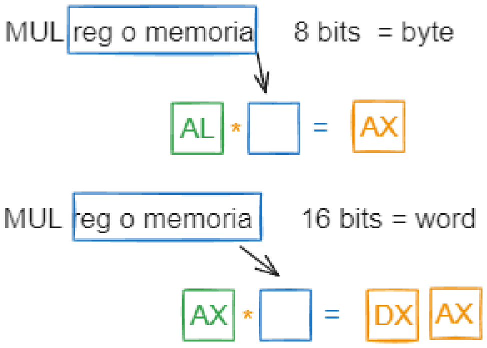
MUL reg o memoria


MUL reg o memoria 8 bits = byte

AL * □ = AX

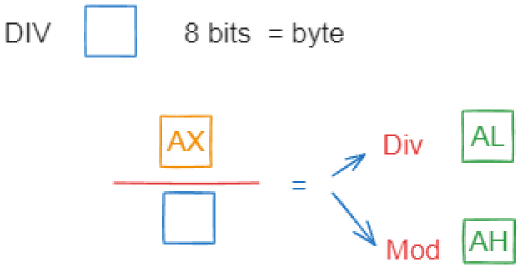
MUL reg o memoria 16 bits = word

AX * □ = DX AX

when operand is a **byte**:
AX = AL * operand.

when operand is a **word**:
(DX AX) = AX * operand.

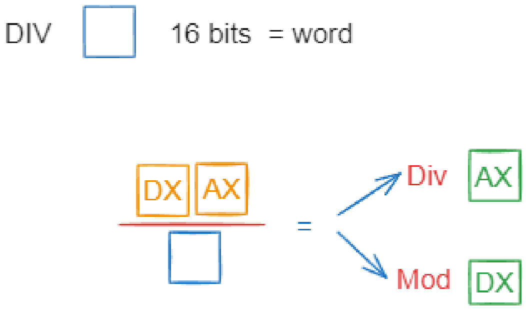
DIV reg o memoria


DIV □ 8 bits = byte

AX / □ = Div AL, Mod AH

when operand is a **byte**:
AL = AX / operand
AH = remainder (modulus)

when operand is a **word**:
AX = (DX AX) / operand
DX = remainder (modulus)

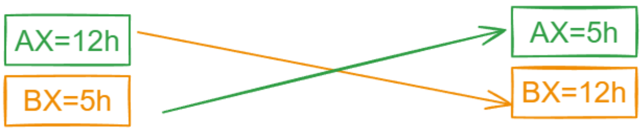

DIV □ 16 bits = word

DX AX / □ = Div AX, Mod DX

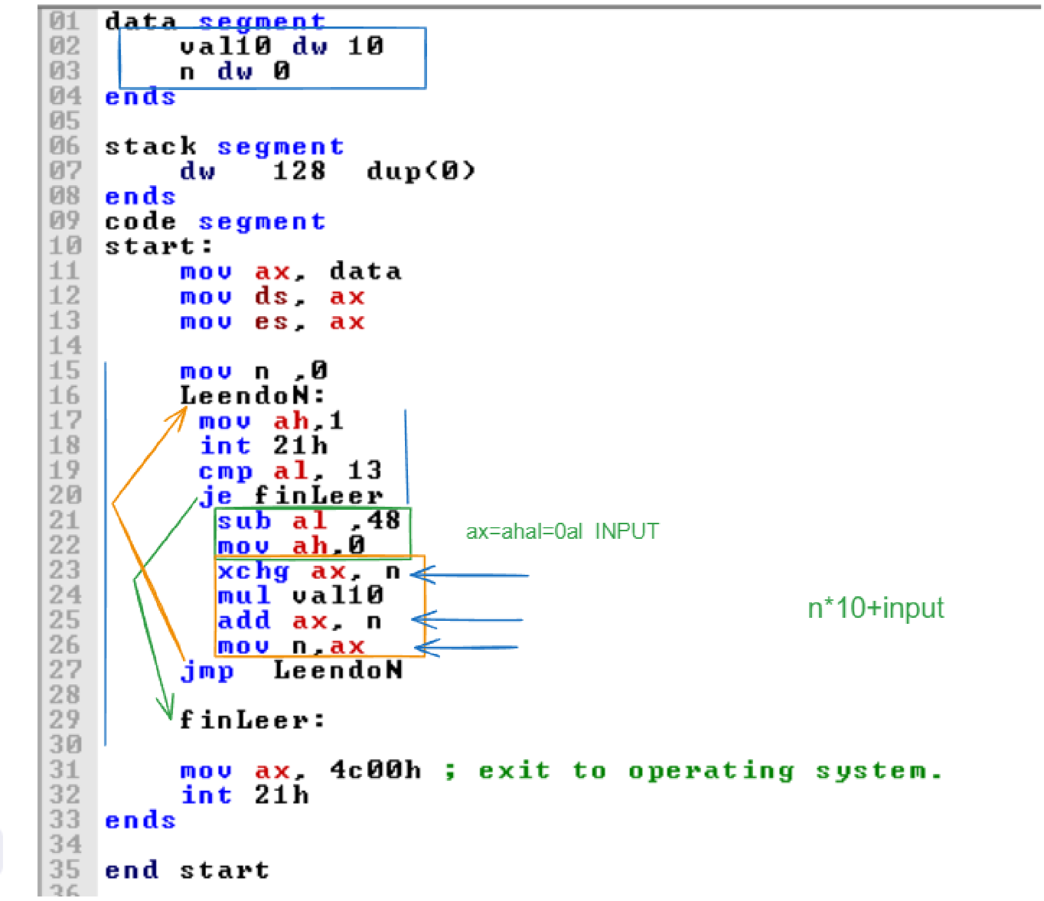
XCHG

XCHG	REG, memory memory, REG REG, REG

XCHG AX,BX


AX=12h
BX=5h
AX=5h
BX=12h

Leer un numero con mas de un digito



```
data segment
        val10 dw 10
        n dw 0
ends

stack segment
        dw    128   dup(0)
ends
code segment
start:
        mov ax, data
        mov ds, ax
        mov es, ax

        mov n ,0
        LeendoN:
          mov ah,1
          int 21h
          cmp al, 13
          je finLeer
           sub al ,48
           mov ah,0
           xchg ax, n
           mul val10
           add ax, n
           mov n,ax
        jmp   LeendoN

        finLeer:

        mov ax, 4c00h ; exit to operating system.
        int 21h
ends

end start
```

ax=ahal=0al INPUT

n*10+input

123

n=0

n=0*10+1

n=1*10+2

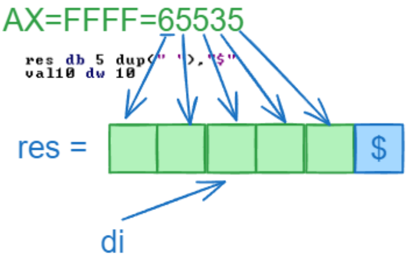
n=12*10+3=123

Reutilizar el código anterior para leer dos números o más números

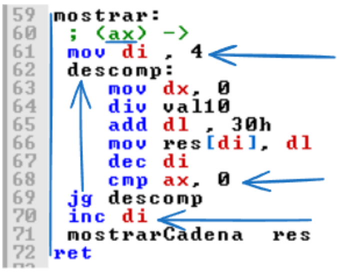
```
data segment
    val10 dw 10
    n dw 0
    n1 dw 0
    n2 dw 0
    enter db  10,13,"$"
ends
stack segment
    dw   128  dup(0)
ends
macro read parms
    call leerNumero
    mov ax, n
    mov parms, ax
endm
code segment
start:
    mov ax, data
    mov ds, ax
    mov es, ax
    read n1
    lea dx, enter
    mov ah,9
    int 21h
    read n2
    mov ax, 4c00h ; exit to operating system.
    int 21h
ends
```

```
leerNumero:
    mov n ,0
    LeendoN:
     mov ah,1
     int 21h
     cmp al, 13
     je finLeer
      sub al ,48
      mov ah,0
      xchg ax, n
      mul val10
      add ax, n
      mov n,ax
    jmp  LeendoN
    finLeer:
ret
end start
```

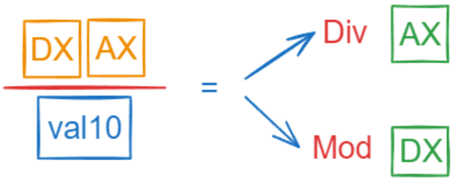
Mostrar un numero con mas de un digito en la pantalla



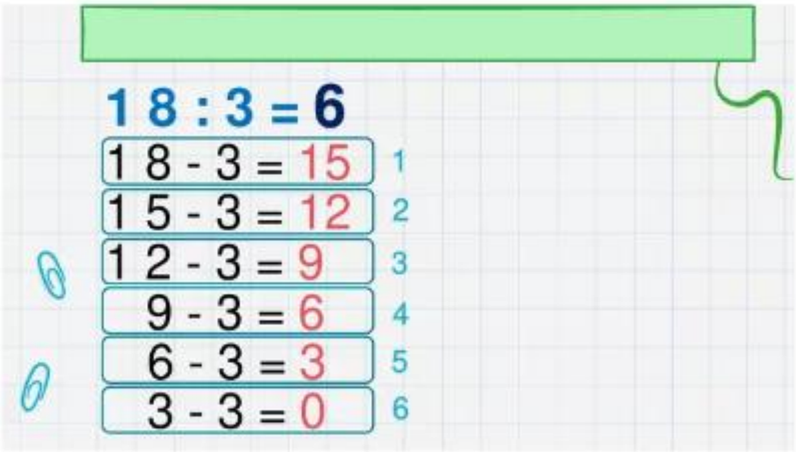
```
res db 5 dup(" "),"$"
val10 dw 10
```

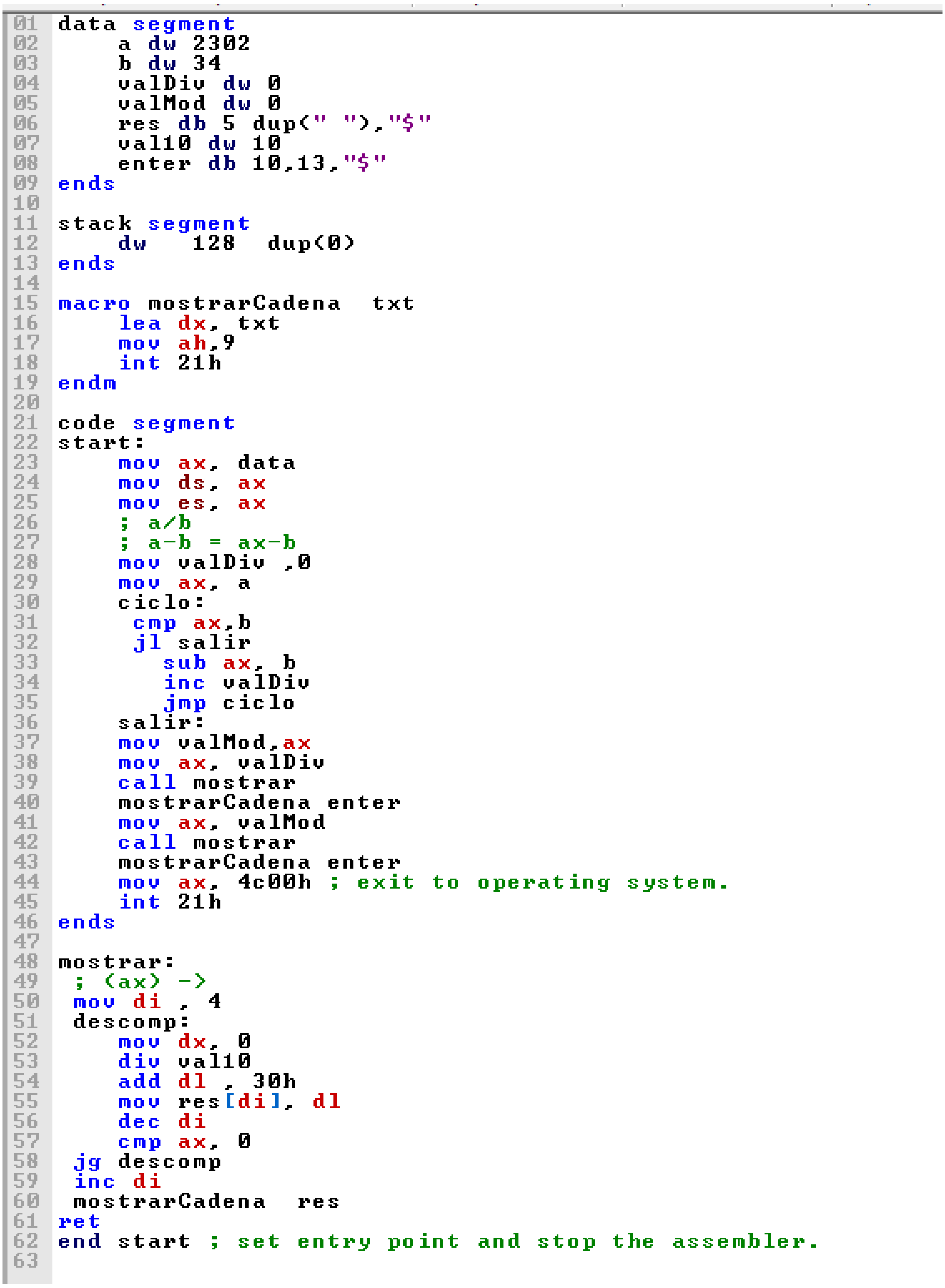


```
mostrar:
  ; (ax) ->
  mov di , 4
  descomp:
     mov dx, 0
     div val10
     add dl , 30h
     mov res[di], dl
     dec di
     cmp ax, 0
  jg descomp
  inc di
  mostrarCadena  res
ret
```



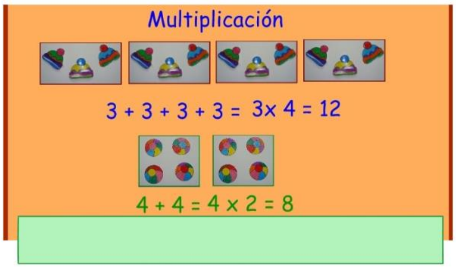
Realizar division con restas sucesivas



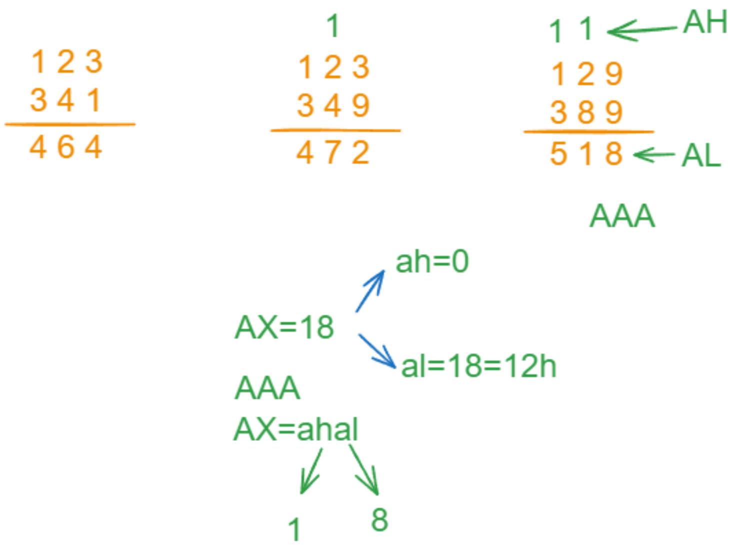


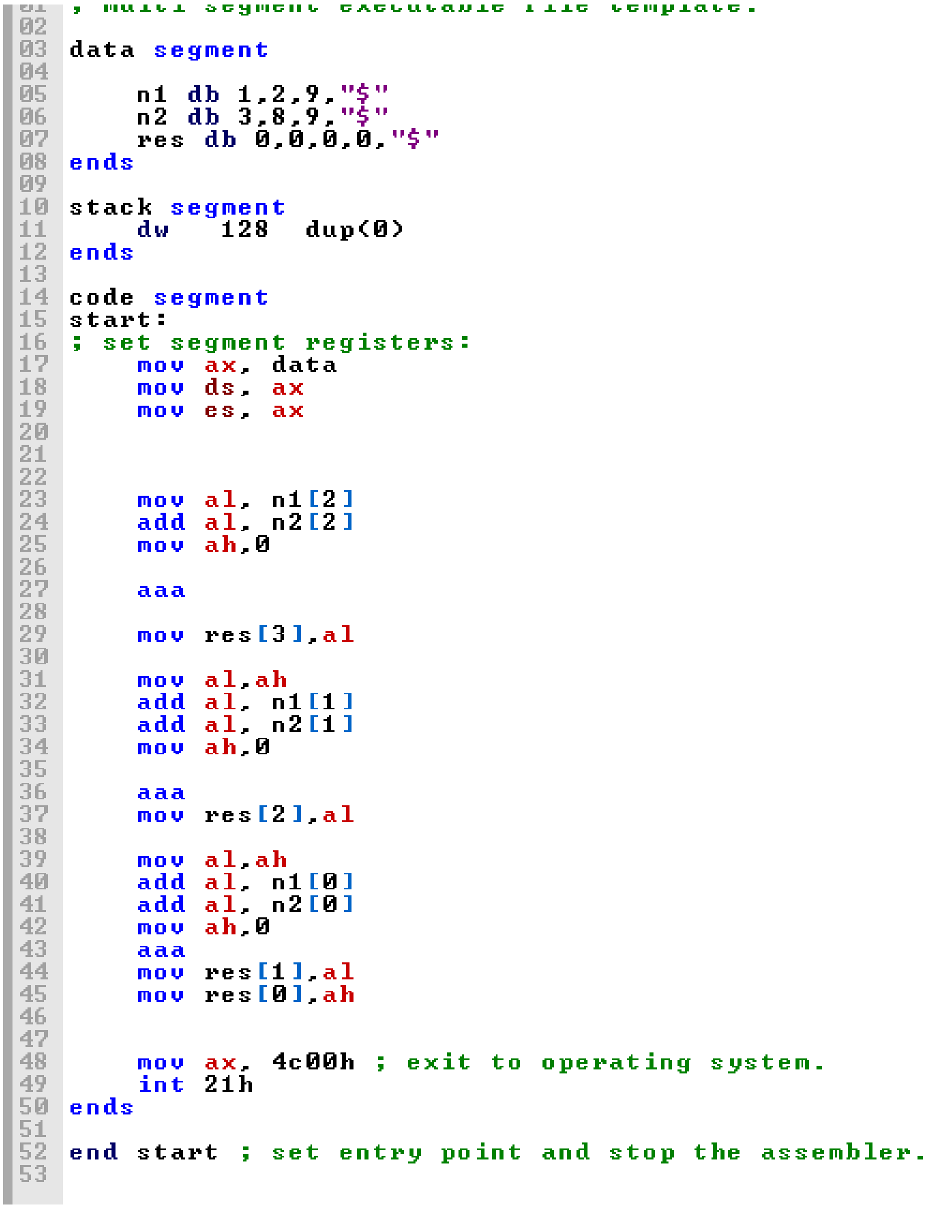
```
data segment
    a dw 2302
    b dw 34
    valDiv dw 0
    valMod dw 0
    res db 5 dup(" "),"$"
    val10 dw 10
    enter db 10,13,"$"
ends

stack segment
    dw   128  dup(0)
ends

macro mostrarCadena  txt
    lea dx, txt
    mov ah,9
    int 21h
endm

code segment
start:
    mov ax, data
    mov ds, ax
    mov es, ax
    ; a/b
    ; a-b = ax-b
    mov valDiv ,0
    mov ax, a
    ciclo:
     cmp ax,b
     jl salir
       sub ax, b
       inc valDiv
       jmp ciclo
    salir:
    mov valMod,ax
    mov ax, valDiv
    call mostrar
    mostrarCadena enter
    mov ax, valMod
    call mostrar
    mostrarCadena enter
    mov ax, 4c00h ; exit to operating system.
    int 21h
ends

mostrar:
 ; (ax) ->
 mov di , 4
 descomp:
    mov dx, 0
    div val10
    add dl , 30h
    mov res[di], dl
    dec di
    cmp ax, 0
 jg descomp
 inc di
 mostrarCadena  res
ret
end start ; set entry point and stop the assembler.
```

Multiplicacion con sumas sucesivas



Suma de digito por digito

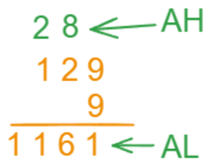




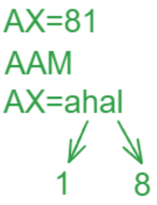
```
; multi segment executable file template.

data segment

    n1 db 1,2,9,"$"
    n2 db 3,8,9,"$"
    res db 0,0,0,0,"$"
ends

stack segment
    dw   128  dup(0)
ends

code segment
start:
; set segment registers:
    mov ax, data
    mov ds, ax
    mov es, ax



    mov al, n1[2]
    add al, n2[2]
    mov ah,0

    aaa

    mov res[3],al

    mov al,ah
    add al, n1[1]
    add al, n2[1]
    mov ah,0

    aaa
    mov res[2],al

    mov al,ah
    add al, n1[0]
    add al, n2[0]
    mov ah,0
    aaa
    mov res[1],al
    mov res[0],ah


    mov ax, 4c00h ; exit to operating system.
    int 21h
ends

end start ; set entry point and stop the assembler.
```

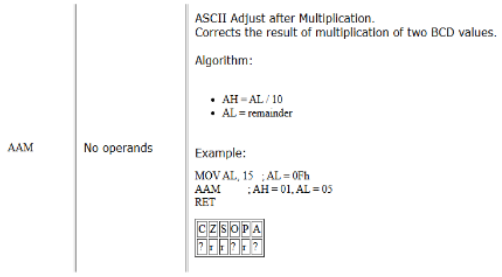
Multiplicacion de digito por digito

```
 123
 341
 ---
 464
```

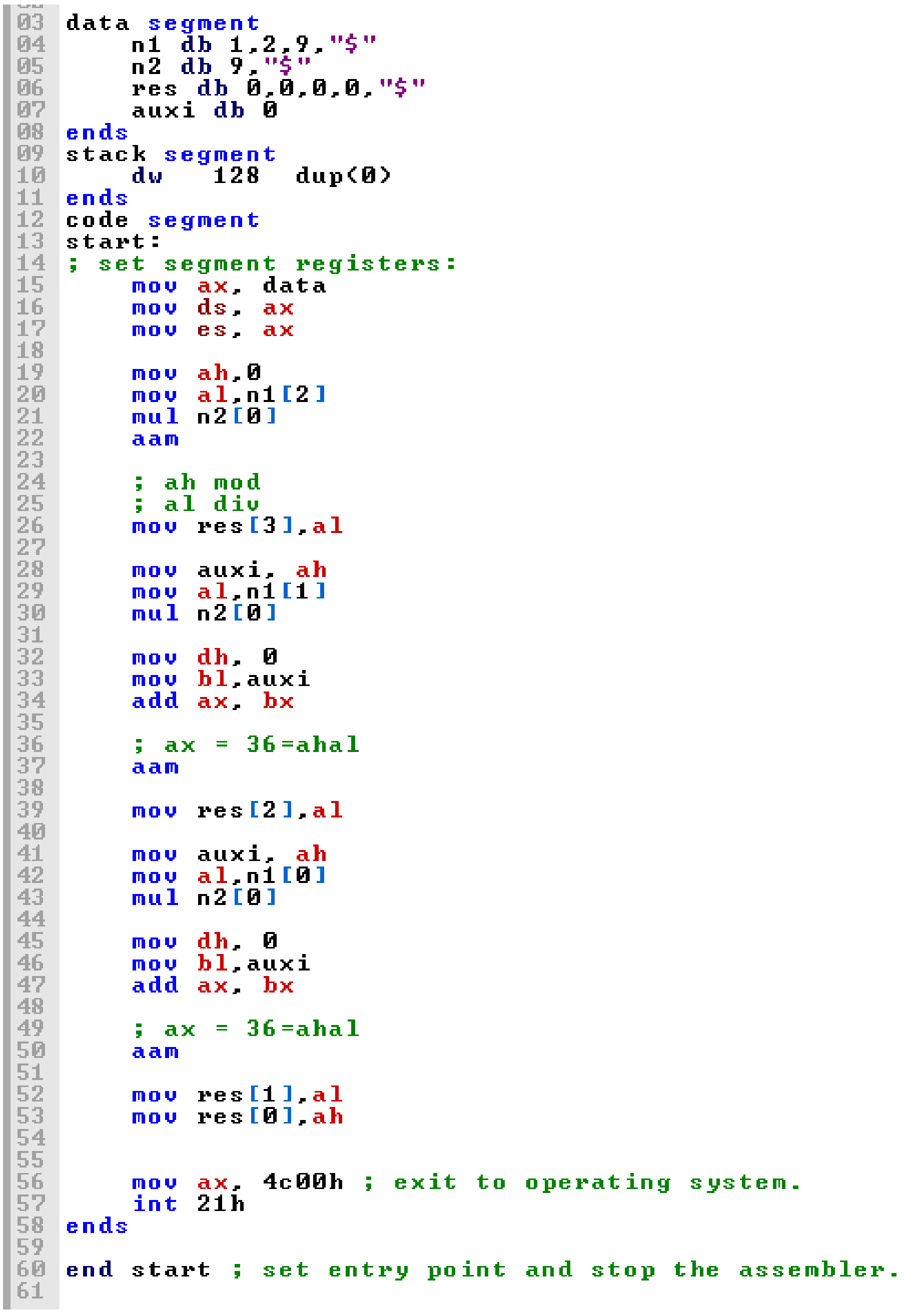


AAM



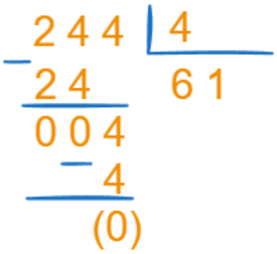


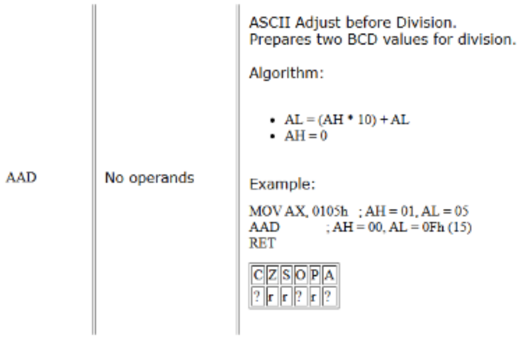
AAM	No operands	ASCII Adjust after Multiplication. Corrects the result of multiplication of two BCD values. Algorithm: • AH = AL / 10 • AL = remainder Example: MOV AL, 15 ; AL = 0Fh AAM ; AH = 01, AL = 05 RET C Z S O P A ? r r ? r ?



```
data segment
    n1 db 1,2,9,"$"
    n2 db 9,"$"
    res db 0,0,0,0,"$"
    auxi db 0
ends
stack segment
    dw   128  dup(0)
ends
code segment
start:
; set segment registers:
    mov ax, data
    mov ds, ax
    mov es, ax

    mov ah,0
    mov al,n1[2]
    mul n2[0]
    aam

    ; ah mod
    ; al div
    mov res[3],al

    mov auxi, ah
    mov al,n1[1]
    mul n2[0]

    mov dh, 0
    mov bl,auxi
    add ax, bx

    ; ax = 36=ahal
    aam

    mov res[2],al

    mov auxi, ah
    mov al,n1[0]
    mul n2[0]

    mov dh, 0
    mov bl,auxi
    add ax, bx

    ; ax = 36=ahal
    aam

    mov res[1],al
    mov res[0],ah


    mov ax, 4c00h ; exit to operating system.
    int 21h
ends

end start ; set entry point and stop the assembler.
```

Div de digito por digito





AAD

No operands

ASCII Adjust before Division.
Prepares two BCD values for division.

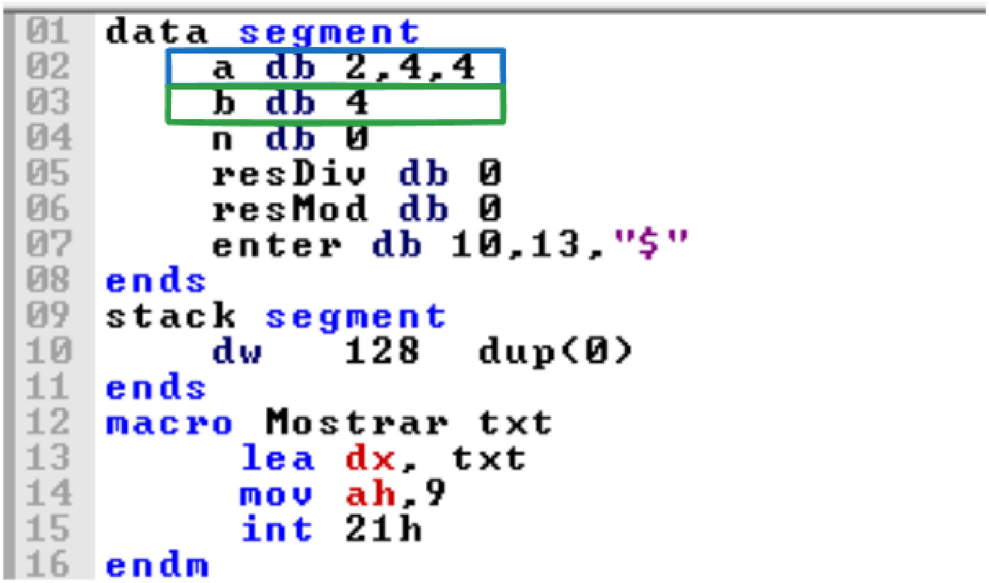
Algorithm:

- AL = (AH * 10) + AL
- AH = 0

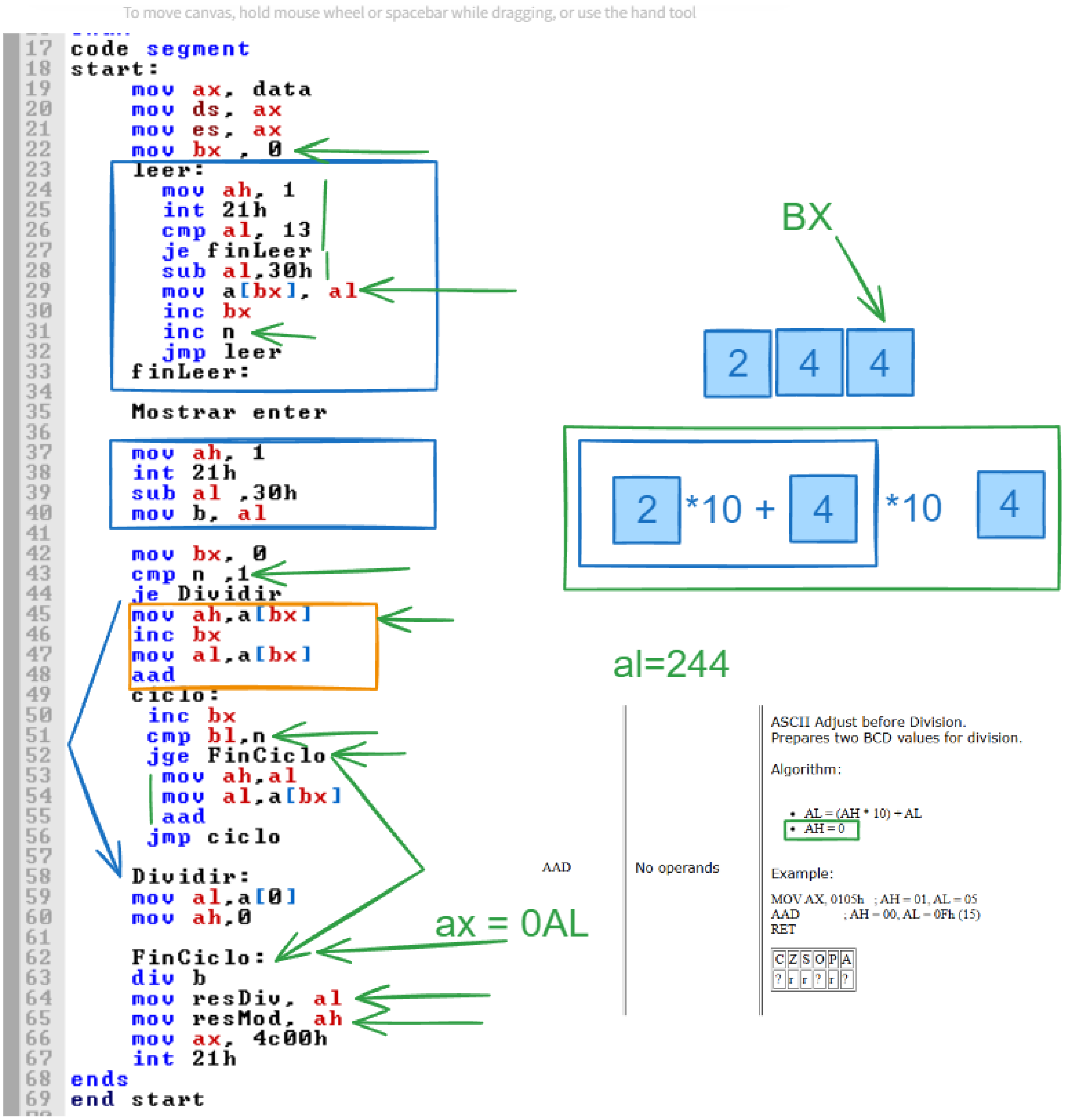
Example:

```
MOV AX, 0105h   ; AH = 01, AL = 05
AAD             ; AH = 00, AL = 0Fh (15)
RET
```

C	Z	S	O	P	A
?	r	r	?	r	?



```
data segment
    a db 2,4,4
    b db 4
    n db 0
    resDiv db 0
    resMod db 0
    enter db 10,13,"$"
ends
stack segment
    dw   128  dup(0)
ends
macro Mostrar txt
        lea dx, txt
        mov ah,9
        int 21h
endm
```



To move canvas, hold mouse wheel or spacebar while dragging, or use the hand tool

```
code segment
start:
     mov ax, data
     mov ds, ax
     mov es, ax
     mov bx , 0
     leer:
        mov ah, 1
        int 21h
        cmp al, 13
        je finLeer
        sub al,30h
        mov a[bx], al
        inc bx
        inc n
        jmp leer
     finLeer:

     Mostrar enter

     mov ah, 1
     int 21h
     sub al ,30h
     mov b, al

     mov bx, 0
     cmp n ,1
     je Dividir
     mov ah,a[bx]
     inc bx
     mov al,a[bx]
     aad
     ciclo:
      inc bx
      cmp bl,n
      jge FinCiclo
       mov ah,al
       mov al,a[bx]
       aad
      jmp ciclo

     Dividir:
     mov al,a[0]
     mov ah,0

     FinCiclo:
     div b
     mov resDiv, al
     mov resMod, ah
     mov ax, 4c00h
     int 21h
ends
end start
```

AAD

No operands

ASCII Adjust before Division.
Prepares two BCD values for division.

Algorithm:

- AL = (AH * 10) + AL
- AH = 0

Example:

MOV AX, 0105h ; AH = 01, AL = 05
AAD ; AH = 00, AL = 0Fh (15)
RET

C	Z	S	O	P	A
?	r	r	?	r	?