Ponto Flutuante no MIPS - Resumo

O MIPS tem 32 registradores de precisão simples (32 bits) para manipular números em ponto flutuante – Registradores nomeados $f0 – $f31

$f0 não é especial e diferentemente de $0 pode representar outros valores e não somente zero.

Instruções de precisão simples para números em ponto flutuante podem carregar, armazenar, operar e realizar outras operações sobre esses registradores.

O MIPS tem também hardware para operações de dupla precisão (64 bits) com números em ponto flutuante – Ele utiliza para isso pares de registradores de precisão simples, existindo 16 pares denominados $f0, $f2, — $f30 – Somente o registrador par deve ser especificado na instrução.

Armazenar números em ponto flutuante

- O armazenamento de um valor de precisão simples é feito com uma pseudo instrução: l.s fd,addr

- A instrução carrega os 32 bits de dados armazenados no endereço addr no registrador de ponto flutuante $fd (onde d pode ser 0, 1, 2, ..., 15) – Não se analisa se os bits definem ou não um número em ponto flutuante – O erro será descoberto quando se tentar usar o número em uma operação.

Carregar números em ponto flutuante

- O carregamento de um valor de precisão simples é feito também com uma pseudo instrução: s.s fd,addr

- A instrução armazena o conteúdo do registrador fd no endereço addr

Em ambos os casos o endereço addr pode ser um endereço simbólico simples ou um endereço indexado.

Carregamento imediato

- Existe uma instrução para o carregamento de uma constante em ponto flutuante em um registrador – A pseudo instruções que realiza esta operação corresponde a diversas instruções de máquina

- Pseudo instrução: **li.s fd, val** – o valor val é carregado no registrador fd – val deve obrigatoriamente ser um valor em ponto flutuante

Exemplo de código:

```
li.s $f1, 1.0          # $f1 = 1.0
li.s $f2, 2.0          # $f2 = 2.0
li.s $f10, 1.0e-5      # $f10 = 0.00001
```

Movimentação de números

Existe uma instrução para movimentar valores entre registradores

```
mov.s $f12, $f0 # $f12 = $f0
```

Operações com números em ponto flutuante

Todas as operações apresentadas na tabela correspondem a instruções de máquina – As versões para números de dupla precisão são obtidas substituindo o “s” por um “d”, ou seja, add.s torna-se add.d para umasoma com números com precisão dupla.

Se o valor em um registrador for ilegal ou se uma operação ilegal (divisão por zero) for realizada uma exceção será lançada – O padrão IEEE 752 descreve o que ocorre nestas situações.

Instrução	Operação
abs.s fd,fs	$fd = \|$fs|
add.s fd,fs,ft	$fd = $fs + $ft
sub.s fd,fs,ft	$fd = $fs - $ft
mul.s fd,fs,ft	$fd = $fs * $ft
div.s fd,fs,ft	$fd = $fs / $ft
neg.s fd,fs	$fd = -$fs

Exemplos

```
## Exemplo1 – Muda os valores em valA e valB
.text
.globl  inicio

inicio:
l.s $f0, valA       # $f0 <-- valA
l.s $f1, valB       # $f1 <-- valB
s.s $f0, valB       # $f0 --> valB
s.s $f1, valA       # $f1 --> valA
li $v0, 10          # código 10 == saída
syscall             # Retorna ao SO.

.data
valA: .float 8.32           # FP de 32
valB: .float -0.6234e4      # FP de 32
## Fim do arquivo
```

```
# Converter Temp. F em C

.text
.globl inicio
inicio:
l.s $f12, fahr          # $f12 = 18.0
jal f2c                 # Chamada da subrotina
li  $v0, 10              # Serviço 10 : Exit
syscall

f2c:
l.s $f16, const5                #$f16 = 5.0
l.s $f18, const9                #$f18 = 9.0
div.s $f16, $f16, $f18          #$f16 = $f16/$f18
l.s $f18, const32                #$f18 = 32.0
sub.s $f18, $f12, $f18           #$f18 = $f12-$f18
mul.s $f0, $f16, $f18           #$f0 = $f16*$f18
jr $ra

.data
const5:  .float    5.0
const9:  .float    9.0
const32: .float    32.0
fahr:    .float    18.0
```

```
# exe9.s - Conversão Fahrenheit-Celsius
.text
.globl __start
__start:
la $a0, questao # Carrega string em $a0
li $v0, 4       # Serviço 4 : imprime string
syscall
li $v0, 6          # Serviço 6 : $f0<--float
syscall
mov.s $f12, $f0     # $f12 = valor digitado
jal f2c           # Chama a subrotina
la $a0, resp        # Carrega string em $a0
li $v0, 4       # Serviço 4 : imprime string
syscall
li $v0, 2       # Serviço 2 : $f12 == float
syscall        #    Apresenta o valor em $f12
li  $v0, 10     # Serviço 10 : Exit
syscall
```

```
f2c:
l.s $f16, const5                    # $f16 = 5.0
l.s $f18, const9                    # $f18 = 9.0
div.s $f16, $f16, $f18   # $f16 = $f16/$f18
l.s $f18, const32                   # $f18 = 32.0
sub.s $f18, $f12, $f18    # $f18 = $f12-$f18
mul.s $f12, $f16, $f18    # $f12 = $f16*$f18
jr $ra                # retorna

#-- Dados -------------------------
.data
const5:   .float 5.0
const9:   .float 9.0
const32:  .float 32.0
questao:  .asciiz "Qual a temperatura
Fahrenheit: "
resp:     .asciiz "\nA temperatura Celsius é: "
```

```
# Exemplos de condições de teste igual, menor ou igual e menor
# Se a condição for verdadeira, seta (faz igual a 1) o Condition Flag 0
# A instrução bclt desvia se o Condition Flag 0 está setado (ou é igual a 1)

.text
.globl main
main:

l.s $f0, const1        # $f0 = 1.0, ou $f0 = 1.0
l.s $f1, const2        # $f0 = 1.0, ou $f0 = 2.0

c.eq.s $f0, $f1 #  set Condition flag 0 if $f0 == $f1
bc1t igual

c.le.s $f0, $f1 #  set Condition flag 0 if $f0 <= $f1
bc1t menorigual

c.lt.s $f0, $f1 #  set Condition flag 0 if $f0 < $f1
bc1t menor

j fim

menorigual:
add.s $f2, $f0,$f1
j fim

igual:
add.s $f2, $f0,$f1
j fim

menor:
add.s $f2, $f0,$f1
j fim

fim:
li $v0 10
syscall

.data
const1:   .float 1.0
const2:   .float 2.0
const3:  .float 3.0
```