((بسمه تعالی))

آز ساختار کامپیوتر و میکروپروسسور

گزارش آزمایش ۲ و۳

جناب دکتر جاهد

اردوان کاویانی - ۹۸۱۰۶۸۷۵

رادمهر کریمیان - ۹۸۱۰۳۵۵۶

۱-۱) برنامه نهایی به شکل زیر است:

```
8  start: nop
9         MVI E,26H
10        MVI D,18H
11        MOV A,D
12        MVI D,00H
13 LOOP:  DAD D       ;4 T-STATE
14        DCR A       ;4 T-STATE
15        JNZ LOOP    ;10 T-STATE (7 if conditional has not
16        HLT
17        RET         ;10 T-STATE
```

که برای تست حاصل ضرب 26 و HEX 18 را محاسبه کردیم.

حاصل:

Flags

Z	☑
S	☐
P	☑
C	☐
AC	☑

Registers

A/PSW	0x 00 56
BC	0x 00 00
DE	0x 00 26
HL	0x 03 90
SP	0x FF FF
PC	0x 08 11

همانطور که انتظار داشتیم پاسخی که در HL ثبت شده 390 است و برابر حاصل ضرب ورودی ها است.

همچنین مقدار PC فلگ ها با جواب همخوانی دارد. (صورت کد)

۱-۲) کد برنامه :

main.asm

```
start:  MVI B,00H   ;B reg represent carry
        LDA 0000H   ;load 0000 of memory in A
        MOV H,A     ;copy A to H
        LDA 0001H   ;copy first oprand(27) in A
        ADD H       ;add second oprand(89) to A
        DAA         ;Adjust decimal ( add 6 if sum>9)
        JNC jump1   ;jump if cy is 0
        INR B       ;if cy=1: B<-cy
jump1:  NOP         ;just for jump
        STA 0061H    ;store 8bit reasult in 0061H
        MOV A,B     ;copy carry to accumulator
        STA 0060H    ;store carry in 0060H
        HLT
```

مقدار ورودی ها را در خانه های 0000 و 0001 حافظه مینویسیم و خروجی در 0060H ذخیره می‌شود:

	0	1	2	3	4	5	6	7	8	
000	89	28	00	00	00	00	00	00	00	C
001	00	00	00	00	00	00	00	00	00	C
002	00	00	00	00	00	00	00	00	00	C
003	00	00	00	00	00	00	00	00	00	C
004	00	00	00	00	00	00	00	00	00	C
005	00	00	00	00	00	00	00	00	00	C
006	01	17	00	00	00	00	00	00	00	C
007	00	00	00	00	00	00	00	00	00	C

۲) عدد ورودی 2 حالت دارد: الف) یا ورودی از 200 بزرگتر است که در این صورت خروجی مقدار ثابت 50 است. ب) یا ورودی کوچکتر از 200 است. در این حالت مقدار ورودی را انقدر از 50 کم می‌کنیم تا مقدار آن از 50 کمتر شود. آنگاه مقدار را در خروجی میریزیم.

شکل کد:

main.asm

```
1  MAIN:   MVI A,176D  ; A<-176
2          MVI B,50D   ; B<-50
3          CPI 200D    ; Compare Immediate A and 200
4          JNC LABEL2  ; Jump to LABEL2 if A>200
5          CMP B       ; Compare A and 50
6          JC LABEL1   ; Jump to Label1 if A<50
7          JNC LOOP    ; Jump to Loop if A>=50
8  LOOP:   SUB B       ; A=A-B
9          CMP B       ; Compare A and 50
10         JC LABEL1   ; Jump to Label1 if A<50
11         JNC LOOP    ; Jump to Loop if A>=50
12 LABEL1: STA 0000H   ; Output
13         HLT
14 LABEL2: LXI H,0 ; Load HL with 0000
15         MOV M,B     ; Output HL = 50 (A > 200)
16         HLT
```

خروجی:

	0	1	2
000	1A	00	0(
001	00	00	0(
002	00	00	0(

3) ابتدا درنظر بگیریم که ورودی 4 کلید ما به شکل یک کد باینری و در چهار بیت اول یک ریجستر ذخیره شده است.

حال یک کد دلخواه 4 بیتی را تصور کنید:

1101

اگر ما از سمت چپ شروع کنیم و به سمت راست بیاییم و در نهایت به رقم اخر رسیدیم مجددا به رقم اول بازگردیم خواهیم داشت:

1101 1101 1101

110111011101

که اگر دقیق تر نگاه کنیم به نسبت تعداد یک های مان یعنی 75 درصد سیگنال ما یک میشود.

کافیست این سیگنال مرتب شده را به خروجی دهیم.

اما ازانجایی که ورودی ما به شکل 0000XXXX می باشد باید ابتدا چهار بیت بالا را نیز برابر با XXXX قرار دهیم. به این منظور ورودی را در اکامولیتور ریخته و با چهار با شیفت دادن به مقدارXXXX0000 می رسیم سپس اگر این دورا جمع کنیم به مقدار XXXXXXXX خواهیم رسید.

حال داریم که با چرخش به شکل لوپ در این خروجی میتوان به مقدار دلخواه رسید

توجه کنید که بیت پایین این سریال را به مقدار 01H اند میکنیم تا خروجی ما صفر یا یک بماند

سپس آن را به خروجی منتقل میکنیم.

```
1   MVI H,0AH ;an example input for our keys
2   MOV A,H ; set our input to acu
3   RLC ; rotate to left *4
4   RLC
5   RLC
6   RLC
7   ADD H ; add H to four time routated H
8   MOV H,A ; set H to A
9   MOV D,A ; set H to A
10  ROTATE: MOV A,D ;set D to A for loop
11      RLC ;rotate A for first time
12      MOV D,A ; renew A cuz we want countine from here not first
13      ANI 01H ; and to make our output
14      OUT 10H ;Output the D0 bit
15
16  JMP ROTATE ;Jump to ROTATE to change logic level
```

برای مثال یک حالت خروجی را برای حالت 1010 میبینیم:

در انتهای خط 9 به زیبایی داریم:

Register	Value	7	6	5	4	3	2	1	0
Accumulator	AA	1	0	1	0	1	0	1	0
Register B	00	0	0	0	0	0	0	0	0
Register C	00	0	0	0	0	0	0	0	0
Register D	AA	1	0	1	0	1	0	1	0
Register E	00	0	0	0	0	0	0	0	0
Register H	AA	1	0	1	0	1	0	1	0
Register L	00	0	0	0	0	0	0	0	0
Memory(M)	00	0	0	0	0	0	0	0	0

سپس برای اولی خروجی داریم که:

Register	Value	7	6	5	4	3	2	1	0
Accumulator	01	0	0	0	0	0	0	0	1
Register B	00	0	0	0	0	0	0	0	0
Register C	00	0	0	0	0	0	0	0	0
Register D	55	0	1	0	1	0	1	0	1
Register E	00	0	0	0	0	0	0	0	0
Register H	AA	1	0	1	0	1	0	1	0
Register L	00	0	0	0	0	0	0	0	0
Memory(M)	00	0	0	0	0	0	0	0	0

و برای دومین خروجی نیز:

Register	Value	7	6	5	4	3	2	1	0
Accumulator	00	0	0	0	0	0	0	0	0
Register B	00	0	0	0	0	0	0	0	0
Register C	00	0	0	0	0	0	0	0	0
Register D	AA	1	0	1	0	1	0	1	0
Register E	00	0	0	0	0	0	0	0	0
Register H	AA	1	0	1	0	1	0	1	0
Register L	00	0	0	0	0	0	0	0	0
Memory(M)	00	0	0	0	0	0	0	0	0

درپایان کد ما در این لینک آماده است.

4)در این قسمت داریم که:

```
1   MVI A,1CH
2   LOOP1:  DCR A
3           JNZ LOOP1
4
5   HLT
```

به راحتی و برای مقدار دهی مقدار دیلی خود داریم که:

$$N_{delay} = 7 + 1 \times (4 + 7) + 27 \times (4 + 10) + 5 = 401$$

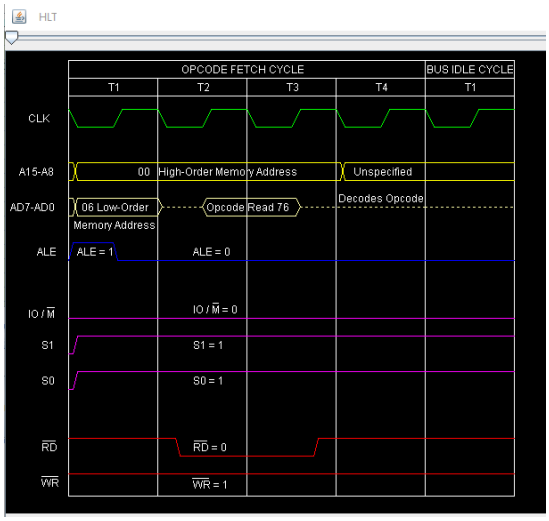
$$T_{delay} = N_{delay} \times \frac{1}{f_{clk}} = 401 \times 0.5\ \mu s = 0.2005\ ms \approx 0.2\ ms$$

که با شبیه سازی نیز داریم:

Type	Value
Stack Pointer(SP)	0000
Memory Pointer (HL)	0000
Program Status Word(PSW)	0054
Program Counter(PC)	0006
Clock Cycle Counter	401
Instruction Counter	58

پس کد ما درست اجرا شده است.

حال برای سیگنال ها و ویو فرم داریم که:



که مطابق خواسته ما می باشد.

همچنین داریم که:

علت مقدار دهی 28 دسیمال به عنوان دیلی نیز بر این علت استوار است که تعداد استیت های ما مشخص شود همچنین t-state 7 در خط اول داریم.

در خط دوم t-state 4 داریم.همچنین در خط سوم t-state 10 داریم که به در محاسبات به خوبی لحاظ شده است.

کد ما نیز در این لینک آمده است.