PRINCIPIOS DE COMPUTADORES

TUTORÍA ACADÉMICA NÚMERO 3:

- **Operaciones aritméticas en punto flotante.**
- **Llamadas al sistema.**

INSTRUCCIONES EN PUNTO FLOTANTE

- MIPS tiene un coprocesador de punto flotante (coprocesador 1 de la máquina) que opera en precisión simple (32 bits) y doble precisión (64 bits).
- El coprocesador tiene sus propios registros de 32 bits que que van desde el $f0 al $f31. Para operaciones de doble precisión se requerirán dos de estos registros, por eso de forma práctica solo se usan los pares. $f0-$f1 almacenan un doble, $f2-$f3 otro y así hasta $f30-$f31
- Los registros de punto flotante tienen igualmente un convenio que se ha de respetar:

Registro	Uso
$f0-$f2	Valores de retorno en punto flotante a subprogramas
$f4-$f10	Registros temporales. No se preservan entre llamadas
$f12-$f14	Primeros 2 parámetros en punto flotante. No preservados
$f16-$f18	Registros temporales. No se preservan entre llamadas
$f20-$f30	Registros salvados. Se preservan entre llamadas.

NOTA: se muestran los registros flotantes pares (doble precisión). Para simple precisión se podrán usar de forma independiente pares e impares, teniendo el mismo uso de convenio.

INSTRUCCIONES ARITMÉTICAS EN PUNTO FLOTANTE

Ejemplo	Significado	Comentario
add.s $f2,$f4,$f6	$f2=$f4+$f6	Suma PF. Precisión simple
sub.s $f2,$f4,$f6	$f2=$f4-$f6	Resta PF. Precisión simple
mul.s $f2,$f4,$f6	$f2=$f4*$f6	Multiplicación PF. Precisión simple
div.s $f2,$f4,$f6	$f2=$f4/$f6	División PF. Precisión simple
abs.s $f2,$f4	$f2 = \|$f4|	Valor absoluto en simple precisión
neg.s $f2,$f4	$f2 = -$f4	Cambio de signo simple precisión
add.d $f2,$f4,$f6	$f2=$f4+$f6	Suma PF. Precisión doble
sub.d $f2,$f4,$f6	$f2=$f4-$f6	Resta PF. Precisión doble
mul.d $f2,$f4,$f6	$f2=$f4*$f6	Multiplicación PF. Precisión doble
div.d $f2,$f4,$f6	$f2=$f4/$f6	División PF. Precisión doble
abs.d $f2,$f4	$f2 = \|$f4|	Valor absoluto en doble precisión
neg.d $f2,$f4	$f2 = -$f4	Cambio de signo doble precisión

CARGA Y ALMACENAMIENTO EN PUNTO FLOTANTE

Ejemplo	Significado	Comentario
`lwc1 $f0,etiqueta` `l.s $f0,etiqueta`	`$f0=Mem[etiqueta]`	Carga en $f0 (simple precisión) en la dir. especificada
`swc1 $f0,etiqueta` `s.s $f0,etiqueta`	`Mem[etiqueta]=$f0`	Almacena en la dir. especificada el registro $f0(simple precisión)
`ldc1 $f0,etiqueta` `l.d $f0,etiqueta`	`$f0=Mem[etiqueta]`	Carga en $f0 (doble precisión) en la dir. especificada
`sdc1 $f0,etiqueta` `s.d $f0,etiqueta`	`Mem[etiqueta]=$f0`	Almacena en la dir. especificada el registro $f0(doble precisión)
`li.s $f0,3.4`	`$f0=3.4`	Carga inmediata simple precisión
`li.d $f0,3.4`	`$f0=3.4`	Carga inmediata doble precisión
`mov.s $f4,$f6`	`$f4 = $f6`	Mueve datos entre registros punto flotante de precisión simple
`mov.d $f4,$f6`	`$f4 = $f6`	Mueve datos entre registros punto flotante de precisión doble
`mtc1 $t0,$f0`	`$f0 = $t0`	Es una copia "cruda" no pasa a flotante. Ojo porque el orden de los operandos no es el de convenio.
`mfc1 $t0,$f0`	`$t0 = $f0`	Igual que antes es copia cruda. Ojo con el orden.

NOTA IMPORTANTE: las 2 últimas letras de los nemónicos terminados en `c1` representan *coprocesador1*, por lo tanto el último carácter es un `1` (uno) y no una `l`(ele)

COPIA Y CONVERSIÓN EN PUNTO FLOTANTE

Ejemplo	Significado	Comentario
mtc1 $t0,$f0	$f0 = $t0	Es una copia "cruda" no pasa a flotante. Ojo porque el orden de los operandos no es el de convenio.
mfc1 $t0,$f0	$t0 = $f0	Igual que antes es copia cruda. Ojo con el orden.
cvt.s.w $f2,$f0	$f2 = $f0	Supone que $f0 tiene la copia cruda de un entero (no formato IEEE754) y lo almacena en $f2 en formato IEEE754 simple precisión
cvt.s.d $f2,$f0	$f2 = $f0	Supone que $f0 está en IEEE754 doble precisión y lo almacena en $f2 en simple precisión.
cvt.d.w $f2,$f0	$f2 = $f0	Supone que $f0 tiene la copia cruda de un entero (no formato IEEE754) y lo almacena en $f2 en formato IEEE754 doble precisión
cvt.d.s $f2,$f0	$f2 = $f0	Supone que $f0 está en IEEE754 simple precisión y lo almacena en $f2 en doble precisión.

Ejemplo (mira los registros en el QTSpim:)

```
li      $t0,32          # $t0 = 32
mtc1    $t0,$f2         # $f2 = 32
cvt.s.w $f4,$f2  # $f4 = 32.0
```

PUNTO FLOTANTE. COMPARACIÓN Y SALTO

Ejemplo	Significado	Comentario
c.eq.s $f2,$f4 (eq,lt,le)	If ($f2==$f4) cond=1 else cond=0	Comparación en precisión simple. Pone cond a 1 o a 0.
c.eq.d $f2,$f4 (eq,lt,le)	If ($f2==$f4) cond=1 else cond=0	Comparación en precisión doble. Pone cond a 1 o a 0.
bclt etiqueta	If (cond==1) go to etiqueta	Salto condicional si cond es cierto
bclf etiqueta	If (cond==0) go to etiqueta	Salto condicional si cond es falso

- Ejemplo:

```
        c.eq.s $f2,$f4 # salta a la etiqueta exit si $f2!=$f4
        bc1f   exit
        ...
        ...
exit:
```

ENTRADA Y SALIDA. LLAMADAS AL SISTEMA

- El simulador nos permite leer y escribir valores de entrada y salida en la ventana de la consola y hacer una salida correcta del programa.
- Usa para ello la llamada al sistema operativo **syscall**
- Se le pasa a la llamada syscall valores en los registros $v0 y adicionalmente en $a0 y $a1.
- Si syscall tiene valor de salida quedará registrado en $v0

ENTRADA Y SALIDA. LLAMADAS AL SISTEMA

Servicio	$v0	Argumentos	Resultados
print_int	1	$a0 = entero a imprimir	
print_float	2	$f12 = flotante a imprimir	
print_double	3	$f12-13 = double a imprimir	
print_string	4	$a0 = direccion de memoria de la cadena a imprimir	
print_char	11	$a0 = carácter a imprimir	
read_int	5		$v0 = entero que se leyó
read_float	6		$f0 = flotante que se leyó
read_double	7		$f0-$f1 = double que se leyó
read_string	8	$a0 = dirección memoria donde se almacenará la cadena a leer (*buffer*) $a1 = número máxio de caracteres admisibles (tamaño del *buffer*)	$v0 = dirección de memoria que contendrá los bytes
read_char	12		$v0 = carácter leído
exit	10		

ENTRADA Y SALIDA. LLAMADAS AL SISTEMA

- La función `print_string` muestra los caracteres contenidos a partir de la dirección indicada en `$a0` hasta encontrar un carácter nulo.
  - Con la directiva `.asciiz` se crea una cadena terminada en un nulo
- Las funciones de lectura `read_int`, `read_float`, y `read_double` siempre esperan hasta que se introduzca un retorno de carro en la entrada.
- La función `read_string` (`$a0` contiene la dirección del buffer y `$a1` contiene `n` el número de caracteres) se comporta de la siguiente manera:
  - Si introducimos menos de `n-1` caracteres y pulsamos al retorno de carro, termina de leer y añade un newline (retorno de carro) seguido de un carácter nulo.
  - Si introducimos `n-1` caracteres, dejará de leer y añadirá únicamente un carácter nulo.
- La función `exit` hace un que un programa termine de forma inmediata.
- La función `read_char` devuelve el cracter en cuanto se pulsa.

ENTRADA Y SALIDA. EJEMPLO.

```
# Programa que hace la suma de dos enteros introducidos
        .data   # Seccion declaracion de datos
cadpet:         .asciiz "Introduza un entero: "
cadsuma:        .asciiz "La suma es: "

        .text   # Seccion de codigo de usuario
main:
        # Se imprime mensaje para introducir primer entero
        li      $v0,4           # $v0=4 funcion print string
        la      $a0,cadpet      # $a0= direccion de la cadena a imprimir
        syscall
        # leo el primer entero y lo almaceno en $s0
        li      $v0,5           # $v0=5 funcion leer un entero.
        syscall                 # el entero leido se queda en $v0
        move    $s0,$v0         # almaceno en $s0 el primer entero leido.
        # Se imprime mensaje para introducir segundo entero
        li      $v0,4           # $v0=4 funcion print string
        la      $a0,cadpet      # $a0= direccion de la cadena a imprimir
        syscall
        # leo el segundo entero y lo almaceno en $s1
        li      $v0,5           # $v0=5 funcion print string
        syscall                 # el entero leido se queda en $v0
        move    $s1,$v0         #almaceno en $s1 el segundo entero leido.

        add     $s3,$s0,$s1     #almaceno en $s3 la suma de los dos enteros
        # Se imprime cadena resultado, el entero suma y un salto de línea
        li      $v0,4           # $v0=4 funcion print_string
        la      $a0,cadsuma     # $a0= direccion de la cadena a imprimir
        syscall
        li      $v0,1           # $v0=4 funcion print_int
        move    $a0,$s3         # $a0=entero a imprimir
        syscall
        li      $v0,11          # $v0=11 funcion print_char
        li      $a0,'\n'        # $a0=caracter a imprimir → salto de línea
        syscall
        # se hace una salida limpia del sistema (exit es codigo 10)
        li      $v0,10
        syscall
```

EJEMPLO. CONVERSIÓN DE GRADOS FAHRENHEIT A CELSIUS

```
# Programa para convertir de grados Fahrenheit a Celsius
# Usa la fórmula C = (F - 32)/1.8
        .data
str_pideF:      .asciiz "Introduzca grados Fahrenheit (entero): "
str_celsius:    .asciiz "Temperatura en grados Celsius: "
factor:         .float  1.8
        .text
main:
        # Sacamos cadena de petición de Fahrenheit
        li      $v0,4
        la      $a0,str_pideF
        syscall
        # Esperamos a que se introduzca un entero
        li      $v0,5
        syscall
        move    $s0,$v0         # Valor recibido a registro salvado $s0
        # Pasamos entero a float
        mtc1    $s0,$f20        # $f20 = $s0
        cvt.s.w $f22,$f20       # $f22 = $f20
        # Hacemos la operación
        li.s    $f4,32.0        # $f4 = 32.0
        sub.s   $f24,$f22,$f4   # $f24 = $f22 - $f4
        l.s     $f4,factor      # $f4 = 1.8
        div.s   $f24,$f24,$f4   # $f24 = $f24 / $f4
        # Mostramos resultado con mensaje previo
        li      $v0,4
        la      $a0,str_celsius
        syscall
        # Imprimimos el resultado
        li      $v0,2
        mov.s   $f12,$f24
        syscall
        # Terminamos el programa
        li      $v0,10
        syscall
```