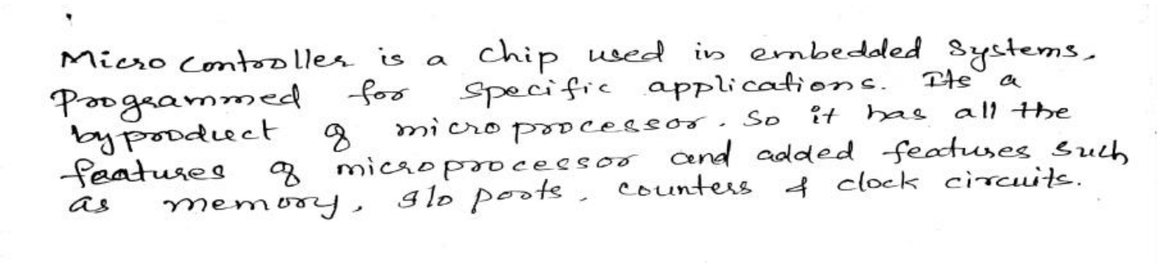
SOLUTIONS OF I IAT QUESTION PAPER

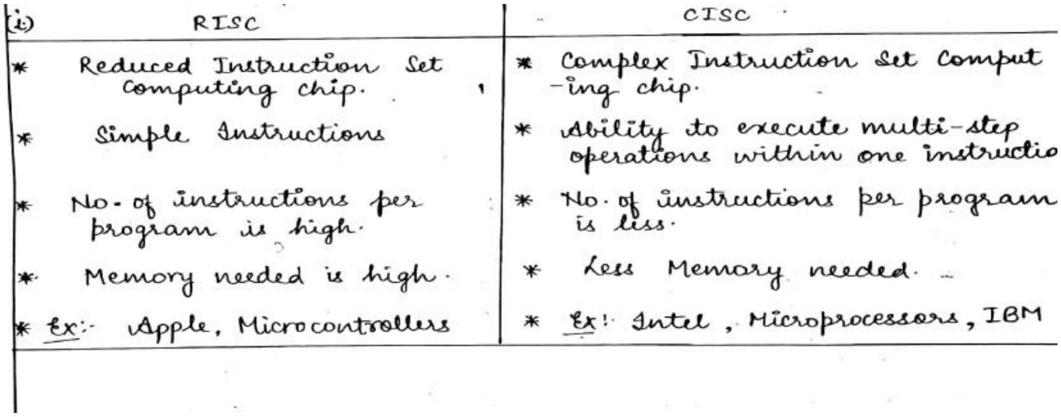
1.a)



Micro controller is a chip used in embedded systems, Programmed for specific applications. Its a byproduct of microprocessor. So it has all the features of microprocessor and added features such as memory, I/O ports, counters & clock circuits.

1M

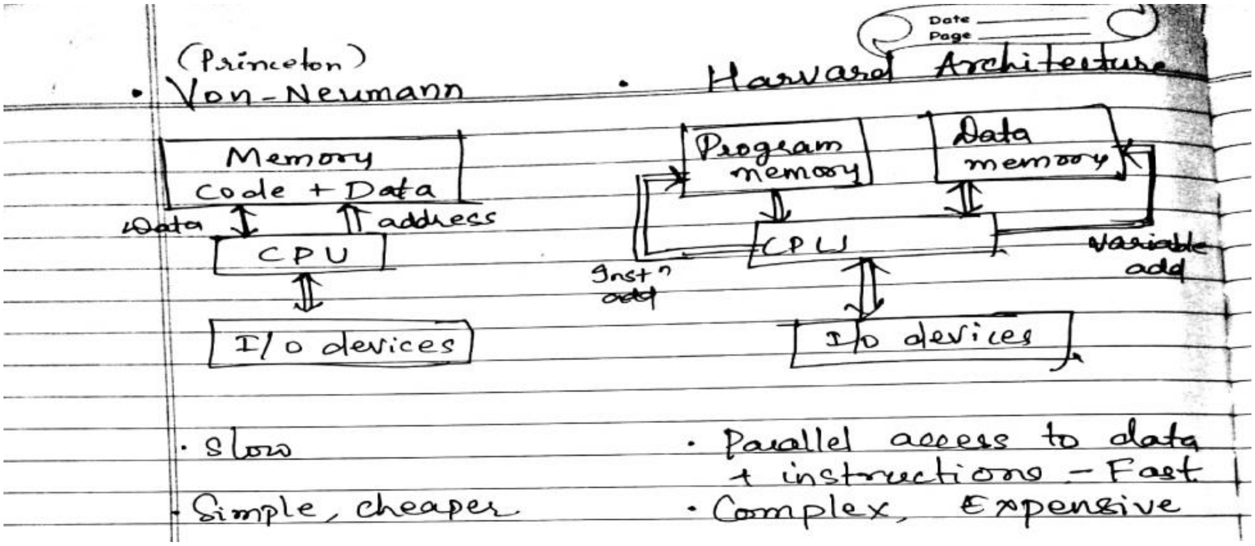
i)



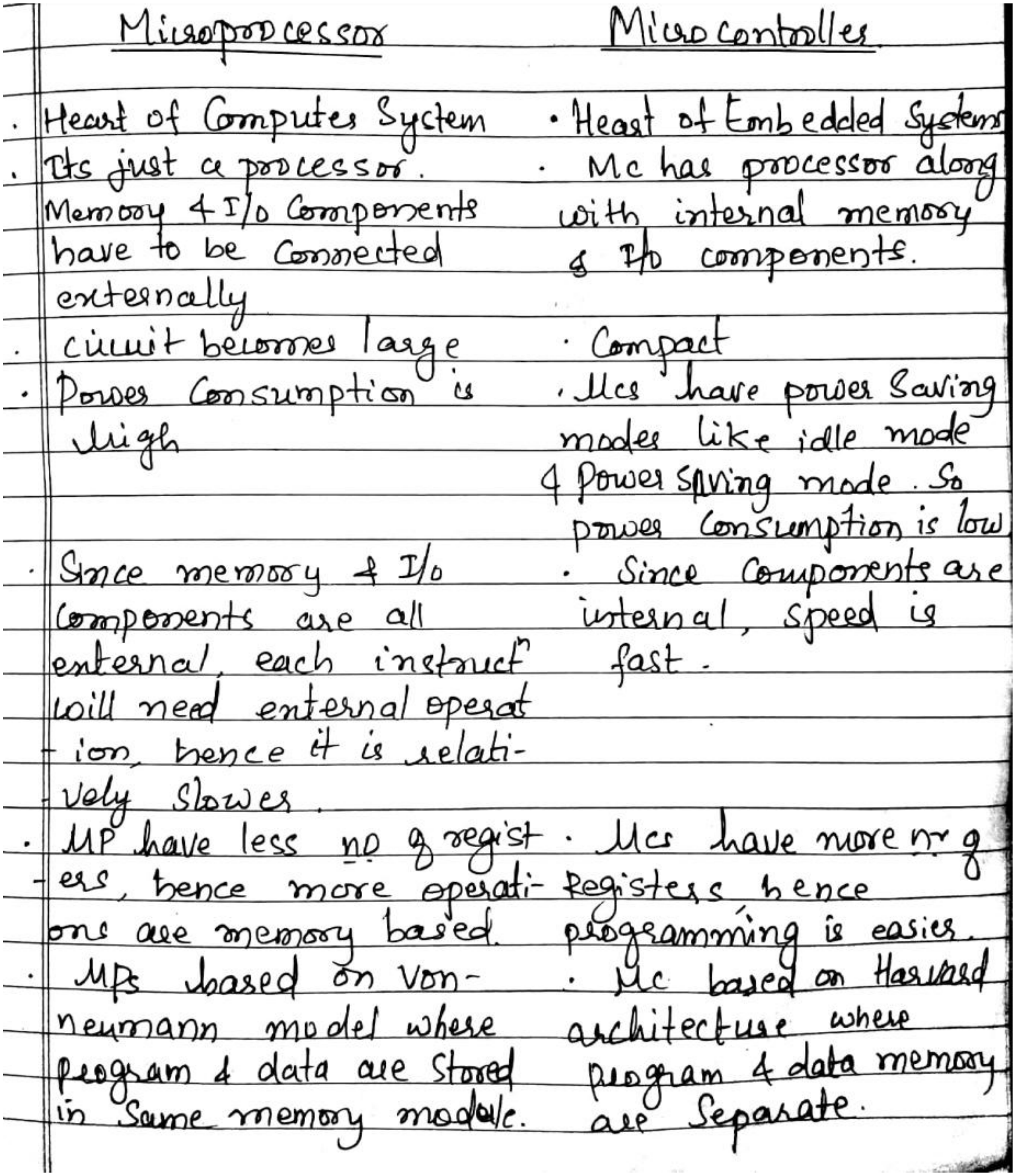
(i) RISC	CISC
* Reduced Instruction Set Computing chip.	* Complex Instruction Set Computing chip.
* Simple Instructions	* Ability to execute multi-step operations within one instructio
* No. of instructions per program is high.	* No. of instructions per program is less.
* Memory needed is high.	* Less Memory needed.
* Ex:- Apple, Microcontrollers	* Ex:- Intel, Microprocessors, IBM

3M

ii)



Von-Neumann	Harvard Architecture
• Slow	• Parallel access to data + instructions - Fast
• Simple, cheaper	• Complex, Expensive

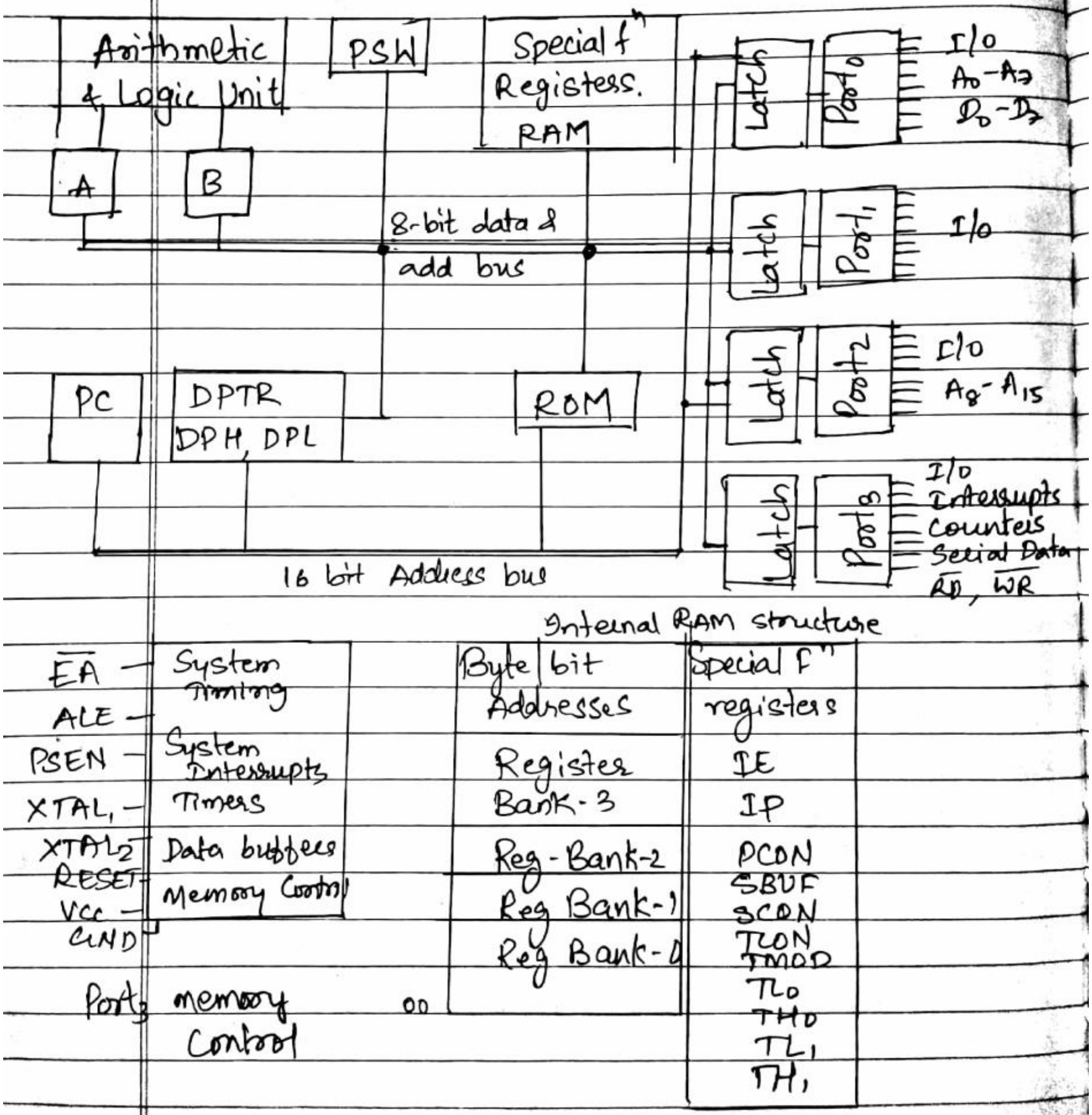


Microprocessor	Micro controller
· Heart of Computer System	· Heart of Embedded System
· Its just a processor. Memory & I/o Components have to be Connected externally	· Mc has processor along with internal memory & I/o components.
· circuit becomes large	· Compact
· Power Consumption is high	· Mcs have power Saving modes like idle mode & Power saving mode. So power Consumption is low
· Since memory & I/o components are all external, each instruct^n will need external operation, hence it is relatively slower.	· Since components are internal, speed is fast.
· MP have less no of registers, hence more operations are memory based.	· Mcs have more nr of Registers hence programming is easier.
· MPs based on Von-neumann model where program & data are Stored in Same memory module.	· Mc based on Harvard architecture where program & data memory are Separate.

3M

2 a)

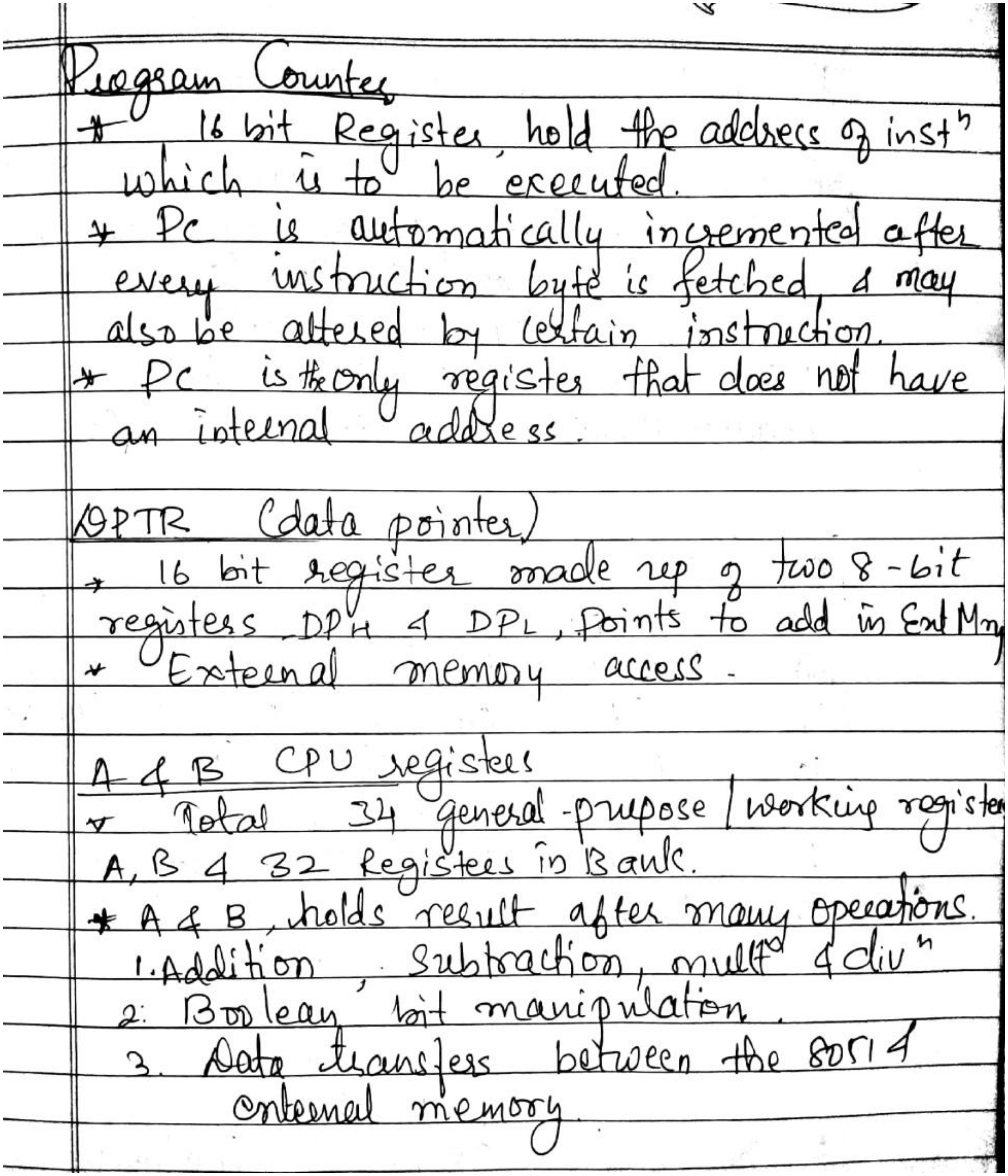
8051 Architecture



Page ______

Specific features of 8051 Architecture.

- Eight bit CPU with registers A (Accumulator and B.
- Siateen-bit program Counter (PC) & data pointer (DPTR)
- Eight bit Stack pointer (SP)
- Internal ROM or EPROM (4K)
- Internal RAM of 128 bytes.
  - Four register banks, each containing 8 ecgisters.
  - 16 bytes, which can be addressed at the bit level.
  - 80 bytes of general purpose data memory
- 32 i/o pins arranged as 4-8 bit poots $P_0 - P_3$.
- Two 16-bit Timer/counters: T_0 and T_1
- Full duplex Serial data receiver/transmitter : SBUF.
- Control registers : TCON, TMOD, SCON, PCON, IP & IE.
- Two exteenal and three internal interrupt Sources.
- Oscillator & clock circuits.



Program Counter

* 16 bit Register, hold the address of inst^n which is to be executed.
* Pc is automatically incremented after every instruction byte is fetched & may also be altered by certain instruction.
* PC is the only register that does not have an internal address.

DPTR (data pointer)

* 16 bit register made up of two 8-bit registers DPH & DPL, Points to add in Ext Mry
* External memory access.

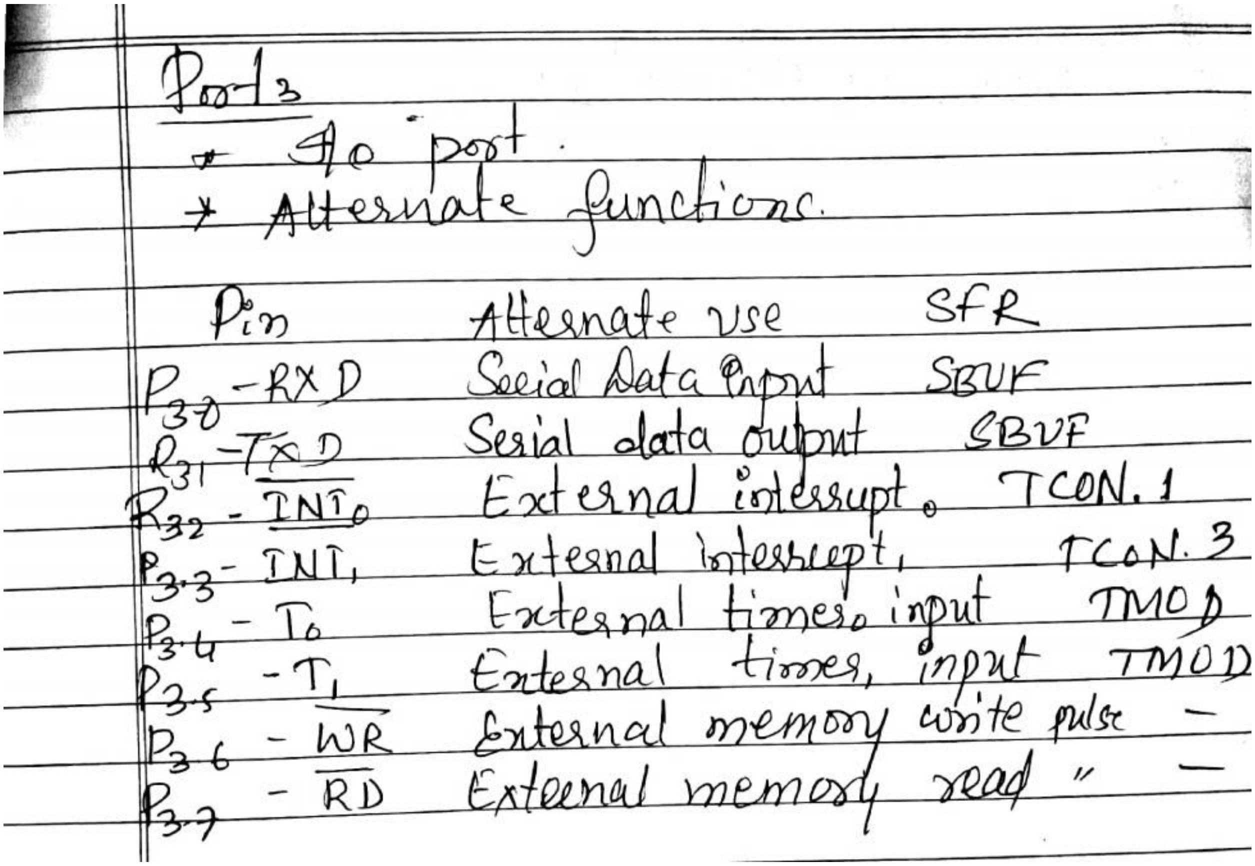
A & B CPU registers

* Total 34 general-purpose / working registers A, B & 32 Registers in Bank.
* A & B, holds result after many operations.
  1. Addition, Subtraction, mult^n & div^n
  2. Boolean bit manipulation.
  3. Data transfers between the 8051 & internal memory.

Architecture diagram – 4M

Explaination – 6M

3 a.



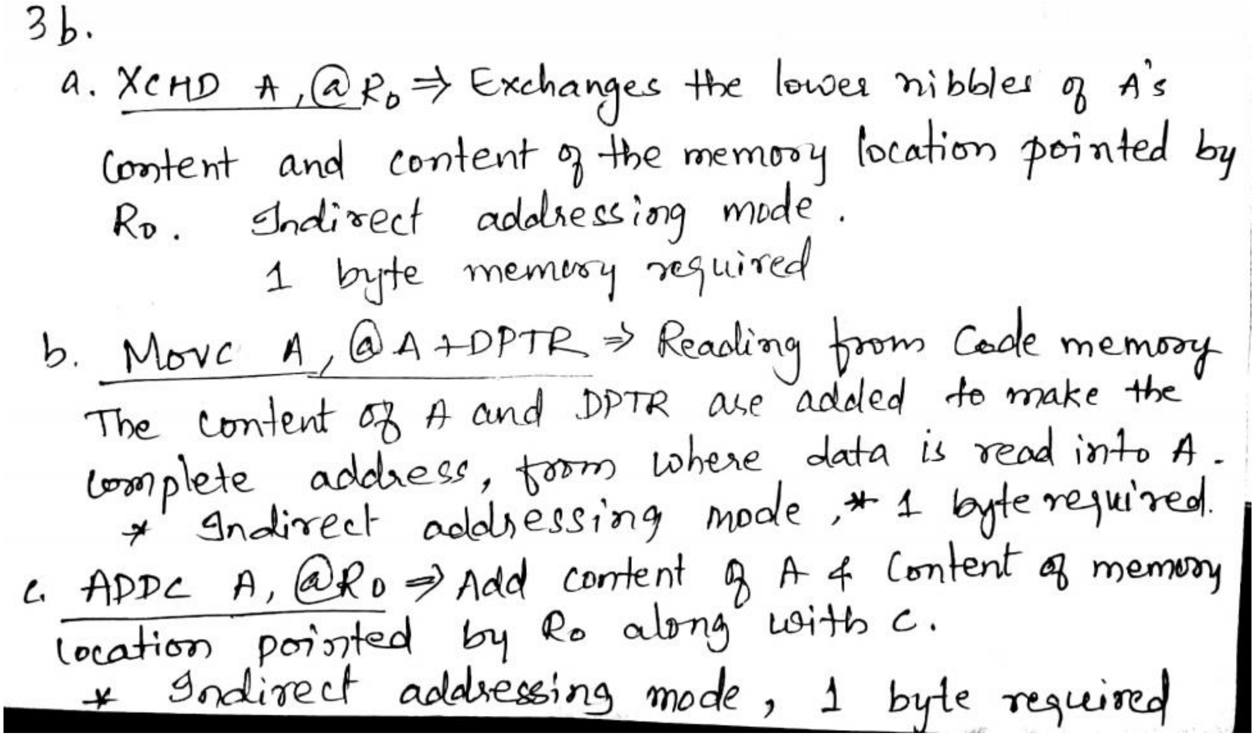
Port 3

- I/O port.
- Alternate functions.

Pin	Alternate use	SFR
$P_{3.0}$ - RXD	Seeial Data Input	SBUF
$P_{3.1}$ - TXD	Serial data output	SBUF
$P_{3.2}$ - $\overline{INT_0}$	External interrupt$_0$	TCON.1
$P_{3.3}$ - $\overline{INT_1}$	External interrupt$_1$	TCON.3
$P_{3.4}$ - T_0	External timer$_0$ input	TMOD
$P_{3.5}$ - T_1	External timer$_1$ input	TMOD
$P_{3.6}$ - $\overline{WR}$	External memory write pulse	—
$P_{3.7}$ - $\overline{RD}$	External memory read "	—

For listing the names of the pins - 2 M

Explain briefly about each pin.   2 M



3 b.

a. XCHD A, @R_0 ⇒ Exchanges the lower nibbles of A's content and content of the memory location pointed by R_0. Indirect addressing mode.

1 byte memory required

b. Movc A, @A+DPTR ⇒ Reading from Code memory The content of A and DPTR are added to make the complete address, from where data is read into A.

* Indirect addressing mode, * 1 byte required.

c. ADDC A, @R_0 ⇒ Add content of A & Content of memory location pointed by R_0 along with C.

* Indirect addressing mode, 1 byte required

Operation explaination - 1M

Addressing mode - 0.5 M

Number of bytes - 0.5M total 2M for each instruction.

2*3=6

4. Explain how to interface 8K ROM and 4K RAM memory to 8051.

Soln: Diagram of connection -4M

Timing diagram-2M

Explaination – 4M

Note: The following diagram shows connecting 16k ROM anf 8k RAM , but in exam its asked for 8k ROM and 4K RAM. For connecting 8K ROM memory 13 (A0-A12) address bits are required and for 4k RAM memory 11 (A0-A11) address bits are required.

Connecting External memory.

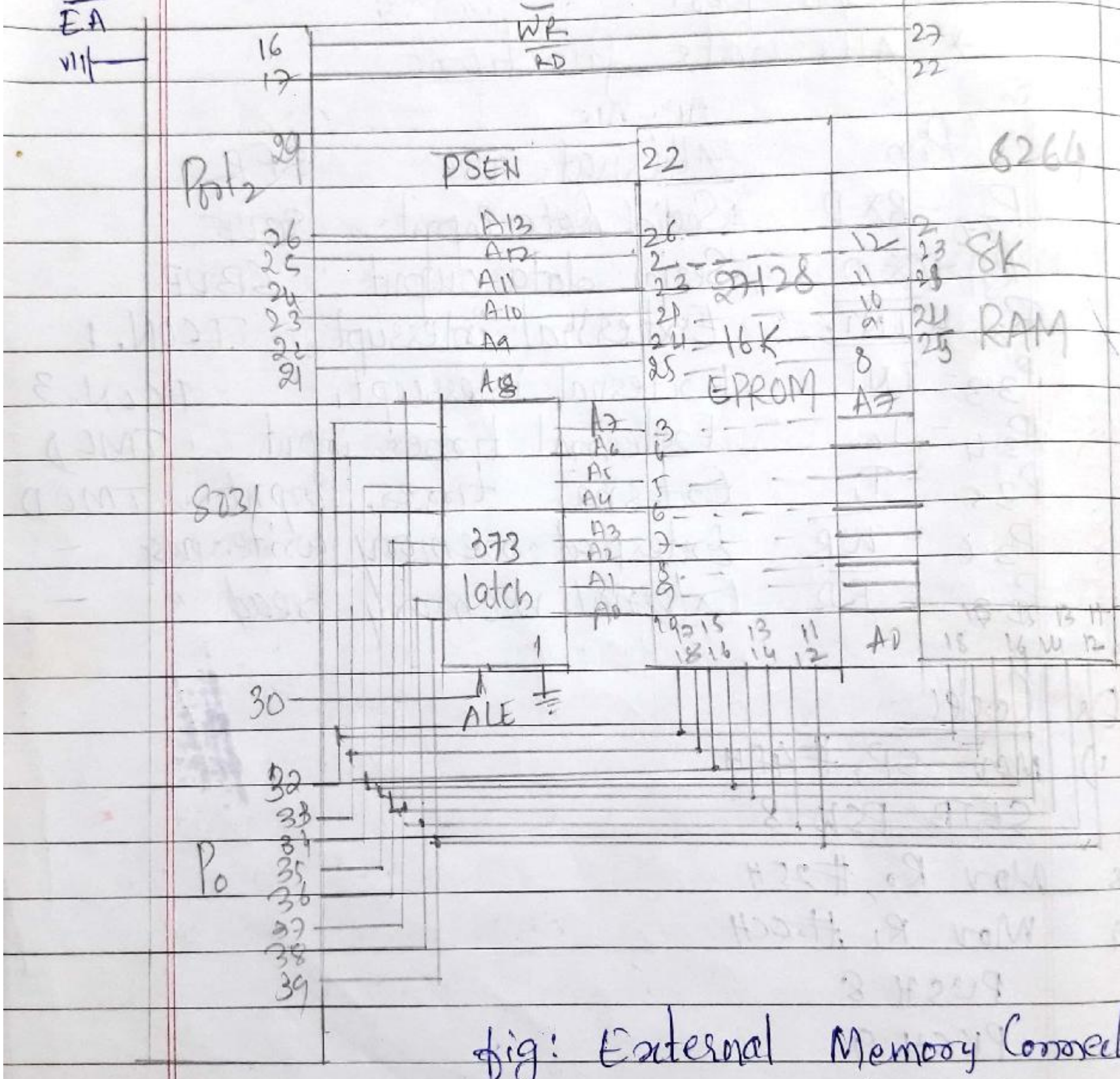
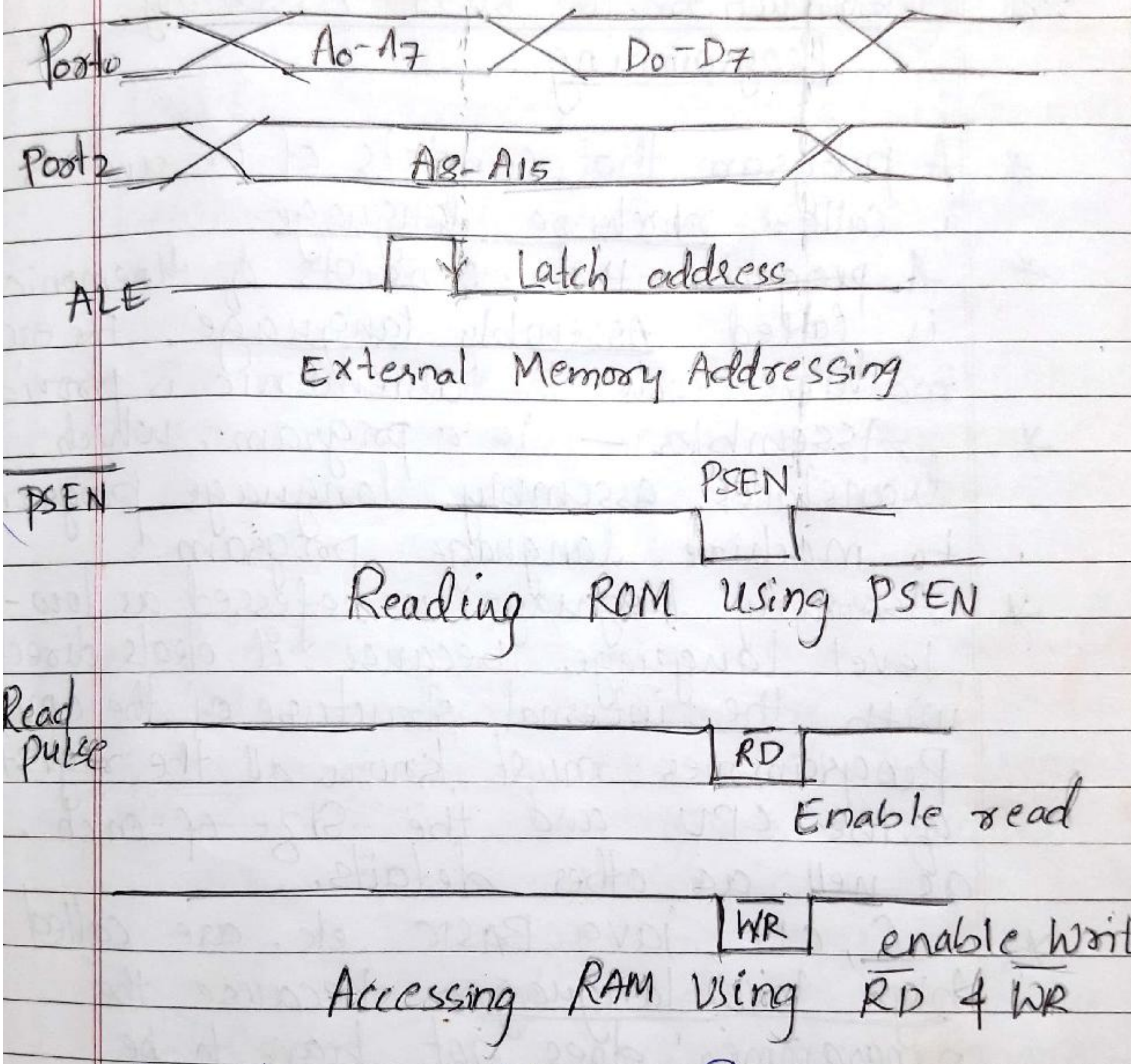


fig: External Memory Connection

$*$ ROM may be expanded to 64k by using 2751 type EPROM & connecting port 2 upper addre lines A_{14} - A_{15} to the chip.

$*$ RAM may be expanded to 64 k-using 62864 chip.



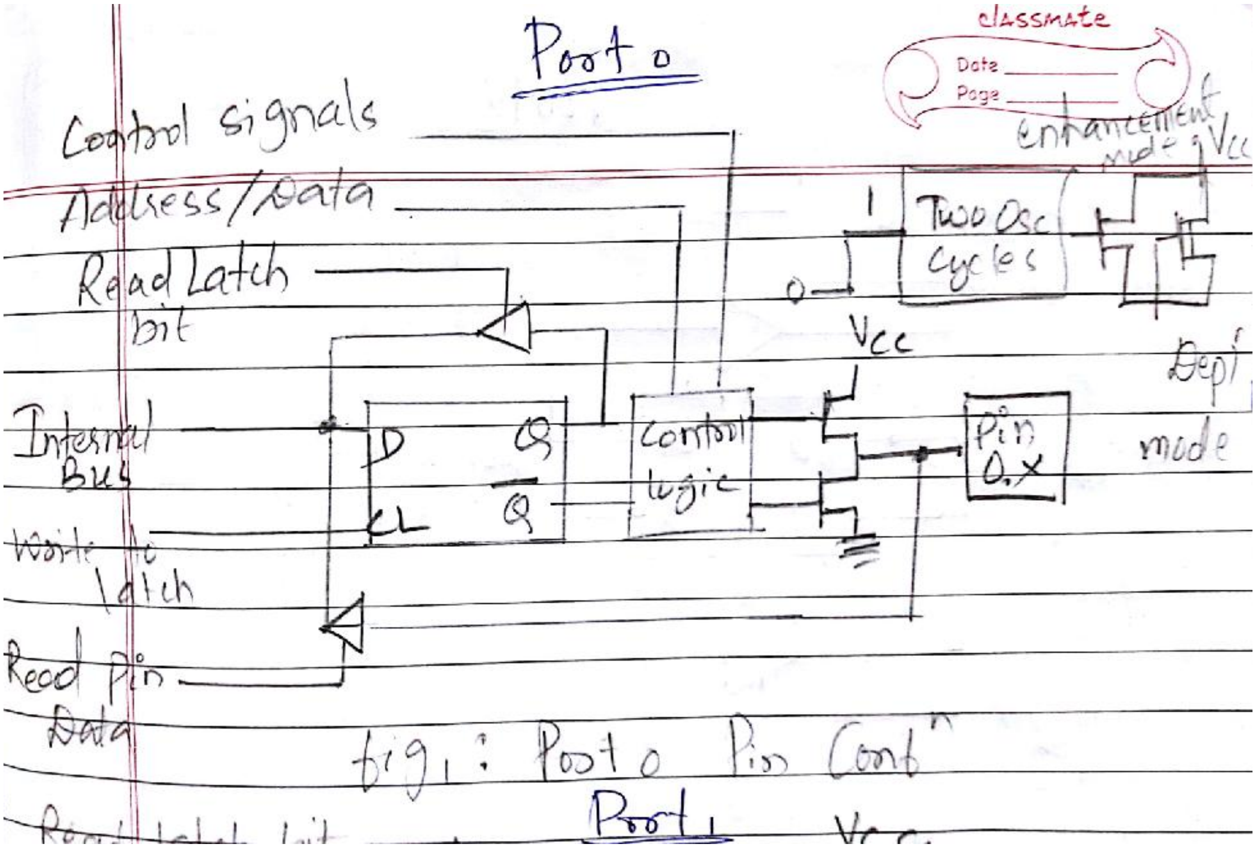
External Memory Addressing

Reading ROM using PSEN

Accessing RAM using RD & WR

Fig: External Memory Timing.

5. a Explain the working of Port0.0 bit .   5M



Explanation

Reset

Port0 : ~~*~~ By default all are i/p ports

a. Port0 as in i/p — 1 is written to, the latch. So both mosfets are off. Hence output pins have floats. hence whatever data written on pin is directly ~~to~~ read by read pin.

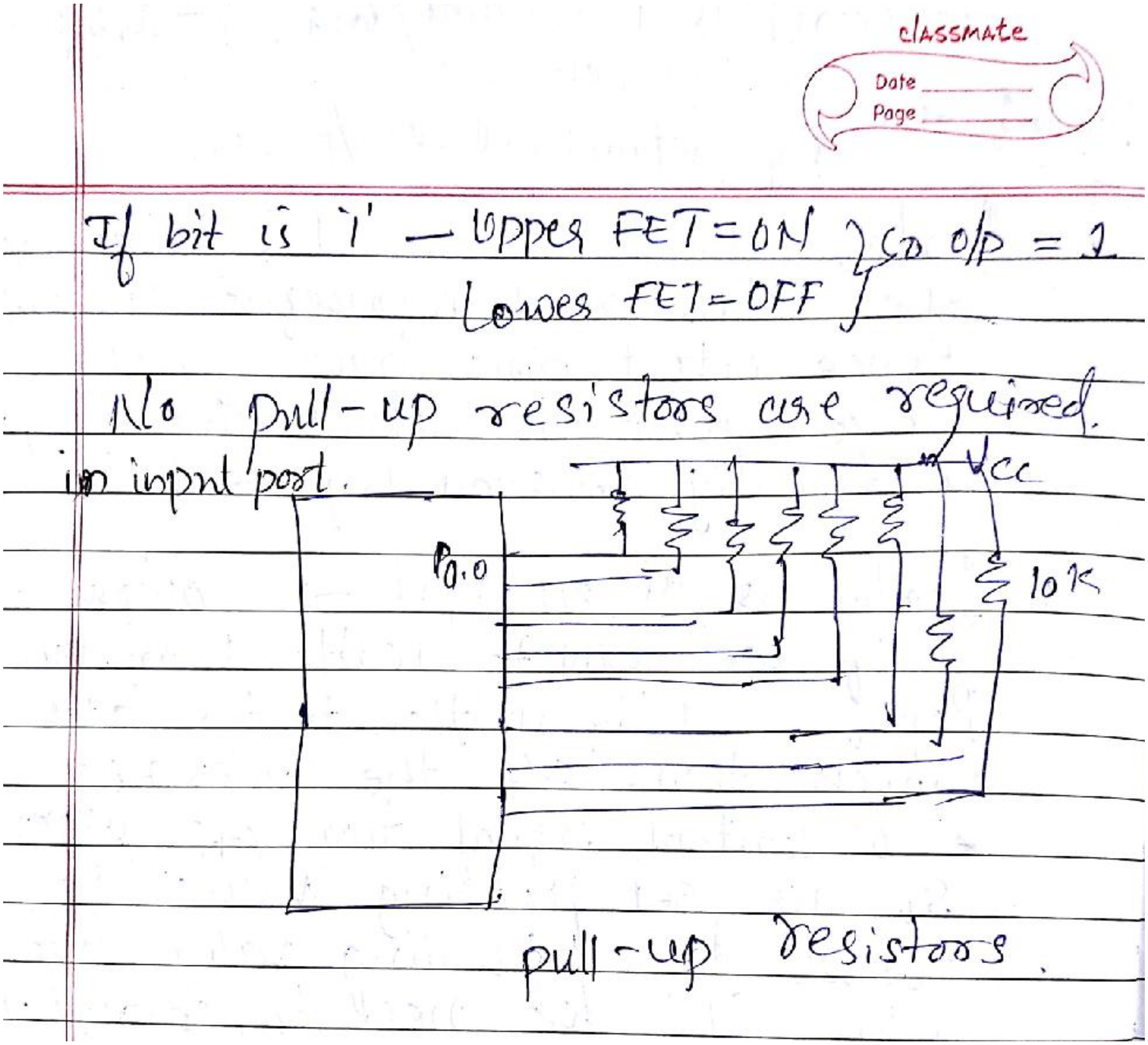
b. Port0 as an o/p port → output port

. If we want to write 1 on pin of Port0, '1' is written to the latch which turns 'off' the lower FET.
* '0' control signal turns off upper FET So we get floating value So to convert that floating value into logic '1' we need to connect the pull up resistor parallel to upper FET. Its done only when Port0 initialized as output Port.

If we want to write '0' on pin of port0, when '0' is written to the latch, the pin is pulled down by the lower FET. Hence o/p becomes zero

c. ~~for~~ add/data → Control is 1, add/data bus control the output driver FETs.

If bit is '0' → Upper FET is OFF, Lower FET is ON → o/p 0



If bit is '1' — Upper FET = ON } So o/p = 1
Lower FET = OFF }

No pull-up resistors are required. in input port.

pull-up resistors.

5. b What is stack memory. Explain PUSH and POP instructions in stack memory.

Soln= Definition of stack memory – 1M

Push and Pop- 2M each

PUSH and POP operation of stack memory in microcontroller

it s part of RAM in which data will store temporary during execution of program.

STACK work on last in first out principal.to store and retrieve data during program execution in stack push and pop instruction work for it.

PUSH:

its used to store data into stack.

POP:

to retrieve data from stack.

SP:

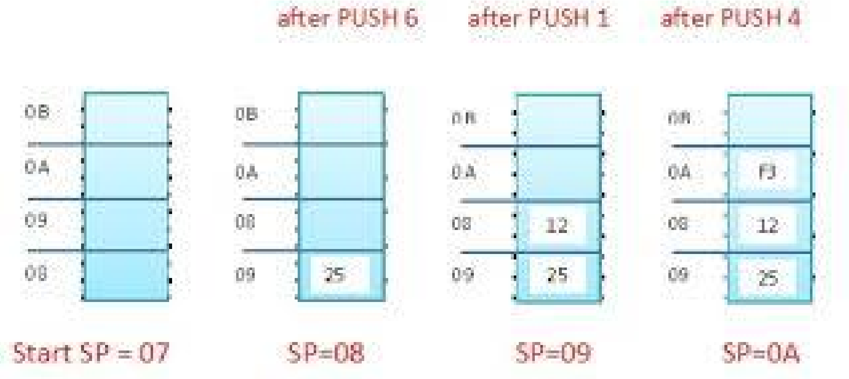
stack pointer is 8 bit register which store value of top of the stack.

by default stack pointer contain 07h.

PUSH AND POP OPERATION:

PUSH:

using push operation stack pointer increased first and then content of register or memory will store on that stack location which stored in SP.



here SP contain =07h

PUSH R6;SP increased by one and contain of R6 store into 08 location.

PUSH R1;SP increased again by one and contain of R1 stored into 09 location.

PUSH R4;SP increased again by one and contain of R4 stored into 0A location.

POP:

In this retrieve data first and then stack pointer decreased by one.

if we write

POP 20h;then content at 0Ah location will copy into 20h then stack pointer decrease by 1.

POP 21h;then content at 09h location will copy into 21h then stack pointer decrease by 1.

6. a Explain the data types and assembly directives in 8051 06 marks

p0

8051 Data Types and Directives.

Datatype

8051 has only one data type. It is 8 bits.

Directives.

Assembler directives give directions to the assembler, The machine codes are not generated for assembler directives in program.

1. DB (define byte)

It is used to define the 8 bit data. When DB is used to define data, the numbers can be in decimal, binary, hex or ASCII formats.

```
          ORG 500H
DATA1 :   DB   28                    // decimal
DATA2 :   DB   00110101 B            // Binary
DATA3 :   DB   39 H                  // HEX
DATA4 :   DB   "2959"                // ASCII N.
          ORG 518H
DATA5 :   DB "My Name is Joe"        // ASCII
                                        characters
```

```
[ MOV DPTR, # DATA,
  CLR A
  MOVC A, @ A+DPTR . ]
```

* "The DB directive is the only directive that can be used to define ASCII Strings larger than two characters."

* Either Single or double quotes can be used around ASCII strings.

2. ORG (origin)

The ORG directive is used to indicate the beginning of the address. The number that comes after ORG can be either in hex or in decimal. If number is not followed by H, it is decimal and the assembler will convert it to hex.

Some assemblers use ".ORG" instead of "ORG" for the origin directive.

3. EQU (equate)

This is used to define a constant without occupying a memory location. The EQU directive does not set aside storage for data item but associates a constant value with a data label so that when the label appears in the program its constant value will be substituted for the label.

COUNT EQU 25

;

MOV R3, #COUNT

When executing the instruction " MOV R3, # COUNT ", the registes r_3 will be loaded with the value 25

* If programmes wants to change the 'count' value, by the use of EQU, he can change it once & assembles will change all of its occusances, rather than changing of each & every ocurance

4 END directive.

* This indicates to the assembles the end of the source file. Its the last line of 8051 program, meaning that in the source code anything after the END directive is ignored by the assembler.

* Some assemblers use ".END" instead of "END".

6. b . Calculate the time required for executing a 2 machine cycle instruction if frequency is

a. 12Mhz b. 11.0592m

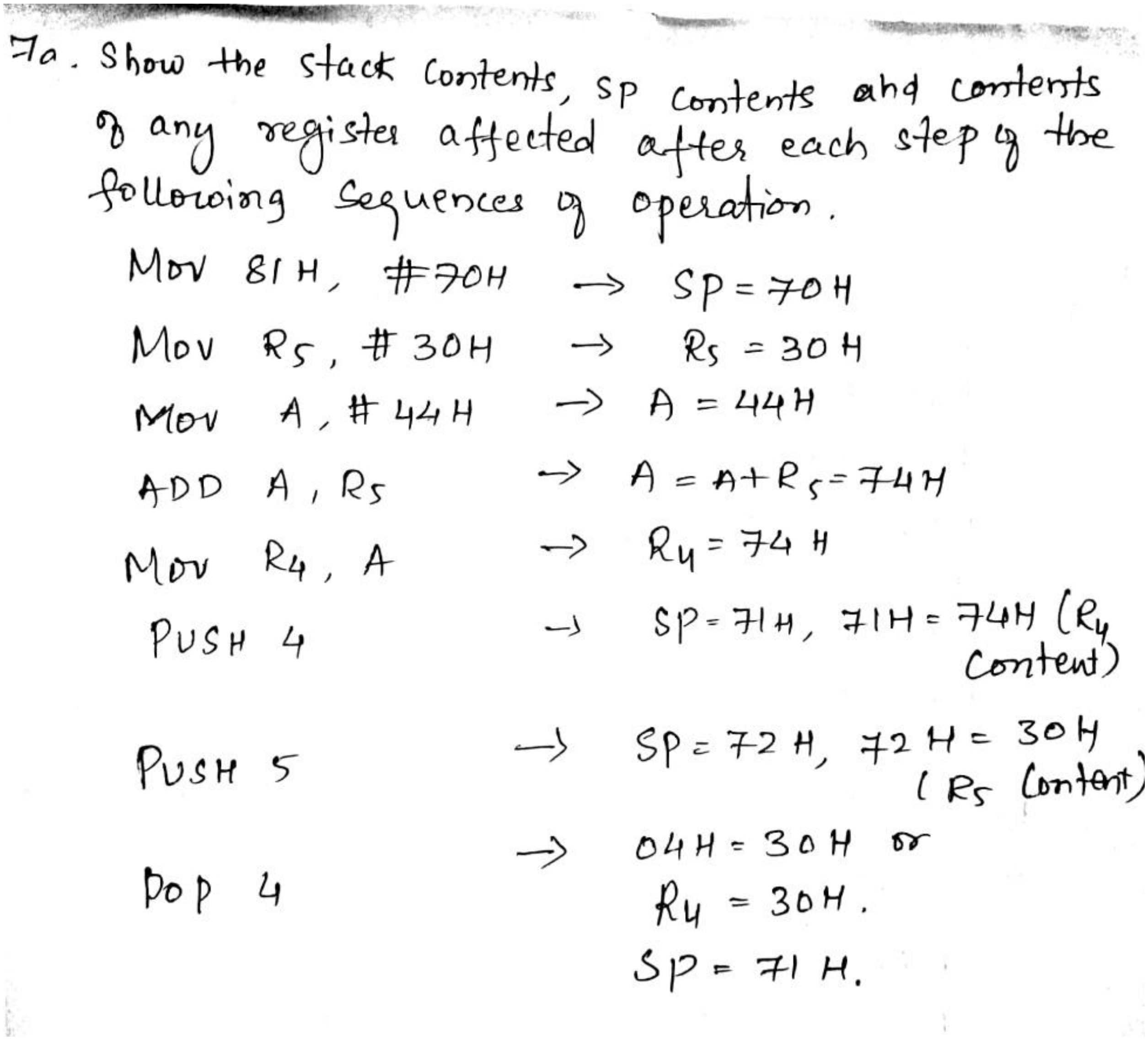
solution-

Time for 2 machine cycle = Time for 2 machine cycle *2

= (1/frequency)*12d *2

a. 12Mhz.
   =(1/12Mhz)*12*2=2 micro sec
b. 11.0592 MHz
   =(1/11.0592Mhz)*12*2= 2.17micro sec

7. a)



7a. Show the stack Contents, SP contents ahd contents of any register affected after each step of the following Sequences of operation.

Mov 81H, #70H → SP = 70H

Mov R_5, #30H → R_5 = 30H

Mov A, #44H → A = 44H

ADD A, R_5 → A = A + R_5 = 74H

Mov R_4, A → R_4 = 74H

PUSH 4 → SP = 71H, 71H = 74H (R_4 Content)

PUSH 5 → SP = 72H, 72H = 30H (R_5 Content)

Pop 4 → 04H = 30H or
R_4 = 30H.
SP = 71H.

7. b Explain the PSW in 8051 micro controller

2 marks - PSW Structure

2 marks - Explaination

Flags and the Program Status Word (PSW)

Flags are 1-bit registers provided to store the results of certain program instructions. Other instructions can test the condition of the flags and make decisions based on the flag states. In order that the flags may be conveniently addressed, they are grouped inside the program status word (PSW) and the power control (PCON) registers.

The 8051 has four math flags that respond automatically to the outcomes of math operations and three general-purpose user flags that can be set to 1 or cleared to 0 by the programmer as desired. The math flags include Carry (C), Auxiliary Carry (AC), Overflow (OV), and Parity (P). User flags are named F0, GF0, and GF1; they are general-purpose flags that may be used by the programmer to record some event in the program. Note that all of the flags can be set and cleared by the programmer at will. The math flags, however, are also affected by math operations.

The program status word is shown in Figure 3.4. The PSW contains the math flags, user program flag F0, and the register select bits that identify which of the four general-purpose register banks is currently in use by the program. The remaining two user flags, GF0 and GF1, are stored in PCON, which is shown in Figure 3.13.

Detailed descriptions of the math flag operations will be discussed in chapters that cover the opcodes that affect the flags. The user flags can be set or cleared using data move instructions covered in Chapter 5.

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	—	P

The Program Status Word (PSW) Special Function Register

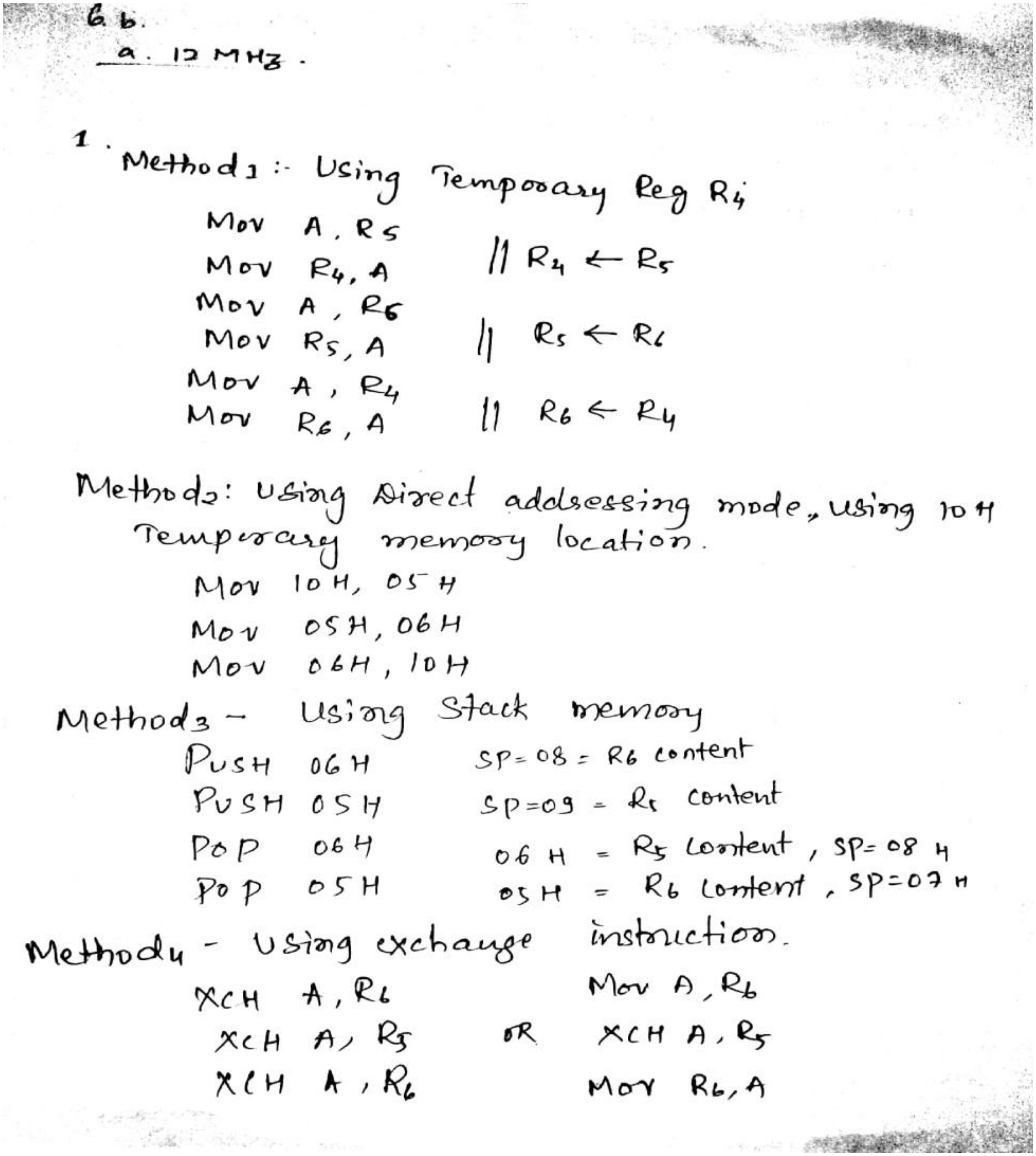
Bit	Symbol	Function		
7	CY	Carry flag; used in arithmetic, jump, rotate, and Boolean instructions		
6	AC	Auxiliary Carry flag; used for BCD arithmetic		
5	F0	User flag 0		
4	RS1	Register bank select bit 1		
3	RS0	Register bank select bit 0		
		RS1	RS0	
		0	0	Select register bank 0
		0	1	Select register bank 1
		1	0	Select register bank 2
		1	1	Select register bank 3
2	OV	Overflow flag; used in arithmetic instructions		
1	—	Reserved for future use		
0	P	Parity flag; shows parity of register A: 1 = Odd Parity		

Bit addressable as PSW.0 to PSW.7

FIGURE 3.4 ◆ PSW Program Status Word Register

8. a) Exchange the contents of R5 and R6 register contents with any four different methods.

2.5 M for each method. 2.5 *4 =10M



6 b.

a. 12 MHz .

1 . Method1 :- Using Temporary Reg R4

```
Mov  A , R5
Mov  R4, A      // R4 ← R5
Mov  A , R6
Mov  R5, A      // R5 ← R6
Mov  A , R4
Mov  R6 , A     // R6 ← R4
```

Method2: Using Direct addressing mode, using 10 H Temperary memory location.

```
Mov  10 H, 05 H
Mov  05 H, 06 H
Mov  06 H , 10 H
```

Method3 – Using Stack memory

```
PUSH  06 H      SP= 08 = R6 content
PUSH  05 H      SP=09 = R5 content
POP   06 H      06 H = R5 content , SP= 08 H
POP   05 H      05 H = R6 content , SP=07 H
```

Method4 - Using exchange instruction.

```
XCH  A , R6            Mov A , R6
XCH  A, R5      OR     XCH A , R5
XCH  A , R6            Mov R6, A
```