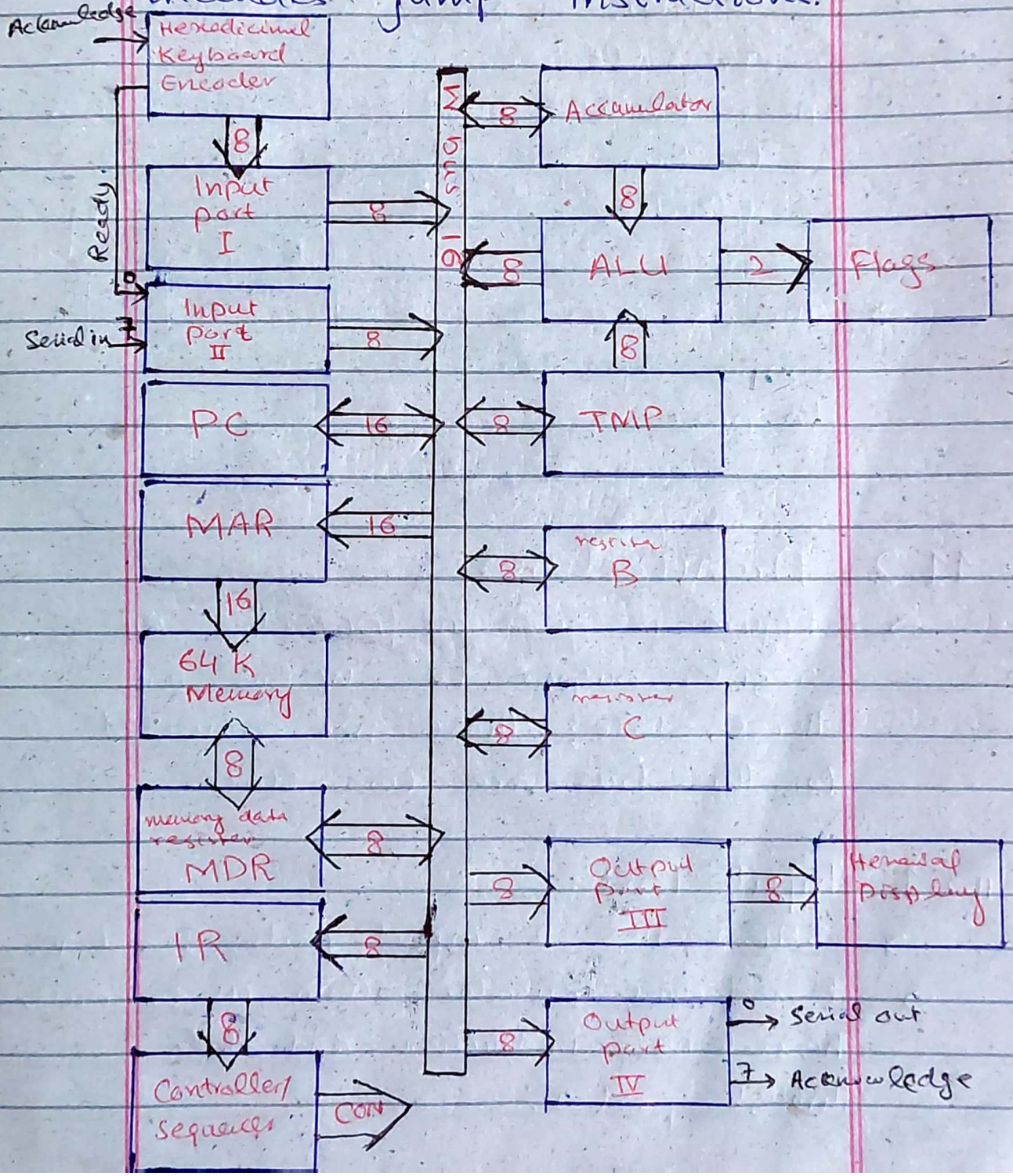
COA Notes We-17/01/2024

Chapter #11: SAP - 2

→ SAP-2 is the next step in evolution toward modern computers because it includes 'jump' instructions.

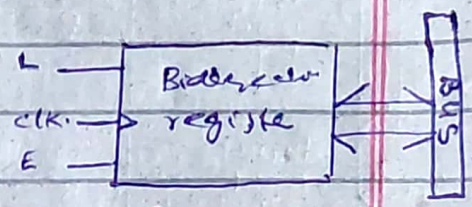


(2) * L = Load
E = Enable.
OP = output
IP = Input

11.1 Bidirectional Registers:-

→ During a computer run either LOAD or ENABLE may be active but not both at same time.

- Active LOAD: words flow from bus to register, so only the output lines are floating
- Active ENABLE: words flow from register to bus, so only input lines are floating.



→ This reduces the wiring capacitance and also the I/O pins.

11.2 Architecture:-

→ Figure is given in [P(#1)]. All register output to W-bus are three-state; those not connected to the bus are two-state.

11.2.1 Input Ports:

→ SAP-2 has 2 input ports; I and II.

→ Hexadecimal keyboard encoder allow us to enter hexadecimal instruction and data through port I.

→ The READY signal to bit 0 of port II from keyboard indicates when data in port-1 is ready/valid.

→ The SERIAL IN signal to bit 7 of port II is used to convert serial data to parallel data.

11.2.2 Program Counter (PC)

→ A 16-bit counter. It can count from 0000 H to FFFF H (or) 0 to 65,535.

OP: 0000 H to FFFF H

IP:

$-\overline{CLK}$, $-\overline{CLR}$, $-C_P$, $-E_P$

$-\overline{CLK}$ ← clk cycles

$-\overline{CLR}$ ↓ , reset to 0000 H

$-C_P$ ↓ PC++

$-E_P$ → PC → bus W.

11.2.3 MAR and Memory:

→ MAR Receive 16-bit address from PC during fetch cycle.

→ The two-state MAR then addresses ~~outputs~~ the desired memory location.

IP: - 16-bit address from PC

- CLK , $-\overline{L_m}$: -0 ⇒ MAR ← bus W

OP: address

→ Memory has address from 0000 H to FFFF H.

→ The memory has a ROM (2K: 0H to 7FF H) which contain a program called a

"monitor", that initializes the computer of power up, interpret key-board and so forth.

→ The rest of memory is a RAM (62 Is 800 H to FFFF H).

IP:

OP:

11.2.4 Memory Data Register (MDR):

→ It is an 8-bit buffer register.

→ It receive data from bus before a write operation, and it sends data to bus after a read operation.

11.2.5 Instruction Register (IR):

→ Although SAP-2 has only 42 instruction but for upward compatibility we will use an 8-bit op-code which can accomodate 256-instructions.

11.2.6 Controller/Sequencer (C/S):

→ As usual controll all execution flow.

→ Produce controll words (micro-instruction)

→ Has more hardware (large no. of instructions)

→ The produced CON (control word) is bigger which determine how the registers react to the next positive clock edge.

11.2.7 Accumulator (Acc):-

→ The 2-state output goes to ALU and 3-state to W-bus.

→ Same As SAP-1 p## 5

11.2.8 ALU and Flags.

→ The ALU performe both arithemetic and logic operations.

→ The SAP-2 has 2 flags which keep track changing condition in flip-flop.

- The sign-flag (S) is set when the accumulator contents become -ve.
- The zero-flag (Z) is set when the accumulator contents becone zero.

11.2.9 TMP, B, C registers.

→ B, instead of TEMP hold the data being added or subtracted from acc.

→ Register B and C are used to move data during program run and accessible to programmers.

11.2.10 Output PORTs:

→ SAP-2 has two output ports; III & IV.

→ Content of accumulator is loaded to port III, which drives a hexadecimal display to see the processed data.

→ SERIAL OUT from pin 0 of port IV is used to convert parallel data from acc into series data.

→ PIN-7 of port IV sends ACKNOWLEDGE to hexadicimal encoder.

→ The ACKNOWLEDGE and READY is part of a concept called handshaking.

11.3 Memory-Reference Instructions (MRI):

→ SAP-2 fetch cycle as same as befor T_1 is the address state, T_3 is the memory state and T_2 is the increment state.

→ An MRI is one that uses the memory during the execution cycle.

11.3.1 LDA and STA:

→ LDA has same meaning; "Load the accumulator" with the address memory dat. eg. LDA 2000 H
(LDA: opcode; 2000 H: operand)

→ STA is a mnemonic for "store the accumulator" to the given address. eg STA 8AH.
(STA: opcode, 8AH: operand)

→ Instruction has two parts:
- opcode /mnemonics
- operand

→ Opcode has double meaning in microprocessor work; it may stands for mnemonics or for binary code used to represent the mnemonics.

11.3.2 MVI

→ The mnemonic for "move imidiate".

→ Tells the computer to Load a designated register with the byte the imidiatly follows the instruction.

eg: MVI A, 37H

This will load acc with

A = 0011 0111

11.4. Register Instructions:

→ Register Instructions are those that are designed to move data from one register to another in the shortest possible time.

11.4.1 MOV

→ MOV is mnemonic for "move". It tell the computer to move data from one to another register

eg MOV A, B [A ← B]

→ This operation is non-destructive, meaning that the data in B is copied but not erased.

→ These instruction are fastest in SAP-1 requiring only one machine cycle.

→ We can move data b/w A, B and C. which can be arranged in 6 different combinations.

11.4.2 ADD and SUB.

→ ADD stands for add the data in designated register to the content of accumulator. i.e ADD B. If A = 04H and B = 02H the A = 06H. after ADD B.

→ SUB means subtract data in designated register from accumulator. i.e SUB C.

→ formats of ADD and SUB are
ADD <B|C> , SUB <B|C>.

11.4.3 INR and DCR:

→ INR is mnemonic for "increment". It tells the computer to increment the designated ~~computer~~ register. i.e INR <A|B|C>.

→ DCR is mnemonic for "decrement" which instruct the computer to decrement the designated register. i.e
DCR <A|B|C>.

11.5 Jump and Call instructions:

→ These instruction can chang program sequence by jumping/braneling to another part of program instead of fetching the next instruction in the usual way.

11.5.1 JMP.

→ JMP is mnemonic for "jump" which tells the computer to get the next instruction from designated memory location.

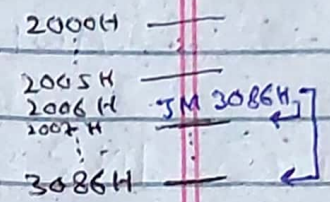
→ Every JMP instruction includes an

(10) * sign-flag S = { 0 ; A ≥ 0.
1 ; A < 0.

* zero-flag Z = { 0 ; A ≠ 0.
1 ; A = 0.

address that is loaded into PC.

JMP 308B H tells the computer to get next instruction from 308B H.



11.5.2 JM:

→ Mnemonic for "jump if minus"; the computer will jump to a designated address if and only if sign-flag (S) is set (i.e 1).

JM 3000H.

11.5.3 JZ

→ Mnemonic for "jump if ~~not~~ zero".

→ Instruct the computer to the designated address only if zero-flag (Z) is set (i.e 1).

11.5.4 JNZ

→ Mnemonic for "jump if not zero".

→ we get a jump when the zero flag is clear and no jump when it is set.

→ JM, JZ and JNZ are called conditional jumps because the

program jump occurs only if certain conditions are satisfied.

→ JMP is unconditionally; the jump always occurs once the instruction is fetched.

11.5.5 CALL and RET.

→ A subroutine is a program stored in memory for possible use in another program.

→ CALL is mnemonic for "call the subroutine". Every call instruction must include the starting address of the desired subroutine.

i.e CALL 5083H

→ RET stands for "return". It is used at the end of every subroutine to tell the computer to go back to the origional program.

→ RET is to subroutine as HLT is to a program. Both tells the computer that something is finished.

→ CALL is unconditionall.

11.5.6 More on flags:

→ If the accumulator (A) goes -ve or zero while one of below instruction is executed the sign or zero flag is set.

→ sign-flag $S = \begin{cases} 0 ; & A \geq 0. \\ 1 ; & A < 0. \end{cases}$

→ zero-flag $Z = \begin{cases} 0 ; & A \neq 0. \\ 1 ; & A = 0. \end{cases}$

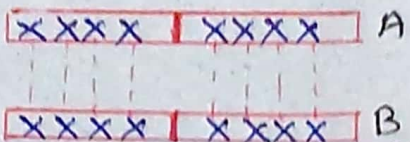
→ Instruction affecting FLAGS:

Instruction	Flag Affected	
	S	Z
Add	✓	✓
SUB	✓	✓
INR	✓	✓
DCR	✓	✓
ANA	✓	✓
ORA	✓	✓
XRA	✓	✓
ANI	✓	✓
ORI	✓	✓
XRI	✓	✓

11.6 Logical instructions:

11.6.1 CMA

→ stands for "complement the



accumulator". This inverts each bit in accumulator producing 1's complement.

11.6.2 ANA

→ means to AND the accumulator content designated register.

eg; ANA <B|C> ∴ A&B

→ Anding is bitwise

11.6.3 ORA

→ Mnemonic for OR the accumulator with the designated register.

eg: ORA <B|C> ∴ A|B

11.6.4 XRA

→ XRA means XOR the accumulator content with the designated register.

i.e XRA <B|C> ∴ A^C

11.6.5 ANI

→ means, "AND imidiate". tell the computer to AND the accumulator content with the byte that imidiatly follows the opcode

i.e ANI C7H.

11.6.6 ORI

→ mnemonic for OR inidiate.

→ Accumulator contents are ored with the byte that follows the opcode ie ORI 5AH.

11.6.7 XRI

→ XOR inidiate.

→ Accumulator contents are XORed with the byte that follows the op-code. ie. XRI D4H.

11.7 Other Instructions:-

11.7.1 NOP:

→ Stands for "no-operation".

→ During its execution, all T states are doing nothing, hence no register changes occur during a NOP

→ It is used to waste time. It takes four T states to fetch and execute this instruction.

11.7.2 HLT

→ Stands for hault.

→ ends data processing.

11.7.3 IN.

→ mnemonic for "input."

→ Tells the computer to transfer data from designated port to the accumulator.

i.e IN <byte>

eg IN 02H : A ← port-II.

11.7.4 OUT

→ mnemonic for "Output"

→ Use to load a port from content of accumulator.

i.e OUT <byte>.

eg OUT 03H : port-III ← A.

11.7.5 RAL

1 → mnemonic for rotate "the accumulator Left".

→ This shift all the bits to left and move MSB to the LSB position.

eg A= 1011 0100

RAL // 0110 1001

11.7.6 RAR

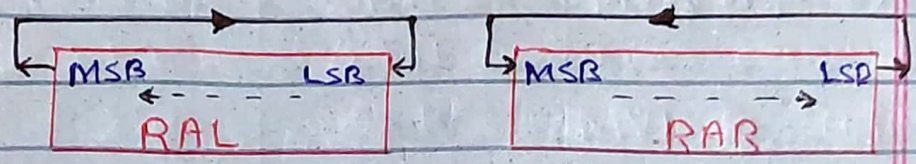
→ rotate the accumulator right.

→ This time the bits are shifted to the right, the LSB going to

the MSB position.

i-e A = 1011 0100

RAR ⇒ A = 0101 1010



11.8 SAP-2 Summary:

→ Some instructions take longer than others to execute.

→ Not all instructions affect the flag.

→ Direct addressing is where we specify the address where the data is to be found.
eg. LDA <address>.

→ Imidiate addressing; instead of giving an address for data, we give the data itself.
eg. MVI A, <byte>.

→ Register addressing; the data is to be loaded from a register in CPU rather than in memory.

i.e MOV. A, B.
It has advantage of speed.

→ Implied addressing; means that the location of the data is contained within the opcode itself.
eg RAL.

→ Each instruction occupies a number of bytes in memory. SAP-2 instruction are either 1, 2 or 3 bytes long.
eg ADD, ANI, CALL respectively.

→ Total byte length of subroutine is 8 bit.

TABLE 11-3. SAP-2 INSTRUCTION SET

Instruction	Op Code	T States	Flags	Addressing	Bytes
ADD B	80	4	S, Z	Register	1
ADD C	81	4	S, Z	Register	1
ANA B	A0	4	S, Z	Register	1
ANA C	A1	4	S, Z	Register	1
ANI byte	E6	7	S, Z	Immediate	2
CALL address	CD	18	None	Immediate	3
CMA	2F	4	None	Implied	1
DCR A	3D	4	S, Z	Register	1
DCR B	05	4	S, Z	Register	1
DCR C	0D	4	S, Z	Register	1
HLT	76	5	None	—	1
IN byte	DB	10	None	Direct	2
INR A	3C	4	S, Z	Register	1
INR B	04	4	S, Z	Register	1
INR C	0C	4	S, Z	Register	1
JM address	FA	10/7	None	Immediate	3
JMP address	C3	10	None	Immediate	3
JNZ address	C2	10/7	None	Immediate	3
JZ address	CA	10/7	None	Immediate	3
LDA address	3A	13	None	Direct	3
MOV A,B	78	4	None	Register	1
MOV A,C	79	4	None	Register	1
MOV B,A	47	4	None	Register	1
MOV B,C	41	4	None	Register	1
MOV C,A	4F	4	None	Register	1
MOV C,B	48	4	None	Register	1
MVI A,byte	3E	7	None	Immediate	2
MVI B,byte	06	7	None	Immediate	2
MVI C,byte	0E	7	None	Immediate	2
NOP	00	4	None	—	1
ORA B	B0	4	S, Z	Register	1
ORA C	B1	4	S, Z	Register	1
ORI byte	F6	7	S, Z	Immediate	2
OUT byte	D3	10	None	Direct	2
RAL	17	4	None	Implied	1
RAR	1F	4	None	Implied	1
RET	C9	10	None	Implied	1
STA address	32	13	None	Direct	3
SUB B	90	4	S, Z	Register	1
SUB C	91	4	S, Z	Register	1
XRA B	A8	4	S, Z	Register	1
XRA C	A9	4	S, Z	Register	1
XRI byte	EE	7	S, Z	Immediate	2

r.e MOV. A, B.
It has advantage of speed.

→ Implied addressing; means that the location of the data is contained within the opcode itself.
eg RAL.

→ Each instruction occupies a number of bytes in memory. SAP-2 instruction are either 1, 2 or 3 bytes long.
eg ADD, ANI, CALL respectively.

→ Total byte length of subroutine is 8 bit.

Examples:
Ex # 11.1:0
Solution

```
MVI A, 49H      ; A ← 49H
MVI B, 4AH      ; B ← 4AH
MVI C, 4BH      ; C ← 4BH
STA 6285H       ; 6285H ← A
HLT             ; Stop data processing.
```

18)

Ex # 11.2:

Solution

Address (Hex)	Contents (hex)	Symbolic
2000 H	3EH	MVI A, 49H
2001 H	49H	; MVI is 2 bit
2002 H	06H	MVI B, 4AH
2003 H	4AH	
2004 H	0EH	MVI C, 4BH
2005 H	4BH	; MVI is 2 bit
2006 H	32H	STA 6285H
2007 H	85H	
2008 H	62H	; STA is 3 bit instruction
2009 H	76H	HLT

→ Why does address get programmed with the lower byte first and the upper byte second? To keep upward compatibility. The 8085 and some other microprocesso keep the same scheme; lower byte into lower and upper byte into upper memory.

Ex # 11.3:- ans ← C15 + 23

5600H ← ans , C ← ans + 1

Solution:

MVI A, 17H ; A ← 23

```
MVI  B, 2DH      ; B ← 24
ADD  B           ; A ← A+B
STA  5600H       ; 5600H ← A
INR  A           ; A++
MOV  C, A        ; C ← A
HLT.
```

Ex# 11.4 ; hand-assemble the above --
Solution.

Address (hex)	Contents (Hex)	Symbolic
2000 H	3E H	MVI A, 17H
2001 "	17 "	
2002 "	06 "	MVI B, 2DH
2003 "	2D "	
2004 "	80 "	ADD B
" 5 "	32 "	STA 5600H
" 6 "	00 "	
" 7 "	56 "	
" 8 "	3C "	INR A
" 9 "	4F "	MOV C, A
" A "	76 "	HLT

Ex # 11.5: Hand assemble the following program

```
MVI, C, 03H
DCR  C
JZR  0009 H
JMP  0002 H
```

Address (Hex)	Content (Hex)	Symbolic
2000 H	0E H	MVI C, 03H
" 1 "	03 H	
" 2 "	0D H	DCR C
" 3 "	CA H	JZ 2009 H
" 4 "	09 H	
" 5 "	20 H	
" 6 "	C3 H	JMP 2002 H
" 7 "	02 H	
" 8 "	20 H	
" 9 "	76 H	HLT

→ The C register act like a presettable down~~timer~~ counter. This is why it is sometimes referred to as counter.

→ There is a loop which itrates 3 times.

→ Ex # 11.8 : multiply 12 x 8.

Solution:

```
        MVI A, 00H      ; Clear accumulator
        MVI B, 0CH      ; B ← 12
        MVI C, 08H      ; C ← 8
REPEAT: ADD B           ; A ← A+B
        DEC C           ; C --
        JZ DONE         ; test for zero.
```

```
        JMP  REPEAT   ; Do it again
DONE    HLT           ; Stop processing.
```

→ Ex # 11.13: The bit in a ....

Solution

```
        IN   02H      ; get byte from port 2
        ANI  01H      ; A ← A & 00...1
                      ;   to isolate bit 0.
        JNZ  YES      ; jump to yes if bit 0 is 1
        MVI  A, 4EH   ; A ← 4EH (=N in ASCII)
        JMP  DONE     ; jump to done
YES:    MVI  A, 59H   ; A ← 59H (=Y in ascii)
DONE:   OUT  03H      ; port III ← A
        HLT           ; Stop processing.
```

→ The immediate byte in ANI 01H (0000 0001) is called a "mask" because its 0s will mask or blank out the corresponding high bits in accumulator.

If $A = A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$

then ANI 01H will produce

$A = 0000\ 000A_0$

Ex # 11.15: Handshaking is a ....

Solution:

```
STATUS:
        IN   02H      ; A ← port-II
        ANI  01H      ; A ← A&01H
                      ; check if ready bit
```

```
JMP STATUS      ; jump back if not ready
IN 01H          ; transfer data in port-I
MOV B, A        ; B ← A
MVI A, 80H      ; Set acknowledge bit.
OUT 04H         ; Output high acknowledge
MVI A, 00H      ; Reset acknowledge bit
OUT 04H         ; Output low acknowledge
HLT             ; Stop.
```

→ handshaking is an interaction b/w a CPU and a peripheral device that takes place during I/O data transfer.

→ The sequence of SAP-2 handshaking is:

- READY bit (bit 0, port II) goes high.
- Input data in port I to the CPU
- ACKNOWLEDGE bit (bit 7, port IV) goes high to reset READY bit.
- Reset the ACKNOWLEDGE bit.

→ Ex # 11.16: ~~How~~ SAP-2 has a ....

Solution

MVI: 7 x 1 x 1μs = 7μs

DCR: 4 x 70 x 1μs = 280 μs

(jump) JNZ: 10 x 69 x 1μs = 690 μs

(no jump) JNZ: $7 \times 1 \times 1\mu s = 7\ \mu s$

NOP: $4 \times 1 \times 1\mu s = 4\ \mu s$

RET: $10 \times 1 \times 1\mu s = 10\ \mu s$

$= 998\ \mu s$

$\approx 1\ ms.$

Now the byte length is

Bytes = MVI + DCR + JNZ + NOP + RET

$= 2 + 1 + 3 + 1 + 1$

$= 8$