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

1

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**

3

ADMIN

- **Reading**
 - **Read 3.1, 3.2, 3.3, 3.4**
 - **Skim 3.5**
 - **Read 3.6 (Floating point – skim details on addition, multiplication, rounding, representation – but pay attention to MIPS instructions for load/store/computations)**
 - **Read 3.8**

2

Bits

- **What do these two binary strings represent?**
0000 0000 0000 0000 0000 0000 0001 0101
0000 0001 0010 0011 0100 0101 0110 0111

- **Bits are...**

- _____ define relationship between
_____ and _____

4

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

5

Example Representations

Unsigned	Sign Mag.	One's Comp.	Two's Comp.
000 = +0	000 = +0	000 = +0	000 = +0
001 = +1	001 = +1	001 = +1	001 = +1
010 = +2	010 = +2	010 = +2	010 = +2
011 = +3	011 = +3	011 = +3	011 = +3
100 = +4	100 = -0	100 = -3	100 = -4
101 = +5	101 = -1	101 = -2	101 = -3
110 = +6	110 = -2	110 = -1	110 = -2
111 = +7	111 = -3	111 = -0	111 = -1

7

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

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

6

Two's Complement Operations

EX: 3-1 ...

- Negating a two's complement number: invert all bits and add 1
- But must write down leading zero bits if there!
- Example:
 - Express -6_{10} in 8-bit binary 2's complement:

8

MIPS

- MIPS signed numbers use...

- 32 bit signed numbers:

```

0000 0000 0000 0000 0000 0000 0000 0000two = 0ten
0000 0000 0000 0000 0000 0000 0000 0001two = + 1ten
0000 0000 0000 0000 0000 0000 0000 0010two = + 2ten
...
0111 1111 1111 1111 1111 1111 1111 1110two = + 2,147,483,646ten
0111 1111 1111 1111 1111 1111 1111 1111two = + 2,147,483,647ten
1000 0000 0000 0000 0000 0000 0000 0000two = - 2,147,483,648ten
1000 0000 0000 0000 0000 0000 0000 0001two = - 2,147,483,647ten
1000 0000 0000 0000 0000 0000 0000 0010two = - 2,147,483,646ten
...
1111 1111 1111 1111 1111 1111 1111 1101two = - 3ten
1111 1111 1111 1111 1111 1111 1111 1110two = - 2ten
1111 1111 1111 1111 1111 1111 1111 1111two = - 1ten

```

9

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:

0010 ->

1010 ->

- This is called

10

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
 - slti vs. sltiu

- *Usually*, the unsigned version will not _____

- Exception:

11

Addition & Subtraction

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

```

      0001      0111      0110
+ 0101      - 0110      - 0101
-----

```

- Easier way to subtract?

12

Addition & Subtraction

- Another example:

```
  0111
+ 0001
-----
```

13

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...

14

Effects of Overflow

EX: 3-11 ...

- 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

15

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

16