

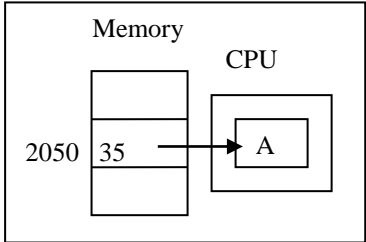
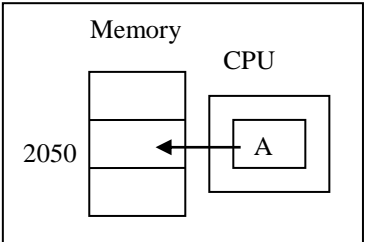
Data Transfer Instructions Related With Memory

There are many ways to transfer data between memory and microprocessor:-

1- Transfer single byte

LDA add:- The content of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator.

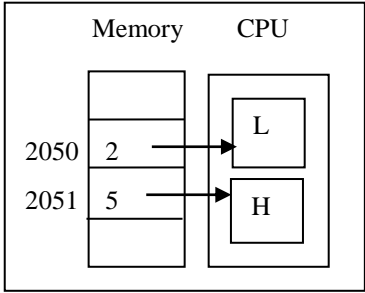
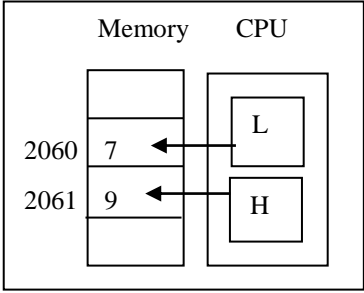
STA add:- The content of accumulator is copied to a memory location specified by the operand.

No	Instruction	Type	No. of Bytes	Function	Effect
1.	LDA add	Data transfer	3	$A=[add]$ Ex. LDA 2050; $a=[2050]=35$	None
					
2.	STA add	Data transfer	3	$[add]=A$ E.g. STA 2050; $[2050]=A$	None
					

2- Transfer two consecutive bytes

LHLD add:- Copies the contents of the memory location pointer out by the 16-bit address in register L and copies the contents of the next memory location in register H.

SHLD add:- The content of register L are stored in the memory location specified by the 16-bit address in the operand and the content of H register are stored in the next memory location.

No	Instruction	Type	No. of Bytes	Function	Effect
1.	LHLD add	Data transfer	3	<p>L=[add], H=[add+1] E.g. LHLD 2050; L=[2050]=2, H=[2051]=5</p> 	None
2.	SHLD add	Data transfer	3	<p>[add]=L, [add+1]=H E.g. SHLD 2060; [2060]=L, [2061]=H</p> 	None

3. Transfer array of data

Many programming applications involve the use of data arrays to organize quantities of input and output data, and an efficient array processing scheme is necessary to be able to access and manipulate this data. Such applications often involve identify, search, replace, sort, analyze and display operations.

Data array processing generally uses a subscript to identify a specific array element. In assembly language programming this subscript is known as an index or pointer. Typically a register or register pair serves as a pointer. The array elements are identified either by reference to their locations in RAM, or by assigning to each element a sequence in a list, the former method requiring a register pair to hold the 16-bits addresses.

Array processing inevitably involves at least one program loop, and loop repetition control is usually associated with the value of the pointer.

3.1 Using register pair BC or DE as pointer

LDAX B/D:- The content of the memory location pointed by a register pair B/D is copied to the accumulator.

STAX B/D:- The content of the accumulator are copied into the memory location specified by the content of the operand (register pair B/D).

No	Inst.	Type	No. of Bytes	Function	Effect
1.	LXI B,add LXI D,add	Data transfer	3	BC =add DE=add	None
2.	LDAX B LDAX D	Data transfer	1	A=[BC] A=[DE	None
3.	STAX B STAX D	Data transfer	1	[BC]= A [DE]=A	None
4.	INX B INX D	Arithmetic	1	BC=BC+1; point to next DE=DE+1 byte	None
5.	DCX B DCX D	Arithmetic	1	BC=BC-1 ; point to DE=DE-1; previous byte	None

3.2 Using register pair HL as pointer

MOV A, M:- The content of the memory location specified by a register pair HL is copied to the accumulator.

MOV M, A:- The content of the accumulator are copied into the memory location specified by the register pair HL.

MOV r, M:- The content of the memory location specified by a register pair HL is copied to the register.

MOV M, r:- The content of the register are copied into the memory location specified by the register pair HL.

MVI M, byte:- The 8-bit data are copied into the memory location specified by the register pair HL.

No	Instruction	Type	No. of Bytes	Function	Effect
1.	LXI H,add	Data transfer	3	HL =add	None
2.	MOV A,M	Data transfer	1	A=M _{HL} =[HL]	None
3.	MOV M,A	Data transfer	1	M _{HL} = [HL]= A	None
4.	INX H	Arithmetic	1	HL=HL+1	None
5.	DCX H	Arithmetic	1	HL=HL -1	None
6.	MOV M, r	Data transfer	1	M _{HL} =[HL]=r	None
7.	MOV r, M	Data transfer	1	r=M _{HL} =[HL]	None
8.	MOV M,M	ERROR			
9.	MVI M, Byte	Data transfer	2	M _{HL} =[HL]=Byte	None

M: Memory location (byte) pointed by HL then M= M_{HL}=[HL]

Example

Reset array of 5 bytes stored in memory started at address 2080

Using DE as pointer	Using HL as pointer
LXI D,2080 ; DE=2080	LXI H,2080 ; HL=2080
MVI C,5 ; C=5 (down-counter)	MVI C,5 ; C=5
NEXT: LDAX D ; A=[DE]	NEXT: MVI M,0 ; [HL]=0
XRA A ; A=0	INX H ; HL=HL+1
STAX D ; [DE]=A	DCR C ; C=C-1
INX D ; DE=DE+1	JNZ NEXT
DCR C ; C=C-1	HLT
JNZ NEXT	
HLT	

- You can simplify the above programs.

Example

For example the following program implements a simple indexing technique to clear an array. The program clears memory location (2070H)-(207FH). Study the action of the register pair (HL) (the index pointer) for each pass through the body of the loop.

```
LXI    H,2070 H    ; Initialize pointer.
More:  MVI    A,00 H    ; Clear accumulator.
       MOV    M,A      ; Store (0) in location pointed by the rp(HL).
       INX   H        ; Point to next location.
       MVI   A,80 H    ; See if past the end of array. (high byte is fixed
                       ; and no. of loops < 256)
       CMP   L        ; By comparing (L) with (70H). (L as up-counter)
       JNZ  More     ; If not done yet, jump back for more.
       RST1          ; Else, quit.
```

The program does not need to check the most significant byte of the index pointer, because it will always be (20H). Notice that the program would stop one location too soon if it compared the contents of register (L) with the number (7FH).

Note: When testing array processing programs that deal with large segments of RAM, it is often better to temporarily modify the program to search through a smaller range of RAM simplify program debugging and result verification.

Class Work

1- Write a program that count the number of data bytes that are equal to (0FH) in a block of 6 bytes stored in memory starting at location (2050H). If all bytes are (0FH) then store (FFH) in memory location (2060H) otherwise store (00H).

Address	HexCode	Label	Opcode	Operands	Comments
2000			LXI	H,2050	; HL=2050
2001	50				
2002	20				
2003			LXI	B,006	; BC=0006
2004	06				
2005	00				
2006			MVI	A,0F	; A=0F
2007	0F				
2008		NEXT:	CMP	M	; A-M _{HL}
2009			JNZ	NOT0F	; if Z=0 then PC=200D
200A	0D				
200B	20				
200C			INR	B	; B=B+1
200D		NOT0F:	INX	H	; HL=HL+1
200E			DCR	C	; C=C-1
200F			JNZ	NEXT	; if Z=0 then PC=2008
2010	08				
2011	20				
2012			MOV	A,B	; A=B
2013			CPI	6	; A-6
2014	06				
2015			JNZ	STORE0	; IF Z=0 then PC=201E
2016	1E				
2017	20				
2018			MVI	A,FF	; A=FF
2019	FF				
201A			STA	2060	; [2060]=A=FF
201B	60				
201C	20				
201D			RST1		; END
201E		STORE0:	XRA	A	; A=A XOR A=0
201F			STA	2060	; [2060]=A=0
2020	60				
2021	20				
2022			RST1		

2- Copy 5 bytes from memory addressed by 2080 to memory addressed by 2090 in sequence.

Address	HexCode	Label	Opcode	Operands	Comments
2000			LXI	H,2080	; HL=2080
2001	80				
2002	20				
2003			LXI	D,2090	; DE=2090
2004	90				
2005	20				
2006			MVI	C,5	; C=5
2007	05				
2008		COPY:	MOV	A,M	; A=M _{HL} =[HL]
2009			STAX	D	; [DE]=A
200A			INX	H	; HL=HL+1
200B			INX	D	; DE=DE+1
200C			DCR	C	; C=C-1
200D			JNZ	COPY	; if Z=0 then pc=2008
200E	08				
200F	20				
2010			RST1		; End

Homework

1. Move the block of data (16-bytes) stored in locations (2070H – 207FH) into locations (2080H – 208FH) in reverse order.
2. Write a program to count the number of data elements in a data set starting with address (20A0H) and ended with value of (00H)
3. Make reverse of 10 numbers stored in memory started at 2070 (exchange 1st no. with 10th no., 2nd no. with 9th no., and so on)
4. 10 numbers stored in memory started at location 2070, insert value 6 after 3rd no. to be the 4th no. (Total no. becomes equal to 11)
5. Find the repetition of value 5 between 10 numbers stored in memory started at 2070
6. Specify the contents of memory location (2070H) to (2074H) after executing the following instructions:

```

                LXI    H,2070 H
                MVI    B,05 H
                MVI    A,01 H
Store:  MOV     M,A
                INR    A
                INX    H
                DCR    B
                JNZ    Store
                RST1

```

Notes

- 1- M_{HL} treated as any 8-bit register
- 2- Using HL as pointer is preferred than using BC or DE as pointer.
- 3- May be we need to use many pointers like HL and BC
- 4- Use down-counter or up-counter as described in the previous lab (Comparison and JUMP instructions), and we can use the low byte of the pointer as up-counter if the high byte of address is fixed with all bytes (no. of bytes < 256) to check the reaching of last location.