



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO SUPERIOR PROGRESO

**PROGRAMA ACADÉMICO DE INGENIERÍA EN
SISTEMAS COMPUTACIONALES**

ASIGNATURA

Lenguaje de interfaz.

DOCENTE

Edgar Alejandro Sagundo Duarte.

TRABAJO

Ejercicios.

PRESENTA

ANA PECH PÉREZ 04190020
MARIANA PÉREZ UCAN 04190022
HENRY TORRES ESQUIVEL 04190031
MANUEL UC NICOLI 04190032
ÁNGEL UICAB CANCHÉ 04190033

Progreso Yucatán, 05 de mayo de 2022.

Actividad 3.3 Ejercicios.

Entregar los **.asm** en formato IBM, un pdf con la portada de los integrantes y capturas de ejecución de los programas, cada programa debe tener en comentarios, el nombre del programador

1. Crea un programa que emplee macros y procedimientos para calcular el promedio de 3 calificaciones de un alumno (3 dígitos máximo) e indique si el alumno esta aprobado o reprobado. Ejemplo: Daniel tiene 70, 80,90, la salida será Daniel esta aprobado con 80.
2. Crea un programa que emplee macros y procedimientos para realizar una calculadora básica (4 operaciones básicas).
3. Crea un programa que emplee macros y procedimientos para leer una cadena y la muestre en una coordenada indicada por el usuario. Validar las coordenadas.
4. Crea un programa que emplee macros y procedimientos para leer una cadena.
5. Desarrollar un programa en ensamblador que permita teclear una vocal (validar que sea solo vocal en mayúscula o minúscula) y mostrarla en forma de texto grande, se terminará cuando se presione la tecla escape.

Objetivo: Resolver problemas reales.

Rubrica:

Rúbrica	Excelente	Muy bien	Regular	Mal
Cantidad y funcionalidad	Todos los ejercicios estuvieron correctos sin errores	Del 90 al 50% de los ejercicios estuvieron correctos con advertencias	Menos del 50% estuvieron correctos con 1 error	No realizó los ejercicios o el 100% esta incorrecto
Entrega	Entrego en tiempo y forma	Entrego en tiempo pero no en forma		No entregó a tiempo y sin forma

INDICE

Código Macro-Procedimiento.asm	4
Código – Ejercicio 1.....	10
Capturas de ejecución.....	14
Código – Ejercicio 2.....	16
Capturas de ejecución.....	25
Código – Ejercicio 3.....	27
Capturas de ejecución.....	30
Código – Ejercicio 4.....	32
Capturas de ejecución.....	40
Código – Ejercicio 5.....	41
Capturas de ejecución.....	44

Código Macro-Procedimiento.asm

;@Manuel Uc Nicoli

;Macros y procedimientos para el uso de las banderas.

TEXTOS SEGMENT

Intro db 'Instituto Tecnologico Superior Progreso',10,13,'\$'

Ejercicio db 10,13,'Ejercicios Ensamblador 3.3',10,13,'\$'

Portada db 10,13,'Ana Pech Perez',10,13,'Mariana Perez Ucan',10,13,'Henry Torres Esquivel',10,13,'Manuel Uc Nicoli',10,13,'Angel Uicab Canche',10,13,'\$'

Fecha db 10,13,'05-Mayo-2022\$'

TEXTOS ENDS

;----- MACROS -----

Mensaje macro txt ;macro para mostrar textos desde una variable del programa.

mov ah,09h

lea dx, txt

int 21h

endm

BanderaSuperior macro ;macro bandera superior.

mov ax,0600h

mov bh,17h;atributos de color fondo y texto

mov CX,0000h;fila inicial en ch, columna inicial en cl

mov dh, 4h

mov dl, 13h

int 10h;ejecuta las interrupciones de video

endm

BanderaInferior macro ;macro bandera inferior.

mov ax,0600h

```

        mov bh,0E7h;atributos de color fondo y texto
mov ch,5h;fila inicial en ch, columna inicial en cl
mov cl,0h
mov dh,9h;fila final en dh, columna final en cl
mov dl,13h
int 10h;ejecuta las interrupciones de video
endm

```

Puntero macro num1 ;macro para poner el puntero en una posicion hacia abajo.

```

        mov ah,02h
        mov dh,num1
        mov dl,0h
        mov bh,0h
        int 10h
endm

```

Puntero2 macro num1,num2 ;macro para poner el puntero en una posicion hacia abajo,derecha.

```

        mov ah,02h
        mov dh,num1
        mov dl,num2 ;sin utilizar un texto con inicio 10,13 en
codigo fuente.
        mov bh,0h
        int 10h
endm

```

Fin macro ;Procedimiento finalizar el programa

```

        mov ah,04ch
        int 21h

```

endm

Introducir macro ;macro para pedir "x" tecla y guardar en una variable al.

mov ah,01h

int 21h ;Coloca la tecla introducida en AL.

endm

LecturaSI macro num1

mov cx,num1

mov si,0

endm

CharsNum macro

mov ah,01h ;Function(character read) Guarda en AL

int 21h ;Interruption DOS functions

sub al,30h ;ajustamos valores

endm

LimpiezaReg macro

xor ax,ax

xor bx,bx

xor cx,cx

xor dx,dx

endm

LimpiezaPantalla macro

mov ax,0600h

mov bh,07h;atributos de color fondo y texto

```
    mov CX,0000h;fila inicial en ch, columna inicial en cl
    mov DX,184fh;fila final en dh, columna final en cl
    int 10h;ejecuta las interrupciones de video
endm
```

EnEspera macro

```
    mov ah,07h
    int 21h
endm
```

PortadaEjer macro

```
    mov ax,textos
    mov ds, ax
    Puntero 6
    Mensaje Intro
    Mensaje Ejercicio
    Mensaje Portada
    Mensaje Fecha
    EnEspera
    LimpiezaPantalla
    puntero 0,0
endm
```

DesempaquetadoResta macro

```
    AAS ;desempaquetado de la suma
    mov cx,ax
    SUB bl,ch
```

```
mov ax,bx
AAS
mov bx,ax
mov ah,02h
mov dl,bh
add dl,30h
int 21h
mov ah,02h
mov dl,bl
add dl,30h
int 21h
mov ah,02h
mov dl,cl
add dl,30h
int 21h
```

EnEspera

endm

DesempaquetadoSuma macro

```
AAM ;desempaquetado de la suma
mov cx,ax
add bl,ch
mov ax,bx
AAM
mov bx,ax
mov ah,02h
mov dl,bh
add dl,30h
```


int 21h

mov ah,02h

mov dl,bl

add dl,30h

int 21h

mov ah,02h

mov dl,cl

add dl,30h

int 21h

EnEspera

endm

Código – Ejercicio 1.

;@Autor: Ana Pech Perez

;Crea un programa que emplee macros y procedimientos para calcular el promedio

;de 3 calificaciones de un alumno (3 dígitos máximo) e indique si el alumno esta aprobado o reprobado.

STACK SEGMENT STACK

DW 64 DUP(?)

STACK ENDS

include 'emu8086.inc' ;Incluye funciones de libreria emu8086

include 'Macro-procedimiento.asm'

;--- PILA ----

DATA SEGMENT

Salto db 10,13,'\$'

NombreA db 10 dup (' '),'\$'

txt1 db 10,13,'Introduzca nombre del alumno: \$'

txt2 db 10,13,'Introduzca la calificacion: \$'

txtA db ' esta aprobado con: \$'

txtR db ' esta reprobado con: \$'

calificacion1 dw ? ;dw va variables de 2 bytes

calificacion2 dw ?

calificacion3 dw ?

resultado dw ?

DATA ENDS

CODE SEGMENT

ASSUME DS:DATA, CS:CODE, SS:STACK

INICIO: PortadaEjer

```
mov AX , DATA
mov DS, AX
```

```
mov ES, AX
```

```
Mensaje txt1
```

```
LecturaSI 10
```

```
EJER3:    Introducir
          cmp al, 13
          je CALIFICACION
          Call Cadena
          Loop EJER3
          jmp CALIFICACION
```

CALIFICACION:

```
Mensaje txt2
```

```
CALL scan_num ; get number in CX.
```

```
MOV calificacion1,cx
```

```
Mensaje txt2
```

```
CALL scan_num ; get number in CX.
```

```
MOV calificacion2,cx
```

```
Mensaje txt2
```

```
CALL scan_num ; get number in CX.
```

```
MOV calificacion3,cx
```

```
LimpiezaReg
```

```
mov ax, calificacion1
add ax, calificacion2
add ax, calificacion3
mov bx, 3
div bx
mov cx,ax
Mensaje Salto
Puntero 10
Mensaje NombreA
cmp cx,70
jb ImprimirReprobado
Mensaje txtA
mov ax,cx
CALL print_num
jmp Final
```

ImprimirReprobado:

```
Mensaje txtR
mov ax,cx
CALL print_num
```

Final: Fin

Cadena proc near

```
mov NombreA[si],al
inc si
```

ret

Cadena endp

DEFINE_SCAN_NUM

;define's con funciones matematicas

DEFINE_PRINT_STRING

DEFINE_PRINT_NUM


DEFINE_PRINT_NUM_UN

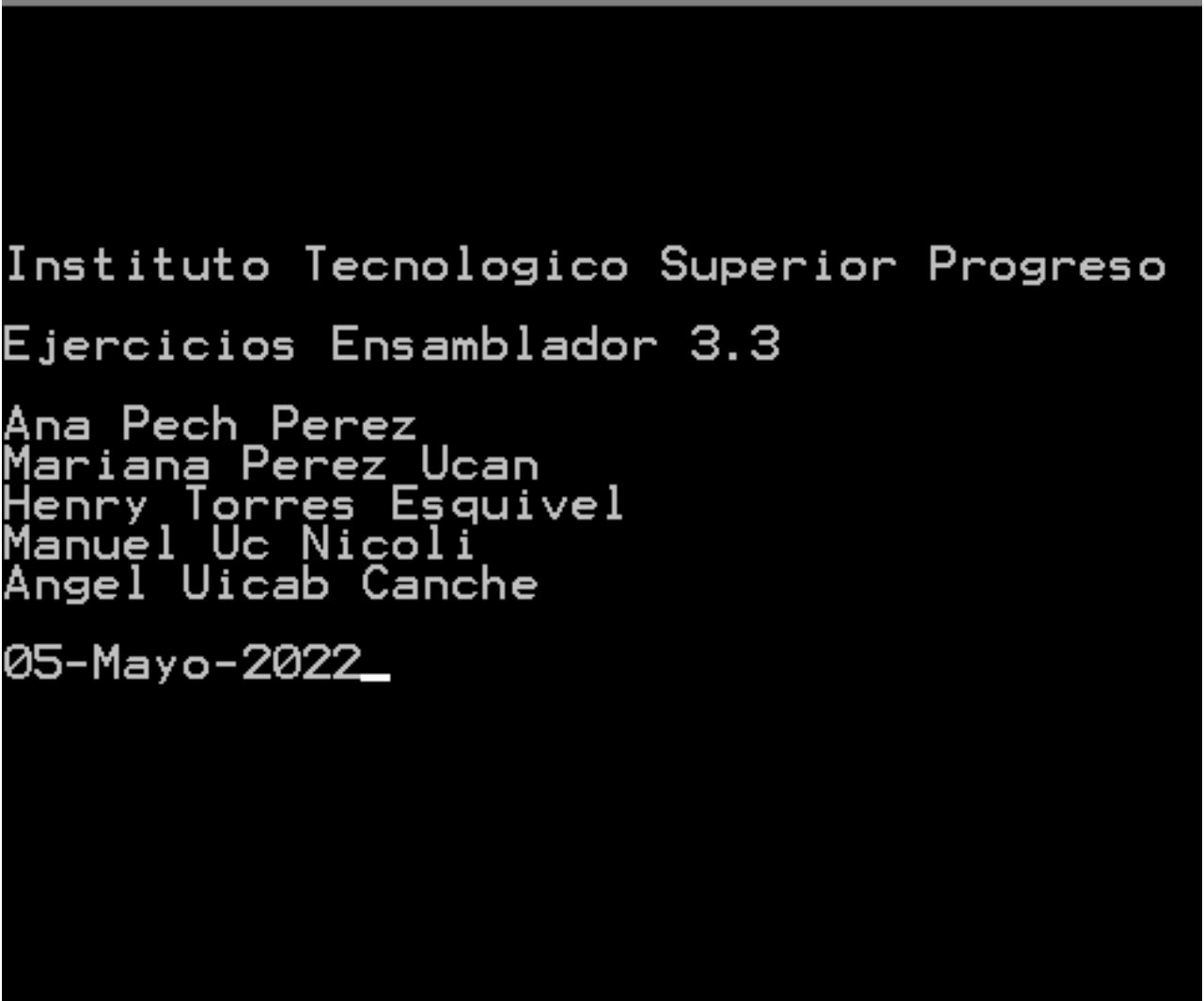
CODE ENDS

ENDS INICIO

Capturas de ejecución.

Todos los ejercicios cuentan con una portada en la cual esta almacenada directamente en el asm donde se encuentran todas nuestras macros adjuntas con el archivo importándolo de manera que usamos en todos los ejercicios el include para demostrar dicha portada en cada uno.

 emulator screen (80x25 chars)



```
Instituto Tecnologico Superior Progreso
Ejercicios Ensamblador 3.3
Ana Pech Perez
Mariana Perez Ucan
Henry Torres Esquivel
Manuel Uc Nicoli
Angel Uicab Canche
05-Mayo-2022_
```

Después que se imprimen cada portada, espera cualquier carácter dentro del teclado para poder continuar con la ejecución del programa.

SCM emulator screen (80x25 chars)

```
Introduzca nombre del alumno: Manuel xd
Introduzca la calificacion: 25
Introduzca la calificacion: 89
Introduzca la calificacion: 100
```

```
Manuel xd esta aprobado con: 71
```

Nos pide por defecto el nombre del estudiante, este con limite establecido de 10 caracteres, por lo consiguiente nos pedirá sus 3 calificaciones, los sumamos y luego hacemos la división correspondiente, limpiando el dx, esto con fin de evitar algún desbordamiento en el registro.

También establecimos como aprobatorio el promedio de 70 o superior de lo contrario, nos mostrara que esta reprobado.

SCM emulator screen (80x25 chars)

```
Introduzca nombre del alumno: doed14ad2d
Introduzca la calificacion: 70
Introduzca la calificacion: 0
Introduzca la calificacion: 70
```

```
doed14ad2d esta reprobado con: 46
```

Después de mostrar el mensaje, automáticamente finaliza el programa.

Código – Ejercicio 2.

;@Autor: Manuel Uc Nicoli

;Crea un programa que emplee macros y procedimientos para realizar una calculadora básica (4 operaciones básicas).

STACK SEGMENT STACK

DW 64 DUP(?)

STACK ENDS

include 'Macro-procedimiento.asm'

include 'Emu8086.inc'

;--- PILA ----

DATA SEGMENT

chr1 db ? ;primer digito

chr2 db ? ;segundo digito

chr3 db ? ;multiplo

chr4 db ?

r1 db ? ;resultado 1

r2 db ? ;resultado 2

r3 db ?

r4 db ?

ac db 0 ;acarreo

ac1 db 0

num1 dw ?

num2 dw ?

txt1 db '!Seleccione una opcion a realizar!\$'

txt3 db '1-Suma 2-Resta 3-Multiplicacion 4-Division 5-Salir\$'

ResultadoM db 10,13,'El resultado de la multiplicacion: \$'

ResultadoS db 10,13,'El resultado de la suma: \$'

ResultadoD db 10,13,'El resultado de la division: \$'


```
ResultadoR db 10,13,'El resultado de la resta: $'  
txt2 db 10,13,'Ingresa un numero (2 digitos max): $'  
DATA ENDS  
;--- DATOS ---
```

```
CODE SEGMENT
```

```
    ASSUME DS:DATA, CS:CODE, SS:STACK
```

```
INICIO: PortadaEjer
```

```
    mov AX , DATA
```

```
    mov DS, AX
```

```
;----- PROCESO MENU CALCULADORA -----
```

```
INICIO2:LimpiezaPantalla
```

```
    Puntero2 0,20
```

```
    Mensaje txt1
```

```
    Puntero2 1,12
```

```
    Mensaje txt3
```

```
    puntero2 2,12
```

```
    Introducir
```

```
    puntero2 4,0
```

```
    cmp al,49
```

```
    je SUM
```

```
    cmp al,50
```

```
    je RES
```

```
    cmp al,51
```

```
    je MULTI
```

```
    cmp al,52
```

```
    je DIVIS
```

```
    cmp al,53
```

je Finalizar
jmp INICIO2

;----- PROCESO SUMA -----

SUM: Mensaje txt2 ;pedir el primer numero de 2 dij

call Num
mov chr1,al ;[chr1].chr2 * chr3 = ac.r1.r2
call Num
mov chr2,al ;[chr1].chr2 * chr3 = ac.r1.r2

Mensaje txt2 ;pedir el segundo numero de 2 dij

call Num
mov chr3,al ;[chr1].chr2 * chr3 = ac.r1.r2
call Num
mov chr4,al ;[chr1].chr2 * chr3 = ac.r1.r2
LimpiezaReg ;Limpieza de registros para empezar la suma
call ProcesoSuma
jmp INICIO2

;----- PROCESO RESTA -----

RES: Mensaje txt2 ;pedir el primer numero de 2 dij

CALL scan_num ; get number in CX.
mov num1,cx

Mensaje txt2 ;pedir el segundo numero de 2 dij

CALL scan_num ; get number in CX.
mov num2,cx

call ProcesoResta

LimpiezaReg

jmp INICIO2

;----- PROCESO MULTIPLICACION -----

MULTI: Mensaje txt2 ;pedir el primer numero de 2 dij

call Num

mov chr1,al ;[chr1].chr2 * chr3 = ac.r1.r2

call Num

mov chr2,al ;[chr1].chr2 * chr3 = ac.r1.r2

Mensaje txt2 ;pedir el segundo numero de 2 dij

call Num

mov chr3,al ;[chr1].chr2 * chr3 = ac.r1.r2

call Num

mov chr4,al

LimpiezaReg

call ProcesoMulti

jmp INICIO2

;----- PROCESO DIVISION -----

DIVIS:Mensaje txt2 ;pedir el primer numero de 2 dij

CALL scan_num ; get number in CX.

mov num1,cx

Mensaje txt2 ;pedir el segundo numero de 2 dij

CALL scan_num ; get number in CX.

mov num2,cx

call ProcesoDivi

LimpiezaReg

```
                jmp INICIO2
Finalizar:      LimpiezaPantalla
                Fin
```

```
Num proc near
    mov ah,01h    ;Function(character read) Guarda en AL
    int 21h       ;Interruption DOS functions
    sub al,30h    ;ajustamos valores
    ret
Num endp
```

```
ProcesoSuma proc near
                ;primero los caracteres se pasan moviendo
                mov bl,chr1
    mov cl,chr2
                mov al,chr3
                ;se pasan los segundos caracteres
    add bl,al
                mov al,chr4
    add cl,al
                Mensaje ResultadoS
                mov ax,cx
                DesempaquetadoSuma
                ret
ProcesoSuma endp
```

```
ProcesoResta proc
                mov ax,num1
```

sub ax,num2

mov bx,ax

Mensaje ResultadoR

mov ax,bx

CALL print_num

EnEspera

ret

ProcesoResta endp

ProcesoMulti proc

;Realizamos operaci?n

mov al,chr4 ;unidad del segundo numero

mov bl,chr2 ;unidad del primer numero

mul bl ;multiplicar

mov ah,0 ;limpiamos ah0

aam ;separamos de hex a dec

mov ac1,ah ;decenas del primera multiplicacion

mov r4,al ;unidades del primera multiplicacion

mov al,chr4 ;unidades del segundo numero

mov bl,chr1 ;decentas del primer numero

mul bl ;multiplicar

mov r3,al ;movemos el resultado de la operacion a r3

mov bl,ac1 ;movemos el acarreo a bl

add r3,bl ;sumamos resultado mas acarreo

mov ah,00h ;limpiamos ah por residuos

mov al,r3 ;movemos el resultado de la suma a al

aam ;separamos de hex a dec

mov r3,al ;guardamos unidades en r3

```

mov ac1,ah ;guardamos decenas en ac1
mov al,chr3 ;al = chr3
mov bl,chr2 ;bl = chr2
mul bl ;AL = chr3*chr2 (BL*AL)
mov Ah,0h ;
AAM ;ASCII Adjustment
mov ac,AH ;ac = AH (Acarreo)
mov r2,AL ;r2 = AL (Unidad del resultado)
mov al,chr3 ;AL = chr3
mov bl,chr1 ;BL = chr1
mul bl ;AL = chr1*chr3 (BL*AL)
mov r1,al ;r1 = AL (Decena del resultado)
mov bl,ac ;BL = Acarreo anterior
add r1,bl ;r1 = r1+ac (r1 + Acarreo)
mov ah,00h ;
mov al,r1 ;AL = r1 (Asignaci?n para el ajust)
AAM ;ASCII Adjustment
mov r1,al ;r1 = AL
mov ac,ah ;ac = AH (Acarreo para la Centena del resultado)

```

;suma final

;R4 resulta ser las unidades de mul y no se toma en cuenta ya que se pasa entero

```

mov ax,0000h ;limpiamos ax

```

```

mov al,r3 ;movemos el segundo resultado de la primera mult a al

```

```

mov bl,r2 ;movemos primer resultado de la segunda mult a bl

```

```

add al,bl ;sumamos

```

```

mov ah,00h ;limpiamos ah

```

```

aam ;separamos hex a dec

```

```

mov r3,al    ;r3 guarda las decenas del resultado final
mov r2,ah    ;r2 se utiliza como nuevo acarreo
mov ax,ax    ;""
mov al,ac1   ;movemos el acarreo de la primera mult a al
mov bl,r1    ;movemos segundo resultado de la segunda mult a bl
add al,r2    ;sumamos el nuevo acarreo de la suma anterior a al
add al,bl    ;sumamos al a bl
mov ah,00h   ;limpiamos el registro ah
aam         ;separamos de hex a dec
mov r1,al    ;r1 guarda las centenas
mov r2,ah    ;ah se sigue utilizando como acarreo
mov al,r2    ;movemos el acarreo a al
mov bl,ac    ;movemos ac a bl
add al,bl    ;sumamos al a bl
;aam        ;separamos hex a dec
mov ac,al    ;mov al a ac como nuestro acarreo final
;Mostramos resultado
Mensaje ResultadoM
mov ah,02h
mov dl,ac
add dl,30h
int 21h     ;Mostramos ac (millar)
mov ah,02H
mov dl,r1
add dl,30h
int 21h     ;Mostramos r1 (centena)
mov ah,02H
mov dl,r3

```

```
    add dl,30h
    int 21h    ;Mostramos r3 (decena)
    mov ah,02H
    mov dl,r4
    add dl,30h
    int 21h    ;unidad
    EnEspera
    ret
```

ProcesoMulti endp

ProcesoDivi proc

```
    Mensaje ResultadoD
    xor dx,dx
    mov ax, num1
    mov bx, num2
    div bx
    CALL  print_num
    EnEspera
    ret
```

ProcesoDivi endp

DEFINE_SCAN_NUM ;define's con funciones matematicas

DEFINE_PRINT_STRING

DEFINE_PRINT_NUM

DEFINE_PRINT_NUM_UN

CODE ENDS

ENDS INICIO

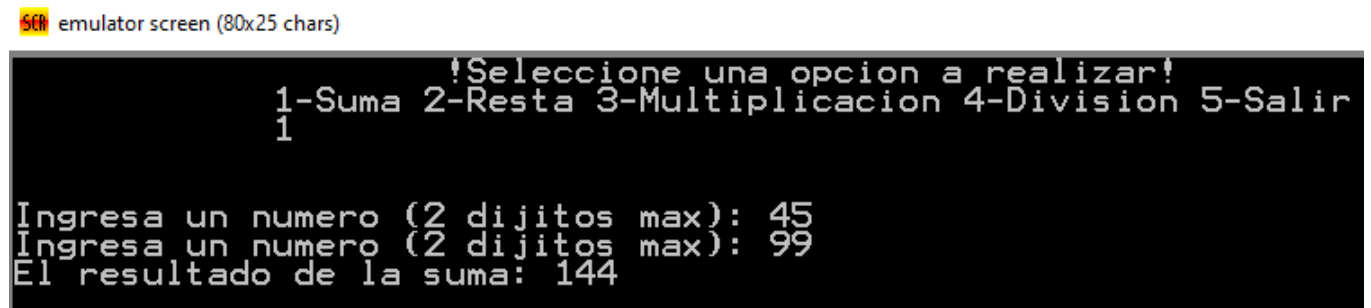
Capturas de ejecución.

En este programa se creó un pequeño menú el cual funciona de la siguiente manera por lo que solo se puede finalizar de manera que se seleccione la opción 5. Este ejercicio fue hecho de manera híbrida incluyendo ajustes y la librería de EMU8086



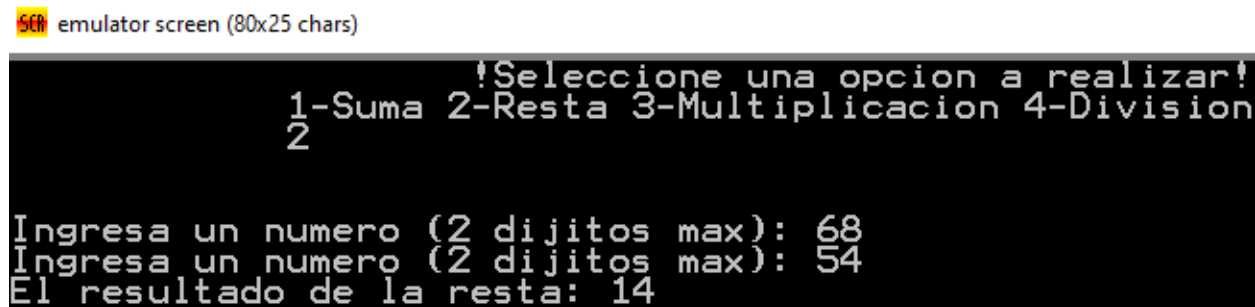
```
emulator screen (80x25 chars)
!Seleccione una opcion a realizar!
1-Suma 2-Resta 3-Multiplicacion 4-Division 5-Salir
```

A continuación, se mostrará como se dan los resultados de la suma, mediante un proceso de desempaquetado AAA, por lo que el proceso de obtener el resultado es de una manera larga.



```
emulator screen (80x25 chars)
!Seleccione una opcion a realizar!
1-Suma 2-Resta 3-Multiplicacion 4-Division 5-Salir
1
Ingresa un numero (2 digitos max): 45
Ingresa un numero (2 digitos max): 99
El resultado de la suma: 144
```

Se queda esperando una tecla proveniente del teclado para que pueda seguir ejecutándose el programa y nos permita visualizar de manera satisfactoria el resultado antes de que retorne al menú de inicio.



```
emulator screen (80x25 chars)
!Seleccione una opcion a realizar!
1-Suma 2-Resta 3-Multiplicacion 4-Division
2
Ingresa un numero (2 digitos max): 68
Ingresa un numero (2 digitos max): 54
El resultado de la resta: 14
```

En la resta se utilizo la manera en la que se usan librería de EMU para poder ejecutarlo en forma que las entradas de dígitos es en 16 bits dw por lo que se almacena en cx, lo cual tenemos que moverlo para poder imprimirlo con esta librería en forma de AX.

```
SCR emulator screen (80x25 chars)

!Seleccione una opcion a realizar!
1-Suma 2-Resta 3-Multiplicacion 4-Division
3

Ingresa un numero (2 digitos max): 58
Ingresa un numero (2 digitos max): 87
El resultado de la multiplicacion: 5046
```

En la multiplicación utilizamos la manera de desempaqueado el cual es AAM por lo que el proceso es demasiado largo por lo que tendremos que esperar unos segundos que nos muestre el resultado.

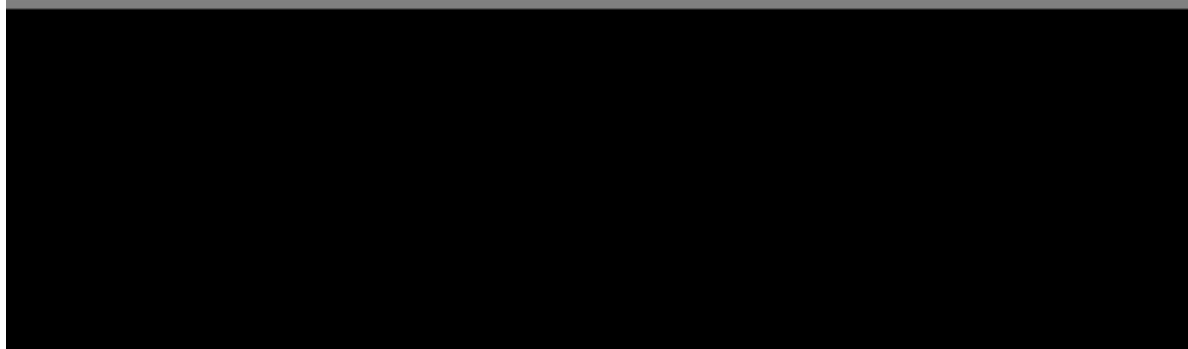
```
SCR emulator screen (80x25 chars)

!Seleccione una opcion a realizar!
1-Suma 2-Resta 3-Multiplicacion 4-Division
4

Ingresa un numero (2 digitos max): 78
Ingresa un numero (2 digitos max): 4
El resultado de la division: 19
```

La división fue en forma en cuestión que se utilizó la librería de EMU por lo que se hizo de la misma manera que la resta, debido que solo nos mostrara de forma entera la división sin el residuo.

```
SCR emulator screen (80x25 chars)
```



Al finalizar solo nos mostrara una pantalla en negro, indicando su finalización.

Código – Ejercicio 3.

;@Autor: Mariana Perez Ucan

;Crea un programa que emplee macros y procedimientos para leer una cadena

;y la muestre en una coordenada indicada por el usuario. Validar las coordenadas.

STACK SEGMENT STACK

DW 64 DUP(?)

STACK ENDS

include "Macro-procedimiento.asm"

;--- PILA ----

DATA SEGMENT

Cadena1 db 10 dup (' '), '\$'

Var1 db 0

Var2 db 0

txt1 db 10,13,'Introduzca su palabra',10,13,'\$'

txt2 db 10,13,'Indique su cordenada "Y"',10,13,'\$'

txt3 db 10,13,'Indique su cordenada "X"',10,13,'\$'

txt4 db 10,13,'Cordenada fuera de lugar, intente de nuevo!',10,13,'\$'

txt5 db 10,13,'Programa finalizado!\$'

DATA ENDS

;--- DATOS ---

CODE SEGMENT

ASSUME DS:DATA, CS:CODE, SS:STACK

INICIO: PortadaEjer

mov AX , DATA

mov DS, AX

INICIO2:Mensaje txt1

LecturaSI 10

EJER3: Introducir
 cmp al, 13
 je CORDENADAS
 Call Cadena
 Loop EJER3
 jmp CORDENADAS

CORDENADAS: Mensaje txt2 ;Validar cordenada Y

 Introducir
 cmp al,'9'
 ja RETORN
 cmp al,'0'
 jb RETORN
 sub al,48

 mov dl,al
 mov Var1,dl

 Mensaje txt3 ;Validar cordenada X

 Introducir
 cmp al,'9'
 ja RETORN
 cmp al,'0'
 jb RETORN
 sub al,48

 mov dl,al
 mov Var2,dl

 Call LimpiarP

Puntero2 Var1,Var2 ;Puntero

Mensaje Cadena1

Mensaje txt5

Fin

RETORN: Mensaje txt4

jmp INICIO2

Cadena proc near

mov cadena1[si],al

inc si

ret

Cadena endp

LimpiarP proc near ;limpiar pantalla.

;funcion para hacer scroll tambien con 7h

mov ax,0600h

mov bh,07h;atributos de color fondo y texto

mov CX,0000h;fila inicial en ch, columna inicial en cl

mov DX,184fh;fila final en dh, columna final en cl

int 10h;ejecuta las interrupciones de video

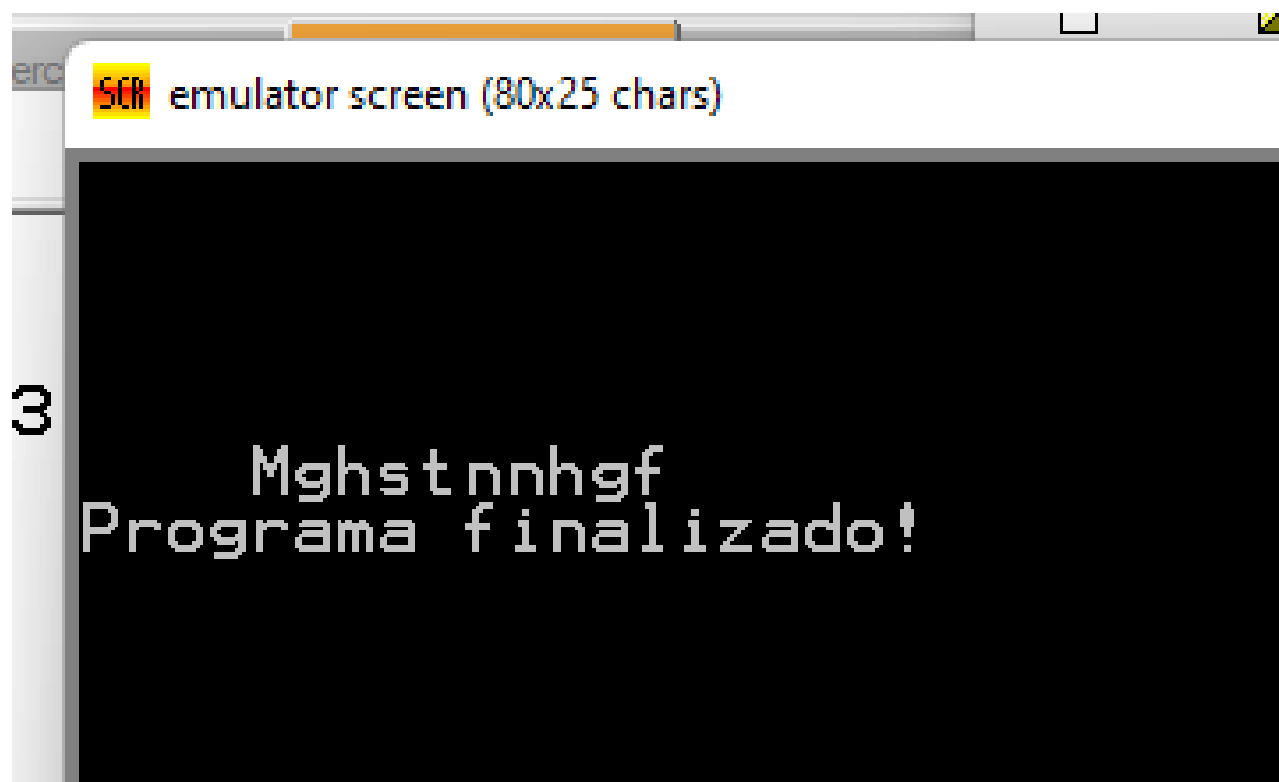
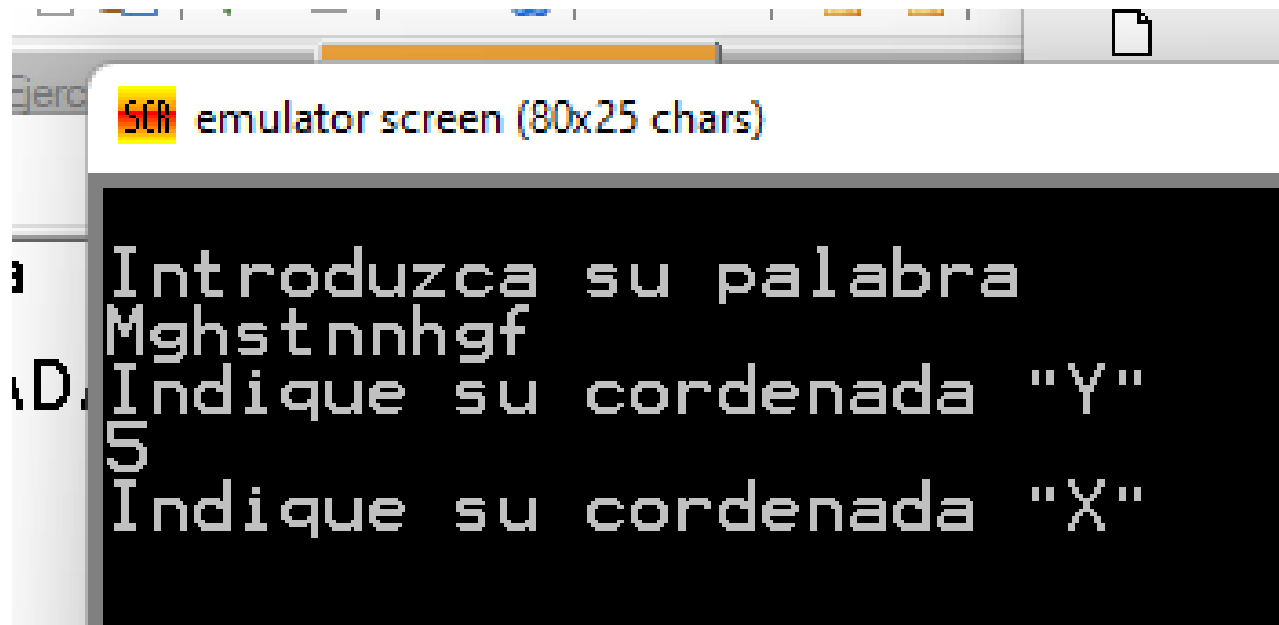
ret

limpiarP endp

CODE ENDS

ENDS INICIO

Capturas de ejecución.



```
emulator screen (80x25 chars)

s: Introduzca su palabra
0: verhytrsrh
Indique su cordenada "Y"
n: A
Cordenada fuera de lugar, intente de nuevo!
A: Introduzca su palabra
D: _
```

```
emulator screen (80x25 chars)

s: Introduzca su palabra
0: verhytrsrh
Indique su cordenada "Y"
n: A
Cordenada fuera de lugar, intente de nuevo!
A: Introduzca su palabra
1: ighjnd
D: Indique su cordenada "Y"
4: Indique su cordenada "X"
9: Cordenada fuera de lugar, intente de nuevo!
Indroduzca su palabra
_
```

Código – Ejercicio 4.

;Autor @Henry Torres Esquivel

;Desarrollar un programa en ensamblador que permita leer

;dos números de 2 dígitos y realice la suma y multiplicación de esos números

STACK SEGMENT STACK

DW 64 DUP(?)

STACK ENDS

;--- PILA ----

DATA SEGMENT

chr1 db ? ;primer digito

chr2 db ? ;segundo digito

chr3 db ? ;multiplo

chr4 db ?

r1 db ? ;resultado 1

r2 db ? ;resultado 2

r3 db ?

r4 db ?

ac db 0 ;acarreo

ac1 db 0

resultado db 10,13,'El resultado de la multiplicacion: \$'

result db 10,13,'El resultado de la suma: \$'

txt db 10,13,'Ingresa un numero \$'

DATA ENDS

;--- DATOS ---

CODE SEGMENT

ASSUME DS:DATA, CS:CODE, SS:STACK

INICIO: mov AX, DATA

mov DS, AX

mov ah,09h

lea dx, txt

int 21h

mov ah,01h ;Function(character read) Guarda en AL

int 21h ;Interruption DOS functions

sub al,30h ;ajustamos valores

mov chr1,al ;[chr1].chr2 * chr3 = ac.r1.r2

mov ah,01h ;Function(character read) Guarda en AL

int 21h ;Interruption DOS functions

sub al,30h ;Ajustamos valores

mov chr2,al ;chr1.[chr2] * chr3 = ac.r1.r2

mov ah,09h

lea dx, txt

int 21h

mov ah,01h ;Function(Read character) Guarda en AL

int 21h ;Interruption DOS Functions

sub al,30h ;Transform(0dec = 30hex)

mov chr3,al ;chr1.chr2 * [chr3] = ac.r1.r2

mov ah,01h ;Function(Read character) Guarda en AL

```
int 21h    ;Interruption DOS Functions
sub al,30h  ;Transform(0dec = 30hex)
mov chr4,al ;chr1.chr2 * [chr3] = ac.r1.r2
```

```
;Realizamos operaci?n
```

```
mov al,chr4 ;unidad del segundo numero
mov bl,chr2 ;unidad del primer numero
mul bl      ;multiplicar
mov ah,0    ;limpiamos ah0
aam        ;separamos de hex a dec
mov ac1,ah  ;decenas del primera multiplicacion
mov r4,al   ;unidades del primera multiplicacion
```

```
mov al,chr4 ;unidades del segundo numero
mov bl,chr1 ;decentas del primer numero
mul bl      ;multiplicar
mov r3,al   ;movemos el resultado de la operacion a r3
mov bl,ac1  ;movemos el acarreo a bl
add r3,bl   ;sumamos resultado mas acarreo
mov ah,00h  ;limpiamos ah por residuos
mov al,r3   ;movemos el resultado de la suma a al
aam        ;separamos de hex a dec
mov r3,al   ;guardamos unidades en r3
mov ac1,ah  ;guardamos decenas en ac1
```

```
mov al,chr3 ;al = chr3
mov bl,chr2 ;bl = chr2
mul bl      ;AL = chr3*chr2 (BL*AL)
```

```
mov Ah,0h    ;  
AAM          ;ASCII Adjustment  
mov ac,AH    ;ac = AH (Acarreo)  
mov r2,AL    ;r2 = AL    (Unidad del resultado)
```

```
mov al,chr3   ;AL = chr3  
mov bl,chr1   ;BL = chr1  
mul bl        ;AL = chr1*chr3 (BL*AL)  
mov r1,al     ;r1 = AL    (Decena del resultado)  
mov bl,ac     ;BL = Acarreo anterior  
add r1,bl     ;r1 = r1+ac (r1 + Acarreo)  
mov ah,00h    ;  
mov al,r1     ;AL = r1 (Asignaci?n para el ajust)  
AAM          ;ASCII Adjustment  
mov r1,al     ;r1 = AL  
mov ac,ah     ;ac = AH (Acarreo para la Centena del resultado)
```

```
;suma final
```

```
;R4 resulta ser las unidades de mul y no se toma en cuenta ya que se pasa entero
```

```
mov ax,0000h  ;limpiamos ax
```

```
mov al,r3     ;movemos el segundo resultado de la primera mult a al  
mov bl,r2     ;movemos primer resultado de la segunda mult a bl  
add al,bl     ;sumamos  
mov ah,00h    ;limpiamos ah  
aam          ;separamos hex a dec  
mov r3,al     ;r3 guarda las decenas del resultado final  
mov r2,ah     ;r2 se utiliza como nuevo acarreo
```

```
mov ax,ax ;""
```

```
mov al,ac1 ;movemos el acarreo de la primera mult a al
```

```
mov bl,r1 ;movemos segundo resultado de la segunda mult a bl
```

```
add al,r2 ;sumamos el nuevo acarreo de la suma anterior a al
```

```
add al,bl ;sumamos al a bl
```

```
mov ah,00h ;limpiamos el registro ah
```

```
aam ;separamos de hex a dec
```

```
mov r1,al ;r1 guarda las centenas
```

```
mov r2,ah ;ah se sigue utilizando como acarreo
```

```
mov al,r2 ;movemos el acarreo a al
```

```
mov bl,ac ;movemos ac a bl
```

```
add al,bl ;sumamos al a bl
```

```
;aam ;separamos hex a dec
```

```
mov ac,al ;mov al a ac como nuestro acarreo final
```

```
;Mostramos resultado
```

```
mov ah,09h
```

```
lea dx, resultado
```

```
int 21h
```

```
mov ah,02h
```

```
mov dl,ac
```

```
add dl,30h
```

```
int 21h ;Mostramos ac (millar)
```

```
mov ah,02H
mov dl,r1
add dl,30h
int 21h    ;Mostramos r1 (centena)
```

```
mov ah,02H
mov dl,r3
add dl,30h
int 21h    ;Mostramos r3 (decena)
```

```
mov ah,02H
mov dl,r4
add dl,30h
int 21h    ;unidad
```

;----- Proceso suma -----

```
xor ax,ax
```

```
xor bx,bx
```

```
xor cx,cx
```

```
xor dx,dx
```

```
;primero los caracteres se pasan moviendo
```

```
mov bl,chr1
```

```
mov cl,chr2
```

```
mov al,chr3
```

```
;se pasan los segundos caracteres
```

```
add bl,al
```

```
        mov al,chr4
add cl,al
```

```
        mov ah,09h
        lea dx,result
        int 21h
```

```
        mov ax,cx
AAM ;desempaquetado de la suma
mov cx,ax
```

```
add bl,ch
mov ax,bx
```

```
AAM
mov bx,ax
```

```
mov ah,02h
mov dl,bh
add dl,30h
int 21h
```

```
mov ah,02h
mov dl,bl
add dl,30h
int 21h
```

```
mov ah,02h
```

```
mov dl,cl
```

```
add dl,30h
```

```
int 21h
```

```
CODE ENDS
```

```
END INICIO
```

Capturas de ejecución.

SCR emulator screen (80x25 chars)

```
Ingresa un numero 67
Ingresa un numero 07
El resultado de la multiplicacion: 0469
El resultado de la suma: 074
```

SCR emulator screen (80x25 chars)

```
Ingresa un numero 21
Ingresa un numero 21
El resultado de la multiplicacion: 0441
El resultado de la suma: 042
```

SCR emulator screen (80x25 chars)

```
Ingresa un numero 99
Ingresa un numero 99
El resultado de la multiplicacion: 9801
El resultado de la suma: 198
```

SCR emulator screen (80x25 chars)

```
Ingresa un numero 02
Ingresa un numero 00
El resultado de la multiplicacion: 0000
El resultado de la suma: 002
```


Código – Ejercicio 5.

;@Autor: Angel Uicab Canche

;Desarrollar un programa en ensamblador que permita teclear una vocal (validar que sea solo vocal en mayúscula o minúscula)

;y mostrarla en forma de texto grande, se terminará cuando se presione la tecla escape.

STACK SEGMENT STACK

DW 64 DUP(?)

STACK ENDS

include "Macro-procedimiento.asm"

;--- PILA ----

DATA SEGMENT

txt1 db 10,13,'Introduzca un caracter',10,13,'\$'

txt2 db 10,13,'Caracter incorrecto',10,13,'\$'

txt3 db 10,13,'Programa finalizado\$'

vocalA db 10,13,'#####',10,13,'### ###',10,13,'### ###',10,13,'###
###',10,13,'#####',10,13,'### ###',10,13,'### ###',10,13,'### ###',10,13,'### ###',10,13,'\$'

vocalE db
10,13,'#####',10,13,'#####',10,13,'###',10,13,'###',10,13,'###',10,13,'#####',10,13,'##
#####',10,13,'###',10,13,'###',10,13,'###',10,13,'#####',10,13,'#####',10,13,'\$'

vocalI db 10,13,'#####',10,13,'#####',10,13,' ### ',10,13,' ### ',10,13,' ### ',10,13,' ###
,10,13,' ### ',10,13,'#####',10,13,'#####',10,13,'\$'

vocalO db 10,13,'##### ',10,13,'# #',10,13,'# #',10,13,'# #',10,13,'# #',10,13,'# #',10,13,'#
#',10,13,'# #',10,13,'##### ',10,13,'\$'

vocalU db 10,13,'### ###',10,13,'### ###',10,13,'### ###',10,13,'### ###',10,13,'###
###',10,13,'##### ',10,13,'##### ',10,13,'\$'

DATA ENDS

;--- DATOS ---

CODE SEGMENT

ASSUME DS:DATA, CS:CODE, SS:STACK

INICIO: PortadaEjer

mov AX , DATA

mov DS, AX

VOCALES:

Mensaje txt1 ;macro creada para este ejercicio

Introducir ;macro para pedir tecla

cmp al, 'A'

je Va

cmp al, 'a'

je Va

cmp al, 'E'

je Ve

cmp al, 'e'

je Ve

cmp al, 'I'

je Vi

cmp al, 'i'

je Vi

cmp al, 'O'

je Vo

cmp al, 'o'

je Vo

cmp al, 'U'

je Vu

cmp al, 'u'

je Vu

cmp al, 32 ;32 espacio en decimal

je fin ;finaliza si es espacio

Mensaje txt2 ;no se encontro caracter

jmp VOCALES ;se inicia el programa

Va: Mensaje vocalA

jmp VOCALES

Ve: Mensaje vocalE

jmp VOCALES

Vi: Mensaje vocalI

jmp VOCALES

Vo: Mensaje vocalO

jmp VOCALES

Vu: Mensaje vocalU

jmp VOCALES

FIN: Mensaje txt3

Fin

CODE ENDS

ENDS INICIO

Capturas de ejecución.

The screenshot shows a C++ program in an IDE. The code defines a function `getChar` that repeatedly prompts the user to enter a character until a valid one is provided. The `main` function calls `getChar` five times, storing the results in an array `array`. The IDE interface includes a toolbar at the top with icons for file operations, a menu bar with options like File, Edit, and Run, and a status bar at the bottom showing '0/16'.

```
#include <iostream>
using namespace std;

char getChar()
{
    char c;
    do
    {
        cout << "Introduzca un caracter\n";
        c = getche();
        if (c != '\n')
            cout << "Caracter incorrecto\n";
    } while (c == '\n');
    return c;
}

int main()
{
    char array[5];
    for (int i = 0; i < 5; i++)
    {
        array[i] = getChar();
    }
    return 0;
}
```