Maulana Azad National Institute of Technology

(An Institute of National Importance)

Bhopal – 462003 (India)

Department of Computer Science & Engineering

ASSIGNMENT SUBMISSION

Microprocessors Lab. (CSE-316)

Submitted by : Jishan Shaikh (Scholar no. 161112013)

Submitted to : Prof. Sanyam Shukla
Department of Computer Science & Engg.
MANIT, Bhopal (India)

Dated by : November 2, 2018 (Friday)

Subject : Microprocessors Lab
CSE-316, V Sem. (B.Tech. in CSE)

Session : Odd Semester 2018

This page intentionally left blank

Index of Programs

8085 Programs

1. Addition of Two numbers

```
# ORG 7000H

LXI     H,7501                  // Get address of 1st no. in HL pair
MOV     A,M                     // Move no. into accumulator
INX     H                       // HL points the address 7502 H
ADD     M                       // Add the 2nd no.
INX     H                       // HL points 7503 H
MOV     M,A                     // Store result in 7503 H
RST 1                           // Terminate

# ORG 7501H                     // Store input at the address
# DB 12H, 13H                   // Get two 8 bit no. in successive location
```

```
# ORG 7000H
LHLD    7601                //Get  1st no. in HL pair from memory 7601 H
XCHG                        //Exchange cont. of DE       HL
LHLD    7603                //Get  2st no. in HL pair from location 7603 H
MVI     C,00                //Clear reg. C.
DAD     D                   //Get HL+DE & store result in HL
JNC     down                //If no carry move to loop/if carry then move to
next step.
INR     C                   //Increment reg.C
MOV     A,C                 //Move carry from reg. C to reg. A
STA     7502                //Store carry at 7502 H
down: SHLD 7500             //Store result in 7500 H.
RST 1                       //Terminate

#ORG 7601H                  // Store input at the address
#DB 13,31,12,10             // Get two 16 bit no. in successive location
```

2. Subtraction of 2 numbers

```
# ORG 7000H
LXI     H, 7501             // Get address of ist no. in HL pair
MOV     A, M                // Move no. into accumulator
INX     H                   // HL points 7502 H.
SBB     M                   // Substract 2nd no. from Ist no.
INX     H                   //HL points 7503 H.
MOV     M, A                // Move contents of acc. to memory
RST 1                                // Terminate

#ORG 7501H                  // Store no. at address
#DB 20,10                   // Get the two 8 bit no. at successive location
```

```
# ORG 7000H

LHLD    7501                // Get 1st 16 bit no. in HL pair
XCHG                        // Exchange HL pair with DE.
LHLD    7503                // Get 2nd 16 bit no. in HL pair
MOV     A, E                // Get lower byte of ist no.
SUB     L                   // Subtract lower byte of 2nd no.
MOV     L, A                // Store the result in reg. L
MOV     A, D                // Get higher byte of Ist no.
SBB     H                   // Subtract higher byte of 2nd no. with borrow
MOV     H,A                 // Move from acc. To H
SHLD    7505                // Store 16 bit result at 7505 H &7506 H
RST 1                       // Terminate

# ORG 7501H                 // Store inputs at the address
# DB 30,40,10,20            // Get two 16 bit no. from successive locations
```

3. Division of 2 Numbers

```
# ORG 7000H

LDA      7501       // [7501]=>A (Divisor)
MOV     B,A         // Take divisor in reg,B
LDA     7502        // Take dividend in reg,A
MVI     C,00        // Quotient=00
CMP     B           // Compare A to B
JC      down        // Jump if carry
up:SUB  B           // Dividend-divisor=>A
INR     C           // C=C+1
CMP     B           // Is dividend < divisor
JNC     up          // If not,go back
down:STA7503        // Store Remainder
MOV     A,C         // C=>A
STA     7504        // Store Quotient
RST 1               // Terminate

# ORG 7501H         // Store the inputs at the address
# DB 06,26          // Get the numbers from successive loc.
```

Multiplication of 2 Numbers

```
# ORG 7000H
LHLD    7501         // Get Multiplicand in H-L pair.
XCHG                 // Exchange HL pair with DE pair
LDA     7503         // Get  2nd  no. in acc.
LXI     H,0000       // Initial product in HL=00
MVI     C,08         // Count=08 in reg .C
up:DAD  H            // Shift partial product left by 1 bit
RAL                  // Rotate multi. by 1 bit. Is multiplier = 1?
JNC     down         // No, go to ahead
DAD     D            // Product=Product + Multiplicand
down:DCRC            // Decrement Count
JNZ     up           // Jump until C=0
```

```
SHLD    7504            // Store result
RST 1                   // Terminate

#ORG 7501H              // Store inputs at the address
# DB 25,00,05           // Get the numbers from successive locations
```

4. Finding Sqrt of a number

```
# ORG 2000H
MVI     C,01           // Place 01 in reg.C
MVI     B,01           // Place odd number 1 in reg.B
MVI     A,24           // Load accumulator with the given number
up:SUB  B              // Subtract odd number from the accumulator
JZ      down           // If accumulator contents are zero, go to Ahead
INR     C              // Increment reg. C
INR     B              // Increment odd number
INR     B              // Increment odd number
JMP     up             // Repeat subtraction
down:MOVA,C            // Move the contents of C  to  A
STA     2050           // Store the result in the memory location 2050H.
RST 1                  // Stop
```

5. Moving a block of data

```
# ORG 2000H
MVI     D,06                   // Place 06 in reg.D
LXI     H, F100                // Block starting address into HL
LXI     B, F200                // Destination address into BC
up:MOV  A,M                    // [HL]=>A
STAX    B                      // A=> [BC]
INX     H                      // Increment HL pair content
INX     B                      // Increment BC pair content
DCR     D                      // Decrement reg.D by 1
JNZ     up                     // Jump until D=0
RST 1                          // Terminate
```

6. Sorting of integers

```
# ORG 2000H
LDA     F100              // Load count from F100 to Acc.
DCR     A                 // Decrement A by 1
MOV     C,A               // A=>C
MOV     B,C               // C=>B
LXI     H,F200            // HL <= F200
up:MOV  A,M               // [HL] =>A
INX     H                 // HL+1=>HL
CMP     M                 // Compare reg. M to A
JC      down              // If A< M jump condition is true
MOV     D,M               // M=> D
MOV     M,A               // A=>M
DCX     H                 // HL-1 => Hl
MOV     M,D               // D<=M
INX     H                 // HL+1=>HL
down:DCR                  // Decrement b by 1
JNZ     up                // Jump until B=0
```

```
DCR     C               // Decrement C by 1
JNZ     2005            // Jump until C=0
RST 1                   // Terminate

# ORG F100H             // Store number count at the address
# DB 04                 // Store Count
#ORG F200H              // Store numbers at the address
#DB DD,CC,BB,AA         // Store numbers at the address
```

7. Checking number of 0s and 1s in the given number

```
# ORG 2000H
MVI     C,00                // Clear reg.C
MVI     D,00                // Clear reg.D
MVI     A,F0                // Take number into Accumulator
MVI     B,08                // Counter 8 loaded in reg.B
up:RLC                      // Rotate left through carry
JNC     down                // Jump if CF=0
INR     D                   // D+1=>D for 1’s counter
JMP     shift               // Unconditional Jump
down:INR                    // C+1=> C for 0’s counter
shift:DCR      B            // B-1=> B
JNZ     up                  // True until B=0
RST 1                       // Terminate
```

8. Finding GCD of two numbers

```
# ORG 2000H
MVI     A,09          // Load first no. in reg.A
MVI     B,07          // Load second No. in reg.B
CMP     B             // Compare B to A
JZ      down          // True if A=B
JNC     shift         // True if A>B
MOV     C,A           // A → C
MOV     A,B           // A ← B
MOV     B,C           // C← B
shift:SUB B           // A-B → A
CMP     B             // Compare B to A
JZ      move          // True if A=B
JNC     shift         // True if A>B
MOV     C,A           // A ← C
MOV     A,B           // A ← B
MOV     B,C           // C → A
JMP     shift         // Unconditional Jump
move:MO A,B           // B → A
down:ST F200          // A → [address]
RST 1                 // Terminate
```

9. Finding LCM of 2 numbers

```
# ORG 2000H
LXI     H,F100        // HL ← F100
MOV     A,M           // A ← [HL]
MOV     C,M           // C←  [HL]
MOV     D,M           // D ← [HL]
INX     H            // HL+1 → HL
```

```
up:SUB  M           // A-M → M
JNC     up          // Jump if A>M
ADD     M           // M+A → A
CPI     00          // Compare A with 00 H
JZ      down        // True if A=00 H
MOV     A,D         // D → A
ADD     C           // A+C → A
MOV     D,A         // A → D
JMP     up          // Unconditional Jump
down:MA,D           // D →A
STA     F200        // A →[F200]
RST 1               // Terminate

# ORG F100H         // Store inputs at the address
# DB 05H ,03 H      // Store two bytes in successive location
```

10. Addition of n, two digit BCD numbers

```
# ORG 2000H
LXI     H,F100                  // HL &8592; F100
MOV     C,M                     // C &8592;[HL]
MVI     D,00                    // Clear reg.D
INX     H                       // HL+1 &8594; HL
DCR     C                       // C-1 &8594; C
MOV     A,M                     // M &8594; A
up:INX  H                       // HL+1 &8594; HL
ADD     M                       // M+A &8594;A
DAA                             // Decimal Adjust After Addition
JNC     down                    // Jump if no carry
INR     D                       // D+1 &8594;D
down:DCRC                       // C-1 &8594;C
JNZ     up                      // Jump if ZF=0
STA     F200                    // A &8592;[F200]
RST 1                           // Terminate

# ORG F100H                     // Store inputs at the address
# DB 04,43,77,555               // Store bytes in successive locations
```

8086 Programs

1. Addition of Two numbers

```
data segment
a db 09h
b db 02h
c db ?
cr db ?
data ends

code segment
assume cs:code,ds:data
start: mov cx,0000h
mov ax,data
mov ds,ax
mov dl,a
mov bl,b
add dl,bl
jnc next
inc cx
next: mov c,dl
mov cr,cl
hlt
code ends
end start
end next
Output
```

```
data segment
a dw 5e6eh
b dw 0a5ah
c dw ?
cr dw ?
data ends

code segment
assume cs:code,ds:data
start: mov cx,0000h
mov ax,data
mov ds,ax
mov dx,a
mov bx,b
mov al,bl
add al,dl
mov ah,bh
adc ah,dh
jnc next
inc cx
next: mov [di+4],ax
mov [di+6],cx
hlt
 code ends
end start
end next
Output
```

2. Subtraction of 2 numbers

```
data segment
a db 01h
b db 5ah
su db ?
br db ?
data ends

code segment
assume cs:code,ds:data
start: mov cx,0000h
mov ax,data
mov ds,ax
mov dl,a
mov bl,b
sub dl,bl
jnc next
inc cx
next:  mov su,bl
mov br,cl
hlt
 code ends
end start
end next
Output
```

```
data segment
a dw 0e4ch
b dw 455ah
su dw ?
br dw ?
data ends

code segment
assume cs:code,ds:data
start: mov cx,0000h
mov ax,data
mov ds,ax
mov dx,a
mov bx,b
sub bx,dx
jnc 0003
inc cx
mov su,bx
mov br,cx
hlt
 code ends
end start
Output
```

3. Division of 2 Numbers

```
data segment
a dw 000eh
b db 52h
```

```
qnt db ?
rmdr db ?
data ends

code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov ax,a
mov bl,b
div bl
mov qnt,al
mov rmdr,ah
hlt
 code ends
end start
Output
```

```
data segment
a dw 04eeh
b dw 0452h
qnt dw ?
rmdr dw ?
data ends

code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov ax,a
mov bx,b
div bx
mov qnt,ax
mov rmdr,dx
hlt
 code ends
end start
Output
```

Multiplication of 2 Numbers

```
data segment
a db 4ch
b db 5ah
mult dw ?
data ends

code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov al,a
mov bl,b
mul bl
mov mult,ax
```

```
hlt
 code ends
end start
Output
```

```
data segment
a dw 014ch
b dw 525ah
data ends

code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov ax,a
mov bx,b
mul bx
mov [di+4],ax
mov [di+6],dx
hlt
 code ends
end start
Output
```

4. Finding Sqrt of a number

```
data segment
a dw 0009h
b dw 1245h
c dw 5bach
d dw 256ah
e dw 0cc4h
data ends

code segment
assume cs:code,ds:data
start:
    MOV AX,DATA
    MOV DS,AX
    MOV CL,05
    DEC CL
up:
    MOV AX,[DI]
    MOV [DI+12],AX
    ADD DI,2
    LOOP up
    HLT
 code ends

end start
end up
Output
```

5. Moving a block of data

```
data segment
a dw 0024h
```

```
data ends

code segment
assume cs:code,ds:data
start:
    MOV AX,DATA
    MOV DS,AX
      MOV AX,a
      MOV CX, 0001
      MOV BX, 0001
up:
    sub ax,bx
    cmp ax,0000
    jz down
      ADD BX, 02
      INC CX
      jmp up
down:
    MOV [di+2], CX
      HLT
hlt
 code ends

end start
end up
end down
Output
```

6. Sorting of integers

```
data segment
a dw 0eeeh
b dw 1245h
c dw 5bach
d dw 8fffh
e dw 0cc4h
data ends

code segment
assume cs:code,ds:data
start:
    MOV AX,DATA
    MOV DS,AX
    MOV CL,05
    DEC CL
up:
    MOV DI,0000H
    MOV DL,CL
up1:MOV AX,[DI]
    ADD DI,2
    CMP AX,[DI]
    JC down
    MOV BX,[DI]
    MOV [DI],AX
    SUB DI,2
    MOV [DI],BX
    ADD DI,2

down: DEC DL
    JNZ up1
    DEC CL
    JNZ up
    HLT
 code ends
end start
end up
end up1
end down
Output
```

7. Checking number of 0s and 1s in the given number

```
data segment
a dw 9fffh
data ends

code segment
assume cs:code,ds:data
start:
    MOV CL,16
    MOV DX,00
    MOV BX,00
    MOV AX,DATA
    MOV DS,AX
    MOV AX,a
 up:ROL AX,1
    JNC down
    INC DL
    JMP down1
down:INC BL
down1:DEC CL
    JNZ up
    MOV [DI+2],DX
    MOV [DI+4],BX
    HLT
 code ends
end start
end up
end down
end down1
Output
```

8. Finding GCD of two numbers

```
data segment
a dw 0015h
b dw 0007h
data ends

code segment
assume cs:code,ds:data
start:
    MOV AX,DATA
    MOV DS,AX
    MOV AX,[DI]
    ADD DI,2
up: CMP AX,[DI]
    JZ down
    JNC next
    MOV BX,[DI]
    MOV [DI],AX
    MOV AX,BX
next:  SUB AX,[DI]
    JMP up
down: MOV [DI+2],AX
    HLT
 code ends
end start
end up
end next
end down
Output
```

9. Finding LCM of 2 numbers

```
data segment
a dw 0012h
b db 07h
data ends

code segment
assume cs:code,ds:data
start:
    MOV AX,DATA
    MOV DS,AX
    MOV AX,[DI]
    ADD DI,2
    MOV BX,AX
    MOV CX,BX
up: DIV [DI]
    CMP AH,0000H
    JZ down
    MOV AX,CX
    ADD AX,BX
    ADC DX,00
    MOV CX ,AX
    JMP up
down:MOV [DI+2],CX
    HLT
 code ends
end start
end up
end down
Output
```

10. Addition of n, two digit BCD numbers

```
data segment
a dw 1234h
b dw 5678h
c dw 4586h
d dw 7890h
e dw 4758h
sum dw ?
cr dw ?
data ends

code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov cl,05h
dec cl
mov bx,00h
mov ax,[di]
next: add di,2
add al,[di]
daa
mov dl,al
mov al,ah
adc al,[di+1]
daa
mov dh,al
mov ax,dx
jnc next1
inc bl
next1: dec cl
jnz next
mov sum,ax
mov cr,bx
hlt
 code ends
end start
end next
Output
```