SAP-2

Introduction

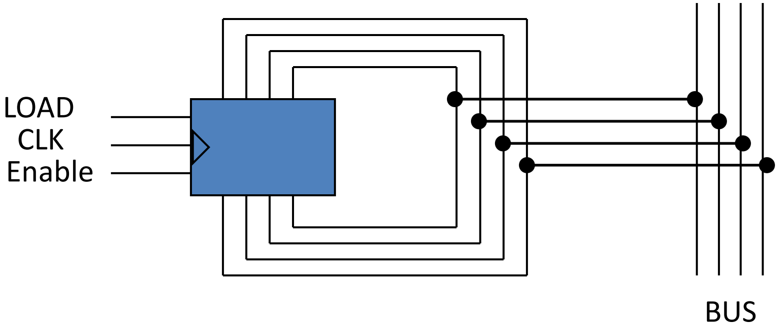
- SAP-2 is the next step in the evaluation toward modern computers.
- It includes **jump** instructions.

Bidirectional Registers

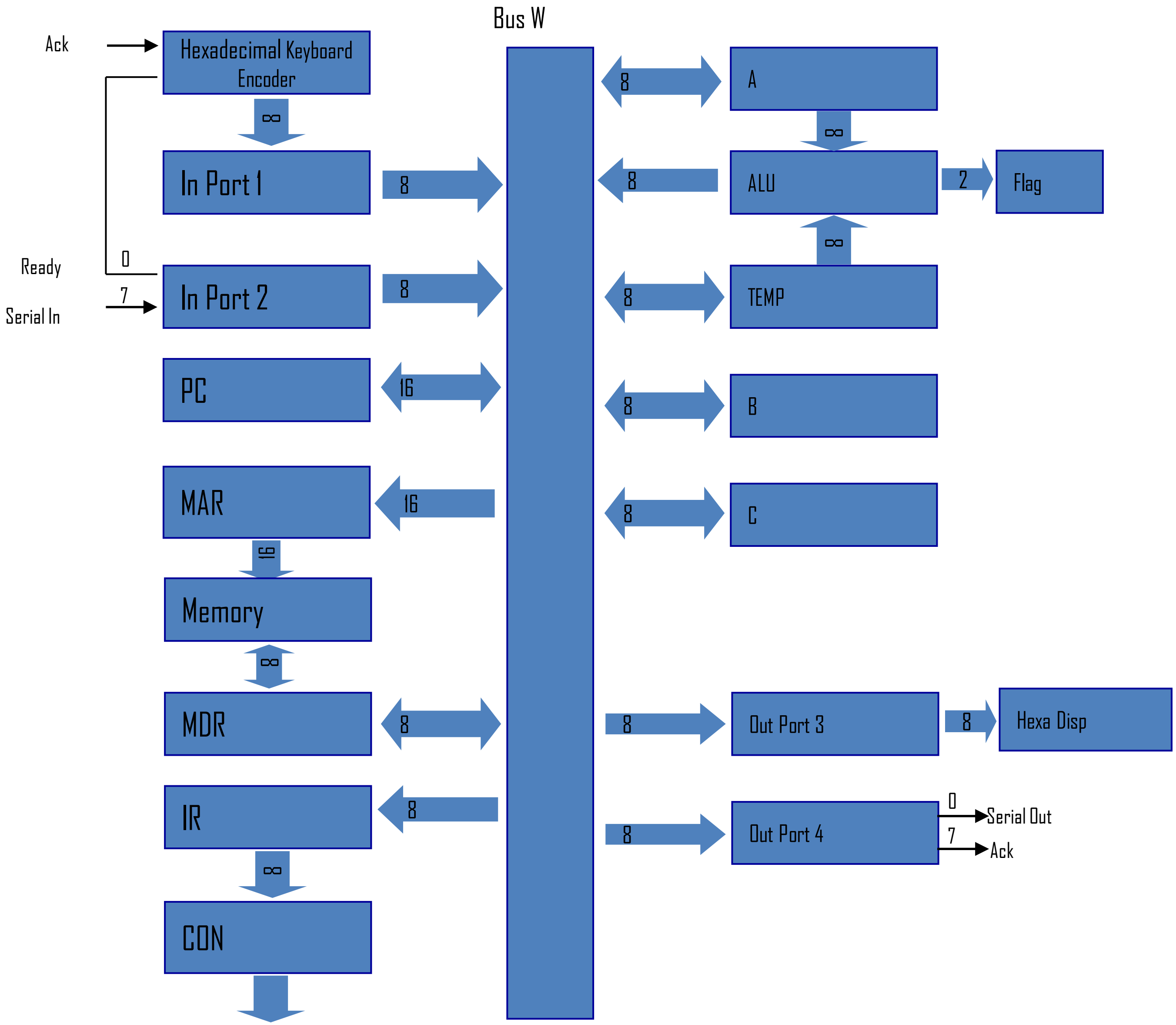
- Either enable or load only active .
- Active LOAD means a binary word flows from bus to register
- Active ENABLE means a binary word flows from register to bus

- Input and output pins are shorted.
- Single set of wires(path) between register and w-bus.

Bidirectional Registers



Architecture



Input ports

Port 1 and Port 2

Port 1

- Hexadecimal keyboard encoder connected to port 1
- Allows to enter hexadecimal instructions and data
- Hexadecimal Keyboard Encoder sends READY signal to bit 0 of port 2 (indicates the data in port 1 is valid)

Port 2

- SERIAL IN signal going to pin 7 of port 2

Program Counter

16 bit address

Thus can count from

PC= 0000 0000 0000 0000

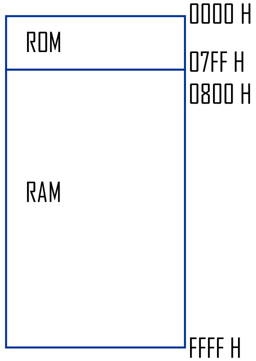
PC= 1111 1111 1111 1111(FFFFH)

Equivalent to 0000H to FFFFH, or decimal 0 to 65,535

LOW CLR’ resets the PC before each run to start with instruction of 0000H

MAR and MEMORY

16- bit address from PC to MAR (From 0000H → FFFFH)



- MAR OUTPUT to RAM
- Memory Capacity(?????)
- 2K ROM(0000H-07FFH) => Monitor Program
- 62K RAM(0800H-FFFFH)

Monitor program: initializes the computer on power-up, interprets the keyboard inputs, and so forth.

Memory Data Register

- 8-bit Register
- Output sets up the RAM
- Receives data from the bus before write operation
- Sends data to the bus after read operation

Instruction Register

- 8-bit op code
- Can accommodate 256 instruction
- Only 42 instruction
- Using 8-bit op code also allows upward compatibility with 8080/8085 instruction set (as they are based on an 8-bit op code)
- All SAP instructions identical with 8080/8085 instructions

Controller Sequencer

As usual

- Generates the control words (microinstructions)
- Coordinate and direct rest of the computer
- Has more hardware(larger number of instruction)
- Control Word is bigger (CON)

Accumulator

Same as SAP-1

ALU and Flags

- ALU: Includes both arithmetic and logical operation
- 4 or more control bits for determining the operation to be performed

- Flag: Represent the status of the arithmetic and logical operation
- Filp flops are used;
    - Zero Flag(Z)
    - Sign Flag(S)

Temp, B and C registers

- **Temporary register (TEMP)**
- Register B and C are used to move data during program run and accessible to programmers.

Output Ports

2 output ports(3 and 4)

- Port 3 : a) Contents of the accumulator loaded to port 3, which drives the Hexadecimal display.

  b) Allows to see processed data.

- Port 4: a) Contents of the accumulator can be sent

  b) Pin 7 of port 4 sends an ACKNOWLEDGE signal to hexadecimal encoder. (Handshaking)

Serial Out (Pin 0 of Port 4): Serial Transmission of data.

Memory Reference Instructions

- Fetch cycle is the same as SAP-1 (T_1, T_2, T_3)
- Uses memory during the fetch cycle
- **During execution cycle, memory may or may not be used**
- Depends on the type of the instruction fetched
- **Relatively slow** as requires more than one memory access during instruction cycle

LDA and STA

- **LDA 2000H:**

  Load the accumulator with the contents of memory location 2000H

- **STA 8000H:**

  Store the accumulator contents at memory location 8000H

MVI

MVI-Move Immediate

MVI A,37H ➔ A = 0011 0111 (load the accumulator with 37H)

MVI A, byte

MVI B, byte

MVI C, byte

Register Instructions

- For moving data directly from one register to another without accessing the memory

MOV

MOV A, B (data of B is copied, not erased)
Ex: A → 34H, B → 9DH,
MOV A, B: A→ 9DH, B→9DH

MOV A, C
MOV B, A
MOV B, C
MOV C, A
MOV C, B

Register Instruction

ADD and SUB

Eg ADD B/SUB B

ADD B
ADD C
SUB B
SUB C

Register Instruction

INR and DCR

INR A/DCR A

INR B/DCR B

INR C/DCR C

Jump And Call Instruction

JMP: (a) tells the computer to get the next instruction from the designated memory location

(b) address is loaded into the program counter

JMP 3000H

JM (Jump if Minus)
JZ(Jump if zero)
JNZ(Jump if not zero)

Jump And Call Instruction

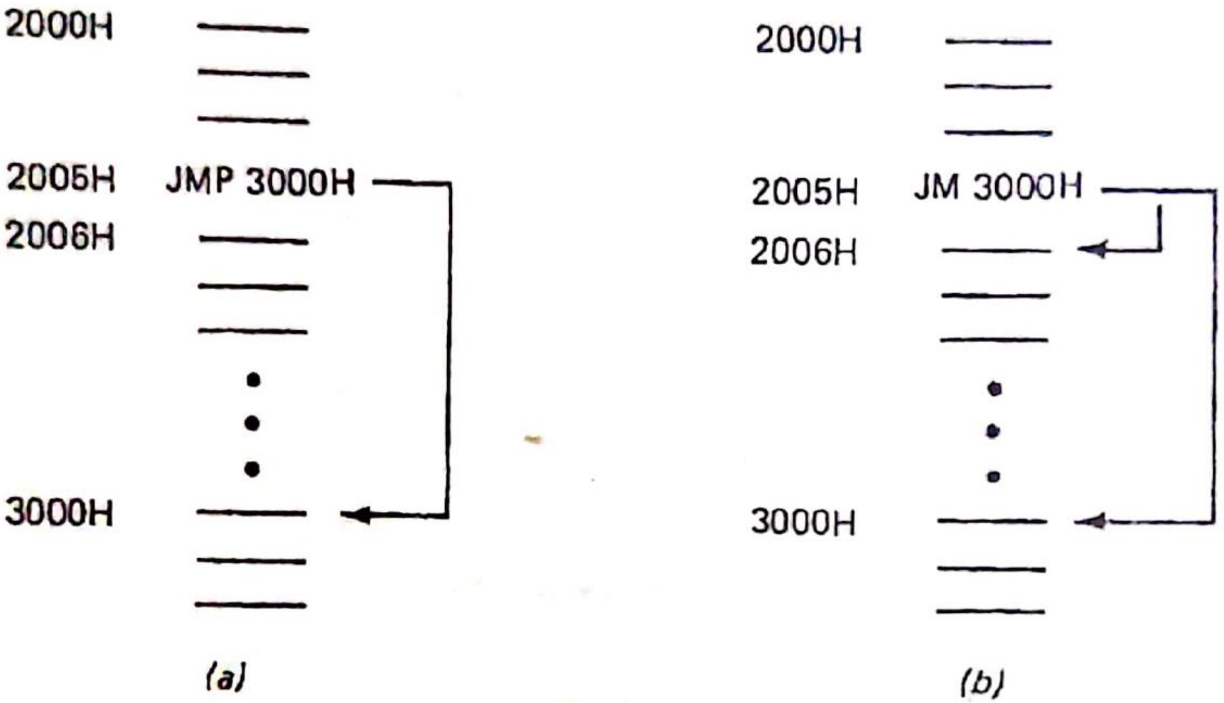


Fig. 11-3 (*a*) Unconditional jump; (*b*) conditional jump.

Jump And Call Instruction

CALL

Subroutine ????

Call is used to call the subroutine

Ret

Return back from subroutine

Program Counter contents ????

-----stored in the last two location of memory (FFFEH and FFFFH)

Logic Instruction

CMA - Complement the accumulator

ANA - **And the accumulator** with specified register

(two available: ANA B, ANA C)

ORA- **OR the accumulator** with specified register

(two available: ORA B, ORA C)

XRA- **XOR the accumulator** with specified register

(two available: XRA B, XRA C)

Logic Instruction Contd.

ANI: And Immediate

E.g. ANI C7H (AND accumulator with immediate data C7H)

ORI: OR immediate

E.g. ORI C7H

XRI: XOR immediate

E.g. XRI C7H

Other Instructions

- **OUT (OUT byte e.g. OUT 03H→** accumulator content to designated port, output ports are numbered 3 and 4**)**

- **HLT** (ends the data processing)

- **IN (Input):** transfers the data from designated input port to accumulator, **e.g. IN 02H**

- **NOP (No Operation):** during execution, all T states do nothing; used to waste time or to delay the data processing; takes four T states to fetch and execute NOP; useful in timing operations. If we put NOP inside a loop and execute it 100 times, a delay of 400 T states is created.

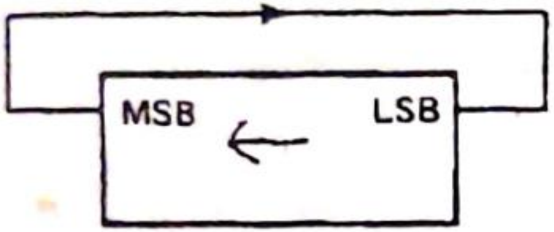
Other Instructions (Contd.)

- **RAL***(Rotate the accumulator left)*

  A=1011 0100

  After execution→

  A=0110 1001

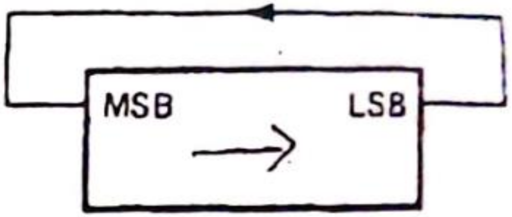


- **RAR** *(Rotate the accumulator right)*

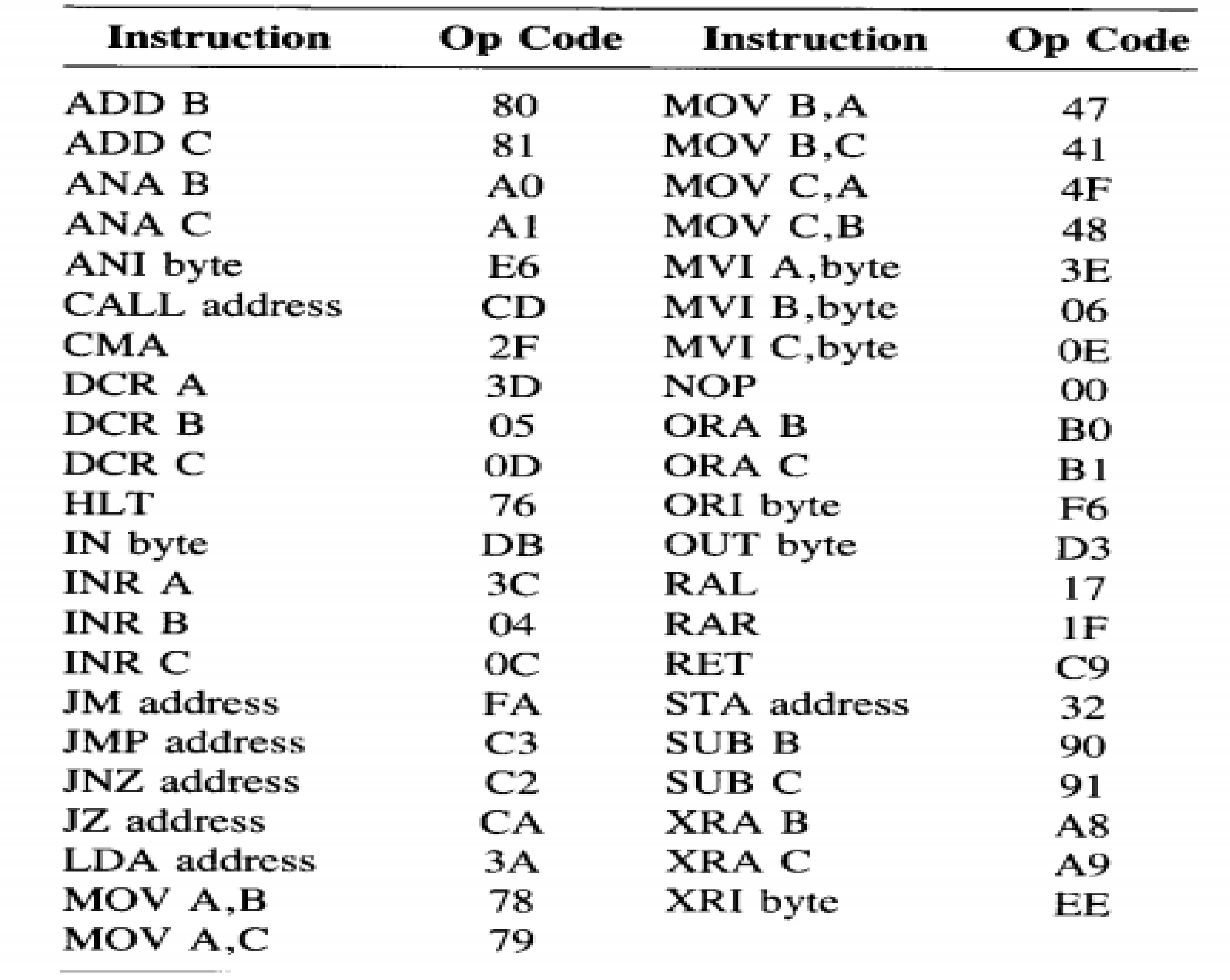
  A= 1011 0100

  After execution→

  A=0101 1010

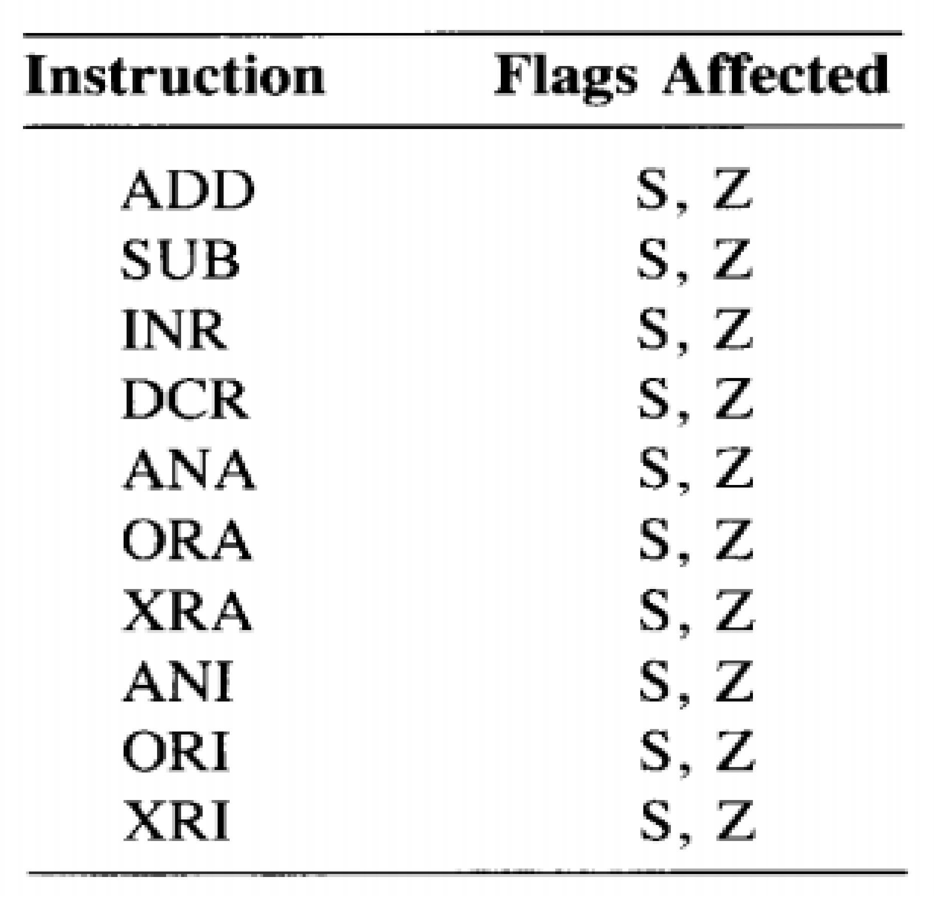


SAP-2 Op codes



Instruction	Op Code	Instruction	Op Code
ADD B	80	MOV B,A	47
ADD C	81	MOV B,C	41
ANA B	A0	MOV C,A	4F
ANA C	A1	MOV C,B	48
ANI byte	E6	MVI A,byte	3E
CALL address	CD	MVI B,byte	06
CMA	2F	MVI C,byte	0E
DCR A	3D	NOP	00
DCR B	05	ORA B	B0
DCR C	0D	ORA C	B1
HLT	76	ORI byte	F6
IN byte	DB	OUT byte	D3
INR A	3C	RAL	17
INR B	04	RAR	1F
INR C	0C	RET	C9
JM address	FA	STA address	32
JMP address	C3	SUB B	90
JNZ address	C2	SUB C	91
JZ address	CA	XRA B	A8
LDA address	3A	XRA C	A9
MOV A,B	78	XRI byte	EE
MOV A,C	79		

Instruction Affecting Flags

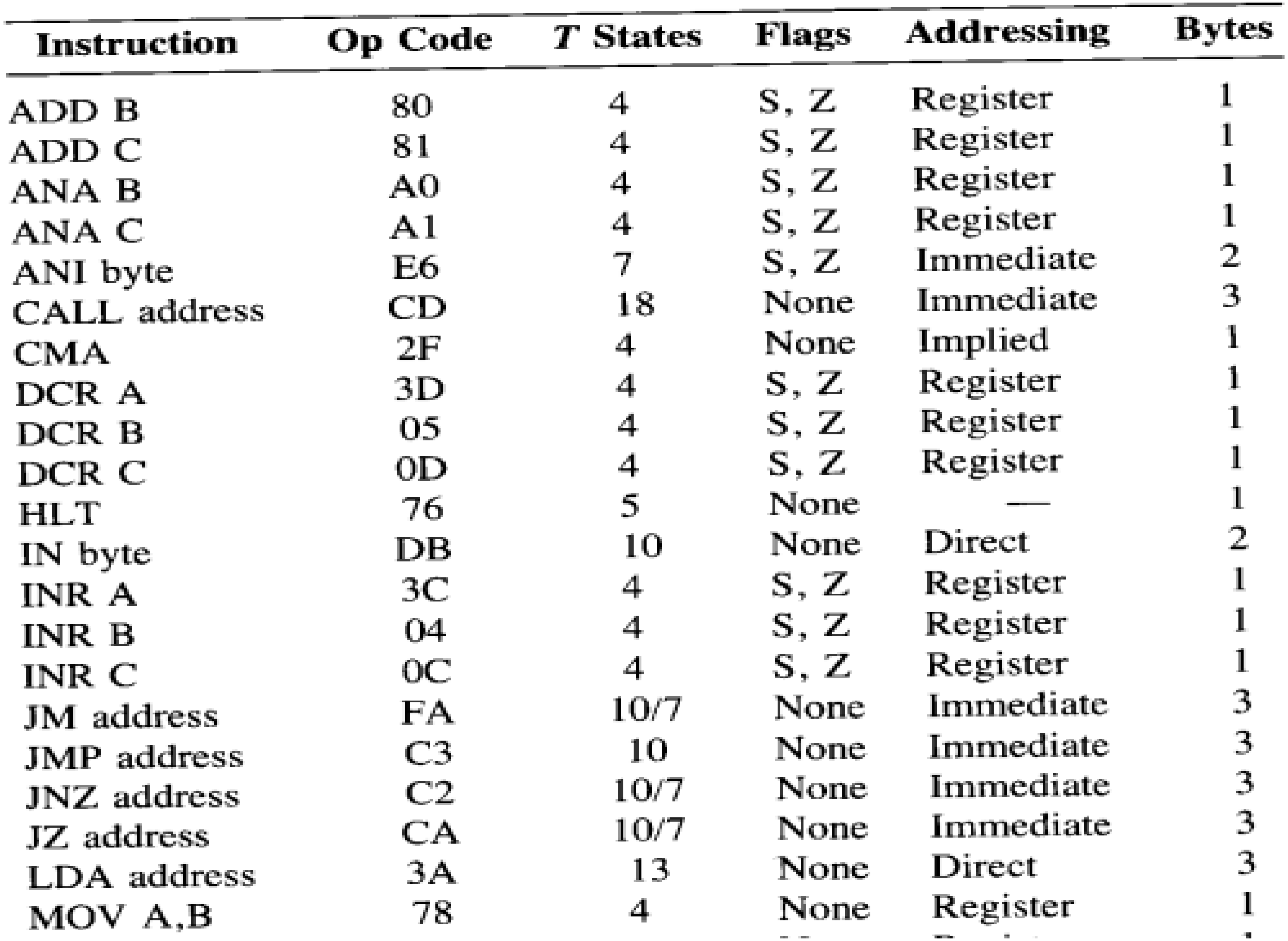


Instruction	Flags Affected
ADD	S, Z
SUB	S, Z
INR	S, Z
DCR	S, Z
ANA	S, Z
ORA	S, Z
XRA	S, Z
ANI	S, Z
ORI	S, Z
XRI	S, Z

T-States

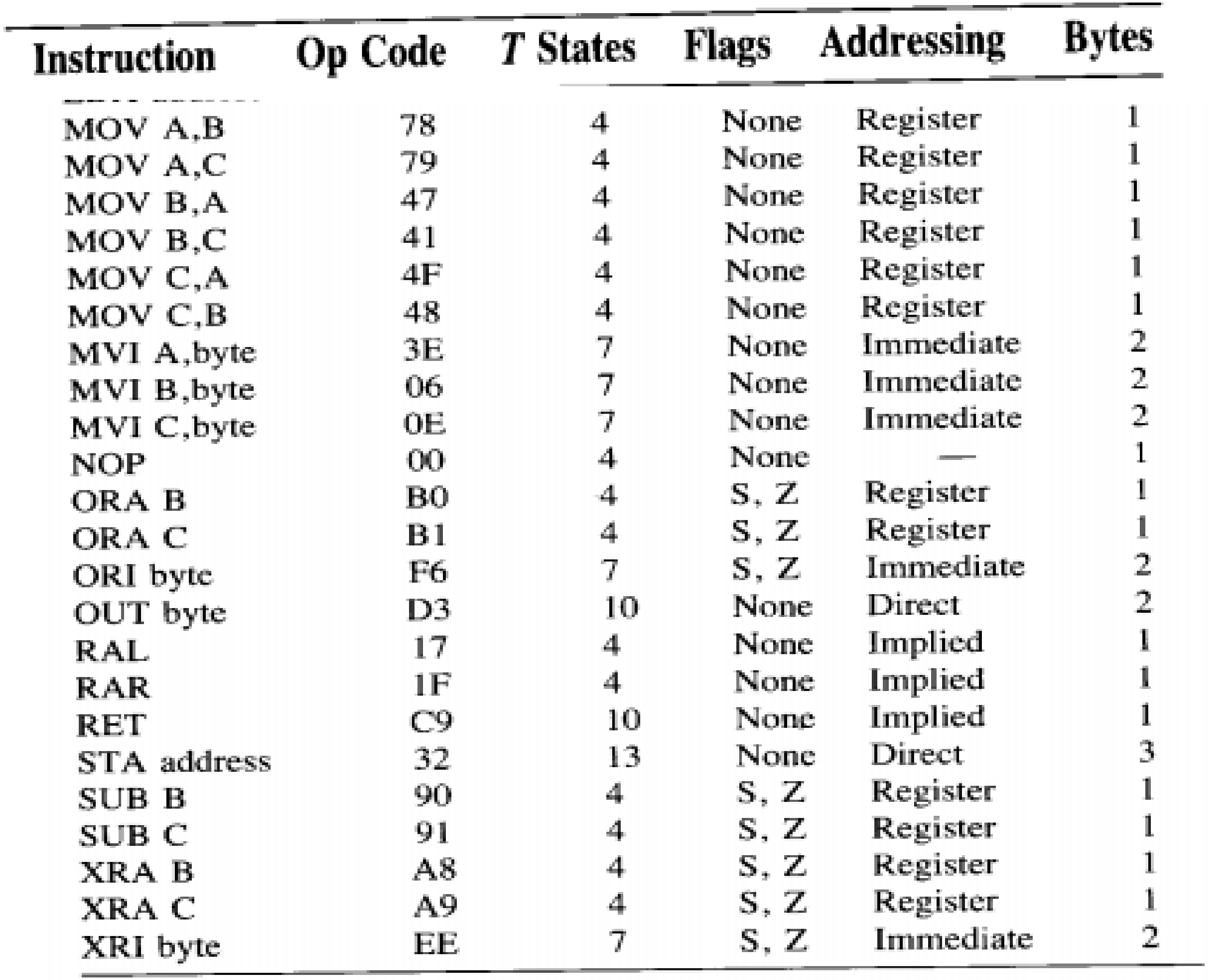
- Fetch → 3 T-States
- Execution → Different instruction requires different # of T-States
- Ex: ADD B → 4
- ANI byte → 7
- CALL → 18
- JM → 10/7

Summary 1



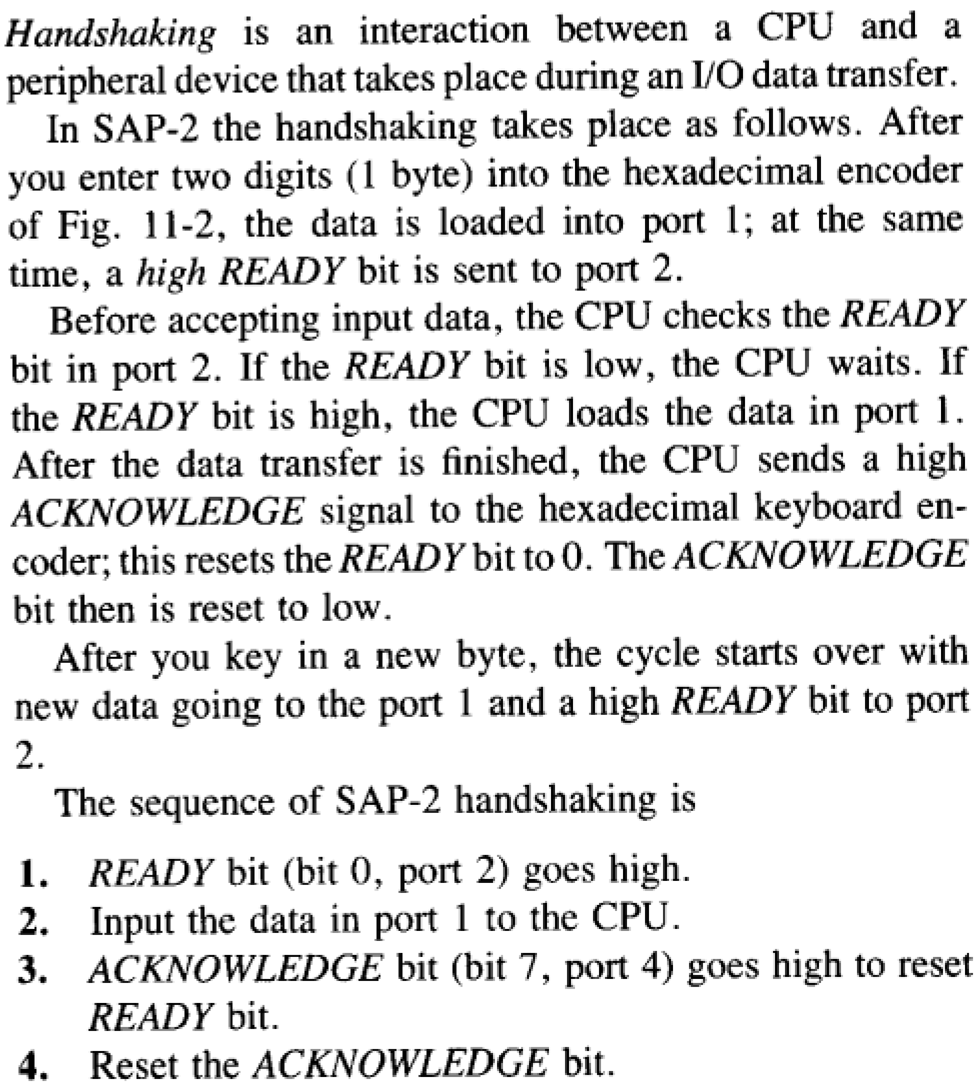
Instruction	Op Code	*T* States	Flags	Addressing	Bytes
ADD B	80	4	S, Z	Register	1
ADD C	81	4	S, Z	Register	1
ANA B	A0	4	S, Z	Register	1
ANA C	A1	4	S, Z	Register	1
ANI byte	E6	7	S, Z	Immediate	2
CALL address	CD	18	None	Immediate	3
CMA	2F	4	None	Implied	1
DCR A	3D	4	S, Z	Register	1
DCR B	05	4	S, Z	Register	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

Summary 2



Instruction	Op Code	*T* States	Flags	Addressing	Bytes
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

Handshaking



Handshaking is an interaction between a CPU and a peripheral device that takes place during an I/O data transfer.

In SAP-2 the handshaking takes place as follows. After you enter two digits (1 byte) into the hexadecimal encoder of Fig. 11-2, the data is loaded into port 1; at the same time, a *high READY* bit is sent to port 2.

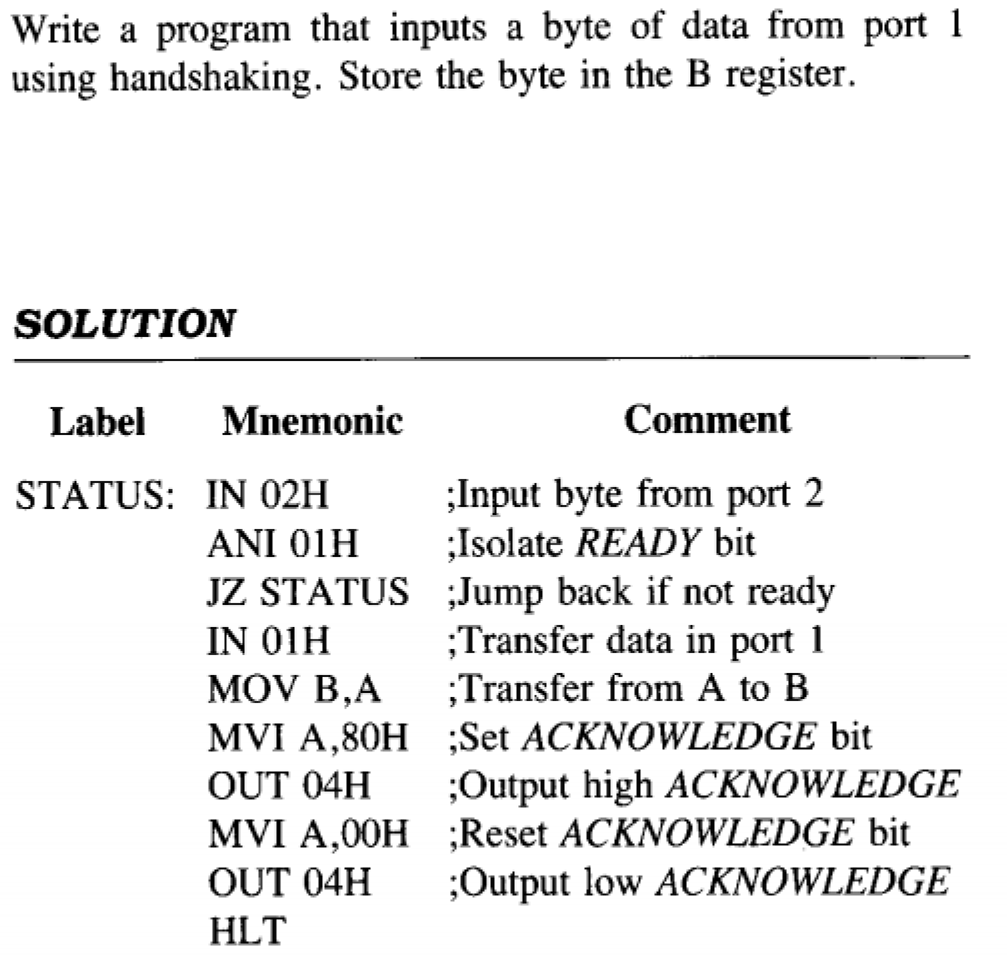
Before accepting input data, the CPU checks the *READY* bit in port 2. If the *READY* bit is low, the CPU waits. If the *READY* bit is high, the CPU loads the data in port 1. After the data transfer is finished, the CPU sends a high *ACKNOWLEDGE* signal to the hexadecimal keyboard encoder; this resets the *READY* bit to 0. The *ACKNOWLEDGE* bit then is reset to low.

After you key in a new byte, the cycle starts over with new data going to the port 1 and a high *READY* bit to port 2.

The sequence of SAP-2 handshaking is

1. *READY* bit (bit 0, port 2) goes high.
2. Input the data in port 1 to the CPU.
3. *ACKNOWLEDGE* bit (bit 7, port 4) goes high to reset *READY* bit.
4. Reset the *ACKNOWLEDGE* bit.

Handshaking



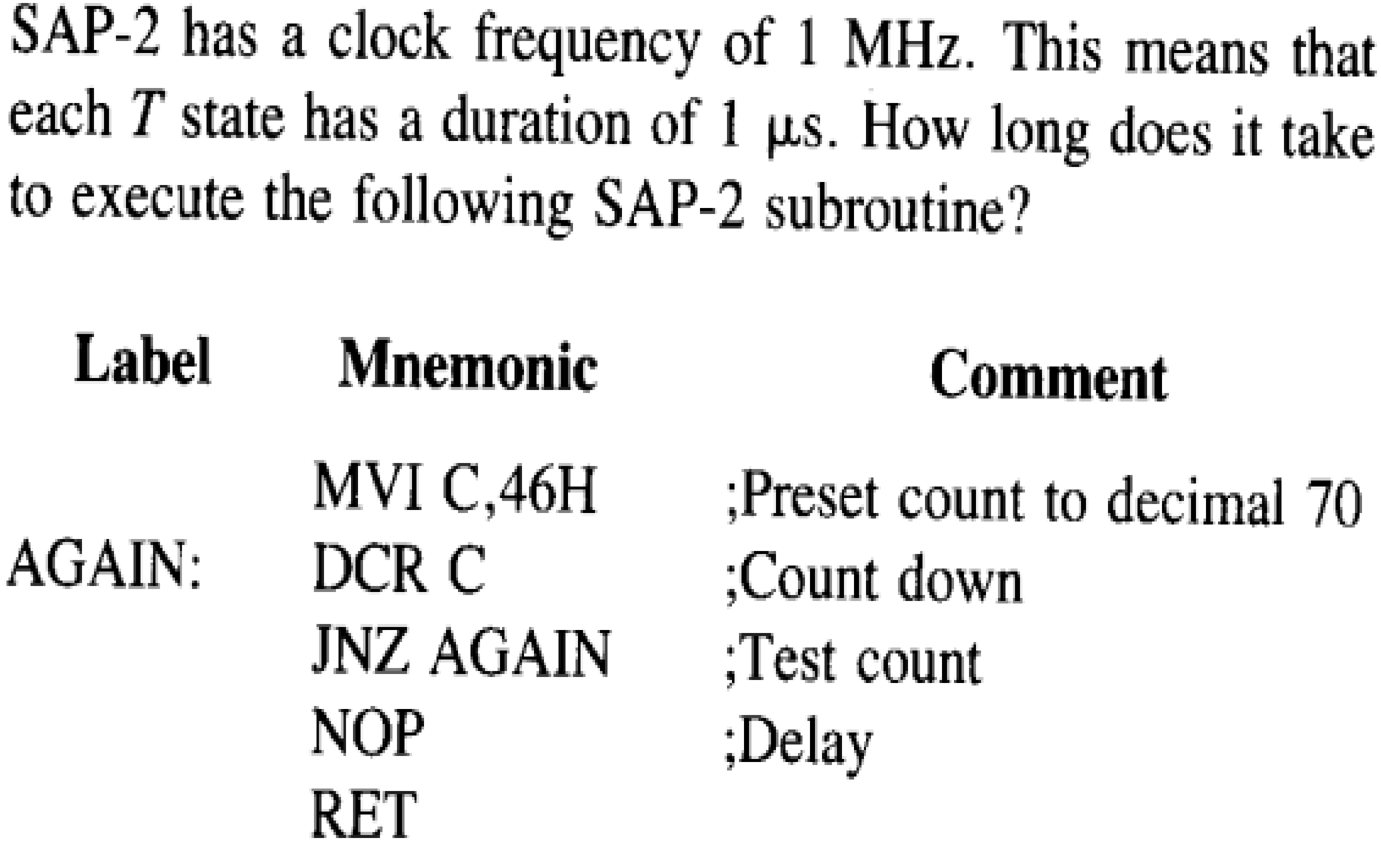
Write a program that inputs a byte of data from port 1 using handshaking. Store the byte in the B register.

SOLUTION

```
  Label     Mnemonic        Comment

STATUS:   IN 02H         ;Input byte from port 2
          ANI 01H        ;Isolate READY bit
          JZ STATUS      ;Jump back if not ready
          IN 01H         ;Transfer data in port 1
          MOV B,A        ;Transfer from A to B
          MVI A,80H      ;Set ACKNOWLEDGE bit
          OUT 04H        ;Output high ACKNOWLEDGE
          MVI A,00H      ;Reset ACKNOWLEDGE bit
          OUT 04H        ;Output low ACKNOWLEDGE
          HLT
```

Math-1

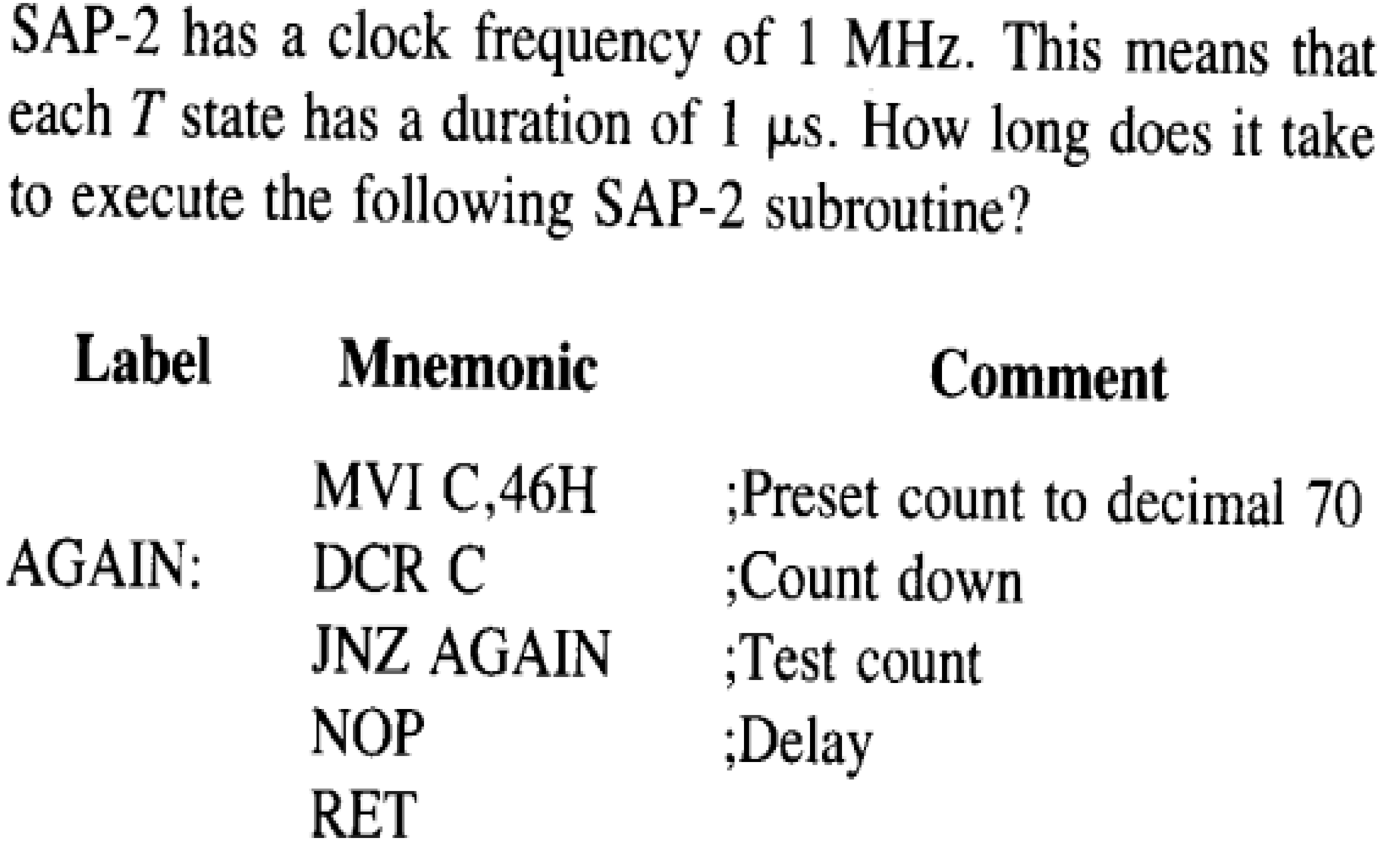


SAP-2 has a clock frequency of 1 MHz. This means that each *T* state has a duration of 1 μs. How long does it take to execute the following SAP-2 subroutine?

```
    Label      Mnemonic           Comment

               MVI C,46H      ;Preset count to decimal 70
AGAIN:         DCR C          ;Count down
               JNZ AGAIN      ;Test count
               NOP            ;Delay
               RET
```

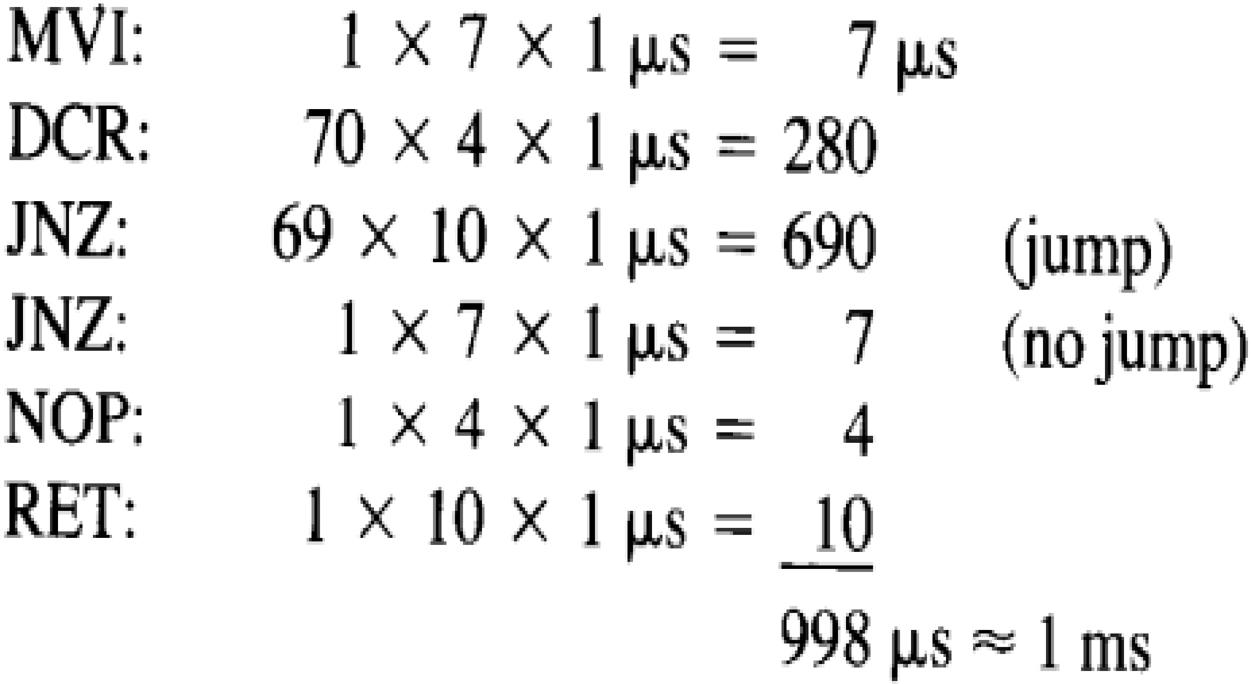
Math-1



SAP-2 has a clock frequency of 1 MHz. This means that each *T* state has a duration of 1 μs. How long does it take to execute the following SAP-2 subroutine?

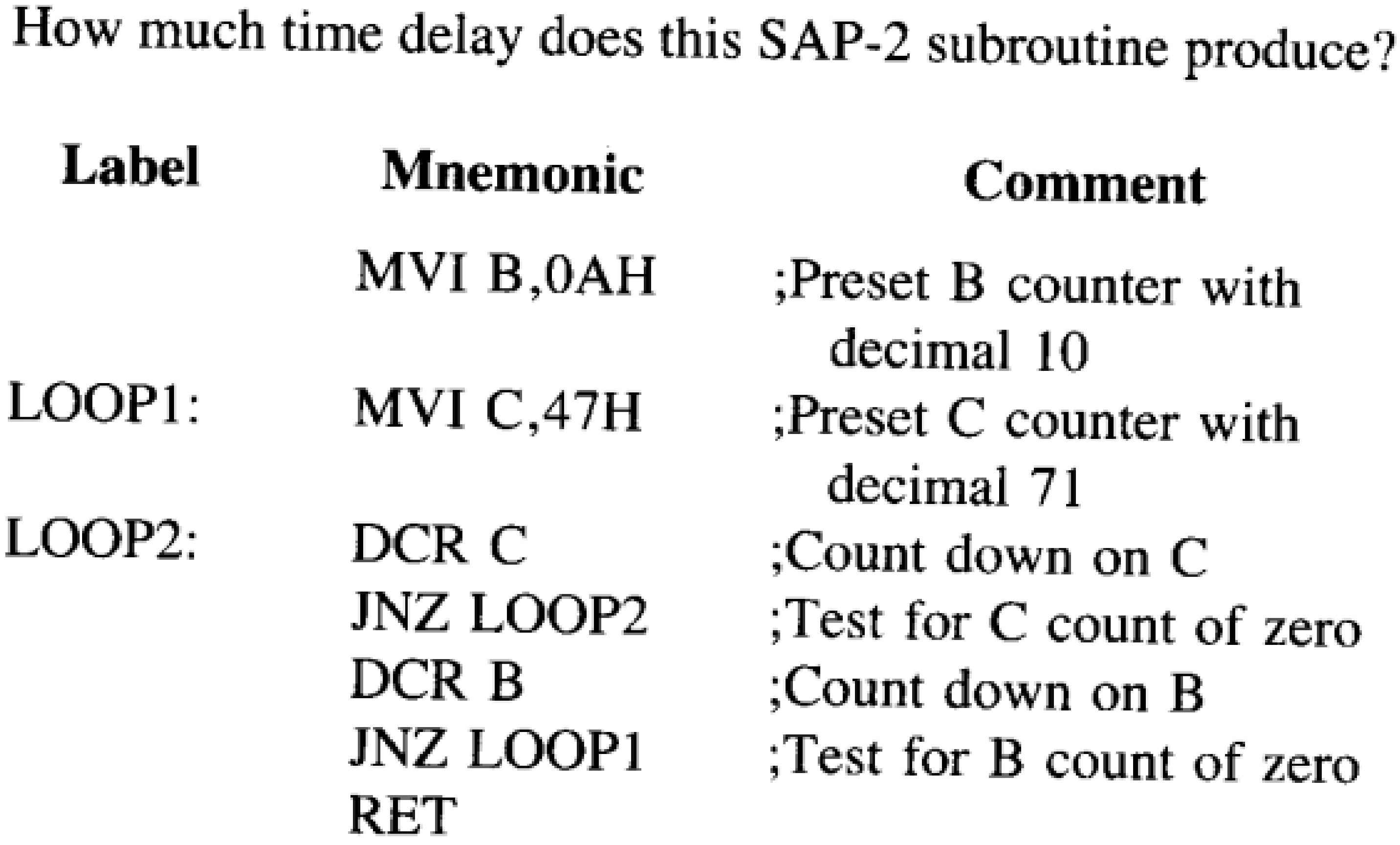
Label	Mnemonic	Comment
	MVI C,46H	;Preset count to decimal 70
AGAIN:	DCR C	;Count down
	JNZ AGAIN	;Test count
	NOP	;Delay
	RET	

Solution of Math-1



MVI:	$1 \times 7 \times 1\,\mu s =$	$7\,\mu s$	
DCR:	$70 \times 4 \times 1\,\mu s =$	280	
JNZ:	$69 \times 10 \times 1\,\mu s =$	690	(jump)
JNZ:	$1 \times 7 \times 1\,\mu s =$	7	(no jump)
NOP:	$1 \times 4 \times 1\,\mu s =$	4	
RET:	$1 \times 10 \times 1\,\mu s =$	$\underline{10}$	
		$998\,\mu s \approx 1\,ms$	

Math-2

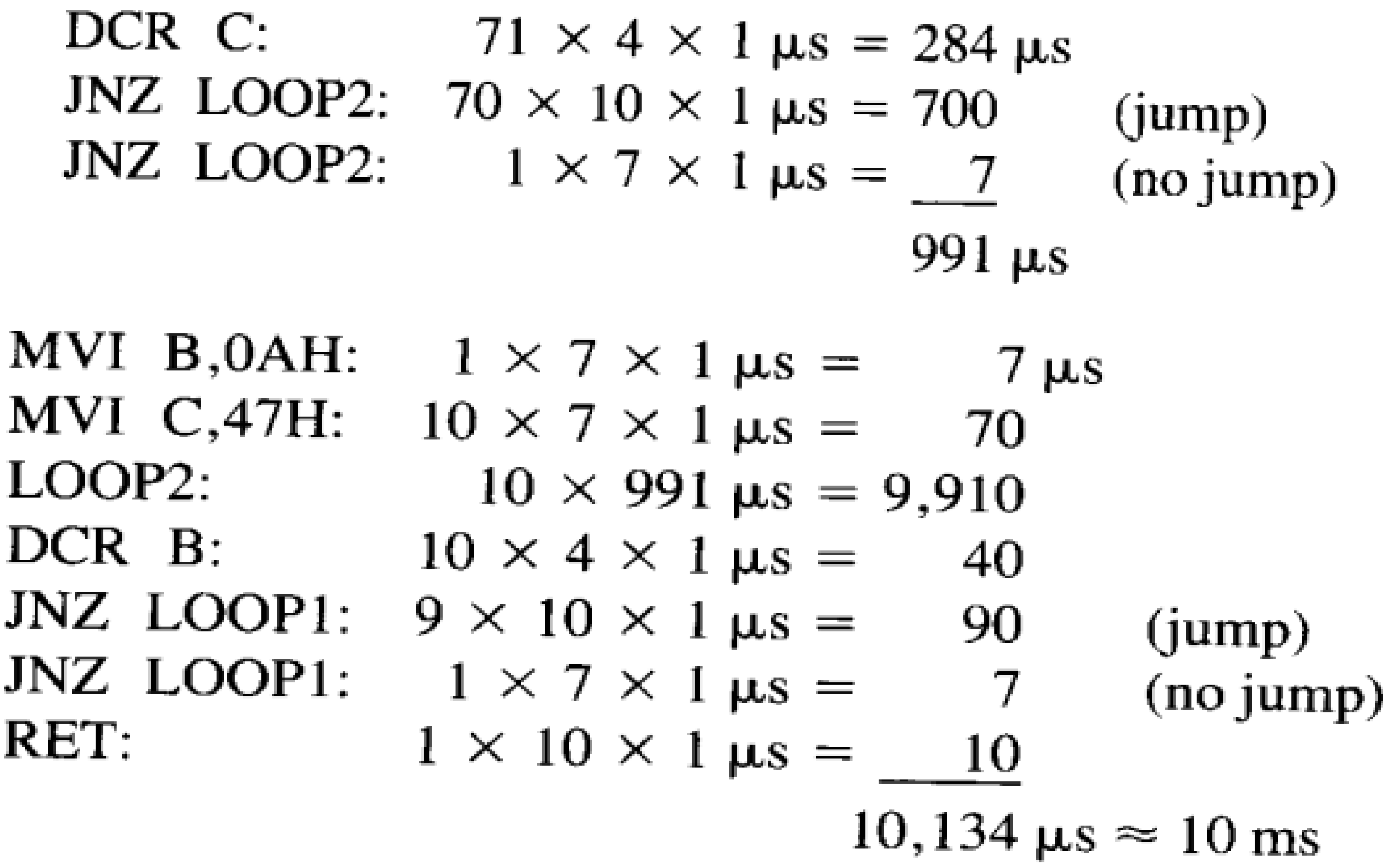


How much time delay does this SAP-2 subroutine produce?

```
Label     Mnemonic       Comment

          MVI B,0AH      ;Preset B counter with
                            decimal 10
LOOP1:    MVI C,47H      ;Preset C counter with
                            decimal 71
LOOP2:    DCR C          ;Count down on C
          JNZ LOOP2      ;Test for C count of zero
          DCR B          ;Count down on B
          JNZ LOOP1      ;Test for B count of zero
          RET
```

Solution of Math-2



DCR C:	71 × 4 × 1 µs =	284 µs	
JNZ LOOP2:	70 × 10 × 1 µs =	700	(jump)
JNZ LOOP2:	1 × 7 × 1 µs =	7	(no jump)
		991 µs	

MVI B,0AH:	1 × 7 × 1 µs =	7 µs	
MVI C,47H:	10 × 7 × 1 µs =	70	
LOOP2:	10 × 991 µs =	9,910	
DCR B:	10 × 4 × 1 µs =	40	
JNZ LOOP1:	9 × 10 × 1 µs =	90	(jump)
JNZ LOOP1:	1 × 7 × 1 µs =	7	(no jump)
RET:	1 × 10 × 1 µs =	10	
		10,134 µs ≈ 10 ms	

Acknowledgement

Md. Iftekharul Islam Sakib

Lecturer

CSE, BUET

Thank you