Punto Flotante

-MIPS utiliza IEEE 754 para números de punto flotante (con coma).

-32 registros desde F0 al F31.

-Único tipo de dato: .double.

-Las instrucciones son casi iguales con la diferencia de que se agrega .D al final.

Ejemplo:

-cargar: l.d
-suma: add.d
-resta: sub.d
-almacenamiento: s.d

-No todas las etapas EX tardan lo mismo: generales(1 ciclo), multiplicación en pf (7 ciclos) suma en pf (4 ciclos), división en pf (24 ciclos).

-Estas instrucciones EX se ejecutan en paralelo, esto tiene ventajas en tiempo de ejecución pero ocurren varios tipos de atascos

Atascos estructurales: provocados por conflictos de recursos. Dos instrucciones intentan acceder a la etapa MEM de forma simultánea. Tiene prioridad la que entre primero al cauce.

Atascos WAR y WAW: ocurren cuando hay dependencia de datos entre dos instrucciones (igual al RAW).
Una instrucción sobrepasa a una anterior, queriendo escribir un registro pendiente de lectura (WAR) o escritura (WAW). El simulador puede generarlos si detecta una situación potencial (puede que realmente no suceda) de dependencia WAR o WAW.

Subrutinas

- Igual que en MSX88 se pueden definir subrutinas se llaman con la instrucción JAL
- No hay manejo implícito de la pila, la dirección de retorno siempre estará en R31 por lo que JR R31 = RET
- Es importante cuidar no sobre escribir R31 o se perderá la dirección de retorno

Convenciones

Ante la cantidad de registros y consideraciones se establecieron convenciones sobre su nombramiento:

-$zero = R0 : Siempre tiene el valor 0
-$ra = r31 : Dirección de retorno de la subrutina **(salvado)**
-$vo-$v1 = r2-r3 : Valores de retorno de la subrutina
-$a0-$a3 = r4-r7 : Argumentos pasados a la subrutina
-$t0-$t9 = r8-r15 y r24-r25 : Registros temporarios
-**$s0-$s7 = r16-r23** : Registros que deben ser salvados **(salvados)**
-$sp = r29 : Stack pointer, tope de la pila **(salvado)**
-$fp = r30 : Frame pointer, puntero de la pila **(salvado)**
-$at = r1 : Assembler Temporary, reservado para el ensamblador
-$k0-$k1 = r26-r27 : Kernel del sistema operativo (recursos y tiempo de procesamiento del núcleo)
-$gp = r28 : Global pointer, puntero a zona de memoria estática **(salvado)**

Salvados: aquellos que en caso de usarse deben ser salvados (en la subrutina, no antes), el resto pueden sobreescribirse sin problemas

Como tal no existe una pila, por convención se utiliza $sp. En MSX88 el SP se iba moviendo de a 2 bytes, en MIPS es de a 8 bytes porque cada celda ocupa 8 bytes
El equivalente a **PUSH** es SD. Ejemplo:

PUSH $t1 ----> daddi $sp, $sp, -8 (desplazamiento de 8 bytes en la "pila")
sd $t1,0 ($sp) ($t1 se va a guardar en la dirección 0 + el valor de $sp)

POP $t1 ----> ld $t1,0($sp) (en $t1 se va a cargar el contenido de la dirección 0 + el valor de $sp)
daddi $sp, $sp, 8 (se ajusta el SP de la "pila")

La pila siempre se inicializa a mano DADDI $sp, $zero, 0x400

1) Simular el siguiente programa de suma de números en punto flotante y analizar minuciosamente la ejecución paso a paso. Inhabilitar Delay Slot y mantener habilitado Forwarding.

```
.data
n1: .double 9.13
n2: .double 6.58
res1: .double 0.0
res2: .double 0.0
.code
l.d f1, n1(r0)
l.d f2, n2(r0)
add.d f3, f2, f1
mul.d f4, f2, f1
s.d f3, res1(r0)
s.d f4, res2(r0)
halt
```

a) Tomar nota de la cantidad de ciclos, instrucciones y CPI luego de la ejecución del programa.

16 ciclos, 7 instrucciones, 2.286 CPI

b) ¿Cuántos atascos por dependencia de datos se generan? Observar en cada caso cuál es el dato en conflicto y las instrucciones involucradas.

4. 1 cuando empieza la suma entre f2 y f1, pero aún no se ha terminado de escribir f2, 2 cuando se ejecuta escribir el resultado de la suma en f3, pero la suma aún no ha terminado de ejecutarse. 1 más cuando el resultado de la multiplicación aún no ha sido escrito y la instrucción quiere escribir este resultado en f4

c) ¿Por qué se producen los atascos estructurales? Observar cuales son las instrucciones que los generan y en qué etapas del pipeline aparecen.

Cuando termina la suma y llega a la fase MEM, ocurre a la vez que se carga el resultado de la suma en f3, ambos intentan entrar a la vez a MEM, ocurre lo mismo entre la multiplicación y la escritura del resultado en el registro f4

d) Modificar el programa agregando la instrucción mul.d f1, f2, f1 entre las instrucciones add.d y mul.d. Repetir la ejecución y observar los resultados. ¿Por qué aparece un atasco tipo WAR?

porque aun no se ha terminado de escribir (fase WB) l.d f2,n2(r0)

e) Explicar por qué colocando un NOP antes de la suma, se soluciona el RAW de la instrucción ADD y como consecuencia se elimina el WAR.

porque da tiempo de que la instrucción antes mencionada llegue a la fase WB antes de que se llame al registro involucrado en esta