horizontal line

**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

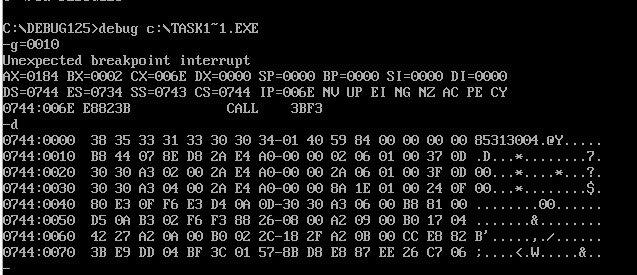
mov [d\_dif],al

int 03

code ends

end

Output:



**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

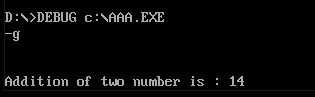
mov ah, 4CH

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

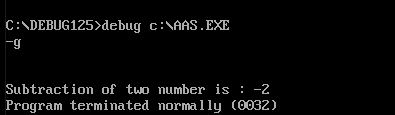
mov ah, 4CH

int 21H

code ends

end start

Output:



**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

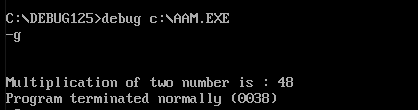
mov ah, 4CH

int 21H

code ends

end start

Output:



**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

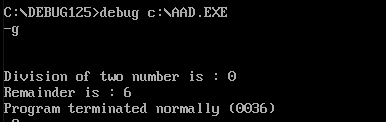
mov ah, 4CH

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:



**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

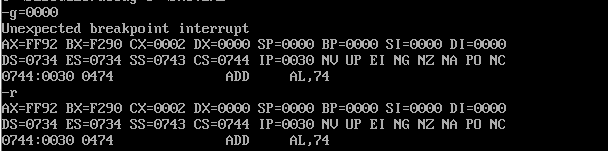
end\_:

int 03

code ends

end

Output:



**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:

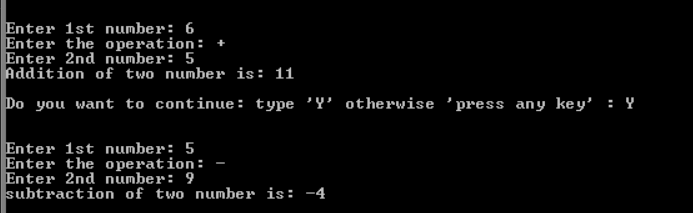
mov ax, 4c00h

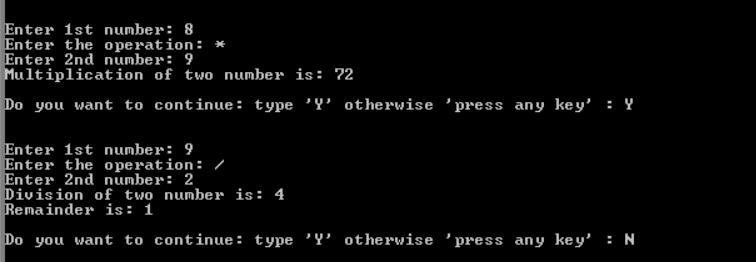
int 21H

code ends

end start

Output:





Nevil Parmar

CE-092

https://nevilparmar.me