

---

**SI232**  
**Slide Set #9:**  
**Computer Arithmetic (Chapter 3)**

1

---

**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 – but pay attention to representation and MIPS instructions)
  - Read 3.8

3

---

**RECAP – HOMEWORK RESPONSIBILITIES**

- I will
  - Ensure problems from text are clear, or write new ones
  - Answer your questions promptly (start sending them!)
  - Continue to be available for EI
  - Provide a stapler
- You should
  - Email/EI questions if you are confused or need help
  - Read the directions carefully
  - Expect to spend some time: for learning, not just rehash of class
  - Start early
  - (optional) Collaborate (though not for projects)
- Suggestions
  - Review your notes from class – sometime the same day
  - Practice and understand problems/exercises

2

---

**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

4

## 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 \_\_\_\_\_

5

## 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

6

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

---

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

7

## 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

8

## 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  $-6_{10}$  in 8-bit binary 2's complement:

9

## Exercise #1

---

- Assume we have 4 bits. Convert the given decimal numbers to the stated binary representations.

	5	-7
Unsigned		
Sign Magnitude		
One's Comp.		
Two's Comp.		

10

## Exercise #2

---

- Convert the given decimal numbers to the stated binary representations.

	-3 (using 4 bits)	-3 (using 6 bits)
Sign Magnitude		
One's Comp		
Two's Comp.		

11

## Exercise #3

---

- Assume the following is in binary two's complement form. What do they represent in decimal?

001011

111011

- Now negate these numbers and show the new binary form:

$-(001011) =$

$-(111011) =$

12

## Exercise #4 – Stretch

- Given N bits, what is the largest and smallest number that each of the following can represent?

	Min	Max
Unsigned		
Sign Magnitude		
Ones Complement		
Twos Complement		

13

## 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

15

## MIPS

- MIPS signed numbers use...

- 32 bit signed numbers:

```

0000 0000 0000 0000 0000 0000 0000 0000 0000two = 0ten
0000 0000 0000 0000 0000 0000 0000 0000 0001two = + 1ten
0000 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
    
```

14

## 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
  - sli vs. sliu
- Usually, the unsigned version will not \_\_\_\_\_
- Exception:

16

## Addition & Subtraction

---

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

0001	0111	0110
+ 0101	- 0110	- 0101

- Easier way to subtract?

17

## Addition & Subtraction

---

- Another example:

0111
+ 0001

18

## 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...

19

## 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

20

## 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