Lab: 4
Familiarization with branching and stack operations

The Branching and stack operations

Branching is the process of transferring the program control to somewhere else instead of executing next instruction. The branching instructions change the content of the program counter content instead of next address. The stack is a portion of the RAM area which is separated to store and retrieve data temporarily without bothering about the address. Stack pointer is the register which is involved in the stack operation. The stack pointer holds the address of the top of the stack. Stack is automatically involved in call and return instructions.

a) The stack operations (`PUSH, POP, SPHL, and XTHL`)

These instructions are used in storing the content of the registers temporarily in the stack and recover the stack content afterwards. These instructions are used as follows:

```
PUSH Rp
POP Rp
SPHL
XTHL
```

Here `Rp` represents reg pair B, D, H, PSW (combination of flag and register).

`PUSH Rp` instruction stores the 16-bit data of the reg pair in the stack. To store the data `PUSH` instruction first decreases the SP and stores the higher byte in this address and decreases SP again and stores the lower byte in this address After the `PUSH` operation the SP is decreased by 2.

`POP` instruction in the other hand retrieve the data stored in the stack. To recover the data from the stack the `POP` instruction first copies the 8-bit data and stored in the register which holds the lower byte of the data and the SP is increased by 1 and the data from this address is copied to the register which hold the higher byte of the data and SP is again increased by one. After the `POP` instruction the SP is increased by 2.

`SPHL` instruction copies the content of the reg. pair HL to the stack pointer.

`XTHL` instruction exchanges the data at the top of the stack with the HL reg. pair.

Example 1: Load the following program:

```
8000 LXI B, BBBB
8003 LXI D, DDDD
8006 PUSH B
8007 PUSH D
8008 LXI B, 1234
800B LXI D, 5678
800E POP D
8012 POP B
8013 RST 5
```

Run the program in single step mode and examine the content of SP and the stack. What happens if the instruction at 800E and 8012 are exchanged? How is the SP and stack affected if you use `SPHL` instruction before the push operation? Change the program and see the effect.

Example 2: Load and verify the following program

```
8000 LXI D, 1122
8003 LXI H, 805A
8006 SPHL
8007 XCHG
8008 XTHL
8009 RST 5

805A 33
805B 44
```

Run this program in single step mode and examine the reg. pair HL before and after the execution of the instruction `XTHL`. Also note the data at 805A and 805B after the program execution.

The flags contents can be changed if the value to be made in the flags register is first stored in the lower part of reg. pair and pushed to the stack. And if the stack content is popped in the PSW, the content in the reg. pair is transferred in the accumulator and flags.

Assignment

1. Write a program to set zero and parity flags and reset other flags.
2 Write a program to set auxiliary flag and reset parity flag without affecting other flags.

b) Jump instructions (`JMP, Jx, PCHL`)

Jump instructions are used to transfer the control of the program to some other location instead of the next instruction. These instructions are used as follows

JMP 16-bit	Unconditional Jump	JM 16-bit	Jump on minus
JNZ 16-bit	Jump on no zero	MP 16-bit	Jump on plus
JZ 16-bit	Jump on zero	JPE 16-bit	Jump on parity even
JNC 16-bit	Jump on no carry	JPO 16-bit	Jump on parity odd
JC 16-bit	Jump on carry		

JMP instruction is the unconditional jump and Jx instruction is the conditional jump. Conditional jumps use the flag conditions for the branching.

PCHL instruction copies the content of the HL reg. pair into PC, i.e., this command branches the control to the location specified by HL reg. pair.

Looping is done with the conditional jump instruction. When we have to insert delay we can use the loops for the delay.

Example 3: Load and run the following program

```
8000 MVI A, 80H
8002 OUT 43H
8004 MVI A, 01
8006 OUT 40
8008 RLC
8009 JMP 8006
800C RST 5
```

Run this program in single step mode and note the output in port A and note the sequence of the execution of the instructions. Does the program terminate? Now insert a delay loop and run the program in full speed.

Example 4: Load the following program

```
8000 MVI A, 80H
8002 OUT 43H
8004 MVI A, 01
8006 MVI B, FF
8008 OUT 40
800A INR A
800B DCR B
800C JNZ 8008
800F RST 5
```

Run this program in single step mode and view the content of the program. Does the program terminate now? Explain how this loop works. Now insert a delay loop and run the program in full speed. Can you see the output?

Example 5: Load the following program

```
8050 MVI A, 80
8052 OUT 43
8054 MVI A, FF
8056 LXI H, 8080
8059 PCHL
805A RST 5
```

Also enter

```
8080 DCR A
8081 OUT 42
8083 JMP 8080
```

Run this program in single step mode and see what happens when PCHL and JMP instructions are executed.

Assignment

3 Write a program to count the no of bits that are 1 in register A.
4 Write a program to add nos. from one to fifty and display the 16 bit result at port A & B.
5 Write a program that will count up from 00 to FF at port A. Be sure to use PCHL command.

c) Call and return instructions

Like JUMP command, CALL command changes the normal sequence of executing instructions. The objective is to have the computer go off and execute a series of program steps, called subroutine. Unlike the JUMP command, the CALL causes the computer to remember where it used CALL, so it can go back to the main program when it finds a RET command during execution.

The call instructions are used as follows.

CALL 16-bit	Unconditional Call	CM 16-bit	Call on minus
CNZ 16-bit	Call on no zero	CP 16-bit	Call on plus
CZ 16-bit	Call on zero	CPE 16-bit	Call on parity even
CNC 16-bit	Call on no carry	CPO 16-bit	Call on parity odd
CC 16-bit	Call on carry		

The return instructions are used as follows.

RET	Unconditional Return	RM	Return on minus
RNZ	Return on no zero	RP	Return on plus
RZ	Return on zero	RPE	Return on parity even
RNC	Return on no carry	RPO	Return on parity odd
RC	Return on carry		

CALL instruction is the unconditional call and Cx instructions are the conditional calls. Similarly RET instruction is the unconditional return and Rx instruction is the conditional return. Conditional call and return use the flag conditions for the branching and return.

Example 6: Load and verify the following program

```
8000 MVI A, 80          8010 INR A              8020 ADI 03
8002 OUT 43             8011 CALL 8020          8022 RET
8004 MVI A, 01          8014 RET
8006 CALL 8010
8009 OUT 42
800B RST 5
```

Run this program and note down the sequence of the execution of the instructions. What is the output at port 42? Note down the SP content before and after the execution of the CALL and RET instructions also observe the stack content.

Conditional calls are useful if the call is to be occurred if some condition is satisfied. The conditional calls occur depending upon the flag conditions.

Example 7: Load the following program

```
8000 MVI A, 80          8020 MVI A, FF          8030 MVI A, 01
8002 OUT 43             8022 OUT 42             8032 OUT 41
8004 LDA 8050           8024 RET                8034 RET
8007 CPI 01
8009 CZ 8020
800C CNZ 8030
800F RST 5

8050 DATA FF
```

Run this program and examine where the jump occurs (at 8020 or at 8030). Change the data at 8050 with 01 and see where the jump occurs. In the above two cases what output do you see in the port.

Conditional return instructions are used in returning from the subroutines when some condition occurs.

Example 8: Load the following program

```
8000 MVI A, 80          8020 LXI B, FFFF
8002 OUT 43             8023 DCR B
8004 MVI A, 01          8024 JNZ 8023
8006 OUT 40             8027 DCR C
8008 CALL 8020          8028 RZ
800B RLC                8029 JMP 8023
800C JMP 8002
```

Run this program in full speed and explaining what is happening.

Assignments

6 Write a program to transfer the data at 8020 to 8030 if the data is greater than 127. You can assume data yourself.

7 Write a program to rotate the data 3C in port A. Call a delay subroutine for the delaying.

8 Write a program that will check whether the bit D_6 of a number stored at 8123 is 0 and its bit D_3 is 1. If the condition satisfies display the number.

9 Write a program that will check whether the number in reg. B is even or not. If the number is even display it in port A.