# Assignment 05 | MFP
# CE-092

Assignment submission for Microprocessor Fundamentals and Programming subject week 5.

nevilparmar24@gmail.com

## Task 1:

Write the programs to verify the instructions AAA, AAS, AAM, AAD, DAA and DAS instructions.

Code:

```
data segment
            n1 db '8'
            n2 db '5'
            sum dw ?
            dif dw ?
            product dw ?
            remainder db ?
            quotient db ?
            d_sum db ?
            d_dif db ?
data ends

code segment
            assume cs:code,ds:data
            mov ax,data
            mov ds,ax

            ;aaa
```

```
sub ah,ah
mov al,[n1]
add al,[n2]
aaa
or ax,3030h
mov [sum],ax

;aas
sub ah,ah
mov al,[n1]
sub al,[n2]
aas
or ax,3030h
mov [dif],ax

;aam
sub ah,ah
mov al,[n1]
mov bl, [n2]
and al,0fh
and bl,0fh
mul bl
aam
or ax,3030h
mov [product],ax

;aad
mov ax,129
aad
mov bl,2
div bl
mov [remainder],ah
```

```
            mov [quotient],al

            ;daa
            mov al,17h
            add al,42h
            daa
            mov [d_sum],al

            ;das
            mov al,2h
            sub al,18h
            das
            mov [d_dif],al

            int 03
code ends
end
```

Output:

```
C:\DEBUG125>debug c:\TASK1~1.EXE
-g=0010
Unexpected breakpoint interrupt
AX=0184 BX=0002 CX=006E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0744 ES=0734 SS=0743 CS=0744 IP=006E NV UP EI NG NZ AC PE CY
0744:006E E8823B            CALL    3BF3
-d
0744:0000  38 35 33 31 33 30 30 34-01 40 59 84 00 00 00 00 85313004.@Y.....
0744:0010  B8 44 07 8E D8 2A E4 A0-00 00 02 06 01 00 37 0D .D...*........7.
0744:0020  30 30 A3 02 00 2A E4 A0-00 00 2A 06 01 00 3F 0D 00...*....*...?.
0744:0030  30 30 A3 04 00 2A E4 A0-00 00 8A 1E 01 00 24 0F 00...*........$.
0744:0040  80 E3 0F F6 E3 D4 0A 0D-30 30 A3 06 00 B8 81 00 ........00......
0744:0050  D5 0A B3 02 F6 F3 88 26-08 00 A2 09 00 B0 17 04 .......&........
0744:0060  42 27 A2 0A 00 B0 02 2C-18 2F A2 0B 00 CC E8 82 B'.....,./......
0744:0070  3B E9 DD 04 BF 3C 01 57-8B D8 E8 87 EE 26 C7 06 ;....<.W.....&..
-
```

## Task 2:

Implement the above instructions. The program should behave in the similar manner without using the above mentioned instructions.

## Task 2.1:

To implement AAA Instruction.

Code:

```
data segment
        addition db 0DH, 0AH, "Addition of two number
is : $"
        space db 0DH, 0AH, "$"
        addResult dw 0
        num1 db 6H
        num2 db 8H
data ends

printString macro string
        mov dx, offset string
        mov ah, 09H
        int 21H
endm

code segment
            assume ds:data, cs:code
    start:
            mov ax, data
            mov ds, ax

    addL:
            mov ah, 0
            mov al, num1
```

```
            add al, num2
            mov cl, 04H
            cmp al, 10H
            jl jump
            add al, 06H
            jmp sep
    jump:
            and al, 0FH
            cmp al, 09H
            jle directa
            add al, 06H
    sep:
            mov ah, al
            and ah, 0F0H
            rol ah, cl
            and al, 0FH
    directa:
            mov [addResult], ax
            printString space
            printString addition
            mov dl, byte ptr addResult[1]
            add dl, 30H
            mov ah, 02H
            int 21H
            mov dl, byte ptr addResult[0]
            add dl, 30H
            mov ah, 02H
            int 21H

            mov ah, 4CH
            int 21H
code ends
```

```
end start
```

Output:


## Task 2.2:

To implement AAS Instruction.

Code:

```
data segment
            subtraction db 0DH, 0AH, "Subtraction of
two number is : $"
            minus db "-$"
            space db 0DH, 0AH, "$"
            sub_ dw 0
            num1 db 6H
            num2 db 8H
data ends

printString macro string
        mov dx, offset string
        mov ah, 09H
        int 21H
endm

code segment
            assume ds:data, cs:code
    start:
```

```
        mov ax, data
        mov ds, ax

_sub:
        mov ah, 0
        mov al, num1
        sub al, num2
        cmp al, 09H
        jnc substracting
        jl direct_s
substracting:
        and al, 0FH
        sub al, 06H
        mov bl, 09H
        sub bl, al
        mov al, bl
        add al, 01H
        mov [sub_], ax
        printString space
        printString subtraction
        printString minus
        jmp next
direct_s:
        mov ah, 0H
        mov [sub_], ax
        printString space
        printString subtraction
        mov dl, byte ptr sub_[1]
        add dl, 30H
        mov ah, 02H
        int 21H
next:
```

```
            mov dl, byte ptr sub_[0]
            add dl, 30H
            mov ah, 02H
            int 21H

            mov ah, 4CH
            int 21H
code ends
end start
```

Output:

```
C:\DEBUG125>debug c:\AAS.EXE
-g


Subtraction of two number is : -2
Program terminated normally (0032)
```

## Task 2.3:

To implement AAM Instruction.

Code:

```
data segment
        multiplication db 0DH, 0AH, "Multiplication of
two number is : $"
        space db 0DH, 0AH, "$"
        num1 db 6H
        num2 db 8H
        hex db 10 dup ('$')
data ends

printString macro string
        mov dx, offset string
        mov ah, 09H
```

```
        int 21H
endm

code segment
            assume ds:data, cs:code
    start:
            mov ax, data
            mov ds, ax
    mulL:
            mov ah, 0
            mov al, num1
            mov bl, num2
            mul bl
            lea si, hex
            mov cx, 0
            mov bx, 10
    loop1:
            mov dx, 0
            div bx
            add dl, 30h
            push dx
            inc cx
            cmp ax, 9
            jg loop1
            add al, 30h
            mov [si], al
    loop2:
            pop ax
            inc si
            mov [si], al
            loop loop2
            printString space
```

```
            printString multiplication
            mov dl, offset hex
            mov ah, 09h
            int 21h

            mov ah, 4CH
            int 21H
code ends
end start
```

Output:

```
C:\DEBUG125>debug c:\AAM.EXE
-g


Multiplication of two number is : 48
Program terminated normally (0038)
```

## Task 2.4:

To implement AAD Instruction.

Code:

```
data segment
            division db 0DH, 0AH, "Division of two
number is : $"
            remainder db 0DH, 0AH, "Remainder is : $"
            space db 0DH, 0AH, "$"
            divResult dw 0
            num1 db 6H
            num2 db 8H
data ends

printString macro string
            mov dx, offset string
```

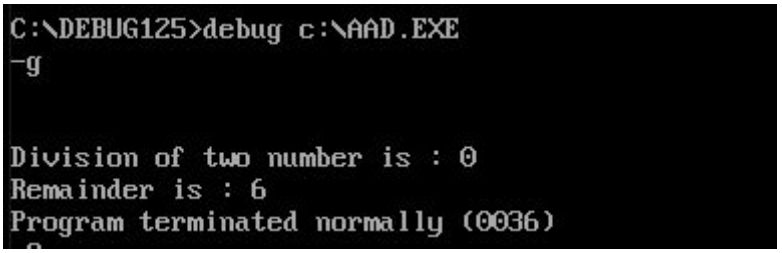
```
            mov ah, 09H
            int 21H
endm

code segment
            assume ds:data, cs:code
    start:
            mov ax, data
            mov ds, ax
    divL:
            mov ah, 0
            mov al, num1
            mov bl, num2
            mov ah, al
            and al, 0FH
            mov cl, 04H
            shr ah, cl
            div bl
            mov [divResult], ax
            printString space
            printString division
            mov dl, byte ptr divResult[0]
            add dl, 30H
            mov ah, 02H
            int 21H
            printString remainder
            mov dl, byte ptr divResult[1]
            add dl, 30H
            mov ah, 02H
            int 21H

            mov ah, 4CH
```

```
            int 21H
code ends
end start
```

Output:




## Task 2.5:

To implement DAA Instruction.

Code:

```
data segment
            daaResultString db 0DH, 0AH, "DAA result is
: $"
            space db 0DH, 0AH, "$"
            daaResult dw 0
            num1 db 6H
            num2 db 8H
data ends

printString macro string
            mov dx, offset string
            mov ah, 09H
            int 21H
endm

code segment
            assume ds:data, cs:code
```

```
start:
        mov ax, data
        mov ds, ax
daaL:
        mov ah, 0
        mov al, num1
        add al, num2
        mov cl, 04H
        cmp al, 10H
        jl jump2
        add al, 06H
        jmp sep2
jump2:
        and al, 0FH
        cmp al, 09H
        jle directd
        add al, 06H
sep2:
        mov ah, al
        and ah, 0F0H
        rol ah, cl
        and al, 0FH
directd:
        mov [daaResult], ax
        printString space
        printString daaResultString
        mov dl, byte ptr daaResult[1]
        add dl, 30H
        mov ah, 02H
        int 21H
        mov dl, byte ptr daaResult[0]
        add dl, 30H
```

```
            mov ah, 02H
            int 21H

            mov ah, 4CH
            int 21H
code ends
end start
```

Output:

```
C:\DEBUG125>debug c:\DAA.EXE
-g


DAA result is : 14
Program terminated normally (0034)
```

## Task 2.6:

To implement DAS Instruction.

Code:

```
code segment
            assume cs:code
            mov al, 23h
            sub al, 31h
            mov bl,al
            mov bh,al
            mov cl,al
            and bl,0f0h
            cmp bl,90h
            jz abc
            jc abc
            sub bl,60h
    abc:
```

```
            mov al,bl
            cmp cl,6fh
            jc pqr
            jz pqr
    pqr:
            and cl,0fh
            cmp cl,9
            jz xyz
            jc xyz
            sub cl,6
    xyz:
            add al,cl
    end_:
            int 03
code ends
end
```

Output:

```
-g=0000
Unexpected breakpoint interrupt
AX=FF92 BX=F290 CX=0002 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0030 NV UP EI NG NZ NA PO NC
0744:0030 0474              ADD     AL,74
-r
AX=FF92 BX=F290 CX=0002 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0734 ES=0734 SS=0743 CS=0744 IP=0030 NV UP EI NG NZ NA PO NC
0744:0030 0474              ADD     AL,74
```

## Task 3:

Implement a calculator for single digit numbers , which take the input through the keyboard and display the result on the screen.

Code:

```
data segment
            space db 0DH, 0AH, "$"
            Heading db 0DH, 0AH, "Basic Calc For 1
```

```
Digit Number!$"
            string1 db 0DH, 0AH, "Enter 1st number : $"
            string2 db 0DH, 0AH, "Enter 2nd number : $"
            operator db 0DH, 0AH, "Enter the operation
: $"
            addition db 0DH, 0AH, "Addition of two
number is : $"
            subtraction db 0DH, 0AH, "subtraction of
two number is : $"
            multiplication db 0DH, 0AH, "Multiplication
of two number is : $"
            division db 0DH, 0AH, "Division of two
number is : $"
            remainder db 0DH, 0AH, "Remainder is : $"
            continue db 0DH, 0AH, "Do you want to
continue : type 'Y' otherwise 'press any key' : $"
            minus db "-$"
            add_ dw 0
            sub_ dw 0
            mul_ dw 0
            div_ dw 0
            ope_ db 1 dup(?)
            res_ db 1 dup(?)
data ends

print_string macro msg
            mov dx, offset msg
            mov ah, 09H
            int 21H
endm

scan_value macro
```

```
            mov ah, 01H
            int 21H
            mov res_, al
endm

user_input macro
            print_string space
            print_string space
            print_string string1
            mov ah, 01H
            int 21H
            mov bl, al
            sub bl, 30H
            print_string operator
            mov ah, 01H
            int 21H
            mov [ope_], al
            print_string string2
            mov ah, 01H
            int 21H
            sub al, 30H
endm

code segment
            assume ds:data, cs:code
    start:
            mov ax, data
            mov ds, ax
            print_string Heading
            print_string space
    up:
            user_input
```

```
        cmp [ope_], '+'
        jz additionL
        cmp [ope_], '-'
        jz subtractionL
        cmp [ope_], '*'
        jz mulLabel
        cmp [ope_], '/'
        jz divL

        ;Addition
additionL:
        mov ah, 0
        add al, bl
        AAA
        mov [add_], ax
        print_string addition
        mov dl, byte ptr add_[1]
        add dl, 30H
        mov ah, 02H
        int 21H
        mov dl, byte ptr add_[0]
        add dl, 30H
        mov ah, 02H
        int 21H
        print_string space
        print_string continue
        scan_value
        cmp res_, 'Y'
        jz up
        jmp _Exit

        ;subtraction
```

```
subtractionL:
        mov bh, 0
        sub bl, al
        AAS
        jnc skip
        neg bx
        print_string subtraction
        mov [sub_], bx
        mov dx, offset minus
        mov ah, 09H
        int 21H
        jmp next
skip:
        print_string subtraction
        mov [sub_], bx
        mov dl, byte ptr sub_[1]
        add dl, 30H
        mov ah, 02H
        int 21H
next:
        mov dl, byte ptr sub_[0]
        add dl, 30H
        mov ah, 02H
        int 21H
        print_string space
        print_string continue
        scan_value
        cmp res_, 'Y'
        jz up
        jmp _Exit

        ;Multiplication
```

```
mulLabel:
        mov ah, 0
        mul bl
        AAM
        mov [mul_], ax
        print_string multiplication
        mov dl, byte ptr mul_[1]
        add dl, 30H
        mov ah, 02H
        int 21H
        mov dl, byte ptr mul_[0]
        add dl, 30H
        mov ah, 02H
        int 21H
        print_string space
        print_string continue
        scan_value
        cmp res_, 'Y'
        jz up
        jmp _Exit

        ;Division
divL:
        xchg al, bl
        mov ah, 0
        AAD
        div bl
        mov [div_], ax
        print_string division
        mov dl, byte ptr div_[0]
        add dl, 30H
        mov ah, 02H
```

```
            int 21H
            print_string remainder
            mov dl, byte ptr div_[1]
            add dl, 30H
            mov ah, 02H
            int 21H
            print_string space
            print_string continue
            scan_value
            cmp res_, 'Y'
            jz up
            jmp _Exit

            ;Exit
    _Exit:
            mov ax, 4c00h
            int 21H
code ends
end start
```

Output:

```
Enter 1st number: 6
Enter the operation: +
Enter 2nd number: 5
Addition of two number is: 11

Do you want to continue: type 'Y' otherwise 'press any key' : Y


Enter 1st number: 5
Enter the operation: -
Enter 2nd number: 9
subtraction of two number is: -4
```

```
Enter 1st number: 8
Enter the operation: *
Enter 2nd number: 9
Multiplication of two number is: 72

Do you want to continue: type 'Y' otherwise 'press any key' : Y


Enter 1st number: 9
Enter the operation: /
Enter 2nd number: 2
Division of two number is: 4
Remainder is: 1

Do you want to continue: type 'Y' otherwise 'press any key' : N
```

Nevil Parmar
CE-092
https://nevilparmar.me