Práctica 5. Usos de la pila: subrutinas anidadas, subrutinas recursivas e invertir orden.

1. Introducción

La pila, se usa para almacenar información temporalmente de nuestros programas. Debido a su característica LIFO (*Last In, First Out*) es útil, por ejemplo, para invertir el orden de los elementos de un vector.

Se denominan subrutinas anidadas aquellas que contienen llamadas a otras subrutinas. Por ejemplo, una subrutina que indica si un número es primo que a su vez contiene una llamada a otra subrutina que sirve para indicar cuántos divisores tiene el número que se le pasa como parámetro.

Un caso especial de subrutinas anidadas son las subrutinas recursivas, que se llaman a sí mismas. Por ejemplo, para calcular el factorial de un número *n* se puede llamar a la propia subrutina con el parámetro *n-1*. Siempre tiene que haber un caso trivial que es el que garantiza que las llamadas no sean infinitas.

2. Ejemplo de uso de la pila para invertir una cadena

Listado 1: Ejemplo de uso de la pila para invertir una cadena.

```
      .data
cad: .asciiz "Introduzca una cadena: "
cad2: .asciiz "La cadena invertida es: "
cadleida:   .space 100
cadinv:     .space 100
      .globl main
      .text
main:
      li $v0, 4
      la $a0, cad
      syscall

      li $v0, 8
      la $a0, cadleida
      li $a1, 100
      syscall

      #Apilo el 0 en la pila para saber cuál es el principio de la pila
      addi $sp, $sp, -1
      sb $zero, ($sp)

      la $t0, cadleida #Puntero a cadena leida
      la $t2, cadinv   #Puntero a cadena invertida

      #Mientras no llegue al \0, leo una letra y la apilo
bucle:
      lb $t1, ($t0)
      beq $t1, $zero, desapilo
      addi $sp, $sp, -1
      sb $t1, ($sp)
      addi $t0, $t0, 1 #El puntero apunta a la siguiente letra
      j bucle
desapilo:
      #Mientras no desapile un 0
      lb $t1, ($sp)
      addi $sp, $sp, 1
```

```
      beq $t1, $zero, fin
      sb $t1, ($t2) #EScribo la letra en la cadena
      addi $t2, $t2, 1 #El puntero apunta al siguiente hueco
      j desapilo

fin:
      li $v0, 4
      la $a0, cad2
      syscall


      li $v0, 4
      la $a0, cadinv
      syscall

      li $v0, 10
      syscall
```

3. Subrutinas

Si se quiere que una subrutina llame a otra (o bien que se llame a sí misma), mediante la instrucción `jal`, existe el problema de que automáticamente se modifica el contenido del registro `$ra`. Así, se pierde cualquier valor que este registro pudiera tener.

Si el programa principal llama a una subrutina A, que a su vez llama a otra subrutina B, se puede volver desde B a A mediante `jr $ra`, pero al haberse sobreescrito el contenido de `$ra` en la segunda llamada, será imposible volver desde A al programa principal.

La idea es que, antes de realizar una llamada desde una subrutina a otra se salve el contenido de `$ra` en la pila. Así, las direcciones de retorno de las distintas llamadas anidadas/recursivas quedarán almacenadas a medida que se vayan produciendo. Para realizar los retornos, se desapilan las direcciones de retorno y se salta a éstas.

Veamos un esquema de llamada a subrutinas anidadas:

Listado 2: Esquema de subrutinas anidadas.

```
      .data
      .globl main
      .text
main:
      # llamamos a la subrutina_1, que a su vez contendrá una llamada
      # a la subrutina_2. Resolveremos el retorno usando la pila.
      jal subrutina_1

      li $v0, 10
      syscall

subrutina_1:

      ...

      # antes de realizar una llamada anidada, tomamos la precaución
      # de apilar la dirección de retorno de subrutina_1, para así
      # poder volver más adelante al programa principal

      addi $sp, $sp, -4
      sw $ra, ($sp)
```

```
      jal subrutina_2

      ...

      # antes de volver al programa principal, desapilamos la dirección
      # de retorno previamente almacenada
      lw $ra, ($sp)
      addi $sp, $sp, 4

      jr $ra # volvemos al main

subrutina_2:

      ...

      jr $ra # volvemos a la subrutina_1, al ser un anidamiento simple
             # no hemos necesitado apilar la dirección de retorno a la
             # subrutina_1
```

Un caso de subrutina anidada son las subrutinas recursivas. En ese caso siempre habrá un caso trivial, que devolverá el resultado y saldrá de la función con `jr $ra`. El caso recursivo necesitará apilar el registro `$ra` antes de la llamada recursiva y desapilar el registro `$ra` antes de finalizar dicho caso recursivo. Además de apilar el registro `$ra`, habrá que apilar toda la información necesaria que se utilice después de la llamada recursiva, puesto que al hacer la llamada no se garantiza que se mantenga el valor de los registros `$a0, $t0`,... etc

Listado 3: Ejemplo de subrutina recursiva.

```
      .data
cad: .asciiz "Introduzca un número: "
cad2: .asciiz "El factorial es: "
      .globl main
      .text
main:
      li $v0, 4
      la $a0, cad
      syscall

      li $v0, 5
      syscall

      move $a0, $v0 #Argumento en $a0: número leído
      jal factorial
      move $t0, $v0 #Resultado en $v0: factorial

      li $v0, 4
      la $a0, cad2
      syscall

      li $v0, 1
      move $a0, $t0
      syscall

      li $v0, 10
      syscall

factorial:
      li $t0, 1
      beq $a0, $zero, caso_trivial
```

```
      beq $a0, $t0, caso_trivial
      #Caso recursivo: apilo $ra
      addi $sp, $sp, -4
      sw $ra, ($sp)
      #Apilo $a0 porque lo necesito después
      addi $sp, $sp, -4
      sw $a0, ($sp)

      addi $a0, $a0, -1 #Argumento en $a0: n-1
      jal factorial

      #Desapilo $a0
      lw $a0, ($sp)
      addi $sp, $sp, 4
      mult $a0, $v0   #Multiplico n * factorial(n-1)
      mflo $v0    #Resultado en $v0

      #Desapilo $ra
      lw $ra, ($sp)
      addi $sp, $sp, 4
      jr $ra   #Fin de la función (caso recursivo)


caso_trivial:
      li $v0, 1 #Resultado vale 1
      jr $ra        #Fin de la función (caso trivial)
```

3. Ejercicios

Se pide:

1. Copiar, ensamblar y probar los programas de los listados 1 y 3.

2. Crear un programa que pida una cadena de texto y un número. Después debe llamar a una subrutina a la que se le enviarán la cadena de texto y el número como parámetros. Dicha subrutina devolverá un 0 si el número insertado es inferior o igual al número de caracteres de la cadena insertada y devolverá un 1 en caso contrario. Para realizar este cálculo deberá llamar, dentro de la subrutina a otra subrutina que se encargará de contar los caracteres que posee la cadena y devolverá el número de caracteres.

3. Realice un programa que pida un número por teclado (se debe controlar si el número insertado es positivo, en caso contrario, se mostrará un mensaje de error) e incluya una subrutina que indique si el número de divisores de un número es par o impar. Para calcular los divisores, dicha subrutina llamará a otra subrutina que calcula el número de divisores del número.
   Aclaraciones:
   - subrutina 1, devuelve 0 si es impar o 1 si es par
   - subrutina 2, devuelve el número de divisores del número insertado

4. Ejercicios de repaso

1. Realice un programa que solicite al usuario una cadena de texto y que cambie las letras 'a' por el carácter 'x'. Para ello ha de usar una función que se le pase un puntero a la cadena y las dos letras y realice dicha tarea.

2. Realice un programa que solicite al usuario un número y compruebe si es un palíndromo. Tras calcular el palíndromo, el programa preguntará al usuario si desea seguir introduciendo más números para comprobar, si inserta un '1' volverá a solicitar al usuario insertar un número, si inserta un '0' saldrá del programa. NOTA: los palíndromos son números que se leen igual de izquierda a derecha que de derecha a izquierda, por ejemplo "10001" es un palíndromo.
3. Teniendo el siguiente segmento de datos:

```
      .data
datos1:
      .byte 14, 23
      .align 2
      .word 47
      .align 1
      .half 9, 12, 15
      .align 2
datos2:
      .space 4
      .byte 'c','o'
```

   a. Sumar el word con valor 47, con el half de valor 15 y guardarlo en el .space 4 como dato word, utilizando la etiqueta datos1.
   b. Guardar el carácter 'd' y el carácter 'p' en la posición del half con valor 12, utilizando la etiqueta datos2. NOTA: 'd' es el carácter siguiente a la letra 'c' y 'p' es el carácter siguiente a la letra 'p'.