Práctico 3 : Programación RISC-V, uso del simulador RARS

Secciones principales de un programa típico.

Comentarios
Los comentarios comienzan con el símbolo # y no son ejecutados por el procesador. Se utilizan para documentar el código y explicar las instrucciones.
Ejemplo:

```
# Este es un comentario que describe el código
```

Segmento de Datos (.data)
Esta sección contiene las variables y datos que utiliza el programa. Aquí se definen cadenas de texto, constantes y otros valores que se almacenan en memoria.
Propósito:

- Declarar datos estáticos, como cadenas o números.
- Asignar etiquetas para referenciar los datos desde el segmento de código.

Ejemplo:

```
      .data
str:    .asciz "Hola mundo en RISC-V!\n"  # Define una cadena de texto
```

Segmento de Código (.text)
Esta sección contiene las instrucciones ejecutables del programa. Es donde se escribe la lógica del programa utilizando las instrucciones del conjunto RISC-V.
Propósito:

- Definir el flujo de ejecución del programa.
- Implementar operaciones como cálculos, control de flujo y llamadas al sistema.

Ejemplo:

```
      .text
main:
    la a0, str
    li a7, 4
    ecall
    li a7, 10
    ecall
```

Registros
RISC-V tiene 32 registros de propósito general (x0-x31) y un registro especial llamado Program Counter (PC). Los registros almacenan valores temporales durante la ejecución del programa.

- Registro x0: Siempre tiene el valor 0 y no puede modificarse.
- Program Counter (PC): Contiene la dirección de la próxima instrucción a ejecutar.

Llamadas al Sistema (ecall)
Las llamadas al sistema permiten interactuar con el entorno operativo, como imprimir texto o finalizar el programa.
Ejemplo:

```
li a7, 4
```

```
ecall
```

Flujo de Control
Se utilizan instrucciones como saltos (j, bgt, etc.) para controlar el flujo del programa.
Ejemplo:

```
bucle:
    addi x5, x5, 1    # Incrementar contador
    bgt x5, x10, fin # Si x5 > x10, salir del bucle
    j bucle          # Repetir bucle

fin:
    li a7, 10
    ecall
```

Parte 1: Primer Programa - "Hola Mundo"

Paso 1: Escribir el código del programa en el simulador.

```
# Programa hola mundo en RISC-V
        .data
str:    .asciz "Hola mundo en RISC-V!\n"

        .text
main:
        la a0, str
        li a7, 4
        ecall

        li a7, 10
        ecall
```

Paso 2: Guardar el programa con la extensión .asm
Paso 3: Ensamblar el programa.

- Hacer clic en Assemble para ensamblar el programa.
- Esta acción **analiza** el programa en **ensamblador** y genera el **código máquina**. Si no hay errores, aparecerá el mensaje: Assemble: operation completed successfully.

Paso 4: Ejecutar el programa.

- Hacer clic en Run para ejecutar el programa.
- En la consola inferior del simulador aparecerá el mensaje: Hola mundo en RISC-V!

Parte 2: Ejecución Paso a Paso

Una característica útil del simulador RARS es la posibilidad de ejecutar programas paso a paso para observar cómo se modifican los registros y qué hace cada instrucción.

Instrucciones para Ejecutar Paso a Paso

1. Desde la pestaña Edit, ensambla tu programa como antes.
2. Haz clic en Run one step at a time (botón justo a la derecha del botón Run).
    - La primera instrucción se resaltará en amarillo.

3. Sigue presionando este botón para ejecutar cada instrucción individualmente.
4. Observa cómo cambian los valores de los registros en la parte derecha del simulador.

Preguntas de Análisis
Ejecutar nuevamente el programa "Hola Mundo" paso a paso y responde las siguientes preguntas:

- ¿Qué instrucción imprime el mensaje?
- ¿Qué instrucción finaliza el programa?
- ¿Cuántas instrucciones se ejecutan hasta que el programa termina?
- ¿Cuál es la dirección de memoria donde está situada la primera instrucción?

Parte 3: Ejercicios

1. **Modificar "Hola Mundo"**: Modificar el programa de "Hola Mundo" para que imprima un mensaje personalizado.
2. **Asignación de registros**: Escribir un programa que asigne los siguientes valores a los registros indicados: x3=3, x4=4, x5=5, x6=6, x7=x7 y x8=8. Ejecutarlo paso a paso para comprobar que funciona correctamente.
3. **Secuencia de valores**: Escribir un programa donde:
    - x3 tome los valores 0,1,2,3,4,5...
    - x4 tome los valores 0,3,6,9,12,15...
    - x5 tome los valores 0,5,10,15,20,25... indefinidamente. Ejecutarlo paso a paso para verificar su funcionamiento.
4. **Ejecutar y analizar código**:

```
        .text
addi x3, x0, 10
a:
addi x3,x3,-1
bgt x3,x0, a


li a7, 10
ecall
```

   ¿Qué hace este código?

5. **Uso de li y mv**: Escribir un programa para inicializar los registros x5, x6, x7, x8 con los valores 5, 6, 7 y 8 respectivamente usando li. Luego, transferir estos valores a los registros x15, x16, x17 y x18 usando mv. Contar cuántas instrucciones tiene el programa.
6. **Suma de los primeros N números naturales**: Escribir un programa que calcule la suma de los primeros N números naturales.

Parte 4: Preguntas de Análisis

1. ¿Cómo se almacenan y manipulan los valores en los registros?
2. ¿Qué diferencias existen entre `li`, `la` y `mv` en ensamblador RISC-V?
3. ¿Qué sucede si no se usa la instrucción `ecall` al final del programa?