VÍRGULA FLUTUANTE NO MIPS (*Floating Point*)

Prof. Dr. Luciano José Senger

Os projetistas da arquitetura MIPS definiram dois coprocessadores que são empregados em conjunto com a unidade central de processamento 1. O coprocessador 0 é empregado para o controle de exceções, e o coprocessador 1 é dedicado ao processamento de operações em vírgula flutuante (`floating point instructions`).

Figura 1: Arquitetura MIPS e seus co processadores

A Figura 2 apresenta o panorama geral para a utilização de vírgula flutuante no MIPS. Como outras arquiteturas, o MIPS admite os formatos de precisão simples e precisão dupla, de acordo com o padrão IEEE 754 (**????**). Para a implementação das operações em vírgula flutuante, os projetistas da arquitetura MIPS decidiram acrescentar 32 registradores de vírgula flutuante separados, chamados pelos mnemônicos de `$f0 a $f31`. Tais registradores tem 32 bits de tamanho cada e são usados para precisão simples e precisão dupla. Em precisão dupla, como são necessários 64 bits (tipo `double` da linguagem C), pares de registradores devem ser adotados, sempre começando com um registrador de índice par (por exemplo `$f0, $f2, $f4, ...`). Como a unidade de vírgula flutuante tem acesso a memória, os projetistas MIPS definiram instruções de `load/store` adicionais para movimentar os dados para os registradores de vírgula flutuante, chamadas de `lwc1` e `swc1`.

`pictures/pf.jpg`

Figura 2: Panorama geral de utilização de operações em PF no MIPS. Há instruções de transferência de dados entre os bancos de registradores da UCP e da unidade de PF (mfc1, mtc1). Há também instruções de lwc1 e swc1 para tranferência de dados entre a unidade de PF e a memória.

Um exemplo de código em *assembly* para o MIPS, para carregar dois números de precisão simples da memória, somá-los e depois armazenar o resultado na memória é apresentado na Listagem 1.

Listagem 1: Programa em MIPS que realiza a soma de dois valores em precisão simples

```
.data
        x : .float 3.14
        y:  .float 0.5
        z:  .float 0.0
.text
.globl main
main:
        la $s0, x
        la $s1, y
        la $s2, z
        lwc1    $f0, 0($s0)
        lwc1    $f1, 0($s1)
        add.s   $f12, $f0, $f1
        swc1    $f12, 0($s2)

        li $v0, 2
        syscall
        li $v0,10
        syscall
```

Como exemplo, a instrução:

```
add.d $f0, $f2, $f4
```

realiza uma operação de soma considerando que há dois valores em em precisão dupla armazenados em `$f2` e `$f4`.

Outros exemplos:

- `sub.s` e `sub.d` para subtração
- `mul.s` e `mul.d` para multiplicação
- `div.s` e `div.d` para divisão

As instruções `mov.s` e `mov.d` copiam dados entre os registradores de vírgula flutuante. Para transferir dados entre o banco de registradores inteiros e o banco de registradores em vírgula flutuante as instruções `mtc1` e `mfc1` são empregadas:

```
mtc1 $t0, $f0 # $f0 = $t0
mfc1 $t0, $f0 # $t0 = $f0
```

Exemplos de `load/store`:

```
lwc1 $f2, 0($a0) # $f2 = M[$a0]
swc1 $f4, 4($sp) # M[$sp+4] = $f4
```

O nome `c1` na instrução significa `coprocessor 1`.

0.1 Exemplo completo

O programa da Listagem 2 realiza a conversão de temperatura de Fahreinheit para Celsius, conforme a equação 1.

$$celsius = \frac{(fahrenheit - 32.0) \times 5.0}{9.0} \tag{1}$$

Nesse caso, optou-se por carregar as constantes para o banco de registradores de inteiros (linhas 17 a 20) e depois transferir os valores para o banco de registradores do coprocessador 1 por meio das linhas 22 a 25. A conversão dos valores das linhas 27 a 30 é necessária para converter de valores em complemento a 2 (inteiros) para vírgula flutuante. Para isso, a instrução `cvt.s.w` é empregada.

Nas linhas 32 a 35 a equação 1 é calculada. Na linha 35, o valor convertido é armazenado em memória por meio da instrução `swc1`.

Listagem 2: Programa em MIPS que realiza a conversão de Fahreinheit para Celsius

```
.data
fahr : .word 70 # 70 graus em Fahreinheit
constante1: .word 32
constante2: .word 5
constante3: .word 9
celsius: .float 0.0
mensagem: .asciiz "O valor em Celsius e igual a : "
.text
.globl main
main:
la $s0, fahr
la $s1, constante1
la $s2, constante2
la $s3, constante3
la $s4, celsius

lw $t0, 0($s0) # valor a ser convertido vai para $t0
lw $t1, 0($s1) # constante1 vai para $t1
lw $t2, 0($s2) # constante1 vai para $t1
lw $t3, 0($s3) # constante1 vai para $t1

mtc1 $t0, $f12 # transferencia  entre os bancos UCP para C1
mtc1 $t1, $f0  # transferencia  entre os bancos UCP para C1
mtc1 $t2, $f1  # transferencia  entre os bancos UCP para C1
mtc1 $t3, $f2  # transferencia  entre os bancos UCP para C1

cvt.s.w $f0, $f0
cvt.s.w $f1, $f1
cvt.s.w $f2, $f2
cvt.s.w $f12, $f12

sub.s $f0, $f12, $f0  # fahrenheit - 32
mul.s $f0, $f0, $f1
div.s $f12, $f0, $f2
swc1 $f12, 0($s4)
```

```
li $v0, 4
la $a0, mensagem
syscall

li $v0, 2
syscall

li $v0,10
syscall
# O valor em Celsius e igual a : 21.11111
```

A Figura 3 detalha o processo de conversão (*casting*) em vírgula flutuante, usado no programa nas linhas 27 a 30, por meio da instrução `cvt.s.w`.

Figura 3: Realização de conversões (*casting*) entre *words* e números em vírgula flutuante

1 Detalhamento

Uma questão importante que os projetistas de computadores enfrentam no suporte à aritmética computacional de vírgula flutuante é se devem usar os mesmo registradores empregados pelas instruções com inteiros ou acrescentar um conjunto especial de vírgula flutuante. Como os programas realizam operações com inteiros e operações com vírgula flutuante sobre dados diferentes, a separação dos registradores o aumentará ligeiramente o número de instruções necessárias para executar um programa. O maior impacto é criar um conjunto separado de instruções de transferência de dados para mover dados entre os registradores de vírgula flutuante e memória. Os benefícios dos registradores de vírgula flutuante separados são:

- existência do dobro dos registradores sem utilizar mais bits no formato de instrução;
- dobro de largura de banda de registradores, com registradores separados para inteiros e números em vírgula flutuante.

A Tabela 1 apresenta as instruções MIPS para vírgula flutuante.

Figura 4: Parthenon e a proporção de ouro

Infelizmente, o MARS não apresenta as informações atualizadas ao longo da simulação dos registradores do coprocessador (Coproc0) de tratamento de exceções.

Tabela 1: Instruções de vírgula flutuante

Instrução	**Exemplo**	**Significado**	**Comentário**
Soma	add.s $f2,$f4,$f6	$f2 = $f4 + $f6	Floating-Point add (single precision)
Subtração	sub.s $f2,$f4,$f6	$f2 = $f4 - $f6	Floating-Point sub (single precision)
Multiplicação	mul.s $f2,$f4,$f6	$f2 = $f4 * $f6	Floating-Point multiply (single precision)
Divisão	div.s $f2,$f4,$f6	$f2 = $f4 / $f6	Floating-Point divide (single precision)
Soma	add.d $f2,$f4,$f6	$f2 = $f4 + $f6	Floating-Point add (double precision)
Subtração	sub.d $f2,$f4,$f6	$f2 = $f4 - $f6	Floating-Point sub (double precision)
Multiplicação	mul.d $f2,$f4,$f6	$f2 = $f4 * $f6	Floating-Point multiply (double precision)
Divisão	div.d $f2,$f4,$f6	$f2 = $f4 / $f6	Floating-Point divide (double precision)
Load	lwc1 $f1,100($2)	$f1 = Memory[$2+100]	32-bit data to FP register
Store	swc1 $f1,100($2)	Memory[$2+100] = $f1	32-bit data to memory
Desvio (se verdade)	bc1t 100	if (cond == 1) go to PC+4+100	PC relative branch if FP condition
Desvio (se falso)	bc1f 100	if (cond == 0) go to PC+4+100	PC relative branch if not condition
Comparação (eq,ne,lt,le,gt,ge)	c.lt.s $f2,$f4	if ($f2 < $f4) cond=1; else cond=0	Floating-point compare less than single precision
Comparação (eq,ne,lt,le,gt,ge)	c.lt.d $f2,$f4	if ($f2 < $f4) cond=1; else cond=0	Floating-point compare less than double precision

Listagem 3: Exemplo de programa para o x86 e uso de syscalls em um sistema Linux

```
bits 64

section .text

global assembly
assembly:
        mov rax, 60
        mov rdi, 0
        syscall
        ret
```

2 Exercícios

Considerando os itens a seguir:

(1) Registrador `$v0`

(2) `syscall`

(3) Registrador `$a0`

(4) Coprocessador 0

(5) Kernel

(6) serviço `print_string`

(7) serviço `read_int`

preencha as lacunas abaixo, conforme o item mais adequado:

1. ( ) Núcleo do sistema operacional, responsável por gerenciar os dispositivos de entrada e saída, como mouse, teclado e tela
2. ( ) Por meio deste registrador, é possível passar um argumento para a chamada de sistema
3. ( ) Parte principal do suporte em hardware para controle de exceções
4. ( ) Instrução MIPS
5. ( ) Selecionado por meio da atribuição do valor igual a 4 no registrador `$v0`
6. ( ) Selecionado por meio da atribuição do valor igual a 5 em `$v0` e devolve o resultado da chamada por meio do mesmo registrador `$v0`
7. ( ) O número da função/serviço do sistema operacional é especificado por meio deste registrador