# Práctica 1: Subrutinas, Parámetros e Interrupciones por Software

## Parte 1: Repaso de VonSim

### Ejercicio 2

```
                ORG 1000H
C               DB 'Z'          ; 'Z' ≡ "Z" (son equivalentes)
RES             DB ?

                ORG 2000H
                MOV AL, 0       ; inicialmente se hipotetiza que el
                                ; carácter no es una mayúscula
                MOV CL, C
                CMP CL, 41H     ; 41H = 'A' (código ASCII)
                JS ALMACENAR    ; ¿CL – 41H < 0? → ¿Carácter < 'A'?
                CMP CL, 5BH     ; 5AH = 'Z' (código ASCII)
                JNS ALMACENAR   ; ¿CL – 5BH ≥ 0? → ¿Carácter > 'Z'?
                MOV AL, 0FFH    ; el carácter es una mayúscula
ALMACENAR:      MOV RES, AL
                HLT
                END
```

### Ejercicio 3

```
      ORG 1000H
C     DB 'Z'          ; este carácter debe ser una mayúscula

      ORG 2000H
      MOV AL, C
      ADD AL, 20H     ; 'z' - 'Z' = 7AH – 5AH = 20H
      MOV C, AL       ; 'z' = 'Z' + 20H (aplicable a cualquier
                      ; conversión)
      HLT
      END
```

### Ejercicio 4

```
            ORG 1000H
MENSAJE     DB "Hola. BuenAZ tardes."
FIN_MSJ     DB ?

            ORG 2000H
            MOV AL, OFFSET FIN_MSJ - OFFSET MENSAJE
            MOV BX, OFFSET MENSAJE
BUCLE:      CMP BYTE PTR [BX], 'A'      ; 'A' ≠ "A" (son distintas)
            JS SIG_CAR                  ; saltar si no es mayúscula
            CMP BYTE PTR [BX], '['      ; '[' = 5BH (consecutivo de 'Z')
            JNS SIG_CAR                 ; saltar si no es mayúscula
            ADD BYTE PTR [BX], 20H      ; convertir mayúscula a
                                        ; minúscula
SIG_CAR:    INC BX                      ; avanzar al siguiente carácter
            DEC AL
            JNZ BUCLE
            HLT
```

```
        END
```

## Parte 2: Entrada/Salida 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
```

a)
El programa imprime en la pantalla del simulador el mensaje "ARQUITECTURA DE COMPUTADORAS-FACULTAD DE INFORMATICA-UNLP".

b)
El programa imprime "UNLP" al final del mensaje porque los números 55H, 4EH, 4CH y 50H, definidos en dicho programa a continuación de la cadena "FACULTAD DE INFORMATICA-", representan los códigos ASCII de los caracteres "U", "N", "L" y "P" respectivamente.

c)
Al ejecutarse la interrupción por software “INT 7”, en BX debe encontrarse previamente almacenada la dirección inicial del mensaje a imprimir en pantalla (en este caso, la dirección 1000H) y, en AL, la longitud del mismo (en este escenario, 57 caracteres).

### Ejercicio 2

```
      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
```

a)
Al ejecutarse la interrupción por software “INT 6”, en BX debe encontrarse previamente la dirección de memoria en donde se almacenará el carácter leído (en este caso, la dirección 1500H). Luego de ejecutar “INT 6”, el carácter leído se encontrará en esa dirección de memoria.

b)
La segunda instrucción “INT 7” imprime en la pantalla del simulador el número previamente ingresado por el usuario durante la ejecución de la interrupción por software “INT 6”. Por ejemplo, si el usuario escribió el número 6, se mostrará en la pantalla el carácter "6".

c)
En la dirección de memoria etiquetada “NUM” (1500H), no se almacena en forma directa el número ingresado por el usuario durante la ejecución de la interrupción por software “INT 6”, sino su respectivo código ASCII o, en otras palabras, la expresión correspondiente a su carácter equivalente. Por ejemplo, si el usuario escribió el número 6, se guardará el carácter “6”, representado por el código ASCII 36H. Este último valor se escribirá efectivamente en la dirección “NUM” y, en consecuencia, será cargado eventualmente en el registro CL.

**Ejercicio 3**
Para la versión originalmente publicada de esta práctica en el sitio web oficial de la cátedra, debe incorporarse en todos los programas propuestos en la consigna del presente ejercicio la interrupción por software “INT 0” antes de la sentencia “END” a fin de detener la ejecución del procesador.

a)

```
      ORG 1000H
A     DB "HO LA"       ; debe eliminarse el espacio para que el
                       ; mensaje tenga cuatro caracteres y se
                       ; imprima completo (sino se muestra "HO L")
B     DB ?

      ORG 2000H
      mov bx, offset A
      mov al, 4        ; puede remplazarse el 4 por “offset B - offset
                       ; A”
      int 7
      int 0
      END
```

b)

```
        ORG 1000H
A       DB "HOLA"
B       DB ?


        ORG 2000H
        mov al, offset A - offset B
        ; offset B - offset A (para obtener la longitud del mensaje,
        ; intercambiar el minuendo minuendo y el sustraendo)
        mov bx, offset A
        int 7
        int 0
        END
```

c)

```
        ORG 1000H
A       DB ?

        ORG 2000H
        int 6
        mov bx, offset A      ; esta instrucción debe situarse y
                              ; ejecutarse antes de la interrupción por
                              ; software “int 6”
        int 0
        END
```

e)

```
        ORG 1000H
A       DB ?

        ORG 2000H
        mov al, 3         ; esta instrucción es innecesaria
        mov bx, A         ; debe almacenarse la dirección de A (“offset
                          ; A”) en lugar del valor almacenado en ella
        int 6
        int 0
        END
```

**Ejercicio 4**

b)

El presente programa imprime en la pantalla del simulador todos los caracteres disponibles en la tabla de códigos ASCII extendido (255 en total).

```
            ORG 1000H
CAR         DB 0              ; 00H = Carácter NUL

            ORG 2000H
            MOV BX, OFFSET CAR
            MOV AL, 1
BUCLE:      INT 7
            INC CAR
```

```
            JNZ BUCLE       ; si CAR es igual a cero, ya se imprimieron
                            ; todos los caracteres disponibles
            INT 0
            END
```

c)

```
            ORG 1000H
CAR         DB "0", 0AH     ; 0AH = Carácter LF (Line Feed o Salto de
                            ; Línea)

            ORG 2000H
            MOV BX, OFFSET CAR
            MOV AL, 2
BUCLE:      INT 7
            INC CAR
            CMP CAR, 3AH    ; 39H = Carácter "9"
            JS BUCLE        ; si CAR es mayor a 39H, ya se imprimieron
                            ; todos los dígitos disponibles
            INT 0
            END
```

**Ejercicio 5**

El presente programa solicita el ingreso de una contraseña de 4 caracteres por teclado, sin visualizarla en pantalla. Si coincide con una clave predefinida (y guardada en memoria), imprimirá el mensaje "Acceso permitido". En caso contrario, mostrará el mensaje "Acceso denegado", terminando la ejecución del programa en ambos escenarios a diferencia de la tarea planteada en la consigna, proponiéndose esta última como desafío a resolver por el alumno.

```
            ORG 1000H
MSJ_ING     DB "INGRESE LA CLAVE:", 0AH     ; 0AH = carácter LF (salto
                                            ; línea)
MSJ_CLAVE   DB "****", 0AH
MSJ_PERM    DB "Acceso permitido"
MSJ_DENEG   DB "Acceso denegado"
FIN_DENEG   DB ?

            ORG 1500H
CLAVE       DB "!5X#"       ; Clave correcta y esperada
CLAVE_ING   DB ?            ; Clave efectivamente ingresada por el
                            ; usuario

            ORG 2000H
            MOV BX, OFFSET MSJ_ING
            MOV AL, OFFSET MSJ_CLAVE - OFFSET MSJ_ING
            INT 7
            MOV AL, 1
            MOV AH, OFFSET CLAVE_ING - OFFSET CLAVE
            MOV CX, 0
LAZO_ING:   MOV BX, OFFSET CLAVE_ING
            ADD BX, CX
            INT 6
            MOV BX, OFFSET MSJ_CLAVE
            ADD BX, CX
```

```
            INT 7
            INC CL
            CMP CL, AH
            JNZ LAZO_ING
            INC BX          ; BX y AL son modificados durante la
                            ; ejecución de la interrupción "INT 7",
                            ; pero una vez finalizada recuperan sus
                            ; valores originales
            INT 7
            MOV CL, 0
LAZO_VAL:   MOV BX, OFFSET CLAVE
            ADD BX, CX
            MOV DL, [BX]
            MOV BX, OFFSET CLAVE_ING
            ADD BX, CX
            CMP [BX], DL
            JNZ FALLO
            INC CL
            CMP CL, AH
            JNZ LAZO_VAL
EXITO:      MOV BX, OFFSET MSJ_PERM
            MOV AL, OFFSET MSJ_DENEG - OFFSET MSJ_PERM
            JMP FIN_PROG
FALLO:      MOV BX, OFFSET MSJ_DENEG
            MOV AL, OFFSET FIN_DENEG - OFFSET MSJ_DENEG
FIN_PROG:   INT 7
            INT 0
            END
```

## Parte 3: Pila, subrutinas y dirección de retorno

### Ejercicio 1

a)

| | | |
|---|---|---|
| **Valores iniciales** | **Registro SP** | 8000h |
| | **Registro AX** | ? |
| | **Registro BX** | ? |

| Orden | Instrucción | Registro SP | Registro AX | Registro BX |
|---|---|---|---|---|
| **1** | `mov ax, 5` | 8000h | 5 | ? |
| **2** | `mov bx, 3` | 8000h | 5 | 3 |
| **3** | `push ax` | 7FFEh | 5 | 3 |
| **4** | `push ax` | 7FFCh | 5 | 3 |
| **5** | `push bx` | 7FFAh | 5 | 3 |
| **6** | `pop bx` | 7FFCh | 5 | 3 |
| **7** | `pop bx` | 7FFEh | 5 | 5 |
| **8** | `pop ax` | 8000h | 5 | 5 |

| Dirección en memoria de pila | Valor/Contenido |
|---|---|
| **7FF8h** | ? |
| **7FF9h** | ? |
| **7FFAh** | 3 |
| **7FFBh** | 0 |
| **7FFCh** | 5 |
| **7FFDh** | 0 |
| **7FFEh** | 5 |
| **7FFFh** | 0 |
| **8000h** | ? |

b)

| | | |
|---|---|---|
| **Valores iniciales** | **Registro SP** | 8000h |
| | **Registro IP** | 2000h |

| Número | Instrucción | Registro SP |
|---|---|---|
| **1** | `org 3000h` | |
| **2** | `rutina: mov bx, 3` | 7FFCh |
| **3** | `ret` | 7FFEh |
| **4** | `org 2000h` | |
| **5** | `push ax` | 7FFEh |
| **6** | `call rutina` | 7FFCh |
| **7** | `pop bx` | 8000h |
| **8** | `hlt` | 8000h |
| **9** | `end` | |

c)

```
      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
```

| | | |
|---|---|---|
| **Valores iniciales** | **Registro SP** | 8000h |
| | **Registro IP** | 2000h |

| Orden | Instrucción | Registro SP |
|---|---|---|
| **1** | `call rut` | 7FFEh |
| **2** | `mov bx, 3` | 7FFEh |
| **3** | `ret` | 8000h |
| **4** | `add cx, 5` | 8000h |
| **5** | `call rut` | 7FFEh |
| **6** | `mov bx, 3` | 7FFEh |
| **7** | `ret` | 8000h |
| **8** | `hlt` | 8000h |

| Orden | Instrucción | Dirección en memoria de pila | Valor/Contenido |
|---|---|---|---|
| **1** | `call rut` | **7FFCh** | ? |
| | | **7FFDh** | ? |
| | | **7FFEh** | 02h |
| | | **7FFFh** | 20h |
| | | **8000h** | ? |

| Orden | Instrucción | Dirección en memoria de pila | Valor/Contenido |
|---|---|---|---|
| **2** | `mov bx, 3` | **7FFCh** | ? |
| | | **7FFDh** | ? |
| | | **7FFEh** | 02h |
| | | **7FFFh** | 20h |
| | | **8000h** | ? |

| Orden | Instrucción | Dirección en memoria de pila | Valor/Contenido |
|---|---|---|---|
| **3** | `ret` | **7FFCh** | ? |
| | | **7FFDh** | ? |
| | | **7FFEh** | ? |
| | | **7FFFh** | ? |
| | | **8000h** | ? |

| Orden | Instrucción | Dirección en memoria de pila | Valor/Contenido |
|---|---|---|---|
| **4** | `add cx, 5` | **7FFCh** | ? |
| | | **7FFDh** | ? |
| | | **7FFEh** | ? |
| | | **7FFFh** | ? |
| | | **8000h** | ? |

| Orden | Instrucción | Dirección en memoria de pila | Valor/Contenido |
|---|---|---|---|
| 5 | `call rut` | **7FFCh** | ? |
| | | **7FFDh** | ? |
| | | **7FFEh** | 06h |
| | | **7FFFh** | 20h |
| | | **8000h** | ? |

| Orden | Instrucción | Dirección en memoria de pila | Valor/Contenido |
|---|---|---|---|
| 6 | `mov bx, 3` | **7FFCh** | ? |
| | | **7FFDh** | ? |
| | | **7FFEh** | 06h |
| | | **7FFFh** | 20h |
| | | **8000h** | ? |

| Orden | Instrucción | Dirección en memoria de pila | Valor/Contenido |
|---|---|---|---|
| 7 | `ret` | **7FFCh** | ? |
| | | **7FFDh** | ? |
| | | **7FFEh** | ? |
| | | **7FFFh** | ? |
| | | **8000h** | ? |

| Orden | Instrucción | Dirección en memoria de pila | Valor/Contenido |
|---|---|---|---|
| 8 | `hlt` | **7FFCh** | ? |
| | | **7FFDh** | ? |
| | | **7FFEh** | ? |
| | | **7FFFh** | ? |
| | | **8000h** | ? |

# Parte 4: Pasaje de parámetros

## Ejercicio 1

| Inciso | Código | A través de | | Por | |
|---|---|---|---|---|---|
| | | Registro | Pila | Valor | Referencia |
| **a)** | `mov ax, 5`<br>`call subrutina` | ✓ | | ✓ | |
| **b)** | `mov dx, offset A`<br>`call subrutina` | ✓ | | | ✓ |
| **c)** | `mov bx, 5`<br>`push bx`<br>`call subrutina`<br>`pop bx` | | ✓ | ✓ | |
| **d)** | `mov cx, offset A`<br>`push cx`<br>`call subrutina`<br>`pop cx` | | ✓ | | ✓ |
| **e)** | `mov dl, 5`<br>`call subrutina` | ✓ | | ✓ | |
| **f)** | `call subrutina`<br>`mov A, dx` | ✓ | | ✓ | |

## Ejercicio 2

a)

```
            ORG 1000H
A           DB 8
B           DB 5
C           DB 4
D           DB ?

            ORG 3000H
CALC:       MOV DL, AL      ; 1° instrucción faltante
            ADD DL, AH
            SUB DL, CL
            RET             ; 2° instrucción faltante

            ORG 2000H
            MOV AL, A       ; 3° instrucción faltante
            MOV AH, B       ; 4° instrucción faltante
            MOV CL, C
            CALL CALC       ; 5° instrucción faltante
            MOV D, DL
            HLT
            END
```

b)

```
            ORG 1000H
A           DB 8
B           DB 5
C           DB 4
D           DB ?

            ORG 3000H
CALC:       PUSH BX          ; SP = 7FF6H
            MOV BX, SP
            ADD BX, 8        ; 1° instrucción faltante
                             ; BX = 7FF6H + 8H = 7FFEH → [BX] = A
            MOV DL, [BX]
            SUB BX, 2        ; BX = 7FFCH → [BX] = B
            ADD DL, [BX]     ; 2° instrucción faltante
            SUB BX, 2        ; BX = 7FFAH → [BX] = C
            SUB DL, [BX]     ; 3° instrucción faltante
            POP BX           ; SP = 7FF8H
            RET              ; 4° instrucción faltante (SP = 7FFAH)

            ORG 2000H        ; SP = 8000H
            MOV AL, A
            PUSH AX          ; SP = 7FFEH
            MOV AL, B        ; 5° instrucción faltante
            PUSH AX          ; SP = 7FFCH
            MOV AL, C        ; 6° instrucción faltante
            PUSH AX          ; SP = 7FFAH
            CALL CALC        ; SP = 7FF8H
            MOV D, DL
            POP AX           ; 7° instrucción faltante (SP = 7FFCH)
            POP AX           ; 8° instrucción faltante (SP = 7FFEH)
            POP AX           ; 9° instrucción faltante (SP = 8000H)
            HLT
            END
```

c)

```
            ORG 1000H
A           DB 8
B           DB 5
C           DB 4
D           DB ?

            ORG 3000H
CALC:       PUSH BX          ; SP = 7FF6H
            MOV BX, SP
            ADD BX, 8        ; BX = 7FFEH → [BX] = OFFSET A
            PUSH BX
            MOV BX, [BX]
            MOV DL, [BX]
            POP BX
            SUB BX, 2        ; BX = 7FFCH → [BX] = OFFSET B
            PUSH BX
            MOV BX, [BX]
            ADD DL, [BX]
```

```
        POP BX
        SUB BX, 2       ; BX = 7FFAH → [BX] = OFFSET C
        PUSH BX
        MOV BX, [BX]
        SUB DL, [BX]
        POP BX
        POP BX          ; SP = 7FF8H
        RET             ; SP = 7FFAH

        ORG 2000H       ; SP = 8000H
        MOV AX, OFFSET A
        PUSH AX         ; SP = 7FFEH
        MOV AX, OFFSET B
        PUSH AX         ; SP = 7FFCH
        MOV AX, OFFSET C
        PUSH AX         ; SP = 7FFAH
        CALL CALC       ; SP = 7FF8H
        MOV D, DL
        POP AX          ; SP = 7FFCH
        POP AX          ; SP = 7FFEH
        POP AX          ; SP = 8000H
        HLT
        END
```

## Ejercicio 3

b)

```
            ORG 1000H
C           DB 'Z'          ; 'Z' ≡ "Z" (son equivalentes)
RES         DB ?

            ORG 3000H
ES_MAYUS:   MOV AH, 0       ; inicialmente se hipotetiza que el
                            ; carácter no es una mayúscula
            CMP AL, 41H     ; 41H = 'A' (código ASCII)
            JS FIN_MAYUS    ; ¿AL – 41H < 0? → ¿Carácter < 'A'?
            CMP AL, 5BH     ; 5AH = 'Z' (código ASCII)
            JNS FIN_MAYUS   ; ¿AL – 5BH ≥ 0? → ¿Carácter > 'Z'?
            MOV AH, 0FFH    ; el carácter es una mayúscula
FIN_MAYUS:  RET

            ORG 2000H
            MOV AL, C
            CALL ES_MAYUS
            MOV RES, AH
            INT 0
            END
```

c)

```
            ORG 1000H
C           DB 'Z'          ; este carácter debe ser una mayúscula

            ORG 3000H
A_MINUS:    ADD AL, 20H     ; 'z' - 'Z' = 7AH – 5AH = 20H
                            ; 'z' = 'Z' + 20H (aplicable a cualquier
                            ; conversión)
            RET

            ORG 2000H
            MOV AL, C
            CALL A_MINUS
            MOV C, AL
            INT 0
            END
```

d)

```
                ORG 1000H
MENSAJE         DB "Hola. BuenAZ tardes."
FIN_MSJ         DB ?

                ORG 3000H
CAD_A_MINUS:    PUSH BX                     ; preservar BX original
                PUSH AX                     ; preservar AX original
BUCLE:          CMP BYTE PTR [BX], 'A'      ; 'A' ≠ "A" (son distintas)
                JS SIG_CAR                  ; saltar si no es mayúscula
                CMP BYTE PTR [BX], '['      ; '[' = 5BH (consecutivo
                                            ; de 'Z')
                JNS SIG_CAR                 ; saltar si no es mayúscula
                ADD BYTE PTR [BX], 20H      ; convertir a minúscula
SIG_CAR:        INC BX                      ; siguiente carácter
                DEC AL
                JNZ BUCLE
                POP AX                      ; recuperar AX original
                POP BX                      ; recuperar BX original
                RET

                ORG 2000H
                MOV AL, OFFSET FIN_MSJ - OFFSET MENSAJE
                MOV BX, OFFSET MENSAJE
                CALL CAD_A_MINUS
                INT 0
                END
```

## Ejercicio 4

b)

```
            ORG 1000H
A           DB 5H           ; A y B deben ser números
B           DB 3H           ; mayores que cero
RES         DW ?
```

```
            ORG 3000H
MUL:        PUSH AX          ; preservar registros
            PUSH CX
            MOV BX, AX       ; BX = dirección de A
            MOV AL, [BX]     ; AL = valor de A
            AND AX, 0FFH     ; AH = 0 (forzar a cero los 8 bits de AH)
            MOV BX, CX       ; BX = dirección de B
            MOV CL, [BX]     ; CL = valor de B
            MOV DX, 0
SUMA:       ADD DX, AX
            DEC CL
            JNZ SUMA
            POP CX           ; restaurar registros
            POP AX
            RET

            ORG 2000H
            MOV AX, OFFSET A      ; cargar parámetros
            MOV CX, OFFSET B
            CALL MUL
            MOV RES, DX
            INT 0
            END
```

c)

```
            ORG 1000H
A           DB 5H            ; A y B deben ser números
B           DB 3H            ; mayores que cero
RES         DW ?

            ORG 3000H
MUL:        PUSH AX          ; preservar registro
            MOV CL, AH       ; CL = valor de B
            AND AX, 0FFH     ; AH = 0 (forzar a cero los 8 bits de AH)
            MOV DX, 0
SUMA:       ADD DX, AX
            DEC CL
            JNZ SUMA
            MOV BX, SP
            SUB BX, 2
            MOV [BX], DX     ; el resultado se almacena encima del tope
                             ; de la pila (no se apila)
            MOV DX, BX       ; cargar en DX la dirección del resultado
            POP AX           ; restaurar registro
            RET

            ORG 2000H
            MOV AL, A        ; cargar parámetros
            MOV AH, B
            CALL MUL
            MOV BX, DX
            MOV DX, [BX]
            MOV RES, DX
            INT 0
            END
```

e)

```
            ORG 1000H
A           DB 5H           ; A y B deben ser números
B           DB 3H           ; mayores que cero
RES         DW ?

            ORG 3000H
MUL:        PUSH AX         ; preservar registros
            PUSH BX
            MOV BX, SP
            ADD BX, 8       ; corrección por los registros BX y AX (4),
                            ; el registro IP (2) y la dirección de la
                            ; variable B (2)
            PUSH BX
            MOV BX, [BX]    ; BX = dirección de A
            MOV AL, [BX]    ; AL = valor de A
            AND AX, 0FFH    ; AH = 0 (forzar a cero los 8 bits de AH)
            POP BX
            SUB BX, 2       ; BX apuntará a la dirección de B apilada
            MOV BX, [BX]    ; BX = dirección de B
            MOV CL, [BX]    ; CL = valor de B
            MOV DX, 0
SUMA:       ADD DX, AX
            DEC CL
            JNZ SUMA
            POP BX          ; restaurar registros
            POP AX
            RET

            ORG 2000H
            MOV AX, OFFSET A      ; cargar parámetros
            MOV BX, OFFSET B
            PUSH AX
            PUSH BX
            CALL MUL
            MOV RES, DX
            POP BX                ; desapilar parámetros
            POP AX
            INT 0
            END
```

# Parte 5: Ejercicios integradores o tipo parcial

## Ejercicio 1

```
            ORG 1000H
MSJ_ING_1   DB "Ingresá la palabra a adivinar:", 0AH
MSJ_ING_2   DB "¡Comenzá a adivinar!", 0AH
LONG_ADIV   DB ?    ; Longitud de la palabra a adivinar
CAR_ADIV    DB ?    ; Último carácter ingresado para adivinar la palabra
INTENTOS    DB 50
MSJ_GAN     DB 0AH, "¡Ganaste!"
MSJ_PERD    DB 0AH, "Perdiste, la palabra a adivinar era:", 0AH
PAL_ADIV    DB ?    ; Palabra a adivinar
```

```
            ORG 3000H
FASE_1:     PUSH BX
            PUSH AX
            PUSH CX
            INT 7
            MOV DL, 0
            PUSH BX
            MOV BX, CX
BUCLE_1:    INT 6
            CMP BYTE PTR [BX], 2EH     ; 2EH = '.'
            JZ FIN_1
            INC BX
            INC DL
            JMP BUCLE_1
FIN_1:      POP BX
            MOV DH, AH
            AND AX, 0FFH
            ADD BX, AX
            MOV AL, DH
            INT 7
            POP CX
            POP AX
            POP BX
            RET

            ORG 4000H
FASE_2:     PUSH DX
            PUSH CX
            PUSH BX
            MOV AH, 0FFH     ; inicialmente se hipotetiza que la persona
                             ; pierde el juego
BUCLE_2:    PUSH BX
            MOV BX, CX
            INT 6
            MOV AL, [BX]     ; AL = carácter ingresado
            POP BX           ; BX apunta al carácter a adivinar
            CMP AL, [BX]
            JNZ FALLO
ACIERTO:    MOV AL, 1
            INT 7            ; imprimir carácter adivinado
            INC BX           ; siguiente carácter a adivinar
            DEC DL
            JNZ BUCLE_2      ; ¿quedan caracteres por adivinar?
            MOV AH, 0        ; la persona gana el juego
            JMP FIN_2
FALLO:      DEC DH
            JNZ BUCLE_2      ; ¿quedan intentos por realizar?
FIN_2:      POP BX
            POP CX
            POP DX
            RET
```

```
            ORG 2000H
            MOV BX, OFFSET MSJ_ING_1
            MOV AL, OFFSET MSJ_ING_2 - OFFSET MSJ_ING_1
            MOV AH, OFFSET LONG_ADIV - OFFSET MSJ_ING_2
            MOV CX, OFFSET PAL_ADIV
            CALL FASE_1
            MOV LONG_ADIV, DL
            MOV DH, INTENTOS
            MOV BX, OFFSET PAL_ADIV
            MOV CX, OFFSET CAR_ADIV
            CALL FASE_2
            CMP AH, 0
            JNZ PERDER
GANAR:      MOV BX, OFFSET MSJ_GAN
            MOV AL, OFFSET MSJ_PERD - OFFSET MSJ_GAN
            JMP FIN
PERDER:     MOV BX, OFFSET MSJ_PERD
            MOV AL, OFFSET PAL_ADIV - OFFSET MSJ_PERD
            ADD AL, LONG_ADIV
FIN:        INT 7
            INT 0
            END
```