8085 PROGRAMS SCHOOL

Rupali & Tejas Shah's Class

DECEMBER 25, 2018

PAALBI

Contents

1 Addition of two binary numbers

```
LXI H,7050 //Initialize memory pointer to 7050
MOV A, M //Read data to accumulator
ADI 05 //Add 05 to contents of accumulator
INX H //Increment memory pointer
MOV M, A //Store result to memory
HLT
```

2 Addition of two 8-bit numbers having 16 bits sum

```
//First number is stored in the memory location 7050
//Second number is stored in the memory location 7051
//Result is stored in memory locations 7052 (LSB) & 7053 (MSB)
        LXI H,7050 //Initialize memory pointer t0 7050
```

```
        MVI C,00 //Initialize C to 00. C will store the MSB (carry) of the addition.
        MOV A, M //Copy the first number to the accumulator
        INX H //Increment the memory pointer
        ADD M //Add the number stored in memory to the contents of the accumulator
        JNC AHEAD //If there is carry then increment C
        INR C
AHEAD: STA 7052 //Store the contents of the accumulator in memory location 7052
        MOV A, C //Copy content of C to accumulator
        STA 7053 //Store the contents of the accumulator in memory location 7053
        HLT
```

3 Addition of a block of numbers

```
//Size of the block is stored in memory location 7050 (Assume size to be 10)
//The block itself is stored starting from memory location 7051
//Result is stored in memory locations immediately after the block (LSB, MSB)
XRA A //XOR accumulator with itself to reset the accumulator
MOV B, A //Initialize B to 00. B will be used to store MSB
LXI H,7050 //Initialize memory pointer to 7050
MOV C, M //Copy contents of the memory location 7050 to C (size of block used as counter)
REPEAT:INX H //Increment memory pointer
ADD M //Add the number stored in memory to the contents of the accumulator
JNC AHEAD //If there is carry then increment B
INR B
AHEAD: DCR C //Decrement the counter by 1
JNZ REPEAT //Repeat the addition if counter is not equal to 0
INX H //Increment the memory pointer
MOV M, A //Store the LSB of the result
INX H //Increment the memory pointer
MOV M, B //Store the MSB of the result
HLT
```

4 Addition of two 4-byte numbers

```
STC //Set carry flag 1
CMC //Complement carry flag (set carry flag to 0)
LXI H,7050 //Starting address of first number
LXI D,7060 //Starting address of second number
LXI B,7070 //Starting address of answer
LDA 704F //Read size of the numbers (4 bytes) as counter repeat:
STA 704F //Store the size of the number
LDAX D //Load 1 byte of second number in the accumulator
ADC M //Add 1 byte of the first number with the data in accumulator
STAX B //Store the result of the addition
INX H //Point the next byte of first number
INX D //Point to the next byte of second number
INX B
LDA 704F //Read the size of the numbers
DCR A //Decrement size by 1 as 1 byte is added
JNZ repeat //Repeat addition till all bytes have been added
JNC skip //See if carry was generated while adding the 4th byte
```

```
MVI A,01 //Load 01 in accumulator STAX
B //Store the carry as the 5th byte skip:
HLT
```

5 Addition of two 8-bit BCD numbers

```
//First number is stored in the memory location 7050
//Second number is stored in the memory location 7051
//Result is stored in memory locations 7052
LXI H,7050 //Initialize memory pointer t0 7050
MVI C,00 //Initialize C to 00. C will store the MSB (carry) of the addition.
MOV A, M //Copy the first number to the accumulator
INX H //Increment the memory pointer
ADD M //Add the number stored in memory to the contents of the accumulator
DAA //Decimal adjust accumulator
JNC AHEAD
INR C //If there is carry then increment C
AHEAD: INX H
MOV M, A //Store the LSB of the result
INX H //Increment the memory pointer
MOV M, C //Store the MSB of the result
HLT
```

6 Addition of a block of BCD numbers

```
//Size of the block is stored in memory location 7050 (Assume size to be 10)
//The block itself is stored starting from memory location 7051
//Result is stored in memory locations immediately after the block (LSB, MSB)
XRA A //XOR accumulator with itself to reset the accumulator
MOV B, A //Initialize B to 00. B will be used to store MSB
LXI H,7050 //Initialize memory pointer to 7050
MOV C, M //Copy contents of the memory location 7050 to C (size of block used as counter)
REPEAT:INX H //Increment memory pointer
ADD M //Add the number stored in memory to the contents of the accumulator
JNC AHEAD //If there is carry then increment B
INR B
AHEAD: DAA //Decimal adjust accumulator
DCR C //Decrement the counter by 1
JNZ REPEAT //Repeat the addition if counter is not equal to 0
INX H //Increment the memory pointer
MOV M, A //Store the LSB of the result
INX H //Increment the memory pointer
MOV A, B //Copy the result of the carry (MSB) to accumulator
DAA //Decimal adjust accumulator
MOV M, A //Store the MSB of the result
HLT
```

7 Addition of a block of data using DAD

```
//Size of the block is stored in memory location 7050 (Assume size to be 10)
//The block itself is stored starting from memory location 7051
```

```
XRA A //XOR accumulator with itself to reset the accumulator
LXI B,00 //Set contents of register B and C to 00
LXI D,00 //Set contents of register D and E to 00
LXI H,7050 //Initialize memory pointer t0 7050
MOV A, M //Read the block size from 7050 and store in accumulator
REPEAT:INX H //Increment the memory pointer
MOV C, M //Copy the number from memory
XCHG //Exchange contents of DE pair (result) with HL pair (memory pointer)
DAD B //Add contents of BE to HL (16-bit addition)
XCHG // Exchange contents of DE pair (result) with HL pair (memory pointer)
DCR A //Decrement counter by 1
JNZ REPEAT //Repeat the addition if counter is not equal to 0
INX H //Increment the memory pointer
MOV M, E //Store the LSB of the answer to memory
INX H //Increment the memory pointer
MOV M, D //Store the MSB of the answer to memory
HLT
```

8 Subtract two numbers and store the absolute difference

```
LXI H,7050 //Initialize memory pointer
MOV A, M //Load a number in accumulator
INX H
SUB M //Subtract number in memory from Number in accumulator
JC ahead //If there is borrow then take 2's complement
CMA
ADI 01 ahead:INX
H
MOV M, A //Store the result
HLT
```

9 Multiplication of two 8-bit numbers

```
//First number is stored in the memory location 7050
//Second number is stored in the memory location 7051
//Result is stored in memory locations 7052 (LSB) & 7053 (MSB)
XRA A //XOR accumulator with itself to reset the accumulator
MOV B, A //Initialize B to 00. B will be used to store MSB
LXI H,7050 //Initialize memory pointer to 7050
MOV C, M //Copy contents of the memory location 7050 to C (first number used as counter)
INX H //Increment the memory pointer
REPEAT: ADD M //Add the second number to the contents of the accumulator
JNC AHEAD //If there is carry then increment B
INR B
AHEAD: DCR C //Decrement counter by 1
JNZ REPEAT //Repeat the addition if counter is not equal to 0
INX H //Increment the memory pointer
MOV M, A //Store the LSB of the result
INX H //Increment the memory pointer
MOV M, B //Store the MSB of the result
```

```
HLT
```

10 Divide two 8-bit numbers

```
//First number is stored in the memory location 7050
//Second number is stored in the memory location 7051
//Result is stored in memory locations 7052 (quotient) & 7053 (remainder)
XRA A //XOR accumulator with itself to reset the accumulator
MOV C, A //Initialize C to 00. C will be used to store quotient
LXI H,7050 //Initialize memory pointer to 7050
MOV A, M //Copy the first number (dividend) to the accumulator
INX H //Increment the memory pointer
REPEAT:CMP M //Compare the second number with the number (divisor) in the accumulator
JC STOP //If the number in accumulator is < second number (divisor) then go to stop
SUB M //Subtract divisor from contents of the accumulator
INR C //Increment the counter (quotient)
JMP REPEAT //Repeat the loop
STOP:INX H //Increment the memory pointer
MOV M, C //Store the quotient in memory location 7052
INX H //Increment the memory pointer
MOV M, A //Store the remainder in memory location 7053
HLT
```

11 Separate the nibbles of an 8-bit number and multiply them

```
//Number is stored in memory location 7050
LXI H,7050 //Initialize memory pointer to 7050
MOV A, M //Load the number in the accumulator
MOV B, M //Load the number in the register B
ANI 0F //AND the content of the accumulator with 0F to get the first 4 bits
INX H //Increment the memory pointer
MOV M, A //Store the first 4 bits in memory location 7051
MOV A, B //Load the number in the accumulator
RRC //Rotate data in the accumulator right by 1
RRC //Rotate data in the accumulator right by 1
RRC //Rotate data in the accumulator right by 1
RRC //Rotate data in the accumulator right by 1
ANI 0F //AND the content of the accumulator with 0F to 4 MSB of the number
INX H //Increment the memory pointer
MOV M, A //Store the 4 MSBs in memory location 7052
DCX H //Decrement memory pointer to point to the first nibble
XRA A //XOR contents of accumulator with itself to set A to 0
MOV B, A //Initialize B to 00. B will be used to store MSB
MOV C, M //Copy the first number to register C and use it as a counter
INX H //Increment the memory pointer
REPEAT: ADD M //Add the second number to the contents of the accumulator
JNC AHEAD //If there is carry then increment B
INR B
AHEAD: DCR C //Decrement counter by 1
JNZ REPEAT //Repeat the addition if counter is not equal to 0
INX H //Increment the memory pointer
```

```
MOV M, A //Store the LSB of the result
INX H //Increment the memory pointer
MOV M, B //Store the MSB of the result
HLT
```

12 Program to find the first occurrence of a number in a given block

```
//Number to be found is stored in memory location 7050
//Size of the block is stored in memory location 7051 (Assume size to be 10)
//The block itself is stored starting from memory location 7052
MVI D,00 //Initialize register D to 00. It will be used to store the result (01 – if found)
LXI H,7050 //Initialize memory pointer to 7050
MOV A, M //Copy the number to be searched to register B
INX H //Increment the memory pointer
MOV C, M //Copy contents of the memory location 7051 to C (size of block used as counter)
REPEAT:INX H //Increment the memory pointer
CMP M //Compare the nth number with the number in the accumulator JNZ
NOTFOUND //Set register D to 01 only if the number is found.
MVI D,01
JMP STOP
NOTFOUND: DCR C //Decrement counter by 1
JNZ REPEAT //Repeat the comparison if counter is not equal to 0
LXI H, FFFF //Set HL to FFFF in case the number is not found
STOP: HLT
```

13 Find the number of times a number occurs in a given block

```
//Number to be found is stored in memory location 7050
//Size of the block is stored in memory location 7051 (Assume size to be 10)
//The block itself is stored starting from memory location 7052
//Register D is used to store the result
MVI D,00 //Initialize register D to 00. It will be used to store the result (01 – if found)
LXI H,7050 //Initialize memory pointer to 7050
MOV A, M //Copy the number to be searched to register B
INX H //Increment the memory pointer
MOV C, M //Copy contents of the memory location 7051 to C (size of block used as counter)
REPEAT:INX H //Increment the memory pointer
CMP M //Compare the nth number with the number in the accumulator
JNZ NOTFOUND //Increment register D by 1 when the number if found
INR D
NOTFOUND: DCR C //Decrement counter by 1
JNZ REPEAT //Repeat the comparison if counter is not equal to 0
STOP: HLT
```

14 Find the number of even numbers and the number of odd numbers in a given block of numbers

```
//Size of the block is stored in memory location 7050 (Assume size to be 10)
//The block itself is stored starting from memory location 7051
//Store the result immediately after the block
```

```
XRA A //XOR contents of accumulator with itself to reset it
MOV D, A //Set register D to 0
MOV E, A //Set register E to 0
LXI H,7050 //Initialize memory pointer t0 7050
MOV C, M //Copy contents of the memory location 7050 to C (size of block used as counter)
REPEAT:INX H //Increment the memory pointer
MOV A, M //Read one number to the accumulator
RRC //Rotate accumulator right
JNC EVEN //If carry flag is not set then number was even else odd
INR D //Increment count of odd numbers
JMP AHEAD //Jump ahead and skip incrementing count of even numbers
EVEN: INR E //Increment count of even numbers
AHEAD: DCR C //Decrement counter by 1
JNZ REPEAT //Repeat the loop if counter value is not 0
INX H //Increment memory pointer
MOV M, D //Store count of odd numbers found in memory
INX H //Increment counter by 1
MOV M, E //Store count of even numbers found in memory
HLT
```

15 Program to exchange a block of memory

```
//Size of both blocks is 05.
//Starting address of the first block is 7050
//Starting address of the second block is 7060
LXI H,7050 //Initialize HL pair to 7050 (starting address of first block)
LXI D,7060 //Initialize DE pair to 7060 (starting address of second block)
MVI C,05 //Store 05 (size of block) register C to use as counter
REPEAT:MOV B, M //Copy 1 byte from first block to register B
LDAX D //Copy 1 byte from second block to accumulator
MOV M, A //Copy contents of accumulator (1 byte from second block) to first block
MOV A, B //Copy data from register B to accumulator
STAX D //Copy data from accumulator (1 byte from the first block) to second block
DCR C //Decrement counter by 1
INX H //Increment HL pair
INX D //Increment DE pair
JNZ REPEAT //Repeat the logic to swap if counter is not 0
HLTM
```

16 Program to transfer a block of data in reverse order

```
//Size of the block is stored in memory location 7050 (Assume size to be 10)
//The block itself is stored starting from memory location 7051
//Transfer the block to memory address starting from 7061
        LXI H,7050 //Initialize memory pointer to 7050
        MOV C, M //Read the size of block in register C
        MVI B,00 //Set contents of register B to 00
        DAD B //Add contents of BC to HL to get the address of the last number in the block
        XCHG //Exchange the contents of DE and HL
        LXI H,7061 //Initialize memory pointer to 7061 (address of target)
```

REPEAT: LDAX D //Read one byte of data from source starting with the last number in the accumulator
MOV M, A //Copy the contents of the accumulator to memory (last number stored in first location)
DCX D //Decrement DE pair
INX H //Increment HL pair
DCR C //Decrement counter by 1
JNZ REPEAT //If counter is not equal to 0 then transfer one more number
HLT

17 Reverse a block in place

```
//Assume block size = 10 from 7050 to 7059
MVI C,05
LXI H, 7050 LXI
D, 7059
repeat: LDAX D //Read the last no to A and B
MOV B, A
MOV A, M //Read value from M to A
STAX D //Store value of A to DE (1st no in last location)
MOV M, B //Store value of B to HL (last no in first location)
INX H
DCX D
DCR C //Decrement counter
JNZ repeat
HLT
```

18 Count the number of zeros in a given 8-bit number

```
//Number is stored in register C
LXI H,7050      //Initialize memory pointer to 7050
MOV C, M        //Copy the number to be checked to register C
MVI B,00        //Initialize register to 00. This will keep count of the number of 0's found in the number
INX H           //Increment the memory pointer
MOV A, M        //Copy the number that needs to be checked to the accumulator
REPEAT:         RRC             //Rotate data in the accumulator right by 1
JC AHEAD        //If carry flag is set then increment count in register B
INR B
AHEAD: DCR C            //Decrement counter by 1
JNZ REPEAT      //Repeat the rotate and if counter is not equal to 0
INX H           //Increment the memory pointer
MOV M, B        //Store the number of 0's found to memory
HLT
```

19 Program to sort a block of numbers using bubble sort

```
//Size of the block is stored in memory location 7050 (Assume size to be 10)
//The block itself is stored starting from memory location 7051
LXI H,7050      //Initialize memory pointer to 7050
MOV C, M        //Read the size of block in register C
DCR C           //Decrement counter by 1
LOOP1: LXI H,7051       //Set memory pointer to the starting of the array
MOV D, C        //Copy outer counter to inner counter
LOOP2: MOV A, M         //Read nth element to A and B
MOV B, M
```

```
INX H          //Increment memory pointer to point to n+1
CMP M          //Compare A (nth number) with M (n+1th number)
JC skip        //If 1st number < 2nd number then skip
MOV A, M       //Swap
MOV M, B
DCX H
MOV M, A
INX H
SKIP:   DCR D          //Decrement inner loop
JNZ LOOP2
DCR C          //Decrement outer loop
JNZ LOOP1
HLT
```

20 Check if given 4-byte hex number is a palindrome

```
//Number is stored from 7050 to 7053
//7060 is set to FF if it is not a palindrome and to 00 if it is a palindrome.
//Example AB0550BA is a palindrome.
XRA A          //XOR contents of accumulator with itself to reset it
STA 7060       //Assuming that the data is a palindrome store 00 in 7060
LXI H,7050     //Initialize memory pointer to 7050 (first address of 4-byte number)
LXI D,7053     //Initialize DE pair to 7053 (last address of 4-byte number)
MVI C,02       //Set counter value to 02
REPEAT:        MOV A, M        //Copy contents of memory to accumulator
RRC            //Rotate accumulator right 4 times to reverse the number
RRC
RRC
RRC
MOV B, A       //Store the rotated number in register B
LDAX D         //Load one byte from the end
CMP B          //Compare nth byte with (last-n+1) th byte
JZ AHEAD       //If the numbers do not match then it is not a palindrome. Set 7060 to FF
MVI A, FF
STA 7060
JMP STOP
AHEAD :INX H           //Increment HL pair
DCX D          //Decrement DE pair
DCR C          //Decrement counter
JNZ REPEAT     //If counter is not 0 then repeat the comparison for the 2nd and 2nd last number
STOP: HLT
```

21 Program to display flags using stack

```
LXI SP,7050    //Initialize stack pointer to 7050
LXI B,0000     //Initialize BC to 0000
PUSH B         //Push 0000 to stack
POP PSW        //Pop stack to PSW setting accumulator and flags to 00
MVI A,00       //Move 00 to accumulator
PUSH PSW       //Push PSW to stack (flags are not affected)
XRA A          //XOR contents of accumulator with itself setting the zero and parity flags
```

PUSH PSW HLT //Push the PSW to stack

22 Convert number stored in ASCII to its binary equivalent

```
LXI H,7050 //Initialize memory pointer to 7050
MOV A, M //Read number to convert
SUI 30 //Subtract 30 from ASCII number
CPI 0A //If the ASCII code is for numbers from A-F then subtract 7
JC NUM1
SUI 07
NUM1: MOV B, A //Store the first nibble
INX H //Increment the memory pointer and read next number and repeat above process.  MOV
A, M
SUI 30
CPI 0A
JC NUM2
SUI 07
NUM2: //The second nibble represents the higher 4 bits. Hence rotate 4 times.
RRC
RRC
RRC
RRC
ADD B //Add lower nibble with higher nibble.
STA 7052 //Store the answer in 7052.
HLT
```

23 Convert a binary number to its ASCII equivalent

```
LXI H,7050 //Initialize memory pointer.
MOV A, M //Read number to convert and store in A and B.
MOV B, A
ANI 0F //Separate lower nibble.
CPI 0A //If nibble is between A and F then add 07.
JC NUM1
ADI 07
NUM1: ADI 30 //Add 30 to get ASCII equivalent.
INX H //Store the ASCII equivalent of lower nibble.
MOV M, A
MOV A, B //Repeat process for higher nibble.
RRC
RRC
RRC
RRC
ANI 0F
CPI 0A
JC NUM2
ADI 07
NUM2: ADI 30
INX H
MOV M, A
HLT
```