Birla Institute of Technology and Science, Pilani – KK Birla Goa Campus

EEE/ECE/INSTR F241

Lab 1 Solutions

---

1. Write an ALP to find the largest number in an array of ten 8-bit numbers stored in memory starting from address 2000H

Solution:

Logic: We will use a loop to compare element by element. C register is used for storing the loop control variable. Current largest number will be stored in the accumulator (A). It is initialized with the first element of the array. HL pair will be used as the pointer to access the memory and compare with the current largest number. If the element in memory is larger than the value in A, value in A is replaced with the element in the memory. We do this comparison 9 times. At the end of the loop, largest number will be in A

```
MVI C,9H        ;C is used as the loop control counter
LXI H,2000H     ;Initialize HL pair to point to the memory where
array is stored
MOV A,M         ;store the first number in A
loop:
     INX H      ;Point to the next number
     CMP M      ;Compare number in A with number in memory
     JNC label  ;If number in A is larger, then go to label
     MOV A,M    ;else copy the number from memory to A.
     label:
     DCR C      ;Decrement loop control variable
     JNZ loop   ;If C becomes 0, exit the loop, else continue
HLT             ;Halt the processor
```

2. Write an ALP to store 2 eight-bit numbers at location 0x1000 and 0x1001. Find their product and store the result from location 0x2000

Solution:

Logic: In 8085, there is no single instruction to multiply numbers. Hence multiplication needs to be implemented by continuous addition. When we multiply two 8 bit numbers, the result could be 16 bits. Hence two registers (A and B) are used to store the result. A will be used to store the lower byte of result and B to store the upper bye. Thus initially both registers need to be initialized to 0. C register is loaded with the multiplicand and used as a loop control variable. Then the multiplier is continuously added to accumulator. If the addition generates a carry, B register is incremented (since it is representing the upper byte). Once the multiplier is added multiplicand times, A register will have lower byte of the product and B register the upper byte. They are then stored in addresses 2000H and 2001H

```
MVI A,0H   ;Clear accumulator
MVI B,0H   ;Upper byte of result
LXI H,1000H
MOV C,M    ;Store multiplicand in C and use as a loop variable
INX H      ;Increment HL to point to multiplier
loop:
     ADD M ;Add multiplier to accumulator
     JNC Next
     INR B ;if there is carry increment upper byte
     Next:
     DCR C ;Decrement loop control variable. Once it exhaust,
exist
     JNZ loop
STA 2000H ;store lower Byte in memory 2000H
          MOV A,B
          STA 2001H ;store upper Byte in memory 2001H
          HLT
```

3. Write an ALP to add the numbers 0x1BC and 0x221 and store the result in memory from address 0x2000

Solution:

Logic: First add the lower bytes and store the result in memory. Then add the upper bytes. While adding upper bytes need to also add the carry generated during the addition of lower bytes.

```
MVI A,BCH
ADI 21H ;Add the lower bytes first
STA 2000H;Store the result.
MVI A,1; Remember flags are not affected by data movement
instructions
ACI 2;Add the upper bytes along with carry of previous addition
STA 2001H
HLT
```

4. Write an ALP to sort an array of ten 8-bit numbers in ascending order which are stored in memory starting from address 2000H

Solution:

Logic: Any sorting algorithm can be used for this purpose. This example is using bubble sort (https://www.geeksforgeeks.org/bubble-sort/). As you can see, the algorithm requires a loop inside another loop. Here we use C register as the outer loop control variable and D as inner loop control variable. After each iteration of the inner loop, position of one number gets fixed (after first iteration we have largest number as the last element. After second iteration, largest and second largest are available so on and so forth). So, the inner loop needs to run one iteration less the outer loop variable

```
MVI C,AH           ;Outer loop variable initialized to 10 (# of
elements)
DCR C
outer_loop:
    MOV D,C        ;Inner loop variable 1 less than outer
    LXI H,2000H    ;Each timer inner loop starts from beginning
of array
    inner_loop:
     MOV A,M       ;Take element from the array
     INX H         ;Point to next element in memory
     CMP M         ;Compare the elements
     JC NO_SWAP    ;If element in memory is larger, no swapping,
else, swap
     MOV B,M       ;swapping
     MOV M,A
     DCX H
     MOV M,B
     INX H
     NO_SWAP:
     DCR D         ;Decrement inner loop variable
     JNZ inner_loop ;If loop variable is 0, exit loop
DCR C              ;Decrement outer loop variable
JNZ outer_loop     ;If loop variable is 0, exit loop
HLT
```

5. Write an ALP to check whether a given number is present in an array. The number to be checked is stored in memory address 1000H and the array is stored starting from address 2000H. The size of the array is stored in address 1001H. If the element is present, store 1H in address 1002H else store 0.

Solution:

Logic: Here we use a loop to compare the elements. C register is used as the loop control variable. The size of the array is in 1000H, hence that is initially stored in C register. The number to be compared is loaded to A register from memory. In the using HL pair we point to the elements of the array and compare with A. As soon as a match if found, 1 is stored in 1002H and program is halted. If we exit the loop means number was not found and in this case 0 is stored at 1002H.

```
LXI H,1000H
MOV A,M  ;get the element to be compared
INX H
MOV C,M  ;get the array size
LXI H,2000H
```

```
loop:
    CMP M          ;Compare the element with memory element
    JNZ noMatch
    MVI A,1        ;If match, store 1 and halt program
    STA 1002H
    HLT
    noMatch:
    INX H
    DCR C
    JNZ loop
MVI A,0             ;Program exited the loop means number was
not found. Store 0 in 1002H and halt the program
STA 1002H
HLT
```

6. Write an ALP to implement the modulus operation. The dividend and divisor and 8-bit numbers. Dividend is stored at address 1000H and divisor at 1001H. Store the result of modulus at address 2000H

Solution

Logic:

8085 doesn't have an instruction to find modulus. Similar to division, modulus needs to be found by continuous subtraction. Before subtraction we can compare dividend with divisor. If dividend is smaller, that itself is the modulus. Similarly if dividend is 0, modulus is 0. Otherwise subtract divisor from dividend and continue to compare.

```
LXI H,1000H
MOV A,M                  ;load the dividend to A
INX H                    ;point to divisor
loop:
     CMP B
     JNZ checkAisLarge   ;If A and [M] are same, modulus is 0
     MVI A,0
     STA 2000H           ;store the result and halt
     HLT
checkAisLarge:
JNC AisLarge             ;If A and [M] are not same, and if A is
smaller, then that is the modulus
STA 2000H                ;Store the result and halt
HLT
AisLarge:
SUB M                    ;If A is larger, subtract divisor and
continue
JMP loop
```