Comparison AND JUMP INSTRUCTIONS OF THE 8085 MICROPROCESSOR

Compare Operations (CMP, CPI) :- These instructions compare the accumulator with the required 8-bit data, setting the appropriate flag and leaving the accumulator untouched.

No	Instruction	Type	No. of Bytes	Function	Effect
1.	CMP r	Logical	1	Compare r with A (A-r)	All
2.	CPI byte	Logical	2	Compare byte with A (A-byte)	All

The result of the comparison is shown by setting the flags as follows:-
If A = (r/byte) then CY=0 and Z=1
If A < (r/byte) then CY=1 and Z=0
If A > (r/byte) then CY=0 and Z=0

Notes of flags affection

We can see from the previous experiments that only arithmetic and logical instructions affect the flag register, and we notice some exceptions.

1- The four rotate instructions, STC, CMC, and DAD instructions affect carry flag only and we can write the effect is: CY.
2- DCR and INR instructions affect all the flag bits except the carry bit and we can write the effect is: All but CY.
3- INX, DCX, and CMA instructions don’t affect any flag bits and we can write the effect is: None.
4- Other arithmetic and logical instructions affect all the flag bits and we can write the effect is: All.

It is known that the flow of some program may be deviated by specific jump instructions. These jumps test the status of the appropriate flags and jump accordingly to the specified address, given by the two bytes following the jump instruction in the order (Low Byte, High Byte). The types of JUMPs supported are:

1. JMP (address):- This instruction jumps unconditionally to the specified address.
2. JZ (address):- This instruction tests the zero flag bit, and jumps to the specified address if this bit is set.
3. JNZ (address):- This instruction tests the zero flag bit, and jumps to the specified address if this bit is reset.
4. JC (address):- This instruction tests the carry flag bit, and jumps to the specified address if this bit is set.
5. JNC (address):- This instruction tests the carry flag bit, and jumps to the specified address if this bit is reset.
6. JM (address):- This instruction tests the sign flag bit, and jumps to the specified address if this bit is set.
7. JP (address):- This instruction tests the sign flag bit, and jumps to the specified address if this bit is reset.
8. JPE (address):- This instruction tests the parity flag bit, and jumps to the specified address if this bit is set (even parity).
9. JPO (address):- This instruction tests the parity flag, bit, and jumps to the specified address if this bit is reset (odd parity).

No	Instruction	Type	No. of Bytes	Function	Effect
1.	JMP address	Branch	3	Pc=address	None
2.	JZ address	Branch	3	Pc=address if Z=1	None
3.	JNZ address	Branch	3	Pc=address if Z=0	None
4.	JC address	Branch	3	Pc=address if CY=1	None
5.	JNC address	Branch	3	Pc=address if CY=0	None
6.	JM address	Branch	3	Pc=address if S=1	None
7.	JP address	Branch	3	Pc=address if S=0	None
8.	JPE address	Branch	3	Pc=address if P=1	None
9.	JPO address	Branch	3	Pc=address if P=0	None

Pc: program counter. **Notice:** Jumps inst. Check the flags but not affect the flags

Example:-

Check if A+B = 0 then increment the result

```
        MVI A, …
        MVI B, …
        ADD B
        JNZ end
        INR A
End: RST1
```

Example:-

Check if A+B = 50 then increment the result

```
        MVI A, …
        MVI B, …
        ADD B
        CPI 50
        JNZ end
        INR A
End: RST1
```

Example:-

Check if A = 0 then increment A

```
        MVI A, …
        ANA A ; (/ ORA A/
              ; CPI 0/ ADI 0)
        JNZ end
        INR A
End: RST1
```

Designing a counter is a frequent programming application. Counters are used primarily to keep track of events.

A counter is designed by loading an appropriate count in a register. A loop is set up decrement the count for a down-counter (counts in the descending order) by using the DCR (decrement by one) instruction or to increment the count for an up-counter (counts in the ascending order) by using the INR (increment by one) instruction. A loop is established to update the counter, and each count is checked to determine whether it has reached the final number; if not the loop is repeated.

Examples:-

1- 1>= No. of loops < 256

Using down-counter:-

```
        MVI C, 8
Loop:   DCR C
        JNZ Loop
        RST1
```

Using UP-counter:-

```
        MVI C, 0
Loop:   INR C
        MOV A, C
        CPI 8
        JNZ Loop
        RST1
```

2- No. of loops = 256

Using down-counter:-

```
        MVI C, 0
Loop:   DCR C
        JNZ Loop
        RST1
```

3- No. of loops > 256

Using down-counter:-

```
        LXI B, 3b5h
Loop:   DCX B ; not effect Z flag
; To affect the Z flag and check if B or
C is not finished then continue the loop
        MOV A, C
        ORA B
        JNZ Loop
        RST1
```

No. of loops >= 256

Using UP-counter:-

```
        LXI B, 0
Loop:   INX B
        MOV A, C
        CPI lowbyteno
        JNZ Loop
        MOV A, B
        CPI highbyteno
        JNZ loop
        RST1
```

Note:- You can see that down-counter is preferred to use than up-counter.

Useful instructions:

PCHL: The contents of register H and L are copied into the program counter. The contents of H are placed as a high-order byte and of L as a low-order byte.

Instruction	Type	No. of Bytes	Function	Effect
PCHL	Branch	1	PC=HL	None

Class Work

1) Compare the value of register A and B, then obtain the last value of A:

A=A+10h when A=B , A=A-5 when A<B, A=A+B when A>B

Address	HexCode	Label	Opcode	Operands	Comments
2000 2001			MVI	A,	; A=
2002 2003			MVI	B,	; B=
2004			CMP	B	; A-B
2005 2006 2007	 10 20		JZ	EQUAL	; IF Z=1 THEN PC=2011
2008 2009 200A	 0D 20		JC	SMALLER	; IF CY=1 THEN PC=200E
200B			ADD	B	; A=A+B
200C	CF		RST1		; End
200D 200E	 05	SMALLER:	SUI	5	; A=A+5
200F	CF		RST1		; End
2010 2011	 10	EQUAL:	ADI	10	; A=A+10
2012			RST1		; End

2) Calculate the result of (8*8) by using two methods.

A) By using Summation method:

Address	HexCode	Label	Opcode	Operands	Comments
2000			XRA	A	; A=A XOR A=0, S=0,Z=1,AC=0, P=1, CY=0
2001 2002 2003	 08 08		LXI	B,808	; B=8, C=8
2004		NEXT:	ADD	B	; A=A+B
2005			DCR	C	; C=C-1
2006 2007 2008	 04 20		JNZ	NEXT	; IF Z=0 THEN PC=2004
2009			RST1		; End

B) By using Rotate method:

Address	HexCode	Label	Opcode	Operands	Comments
2000			MVI	A,8	; A=8
2001	08				
2002			RAL		; A=A*2=10
2003			RAL		; A=A*2=20
2004			RAL		; A=A*2=40
2005			RST1		; End

3- Calculate 11/4=2 and the remainder is 3

Address	HexCode	Label	Opcode	Operands	Comments
2000			MVI	A,0B	; A=0B
2001	0B				
2002			LXI	B, 400	; B=4, C=0
2003	00				
2004	04				
2005		NEXT:	CMP	B	; A-B
2006			JC	END	; IF CY=1 THEN PC=200E
2007	0E				
2008	20				
2009			SUB	B	; A=A-B
200A			INR	C	;C=C+1
200B			JMP	NEXT	; PC=2005
200C	05				
200D	20				
200E		END:	RST1		; End

4- Calculate the result of $C=B^2+5$ when B=5

Address	HexCode	Label	Opcode	Operands	Comments
2000			LXI	B,0505	; B=05, C=05
2001	05				
2002	05				
2003			XRA	A	; A=A XOR A=0, S=0,Z=1,AC=0, P=1, CY=0
2004		NEXT:	ADD	B	; A=A+B
2005			DCR	C	; C=C-1
2006			JNZ	NEXT	; IF Z=0 THEN PC=2004
2007	04				
2008	20				
2009			ADD	B	; A=A+B
200A			MOV	C,A	; C=A
200B		END:	RST1		; End

Homework

1- exclusive or register A and B, then add 3 to register C if the parity is even otherwise add 30h to C, when A=35h, B=20h, C=10h

2- Check if the content of register B is even then C=1, otherwise C=2. (by using two methods)

3- Calculate the sum of numbers between 10 and 1.