IC220
Slide Set #9:
Computer Arithmetic (Chapter 3)

Chapter Goals

- Introduce 2’s complement numbers
  - Addition and subtraction
  - Sketch multiplication, division
- Overview of ALU (arithmetic logic unit)
- Floating point numbers
  - Representation
  - Arithmetic operations
  - MIPS instructions

Bits

- What do these two binary strings represent?
  0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0101
  0000 0000 0000 0000 0000 0000 0000 0000 0000 0011 0110

- Bits are...

- ______________ define relationship between
  ___________ and ___________
Bits as Numbers: Complications

- Numbers are finite
- Fractions and real numbers
- Negative numbers
- MIPS typically uses 32 bits for a number
  - But we’ll often demonstrate with fewer for simplicity
- MSB vs LSB

Integers: Possible 3-bit Representations of 2 and -2

1. Unsigned
2. Sign and Magnitude
3. One’s Complement
4. Two’s Complement

Example Representations

<table>
<thead>
<tr>
<th>Unsigned</th>
<th>Sign Mag.</th>
<th>One’s Comp.</th>
<th>Two’s Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>000 = +0</td>
<td>000 = +0</td>
<td>000 = +0</td>
<td>000 = +0</td>
</tr>
<tr>
<td>001 = +1</td>
<td>001 = +1</td>
<td>001 = +1</td>
<td>001 = +1</td>
</tr>
<tr>
<td>010 = +2</td>
<td>010 = +2</td>
<td>010 = +2</td>
<td>010 = +2</td>
</tr>
<tr>
<td>011 = +3</td>
<td>011 = +3</td>
<td>011 = +3</td>
<td>011 = +3</td>
</tr>
<tr>
<td>100 = +4</td>
<td>100 = +0</td>
<td>100 = -3</td>
<td>100 = -4</td>
</tr>
<tr>
<td>101 = +5</td>
<td>101 = -1</td>
<td>101 = -2</td>
<td>101 = -3</td>
</tr>
<tr>
<td>110 = +6</td>
<td>110 = -2</td>
<td>110 = -1</td>
<td>110 = -2</td>
</tr>
<tr>
<td>111 = +7</td>
<td>111 = -3</td>
<td>111 = -0</td>
<td>111 = -1</td>
</tr>
</tbody>
</table>

Two’s Complement Operations

- Negating a two’s complement number: invert all bits and add 1
- But must write down leading zero bits if there!
- Example:
  - Express -6₁₀ in 8-bit binary 2’s complement:
MIPS

- MIPS signed numbers use...

- 32 bit signed numbers:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 0000 0000 0000 0000 0000 0000 0000</td>
<td>0</td>
</tr>
<tr>
<td>0000 0000 0000 0000 0000 0000 0000 0000</td>
<td>+1</td>
</tr>
<tr>
<td>0000 0000 0000 0000 0000 0000 0000 0000</td>
<td>+2</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>0111 1111 1111 1111 1111 1111 1111 1110</td>
<td>+2,147,483,646</td>
</tr>
<tr>
<td>0111 1111 1111 1111 1111 1111 1111 1111</td>
<td>+2,147,483,647</td>
</tr>
<tr>
<td>1000 0000 0000 0000 0000 0000 0000 0000</td>
<td>-2,147,483,648</td>
</tr>
<tr>
<td>1000 0000 0000 0000 0000 0000 0000 0000</td>
<td>-2,147,483,647</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>1111 1111 1111 1111 1111 1111 1111 1101</td>
<td>-3</td>
</tr>
<tr>
<td>1111 1111 1111 1111 1111 1111 1111 1110</td>
<td>-2</td>
</tr>
<tr>
<td>1111 1111 1111 1111 1111 1111 1111 1111</td>
<td>-1</td>
</tr>
</tbody>
</table>

Signed vs. unsigned numbers

- Some values don’t make sense as negative numbers

- MIPS allows values to be signed or unsigned

- Different instructions to deal with each case
  - add vs. addu
  - lb vs. lbu
  - addi vs. addiu
  - slli vs. slliu

- Usually, the unsigned version will not _______________

- Exception:

Two’s Complement Operations

- Converting n bit numbers into numbers with more than n bits:
  - MIPS 16 bit immediate gets converted to 32 bits for arithmetic
  - copy the most significant bit (the sign bit) into the other bits

- 4 -> 8 bit example:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010</td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td></td>
</tr>
</tbody>
</table>

  - This is called

Addition & Subtraction

- Just like in grade school (carry/borrow 1s)

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td></td>
</tr>
<tr>
<td>+ 0101</td>
<td>- 0110</td>
</tr>
</tbody>
</table>

- Easier way to subtract?
### Addition & Subtraction

- Another example:
  
  $\begin{array}{c}
  0111 \\
  + 0001 \\
  \hline
  1000
  \end{array}$

### Detecting Overflow

- Overflow -- result too large for finite computer word
- Is overflow possible if adding...
  - a positive and a negative number?
  - two positive numbers?
  - two negative numbers?

- Subtraction:
  - Invert the second number to test
  - So no overflow possible when signs are...

### Effects of Overflow

- An exception (interrupt) occurs
  - Control jumps to predefined address for exception
  - Interrupted address is saved for possible resumption
- Details based on software system / language
  - Example: flight control vs. homework assignment
  - C always ignores overflow
- Don't always want to detect overflow
  - "Unsigned" arithmetic instructions will ignore:
    - `addu`, `addiu`, `subu`

### Summary: Advantages of Two’s Complement

- How to negate a number?
- How many zeros?
- How add positive and negative numbers?
- Consequently, essentially all modern computers use this