Práctica 4. La pila: rutinas anidadas y uso compartido de parámetros.

Objetivos.

- Utilización de la pila.
- Llamada a rutinas anidadas.
- Uso compartido de parámetros.

Fundamentos teóricos.

- Páginas de la 34 a 39 del manual de prácticas.
- Transparencias de prácticas sobre la pila.

Desarrollo.

P1). Haciendo uso de la pila para solucionar el problema de las rutinas anidadas y el uso compartido de parámetros (**solución convenio guardar invocador**), desarrollar un programa que determine el número de veces que se repiten las vocales (minúsculas) en una frase introducida por consola.

El código estará formado por un segmento de datos, un programa principal y una rutina (*rutina1*) que llamará, a su vez, a diversas rutinas de manejo de la consola. Además de rutinas anidadas, deberá darse al menos un problema de registros compartidos entre el programa principal y la *rutina1*.

La siguiente guía puede servir de ayuda:

i. El segmento de datos almacenará las vocales y las cadenadas utilizadas para mostrar el resultado por pantalla, ejemplo:

```
            .data
vocales:    .asciiz  "aeiou"          # Vocales
result:     .asciiz  "x se repite "   # Resultado por pantalla
enter:      .asciiz  "\n"             # introduce un salto de línea
frase:      .space 101                # Espacio reservado para la frase
```

ii. El programa principal solicitará la introducción de una frase por consola (máx. 100 caracteres). A continuación implementará un bucle **while-do** que se repetirá 5 veces para leer una a una las vocales almacenadas en memoria y llamar iterativamente a la *rutina1*.

Nota: El registro $t0 se utilizará para controlar el número de iteraciones del bucle while-do y el registro $s0 para almacenar y pasar la vocal a la *rutina1*.

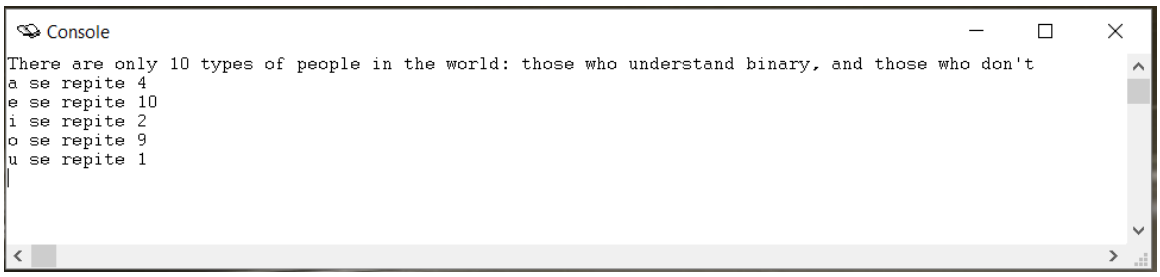
iii. La rutina1 implementará un bucle **repeat-until** para comparar la vocal (en $s0) con todos los caracteres de la frase e ir acumulando el número de coincidencias en un registro ($s1). Finalizado el bucle, la *rutina1* llamará a las rutinas de consola para mostrar el resultado por pantalla.

Nota: El registro $t0 será utilizado para controlar el número de iteraciones del bucle repeat-until y los registro $s0 y $s1 para pasar a las rutinas de consola, respectivamente, la vocal y su número de coincidencias.

iv. Las rutinas de consola imprimirán por consola la vocal y su número de coincidencias.

Nota: estas rutinas podrán denominarse: get_string, print_string, print_int.

(En este ejemplo, el problema del uso compartido de parámetros lo produce el registro $t0)



Resultado del ejercicio 1.

```
                .data
vocales:        .asciiz     “aeiou”             # Vocales
result:         .asciiz     “x se repite “      # Resultado por pantalla
enter:          .asciiz     “\n”                # introduce un enter
frase:          .space 101                      # Espacio reservado para la frase

        .text
        .globl  __start
__start:                                #Inicio del programa
        la      $4,frase                #Introduccion de frase por consola
        li      $5,100
        jal     get_string

        li      $t0,0
bucle1: beq     $t0,5,fin               # Bucle while-do
        lb      $s0,vocales($t0)        # Lee una vocal en cada iteración y carga en $s0
        subu    $sp,$sp,4               # Convenio guardar invocador → apilar ($t0)
        sw      $t0,0($sp)
        jal     rutina1
        lw      $t0,0($sp)              # Convenio guardar invocador → desapilar ($t0)
        addu    $sp,$sp,4
        add     $t0,$t0,1               # incrementar contador del bucle ($t0=$t0+1)
        j       bucle1                  # Retorno del bucle while-do

        ########## Subrutinas de programa ##########
rutina1:
        subu    $sp,$sp,4               # Apilar dirección de retorno
        sw      $31,0($sp)

        li      $t1,0                   # Inicialización de variables
        li      $s1,0
bucle2: lb      $t0,frase($t1)          # Bucle repeat-until
        bne     $s0,$t0,noacu           # comprueba si la vocal ($s0) coincide con el carácter ($t0).
        add     $s1,$s1,1               # incrementa acumulador de coincidencias ($s1=$s1+1)
noacu:  add     $t1,$t1,1               # incrementa puntero de carácter de siguiente carácter.
        bne     $t0,$0,bucle2           # Retorno bucle repeat-until

        sb      $s0,result              # sustituye en la cadena result, ‘x’ por la vocal
        la      $4,result               # Imprime por consola la cadena result
        jal     print_string
        move    $4,$s1                  # Imprime por consola el numero de coincidencias
        jal     print_int
        la      $4,enter                # Imprime por consola salto de línea
```

```
        jal     print_string

        lw      $31,0($sp)      # Desapilar dirección de retorno
        addu    $sp,$sp,4
        jr      $31             # Retorno programa principal

#---------------------------------------------------------------------
        ########## Subrutinas de consola ##########
get_string:                     # Rutina para solicitar cadena de texto
        li      $v0, 8
        syscall
        jr      $31

print_string:                   # Rutina para imprimir cadena de texto
        li      $v0, 4
        syscall
        jr      $31

print_int:                      # Rutina para imprimir un entero
        li      $v0, 1
        syscall
        jr      $31
fin:
        li      $v0,10          #Cierra consola y finaliza el programa
        syscall
        .end
#---------------------------------------------------------------------
```

Solución P1.

P2). Dado el siguiente programa y suponiendo que acaba de concluir su ejecución, complete el trozo de tabla indicando cuál es el estado en el que queda la memoria de pila. A continuación indique que mensaje aparece en la consola tras la ejecución.

```
        .data
datw:   .word   0xBEBACAFE
        .byte   0xBB, 0xE7, 0x89
        .byte   0xE9, 0xF6, 0xCB
        .byte   0xE7
        .text
        .globl  __start
__start:
        lw      $t0,datw
        sw      $0,($sp)
bucle:
        lb      $t1,datw+4($t4)
        xor     $t1,$t0,$t1
        srl     $t0,$t0,4
        sub     $sp,$sp,1
        sb      $t1,($sp)
        add     $t4,$t4,1
        bne     $t4,7,bucle
        la      $a0,($sp)
        li      $v0,4
        syscall
```

Origen→

addr	Datos
0x7fffeffc	0x00
0x7fffeffb	0x45
0x7fffeffa	0x48
0x7fffeff9	0x43
0x7fffeff8	0x45
0x7fffeff7	0x4C
0x7fffeff6	0X20
0x7fffeff5	0X59

¿ Qué mensaje aparece en la consola tras la ejecución ?

Y_LECHE

Carácter	Ascii (Hex)	Ascii (dec)
NULL	0x00	00
espacio	0x20	32
“A”	0x41	65
“B”	0x42	66
“C”	0x43	67
“D”	0x44	68
“E”	0x45	69
“F”	0x46	70
“G”	0x47	71
“H”	0x48	72
“I”	0x49	73
“J”	0x4A	74
“K”	0x4B	75
“L”	0x4C	76
“M”	0x4D	77
“N”	0x4E	78
“O”	0x4F	79
“P”	0x50	80
“Q”	0x51	81
“R”	0x52	82
“S”	0x53	83
“T”	0x54	84
“U”	0x55	85
“V”	0x56	86
“W”	0x57	87
“X”	0x58	88
“Y”	0x59	89
“Z”	0x5A	90