COSC 2021: Computer Organization
Instructor: Dr. Amir Asif
Department of Computer Science
York University
Handout # 6
Unsigned integer and floating point instructions

Topics:
1. Unsigned instructions
2. Floating Point: IEEE 754 single and double precision formats
3. Floating Point Registers and Instructions

Patterson: Sections 3.1 – 3.2, 3.5

Unsigned and Signed Arithmetic

MIPS has a separate format for unsigned and signed integers

1. Unsigned integers
   — are saved as 32-bit words
   — Example: Smallest unsigned integer is $00000000_{hex} = 0_{ten}$
     Largest unsigned integer is $ffffffff_{hex} = 4{,}294{,}967{,}295_{ten}$
2. Signed integers
   — are saved as 32-bit words in 2's complement with the MSB reserved for sign
   — If MSB = 1, then the number is negative
   — If MSB = 0, then the number is positive
   — Example:
     Smallest signed integer: $1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two}$
     $= -(2^{31})_{10} = -2{,}147{,}483{,}648_{10}$
     Largest signed integer: $0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two}$
     $= (2^{31} - 1)_{10} = 2{,}147{,}483{,}647_{10}$

MIPS Commands for Unsigned Numbers

Category	Instruction	Example	Meaning	Comments
Arithmetic	add(u)	`add $s1,$s2,$s3`	`$s1 ← $s2+$s3`	Most arithmetic
	Subtract(u)	`sub $s1,$s2,$s3`	`$s1 ← $s2-$s3`	instruction have
	addi(u)	`add $s1,$s2,100`	`$s1 ← $s2+100`	unsigned format
Data Transfer	load word	`lw $s1,100($s2)`	`$s1 ← Mem[$s2+100]`	
	store word	`lw $s1,100($s2)`	`Mem[$s2+100] ← $s1`	
	load byte unsigned	`lbu $s1,100($s2)`	`$s1 ← Mem[$s2+100]`	Unsigned
	store byte	`sb $s1,100($s2)`	`Mem[$s2+100] ← $s1`	1 byte only
Conditional branch	branch on equal	`beq $s1,$s2,L`	`if($s1==$s2) go to L`	
	branch not equal	`bne $s1,$s2,L`	`if($s1!=$s2) go to L`	
	set on less than	`slt $s1,$s2,$s3`	`if($s2<$s3) $s1 = 1 else $s1 = 0`	
	set on less than immediate	`slti $s1,$s2,10`	`if($s2<10) $s1 = 1 else $s1 = 0`	
	set less than unsigned	`sltu $s1,$s2,$s3`	`- same as slt -`	Unsigned
	slt unsign immediate	`sltui $s1,$s2,100`	`- same as slti -`	Unsigned
Unconditional jump	jump	`j 2500`	`go to (4 x 2500)`	
	jump register	`jr $ra`	`go to $t1`	
	jump and link	`jal fact`	`go to fact; set $ra = PC + 4`	

Addition and Subtraction

In MIPS, addition and subtraction for signed numbers use 2's complement arithmetic

Example 1: Add 10_{ten} and 15_{ten}

Step 1: Represent the operands in 2's complelement

$10_{ten} = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1010_{two}$

$15_{ten} = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111_{two}$

Step 2: Perform bit by bit addition using table 1.

$10_{ten} + 15_{ten}$

$= 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1001_{two}$

$= 25_{ten}$

Example 2: Subtract 15_{ten} from 10_{ten}

The problem is reduced to $(10_{ten} + (-15_{ten}))$

$10_{ten} = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1010_{two}$

$-15_{ten} = 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 0001_{two}$

$10_{ten} - 15_{ten}$

$= 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1011_{two}$

$= -5_{ten}$

bit 1	bit 2	Prev. Carry	Sum	Next Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 1: Truth Table for addition

Overflow (1)

Recall that:

Smallest signed integer: $1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two}$

$= -(2^{31})_{ten} = -2{,}147{,}483{,}648_{ten}$

Largest signed integer: $0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two}$

$= (2^{31} - 1)_{ten} = 2{,}147{,}483{,}647_{ten}$

What happens if the result of an operation is more than the largest signed integer or less than the smallest signed integer?

Example: Add $2{,}147{,}483{,}640_{ten}$ and 28_{ten}

28_{ten} $= 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 1100_{two}$

$2{,}147{,}483{,}640_{ten}$ $= 0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1000_{two}$

$28_{ten} + 2{,}147{,}483{,}640_{ten}$ $= 1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001\ 0100_{two}$

$= -2{,}147{,}483{,}628_{ten}$

Overflow caused the value to be perceived as a negative integer

Overflow (2)

When can overflow occur?

Operation	Operand A	Operand B	Result indicating overflow
A + B	A >= 0	B >= 0	< 0
A + B	A < 0	B < 0	>= 0
A − B	A >= 0	B < 0	< 0
A − B	A < 0	B >= 0	>= 0

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

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>S</td><td colspan="8">exponent</td><td colspan="23">fraction</td></tr>
<tr><td></td><td colspan="8">(8 bits)</td><td colspan="23">(23 bits)</td></tr>
</table>

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

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

is 1023.

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

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>S</td><td colspan="11">exponent</td><td colspan="20">fraction</td></tr>
<tr><td></td><td colspan="11">(11 bits)</td><td colspan="20">(Total of 52 bits)</td></tr>
</table>

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">fraction (continued)</td></tr>
</table>

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 $\pm 2.0_{ten} \times 10^{38}$, while the smallest fraction that can be represented is $\pm 2.0_{ten} \times 10^{-38}$.

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

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 copr.1	`lwc1 $f2,100($s2)`	`$f2 ← Mem[$s2+100]`	Single Prec.
	store word copr.1	`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 (Assembler Instruction)
       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
```