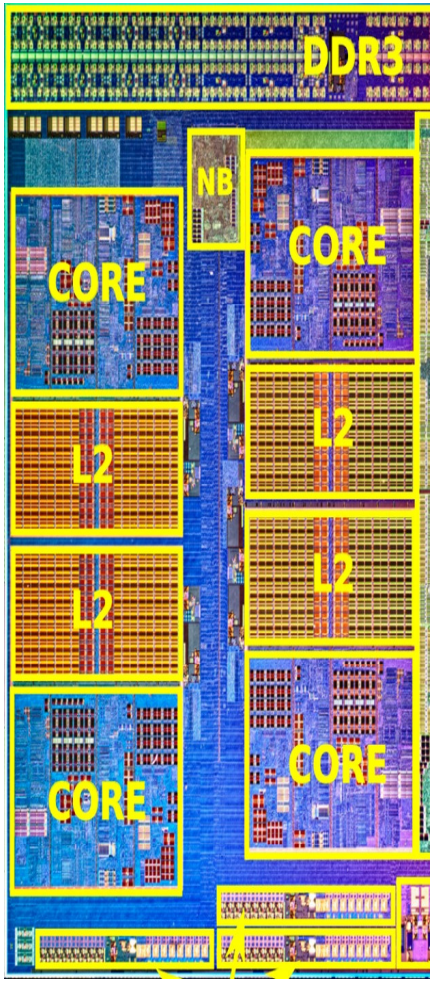
0273359 – Arquitetura e Organização de Computadores 1



Fonte: http://www.techspot.com/article/904-history-of-the-personal-computer-part-5/

Ponto Flutuante

Luciano de Oliveira Neris
luciano@dc.ufscar.br

Adaptado de slides do prof. Marcio Merino Fernandes

Departamento de Computação
Universidade Federal de São Carlos

2

Números em Ponto Flutuante

Números em Ponto Flutuante (FP)

- Linguagens de programação podem usar nros c/ frações
  - Estes são chamados números de ponto flutuante (floating-point numbers)
  - Exemplos:

    3.14159265… (π)

    2.71828… (*e*)

    0.000000001 (1 nanosegundo)

    86,400,000,000,000 (86 trilhões e 400 milhões)

Números em Ponto Flutuante (FP)

- Podemos usar notação científica p/ representar:
  - Números muito pequenos (ex: 1.0×10^{-9} -> 0.000000001)
  - Números muito grandes (ex: 8.64×10^{13} -> 86,400,000,000,000)
  - Notação Científica : $\pm d . f_1 f_2 f_3 f_4 \ldots \times 10^{\pm e_1 e_2 e_3}$

Números em Ponto Flutuante (FP)

- Exemplos em base 10:
  - 5.341×10^{3} , 0.05341×10^{5} , -2.013×10^{-1} , -201.3×10^{-3}

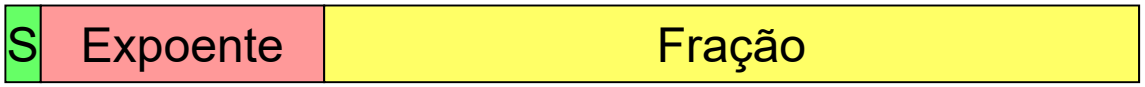
decimal point

- Exemplos em base 2:
  - 1.00101×2^{23} , 0.0100101×2^{25} , -1.101101×2^{-3} , -1101.101×2^{-6}
    - *(expoentes em base 10 p/ facilitar leitura)*
  - $(1101.101)_2 = 2^3+2^2+2^0+2^{-1}+2^{-3} = 13.625$

- Números em FP devem ser normalizados:
  - Exatamente um dígito diferente de zero antes do ponto
  - Ex-Números FP Normalizados: 5.341×10^{3}; -1.101101×2^{-3}
  - Ex-Números FP Não Normalizados 0.05341×10^{5}; 1101.101×2^{-6}

Representation de Números FP

- Um números em FP é representado pela tupla (S, E, F):
  - *S = Bit de Sinal* (0 = positivo, 1 = negativo)
    - Representação chamada de sinal e magnitude
  - E = Expoente
    - Números grandes possuem expoente positivo grande.
    - Números pequenos possuem expoente negativo
    - Quanto maior o número de bits do expoente, mair a faixa de valores representados.
  - *F* = Fração (ou mantissa)
    - Quanto maior o número de bits de fração, maior a precisão do nro.



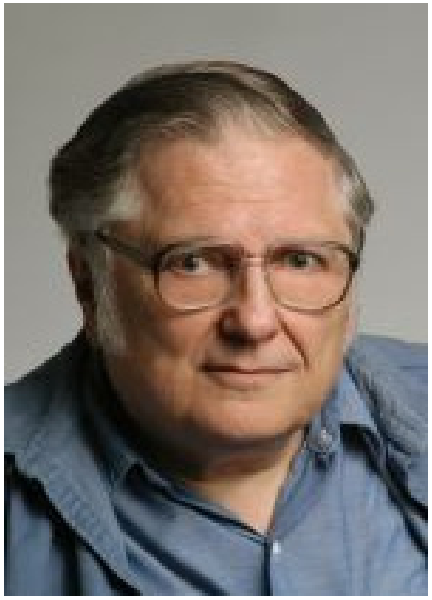
$$\text{Valor do Nro em FP} = (-1)^{S} \times \text{val}(F) \times 2^{\text{val}(E)}$$

FP: Padrão IEEE 754

FP: Padrão IEEE 754

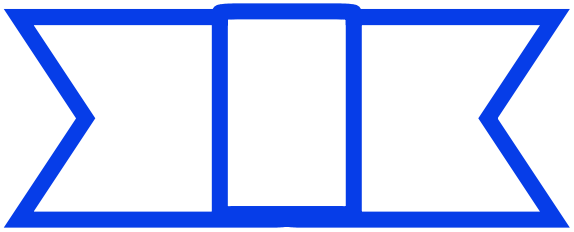
- Usado em praticamente todo computador nos últimos 30 anos
  - Padrão simplifica a portabilidade, desenvolvimento de algoritmos, melhora a precisão no processamento.

FP: Padrão IEEE 754 (1985)



Prof. William Kahan

Criador do Padrão FP IEEE754

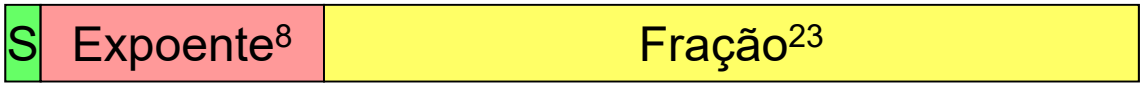


1989
ACM Turing
Award Winner!

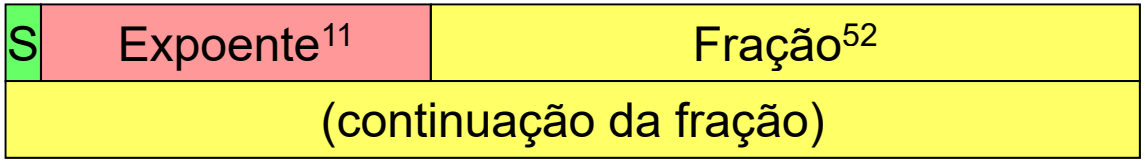
http://www.cs.berkeley.edu/~wkahan/ieee754status/754story.html

FP: Padrão IEEE 754

- Precisão Simples: 32 bits
  - 1-bit sinal + 8-bit expoente + 23-bit fração



- Precisão Dupla: 64 bits
  - 1-bit sinal + 11-bit expoente + 52-bit fração



FP: Padrão IEEE 754

- Para um número normalizado:

- Significando= $(1.F)_2 = (1.f_1f_2f_3f_4\ldots)_2$
  - IEEE 754 assume que implicitamente o "1." (não armazenado)
- Valor de Nro em FP Normalizado:

$$(-1)^S \times (1.F)_2 \times 2^{val(E)}$$

$$(-1)^S \times (1.f_1f_2f_3f_4\ldots)_2 \times 2^{val(E)}$$

$$(-1)^S \times (1 + f_1\times2^{-1} + f_2\times2^{-2} + f_3\times2^{-3} + f_4\times2^{-4}\ldots)_2 \times 2^{val(E)}$$

$(-1)^S = 1$ p/ $S=0$, $(-1)^S = -1$ p/ $S=1$

Expoente: Representação Polarizada (*Biased*)

- Opções p/ se representar o expoente:
  - Sinal + magnitude
  - Complemento de 2
  - IEEE 754: Representação Polarizada

- Valor do expoente = val(E) = E – Bias (Bias = cte)

Expoente: Representação Polarizada (*Biased*)

- Valor do expoente = val(*E*) = *E* – Bias (Bias =cte) constant)
- Expoente= 8 bits p/ precisão simples
  - *E* assume um valor entre 0 e 255
  - *E* = 0 e *E* = 255: reservados p/ uso especial (ver depois…)
  - *E* = 1 a 254: usados p/ números FP normalizados
  - Bias = 127 (metade de 254) → val(*E*) = *E* – 127
  - Ex: val(*E*=1) → Exp= -126
  - Ex: val(*E*=127) → Exp= 0
  - Ex: val(*E*=254) → Exp= = 127

Expoente: Representação Polarizada (*Biased*)

- Expoente= 11 bits p/ precisão dupla
  - E assume um valor entre 0 e 2047
  - E = 0 e E = 2047 reservados p/ uso especial (ver depois...)
  - E = 1 to 2046 usados p/ números FP normalizedos
  - Bias = 1023 (metade de 2046), val(E) = E – 1023
  - val(E=1) = -1022, val(E=1023) = 0, val(E=2046) = 1023

- Valor de Nro em FP Normalizado:

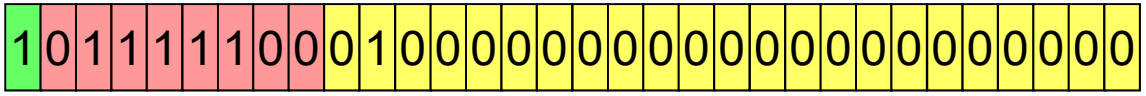
$$(-1)^S \times (1.F)_2 \times 2^{E - \text{Bias}}$$ ☹

$$(-1)^S \times (1.f_1f_2f_3f_4\ldots)_2 \times 2^{E - \text{Bias}}$$

$$(-1)^S \times (1 + f_1\times2^{-1} + f_2\times2^{-2} + f_3\times2^{-3} + f_4\times2^{-4}\ldots)_2 \times 2^{E - \text{Bias}}$$

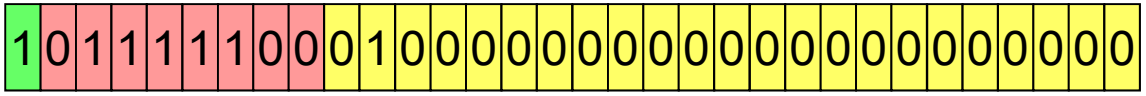
Exemplo 1 – Precisão Simples

- Qual o valor decimal do seguinte nro FP em precisão simples ?



Exemplo 1 – Precisão Simples

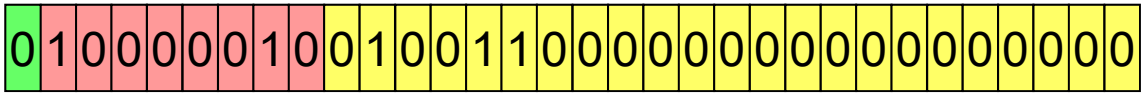
- Qual o valor decimal do seguinte nro FP em precisão simples ?



- Solução:
  - Sinal = 1 → **negativo**
  - Expoente E = $(01111100)_2$ = 124, *E* – bias = 124 – 127 = –3
  - Fração F = $(1.0100 \ldots 0)_2 = 1 + 2^{-2} = 1.25$ (1. é implicito)
  - Valor Decimal= $-1.25 \times 2^{-3} = -0.15625$

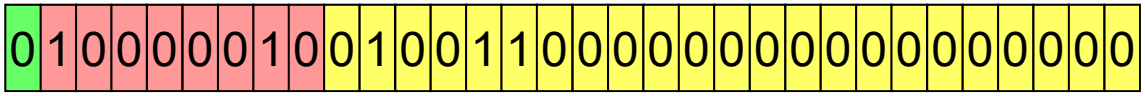
Exemplo 2 – Precisão Simples

- Qual o valor decimal do seguinte nro FP em precisão simples ?



Exemplo 2 – Precisão Simples

- Qual o valor decimal do seguinte nro FP em precisão simples ?

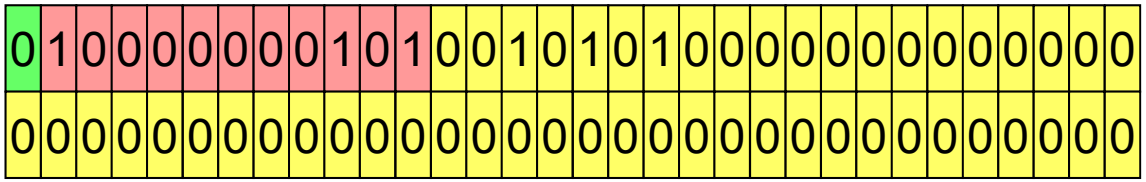


- Solução :

  - Valor Decimal= $+(1.01001100\ \ldots\ 0)_2 \times 2^{130-127} =$
    $(1.01001100\ \ldots\ 0)_2 \times 2^3 = (1010.01100\ \ldots\ 0)_2 =$ 10.375

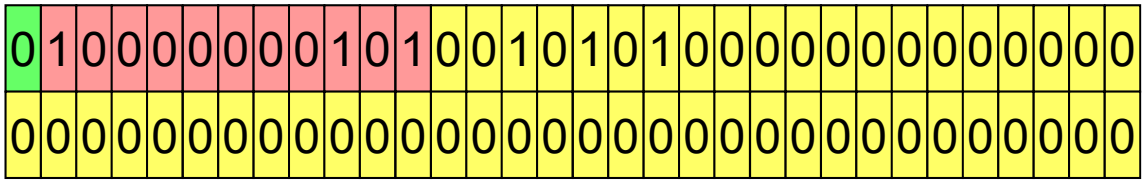
Exemplo 3 – Precisão Dupla

- Qual o valor decimal do seguinte nro FP em precisão dupla?



Exemplo 3 – Precisão Dupla

- Qual o valor decimal do seguinte nro FP em precisão dupla?



- Solução:
  - Expoente = $(10000000101)_2$ – Bias = 1029 – 1023 = 6
  - Valor Decimal = $(1.00101010 \ldots 0)_2 \times 2^6$ = $(1001010.10 \ldots 0)_2$ = 74.5

Caminho Inverso: Converter Decimal p/ FP

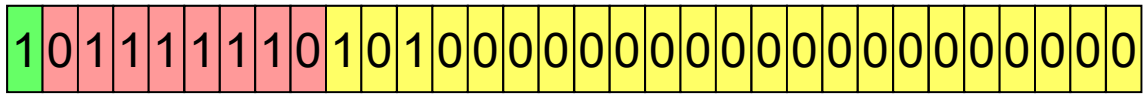
- Exemplo: Representar –0.8125 em FP:

Caminho Inverso: Converter Decimal p/ FP

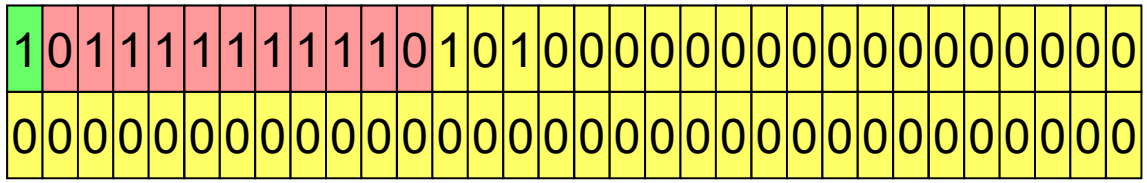
- Exemplo: Representar –0.8125 em FP:
- Solução:
  - Bits da parte fracionária obtidos via multiplicalçao por 2:
    - 0.8125 × 2 = 1.625
    - 0.625 × 2 = 1.25
    - 0.25 × 2 = 0.5
    - 0.5 × 2 = 1.0
      - Parar quando parte fracionária= 0

$0.8125 = (0.1101)_2 = ½ + ¼ + 1/16 = 13/16$

  - Fração = $(0.1101)_2 = (1.101)_2 \times 2^{-1}$ (Normalizado)
  - Expoente = –1 + Bias = 126 (precisão simple) and 1022 (dupla)



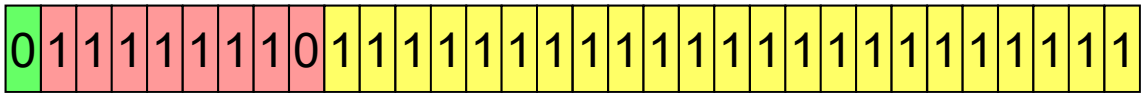
Precisão Simples



Precisão Dupla

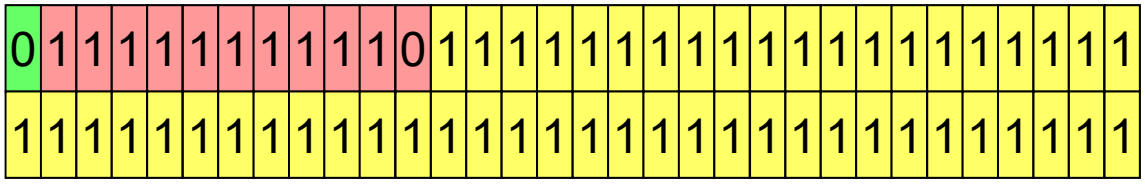
MAIOR Nro FP Normalizado

- Precisão Simples (PS):



  - Expoente – bias = 254 – 127 = 127 (maior expoente p/ PS)
  - Fração = $(1.111 \ldots 1)_2 = \sim 2$
  - Valor decimal ≈ $2 \times 2^{127} \approx 2^{128} \approx 3.4028 \ldots \times 10^{38}$

- Precisão Dupla (PD):

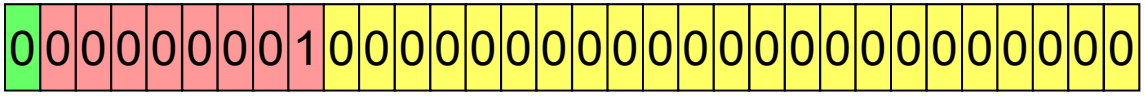


  - Valor decimal ≈ $2 \times 2^{1023} \approx 2^{1024} \approx 1.79769 \ldots \times 10^{308}$

- Overflow: expoente é muito grande para ser armazenado no campo de expoente.

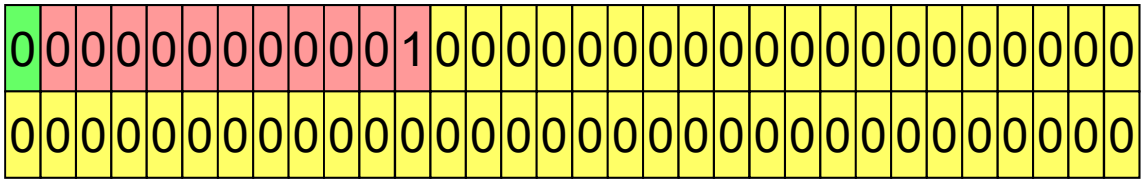
MENOR Nro FP Normalizado

- Precisão Simples (PS):



  - Expoente – bias = 1 – 127 = –126 (menor expoente p/ PS)
  - Fração = $(1.000 \ldots 0)_2 = 1$
  - Valor decimal = $1 \times 2^{-126} = 1.17549 \ldots \times 10^{-38}$

- Precisão Dupla (PD):



  - Valor decimal = $1 \times 2^{-1022} = 2.22507 \ldots \times 10^{-308}$

- Underflow: expoente é muito pequeno para ser armazenado no campo de expoente.

Zero, Infinito, NaN

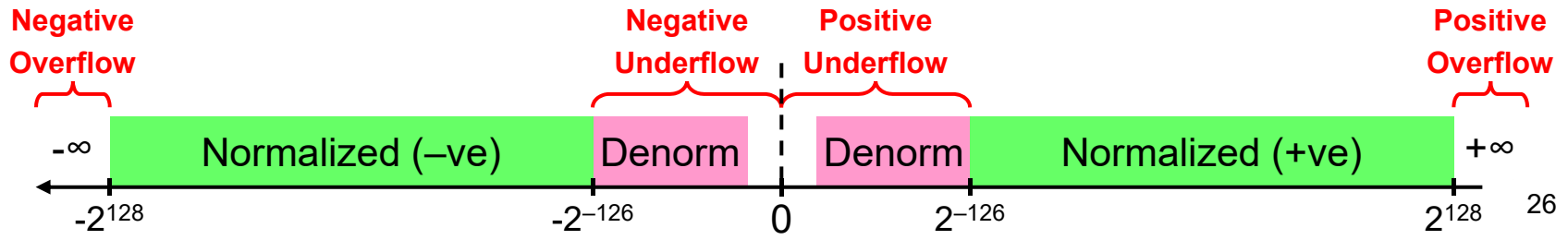
- Zero
  - Campos Expoente $E = 0$, Fração $F = 0$
  - +0 e –0 são possíveis, de acordo c/ o bit de sinal *S*
- Infinito
  - Infinito= valor especial representado c/ *E máximo e* $F = 0$
    - Precisão Simples: $E = 255$
    - Precisão Dupla: $E = 2047$
  - Infinito é gerado em caso de overflow ou divisão por zero
  - Despendendo do sinal, temos $+\infty$ ou $-\infty$.
- NaN (Not a Number)
  - NaN = valor especial representado c/ *E máximo e* $F \neq 0$
  - NaN é gerada a partir de situações excepcionais, como 0/0 ou sqrt(nro negativo)
  - Operação usando um NaN resulta em um NaN.

Números Desnormalizados

- Padrão IEEE 754 usa números desnormalizados para:
  - Preencher o gap entre 0 e o menor nro normalizado.
  - Permitir underflow gradual até zero
- Desnormalizado: expoente $E = 0$, fração $F \neq 0$
  - 1. implícito se torna **0.** (não normalizado)
- Valor do número desnormalizado ($S, 0, F$)

Precisão Simples: $(-1)^S \times (0.F)_2 \times 2^{-126}$

Precisão Dupla: $(-1)^S \times (0.F)_2 \times 2^{-1022}$

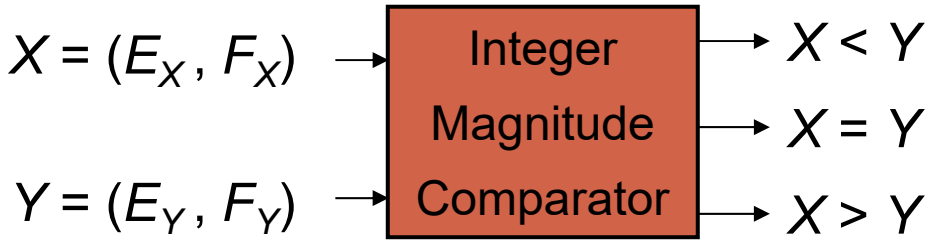


Regras p/ Cálculos c/ Valores Especiais

Operação	Resultado
n / ±∞	±0
±∞ x ±∞	±∞
nonzero / 0	±∞
∞ + ∞	∞ (similar for -∞)
±0 / ±0	NaN
∞ - ∞	NaN (similar for -∞)
±∞ / ±∞	NaN
±∞ x ±0	NaN
NaN op qualquer nro	NaN

Comparação entre nros FP

- Nros no padrão IEEE 754 são ordenados.
  - …porque o expoente usa representação polarizada (bias)
  - Expoente antes da Fração → ordena a magnitude.
    - Maior Expoente ⇒ Maior Magnitude
    - Expoentes iguais ⇒ Maior fração ⇒ Maior Magnitude
    - $0 < (0.F)_2 \times 2^{E_{min}} < (1.F)_2 \times 2^{E-Bias} < \infty$ ($E_{min} = 1 - Bias$)
  - Bit de sinal é o mais significativo ⇒ teste rápida p/ sinal
- Conclusão: comparação de inteiros é suficiente para comparar dois números em FP.

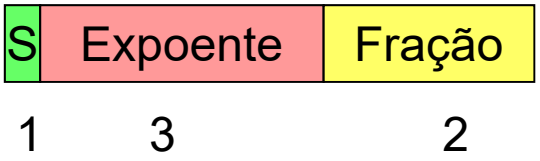


Codificação IEEE 754: Resumo

Single-Precision	Expoente = 8	Fração = 23	Value
Nro Normalizado	1 a 254	Qualquer	$\pm (1.F)_2 \times 2^{E-127}$
Nro Desnormalizado	0	Não-zero	$\pm (0.F)_2 \times 2^{-126}$
Zero	0	0	± 0
Infinito	255	0	± ∞
NaN	255	Não-zero	NaN

Double-Precision	Expoente = 11	Fração = 52	Value
Normalizado Number	1 to 2046	Qualquer	$\pm (1.F)_2 \times 2^{E-1023}$
Nro Desnormalizado	0	Não-zero	$\pm (0.F)_2 \times 2^{-1022}$
Zero	0	0	± 0
Infinito	2047	0	± ∞
NaN	2047	Não-zero	NaN

Exercício: Padrão FP c/ 6 bits



- Bit de sinal
- 3 bits p/ expoente c/ bias= 3
- 2 bits p/ fração

- Formato igual a IEEE-754
  - Normalizado, Desnormalizado
  - Representação de 0, infinito e NaN

- Valor dos números normalizados: $(-1)^S \times (1.F)_2 \times 2^{E-3}$
- Valor dos números desnormalizados: $(-1)^S \times (0.F)_2 \times 2^{-2}$

Construa uma tabela c/ todos os nros possíveis p/ este padrão FP de 6 bits.

Expoente: Valore Possíveis

Exp.	exp	E	2^E	
0	000	-2	¼	Desnormalizado
1	001	-2	¼	
2	010	-1	½	
3	011	0	1	Normalizado
4	100	1	2	
5	101	2	4	
6	110	3	8	
7	111	n/a		Inf ou NaN

Possíveis Valores

s	exp	frac	E	value	
0	000	00	-2	0	
0	000	01	-2	1/4*1/4=1/16	Menor desnormalizado
0	000	10	-2	2/4*1/4=2/16	
0	000	11	-2	3/4*1/4=3/16	Maior desnormalizado
0	001	00	-2	4/4*1/4=4/16=1/4=0.25	Menor normalizado
0	001	01	-2	5/4*1/4=5/16	
0	001	10	-2	6/4*1/4=6/16	
0	001	11	-2	7/4*1/4=7/16	
0	010	00	-1	4/4*2/4=8/16=1/2=0.5	
0	010	01	-1	5/4*2/4=10/16	
0	010	10	-1	6/4*2/4=12/16=0.75	
0	010	11	-1	7/4*2/4=14/16	

Possíveis Valores

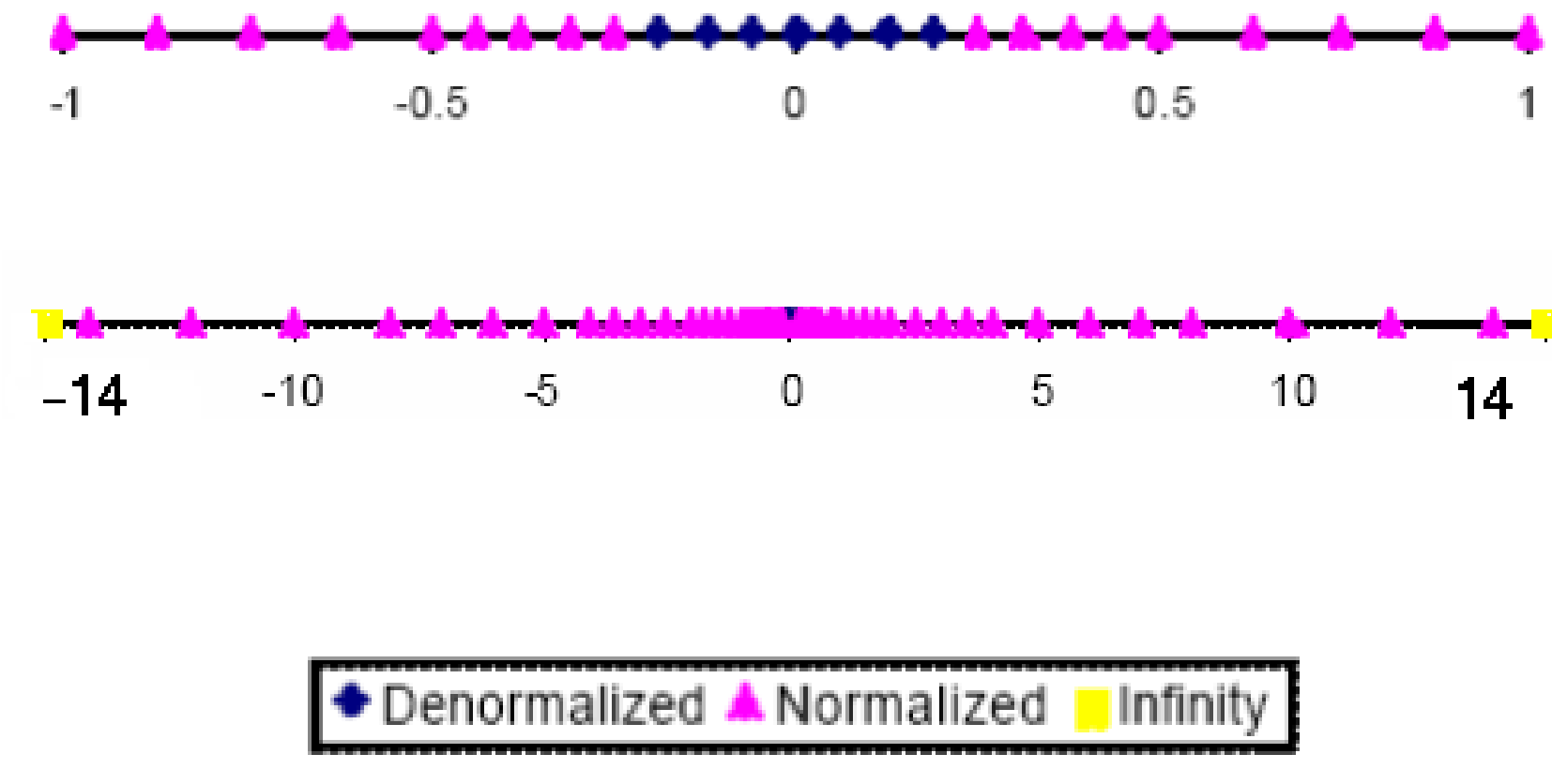
s	exp	frac	E	value
0	011	00	0	4/4*4/4=16/16=1
0	011	01	0	5/4*4/4=20/16=1.25
0	011	10	0	6/4*4/4=24/16=1.5
0	011	11	0	7/4*4/4=28/16=1.75
0	100	00	1	4/4*8/4=32/16=2
0	100	01	1	5/4*8/4=40/16=2.5
0	100	10	1	6/4*8/4=48/16=3
0	100	11	1	7/4*8/4=56/16=3.5
0	101	00	2	4/4*16/4=64/16=4
0	101	01	2	5/4*16/4=80/16=5
0	101	10	2	6/4*16/4=96/16=6
0	101	11	2	7/4*16/4=112/16=7

Possíveis Valores

s	exp	frac	E	value
0	110	00	3	4/4*32/4=128/16=8
0	110	01	3	5/4*32/4=160/16=10
0	110	10	3	6/4*32/4=192/16=12
0	110	11	3	7/4*32/4=224/16=14
0	111	00		∞
0	111	01		NaN
0	111	10		NaN
0	111	11		NaN

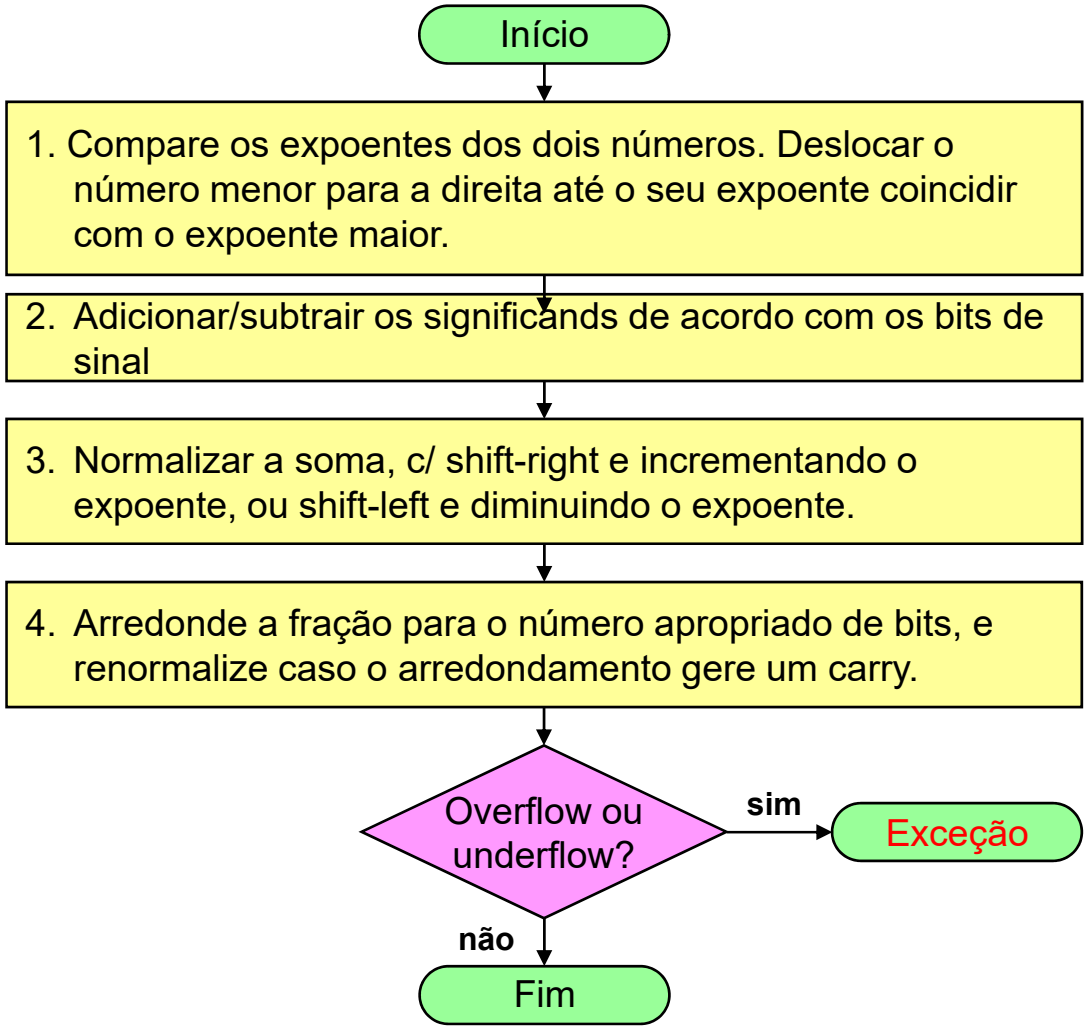
Maior normalizado

Distribuição de Valores



FP: Adição e Subtração

Floating Point: Adição / Subtração



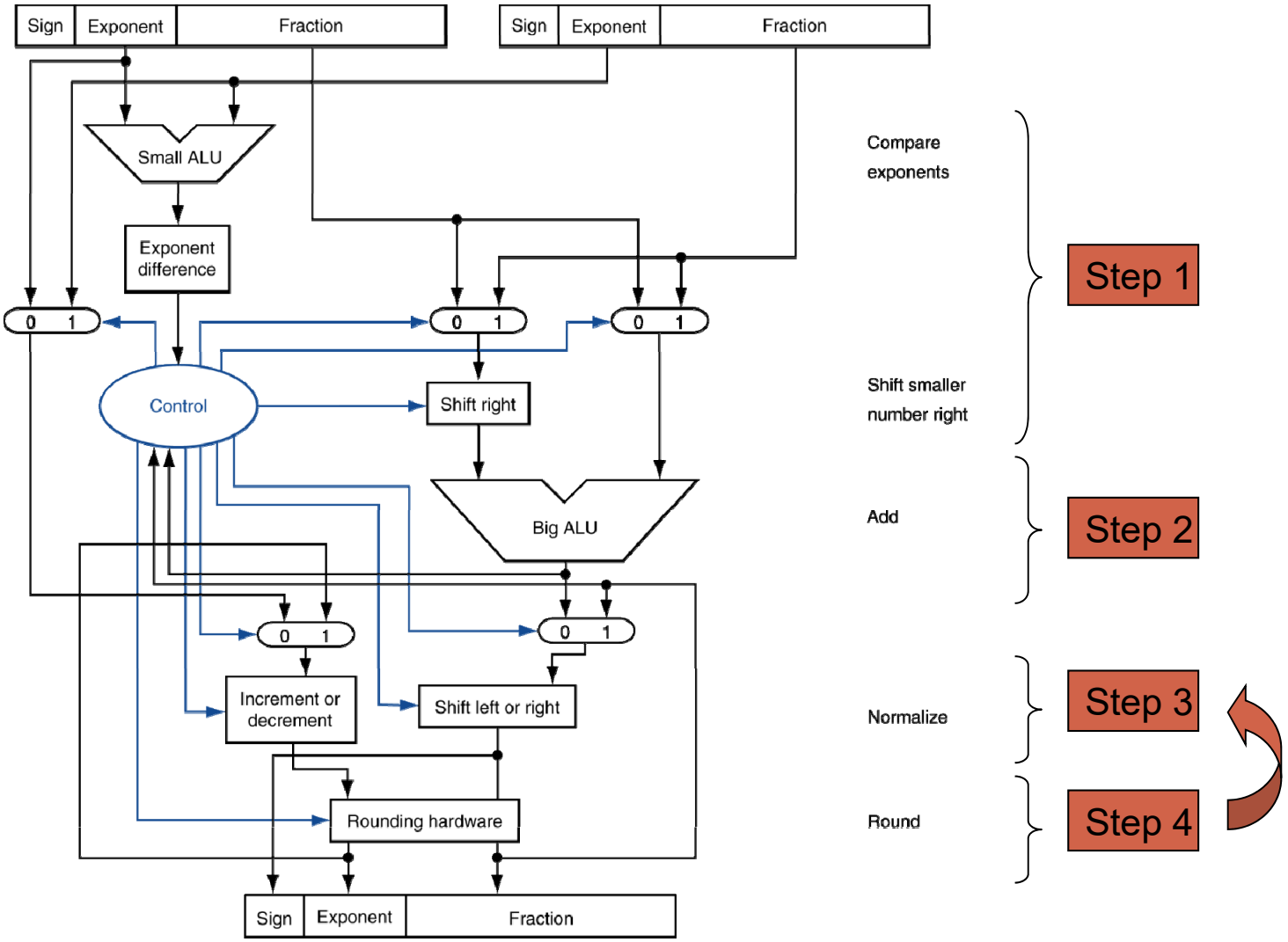
Shift-Rigth fração por:
$d = | E_X - E_Y |$

Adicionar significands quando os sinais de X e Y são idênticos, subtrair quando forem diferentes.
$X - Y$ torna-se $X + (-Y)$

Normalização desloca para a direita por 1 se houver um carry, ou shift-left pelo número de zeros à esquerda no caso de subtração

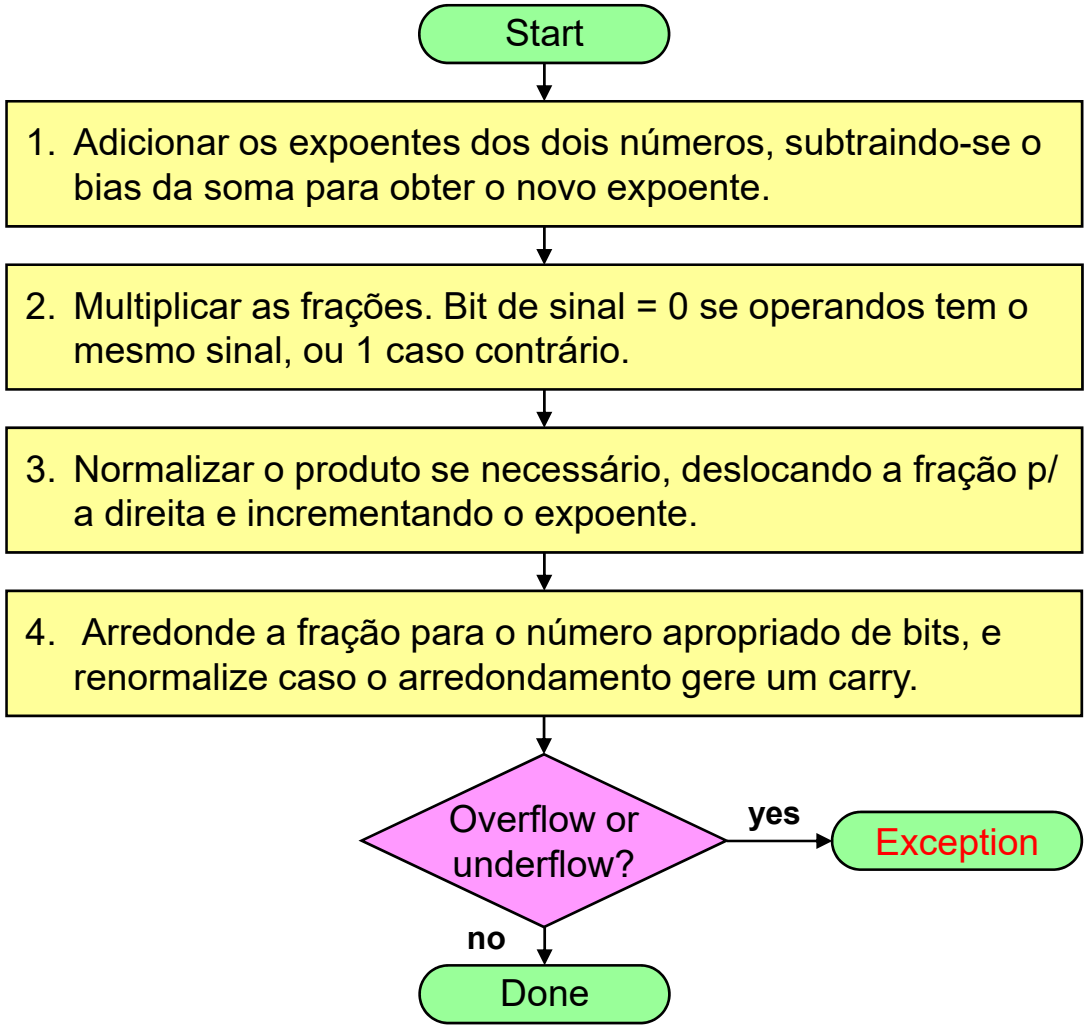
Arredondamento trunca a fração, ou adiciona 1 à fração

Floating Point: Add/Sub Block Diagram



FP: Multiplicação

Floating Point: Multiplicação



Somar expoentes

$E_Z = E_X + E_Y - Bias$

Bit de sinal $S_Z = S_X$ **xor** S_Y

Considerando que as frações são da forma $1.F_X$ e $1.F_Y$, ou seja, ≥ 1 e < 2, o produto delas é ≥ 1 e < 4.

Para normalizar o produto, deslocar apenas 1 bit p/ direita e incrementar o expoente.

Arredondamento trunca a fração, ou adiciona 1 à fração

Floating Point: Mul Block Diagram

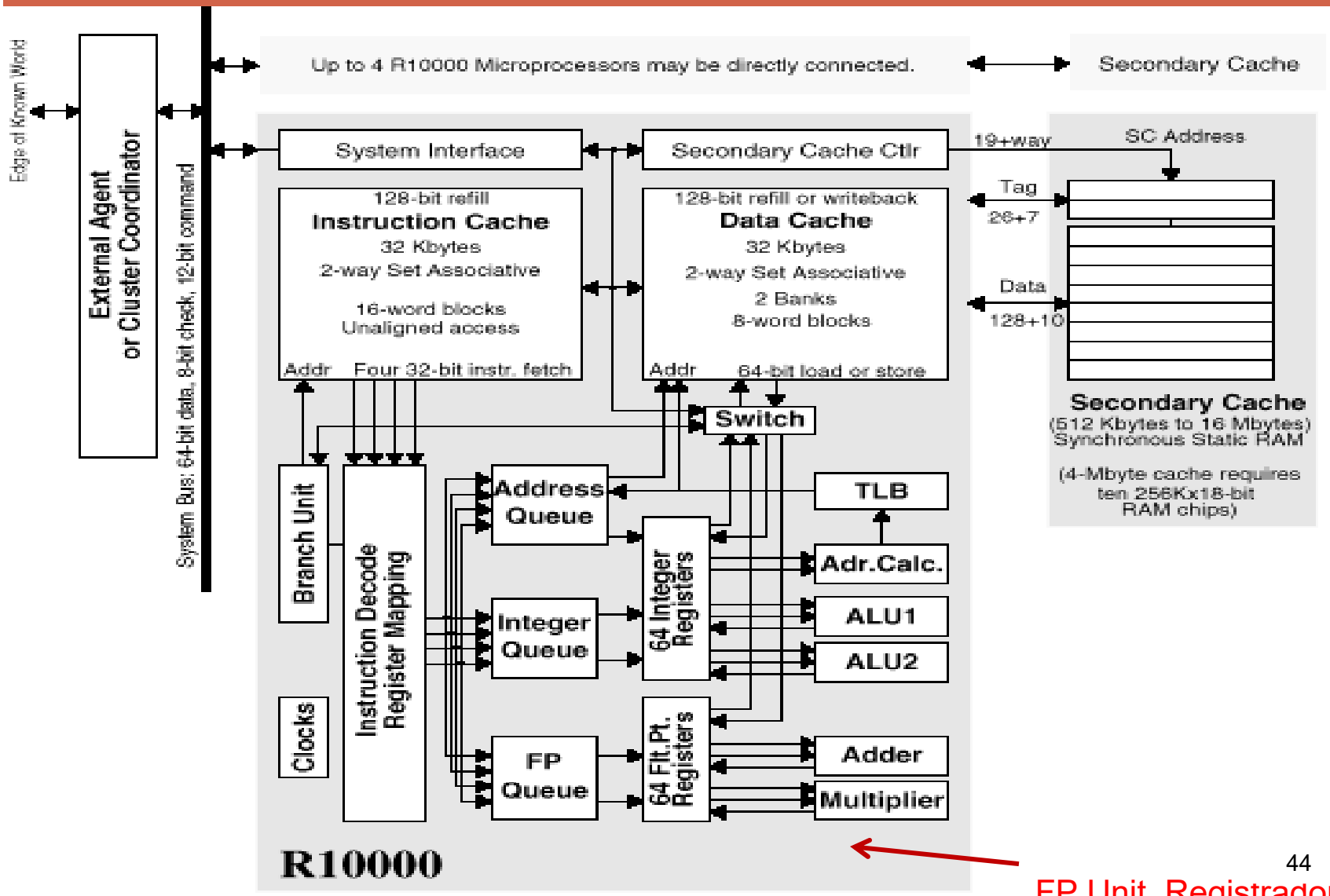
- Hardware p/ multiplicação em FP tem complexidade similar ao da adição/subtração.
  - … porém, usa multiplicação p/ as frações (significandos) ao invés de somas.
- A unidade aritmética de hardware p/ ponto flutuante geralmente executa as seguintes operações:
  - Adição, subtração, multiplicação, divisão, inverso, raiz-quadrada, conversão Inteiro ↔ FP.
- Essas operações geralmente demoram vários ciclos, mas podem ser executados c/ o esquema de pipeline.
- A unidade de FP pode ser vista como uma ULA especializada, acessando um banco de registradores também especializado.

FP: Instruções em MIPS Assembly

MIPS FP Coprocessor / Assembly

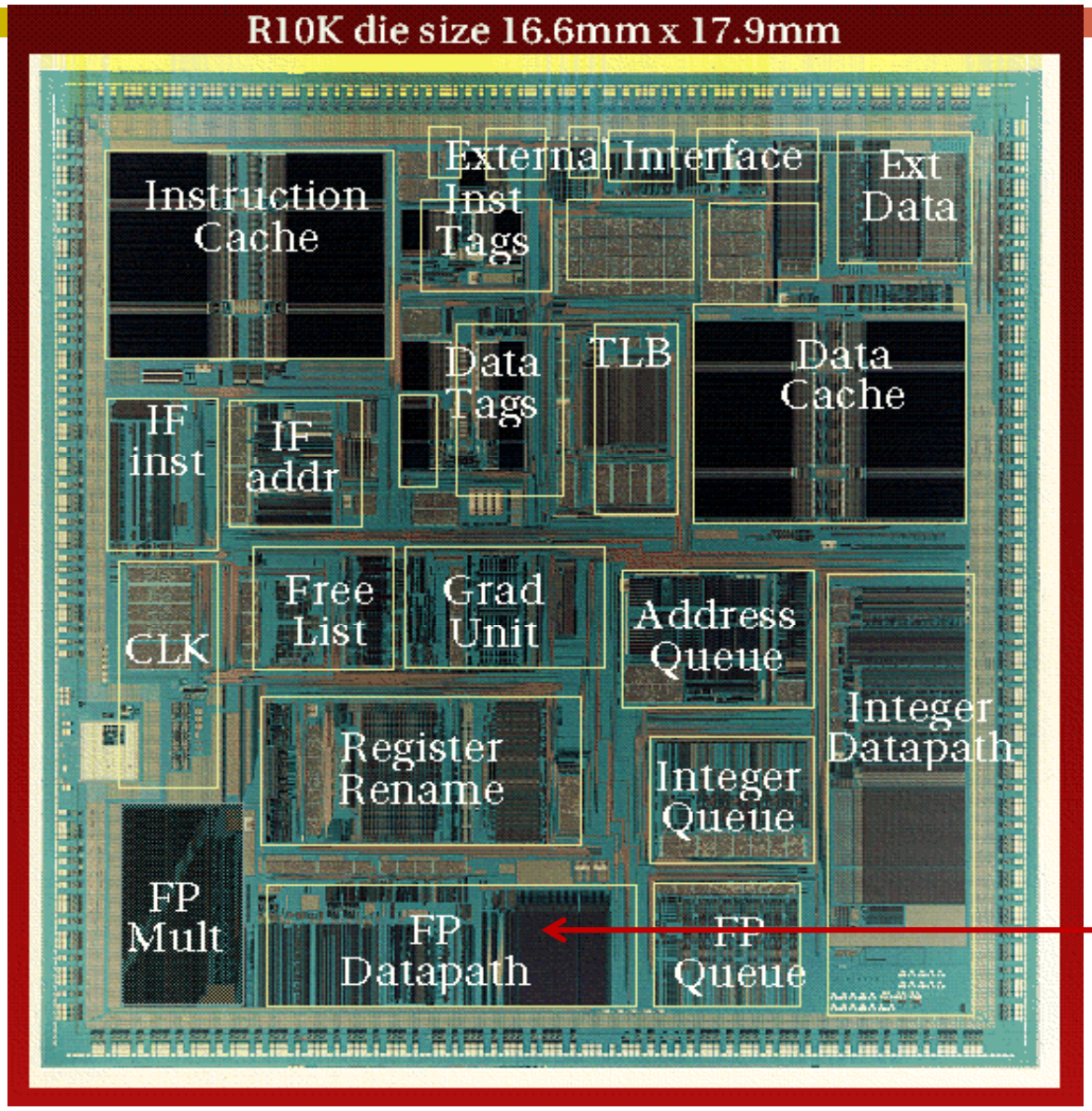
- Chamado Coprocessor 1 ou Floating Point Unit (FPU)
- 32 Registradores FP: $f0, $f1, …, $f31
  - 32 bits p/ precisão simples
  - Pares de registradore p/ precisão dupla (ex: $f4:$f5)
- Instruções diferentes p/ precisão simples/dupla
  - precisão simples : add.s, sub.s, mul.s, div.s (.s extensão)
  - precisão dupla : add.d, sub.d, mul.d, div.d (.d extensão)

Processador MIPS R-10000 (1996)



FP Unit, Registradores

Processador MIPS R-10000 (1996)



FP Unit, Registradores

MIPS: FP Instruções Aritméticas

Instrução	Significado						
add.s fd, fs, ft	(fd) = (fs) + (ft)						
add.d fd, fs, ft	(fd) = (fs) + (ft)						
sub.s fd, fs, ft	(fd) = (fs) – (ft)						
sub.d fd, fs, ft	(fd) = (fs) – (ft)						
mul.s fd, fs, ft	(fd) = (fs) × (ft)						
mul.d fd, fs, ft	(fd) = (fs) × (ft)						
div.s fd, fs, ft	(fd) = (fs) / (ft)						
div.d fd, fs, ft	(fd) = (fs) / (ft)						
sqrt.s fd, fs	(fd) = sqrt (fs)						
sqrt.d fd, fs	(fd) = sqrt (fs)						
abs.s fd, fs	(fd) = abs (fs)						
abs.d fd, fs	(fd) = abs (fs)						
neg.s fd, fs	(fd) = – (fs)						
neg.d fd, fs	(fd) = – (fs)						

MIPS: FP Instruções Load/Store

- lwc1: load word coprocessor 1
- ldc1: load double coprocessor 1
- swc1: store word coprocessor 1
- sdc1: store double coprocessor 1

Registrador de uso geral usado como base (ex: $t0).

Instrução	Significado				
lwc1 $f2, 40($t0)	($f2) = Mem[($t0)+40]				
ldc1 $f2, 40($t0)	($f2) = Mem[($t0)+40]				
swc1 $f2, 40($t0)	Mem[($t0)+40] = ($f2)				
sdc1 $f2, 40($t0)	Mem[($t0)+40] = ($f2)				

- Nomes alternativos p/ as instruções:
  - l.s = lwc1 (load FP single), l.d = ldc1 (load FP double)
  - s.s = swc1 (store FP single), s.d = sdc1 (store FP double)

MIPS: FP Instruções p/ Movimentação de dados

- Movimentação de dados entre registradores gerais e FP
  - mfc1: move from coprocessor 1 (to general purpose register)
  - mtc1: move to coprocessor 1 (from general purpose register)
- Movimentação de dados entre registradores FP
  - mov.s: move single precision float
  - mov.d: move double precision float = even/odd pair of registers

Instrução	Significado						
mfc1 $t0, $f2	($t0) = ($f2)						
mtc1 $t0, $f2	($f2) = ($t0)						
mov.s $f4, $f2	($f4) = ($f2)						
mov.d $f4, $f2	($f4) = ($f2)						

MIPS: FP Instruções p/ Conversão de dados

- Instrução de Conversão Genérica: cvt.x.y
  - Converte p/ formato destino x a partir do formato origem y
- Formatos Suportados
  - Single precision float = .s (single precision float in FP register)
  - Double precision float = .d (double float in even-odd FP register)
  - Signed integer word = .w (signed integer in FP register)

Instrução	Significado						
cvt.s.w fd, fs	to single from integer						
cvt.s.d fd, fs	to single from double						
cvt.d.w fd, fs	to double from integer						
cvt.d.s fd, fs	to double from single						
cvt.w.s fd, fs	to integer from single						
cvt.w.d fd, fs	to integer from double						

MIPS: FP Instruções de Comparação e Desvio

- FP unit (co-processor 1) tem uma cflag (bit) condicional
  - Set p/ 0 (false) ou 1 (true) a partir da instrução de comparação
- 3 tipos de comparações: ==, <, <=
- 2 instruções de desvio baseadas na flag de comparação.

Instrução	Significado
c.eq.s fs, ft	cflag = ((fs) == (ft))
c.eq.d fs, ft	cflag = ((fs) == (ft))
c.lt.s fs, ft	cflag = ((fs) < (ft))
c.lt.d fs, ft	cflag = ((fs) < (ft))
c.le.s fs, ft	cflag = ((fs) <= (ft))
c.le.d fs, ft	cflag = ((fs) <= (ft))
bc1f Label	branch if (cflag == 0)
bc1t Label	branch if (cflag == 1)

MIPS: FP Diretivas de Dados (declaração)

- .FLOAT Directive
  - Armzena os valores declarados em FP precisão simples
- .DOUBLE Directive
  - Armzena os valores declarados em FP precisão dupla
- Exemplos
  - **var1: .FLOAT 12.3, -0.1**
  - **var2: .DOUBLE 1.5e-10**
  - **pi: .DOUBLE 3.1415926535897924**

Chamadas de Syscall p/ FP

Service	$v0	Arguments / Result
Print Integer	1	$a0 = integer value to print
Print Float	2	$f12 = float value to print
Print Double	3	$f12 = double value to print
Print String	4	$a0 = address of null-terminated string
Read Integer	5	$v0 = integer read
Read Float	6	$f0 = float read
Read Double	7	$f0 = double read
Read String	8	$a0 = address of input buffer $a1 = maximum number of characters to read
Exit Program	10	
Print Char	11	$a0 = character to print
Read Char	12	$a0 = character read

Exemplo 1: Area do Círculo

```
.data
  pi:     .double          3.1415926535897924
  msg:    .asciiz          "Circle Area = "
.text
main:
  ldc1    $f2, pi          # $f2,3 = pi
  li      $v0, 7           # read double (radius)
  syscall                  # $f0,1 = radius
  mul.d   $f12, $f0, $f0   # $f12,13 = radius*radius
  mul.d   $f12, $f2, $f12  # $f12,13 = area
  la      $a0, msg
  li      $v0, 4           # print string (msg)
  syscall
  li      $v0, 3           # print double (area)
  syscall                  # print $f12,13
```

Exemplo 2: °F to °C

- **C code:**

```
float f2c (float fahr) {
  return ((5.0/9.0)*(fahr - 32.0));
}
```

  - **fahr in $f12, result in $f0, literals in global memory space**

- **MIPS code:**

```
f2c: lwc1  $f16, const5($gp)
     lwc2  $f18, const9($gp)
     div.s $f16, $f16, $f18
     lwc1  $f18, const32($gp)
     sub.s $f18, $f12, $f18
     mul.s $f0,  $f16, $f18
     jr    $ra
```

Fogete Ariane 5 (1996)

- Explodiu 37 segundos após decolagem
- Custo: U$ 7 Bilhões
- Tempo Desenvolvimento: 10 anos
- Por que?
  - Foi computada a velocidade horizontal como número em ponto flutuante de 64 bits.
  - Convertido para inteiro de 16-bits
  - Funcionou bem para Ariane 4
  - Ocorreu overflow para Ariane 5 (mesmo software)

Outros desastres devido a cálculos numéricos errados:
http://www.ima.umn.edu/~arnold/disasters/