Unit-2 Micro-controller 8051 Architecture

2.1 Difference between micro-controller & general purpose Microprocessor -

5 marks

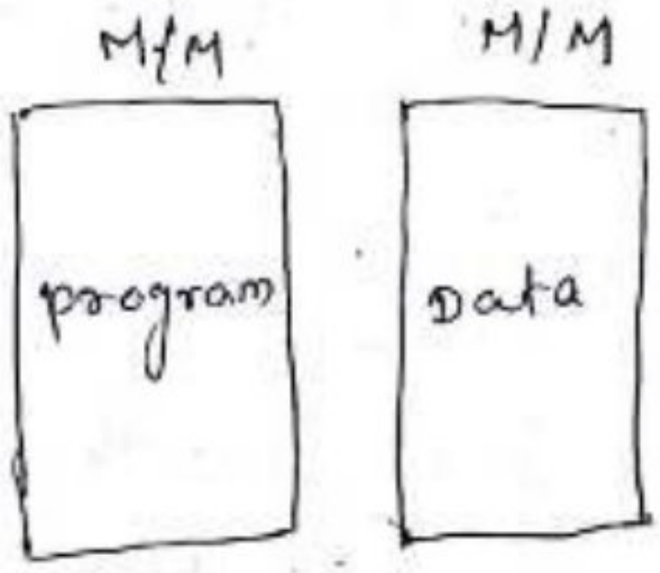
Microprocessor	Micro-controller
→ Used for general Purpose application.	→ Used for specific Purpose application.
→ No internal memory	→ Contains internal memory
→ No interfacing ckt like (timer & Counter)	→ Contains interfacing circuit.
→ Can not be used in compact system	→ can be used in compact system
→ Micro-processor have less no of register, hence more operations are memory based.	→ Micro-Controller have more no of register, hence the programs are easier to write.
→ It is heart of computer system.	→ It is heart of embedded system.
→ Cost is high	→ Cost is less.
→ Mainly used in personal computer.	→ Used mainly in washing machine, MP3 player

Scanned with CamScanner

→ Von-Neumann Architecture or princeton Architecture

→ Ex - Intel 8085, 8080, MC6800, Z80

→ Harvard Architecture

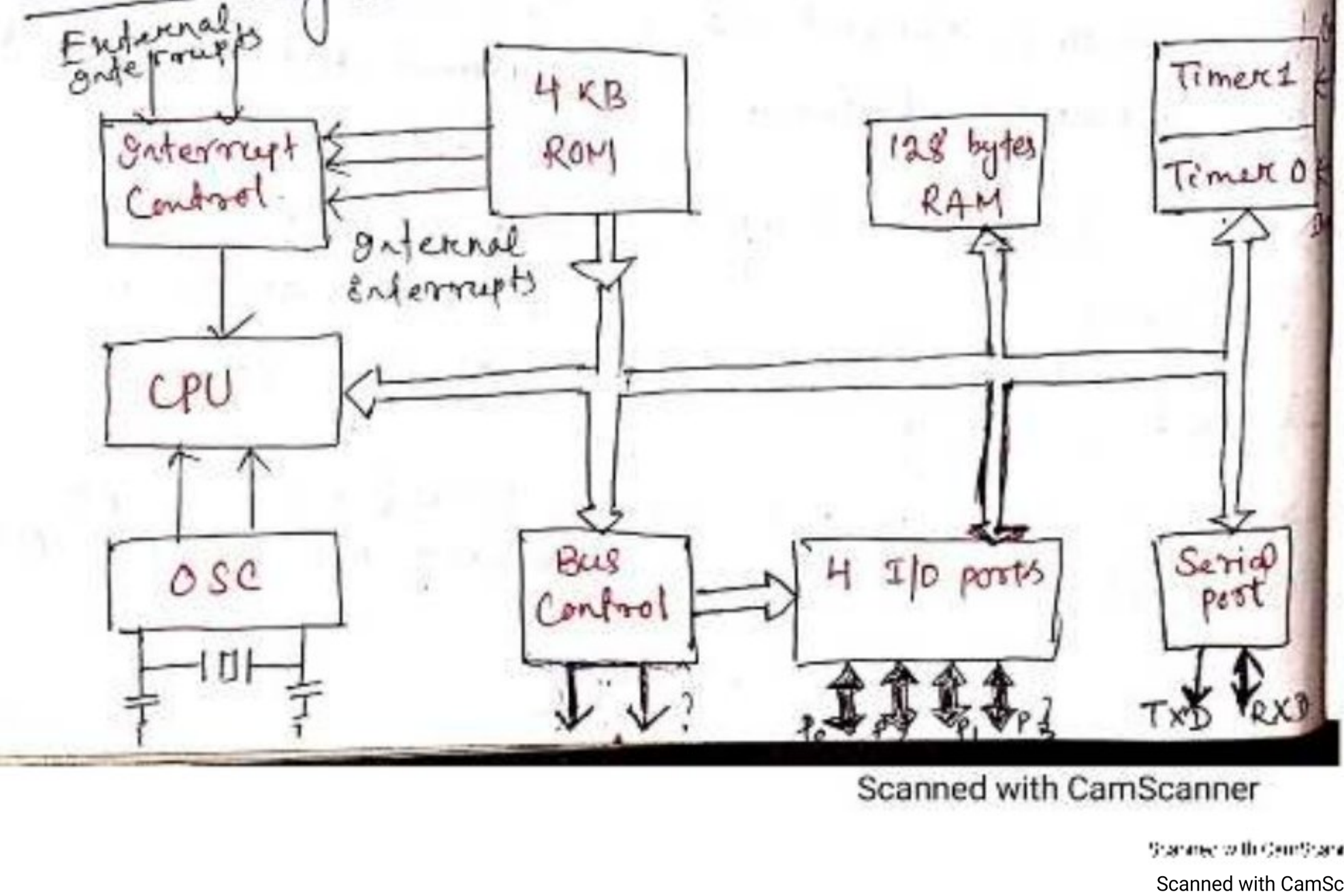


Ex - Intel 8051
Intel 80196
TMS 1000

2.2 Block Diagram of the Architectural of 8051.

5 Marks

Block Diagram of 8051.



Scanned with CamScanner

Following are the resources offered by the 8051 Microcontroller.

1. 8-bit CPU
2. On-chip Oscillator
3. 4 KB of ROM
4. 128 bytes of RAM
5. 21 special function Registers
6. 32 I/O lines
7. 64 KB address space for external data memory.
8. 64 KB address space for program
9. Two 16-bit timer/counters
10. Five source interrupt structure
11. Full duplex serial port
12. Bit addressability
13. Powerful bit processing capability.

* : The MCS-51 family includes pin compatible chip like:
  - → Intel 8031
  - → Intel 8051
  - → Intel 8751

The 8051 ~~have 4K~~ of family of MC is referred to as the MCS-51 architecture.

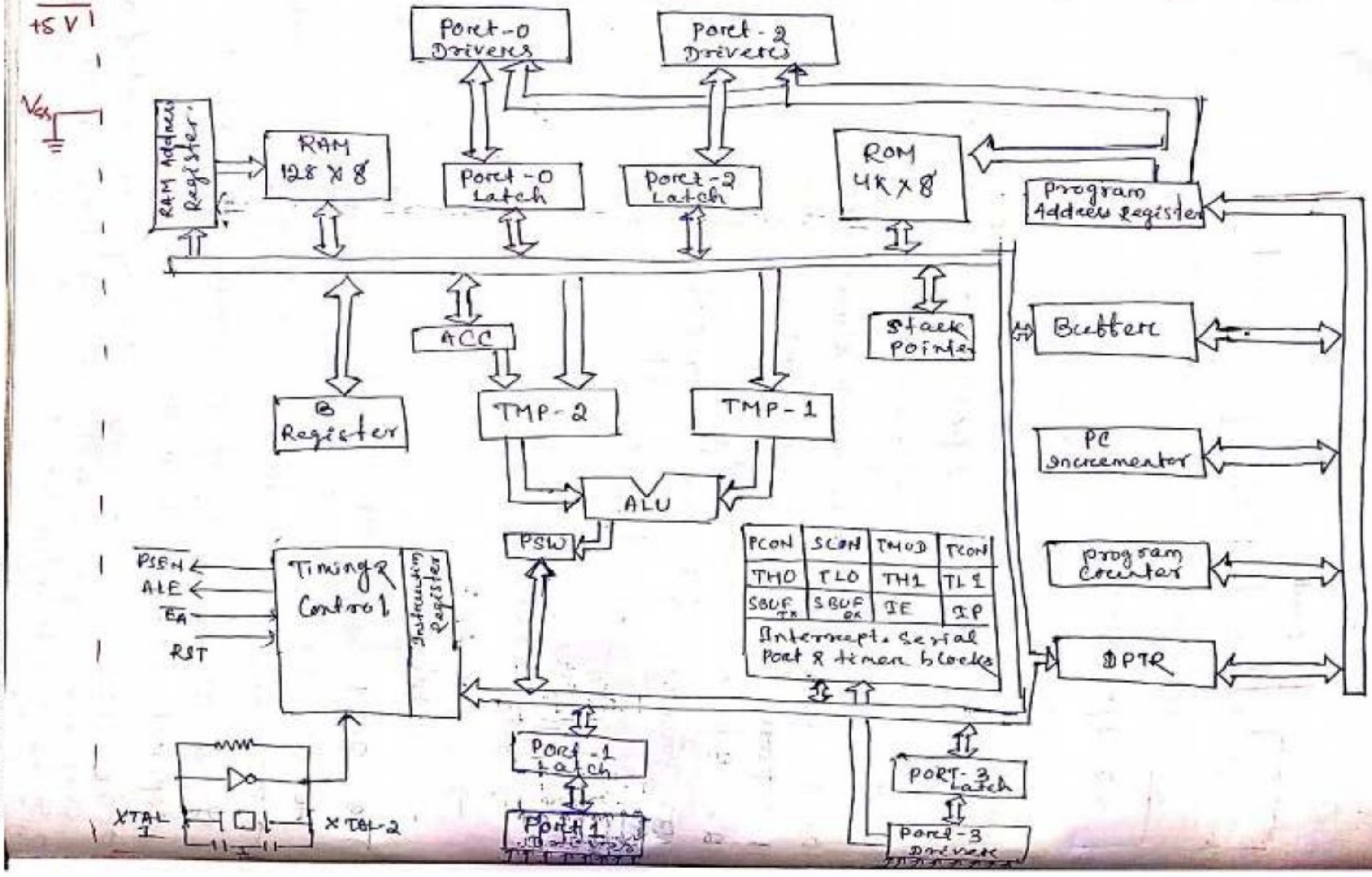
→ This have 8-bit data bus. They are capable of addressing 64K of program memory & separate 64K of data memory.

→ The 8051 have 4K code memory implemented as on Chip Read only. The 8051 have 128 bytes RAM.

→ It has two timer/counter, a serial port, 4 general purpose I/O I/P/O/P & interrupt control, logic with 5 source interrupt

Scanned with CamScanner

8051 Architecture

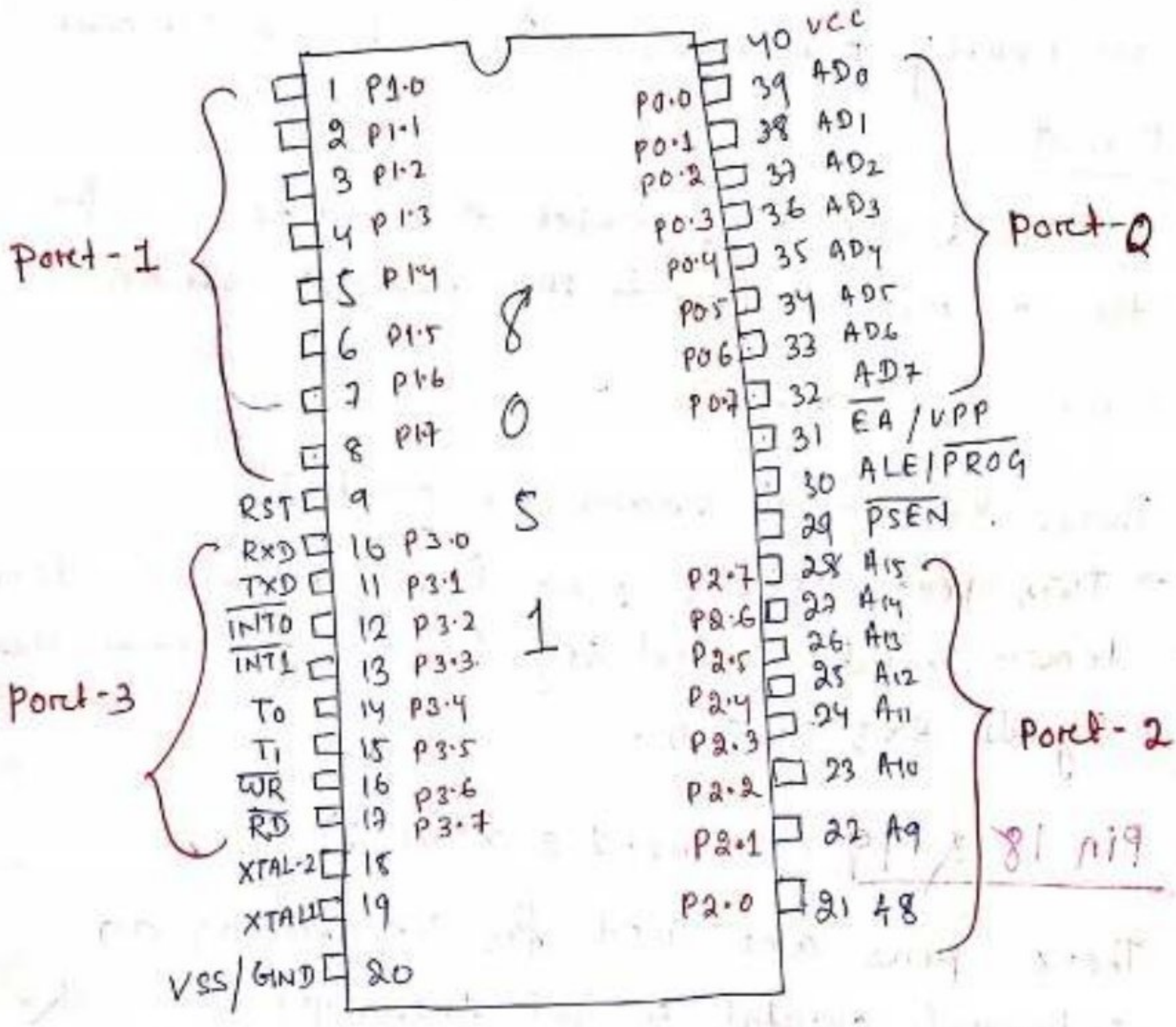


Scanned with CamScanner

2.3 PIN Diagram Features of the 8051

@

10-Marks



VCC & VSS (pin 20) ×40

VCC is the power supply pin. It is connected to a 5V regulated supply. Vss is the ground pin.

Port 0 (P0.0 to P0.7)

These are 8-bit bi-directional input/output port pins.

They are bit addressable.

Scanned with CamScanner

PIN 1 to 8 (Port 1)

These pins are known as port-1. This port does not serve any other functions. It is internally pulled up, bi-directional I/O port.

Pin 9

It is a RESET pin, which is used to reset the microcontroller to its initial values.

Pin 10 to 17 (Port 3)

These pins are known as port 3

→ This port serves some functions like interrupt, timer input, control signal, serial communication signals RXD & TXD.

Pin 18 & 19 (XTAL-1 & XTAL-2)

These pins are used for interfacing an external crystal to get the system clock.

PIN-20

This pin provides the power supply to the circuit.

PIN 21 to 28

These pins are known as port-2. It serves as I/O port. Higher order address bus signals are also multiplexed using this port.

Scanned with CamScanner

PIN-29

This is PSEN pin which stands for program store Enable. It is used to read a signal from the external program memory.

PIN 30

This is EA pin which stands for External access input. It is used to enable/disable the external memory interfacing.

PIN 31

This is ALE PIN which stands for Address Latch Enable. It is used to demultiplex the address data signal of port.

PIN. 32 to 39

These pins are known as port 0. It serves as I/O port.
Lower order address & data bus signals are multiplexed using this port.

PIN-40

This pin is used to provide power supply to the circuit.

Scanned with CamScanner

Programming Model of INTEL 8051

Def? User can access any of the area.

There are different types of resources are used in programming model of 8051.

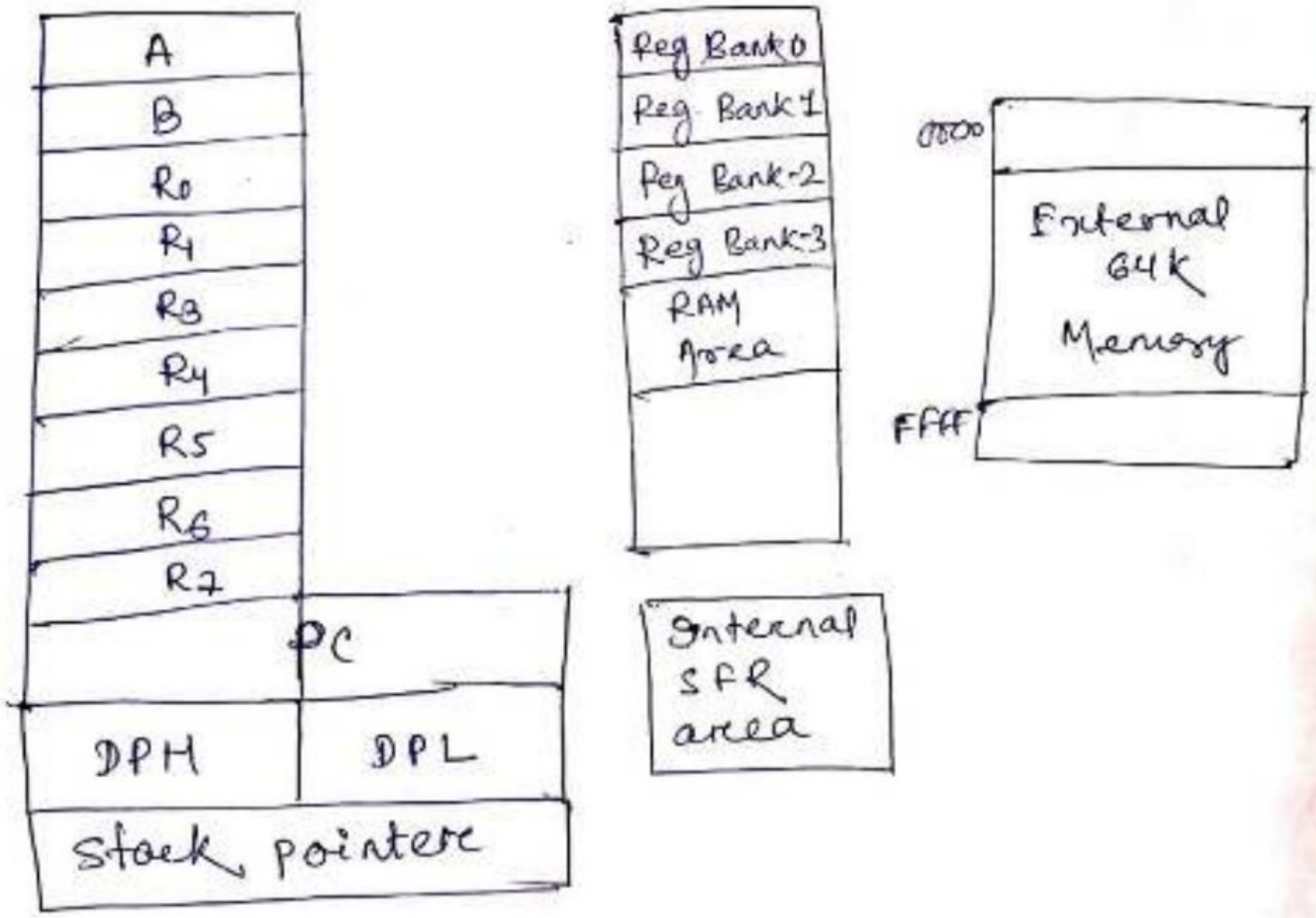
such as,

→ Memory

→ Special Function Register

→ Program status word

The user will have to refere to these resource quite often during the exercise of writing.



Scanned with CamScanner

8051 Register

SFR(Special Function Register)

SFR can be seen as a sort of control panel for managing & monitoring the microcontroller.

→ Everey register & each of the belonging bits has its name, specified address in RAM & strictly defined role (eg. controlling timer, interrupt, serial connection etc)

→ Although there are 128 available memory slot for allocating SFR register.

→ The rest has been left open intentionally to allow future upgrades while retaining the compatibility with earlier model.

→ This fact makes possible to use programs developed for obsolate models long ago.

Name	Function	Hex Address	Bit Addressable
A	Accumulator	E0	Yes
B	Arithmatic	F0	Yes
DPTR	Data pointer	—	
DPH	Address Ext. Memory	83	No
DPL	Address Ext. Memory	82	No

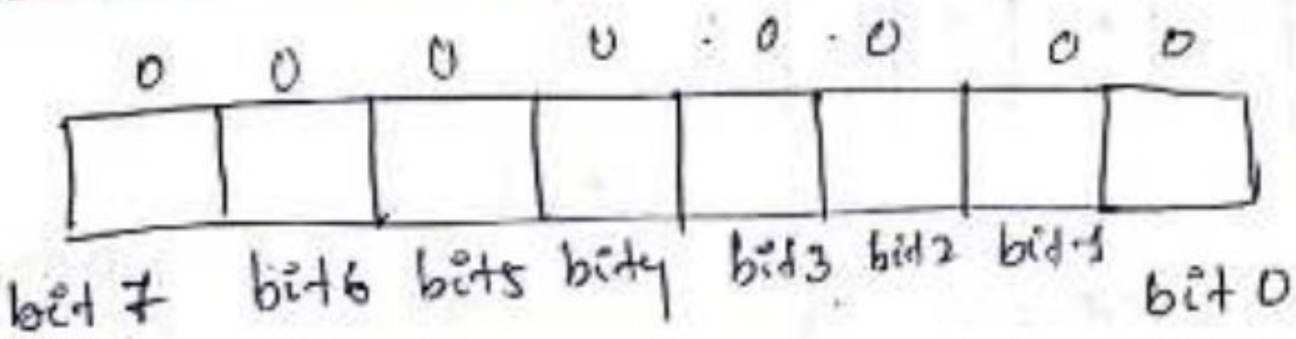
Scanned with CamScanner

Name	Function	Hex Address	Bit Addressable
IE	Interrupt Enable Control	A8	Yes
Ip	Interrupt priority	B8	Yes
P0	I/O port Latch	80	Yes
P1	"	90	Yes
P2	"	A0	Yes
P3	"	B0	Yes
PCON	Power Control	87	No
PSW	Program Status	D0	Yes
SCON	Serial port Control	98	Yes
SBUF	Serial port Data Buffere	99	No
SP	Stack pointere	81	No
TMOD	Timere/ coundere mode Control	89	Yes

Scanned with CamScanner

Name	Function	Hex Address	Bit Addressable
TWN	Timer/counter control	88	Yes
TL0	Timer-0 Low byte	8A	No
TH0	Timer 0 High byte	8B	No
TL1	Timer 1 Low byte	8C	No
TH	Timer 1 high byte	8D	-

Accumulator



This is a general purpose register, which stores the result before programing any operation upon an operand.

→ All the results obtained from arithmatical operations performed by the ALU are stored in the accumulator.

→ Data to be moved from one register to another must go through the accumulator.

→ More than half instructions used by 8051 microcontroller use somehow the accumulator.

Scanned with CamScanner

B- Register

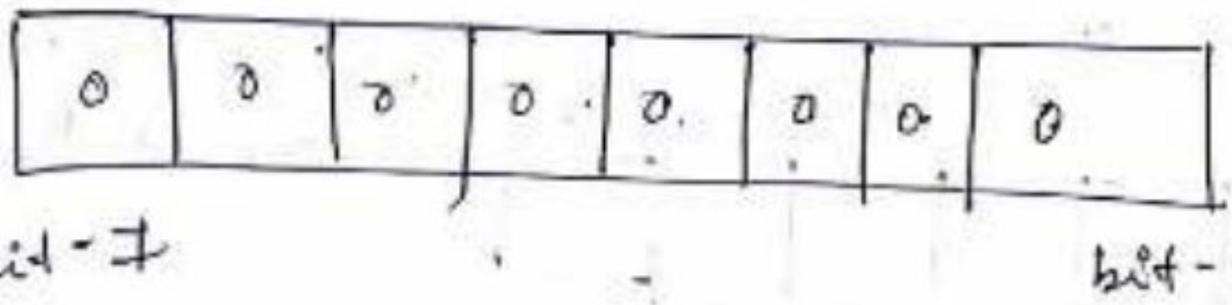
Multiplication & Division can be performed only upon numbers stored in the A & B registers.

→ All other instruction in the program can use as a spare accumulator.

→ Instructions of Multiplication & Division can be applied only to operands located in registers A & B.

→ Other instructions can use this register as a secondary accumulator.

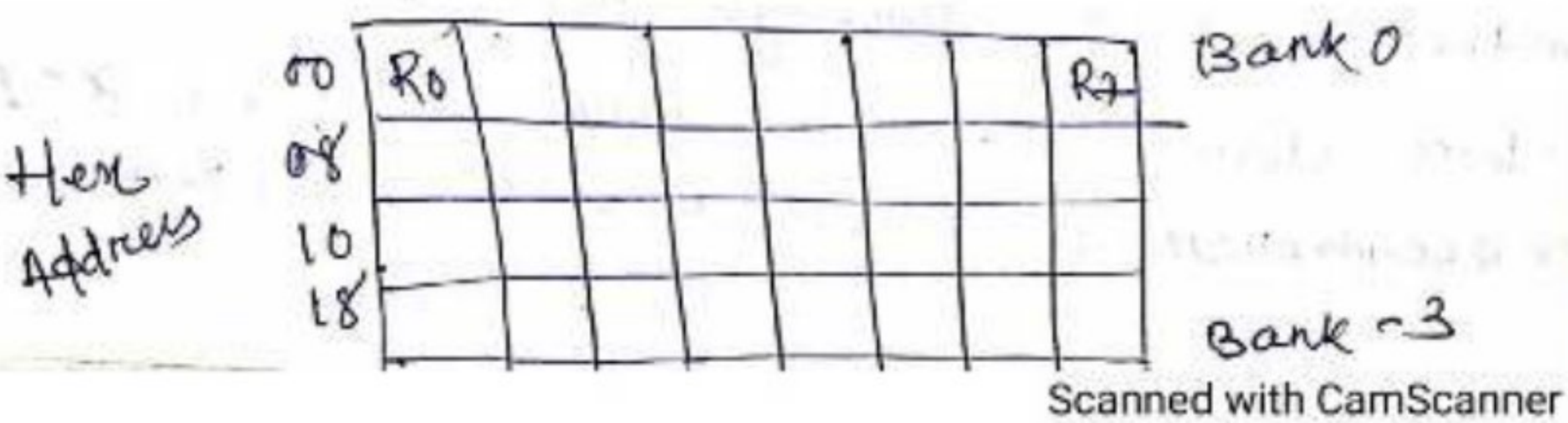
B



→ During the process of writing a program, each register is called by its name so that their exact address are not important for the user.

→ During compilation, their names will be automatically replaced by appropiate address

R- Registers



Scanned with CamScanner

This is a common name for 8 general purpose registers ($R_0, R_1 \ldots R_7$)

→ They occupy 4 banks with PAM.

→ Similar to RC, they are used for temporary storing variables & intermediate results during operation.

→ Active of Bank depends upon the PSW Register

→ Active bank is a bank the register of which are currently used.

EX · $(R_1 + R_2) - (R_3 + R_4)$

MOV A, R3 ; - Move a number from R_3 into Accumulator.

ADD A, R4 ; - Add no from R_4 to Accumulator

MOV R5, A ; - Temporarily move the Result from Accumulator into R5.

MOV A, Ri ; - Move the no from R_1 into Accumulator.

ADD A, R2 ; -Add no from R_2 to Accumulate

SUBB A, R5 ; - Subtract number from R5 ($R_3 + R_4$)

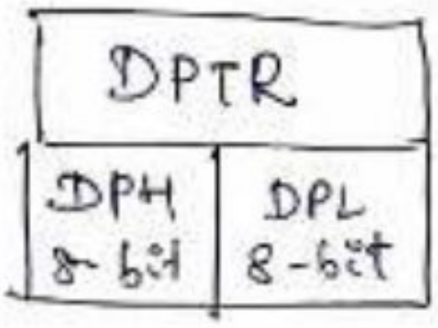
Scanned with CamScanner

Data pointer (DPTR)

The data pointer registere is commonly known as DPTR.

→ It is a 16-bit registere use to hold the 16-bit address this registeres is used as a point to hold the address of external memory location to access.

→ It is made of two 8-bit regester.



Program Counter

→ It has no-onchip RAM memory.

→ It is a 16 bit register & always point & to address of next instruction in the memory to be executed.

→ 16 bit register can be access from (0000 to FFFF)

Stack pointer

Stack refers to an area of a internal RAM that is used to store & retrieve the stack & called stack pointer.

→ 8-bit stack pointer is used by 8051 to hold the external ROM address that is called the top of the stack.

Scanned with CamScanner

I/o ports

8051 has 4 ports P0, P1, P2, P3 - All ports have 8 bit length

→ All the ports are bidirectional

→ Port 0 is an open drain bidirectional i/o port

→ It is also multiplexed because address & data bus during access to external memory

P0.0 → AD0

P0.1 → AD1 ....

Port-1

It is an 8-bit bidirectional i/o port used for user purpose.

Port-2

It is also an bi-directional

→ These also provides higher order address byte during the access to external memory

P2.0 → A8 , P21 → A9 .

Port-3

It is a bi-directional i/o ports used for general purpose & are for some special purpose as below.

P3.0 → RXD

P3.1 → TXD

P3.2 → INT0

P3.3 → INT1

P3.4 → T0

P3.5 → T1

P3.6 → WR

P3.7 → RD

Scanned with CamScanner

It is always follow the operation of last in first out for storing of any data of CPU register on the stack is called push & retrieving the data is called POP that means the stack always use push & pop instruction.

PSW (Program Status Word)

It is an 8-bit register commonly a flag registere.

→ It contains

→ Math Flag
→ program Flag
→ Register select bit and used for managing the program during run time

7	6	5	4	3	2			
CY	AC	F0	RS1	RS0	OV	X	X	P

Register select.

Parity

It contains of ACC is even then this bit set to 1, else it is 0.

→ Basically used for data tx in serial port.

→ bit-1 → Intended for the upcoming MC model & not used.

Scanned with CamScanner

bit 2 → Overflow bit.

⇒ If result of mathematic operation exceeding 255 decided, then OV is said to be 1, otherwise 0.

Register Select

0	0	→ Bank-0	— 00 - 07 H
0	1	→ Bank-1	— 08 - 0F H
1	0	→ Bank-2	— 10 - 17 H
1	1	→ Bank-3	— 18 - 1F H

F0 - Flag-0

This is general purpose bit available for use.

AC It is used for BCD operation

CY Carry flag is auxiliary bit used for all arithmatical instruction.

OV It occures, when the result of a arithmatica operation is largere than 255 & can not be stored in one register.

→ Overflow condition causes OV bit be set(1), otherwise; it will be cleared (0).

Scanned with CamScanner

Timers / Counters

8051 microcontrollers employ a quartz crystal chore frequency & highly stable & accurate.
→ It has two timers & counters mark as To & T_1.
→ There purpose to measure the time & count external occurances but can also be used as clock in serial communication purpose which is called boud rate.

TMOD (Timer Mode)

TMOD is dedicated to 2 timer To & T_1 & cache Considere to be deplicate in 4-bit register which controls the action of 1 of the timers

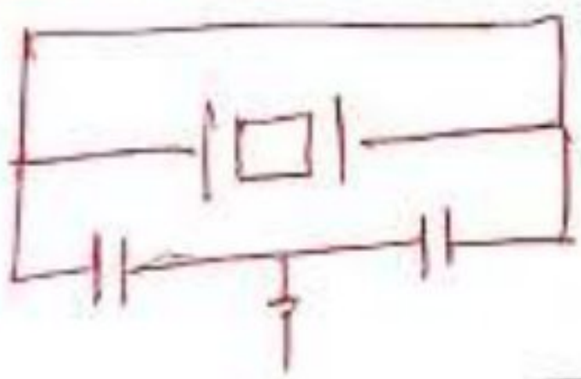
TCON

TCON has a control bit & flags for the timer.

UART

Universal Asesynchronous communication.
→ It is a serial port capable of sending & receiving data simultaneously & there is an SBUF register which holds the data to be send to the line & accept data from the line.

Oscillator ckt



Scanned with CamScanner

Register ($R_0 - R_7$)

00 — 00H - 07 H
01 → 80H - 0F H
10 → 10H - 17 H
11 → 18H - 1F H

Timing Control

The internal timing & control signals are $\overline{RD}$, $\overline{WR}$, $\overline{ALE}$ are generated by the timing & control unit for various operation.

Interrupt parity Control

The interrupt parity control is used for various interrupt control in 8051 operation.

External Interrupt 0 - INT0
Timer interrupt 0 - TF0
External interrupt 1 - INT 1
Timer interrupt 1 - TF-1
Serial Communication - RI + TI

— X —

21/1/2020

Assignment

Scanned with CamScanner

Ch-3 8051 Addressing Modes & Instruction Set (35)

3.1 Explain different Addressing mode of 8051 (3-5M)

1B | opcode | → Types of operation to be performed

2B | opcode | operand | → 8 bit data / Address

3B | opcode | operand | MSB of operand |

Opcode

It is the representation of operation or action to be performed.

Operand

Where the operation to be performed. It is either data or address & it is always follows the opcode.

What is the importance of Addressing Mode ?

Various methods for accessing data or address to be needed is called addressing mode.

→ It defines the way in which operand are accessed by instructions

→ An instruction is used to load or transfer the data from a source to a destination.

Scanned with CamScanner

→ The source may be any register, internal memory, external memory or any one of the I/O port.

mov A, 11 30
↓ ↓ Source.
Destination

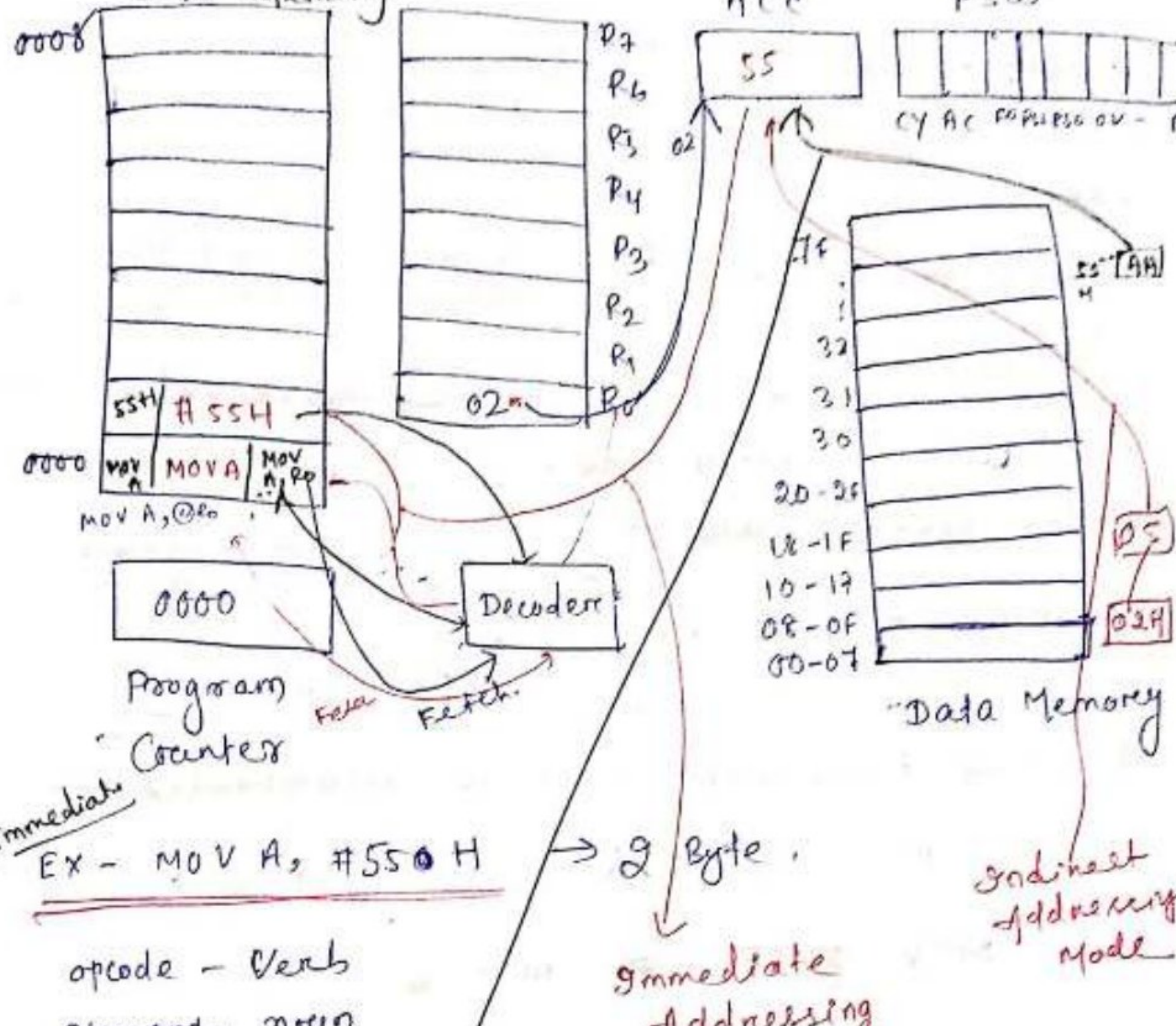
→ The destination may be any register, memory, internal or external or I/O devices.
→ The data is called normally operand, & the various method for access needed in the execution of an instruction is called addressing mode.

Types of Addressing Mode

→ Immediate A.M.
→ Register A.M
→ Direct A.M
→ Indirect A.M
→ Relative A.M
→ Absolute A.M
→ Bit. Inherent A.M
→ Bit Direct A.M
→ Long AM.

Scanned with CamScanner

Addressing Mode



Immediate

Ex - MOV A, #550H → 2 Byte.

opcode - Verb

operand - noun

Immediate Addressing

Direct Addressing

Direct

Ex - (MOV A) [55H] → 2 Byte.

opcode Address

Registere Addressing

Ex MOV A, R0 → 1 Byte

Indirect Addressing Mode

MOV A, @ R0 → 1 Byte

Scanned with CamScanner

Immediate Addressing Mode

The data is provided as per the instruction which is immediately executed during the execution.

→ Source operand is a constant rather than variable.

→ The constant operand can be incorporated into the instruction byte.

→ The operand are preceeding by '#' symbol

→ MOV A, # FF

→ Load immediate data to accumulator

MOV R0, # 6 , 06 moved to R0

→ MOV DPTR, #. data 16

→ ADD A, # Data

→ SUBB A, # Data

→ MOV Rn, # Data.

Registere Addressing

This involves the use of registers to hold the data to be manipulated.

→ In this mode the selected register bank containing registere R0 - R7 can be accessed by certain instruction which carry 3-bit register specification with in the opcode of the instruction.

Scanned with CamScanner

MOV A, R0

MOV R1, A

ADD A, Rn

SUBB A, Rn

Direct Addressing

It provides to allow us access internal data memory including SFR.

→ It uses the lower 128 bytes of internal RAM & SFR.

MOV A, 33H

D S

operand is specified by 8-bit address field in the instruction that is lower 128 byte of internal data RAM & SFR can be directly accessed by single byte address.

MOV 32, R1

ADD A, Rn

SUBB A, R0

MOV Rn, A

Indirect Addressing Mode

SPR are not addressed by this mode.

→ It uses the register to hold the actual address that will be used in Data movement

MOV A, @ R0

Scanned with CamScanner

In this addressing the instruction specified a register which contains the address of the operand both internal & external RAM can be addressed.

→ In this mode R_0 or R_1 of selected bank of Stack pointer may operate as pointer reg. for 8-bit address.

→ The indirect addressing represented by @ before R_0 & R_1

EX - MOV @ R_1, A

ADD A, @ R_0 (Add the content of address specified by R_0)

SUBB A, @ R_1 (subtract content of address specified by R_1 from accumulator)

Index Addressing

A separate register either program counter or DPTR is used to hold the base address & used to hold the offset address.

→ It allows a byte to be accessed from the program memory whose address is calculated as the sum of base Register & index Register, Accumulator.

MOVC A, @A + DPTR

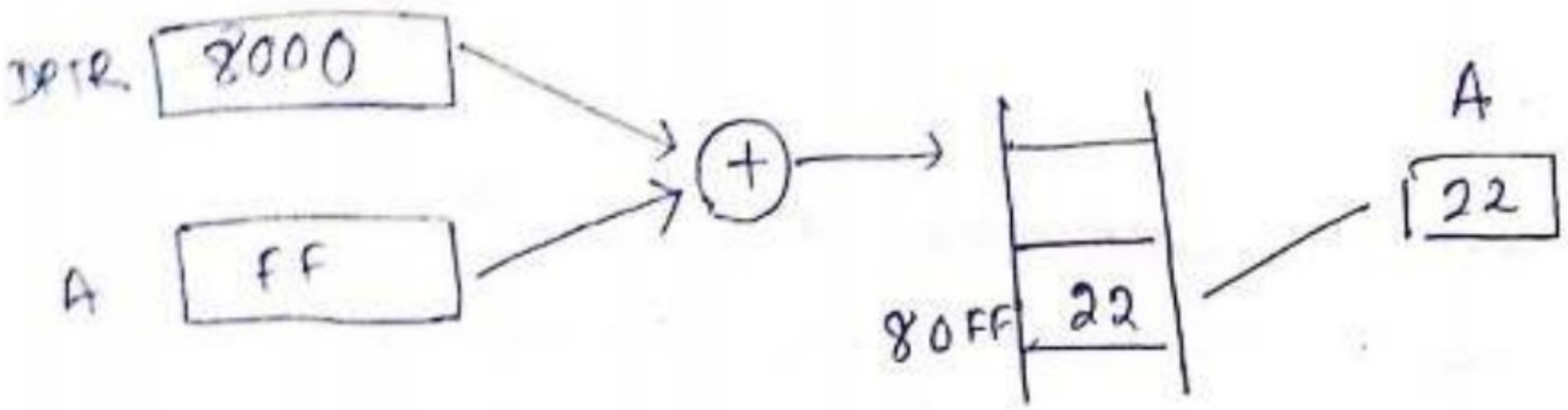
ACC ← ACC + DPTR

← 02H + 0E7E H

← 0E80 H

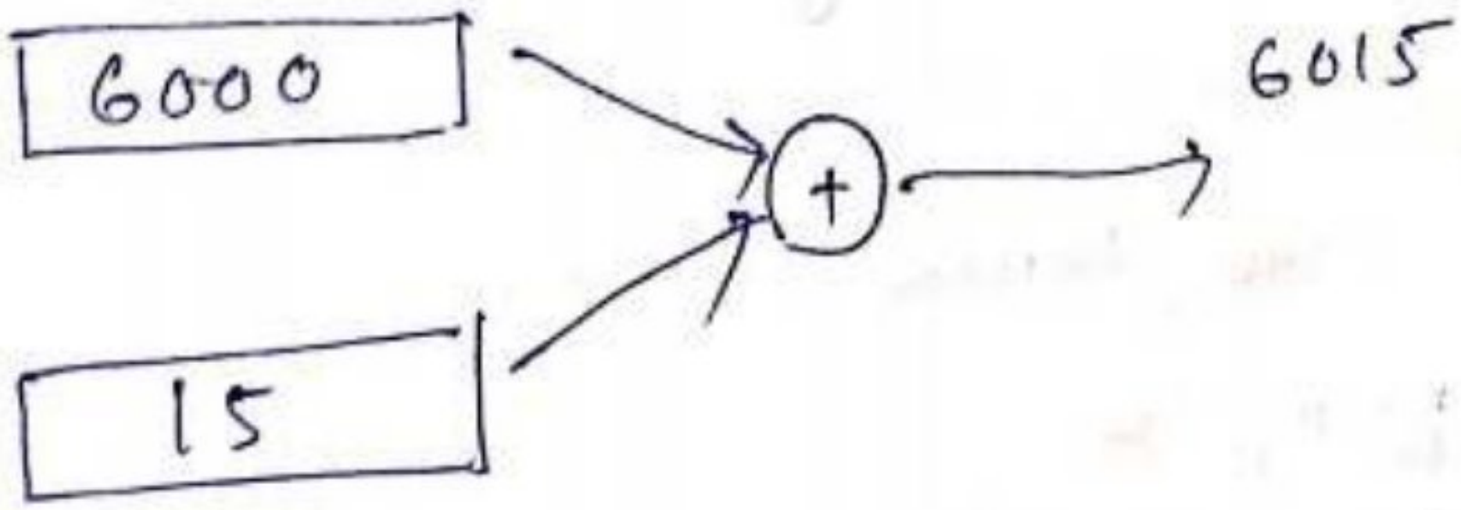
← 56 H

Scanned with CamScanner



MOVC

Move a byte of data from DPTR whose address can be determined by the sum of accumulator & the content of DPTR to Accumulator.



Scanned with CamScanner

8051 Instruction Set

Instruction

It is a set of commands applied to Micro-Controller to perform certain specified operation.

→ Total Instruction in 8051 is 255.

→ The instructions are group into 8 groups such as → Arithmatic
→ Logical
→ Data Transfer
→ Boolean
→ Branching

Data Transfer Instructions

It is divided into three classes

→ General purpose
→ Accumulator specific
→ Address object.

General purpose Transfer

→ MOV performs a bit or a byte transfer from the source operand to the destination operand.

→ PUSH increment the SP register & then transfers a byte from the source to the stack element, currently addressed by stack pointer.

→ POP transfers a byte operand from the stack element addressed by the SP register to the destination & then decrements stack Pointer.

Scanned with CamScanner

MOV < dest. byte > , < sor. byte >

(Move byte variable)

MOV A, Rn

A ← Rn

MOV A, direct

A ← direct

MOV A, @Ri (where Ri = R0 or R1)

A ← ((Ri))

MOV A, # data

A ← # data

MOV Rn, A

(Rn) ← (A)

MOV Rn, direct

(Rn) ← direct

MOV Rn, # data

(Rn) ← # data

MOV direct, A

direct ← (A)

MOV direct, Rn

(direct) ← (Rn)

Scanned with CamScanner

MOV direct, direct

(direct) ← (direct)

MOV direct, @Ri

direct ← ((Ri))

MOV direct, # data

(direct) ← data

MOV @Ri, A

((Ri)) ← (A)

MOV @Ri, direct

((Ri)) ← direct

MOV @Ri, # data

((Ri)) ← # data

It may be noted that the contents of a register in the register bank can not be transferred to another register in the register bank, since the instruction MOV Rm, Rn is not present.

MOV <dest-bit>, <src-bit> Move bit data

The Boolean variable, indicated by the second operand is copied into the location specified by the first operand.

MOV C, bit

C ← bit

MOV bit, C

PUSH direct (push onto stack)

SP ← SP+1, ((SP)) ← direct

Scanned with CamScanner

pop direct (Pop from stack)

direct ← ((SP))

SP ← SP - 1

Accumulator - Specific Transfer

XCHA, <byte> (Exchange accumulator with byte variable)

The XCH exchanges the content of Accumulator with those of the indicated variable.
The source/ destination operand can use register, direct, or register indirect addressing

XCH A, Rn

XCH A, direct

XCH A, @Ri (where Ri = R0 or R1)

(A) ↔ ((Ri))

XCHDA, @ Ri (Exchange digit)

(A) 3 - 0 ↔ ((Ri)) 3 - 0

MOVX <dest-byte>, <src-byte> (Move external)

The MOVX instructions transfer data between the accumulator & a byte of the external data memory.

MOVX A, @Ri

MOVX @ Ri, A

MOVX A, @ DPTR

MOVX @ DPTR, A

Scanned with CamScanner

MOVC A, @A + [base-reg] (Move code byte)

The MOVC instructions load the accumulator with a code byte, or a constant from the program memory, .

→ No flags are affected.

MOVC A, @A + DPTR

(A) ← ((A) + (DPTR))

MOVC A, @A + PC

(PC) ← (PC) + 1

A ← ((A) + (PC))

Address - object Transfer

MOV DPTR, # data loads 16-bit of immediate data into a pair of destination registers, DPH & DPL.

MOV DPTR, # data 16 (Load data pointer with a 16-bit Constant

The data pointer is loaded with the 16-bit constant indicated.

→ No flags are affected

(DPTR) ← # data 16

Example: The internal RAM location 30H holds 40H. The value stored in RAM location 40H is 10H. What will be the value in locations 30H 40H after the following instructions are executed?

Scanned with CamScanner

MOV R0, # 30H → (R0) = 30H

MOV A, @R0 → (A) = 40H

MOV R1, A → (R1) = 40H

MOV A, 7FH → (A) = 7FH

MOV @Ri, A → RAM(40H) = 7FH

XCHD A, @R0 → RAM(30H) = 4FH, (A) = 70H

Arithmatic Instructions

The 8051 provides the basic mathematical operations like addition, subtraction, multiplica division, increment, decrement etc.

→ Only the 8-bit operations using unsigned arithmatic are supported directly.

→ The overflow flag permits the addition & subtraction operations to serve both unsigned & signed binary register.

Addition (There are 4 addition operation)

→ ADD A, <src.byte> Add.

→ ADD adds the byte variables indicated to the accumulator, leaving the result, in the accumula

→ Four source operand addressing modes are allowed - register, direct, register indirect & immediate.

ADD A, Rn

(A) ← (A) + (Rn)

Scanned with CamScanner

ADD A, direct

(A) ← (A) + direct

ADD A, @ Ri

(A) ← (A) + ((Ri))

ADD A, # data

(A) ← (A) + # data

ADDC A, < src. byte > (Add with carry)

The ADDC simultaneously adds the byte variables indicated, the carry flag & the accumulator contents, leaving the result in the accumulator,

→ Four source operand addressing are allowed.

ADDC A, Rn

(A) ← A + C + (Rn)

ADDC A, direct

(A) ← (A) + (C) + Direct

ADDC A, @ Ri

(A) ← (A) + (C) + ((Ri))

ADDC A, # data

(A) ← (A) + (C) + # data.

DA A (Decimal - adjust accumulator for addition)

The DA A adjusts the 8bit value in the accumulator, resulting from the earliere addition

Scanned with CamScanner

of two variables (each in Packed BCD), producing two 4-bit digits.

→ Any ADD or ADDC instruction may have been used to perform the addition.

INC DPTR (Increment Data pointer)

(DPTR) ← (DPTR) + 1

Subtraction

→ SUBB performs a subtraction of the second source operand from the first operand, subtract 1 if the C-flag is found previously set.

→ DEC (decrement) performs a subtraction of 1 from the source operand.

SUBB A, <src-byte> (subtract with borrow)

SUBB subtracts the indicated variable & carry flag together from the accumulator, leaving the result in the accumulator.

SUBB A, Rn (A) ← (A) - (C) - (Rn)

SUBB A, direct (A) ← (A) - (C) - direct

SUBB A, @Ri (A) ← (A) - (C) - ((Ri))

SUBB A, # data (A) ← (A) - (C) - # data

DEC byte (Decrement)

DEC A, (A) ← (A) - 1

DEC Rn, (Rn) ← (Rn) - 1

DEC Direct, (direct) ← direct - 1

DEC @Ri, ((Ri)) ← ((Ri)) - 1

Scanned with CamScanner

INC < byte > (Increment)

→ INC increments the indicated variable by 1

→ NO flags are affected.

→ Four addressing modes are allowed.
A, Register, direct, register indirect

INC A
(A) ← (A) + 1

INC (Rn)
(Rn) ← (Rn) + 1

INC direct
(direct) ← direct + 1

INC @ Ri
((Ri)) ← ((Ri))

Multiplication

MUL performs an unsigned multiplication of register A by register B, returning a double byte result.

MUL AB (Multiply)
The MUL AB multiplies the unsigned 8-bit integers in the accumulator & register B.
→ The lower order byte of the 16-bit product is left in the acc^r, & higher order byte in register B.

Scanned with CamScanner

If the product is greater than 255 (OFF H) (5)
the overflow flag is set, otherwise it is cleared.
→ The carry flag is always cleared.

(A) 7-0 ← (A) X (B)

(B) 15-8

Division

The DIV performs an unsigned division of register A by register B.

DIV AB (divide)

The DIV AB divides the unsigned 8-bit integer in the accumulator by the unsigned 8-bit integer in register B.

→ The accumulator receives the integer part of the quotient, register B receives the integer remainder.

(A) 15-8 ← (A)/(B)

(B) 7-0

Example

Register R0 contains 7EH (01111110 b)
The internal RAM location 7EH & 7FH contains FF H & 40 H respectively. The following instruction sequence is executed.

```
INC @R0
MOV A, @R0
INC R0
INC @R0
```

What will be the content of RAM locations 7EH, 7FH,

Scanned with CamScanner

Solution

Initially, R0 = 7EH, (A) = ? RAM (7EH) = FF H

RAM (7FH) = 40 H

After

1st Instruction -(R0) = 7EH, A = ( ) ?

RAM (7EH) = 60 H

RAM (7FH) = 40 H

2nd Instruction (R0) = 7EH, A = 00H, RAM (7EH) = 00H

RAM (7FH) = 40 H

3rd Instruction (R0) = 7FH, A = 00H, RAM (7EH) = 00H

RAM (7FH) = 40 H

4th Instruction (R0) = 7FH, A = 00H, RAM (7EH) = 00 H

RAM (7FH) = 41 H

Logi
DPK
→
do
→

Scanned with CamScanner

Logical Instruction

Logical instructions, which perform logical operation like AND, OR, XOR, NOT, Rotate, clear & swap.

→ Logical instructions are performed on byte of data on a bit-by bit basic.

→ Mnemonics associated with logical instructions are ANL, ORL, XRL, CLR, CPL, RL, RLC, RR, RRC, SWAP

ANL A, # data — A ← A AND Data (2B)

A, Rn — A ← A AND Rn (1B)

A, Direct — A ← A AND Direct (2B)

A, @Ri — A ← A AND @Ri (1B)

Direct, # data — Direct ← Direct AND # data. (3B)

ORL A, # data — A ← A OR Data.

A, Rn — A ← A OR Rn.

A, direct — A ← A OR (Direct)

A, @ Ri — A ← A OR @ Ri.

Direct, A — Direct ← Direct OR A.

Direct, # data — Direct ← Direct OR # data.

XRL A, # data — A ← A XRL Data

A, Rn — A ← A XRL Rn

A, Direct — A ← A XRL Direct

A, @ Ri — A ← A XRL ((Ri))

Direct A — Direct ← Direct XRL A

Direct, # data — Direct ← Direct XRL # data

Scanned with CamScanner

CLR A — A ← 00H

CPL A — A ← $\overline{A}$

RL A — Rotate Accumulator left

RLC A — Rotate Acc^r left through carry

RR A — Rotate Acc^r Right

RRC A — Rotate Acc^r Right through carry.

SWAP A — Swap Nibbles with Acc^r.

Boolean or Bit Manipulation Instructions

As the name suggest, Boolean or Bit Manipulation Instructions will deal with bit variables -

→ There is a special bit addressable area in the RAM & some SFR are also bit addressable.

→ The Mnemonics corresponding to the Boolean or Bit Manipulation instructions are

CLR, SETB, MOV, JC, JNC, JB, JNB, JBC, ANL, ORL, CPL.

→ CLR C — C ← 0 (C = carry bit) 1B

→ CLR Bit — Bit ← 0 (Bit = Direct Bit) 2B

→ SET C — C ← 1 (1B)

SET Bit — Bit ← 1 (2B)

→ CPL C — C ← $\overline{C}$ (1B)

→ CPL Bit — Bit ← $\overline{Bit}$ (2B)

Scanned with CamScanner

ANL C, /Bit — C ← C $\overline{Bit}$ (2B)

ANL C, Bit — C ← C. Bit (2B)

ORL C, /Bit — C ← C + $\overline{Bit}$ (2B)

ORL C, Bit — C ← C + Bit (2B)

MOV C, Bit — C ← Bit (2B)

MOV Bit C — Bit ← C (2B)

JC rel — Jump if carry (C) is set (2B)

JNC rel — Jump if carry (C) is not set (2B)

JB Bit, rel — Jump if Direct bit is set (3B)

JNB Bit, rel — Jump if direct bit is not set (3B)

JBC Bit, rel — Jump if direct bit is set & clear Bit (3B)

Program Branching Instructions

These instructions control the flow of program logic.

→ The Mnemonics of the program branching instructions are

LJMP, AJMP, SJMP, JZ, JNZ, CJNE, DJNZ

NOP, LCALL, ACALL, RET, RETI, JMP

→ All these instructions, except the NOP (No operation) affect the PC in one way or other.

→ Some of these instructions has decision making capability before transfering control to other part of program.

Scanned with CamScanner

ACALL ADDR11 — Absolute Subroutine Call (2B)

PC + 2 → (SP); ADDR 11 → PC

LCALL ADDR16 — Long Subroutine Call (3B)

PC + 3 → (SP); ADDR16 → PC

RET — Return from subroutine (1B)

(SP) → PC

RETI — Return from interrupt (1B)

AJMP ADDR11 — Absolute Jump (2B)

ADDR11 → PC

LJMP ADDR16 — Long Jump (3B)

ADDR16 → PC

SJMP rel — Short Jump (2B)

pc + 2 + rel → PC

JMP@A + DPTR — A + DPTR → PC (1B)

JZ rel — If A = 0, Jump to PC + rel (2B)

JNZ rel — If A ≠ 0, Jump to PC + rel (2B)

CJNE A, direct, rel — Compare (Direct) with A (3B)

A, #data, rel — Jump to PC + rel (3B)

Data with # not required.

Rn, #data, rel — Data with Rn (3B)

@Ri, #data, rel — Data with @ Ri (3B)

Scanned with CamScanner