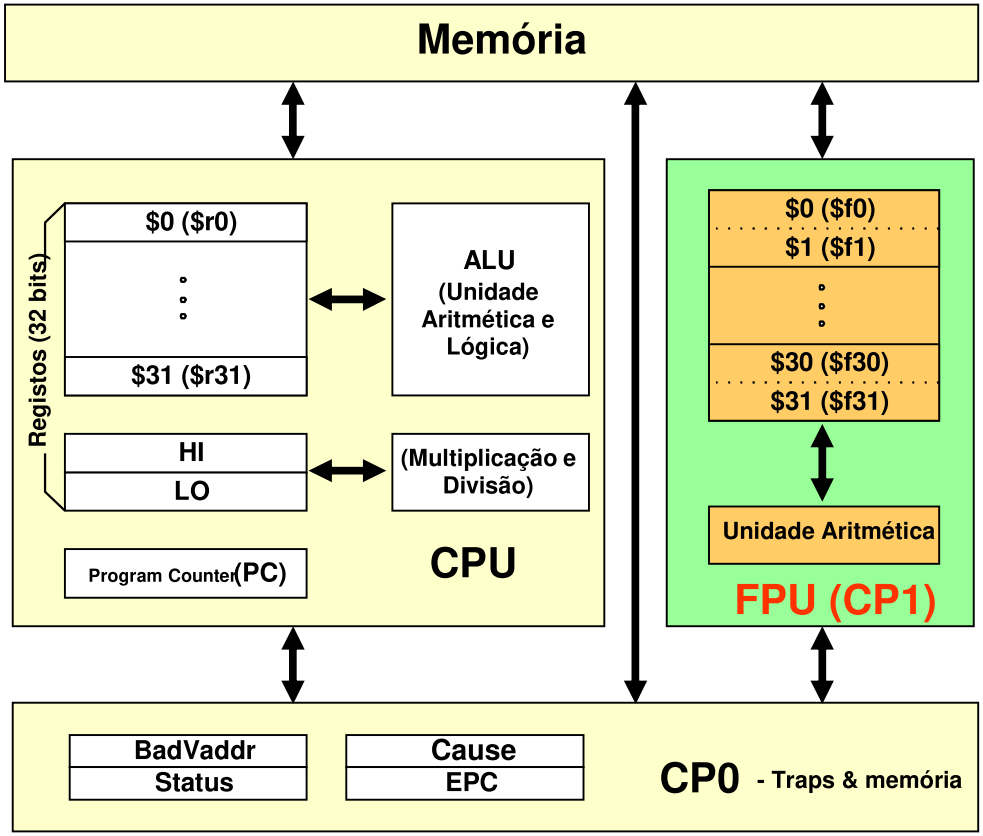
Cálculo em Vírgula Flutuante no MIPS



FPU: Floating Point Unit

- O MIPS inclui um coprocessador aritmético (Coprocessador 1) capaz de efetuar operações aritméticas em vírgula flutuante, usando a norma IEEE 754
- Esse coprocessador tem o seu próprio espaço de armazenamento composto por um conjunto de 32 registos de 32 bits cada, e o seu próprio conjunto de instruções (ISA)

Vírgula Flutuante no MIPS – registos

- Os registos do coprocessador 1 são designados por **$fn**, em que o indíce **n** toma valores entre 0 e 31 ($f0, $f1, $f2, …)
- Cada par de registos consecutivos [**$fn,$fn+1**] (**com n par**) pode funcionar como um registo de 64 bits para armazenar valores em **precisão dupla**.
- A referência ao conjunto de 2 registos faz-se sempre indicando como operando o **registo par** ($f0, $f2, $f4,…)
- **Apenas os registos de índice par** podem ser usados no contexto das instruções

Vírgula Flutuante no MIPS – instruções aritméticas

```
abs.p   FPdst,FPsrc           #Absolute Value
neg.p   FPdst,FPsrc           #Negate
div.p   FPdst,FPsrc1,FPsrc2   #Divide
mul.p   FPdst,FPsrc1,FPsrc2   #Multiply
add.p   FPdst,FPsrc1,FPsrc2   #Addition
sub.p   FPdst,FPsrc1,FPsrc2   #Subtract
```

- O sufixo **.p** representa a precisão com que é efetuada a operação (simples ou dupla); na instrução é substituído pelas letras **.s** ou **.d** respetivamente
- Exemplos:
  - `add.s $f0,$f4,$f6 #$f0=$f4 + $f6`
  - `div.d $f4,$f0,$f8 #$f4($f5)=$f0($f1) / $f8($f9)`

Vírgula Flutuante no MIPS – conversão entre tipos

```
cvt.d.s FPdst,FPsrc   #Convert Float to Double
cvt.d.w FPdst,FPsrc   #Convert Integer to Double
cvt.s.d FPdst,FPsrc   #Convert Double to Float
cvt.s.w FPdst,FPsrc   #Convert Integer to Float
cvt.w.d FPdst,FPsrc   #Convert Double to Integer
cvt.w.s FPdst,FPsrc   #Convert Float to Integer
```

- A letra mais à direita especifica o formato original; a letra do meio, especifica o formato do resultado - **s**: float (single), **d**: double, **w**: inteiro
- As **conversões** entre tipos de representação são efetuadas pela FPU: **os registos operando e destino das instruções são obrigatoriamente registos da FPU**

Conversão entre tipos – exemplos

```
$f0=0xC0D00000 = 11000000110100000000000000000000
               = 11000000110100000000000000000000
               = -1.625 x 2^2 = -6.5

cvt.d.s $f6,$f0 #Convert Float to Double
     E = (129-127) + 1023 = 1025 = 10000000001_2
     $f6=0x00000000 $f7=1 10000000001 1010000...0
     $f6=0x00000000 $f7=0xC01A0000

cvt.w.s $f8,$f0 #Convert Float to Integer
     Exp = (129-127) = 2
     Val = -1.625 x 2^2 = -6.5
     Resultado: (int)(-6.5) = trunc(-6.5) = -6
     $f8=0xFFFFFFFA  (-6 em complemento para 2)
```

Vírgula Flutuante no MIPS – instruções de transferência

- **Transferência de informação** entre registos do CPU e da FPU, e entre registos da FPU


```
mtc1    CPUSrc,FPdst    #Move to Coprocessor 1
                        #Ex: mtc1 $t0,$f4
mfc1    CPUdst,FPsrc    #Move from Coprocessor 1
                        #Ex: mfc1 $a0,$f6
mov.s   FPdst,FPsrc     #Move from FPsrc to FPdst (single)
                        #Ex: mov.s $f4,$f8
mov.d   FPdst,FPsrc     #Move from FPsrc to FPdst (double)
                        #Ex: mov.d $f2,$f0
```

- Estas instruções copiam o conteúdo integral do registo fonte para o registo destino
- **Não fazem qualquer tipo de conversão entre tipos de informação**

Vírgula Flutuante no MIPS – instruções de transferência

- **Transferência de informação** entre registos da FPU e a memória

Registo da **FPU** | **Endereço de memória**

```
l.s     FPdst,offset(CPUreg)  #Load Float from memory
                              #Ex: l.s $f0,4($a0)
s.s     FPsrc,offset(CPUreg)  #Store Float into memory
                              #Ex: s.s $f0,0($a0)
l.d     FPdst,offset(CPUreg)  #Load Double from memory
                              #Ex: l.d $f4,8($a1)
s.d     FPsrc,offset(CPUreg)  #Store Double into memory
                              #Ex: s.d $f4,16($t0)
```

Instruções nativas (só muda a mnemónica):

```
lwc1    FPdst,offset(CPUreg)  #Load Float from memory
swc1    FPsrc,offset(CPUreg)  #Store Float into memory
ldc1    FPdst,offset(CPUreg)  #Load Double from memory
sdc1    FPsrc,offset(CPUreg)  #Store Double into memory
```

Vírgula Flutuante no MIPS – Manipulação de constantes

- Nas instruções da FPU do MIPS os operandos têm que residir em registos internos, o que significa que **não há suporte para a manipulação direta de constantes**. Como lidar então com operandos que são constantes?
- **Método 1**:
  - Determinar, manualmente, o valor que codifica a constante (32 bits para precisão simples ou 64 bits para precisão dupla)
  - Carregar essa constante em 1 ou 2 registos do CPU e copiar o(s) seu(s) valor(es) para o(s) registo(s) da FPU
- **Método 2**:
  - Usar as directivas "`.float`" ou "`.double`" para definir em memória o valor da constante: 32 bits (`.float`) ou 64 bits (`.double`)
  - Ler o valor da constante da memória para um registo da FPU usando as instruções de acesso à memória (`l.s` ou `l.d`)

Vírgula Flutuante no MIPS – Manipulação de constantes

- O MARS disponibiliza duas instruções virtuais que permitem usar o método 2 (definição da constante em memória) de forma simplificada. Essas instruções têm o seguinte formato:

```
l.s  FPdst,label #Ex: l.s $f0,K1
l.d  FPdst,label #Ex: l.d $f4,K1
```

  em que "`label`" representa o endereço onde a constante está armazenada em memória.

- A decomposição em instruções nativas destas instruções é (admitindo, por exemplo, que K1 corresponde ao endereço `0x10010008`):

```
l.s  $f0,k1
   lui $1,0x1001
   l.s $f0,0x0008($1)
```

```
l.d  $f4,k1
   lui $1,0x1001
   l.d $f4,0x0008($1)
```

Vírgula Flutuante no MIPS – instruções de decisão

- A tomada de decisões envolvendo quantidades em vírgula flutuante realiza-se de forma distinta da utilizada para o mesmo tipo de operação envolvendo quantidades inteiras
- Para quantidades em vírgula flutuante são necessárias duas instruções em sequência: uma **comparação das duas quantidades, seguida da decisão** (que usa a informação produzida pela comparação):
  - A instrução de comparação coloca a **True** ou **False** uma *flag* (1 bit), dependendo de a condição em comparação ser verdadeira ou falsa, respetivamente
  - Em **função do estado dessa *flag*** a instrução de decisão (instrução de salto) pode alterar a sequência de execução

Cálculo em Vírgula Flutuante no MIPS

- **Instruções de comparação**:

```
c.xx.s FPUreg1, FPUreg2   # compare float
c.xx.d FPUreg1, FPUreg2   # compare double
```

Em que **xx** pode ser uma das seguintes condições:

```
EQ - equal
LT - less than
LE - less or equal
```

Exemplos:

```
c.eq.s $f0,$f2 / c.le.d $f4,$f8
```

- **Instruções de salto**:

```
bc1t label   # branch if true
bc1f label   # branch if false
```

Vírgula Flutuante no MIPS – instruções de decisão

```
    float a, b;
    ...

    if( a > b)
        a = a + b;
    else
        a = a - b;
```

```
# a: $f0
# b: $f2
...
if:   c.le.s $f0, $f2          # if(a > b)
      bc1t   else              # {
      add.s  $f0, $f0, $f2     #     a = a + b;
      j      endif             # }
                               # else
else: sub.s  $f0, $f0, $f2     #     a = a - b;
endif:...
```

Convenções de utilização dos registos

- Registos para **passar parâmetros** para sub-rotinas (do tipo float ou double):
  - **$f12** ($f13), **$f14** ($f15), por esta ordem
- Registos para **devolução de resultados** das sub-rotinas:
  - **$f0** ($f1)
- Registos que **podem** ser livremente usados e alterados pelas sub-rotinas ("caller-saved"):
  - **$f0** ($f1) a **$f18** ($f19)
- Registos que **não podem** ser alterados pelas sub-rotinas ("callee-saved"):
  - **$f20** ($f21) a **$f30** ($f31)