LECTURE:14 STACK & SUBROUTINES

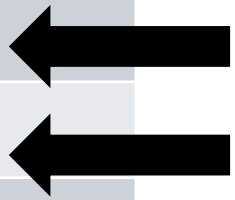
COURSE INSTRUCTOR: DR. DEVYANI GUPTA

Stack

B=05 LXI B,0571

C=71 PUSH B

Stack pointer	Data on Stack	Stack pointer address
SP-2	71	FFFD
SP-1	05	FFFE
SP	XX	FFFF



PUSHING on stack starts from here

Initial SP

SP-1 is the higher address, so, higher data byte will be stored.

SP-2 is the lower address, so, lower data byte will be stored.

√	2000		LXI SP,FFFF	31	3	3	10
	2001			FF			
	2002			FF			
√	2003		LXI B,0571	01	3	3	10
	2004			71			
	2005			05			
√	2006		PUSH B	C5	1	3	12
√	2007		HLT	76	1	2	5

Memory Address	Value
2000	31
2001	FF
2002	FF
2003	01
2004	71
2005	05
2006	C5
2007	76
FFFD	71
FFFE	05

Register	Value	7	6	5	4	3	2	1	0
Accumulator	00	0	0	0	0	0	0	0	0
Register B	05	0	0	0	0	0	1	0	1
Register C	71	0	1	1	1	0	0	0	1
Register D	00	0	0	0	0	0	0	0	0
Register E	00	0	0	0	0	0	0	0	0
Register H	00	0	0	0	0	0	0	0	0
Register L	00	0	0	0	0	0	0	0	0
Memory(M)	00	0	0	0	0	0	0	0	0

Resister	Value	S	Z	*	AC	*	P	*	CY
Flag Resister	00	0	0	0	0	0	0	0	0

Type	Value
Stack Pointer(SP)	FFFD
Memory Pointer (HL)	0000
Program Status Word(PSW)	0000
Program Counter(PC)	2007
Clock Cycle Counter	42
Instruction Counter	5

Stack pointer	Data on Stack	Stack pointer address
SP	71	FFFD
SP+1	05	FFFE
SP+2	XX	FFFF

Initial SP or top of STACK

Final SP

Popping starts from TOP of STACK

*	Address	Label	Mnemonics	Hexcode	Bytes	M-Cycles	T-States
√	2000		LXI SP,FFFF	31	3	3	10
	2001			FF			
	2002			FF			
√	2003		LXI B,0571	01	3	3	10
	2004			71			
	2005			05			
√	2006		PUSH B	C5	1	3	12
√	2007		POP D	D1	1	3	10
√	2008		HLT	76	1	2	5

Register	Value	7	6	5	4	3	2	1	0
Accumulator	00	0	0	0	0	0	0	0	0
Register B	05	0	0	0	0	0	1	0	1
Register C	71	0	1	1	1	0	0	0	1
Register D	05	0	0	0	0	0	1	0	1
Register E	71	0	1	1	1	0	0	0	1
Register H	00	0	0	0	0	0	0	0	0
Register L	00	0	0	0	0	0	0	0	0
Memory(M)	00	0	0	0	0	0	0	0	0

Resister	Value	S	Z	*	AC	*	P	*	CY
Flag Resister	00	0	0	0	0	0	0	0	0

Type	Value
Stack Pointer(SP)	FFFF
Memory Pointer (HL)	0000
Program Status Word(PSW)	0000
Program Counter(PC)	2008
Clock Cycle Counter	104
Instruction Counter	12

Instructions related to stacks

- **LXI SP, 16 bit value** Eg. LXI SP 4000.
- **SPHL** Load SP with HL contents
- **PCHL** Load PC with HL contents
- **INX SP**
- **DCX SP**
- **PUSH Rp** Eg PUSH B
- **POP Rp** Eg POP B
- **CALL 16 bit address** Eg CALL 2000 •
- **RSTn** Eg RST1
- **RET**
- **XTHL** Exchange HL and SP

Uses of stacks

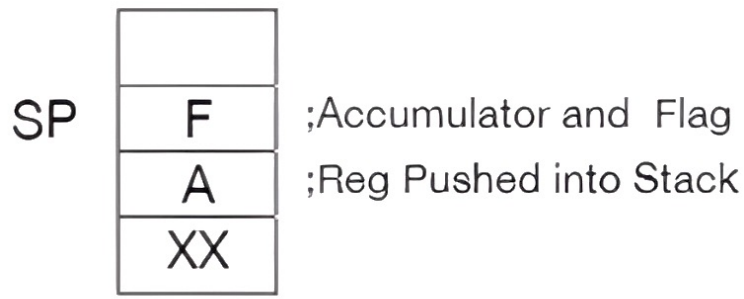
- To store data
- To store return address
- To read/write flags
- To pass parameters to the subroutine

To Read the contents of Flag register

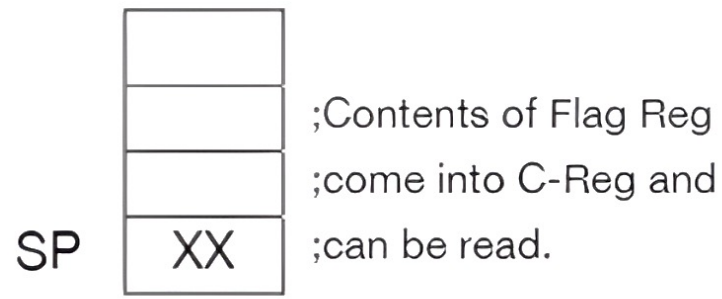
- The programmer pushes PSW into the stack and then pops it into any register pair (BC).
- The contents of the flag register are thus available in the C register (lower register).

Eg:

PUSH PSW



POP B

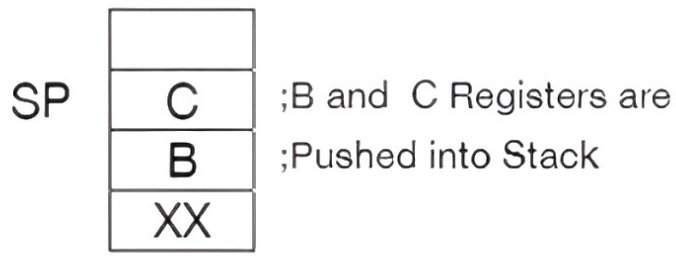


To Write into the Flag register

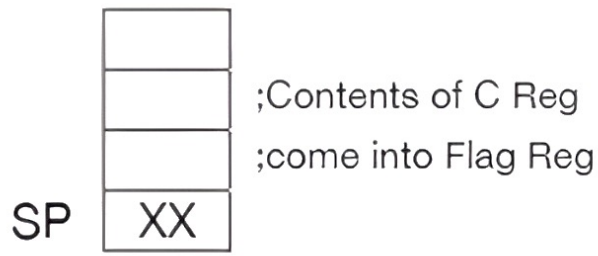
- The programmer loads the appropriate byte into the lower register of a register pair (eg: C reg of BC pair).
- Then the register pair is pushed into the stack.
- These contents are then popped into the PSW, and thus the byte that was originally loaded into C register is now **written** into the Flag register.

Eg:

PUSH B



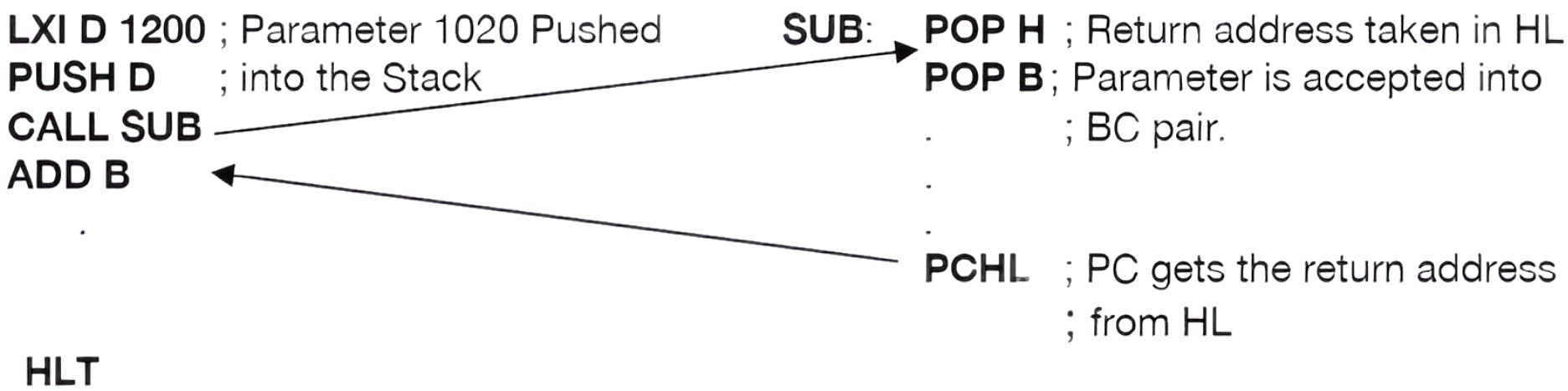
POP PSW



To Pass parameters to a Sub-Routine

The programmer can use the Stack to pass parameters to Sub-Routines.
Before calling the Sub-Routine the **parameter** is **Pushed** into the Stack and then **inside** the **Sub-Routine** the **parameter** is **Popped from** the **Stack**.

Eg:



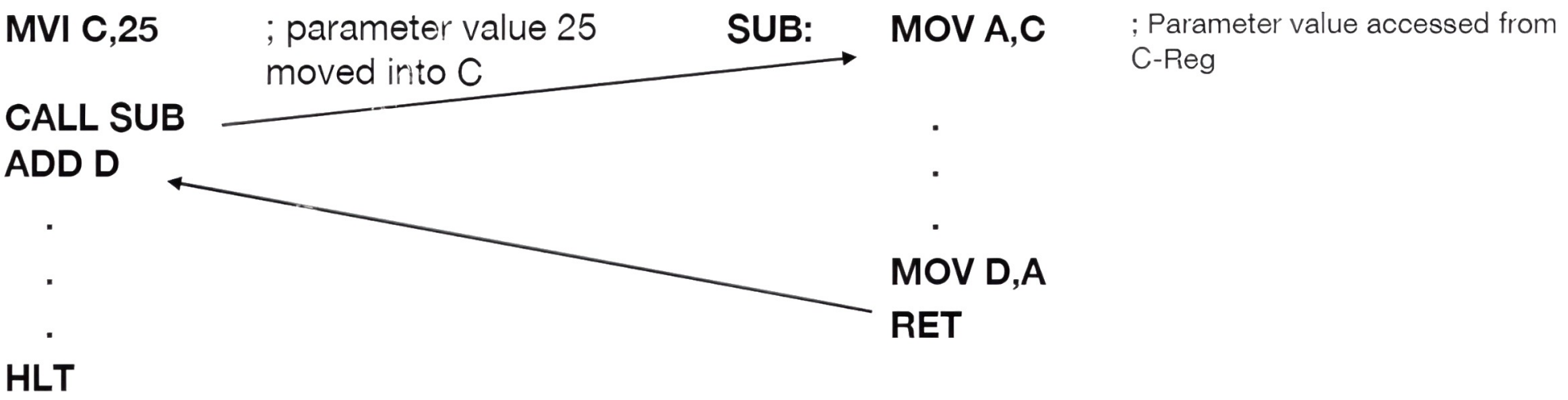
Subroutine

- Subroutines are parts of a program, which can be re-executed by the programmer.
- Subroutines are Called (invoked) using the CALL instruction; control returns to the main program using the RET instruction.
- Subroutines generally perform tasks, which are used regularly by the program such as numerical calculations, Interrupt Service Routines (ISR) etc.
- Subroutines are useful in the following ways:
  - Causes Code Re-Usability hence reduces the size of the Program and also saves time.
  - Makes the program easy to Maintain as it becomes Modular.

Passing parameters to Subroutine

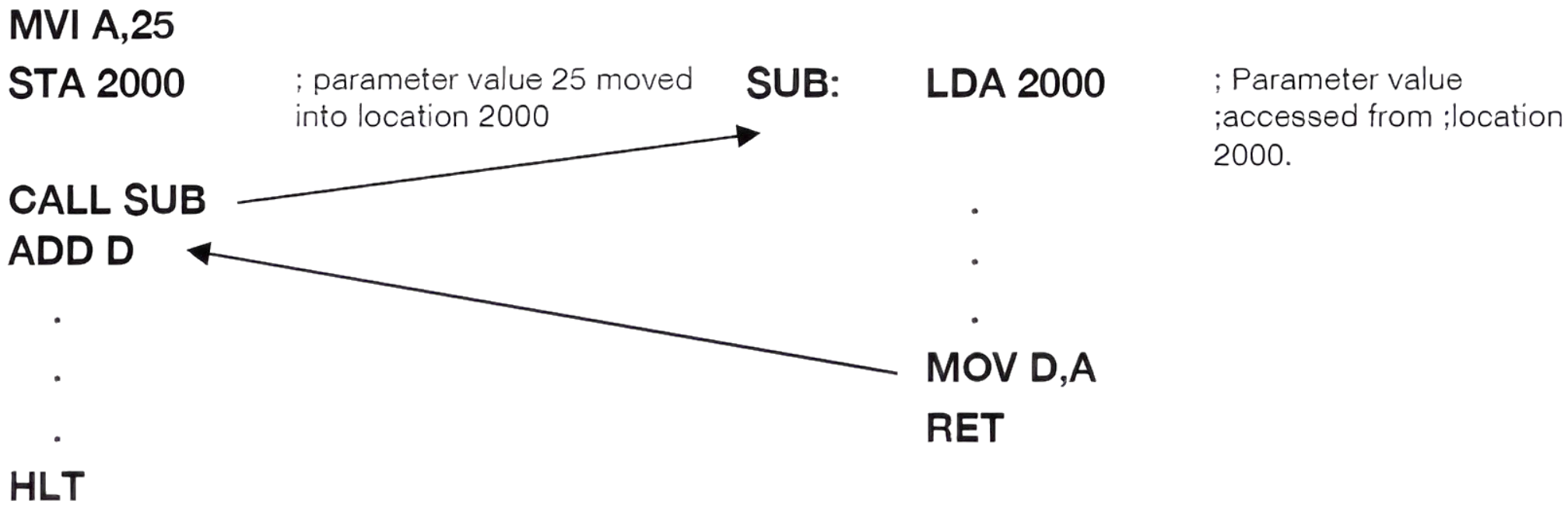
Four methods of passing parameter:

1. Using Register



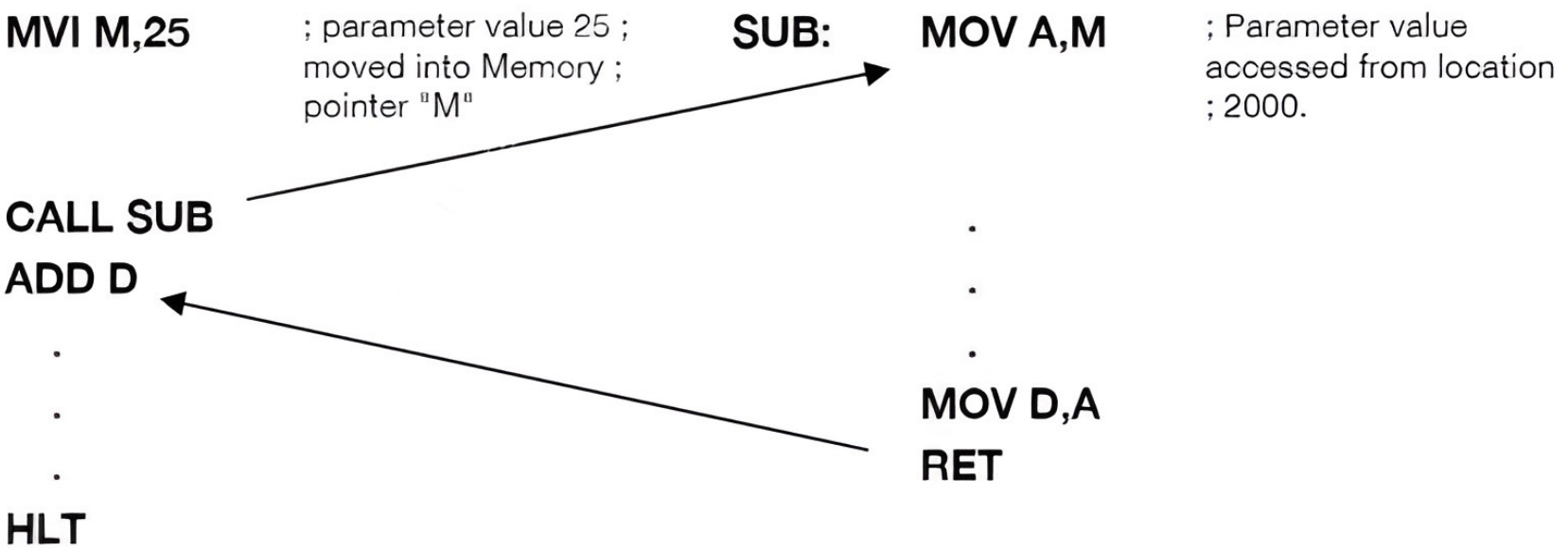
❖ Simplest but occupy a general-purpose register and limited parameter can be pass

2. Using Memory



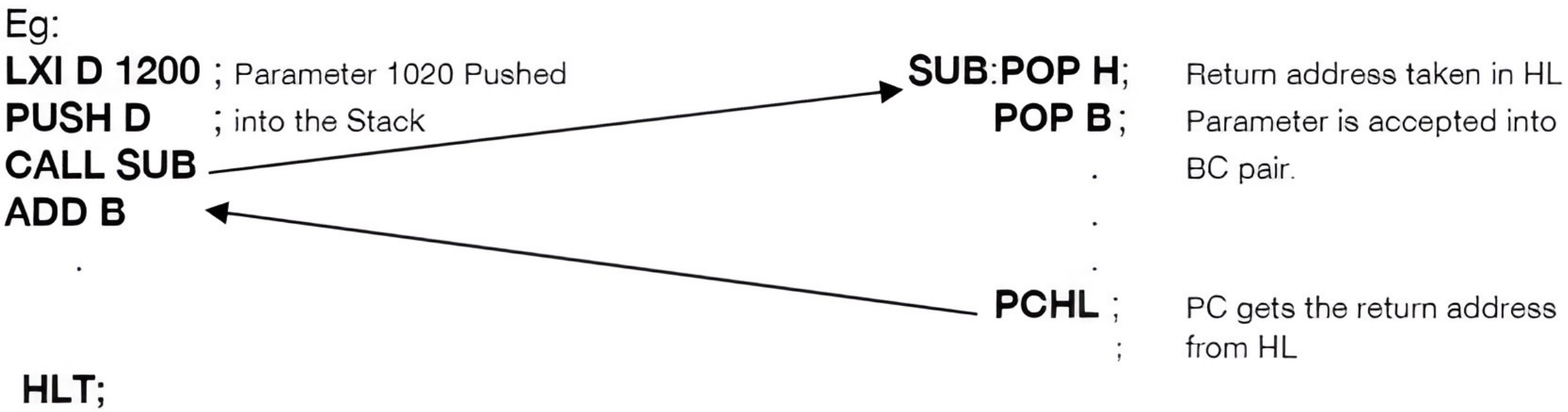
❖ More parameters can be passed **HOWEVER** Programmer needs to remember the memory locations

3. Using Memory Pointer



- Care has to be taken so that HL pair does not change until the Subroutine accesses the value of M.
- If there are more parameters, troublesome method for accessing.

4. Using STACK



- The parameter value to be passed is Pushed into the Stack before calling the SubRoutine.
- The SubRoutine Pops the passed parameter from the Stack.

2. Read the following program and answer the questions given below.

Line No.	Mnemonics
1	LXI SP,0400H
2	LXI B,2055H
3	LXI H,22FFH
4	LXI D,2090H
5	PUSH H
6	PUSH B
7	MOV A,L
↓	↓
20	POP H

a. What is stored in the stack pointer register after the execution of line 1?

b. What is the memory location of the stack where the first data byte will be stored?
c. What is stored in memory location 03FEH when line 5 (PUSH H) is executed?
d. After the execution of line 6 (PUSH B), what is the address in the stack pointer register, and what is stored in stack memory location 03FDH?
e. Specify the contents of register pair HL after the execution of line 20 (POP H).

Q: Explain the functions of the following routines:

(a)LXI SP, 209F H
MVI C, 00 H
PUSH B
POP PSW
RET

(b) LXI SP, STACK
PUSH B
PUSH D
POP B
POP D
RET