Universidad Nacional Autónoma de México

Facultad de Ciencias

Organización y Arquitectura de Computadoras
2025-2

Práctica 06

Docentes:
José Galaviz Ricardo Pérez Ximena Lezama

Autores:
Fernanda Ramírez Juárez Ianluck Rojo Peña

Fecha de entrega: Jueves 27 de marzo de 2025

Preguntas.

1. ¿Qué hacen las instrucciones de tipo FR y de tipo FI? Da algunos ejemplos de instrucciones de este tipo y menciona por que están separadas de las otras 3 principales.

   Las instrucciones de tipo **FR** *(Floating-Point Register)* y **FI** *(Floating-Point Immediate)* son aquellas que se encargan de realizar operaciones con números de punto flotante. A diferencia de las instrucciones que trabajan con enteros, estas están diseñadas específicamente para manejar números con partes fraccionarias, lo cual es esencial para cálculos científicos, gráficos y muchas otras aplicaciones.

   La diferencia principal entre las instrucciones de tipo **FR** *(Floating-Point Register)* y **FI** *(Floating-Point Immediate)* radica en el tipo de operando que utilizan. Las instrucciones FR operan principalmente con valores que se encuentran en los registros de punto flotante y las instrucciones FI operan con un valor inmediato (una constante codificada directamente en la instrucción) y/o realizan transferencias de datos entre la memoria y los registros de punto flotante.

   Ejemplos de instrucciones FR:

   - `ADD.S $f2, $f4, $f6`: Suma el contenido de los registros de punto flotante $f4 y $f6, y guarda el resultado en $f2 (operación de precisión simple).
   - `MUL.D $f8, $f10, $f12`: Multiplica el contenido de los registros de punto flotante $f10 y $f12, y guarda el resultado en $f8 (operación de doble precisión).
   - `DIV.S $f14, $f16, $f18`: Divide el contenido del registro $f16 entre el de $f18, y guarda el resultado en $f14 (precisión simple).
   - `SQRT.D $f20, $f22`: Calcula la raíz cuadrada del valor en $f22 y la guarda en $f20 (doble precisión).

- `C.LT.S $f24, $f26`: Compara si el valor en $f24 es menor que el valor en $f26. El resultado de la comparación se guarda en un registro de condición especial.

Ejemplos de instrucciones de tipo FI:

- `L.S $f28, offset($rs)`: Carga un valor de punto flotante de precisión simple desde la dirección de memoria calculada como *offset* + contenido de $rs al registro $f28. Aunque involucra memoria, la carga de un valor de punto flotante se considera dentro de este grupo por su naturaleza de dato.
- `S.D $f30, offset($rt)`: Almacena el valor de punto flotante de doble precisión del registro $f30 en la dirección de memoria calculada como *offset* + contenido de $rt.

Estas instrucciones están separadas de las otras tres principales *(tipo R, tipo I y tipo J)* por lo siguiente:

- Los números de punto flotante se representan y manipulan de manera distinta, lo que crea la necesidad de tener hardware especializado dentro del procesador para realizar las operaciones de manera eficiente.
- Las arquitecturas tienen un conjunto de registros separado (como los registros $f en MIPS) para almacenar valores de punto flotante. Esto permite optimizar el diseño del hardware para cada tipo de dato.
- Las operaciones de punto flotante son más complejas a nivel de hardware que las operaciones enteras básicas. Separarlas permite que la unidad de enteros se mantenga más simple y rápida para las tareas que no requieren aritmética de punto flotante.
- Las instrucciones de punto flotante tienen formatos diferentes para acomodar los operandos de punto flotante y los códigos de operación específicos para estas operaciones.

2. ¿Qué es la Portabilidad de Arquitecturas **(Cross-Architecture Porting)**?

   La Portabilidad de Arquitecturas **(Cross-Architecture Porting)** se refiere a la capacidad de un programa o software para una arquitectura de hardware de ser ejecutado en otra arquitectura de hardware diferente con una mínima o ninguna modificación en su código fuente.

   Lograr una portabilidad completa a nivel de código ensamblador es extremadamente difícil, ya que el lenguaje ensamblador es naturalmente específico de una arquitectura en particular. Cada arquitectura tiene su propio conjunto de instrucciones, organización de registros, modos de direccionamiento y convenciones de llamada.

   En la práctica, la portabilidad de arquitecturas generalmente se logra escribiendo software en lenguajes de alto nivel **(como C, C++, Java o Python)** que son independientes de la arquitectura base. Luego, este código fuente se compila para la arquitectura de destino utilizando un compilador específico para esa arquitectura. El compilador se encarga de traducir el código de alto nivel a las instrucciones de ensamblador nativas de la arquitectura objetivo.

   Por lo tanto, cuando hablamos de portabilidad de arquitecturas, rara vez se refiere a tomar un programa escrito directamente en ensamblador para una arquitectura y ejecutarlo sin

cambios en otra. Más bien, implica diseñar sistemas y escribir código de manera que pueda ser adaptado y recompilado fácilmente para diferentes plataformas de hardware.

3. Menciona 5 llamadas a sistema *(syscall)* que puedes usar en MIPS. Menciona su código de instrucción y qué es lo que hace.

Servicio	**Operación**	**Código en $v0**	**Argumentos**	**Resultado**
print_int	Imprime un número entero (32 bits)	1	$a0 = entero a imprimir	Ninguno
print_string	Imprime una cadena de caracteres con terminación null	4	$a0 = dirección en memoria de la cadena a imprimir	Ninguno
print_char	Imprime un carácter	11	$a0 = carácter a imprimir	Ninguno
read_int	Lee el número entero ingresado por el usuario	5	Ninguno	Regresa el número en $v0
read_string	Igual que la función *fgets()* de la biblioteca estándar en C	8	$a0 = dirección en memoria del buffer de entrada de la cadena, $a1 = longitud del buffer de la cadena (n)	Ninguno

4. ¿Cuáles son los 3 tipos principales de instrucciones? Menciona que comportamiento tiene cada tipo de instrucción.

1) **Operaciones de tipo R (Register-type):**
Las instrucciones de tipo R se usan cuando todos los valores involucrados en la operación se encuentran en los registros del procesador. Estas instrucciones realizan operaciones aritméticas y lógicas sin acceso a memoria.

El formato general de una instrucción tipo R es:

```
OP  rd, rs, rt
```

Donde:

- **rs** y **rt** son los registros fuente.
- **rd** es el registro destino donde se almacenará el resultado.

Aquí un ejemplo de la instrucción `add`:

```
add $t0, $t1, $t2   # $t0 = $t1 + $t2
```

2) **Operaciones de tipo I (Immediate-type):**
Las instrucciones de tipo I se utilizan cuando es necesario operar con un valor inmediato (constante) y un registro. Los valores inmediatos en MIPS tienen un máximo de

16 bits, por lo que números más grandes no pueden ser manejados directamente con instrucciones inmediatas.

El formato general de una instrucción tipo I es:

```
OP  rt, IMM(rs)
```

Donde:

- **rt** es el registro donde se almacenará el resultado.
- **IMM** es el valor inmediato.
- **rs** es el registro base (en operaciones con memoria).

Ejemplo de la instrucción `addi`:

```
addi $t0, $t1, 10   # $t0 = $t1 + 10
```

3) **Operaciones de tipo J (Jump-type):**
Las instrucciones de tipo J son utilizadas para realizar saltos a direcciones específicas en la memoria. Estas instrucciones tienen mayor capacidad para manejar valores inmediatos, ya que trabajan con direcciones de memoria más grandes.

El formato general de una instrucción tipo J es:

```
OP  LABEL
```

Donde:

- **LABEL** es la dirección de memoria o etiqueta a donde se debe saltar.

Ejemplo de la instrucción `j`:

```
j etiqueta  # Salta a la dirección de memoria etiquetada
```