SUBROUTINE AND STACK OPERATIONS

In this experiment, a rather important group of instructions is further studied namely the subroutines commands. The subroutine is a program which is executed by (CALL) instruction and after finishing it operation its returns to the address called from by a (RET) instruction. This is useful for replacing repetitive block of operations or for properly organizing programs.

The stack is a section of memory utilizing as a Last In First Out (LIFO). This operation is useful for keeping track of the program flow, i.e. the last subroutine call is stored on the top of the stack. Therefore when a return from a subroutine is executed, the address is got from the top of the stack which is reduced by two, to point to a next subroutine address.

The stack can be simulated by a pack of books where the last book is put in the top of the stack, and it will be then the first book to be removed from the stack.

Stack Instructions

1. PUSH rp :- This instruction puts the content of the required rp on the stack, the mechanism of this instruction are:
   a. Decrement the stack pointer by two.
   b. Put rp on the stack.

2. POP rp :- This instruction loads a register pair from the contents of the top of stack, the mechanism of this instruction are:

   a. Get rp from top of the stack.
   b. Increment stack pointer by two.
3. XTHL: The contents of the L register are exchanged with the stack location pointed out by the content of the stack pointer register. The contents of the H register are exchanged with the next stack location (sp+1).
4. SPHL: Loads the content of the H and L registers into the stack pointer register, the contents of the H register provide the high-order address and the contents of the L register provide the low-order address.

No	Instruction	Type	No. of Bytes	Function	Benefits	Effect
1.	Push rp/psw	stack	1	Push rp: (sp=sp-2, M_{sp}=Lreg M_{sp+1}=Hreg) Push psw: (sp=sp-2, M_{sp}=F M_{sp+1}=A)	1- To save the contents of useful registers which needed later 2- Exchange data 3- subroutine	None
2.	Pop rp/psw	stack	1	Pop rp: (Lreg=Msp, Hreg=M_{sp+1}, sp=sp+2) Pop psw: (F=Msp, A=M_{sp+1}, sp=sp+2)		None
3.	XTHL	Data Transfer	1	HL ⟷ Top of Stack	To make Exchange	None
4.	LXI sp,add	Data Transfer	3	SP=add	To let the sp points from specific location of memory	None
5.	SPHL	Data Transfer	1	SP=HL		None
6.	INX SP	Arithmetic	1	SP=SP+1		None
7.	DCX SP	Arithmetic	1	SP=SP-1		None
8.	DAD SP	Arithmetic	1	HL=HL+SP		CY

PSW: high byte=A, Low byte=Flag register
SP: stack pointer

Useful instructions:

XCHG: The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E.

Instruction	Type	No. of Bytes	Function	Effect
XCHG	Data Transfer	1	HL ⟷ DE	None

Subroutine Instructions

The subroutine operation can be simplified by the block diagram given in Figure 6.

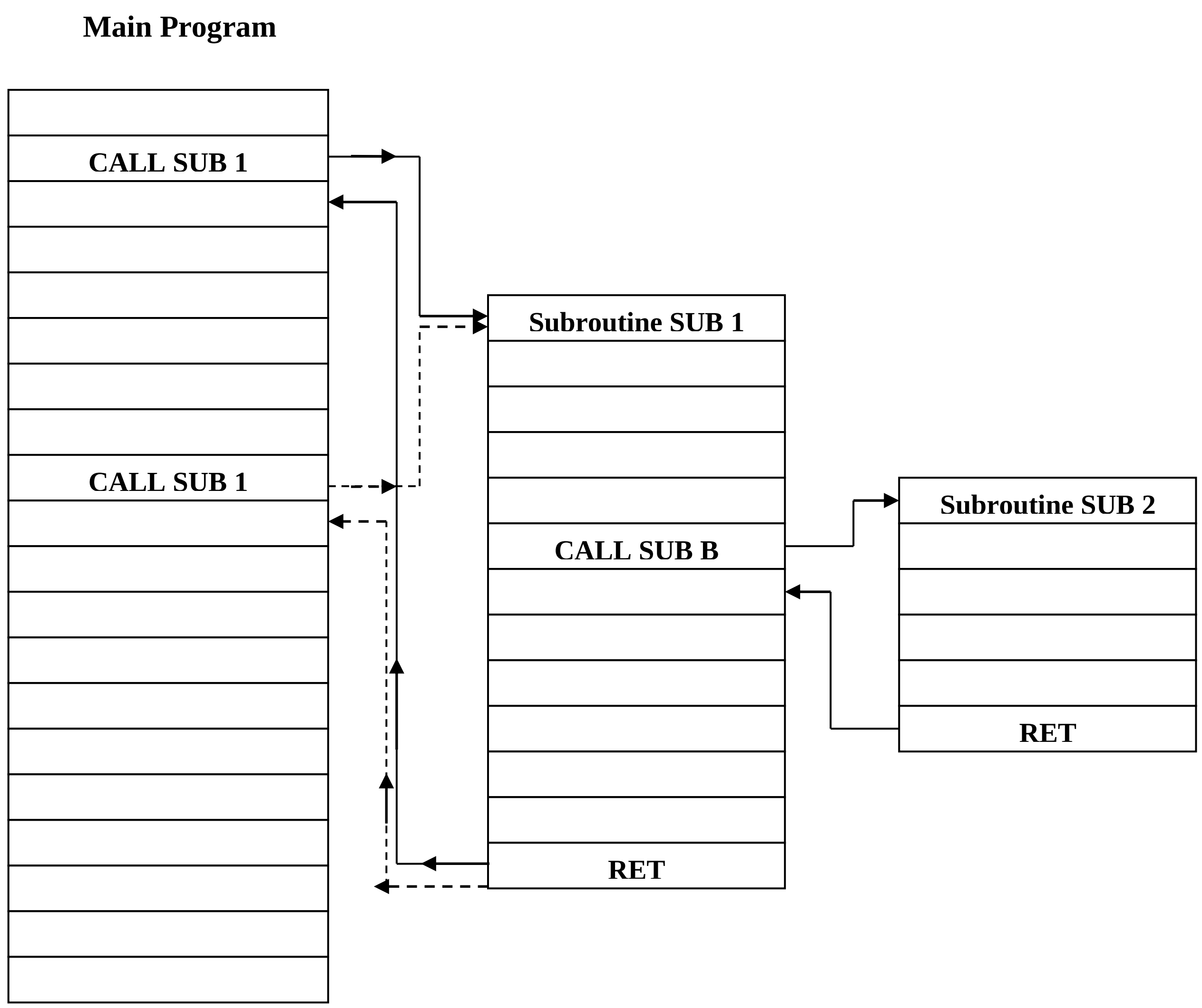

Figure (6): illustrating the mechanism of subroutine operations.

The subroutine instructions falling under this category are:

1. CALL (address): This instruction performs an unconditional call to a subroutine at the assumed address. The mechanics behind the CALL instruction are:
   a. The top of the stack is decremented by two.
   b. Program counter (after fetch instruction) is put on the stack.
   c. Program counter is then modified to the new address.

   This mechanics can be viewed clearly from this example (PC=2000), (SP=2099):

			PC after fetch	PC after execute	SP after execute
2000	ADI	04	2002	2002	2099
2002	CALL	2040	2005	2040	2097
2005	ANA	B			

2. CC (address):- This instruction performs the conditional subroutine call to the given address, if the carry flag bit is set.

3. CNC (address):- This instruction performs the conditional subroutine call to the given address, if the carry flag bit is reset.

4. CZ (address):- this instruction performs the conditional subroutine call to the given address, if the zero flag bit is set.

5. CNZ (address):- This instruction performs the conditional subroutine call to the given address, if the zero flag bit is reset.

6. CM (address):- This instruction performs the conditional subroutine call to the given address, if the sign flag bit is set.

7. CP (address):- This instruction performs the conditional subroutine call to the given address, if the sign flag bit is reset.

8. CPE (address):- This instruction performs the conditional subroutine call to the given address, if the parity flag bit is set.

9. CPO (address):- This instruction performs the conditional subroutine call to the given address, if the parity flag bit is reset.

No	Instruction	Type	No. of Bytes	Function	Effect
1.	Call add	Branch	3	unconditional	None
2.	CC add	Branch	3	call if CY=1	None
3.	CNC add	Branch	3	call if CY=0	None
4.	CZ add	Branch	3	call if Z=1	None
5.	CNZ add	Branch	3	call if Z=0	None
6.	CM add	Branch	3	call if S=1	None
7.	CP add	Branch	3	call if S=0	None
8.	CPE add	Branch	3	call if P=1	None
9.	CPO add	Branch	3	call if P=0	None

10. RET:- This instruction performs an unconditional return from the subroutine to the calling entry point of it. The mechanism of the RET instruction are:

    a. Move the contents of the stack pointer to the program counter.
    b. Increase stack pointer by two.

    This can be illustrating by the following example (PC=2040), (SP=2097):

			PC after fetch	PC after execute	SP after execute
2040	SBI	03	2042	2042	2097
2042	RET		2043	2005	2099
2043					

The conditional instructions are: RZ, RNZ, RC, RNC, RPE, RPO, RP and RM. The conditions, in which these instructions are operating according to flags, are the same as these illustrating with the conditional call instructions.

No	Instruction	Type	No. of Bytes	Function	Effect
1.	RET	Branch	1	unconditional	None
2.	RC	Branch	1	Return if CY=1	None
3.	RNC	Branch	1	Return if CY=0	None
4.	RZ	Branch	1	Return if Z=1	None
5.	RNZ	Branch	1	Return if Z=0	None
6.	RM	Branch	1	Return if S=1	None
7.	RP	Branch	1	Return if S=0	None
8.	RPE	Branch	1	Return if P=1	None
9.	RPO	Branch	1	Return if P=0	None

Notes

- Sometimes in the subroutines we need to use some registers which content value are useful in the main program then we must save them by pushing them at the start of this subroutine and popping them at the end of this subroutine to retrieve them content for the main program.
- Sometimes we need to pass parameters to the subroutine. For example when the subroutine works on register D and in the main program we need this subroutine to work on register B then before calling this subroutine we must move the content of register B to register D.

Class Work

1. Write a program to reset the flag register.

Address	HexCode	Label	Opcode	Operands	Comments
2000 2001 2002	 50 20		LXI	SP,2050	; SP=2050
2003 2004 2005	 00 00		LXI	H,00	; HL=00
2006			PUSH	H	; [SP]=L, [SP+1]=H
2007			POP	PSW	; A=[S], F=[SP+1]
2008			RST1		; RST1

2. Write a program using subroutine which adds two registers and check the result if it is equal to 10 then return to main program, otherwise add 5 to the result and return. In both cases in the main program after returning from this subroutine subtract 3 from the result.

Address	HexCode	Label	Opcode	Operands	Comments
2000 2001 2002	 50 20		LXI	SP,2050	; SP=2050
2003 2004			MVI	A,	; A=
2005 2006			MVI	B,	; B=
2007 2008 2009	 0D 20		CALL	ADDS	; PC=
200A 200B	 03		SUI	3	A=A-3
200C			RST1		; END
200D		ADDS:	ADD	B	; A=A+B
200E 200F	 0A		CPI	0A	; A-0A
2010			RZ		; IF Z=1 then PC=address after call
2011 2012	 05		ADI	5	; A=A+5
2013			RET		; PC=address after call

3. C= (B^2+5) AND (D^2+5)

Address	HexCode	Label	Opcode	Operands	Comments
2000 2001 2002	 50 20		LXI	SP,2050	; SP=2050
2003 2004			MVI	B,	; B=
2005			MOV	A,B	; A=B
2006 2007 2008	 13 20		CALL	FINDS	; PC=2013
2009			MOV	L,A	; L=A
200A 200B			MVI	D,	; D=
200C			MOV	A,D	; A=D
200D 200E 200F	 13 20		CALL	FINDS	; PC=2013
2010			ANA	L	; A=A AND L
2011			MOV	C,A	; C=A
2012			RST1		; END
2013		FINDS:	MOV	E,A	; E=A
2014			MOV	C,A	; C=A
2015			DCR	C	; C=C-1
2016		NEXT:	ADD	E	; A=A+E
2017			DCR	C	; C=C-1
2018 2019 201A	 16 20		JNZ	NEXT	; IF Z=0 then PC=2016
201B 201C	 05		ADI	5	A=A+5
201D			RET		; PC=ADDRESS AFTER CALL

Homework

1. Exchange DE, HL using all possible methods.
2. C=(9*7)+(6*4)
3. Write a program to copy the odd number of any array started at location (2060H) and ended at location (2065H) to another array starting at address (2080H). The even number must be stored at memory locations starting at address (20A0H). All the copy operations above must be carried out in a subroutine.
4. Write a program to find the average of four data bytes stored at memory locations (2030H-2033H). Store the result in register (E). The summation of the data bytes must be carried out in a subroutine. Assuming that the result of summation does not exceed an (8-bit). Data: (10,2f,1d,4a)