CMR
INSTITUTE OF
TECHNOLOGY

USN

Internal Assesment Test – I and solutions and scheme

Sub:	Microcontroller 8051 (Open Elective)							Code:	15EC563
Date:	21 / 09 / 2014	Duration:	90 mins	Max Marks:	50	Sem:	V	Branch:	CSE/ISE

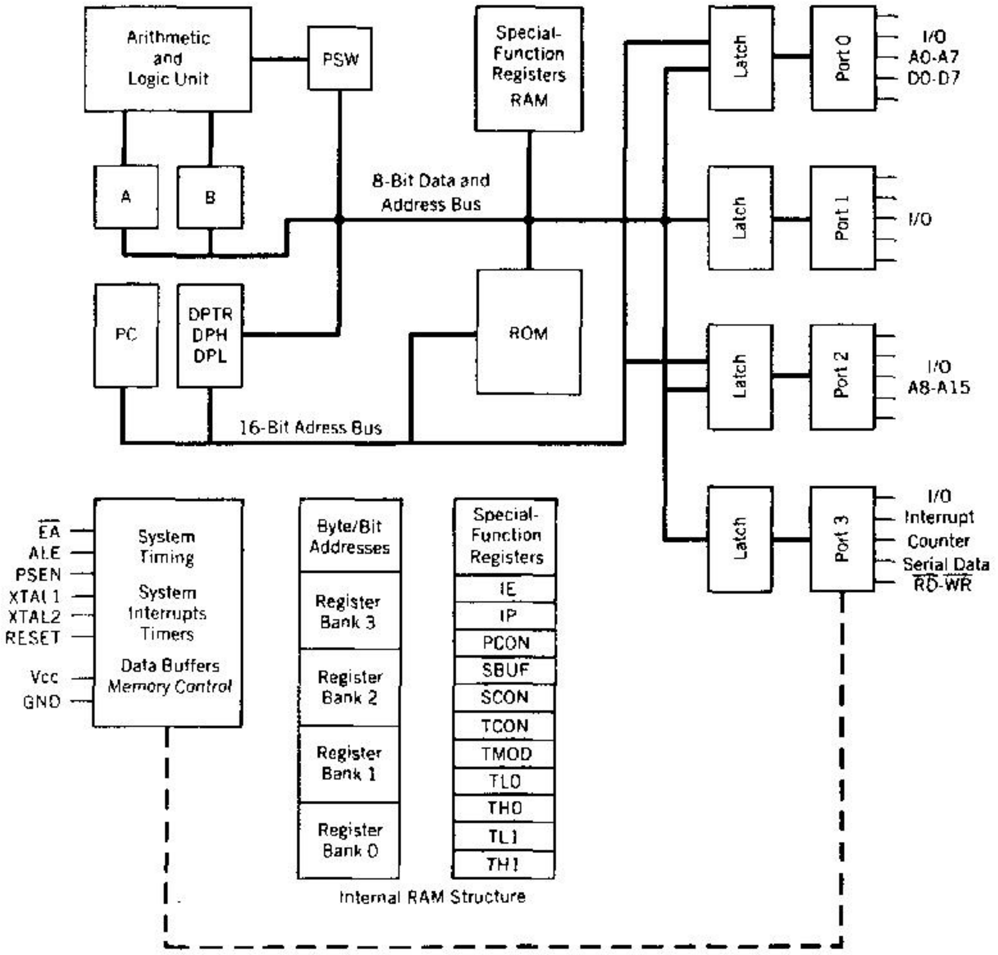
Marks

1 Explain the architecture of 8051 Microcontroller with neat block diagram. [10]

Block diagram =5 marks

Explanations of each block =5 marks

Solution:-



- 8051 has 4 K Bytes of internal ROM. The address space is from 0000 to 0FFFh. If the program size is more than 4 K Bytes 8051 will fetch the code automatically from external memory.
- Accumulator is an 8 bit register widely used for all arithmetic and logical operations. Accumulator is also used to transfer data between external memory. B register is used along with Accumulator for multiplication and division. A and B registers together is also called MATH registers.

- PSW (Program Status Word). This is an 8 bit register which contains the arithmetic status of ALU and the bank select bits of register banks.

CY	AC	F0	RS1	RS0	OV	-	P

CY - carry flag
AC - auxiliary carry flag
F0 - available to the user for general purpose

RS1,RS0 - register bank select bits
OV - overflow
P - parity

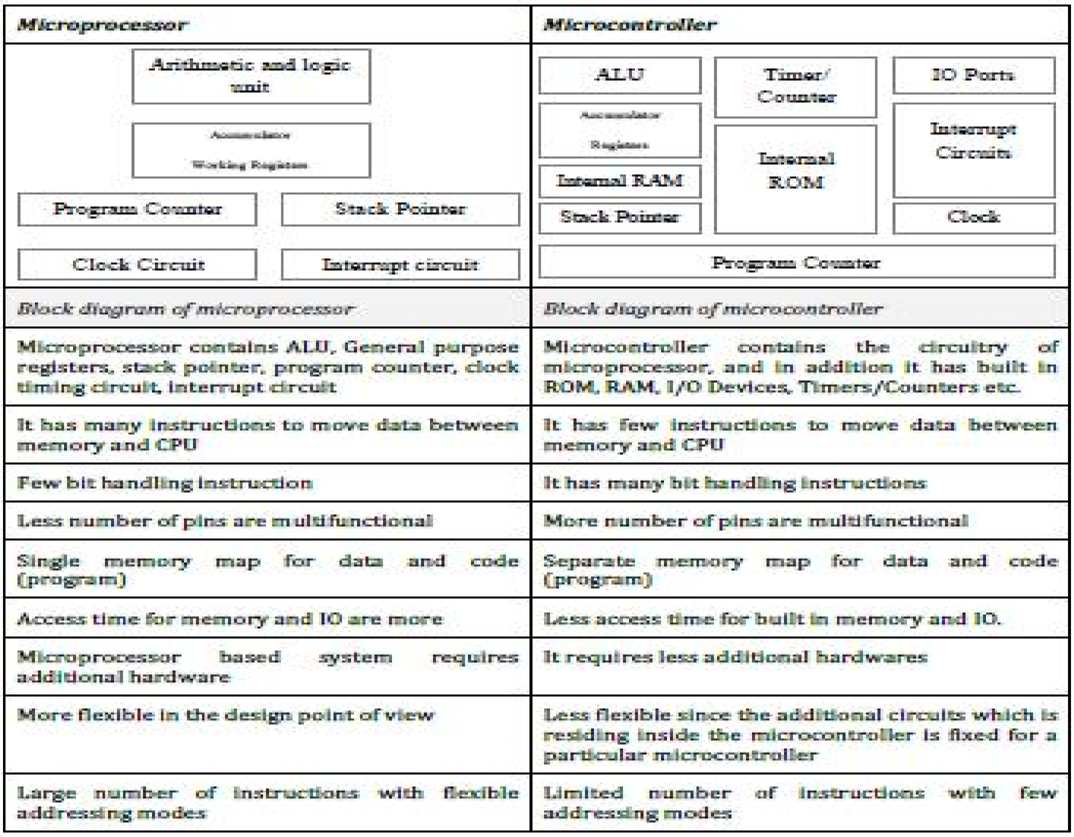
- Stack Pointer (SP) – it contains the address of the data item on the top of the stack. Stack may reside anywhere on the internal RAM. On reset, SP is initialized to 07 so that the default stack will start from address 08 onwards.
- Data Pointer (DPTR) – DPH (Data pointer higher byte), DPL (Data pointer lower byte). This is a 16 bit register which is used to furnish address information for internal and external program memory and for external data memory.
- Program Counter (PC) – 16 bit PC contains the address of next instruction to be executed. On reset PC will set to 0000. After fetching every instruction PC will increment by one.

2 Distinguish between Microprocessor and Microcontroller.

[10]

1.5 Marks for each difference.

Solution:-


Arithmetic and logic unit
Accumulator
Working Registers
Program Counter
Stack Pointer
Clock Circuit
Interrupt circuit
ALU
Timer/ Counter
IO Ports
Accumulator
Registers
Internal RAM
Internal ROM
Interrupt Circuits
Stack Pointer
Clock
Program Counter

Microprocessor	*Microcontroller*
Block diagram of microprocessor	*Block diagram of microcontroller*
Microprocessor contains ALU, General purpose registers, stack pointer, program counter, clock timing circuit, interrupt circuit	Microcontroller contains the circuitry of microprocessor, and in addition it has built in ROM, RAM, I/O Devices, Timers/Counters etc.
It has many instructions to move data between memory and CPU	It has few instructions to move data between memory and CPU
Few bit handling instruction	It has many bit handling instructions
Less number of pins are multifunctional	More number of pins are multifunctional
Single memory map for data and code (program)	Separate memory map for data and code (program)
Access time for memory and IO are more	Less access time for built in memory and IO.
Microprocessor based system requires additional hardware	It requires less additional hardwares
More flexible in the design point of view	Less flexible since the additional circuits which is residing inside the microcontroller is fixed for a particular microcontroller
Large number of instructions with flexible addressing modes	Limited number of instructions with few addressing modes

3 Explain Memory organization in 8051.

[10]

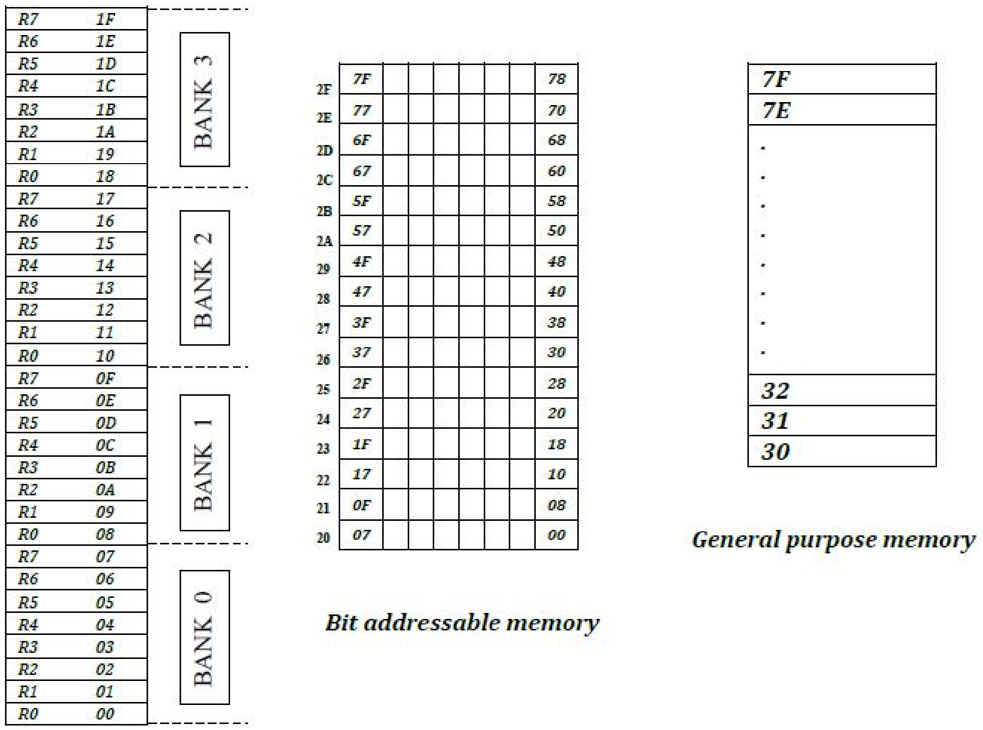
RAM memory organization block diagram 6 marks

Rom 2 Marks

Explanation of SFR 2 Marks

Solution:-

Internal RAM organization



Register Banks: 00h to 1Fh. The 8051 uses 8 general-purpose registers R0 through R7 (R0, R1, R2, R3, R4, R5, R6, and R7). There are four such register banks. Selection of register bank can be done through RS1,RS0 bits of PSW. On reset, the default Register Bank 0 will be selected.

Bit Addressable RAM: 20h to 2Fh . The 8051 supports a special feature which allows access to bit variables. This is where individual memory bits in Internal RAM can be set or cleared. In all there are 128 bits numbered 00h to 7Fh. Being bit variables any one variable can have a value 0 or 1. A bit variable can be set with a command such as SETB and cleared with a command such as CLR.

Example instructions are:

SETB 25h ; sets the bit 25h (becomes 1)

CLR 25h ; clears bit 25h (becomes 0)

Note, bit 25h is actually bit 5 of Internal RAM location 24h.

The Bit Addressable area of the RAM is just 16 bytes of Internal RAM located between 20h and 2Fh.

General Purpose RAM: 30h to 7Fh. Even if 80 bytes of Internal RAM memory are available for general-purpose data storage, user should take care while using the memory location from 00 -2Fh since these locations are also the default register space, stack space, and bit addressable space. It is a good practice to use general purpose memory from 30 – 7Fh. The general purpose RAM can be accessed using direct or indirect addressing modes.

4 Explain Arithmetic and logical instructions of 8051 with examples.

[10]

Each instructions 1.5 marks (includes explanation with example)

Solution:-

Arithmetic instructions.

The 8051 can perform addition, subtraction. Multiplication and division operations on 8 bit numbers.

Addition

In this group, we have instructions to

i. Add the contents of A with immediate data with or without carry.

i. ADD A, #45H

ii. ADDC A, #OB4H

ii. Add the contents of A with register Rn with or without carry.

i. ADD A, R5

ii. ADDC A, R2

iii. Add the contents of A with contents of memory with or without carry using direct and indirect addressing

i. ADD A, 51H

ii. ADDC A, 75H

iii. ADD A, @R1
iv. ADDC A, @R0

CY AC and OV flags will be affected by this operation.
Subtraction
In this group, we have instructions to
i. Subtract the contents of A with immediate data with or without carry.
i. SUBB A, #45H
ii. SUBB A, #OB4H
ii. Subtract the contents of A with register Rn with or without carry.
i. SUBB A, R5
ii. SUBB A, R2
iii. Subtract the contents of A with contents of memory with or without carry using direct and indirect addressing
i. SUBB A, 51H
ii. SUBB A, 75H
iii. SUBB A, @R1
iv. SUBB A, @R0

CY AC and OV flags will be affected by this operation.

Multiplication
MUL AB. This instruction multiplies two 8 bit unsigned numbers which are stored in A and B register. After multiplication the lower byte of the result will be stored in accumulator and higher byte of result will be stored in B register.
Eg. MOV A,#45H *;[A]=45H*
MOV B,#0F5H *;[B]=F5H*
MUL AB *;[A] x [B] = 45 x F5 = 4209*
;[A]=09H, [B]=42H

Division
DIV AB. This instruction divides the 8 bit unsigned number which is stored in A by the 8 bit unsigned number which is stored in B register. After division the result will be stored in accumulator and remainder will be stored in B register.
Eg. MOV A,#45H *;[A]=0E8H*
MOV B,#0F5H *;[B]=1BH*
DIV AB *;[A] / [B] = E8 /1B = 08 H with remainder 10H*
;[A] = 08H, [B]=10H

DA A (Decimal Adjust After Addition).
When two BCD numbers are added, the answer is a non-BCD number. To get the result in BCD, we use DA A instruction after the addition. DA A works as follows.
> If lower nibble is greater than 9 or auxiliary carry is 1, 6 is added to lower nibble.
> If upper nibble is greater than 9 or carry is 1, 6 is added to upper nibble.

Eg 1: MOV A,#23H
MOV R1,#55H
ADD A,R1 // [A]=78
DA A // [A]=78 ***no changes in the accumulator after da a***
Eg 2: MOV A,#53H
MOV R1,#58H
ADD A,R1 // [A]=ABh
DA A // [A]=11, C=1 . ANSWER IS 111. ***Accumulator data is changed after DA A***

Increment: *increments the operand by one.*
INC A INC Rn INC DIRECT INC @Ri INC DPTR
INC increments the value of source by 1. If the initial value of register is FFh, incrementing the value will cause it to reset to 0. The Carry Flag is not set when the value "rolls over" from 255 to 0.
In the case of "INC DPTR", the value two-byte unsigned integer value of DPTR is incremented. If the initial value of DPTR is FFFFh, incrementing the value will cause it to reset to 0.

Decrement: *decrements the operand by one.*
DEC A DEC Rn DEC DIRECT DEC @Ri
DEC decrements the value of *source* by 1. If the initial value of is 0, decrementing the value will cause it

to reset to FFh. The Carry Flag is not set when the value "rolls over" from 0 to FFh.

DIV AB. This instruction divides the 8 bit unsigned number which is stored in A by the 8 bit unsigned number which is stored in B register. After division the result will be stored in accumulator and remainder will be stored in B register.

Eg. MOV A,#45H *;[A]=0E8H*
MOV B,#0F5H *;[B]=1BH*
DIV AB *;[A] / [B] = E8 /1B = 08 H with remainder 10H*
;[A] = 08H, [B]=10H

DA A (Decimal Adjust After Addition).
When two BCD numbers are added, the answer is a non-BCD number. To get the result in BCD, we use DA A instruction after the addition. DA A works as follows.
> If lower nibble is greater than 9 or auxiliary carry is 1, 6 is added to lower nibble.
>If upper nibble is greater than 9 or carry is 1, 6 is added to upper nibble.

Eg 1: MOV A,#23H
MOV R1,#55H
ADD A,R1 // [A]=78
DA A // [A]=78 ***no changes in the accumulator after da a***
Eg 2: MOV A,#53H
MOV R1,#58H
ADD A,R1 // [A]=ABh
DA A // [A]=11, C=1 . ANSWER IS 111. ***Accumulator data is changed after DA A***

Increment: *increments the operand by one.*
INC A INC Rn INC DIRECT INC @Ri INC DPTR
INC increments the value of source by 1. If the initial value of register is FFh, incrementing the value will cause it to reset to 0. The Carry Flag is not set when the value "rolls over" from 255 to 0.
In the case of "INC DPTR", the value two-byte unsigned integer value of DPTR is incremented. If the initial value of DPTR is FFFFh, incrementing the value will cause it to reset to 0.

Decrement: *decrements the operand by one.*
DEC A DEC Rn DEC DIRECT DEC @Ri
DEC decrements the value of *source* by 1. If the initial value of is 0, decrementing the value will cause it to reset to FFh. The Carry Flag is not set when the value "rolls over" from 0 to FFh.
leaving the resulting value in *destination*. The value in source is not affected.

Logical OR
ORL destination, source: ORL does a bitwise "OR" operation between *source* and *destination*,
ORL A,#DATA ORL A, Rn
ORL A,DIRECT ORL A,@Ri
ORL DIRECT,A ORL DIRECT, #DATA

Logical Ex-OR
XRL destination, source: XRL does a bitwise "EX-OR" operation between *source* and *destination*, leaving the resulting value in *destination*. The value in source is not affected. " XRL " instruction logically EX-OR the bits of source and destination.
XRL A,#DATA XRL A,Rn
XRL A,DIRECT XRL A,@Ri
XRL DIRECT,A XRL DIRECT, #DATA

Logical NOT
CPL complements *operand*, leaving the result in *operand*. If *operand* is a single bit then the state of the bit will be reversed. If *operand* is the Accumulator then all the bits in the Accumulator will be reversed.
CPL A, CPL C, CPL bit address

SWAP A – Swap the upper nibble and lower nibble of A.

5 Write an ASM program to move data #50h from Memory location 35h to 40h. Use minimum length (size) coding with moderate timing. Note down both size of the code as well as timing (in machine cycle).

[10]

Minimum 2 codes for comparison. Each code is 3 Marks

Solution:-
MOV 35H, #50H
MOV 36H,#50H
MOV 37H,#50H
MOV 38H,#50H
MOV 39H,#50H
MOV 3AH,#50H

	MOV 3BH,#50H **MOV 3CH,#50H** **MOV 3DH,#50H** **MOV 3EH,#50H** **MOV 3FH,#50H** **MOV 40H,#50H** **TOTAL:- 36 BYTES , 12 LINES** To calculate timing, Each instruction takes 1 machine cycle hence totally **12 machine cycles** is used. **Another solution:-** **Start:** **MOV A,#35H (1 MC)** **BACK: MOV @A,#50H (1 MC)** **INC A (1 MC)** **CJNE A,#40H, BACK (2 MC)** **END:** **TOTAL 10BYTES, 5 LINES** **Here, 6 Machine cycle.**	Calculation of Timing, size and total lines for each code 2 marks
6	Write an ASM program to add two numbers stored in Register bank 3 (R3, R4) subtract the result from Register bank 2 (R3, R4) store the data in External memory 8000h and 8001h(higher byte and lower byte respectively). **Solution:-** We need to add bank 3 Reg R3 + R4 =X, and subtract R3-X =[8000h] and R4-X=[8001h] Step 1 :- Change PSW status to read Reg bank 3, collect the data in R3, R4 and save it in A reg Step 2:- Change PSW status to read Reg bank 2, collect the date in R2,R4 subtract with value in A Step 3:- store the result in External memory using movx.	[10] Each step with code is 3 marks Comments 1 mark.