Lecture 7 计算机的浮点数运算

0. 大纲

- 浮点数表达
- 浮点数加法
- 浮点数乘法
- MIPS中的浮点数指令
- 计算精确性
- 子字并行

1. 浮点数表达

浮点数，是非整数的其它数的表达，包括非常小和非常大的数

科学计数法

- -2.34×10^{56}
- $+0.002 \times 10^{-4}$
- $+987.02 \times 10^{9}$

二进制表示

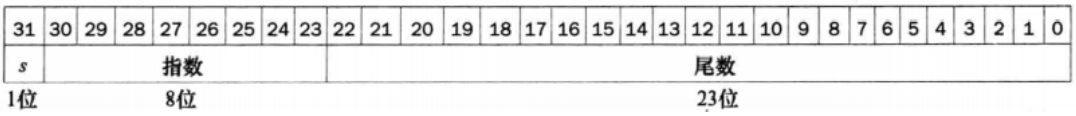
- $1.xxxxxxx_2 \times 2^{yyyy}$

C语言表示

```
float
double
```

IEEE浮点数格式

$$\pm 1.xxxxxx_2 \times 2^{yyyy}$$



- S：符号位（0-正数，1-负数）
- Exponent：指数位（单精度8bit，双精度11bit）
  表示小数点的位置
- Fraction：尾数位（单精度23bit，双精度52bit）
  其值在0和1之间

MIPS浮点数表示的数值范围是：$[2.0_{10} \times 10^{-38}, 2.0_{10} \times 10^{38}]$

- 当然还是可能存在 overflow 的情况

浮点数的表示

- 单精度表示（32-bit）
  $[2.0_{10} \times 10^{-38}, 2.0_{10} \times 10^{38}]$
- 双精度表示（64-bit），占用两个MIPS字
  $[2.0_{10} \times 10^{-308}, 2.0_{10} \times 10^{308}]$

偏阶 **biased**

如果使用补码计数法，那么负数的指数位高位为1，在浮点数表示下就是一个比较大的数了

我们希望计数法可以将最小的负指数标识为 $00..00_2$，而最大的正指数表为 $11..11_2$ 这种计数法称为带偏阶计数法 biased notation，从指数减去偏阶，才能得到真正的值

- 单精度的偏阶为 127
- 双精度的偏阶为 1023

给定指数带偏阶后，浮点数的表示为

$$(-1)^S \times (1 + Fraction) \times 2^{(Exponent - Bias)}$$

故指数为带偏阶的范围为（不确定）

- 单精度[0, 255] → [-127, 128]
- 双精度[0, 2047] → [-1023, 1024]

浮点数精度

相对准确性，所有的尾数位都是有效的

- 单精度：$approx \times 2^{-23}$
  相当于$23 \times log_{10}2 \approx 23 \times 0.3 \approx 6$个十进制精度
- 双精度：$approx \times 2^{-52}$

  $52 \times log_{10}2 \approx 52 \times 0.3 \approx 16$$个十进制精度

示例

例题1

以单/双精度形式表示 -0.75_{10}

$$-0.75_{10} = -1.1_2 \times 2^{-1}$$
$$(-1)^1 \times (1 + 0.1000\ 0000\ 0000\ 0000\ 0000\ 000_2) \times 2^{(126-127)}$$
$$(-1)^1 \times (1 + 0.1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2) \times 2^{(1022-1023)}$$

- $-0.75 = (-1)^1 \times 1.1_2 \times 2^{-1}$
- S = 1
- Fraction = $1000...00_2$
- Exponent = –1 + Bias
  - Single: –1 + 127 = 126 = 01111110_2
  - Double: –1 + 1023 = 1022 = 01111111110_2

Single: 10111111010000...00

Double: 10111111111010000...00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1位　　8位　　23位

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1位　　11位　　20位

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

例题2

二进制转十进制浮点数

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	.	.	.

$$(-1)^S \times (1 + Fraction) \times 2^{(Exponent-Bias)}$$
$$= (-1)^1 \times (1 + 0.25) \times 2^{(129-127)} = -5.0$$

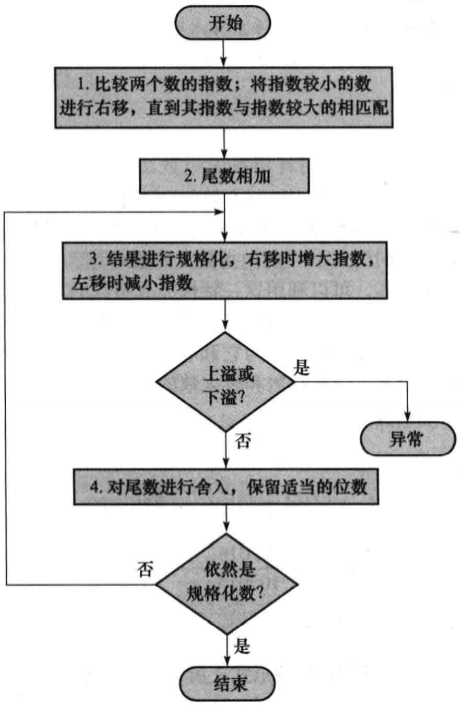
例题3

假设有一个 16 位的 IEEE 754-2008 浮点格式，其中有 5 位指数位。那么它可能表示的数的范围是多少？

1. $1.0000\ 0000\ 00 \times 2^{0}$ 到 $1.1111\ 1111\ 11 \times 2^{31}$，0
2. $\pm 1.0000\ 0000\ 0 \times 2^{-14}$ 到 $\pm 1.1111\ 1111\ 1 \times 2^{15}$，$\pm 0$，$\pm \infty$，NaN
3. $\pm 1.0000\ 0000\ 00 \times 2^{-14}$ 到 $\pm 1.1111\ 1111\ 11 \times 2^{15}$，$\pm 0$，$\pm \infty$，NaN
4. $\pm 1.0000\ 0000\ 00 \times 2^{-15}$ 到 $\pm 1.1111\ 1111\ 11 \times 2^{14}$，$\pm 0$，$\pm \infty$，NaN

2. 浮点数加法

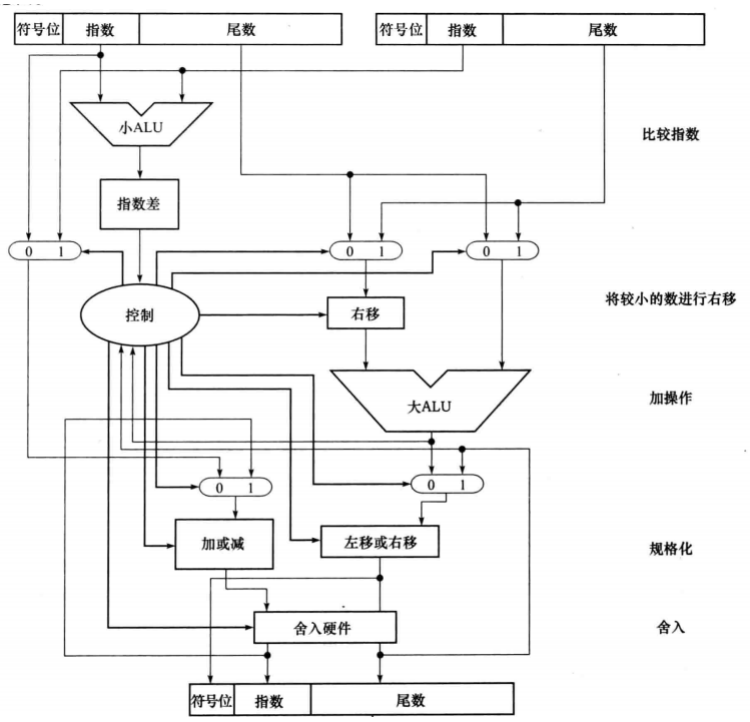
示例



考虑浮点数加法 $1.000_2 \times 2^{-1} + -1.110_2 \times 2^{-2} (0.5 + -0.4375)$

1. 二进制点对齐（较小指数的数向有较大指数的数对齐）
   $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1}$
2. 将有效数相加
   $1.000_2 \times 2^{-1} + -0.111_2 \times 2^{-1} = 0.001_2 \times 2^{-1}$
3. 规范科学技术形式，并检查是否溢出
   $1.000_2 \times 2^{-4}$ 没有溢出
4. 如有必要，舍入
   $1.000_2 \times 2^{-4} = 0.0625$（不需要四舍五入）

浮点数加法器



用于浮点加的算术单元的结构框图

1. 首先，使用一个小的ALU将两个指数相减来决定哪个指数大及大多少
2. 指数差将控制三个多路复用器：从左到右，选择出较大的指数、较小数的有效数和较大数的有效数
3. 较小数的有效数通过右移后，和较大数的有效数用一个大的ALU相加
4. 规范化步骤将和左移或者右移,同时增加或者减少指数
5. 舍入产生最后的结果,这样也有可能需要再次规格化,然后产生最后的结果

在一个时钟周期内完成它将花费太长时间

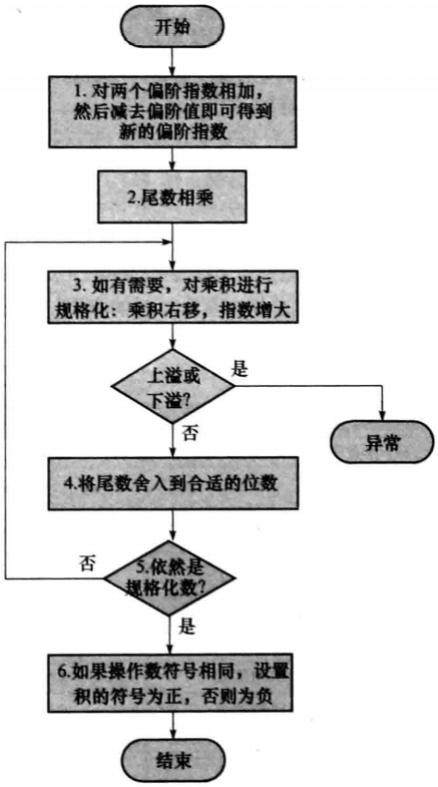
- 比整型运算要长得多
- 慢一点的时钟会惩罚所有的指令

FP加法器通常需要几个周期

- 可以流水线 pipelined 操作

3. 浮点数乘法

示例



考虑浮点数加法 $1.000_2 \times 2^{-1} \times -1.110_2 \times 2^{-2} (0.5 \times -0.4375)$

1. 计算指数位
   不像加法，我们字还是简单的将源操作数的指数相加作为积的指数，但是需要注意，带有偏阶的指数想要获得相同的结果，**必须要两个指数相加再减去127**，即 $Exp_1 + Exp_2 - 127$ 才能得到正确结果
   $-3 + 127 = 124$
2. 有效数相乘
   $1.000_2 \times 1.110_2 = 1.110000_2$
3. 标准化结果
   我们需要保存4位，所以为 $1.110_2 \times 2^{-3}$
4. 舍入
   不需要舍入
5. 计算符号位
   由于初始源操作数符号相异，所以积的符号为负，因此结果为 $-1.110_2 \times 2^{-3}$

4. MIPS中的浮点数指令

操作介绍

FP硬件是 coprocesssor1

- 扩展ISA的辅助处理器

单独的浮点寄存器FP

- 单精度： $f0, $f1, ... $f31
- 双精度：($f0,$f1)，($f2,$f3)（偶数-奇数对）

FP指令只对FP寄存器进行操作

- 程序通常不会对FP数据执行整数操作，反之亦然
- 使用最小代码大小影响的更多寄存器

常见浮点数指令

加载/保存指令

```
lwc1 # 加载单精度浮点数
ldc1 # 加载双精度浮点数
swc1 # 保存单精度浮点数
sdc1 # 保存双精度浮点
```

单精度运算

```
add.s
sub.s
mul.s
div.s
```

双精度运算

```
add.d
sub.d
mul.d
div.d
```

单精度/双精度比较

```
c.eq.s # 单精度等于比较
c.lt.s # 单精度小于比较
c.le.s # 单精度小于等于

c.eq.d # 双精度等于比较
c.gt.d # 双精度大于比较
c.ge.d # 双精度大于等于
```

条件分支

```
bc1t # 如果浮点标志为真则进行跳转
bc1f # 如果浮点标志为假则进行跳转
```

5. 计算精确性

浮点数通常是一个无法表示的数的近似，IEEE 754提供几种舍入模式，在中间计算时，右边总时多保留两位，分别称为**保护位**和**舍入位**

- 保护位 guard ：在右边多保留两位中的**首位**，用于提高舍入精度
- 舍入位 round：在右边多保留两位张的**第二位**，使浮点中间结果满足浮点格式，得到最接近的数

示例

例题 • 使用保护位来舍入

将 $2.56_{10} \times 10^{0}$ 和 $2.34_{10} \times 10^{2}$ 相加，假设我们有 3 位十进制尾数。首先使用保护位和舍入位将其舍入到只有三位尾数的最近数，然后不用保护位和舍入位再做一次（舍入）。

答案

首先我们右移较小数以对齐指数，所以 $2.56_{10} \times 10^{0}$ 变为 $0.025\,6_{10} \times 10^{2}$。因为有了保护位和舍入位，所以当我们对齐指数时可以表示两个最低位。保护位为 5 而舍入位为 6。求和：

$$\begin{array}{rr} & 2.340_{10} \\ + & 0.025\,6_{10} \\ \hline & 2.365\,6_{10} \end{array}$$

因此，和为 $2.365\,6_{10} \times 10^{2}$。因为需要舍入掉两位，所以我们需要以 50 为分水岭，在其值为 0 ~ 49 之间时舍掉，在 51 ~ 99 之间时向上舍入。向上舍入这个和，变为 $2.37_{10} \times 10^{2}$。

在计算中，在没有保护位和舍入位的情况下舍入掉两位。新的和为：

$$\begin{array}{rr} & 2.34_{10} \\ + & 0.02_{10} \\ \hline & 2.36_{10} \end{array}$$

浮点数的精确性通常是用**尾数的最低有效位上有多少 bit 误差**来衡量，这种衡量称为**尾数最低位 units in the last place，ulp**

如果一个数在最低位上少2，则称其少了2个ulp，IEEE 754保证了计算机使用的数的误差都在半个**ulp**以内

- 额外的精度
- 舍入方式的选择
  向上四舍五入，向下四舍五入，截断，向靠近的偶数舍入
- 允许程序员微调计算的数值行为
- 硬件复杂性、性能和市场需求之间的平衡

6. 并行性和计算机算数：子字并行

子字并行 Subword Parallelism

图形和音频应用程序可以同时对这类数据向量在128bit内对其进位链进行分割

示例：128 bit：

- 16个：8bit-8bit计算
- 8个：16bit-16bit计算按
- 4个：32bit-32bit计算

将这种一个 wide word 内部进行的并行操作称为子字并行，也称为

- 数据级并行
- 向量并行
- 单指令、多数据 Single Instruction, Multiple Data（SIMD）

支持子字并行的操作系统

- ARM 在 NEON 多媒体指令集中增加了多条指令来支持子字并行
- x86中的 MMX（MultiMedia eXtension 多媒体扩展）指令和 SSE（Streaming SIMD Extension 流处理SIMD扩展） 指令

7. 总结

数据类型的指令总结

计算机数和真实世界里的数的主要不同是计算机数的大小是有限制的，因此限制了其精度

计算的数字有可能太大或太小而无法在一个字中表示，程序员必须记住这些限制并相应地编程

C 类型	Java 类型	数据传送	操作
int	int	lw,sw,lui	addu, addiu, subu, mult, div, AND, ANDi, OR, ORi, NOR,slt,slti
Unsigned int	—	lw,sw,lui	addu, addiu, subu, multu, divu, AND, ANDi, OR, ORi, NOR,sltu,sltiu
char	—	lb,sb,lui	add, addi, sub, mult, div, AND, ANDi, OR, ORi, NOR, slt,slti
—	char	lh,sh,lui	addu, addiu, subu, multu, divu, AND, ANDi, OR, ORi, NOR,sltu,sltiu
float	float	lwc1,swc1	add. s,sub. s,mult. s,div. s,c. eq. s,c. lt. s,c. le. s
double	double	l. d,s. d	add. d,sub. d,mult. d,div. d,c. eq. d,c. lt. d,c. le. d