CSE 2021
Computer Organization

Hugh Chesser,
CSEB 1012U

W4-W

Agenda for Today

1. Floating Point Addition, Multiplication
2. FP Instructions
3. Quiz 1

Patterson: Sections 3.5

Floating Point: Single Precision

1. In MIPS, decimal numbers are represented with the IEEE 754 binary representation that uses the **normalized** standard scientific binary notation defined as

$$(-1)^S \times (1 + \text{fraction})_{two} \times 2^{\text{exponent}-\text{bias}}$$

2. A number in normalized scientific notation has a mantissa that has no leading 0's and must be of the form (1 + fraction) . For example, the binary representations 2.0×2^{-5}, 0.5×2^{-3}, 4.0×2^{-6}, and 1.0×2^{-4} are all equivalent but only 1.0×2^{-4} is the normalized scientific binary notation.
3. MIPS allows for two floating point representations: Single precision and double precision.
4. Single precision has a bias of 127 while double precision has a bias of 1023.
5. In single precision, the floating point representation is 32 bit long and has the following form

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
S	**exponent**								**fraction**																						
	(8 bits)								**(23 bits)**																						

where S represents the sign bit, which is 1 for negative numbers and 0 for positive numbers.

Activity 2:

Represent -0.75_{ten} in single precision of IEEE 754 binary representation.

Floating Point: Double Precision

1. In double precision, the value of bias in

$$(-1)^S \times (1 + \text{fraction})_{\text{two}} \times 2^{\text{exponent}-\text{bias}}$$

is 1023.

2. In single precision, the floating point representation is 64 bit long and has the following form

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
S	**exponent**											**fraction**																			
	(11 bits)											**(Total of 52 bits)**																			

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
fraction (continued)																															

Activity 3:

Represent -0.75_{ten} in double precision of IEEE 754 binary representation.

Activity 4:

Show that the largest magnitude that can be represented using single precision is $\pm 6.8_{\text{ten}} \times 10^{38}$, while the smallest fraction that can be represented is $\pm 5.9_{\text{ten}} \times 10^{-38}$.

Floating Point Registers

Name	Example	Comments
32 floating point registers each is 32 bits long	`$f0,$f1,$f2,$f3,$f4,…,$f31`	**MIPS floating point registers are used in pairs for double precision numbers**
Memory w/ 2^{30} words	**Memory[0], Memory[4], … Memory[4294967292]**	**Memory is accessed one floating point (single or double precision) at a time**

The following is the established register usage convention for the floating point registers:

`$f0,$f1,$f2,$f3:`	Function-returned values
`$f4,$f5,…, $f11:`	Temporary values
`$f12,$f13,$f14,$f15:`	Arguments passed into a function
`$f16,$f17,$f18,$f19:`	More Temporary values
`$f20,$f21,…, $f31:`	Saved values

A handy online calculator/converter for IEEE 754 FP format is at:

http://babbage.cs.qc.edu/IEEE-754/Decimal.html

Floating Point Instructions

Category	Instruction	Example	Meaning	Comments
Arithmetic	**FP add single**	`add.s $f2,$f4,$f6`	`$f2 ← $f4+$f6`	**Single Prec.**
	FP subtract single	`sub.s $f2,$f4,$f6`	`$f2 ← $f4-$f6`	**Single Prec.**
	FP multiply single	`mul.s $f2,$f4,$f6`	`$f2 ← $f4×$f6`	**Single Prec.**
	FP divide single	`div.s $f2,$f4,$f6`	`$f2 ← $f4/$f6`	**Single Prec.**
	FP add double	`add.d $f2,$f4,$f6`	`$f2 ← $f4+$f6`	**Double Prec.**
	FP subtract double	`sub.d $f2,$f4,$f6`	`$f2 ← $f4-$f6`	**Double Prec.**
	FP multiply double	`mul.d $f2,$f4,$f6`	`$f2 ← $f4×$f6`	**Double Prec.**
	FP divide double	`div.d $f2,$f4,$f6`	`$f2 ← $f4/$f6`	**Double Prec.**
Data Transfer	**load word FP Single**	`lwc1 $f2,100($s2)`	`$f2 ← Mem[$s2+100]`	**Single Prec.**
	store word FP Single	`swc1 $f2,100($s2)`	`Mem[$s2+100] ← $f2`	**Single Prec.**
Conditional branch	**FP compare single (eq, ne, lt, le, gt, ge)**	`c.lt.s $f2,$f4`	`if($f2<$f4)cond = 1, else cond = 0`	**Single Prec.**
	FP compare double (eq, ne, lt, le, gt, ge)	`c.lt.d $f2,$f4`	`if($f2<$f4)cond = 1, else cond = 0`	**Double Prec.**
	Branch on FP true	`bc1t 25`	`if cond==1 go to PC+100+4`	**Single/ Double Prec.**
	Branch on FP false	`bc1f 25`	`if cond==0 go to PC+100+4`	**Single/ Double Prec.**

Example

```
# calculate area of a circle
        .data
Ans:    .asciiz     "The area of the circle is: "
Ans_add: .word      Ans                         # Pointer to String (Ans)
Pi:     .double     3.1415926535897924
Rad:    .double     12.345678901234567
Rad_add: .word      Rad                         # Pointer to float (Rad)
        .text
main:   lw $a0, Ans_add($0)                     # load address of Ans into $a0
        addi $v0, $0, 4                         # Sys Call 4 (Print String)
        syscall
#----------------                               # load float (Pseudoinstruction)
        la $s0, Pi                              # load address of Pi into $s0
        ldc1 $f2, 0($s0)                        # $f2 = Pi
#----------------                               # load float (MIPS Instruction)
        lw $s0, Rad_add($0)                     # load address of Rad into $s0
        ldc1 $f4, 0($s0)                        # $f4 = Rad
        mul.d $f12, $f4, $f4
        mul.d $f12, $f12, $f2
        addi $v0, $0, 3                         # Sys Call 3 (Print Double)
        syscall
exit:   jr $ra
```