Práctica 14. Saltos

Objetivos

- Conocer cómo se realizan saltos en ensamblador.

1. Introducción

Para poder repetir código se utilizan las instrucciones de salto, o bien condicional o bien incondicionalmente.

El lenguaje ensamblador no dispone de las clásicas estructuras de control de flujo de programa (`if`, `for`, `while`) presentes en los lenguajes de alto nivel. Normalmente, como casi siempre cuando se trata de programar en ensamblador, hay que utilizar mecanismos de bajo nivel para implementar estas estructuras “a mano”.

Las instrucciones que permiten saltar a otro punto del programa en función del valor de un registro, o bien en función del resultado de comparar dos registros entre sí son las instrucciones de **salto condicional.**

También se ofrece la posibilidad de saltar a un punto determinado del código sin necesidad de que se cumpla ninguna condición. Las instrucciones que lo permiten son las de **salto incondicional.**

Por último, hay un tercer tipo de instrucciones de salto. Son las instrucciones de **llamadas a subrutinas.**

En general, las instrucciones de salto harán referencia a otras instrucciones a las que se quiera "saltar". Para hacer referencia a instrucciones del segmento de texto se utilizarán únicamente **etiquetas** que precedan a las instrucciones de destino (las cuales estarán en el segmento de texto).

2. Instrucciones de salto condicional

En el repertorio de instrucciones de RISC-V se incluyen las siguientes instrucciones de salto condicional:

- `beq rs1, rs2, etiqueta` salta a `etiqueta` si los registros son iguales
- `bne rs1, rs2, etiqueta` salta a `etiqueta` si los registros son distintos
- `blt rs1, rs2, etiqueta` salta a `etiqueta` si el primer registro es menor que el segundo
- `bge rs1, rs2, etiqueta` salta a `etiqueta` si el primer registro es mayor o igual que el segundo

3. Instrucciones de salto incondicional

En ocasiones, se quiere saltar incondicionalmente a una instrucción precedida por una etiqueta. Para ello se usa la instrucción `jal x0, etiqueta` o la pseudoinstrucción `j etiqueta`.

4. Subrutinas

El diseño de un programa puede simplificarse si se utilizan adecuadamente las subrutinas. Éstas permiten dividir un problema grande en otros más pequeños y evitan tener que repetir fragmentos de código. Las estudiaremos en la práctica siguiente.

5. Ejemplos

5.1. Ejemplo de uso de llamadas al sistema para leer/escribir cadenas

El siguiente programa lee una cadena y la muestra por pantalla.

```
1   .data
2 cad1: .string "Introduzca una cadena: "
3 cad2: .zero 100
4   .text
5   .globl main
6 main:
7   li a7, 4       #Llamada al sistema para imprimir cadena
8   la a0, cad1 #a0 = dirección de la cadena a imprimir
9   ecall
10
11  li a7, 8  #Llamada al sistema para leer cadena
12  la a0, cad2  #a0 = dirección donde guardo la cadena leída
```

```
13   li a1, 100  #a1 = longitud de la cadena a leer
14   ecall
15
16   li a7, 4
17   la a0, cad2
18   ecall
19
20   li a7, 10
21   ecall
```

5.2. Ejemplo de uso de instrucciones de salto

Un ejemplo de uso de instrucciones de salto es el siguiente programa, que lee un número e imprime si es positivo o negativo:

```
1    .data
2
3  cad: .string "Introduzca un número: "
4  pos: .string "El número es positivo o cero"
5  neg: .string "El número es negativo"
6
7    .globl main
8    .text
9  main:
10   li a7, 4
11   la a0, cad
12   ecall
13
14   li a7, 5    #Llamada al sistema para leer entero
15   ecall
16   mv t0, a0 #El entero leído queda en a0
17
18   blt t0, x0, esnegativo #salta a esnegativo si t0 es < 0
19   #Si no salta, es positivo o 0
20   li a7, 4
21   la a0, pos
22   ecall
23   #salta para no ejecutar las instrucciones siguientes
24   j fin #también se puede poner jal x0, fin
25
26 esnegativo:
27   #si es negativo habrá saltado aquí, imprimo "es negativo"
28   li a7, 4
29   la a0, neg
30   ecall
31
32 fin:  #fin del programa
33   li a7, 10
```

```
  ecall
```

Otro ejemplo de código que calcula la longitud de una cadena:

```
  .data
cad1: .string "Introduzca una cadena: "
cad2: .zero 100
cad3: .string "La longitud de la cadena es: "
  .text
  .globl main
main:
  li a7, 4
  la a0, cad1
  ecall

  li a7, 8
  la a0, cad2
  li a1, 100
  ecall

  la t2, cad2     #Puntero a la cadena

  li t0, 0 #contador de caracteres
bucle:
  lb t1, 0(t2) # leo una letra y la guardo en t1
  beq t1, zero, fin #si es el \0 se ha terminado la cadena
  addi t0, t0, 1  # sumo uno al contador de la longitud
  addi t2, t2, 1  # muevo el puntero para que apunte a la letra
   siguiente
  j bucle # salto a bucle

fin:
  li a7, 4
  la a0, cad3
  ecall

  li a7, 1
  mv a0, t0  # Imprimo el contador de caracteres
  ecall

  li a7, 10
  ecall
```

6. Ejercicio

6.1. Ejercicio

Realice un programa que lea una cadena por teclado y calcule la suma de todos los valores ASCII de los caracteres que la forman y además el número de caracteres iguales a la letra 'a' que forman dicha cadena (dos contadores). Nota: para la lectura de la cadena debe emplearse la llamada al sistema *read_string*. Hay que imprimir los resultados por pantalla (la suma de los valores ASCII y el número de letras 'a').

6.2. Ejercicio

Realice un programa que pida un número por teclado y cuente el número de unos que tiene cuando se representa en binario, imprimiendo el resultado por pantalla.

6.3. Ejercicio

Considérese el siguiente fragmento de código que representa el segmento de datos:

```
1   .data
2 datos1:  .half 4,7,2
3   .align 2
4   .word 1, -5
5   .byte 4
6   .align 1
7   .half 3
8   .align 2
9   .word -3
10  .zero 2
```

Escriba un programa que, utilizando los modos de direccionamiento realice las siguientes operaciones sin modificar el segmento de datos:

- Calcular la suma del primer `word` que vale 1 colocado a partir de la etiqueta `datos1` y del `half` que tiene valor 3 y colocar este resultado en el espacio de 2 bytes. Imprimir y comprobar el valor.
- Calcular la suma del `word` que vale -5 y del que vale -3 y colocar este resultado (usando 2 bytes) donde el `half` que vale 2. Imprimir y comprobar el valor.