**PARTE 1 - Repaso VonSim  
01 - Contar letras ⭐ Escribir un programa que dado un string llamado MENSAJE, almacenado en la memoria de datos, cuente la cantidad de veces que la letra ‘a’ (minúscula) aparece en MENSAJE y lo almacene en la variable CANT. Por ejemplo, si MENSAJE contiene “Hola, Buenas Tardes”, entonces CANT debe valer 3**

ORG 1000h

LETRA DB "a"

MENSAJE DB "Hola, Buenas Tardes"

CANT DB ?

ORG 2000h

MOV BX, OFFSET MENSAJE

MOV CL, OFFSET CANT - OFFSET MENSAJE

MOV DH, LETRA

MOV AL, 0

LAZO: CMP DH, [BX]

JNZ SALTO

INC AL

SALTO: INC BX

DEC CL

JNZ LAZO

MOV CANT, AL

HLT

END

**02 - Es mayúscula ⭐ Escribir un programa que determine si un carácter (un string de longitud 1) es una letra mayúscula. El carácter está almacenado en la variable C, y el resultado se guarda en la variable RES de 8 bits. Si C es mayúscula, debe almacenarse el valor 0FFh en RES; de lo contrario, debe almacenarse 0. Pista: Los códigos de las mayúsculas son todos consecutivos. Buscar en la tabla ASCII los caracteres mayúscula, y observar qué valores ASCII tienen la ‘A’ y la ‘Z’.**

ORG 1000H  
C DB “A”  
RES DB ?

ORG 2000H  
 MOV AL, C  
 CMP AL, “A”; Compara el carácter con la primera mayúscula  
 JS NO\_ES\_MAYUS; Si esa comparación tiene signo, la letra esta antes en la tabla ASCII y no es mayúsculas  
 CMP AL, “Z”; Compara el carácter con la última mayúscula  
 JNS NO\_ES\_MAYUSCULA; Si la comparación es positiva, significa que el carácter es mas grande que Z y excede el rango de las mayúsculas

MOV RES, 0FFH  
 JMP FIN  
  
NO\_ES\_MAYUS: MOV AL, 0  
FIN: HLT  
 END

**03 - Convertir carácter a minúscula ⭐ Escribir un programa que convierta un carácter de mayúsculas a minúsculas. El carácter está almacenado en la variable C. Asumir que el carácter es una mayúscula. Pista: Las mayúsculas y las minúsculas están en el mismo orden en el ASCII, y por ende la distancia entre, por ejemplo, la “A” y la “a” es la misma que la distancia entre la “Z” y la “z”.**

ORG 1000H

C DB “A”

ORG 2000H

MOV AL, C  
 ADD AL, 20H// Entre “A” y “a” hay 32 caracteres. 32 =20H  
 MOV C, AL

HLT  
 END

**04 - Convertir string a minúscula ⭐ Escribir un programa que convierta todos los carácteres de un string MENSAJE a minúscula. Por ejemplo, si MENSAJE contiene “Hola, Buenas Tardes”, luego de ejecutar el programa debe contener “hola, buenas tardes”.**

ORG 1000H

MENSAJE DB “Hola, Buenas Tardes”

FIN\_MSJ DB ?

ORG 2000H

MOV BX, OFFSET MENSAJE  
 MOV CL, OFFSET FIN\_MSJ – OFFSET MENSAJE  
LAZO: MOV DH, [BX]  
 CMP DH, ‘A’  
 JS NO\_ES\_MAY  
 CMP DH, ‘Z’  
 JNS NO\_ES\_MAY  
 MOV AL, BYTE PTR [BX]  
 ADD AL, 20H  
 MOV [BX], AL  
 INC BX  
 DEC CL  
 JNZ LAZO  
 JMP FIN  
NO\_ES\_MAY: INC BX  
 DEC CL  
 JNZ LAZO  
 JMP FIN  
FIN: HLT  
 END

**PARTE 2-E/S con interrupciones por software**

Ejercicio 1

ORG 1000H  
MSJ DB "ARQUITECTURA DE COMPUTADORAS-"  
DB "FACULTAD DE INFORMATICA-"  
DB 55H  
DB 4EH  
DB 4CH  
DB 50H  
FIN DB ?

ORG 2000H  
MOV BX, OFFSET MSJ  
MOV AL, OFFSET FIN - OFFSET MSJ  
INT 7  
INT 0  
END  
  
El programa imprime, ARQUITECTURA DE COMPUTADORAS- FACULTAD DE INFORMATICA-UNLP.  
Porque se guarda el inicio de la cadena en Bx y la longitud en Al y luego se ejecuta INT 7.  
Escribe UNLP porque los valores 55h, 4Eh, 4Ch y 50h reprensental los caracteres UNLP en la tabla ASCII.

Ejercicio 2 – Lectura desde el teclado

ORG 1000H  
MSJ DB “INGRESE UN NUMERO:”  
FIN DB ?  
  
 ORG 1500H  
NUM DB ?  
  
 ORG 2000H  
 MOV BX, OFFSET MSJ  
 MOV AL, OFFSET FIN – OFFSET MSJ  
 INT 7  
 MOV BX, OFFSET NUM  
 INT 6  
 MOV AL, 1  
 INT 7  
 MOV CL, NUM  
 INT 0  
 END  
  
Con referencia a la INT 6 se debe guardar antes en el registro BX, la dirección de memoria en donde se va a almacenar el dato ingresado desde teclado.  
INT 7 imprime el mensaje INGRESE UN NUMERO, luego de que el numero es ingresado, se llama nuevamente a INT 7 previo a haber cambiado los valores BX y AL para que pueda imprimir en pantalla el valor ingresado.  
En CL queda almacenado el valor ingresado por teclado.

Ejercicio 3 – Errores al ingresar e imprimir caracteres  
  
Programa 1

ORG 1000H  
A DB “HO LA”  
B DB ?

ORG 2000H  
 MOV BX, OFFSET A  
 MOV AL, 4  
 INT 7  
 END  
  
Al programa le falta la INT 0 para detener el programa. Además cuando se pone en el registro AL la cantidad de caracteres se lo hace de manera incorrecta ya que la cantidad de caracteres de A es igual a 5. Se debería en lugar de 4, la sentencia OFFSET B – OFFSET A para que el programa pueda calcularlo.

Programa 2

ORG 1000H  
A DB “ARQ”  
B DB ?

ORG 200H  
 MOV AL,3  
 MOV BX, A  
 INT 7  
 END

El programa no realiza la INT 0 para que se produzca la detención. También es erróneo pasar en BX el valor de A, lo que hay que pasarle a BX es la dirección de memoria de A.  
  
Programa 3

ORG 1000H  
A DB “HOLA”  
B DB ?

ORG 2000H

MOV AL, OFFSET A-OFFSET B  
MOV BX, OFFSET A  
INT 7  
END

El programa esta pasando mal los valores de la cantidad de caracteres que deben guardarse en AL. Debe ser OFFSET B - OFFSET A. Esta faltando la instrucción INT 0 para indicar la finalización del programa.

Programa 4

ORG 1000H

A DB ?

ORG 2000H  
MOV AL, 3  
MOV BX, A  
INT 6  
END

En el registro BX debe guardarse la dirección en memoria donde se va a almacenar el carácter leído (MOV BX, OFFSET A). Además, falta la instrucción INT 0. Parece innecesario la sentencia MOV AL,3.  
  
Programa 5

ORG 1000H

A DB ?

ORG 2000H

INT 6  
 MOV BX, OFFSET A  
 END

El problema principal es que antes de llamar a la interrupción, se debe guardar la dirección de memoria donde se almacenara el carácter ingresado. También falta la instrucción de cierre INT 0.-  
  
Programa 6

ORG 1000H

A DB ?

ORG 2000H

MOV BX, A  
 INT 6  
 MOV AL, 1  
 INT 7  
 END

El error está en el dato que se ingresa en BX, se debería guardar la dirección de memoria (OFFSET A) y no A, antes de llamar la INT 6. Falta cerrar con INT 0.

Ejercicio 4 – Mostrar caracteres individuales ⭐  
a) Escribir un programa que muestre en pantalla las letras mayúsculas (“A” a la “Z”).  
Pista: Podes buscar los códigos de la “A” y la “Z” en una tabla de códigos ASCII. No utilizar un vector.  
Usar una sola variable de tipo db, e incrementar el valor de esa variable antes de imprimir.

b) ¿Qué deberías modificar en a) para mostrar los dígitos (“0” al “9”)? ¿Y para mostrar todos los caracteres disponibles en el código ASCII? Probar en el simulador.

c) Modificar el ejercicio b) que muestra los dígitos, para que cada dígito se muestre en una línea separada.  
Pista: El código ASCII del carácter de nueva línea es el 10, comúnmente llamado “\n” o LF (“line feed” por sus siglas en inglés y porque se usaba en impresoras donde había que “alimentar” una nueva línea).

A)

ORG 1000H  
LETRA DB ”A”

ORG 2000H

MOV BX, OFFSET LETRA  
 MOV AL, 1  
IMPRIMIR: INT 7  
 INC BYTE PTR [BX]  
 MOV CL, [BX]  
 CMP CL, 5B lo comparo con el valor que le sigue a Z en la tabla ASCII para poder hacer el salto después.  
 JS IMPRIMIR  
 INT 0  
 END  
  
Para mostrar de cero a 0 debería cambiar el valor de LETRA a él valor que corresponde al ‘0’ en ASCII y comprarlo con el valor que le sigue al ‘9’ en la tabla.

b) ORG 1000H  
LETRA DB ”0”

ORG 2000H

MOV BX, OFFSET LETRA  
 MOV AL, 1  
IMPRIMIR: INT 7  
 INC BYTE PTR [BX]  
 MOV CL, [BX]  
 CMP CL, ‘9’ +1 lo comparo con el valor que le sigue a 9 en la tabla ASCII para poder hacer el salto después.  
 JS IMPRIMIR  
 INT 0

c)

ORG1000H  
CARÁCTER DB ‘0’  
 DB 0AH ---> valor ascii del salto de línea

ORG 2000H  
 MOV BX, OFFSET CARACTER  
 MOV AL, 2  
IMPRIMIR: INT 7  
 INC BYTE PTR [BX]  
 MOV CL, [BX]  
 CMP CL, ‘9’ + 1  
 JS IMPRIMIR  
 INT 0  
 END

Ejercicio 5 - Acceso con contraseña ⭐⭐   
Escribir un programa que solicite el ingreso de una contraseña de 4 caracteres por teclado, sin visualizarla en pantalla. En caso de coincidir con una clave predefinida (y guardada en memoria) que muestre el mensaje "Acceso permitido"; caso contrario mostrar el mensaje "Acceso denegado", y volver a pedir que se ingrese una contraseña. Al 5to intento fallido, debe mostrarse el mensaje “Acceso BLOQUEADO” y terminar el programa.

ORG 1000H

MSJ\_ING DB "INGRESE LA CLAVE:", 0AH  
MSJ\_PERM DB "ACCESO PERMITIDO"  
MSJ\_DENE DB "ACCESO DENEGADO"  
MSJ\_BLO DB "ACCESO BLOQUEDO"  
FIN\_MSJ DB ?

ORG 1500H

CLAVE DB "A5HX"  
CLAVE\_ING DB ?

ORG 2000H  
MOV BX, OFFSET MSJ\_ING  
MOV AL, OFFSET MSJ\_CLAVE - OFFSET MSJ\_ING  
INT 7  
MOV CH, 5; CANTIDAD INTENTOS

RESET: MOV CL, 4; CANTIDAD CARACETERES  
MOV BX, OFFSET CLAVE\_ING

LOOP: MOV AL, 1  
INT 6  
INC BX  
DEC CL  
JNZ LOOP

MOV CL, 4; CANTIDAD DE COMPARACIONES  
MOV BX, OFFSET CLAVE\_ING

COMPARA: MOV AL, [BX] ; Cargar un carácter de CLAVE\_ING en AL  
MOV AH, AL; PONGO EN DL LO QUE HAY EN AL  
MOV DX, BX; GUARDO EL PUNTERO TEMPORALMENTE  
SUB BX, OFFSET CLAVE\_ING; Calcular la posición relativa  
ADD BX, OFFSET CLAVE; Apuntar BX al carácter en CLAVE  
MOV AL, [BX]; Cargar el carácter correspondiente de CLAVE en AL  
CMP AL, AH; COMPARO CARACTERES  
JNZ INCOR

MOV BX, DX

INC BX

DEC CL

JNZ COMPARA

MOV BX, OFFSET MSJ\_PERM

MOV AL, OFFSET MSJ\_DENE - OFFSET MSJ\_PERM

INT 7

JMP FIN

INCOR: MOV BX, OFFSET MSJ\_DENE

MOV AL, OFFSET MSJ\_BLO - OFFSET MSJ\_DENE

INT 7

DEC CH

JNZ RESET

MOV BX, OFFSET MSJ\_BLO

MOV AL, OFFSET FIN\_MSJ - OFFSET MSJ\_BLO

INT 7

FIN: INT 0

END

**PARTE 3 - Pila, subrutinas y dirección de retorno**.

**A) Uso de la pila ⭐**

Si el registro SP vale 8000h al comenzar el programa, indicar el valor del registro SP luego de ejecutar cada una de las instrucciones de la tabla, en el orden en que aparecen. Indicar, de la misma forma, los valores de los registros AX y BX.

|  |  |  |  |
| --- | --- | --- | --- |
| **Instrucción** | **Valor del registro SP** | **AX** | **BX** |
| mov ax,5 | 8000h | 5 | ? |
| mov bx,3 | 8000h | 5 | 3 |
| push ax | 8000h-2 = 7FFEh | 5 | 3 |
| push ax | 7FFEh-2 = 7FFCh | 5 | 3 |
| push bx | 7FFCh-2 = 7FFAh | 5 | 3 |
| pop bx | 7FFAh+2 = 7FFCh | 5 | 3 |
| pop bx | 7FFCh+2 = 7FFEh | 5 | 5 |
| pop ax | 7FFEh+2 = 8000h | 5 | 5 |

**B) Llamadas a subrutinas y la pila⭐**

Si el registro SP vale 8000h al comenzar el programa, indicar el valor del registro SP luego de ejecutar cada instrucción. Considerar que el programa comienza a ejecutarse con el IP en la dirección 2000h, es decir que la primera instrucción que se ejecuta es la de la línea 5 (push ax).   
Nota: Las sentencias ORG y END no son instrucciones sino indicaciones al compilador, por lo tanto no se ejecutan.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Instrucción** | **Valor del registro SP** |  |
| 1 | **org 3000h** | --------- |  |
| 2 | rutina: mov bx, 3 | 7EECh |  |
| 3 | ret | 7FFCh + 2 = 7FFEh |  |
| 4 | **org 2000h** | --------- |  |
| 5 | push ax | 8000h - 2 = 7FFEh |  |
| 6 | call rutina | 7FFEh - 2 = 7FFCh |  |
| 7 | pop bx | 7FFEh + 2 = 8000h |  |
| 8 | hlt | 8000h |  |
| 9 | **end** | --------- |  |

C) Llamadas a subrutinas y dirección de retorno⭐

Si el registro SP vale 8000h al comenzar el programa, indicar el valor de SP y el contenido de la pila luego de ejecutar cada instrucción. Si el contenido es desconocido/basura, indicarlo con el símbolo ?. Considerar que el programa comienza a ejecutarse con el IP en la dirección 2000h, es decir que la primera instrucción que se ejecuta es la de la línea 5 (call rut). Se provee la ubicación de las instrucciones en memoria, para poder determinar la dirección de retorno de la rutina.

Además, explicar detalladamente:

a) Las acciones que tienen lugar al ejecutarse la instrucción call rut.

b) Las acciones que tienen lugar al ejecutarse la instrucción ret.

ORG 3000h

rut: mov bx, 3 Dirección 3000h  
ret Dirección 3002h

ORG 2000h

call rut Dirección 2000h   
add cx, 5 Dirección 2002h  
call rut Dirección 2004h  
hlt Dirección 2006h  
end

**call rut:** se hace un llamado a subrutina, SP pasa de 8000h a 7FFEh. En la pila se guarda el valor 2002h. En el 7FFFh queda 20, en el 7FFEh queda 02

**mov bx, 3:** no se modifica ni el SP ni el valor de la pila.

**Ret**: se retorna de la subrutina, SP vuelve al valor 8000h. Se saca el valor de 2002h y la pila queda con un valor desconocido.

**Add cx, 5:** No se modifica ni el SP ni la pila.

**Call rut:** se hace un llamado a subrutina, SP pasa de 8000h a 7FFEh. En la pila se guarda el valor 2006h. En el 7FFFh queda el 20 y en el 7FFEh el 06

**mov bx, 3:** no se modifica ni el SP ni el valor de la pila.

**Ret**: se retorna de la subrutina, SP vuelve al valor 8000h. Se saca el valor de 2006h y la pila queda con un valor desconocido.

**Hlt**: no se modifica el SP ni la pila.

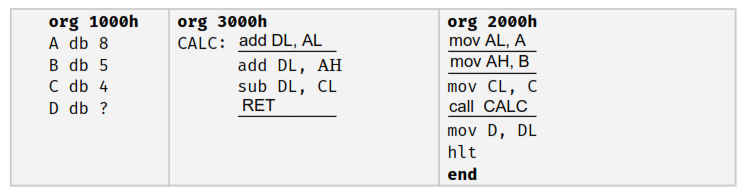
**PARTE 4 – Pasaje de parámetros**

1) Tipos de Pasajes de Parámetros⭐ Indicar con un tilde, para los siguientes ejemplos, si el pasaje del parámetro es por registro o pila, y por valor o referencia:

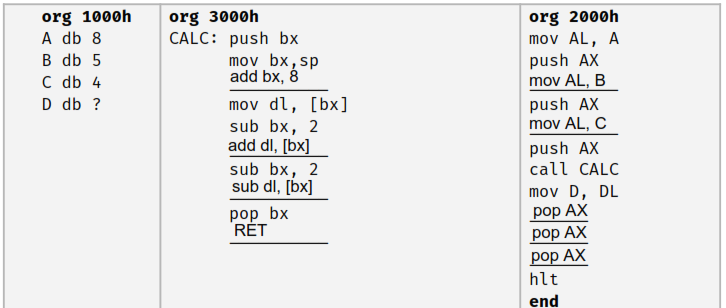
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Código | A través de: | | Por: | |
| Registro | Pila | Valor | Referencia |
| mov ax,5 call subrutina | x |  | x |  |
| mov dx,offset A call subrutina | x |  |  | x |
| mov bx, 5 push bx call subrutina pop bx |  | x | x |  |
| mov cx, offset A push cx call subrutina pop cx |  | x |  | x |
| mov dl, 5 call subrutina | x |  | x |  |
| Call subrutina mov A, dx | x |  | x |  |

2) Pasaje de parámetros a través de registros y la pila⭐

A) Completar las instrucciones del siguiente programa, que envía a una subrutina 3 valores A, B y C a través de registros AL, AH y CL, calcula AL+AH-CL, y devuelve el resultado en DL.



B) Idem el inciso anterior, pero los valores A, B y C se reciben mediante pasaje de parámetros por valor a través de la pila. El resultado se devuelve de igual forma por el registro dl y por valor.



C) Modificar el programa anterior para enviar los parámetros A, B y C a través de la pila pero ahora por referencia.

ORG 1000H

A DB 8

B DB 5

C DB 4

D DB ?

org 3000h

calc: push bx

mov dx,0

mov bx, sp

add bx, 8

push bx

mov bx, [bx]

mov dl, [bx]

pop bx

sub bx, 2

push bx

mov bx,[bx]

add dl, [bx]

pop bx

sub bx,2

mov bx, [bx]

sub dl, [bx]

pop bx

ret

org 2000h

MOV AX, OFFSET A

PUSH AX

MOV AX, OFFSET B

PUSH AX

MOV AX, OFFSET C

PUSH AX

call calc

mov d, dl

hlt

end

**3) Primeras subrutinas**

Re implementar los programas del Ejercicio 1 - Parte 1, pero ahora implementando las siguientes subrutinas. En todos los casos, recibir los valores por parámetros pasados por registro, y devolver el resultado también por valor y por registro.

1. 📄 CONTAR\_CAR ⭐ Recibe la dirección de comienzo de un string en BX, su longitud en AL, y el carácter a contar en AH. Retorna en CL la cantidad de veces que aparece el carácter.

ORG 1000h

LETRA DB "a"  
MENSAJE DB "Hola, Buenas Tardes"  
CANT DB ?

ORG 3000h

CONTAR\_CAR: MOV CL,0  
 LAZO: CMP AH, [BX]  
 JNZ SALTO  
 INC CL  
 SALTO: INC BX  
 DEC AL  
 JNZ LAZO

ORG 2000h

MOV BX, OFFSET MENSAJE  
MOV AL, OFFSET CANT - OFFSET MENSAJE  
MOV AH, LETRA  
CALL CONTAR\_CAR  
MOV CANT, CL  
HLT  
END

2. 📄 ES\_MAYUS ⭐ Recibe un carácter en el registro AL y retorna en AH el valor 0FFh si es mayúscula y 0 de lo contrario.

ORG 1000H

LETRA DB ‘A’  
RES DB ?

ORG 3000H

ES\_MAYUS: CMP AL, ‘A’  
 JS NO\_ES\_MAYUS  
 CMP AL, ‘Z’ +1  
 JNS NO\_ES\_MAYUS  
 MOV AH, 0FFH  
 JMP FIN  
NO\_ES\_MAYUS: MOV AH, 0  
FIN: RET

ORG 2000H

MOV AL, LETRA  
 CALL ES\_MAYUS  
 MOV RES, AH  
 HLT  
 END

3. 📄 A\_MINUS⭐ Recibe un carácter mayúscula en AL y lo devuelve como minúscula.  
 ORG 1000H  
CARAC DB ‘A’  
  
 ORG3000H

A\_MINIS: ADD AL, 20H  
  
 ORG 2000H  
 MOV AL, CARAC  
 CALL A\_MINUS  
 MOV CARAC, AL  
 HLT  
 END

4. 📄 STRING\_A\_MINUS ⭐⭐ Recibe la dirección de comienzo de un string en BX, su longitud en AL. Recorre el string, cambiando a minúscula las letras que sean mayúsculas. No retorna nada, sino que modifica el string directamente en la memoria.

Org 1000 h

MENSAJE DB ‘Hola, Buenas Tardes’  
FIN\_MSJ DB ?

ORG 3000H

STRING\_A\_MINUS: LAZO: CMP BYTE PTR, ‘A’  
JS NO\_ES MAY  
CMP BYTE PTR [BX], ‘Z’+1  
JNS NO\_ES\_MAYUS  
ADD BYTE PTR [BX], 20H  
INC BX  
DEC AL  
JNZ LAZO  
JMP FIN  
NO\_ES\_MAYUS: INC BX  
DEC AL  
JNZ LAZO  
FIN: RET

ORG 2000H

MOV BX, OFFSET MENSAJE  
 MOV AL, OFFSET FIN\_MSJ – OFFSET MSJ  
 CALL STRING\_A\_MINUS  
 HLT  
 END

**4) Multiplicación de números sin signo con parámetros**

El pasaje de parámetros más usual suele ser por valor y por registro. No obstante, en algunas ocasiones también se utilizan pasajes de parámetros más avanzados que permiten más flexibilidad o eficiencia.

Escribir un programa que tenga dos valores de 8 bits A y B almacenados en su memoria y realice la multiplicación de A y B. El resultado se debe guardar en la variable RES de 16 bits, o sea que RES = A ⨯ B.

Para hacerlo, implementar una subrutina MUL:

A. ⭐ Pasando los parámetros por valor desde el programa principal a través de los registros AL y AH, y devolviendo el resultado a través del registro AX por valor.

ORG 1000H

A DB 7  
 B DB 6  
 RES DW ?

ORG 3000H

MULT: MOV CX, 0  
 LOOP: ADD CL, AH  
 DEC AH  
 JNZ LOPP  
 MOV AX, CX  
 RET

ORG 2000h

MOV AL, A  
 MOV AH, B  
 CALL MULT  
 MOV RES, CX  
 HLT  
 END

B. ⭐⭐ Pasando los parámetros por referencia desde el programa principal a través de registros, y devolviendo el resultado a través de un registro por valor.

ORG 1000h

A DW 5

B DW 4

RES DW ?

ORG 3000h

MULT: MOV CX, 0

MOV DX,0

MOV DX, [BX]

MOV BX, AX

MOV AX, [BX]

LOOP:ADD CX, DX

DEC AX

JNZ LOOP

FIN: RET

ORG 2000H

MOV AX, OFFSET A

MOV BX, OFFSET B

CALL MULT

MOV RES, CX

HLT

END

**Parte 5: Ejercicios integradores o tipo parcial**

**1) Ahorcado secuencial⭐⭐**

Escribir un programa que permita a una persona desafiar a otra jugando al ahorcado secuencial. En el ahorcado secuencial, a diferencia del tradicional, hay que adivinar las letras en orden. Por ejemplo, si la palabra a adivinar es “alma”, la persona que adivina debe ingresar primero la “a”, luego la “l”, luego la “m” y finalmente debe ingresar nuevamente la “a”.

El programa tiene dos fases: primero, una persona carga la palabra a adivinar, y luego la otra persona adivina la palabra.  
Fase 1: Se debe mostrar el mensaje “Ingresá la palabra a adivinar: ”. Luego, se debe leer un stringhasta que llegue el carácter “.”, y al terminar de leer, se debe mostrar el mensaje “Comenzá a adivinar!”.  
Fase 2: se deben leer carácteres hasta que la persona termine de adivinar todo el string, o se le acaben los intentos.

Si la persona ingresa un carácter que coincide con el que tenía que adivinar, se muestra ese carácter en pantalla, y se avanza al carácter siguiente del string a adivinar. De lo contrario, no se muestra nada, y la persona debe seguir intentando. Sí adivinó todo el string, debe mostrarse el mensaje “Ganaste!”. La persona tiene 50 intentos de letras para adivinar el string. Si se acaba la cantidad de intentos y no adivinó todo el string, debe mostrarse el mensaje “Perdiste, el string era S”, donde S es el string aadivinar completo.

ORG 1000H

mensIni db "Ingresa la palabra a adivinar: ",0ah

mensPost db "Comenza a adivinar! ",0ah

win db "Has ganado! ",0ah

lose db "Has perdido! ",0ah

maxIntentos db 5

car db ?

palabra db ?

org 3000h

leerPalabra:

mov bx, offset palabra

loop:

int 6

cmp byte ptr [bx], '.'

jz salir

inc bx

jmp loop

salir:

ret

leerLetra:

mov cx, bx ;guardo el contenido de bx

mov bx, offset car

int 6

mov bx, cx

mov cl, byte ptr [bx] ; uso cl para traer a registro letra a adivinar

cmp cl, car

jnz noAcerto

mov al, 1

int 7

inc bx

noAcerto: ret

success:

mov al,1

mov bx, offset car

mov car, 0ah

int 7

mov bx, offset win

mov al, offset lose - offset win

int 7

jmp fin

ret

notSuccess:

mov al,1

mov bx, offset car

mov car, 0ah

int 7

mov bx, offset lose

mov al, offset maxIntentos - offset lose

int 7

jmp fin

ret

ORG 2000H ;ah contador de cant intentos

mov bx, offset mensIni

mov al, offset mensPost - offset mensIni

int 7

call leerPalabra

mov bx, offset mensPost

mov al, offset win - offset mensPost

int 7

mov bx, offset palabra

mov ah,0

bucle:

cmp ah, maxIntentos

jz exitWrong

cmp byte ptr [bx],'.'

jz printSuccess

call leerLetra

inc ah

jmp bucle

exitWrong: call notSuccess

printSuccess: call success

fin:int 0

END