] [04]	CO1 CO2	L1 L3

a.

Sl.no	Microprocessor	Microcontroller
1	General purpose processor	Specific application controller
2	Contains no RAM, no ROM, no I/O ports on chip itself.	Contains RAM, ROM, I/O ports on chip itself
3	Size of RAM/ROM can vary	Size of RAM/ROM is fixed
4	Makes the system bulkier	Make the system compact
5	More expensive	Less expensive
6	It has less bit handling instructions	It has more bit handling instructions
7	Less number of pins have multiplexed functions	more number of pins have multiplexed functions
8	More flexible in designer point of view	Less flexible in designer point of view
9	Limited power saving options compared to microcontrollers	Includes lot of power saving features
10	Eg: Desktop PC, 8086, i7	Eg: Digital Camera, 8051, msp430
11	Execution faster	Compared to up slower
12	More general purpose registers	Less number of gen purpose registers
13	More addressing modes	Less addressing modes
14	Design time is more	Application design time less
15	Microprocessors are based on von Neumann model/architecture where program and data are stored in same memory module	Micro controllers are based on Harvard architecture where program memory and Data memory are separate
16	Cannot be used in compact systems and hence inefficient	Can be used in compact systems and hence it is an efficient technique

17	Example code:
	ADD AX,BX
	ADD AX,CX
	ADD AX,DX
		Example code:
		MOV A,#2fh
		MOV B,#2fh
		ADD A,B

18	It cannot be used as stand alone	Can be used as stand alone.
19	May or may not be real-time application oriented	Real-time application oriented
20	Fig : a	Fig :b

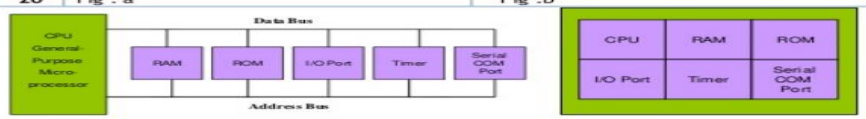


Fig : (a) Microprocessor Fig : (b) Microcontroller

Any 6 of these differences……6

b. (i) For a two machine cycle instruction, where the crystal frequency is 12 MHz……2

12 MHz/12 = 1 MHz
1/1 MHz = 1 µs
2*1 = 2 µs

(ii) For a two machine cycle instruction, where the crystal frequency is 11.0592 MHz…..2

11.0592 MHz/12 = 0.9216 MHz
1/0.9216 MHz = 1.085 µs
2*1.085 µs = 2.170 µs

3	Write an assembly language program to convert ASCII to unpaced BCD and vice versa. Also write ALP to convert ASCII to Decimal and vice versa. Include suitable comments.	[10]	CO2	L4

<u>ASCII to Unpacked BCD</u>

```
ORG 00 H
MOV R0, #10H      ; Initializing pointer R0
MOV A, @R0        ;Move pointer contents(ASCII number) to A
ANL A, #0F H      ;Mask the upper nibble(3)
MOV R2, A         ;Move unpacked BCD to R2……..2.5
```

<u>Unpacked BCD to ASCII</u>

```
ORG 00 H
MOV R0, #10H      ; Initializing pointer R0
MOV A, @R0        ;Move pointer contents(Unpacked BCD) to A
ORL A, #30 H      ;OR 30 H to unpacked BCD to make it ASCII
MOV R3, A         ;Move ASCII to R3…….2.5
```

ASCII to Packed BCD

```
ORG 00 H
MOV R0, #10H      ; Initializing pointer R0
MOV A, @R0        ;Move pointer contents(ASCII number) to A
ANL A, #0F H      ;Mask the upper nibble(3)
MOV R2, A         ;Move unpacked BCD to R2.
INC R0            ;Fetch the other ASCII byte.
ANL A, #0F H      ;Mask the upper nibble(3)
ORL A, R2         ;Add the first result in R2 with A to get packed BCD
MOV R4, A         ;Move packed BCD to R4…….2.5
```

Packed BCD to ASCII

```
ORG 00 H
MOV R0, #10H      ; Initializing pointer R0
MOV A, @R0        ;Move pointer contents(Packed BCD) to A
ANL A, #0F H      ;Mask upper nibble
ORL A, #30 H      ;Add 3 to upper nibble
MOV R5, A         ;Move first ASCII to R5
MOV A, @R0        ;Fetch packed BCD again
ANL A, #0F0 H     ;Mask lower nibble
SWAP A            ;Interchange lower and upper nibble
ORL A, #30 H      ;Add 3 to upper nibble
MOV R6, A         ;Move second ASCII to R6……...2.5
```

4	List and explain different addressing modes of 8051 with suitable examples.	[10]	CO1	L1

The CPU can access data in various ways, which are called *addressing modes*

- Immediate
- Register
- Direct
- Register indirect
- Indexed

Accessing memories

Immediate Addressing Mode

The source operand is a constant

- The immediate data must be preceded by the pound sign, "#"
- Can load information into any registers, including 16-bit DPTR register
  - DPTR can also be accessed as two 8-bit registers, the high byte DPH and low byte DPL

```
MOV A,#25H
MOV R4,#62
```

Register Addressing Mode

Use registers to hold the data to be manipulated

```
MOV A,R0
MOV R2,A
```

Direct Addressing Mode

```
MOV R0,40H
MOV 56H,A
```

Indirect Addressing Mode

A register is used as a pointer to the data

- Only register R0 and R1 are used for this purpose
- R2 – R7 cannot be used to hold the address of an operand located in RAM

When R0 and R1 hold the addresses of RAM locations, they must be preceded by the "@" sign

Indexed Addressing Mode

Indexed addressing mode is widely used in accessing data elements of look-up table entries located in the program ROM

The instruction used for this purpose is

```
MOVC A,@A+DPTR
```

- Use instruction MOVC, "C" means code
- The contents of A are added to the 16-bit register DPTR to form the 16-bit address of the needed data

Each addressing mode with an example carries 2 marks.

5	Explain the following instructions, explaining their expressing mode and byte size. (i) XCH A, @R0 (ii) MOVC A, @A+DPTR (iii) SUBB A, #55H (iv) DA A (v) ORL C, 100	[10]	CO2	L4

(i) XCH A, @R0……2

The contents of the accumulator and the location content pointed by R0 are exchanged.
Addressing Mode: Indirect
Byte Size: 1

(ii) MOVC A, @A+DPTR……..2

The content of the location pointed by the sum of accumulator and DPTR will be loaded into Accumulator.
Addressing Mode: Indexed
Byte Size: 1

(iii) SUBB A, #55H ……2

The immediate data 55 H will be subtraced from the contents of the accumulator along with borrow And the result will be saved in Accumulator.
Addressing Mode: Immediate
Byte Size: 1

(iv) DA A…….2

Decimal Adjust is used only after addition and it changes the Hexadecimal result to BCD.
Addressing Mode: Register
Byte Size: 1

(v) ORL C, 100……..2

Logical OR the contents of the location 100d with C and store the result in C.
Addressing Mode: Direct
Byte Size: 1/8

6	Explain the operation of following code with respect to stack
	MOV SP, #10H
	PUSH SP
	POP 0E0H
	ADD A, #10H
		[10]	CO1	L4

```
MOV SP, #10H      ;Immediate data 10 H will be moved to Stack Pointer……2.5
PUSH SP           ;SP data 10 H will be copied to 11 H location……..2.5
POP 0E0H          ;Data 10 H will be loaded into locaiton 0E0 H that is Accumulator……..2.5
ADD A, #10H       ;Immediate data 10H will be added to 10H in accumulator and the sum is saved in A again,2.5
```

NOTE: Stack PUSH and POP diagrams to be shown.

7	a. Explain ORG, END, DB and EQU directives. b. Write a program to add 5 numbers. Numbers are stored between internal RAM 60H and 64H. Store the result in R0 and A.	[04] [06]	CO2 CO2	L1 L4

a. ORG is to Originate the program from
END is to End the program at
DB is to define byte
EQU is to equate a constant value to a variable.
Each with an example carries 4 marks……..4

```
b.          MOV R2, #4
            MOV R1, #60 H
            MOV A, @R1
REPEAT: INC R1
            ADD A, @R1
            JC GOTO
            SJMP SKIP
GOTO: INC R0
SKIP: DJNZ R2, REPEAT
            END
```

With suitable comments……6

8	a. Explain the calculation of checksum byte in ROM with an example. b. Write a program to load accumulator with the value 55H and complement the content of the accumulator 900 times.	[05] [05]	CO2 CO2	L4 L4

a.

To ensure the integrity of the ROM contents, every system must perform the checksum calculation

- The process of checksum will detect any corruption of the contents of ROM
- The checksum process uses what is called a *checksum byte*
  - The checksum byte is an extra byte that is tagged to the end of series of bytes of data

- To calculate the checksum byte of a series of bytes of data
  - Add the bytes together and drop the carries
  - Take the 2's complement of the total sum, and it becomes the last byte of the series
- To perform the checksum operation, add all the bytes, including the checksum byte
  - The result must be zero
  - If it is not zero, one or more bytes of data have been changed

With example carries 5 marks……5

```
b.          MOV A, # 55H
            MOV R0, #10
AGAIN: MOV R1, #90
REPEAT:  CPL A
            DJNZ R1, REPEAT
            DJNZ R0, AGAIN
```

END
With suitable comments……5

9	Write an assembly language program to find cube of a number.	[10]	CO2	L4

```
ORG 00 H
MOV R0, # 30 H
MOV A, @R0
MOV B, A
MUL AB
MOV R4, B
MOV B, @R0
MUL AB
MOV 50 H, A
MOV R5, B
MOV A, R4
MOV B, @R0
MUL AB
ADD A, R5
MOV 51 H, A
MOV A, B
ADDC A, #00H
MOV 52 H, A
END
```

With suitable comments……10

10	Write an assembly language program to subtract two 16 bit numbers stored in external memory and store the results in internal memory.	[10]	CO2	L4

```
ORG 00 H
MOV R0, #30 H          ;initializing internal memory
MOV DPTR, #5000 H      ;initializing external memory
MOV A, @DPTR           ;move first data from external memory to A
INC DPL                ;increment DPL twice to point to second number's LSB
INC DPL
CLR C                  ;clear borrow
SUBB A, @DPTR          ;subtract second LSB from first and save the result in A
MOV @R0, A             ;move the difference to 30 H internal memory
DEC DPL                ;point to first MSB
MOV A, @DPTR           ;move first MSB to A from DPTR
INC DPL                ;point to second MSB
INC DPL
SUBB A, @DPTR          ;subtract the two MSBs along with borrow and save data in internal RAM
INC R0
MOV @R0, A
END
```

……..10