Práctica 3. Modos de direccionamiento, consola y estructuras de decisión.

Objetivos.

- Aprender los modos de direccionamiento del MIPS.
- Manejar las llamadas al sistema y uso de subrutinas.
- Como imprimir y leer desde consola.
- Manipulación y carga de datos de memoria.
- Entender las estructuras de decisión.

Fundamentos teóricos.

- Página 23 del manual de prácticas.
- Página 31, 32 del manual de prácticas.
- Transparencias sobre estructuras de decisión

Desarrollo

P1) Crear un programa con la estructura de decisión *while-do* que escriba en la consola los caracteres ASCII de la “A” a la “T” separados por espacios.

```
        .data
char:   .space 1
        .byte   00
space: .asciiz " "

        .text
        .globl  __start
__start:
        li      $t0, 0x41                       # valor inicial = A
loop:
        beq     $t0, 0x55, exit
        sb      $t0, char
        la      $a0, char
        jal     print_string
        add     $t0, $t0, 1
        la      $a0, space
        jal     print_string
        j       loop

print_string:
        li      $v0, 4
        syscall
        jr      $ra

exit:
        .end
```

Solución P1.

```
        .data
chars:  .space 45

        .text
        .globl __start
__start:
        li      $t0, 0x41               # ascii para "A" (valor inicial)
        li      $t1, 0x20               # ascii para " " (espacio)
        li      $t3,0                   # puntero a memoria
loop:
        beq     $t0, 0x55, print        #ascii 0x55 --> "U"
        sb      $t0, chars($t3)
        sb      $t1, chars+1($t3)
        add     $t0, $t0, 1
        add     $t3, $t3, 2
        j loop

print:  la      $a0, chars              # Imprime los caracteres A-T
        li      $v0, 4
        syscall
        .end
```

Solución P1 (Alternativa).

P2) Escribir el código ensamblador del siguiente juego basándose en la estructura *repeat-until*:

i) El usuario debe introducir el valor de una carta, que deberá estar comprendido entre 1 y 10.

ii) El programa solo deberá continuar si la carta introducida por el usuario está entre los valores válidos. En caso de fallo simple, el usuario tendrá otro intento para introducir la opción deseada. En caso de fallo doble, el programa terminará con el mensaje por consola “Número de intentos agotado. Fin del juego”.

iii) El usuario podrá elegir hasta 5 cartas. Esto es, los pasos i) y ii) se deberán ejecutar 5 veces. Se deberá emplear la estructura *repeat-until*.

iv) Al finalizar, el programa deberá sacar por consola la carta más alta introducida por el usuario. Se deberá emplear la estructura *repeat-until* para implementar el algoritmo de búsqueda.

```
            .data
card:       .space    5
            .byte     0xFF
msg1:       .asciiz "\n Introduce una carta "
msg2:       .asciiz "\n Número de intentos agotado. Fin del juego "
msg3:       .asciiz "\n La carta mayor es "

            .text
            .globl __start
__start:
            li        $t2, 1      ## contador de numero de fallos
            li        $t3, 0      ## contador de iteraciones

begin:      la        $a0, msg1                 # demand card
            jal       print_string

            jal       read_int            # read and store card
            move      $t0, $v0
            sb        $t0, card($t3)

            blt       $t0, 1, notanumber  # verify card
            bgt       $t0, 10, notanumber
            add       $t3, $t3, 1               # it is between 1 and 10

            bne       $t3, 5, begin             # end of loop?

            li        $t3, 0                    # iteracion para busqueda de mayor
find:       lb        $t0, card
            lb        $t1, card + 1($t3)
            bgt       $t1, $t0, actualiza
            move      $t2, $t0            # $t2 guardará el mayor
            j         sigue
actualiza:  move      $t2, $t1
sigue:      sb        $t2, card           # en card guardo el mayor de momento
            add       $t3, $t3, 1
            bne       $t3, 4, find

            la        $a0, msg3
            jal       print_string

            lb        $a0, card
            jal       print_int

            j         __start

notanumber:
            beq       $t2, 2 final_message
            add       $t2, $t2, 1               # contador de fallos
            j         begin

final_message:
            la        $a0, msg2
            jal       print_string
            j         fin

######################################## subrutinas de syscall
read_int:   li        $v0, 5
            syscall
            jr        $ra

print_string: li      $v0, 4
            syscall
            jr        $ra

print_int:  li        $v0, 1
            syscall
            jr        $ra

fin:
            .end
```

Solución P2.

P3) Realizar un programa de acuerdo a las siguientes especificaciones:

i) Un mensaje de consola ofrecerá 3 opciones al usuario, que deberá elegir entre:
   [1]. Añadir un nuevo usuario
   [2]. Buscar un usuario existente
   [3]. Eliminar un usuario
ii) Para implementar estas opciones se debe emplear una estructura de tipo switch-case.
iii) El programa deberá permitir almacenar hasta 100 usuarios
iv) Si el usuario no introduce un número válido, el programa deberá terminar.

Variante A:

v) Si el usuario elige la opción [1], el programa deberá sacar por pantalla el mensaje “Opción 1 seleccionada” y volver al menú principal (paso i).
vi) Si el usuario elige la opción [2], el programa deberá sacar por pantalla el mensaje “Opción 2 seleccionada” y volver al menú principal (paso i).
vii) Si el usuario elige la opción [3], el programa deberá sacar por pantalla el mensaje “Opción 3 seleccionada” y volver al menú principal (paso i).

Variante B:

v) Si el usuario elige la opción [1], el programa deberá pedir el nombre del usuario (de máximo 30 caracteres) y su edad. Esta información deberá almacenarse en el segmento de datos. Una vez introducida, el programa volverá al menú principal (paso i).
vi) Si el usuario elige la opción [2], el programa deberá pedir el nombre del usuario. En caso de que este nombre coincida exactamente con un nombre ya almacenado, el programa deberá sacar por consola el nombre y la edad de dicho usuario. Si el usuario no existiese, se informará con el correspondiente mensaje y se volverá al menú principal (paso i).
vii) Si el usuario elige la opción [3], el programa deberá pedir el nombre del usuario que se desea eliminar. En caso de que este nombre coincida exactamente con uno ya registrado, el programa deberá borrar esta información del segmento de datos, y sacar el mensaje correspondiente por consola. Una vez eliminado el programa volverá al menú principal.

```
            .data
msg1:       .asciiz "\n [1] Añadir un nuevo usuario "
msg2:       .asciiz "\n [2] Buscar un usuario existente "
msg3:       .asciiz "\n [3] Eliminar un usuario \n\n\t"
msg4:       .asciiz "\n Opción 1 seleccionada"
msg5:       .asciiz "\n Opción 2 seleccionada"
msg6:       .asciiz "\n Opción 3 seleccionada"

            .text
            .globl  __start
__start:
            li      $t3, 0              # $t3 guarda el numero de usuarios introducidos
inicio:
            la      $a0, msg1
            jal     print_string
            la      $a0, msg2
            jal     print_string
            la      $a0, msg3
            jal     print_string
            jal     read_int
            move    $t0, $v0

            blt     $t0, 1, fin         # comprueba intervalo de opciones
            bgt     $t0, 3, fin
            sub     $t0, $t0, 1                 # resto 1 para aplicar estructura optimizada switch-case
            sll     $t1, $t0, 3                 # multiplica por 8 para sacar la dirección de salto
            la      $t2, op1
            add     $t2, $t2, $t1
            jr      $t2
op1:
            jal     opcion1
            j       fin
op2:
            jal     opcion2
            j       fin
op3:
            jal     opcion3
            j       fin

opcion1:                    ############# opcion 1
            la      $a0, msg4
            li      $v0, 4
            syscall
            jr      $ra

opcion2:                    ############# opcion 1
            la      $a0, msg5
            li      $v0, 4
            syscall
            jr      $ra

opcion3:                    ############# opcion 1

            la      $a0, msg6
            li      $v0, 4
            syscall
            jr      $ra

############ rutinas de consola
read_int:
            li      $v0, 5
            syscall
            jr      $ra

print_string:
            li      $v0, 4
            syscall
            jr      $ra

fin:
            li      $v0,10
            syscall
            .end
```

Solución P3 (VARIANTE A).