Outline of Lecture

- Representing Numbers
- Negative Numbers
- Simple Binary arithmetic

Arithmetic for Computers

Goal: ***How does the computer hardware perform add, subtract, multiply, or divide numbers (big portion of any processor design)?***

- **Before we determine how the computer hardware performs arithmetic operations, we should determine how numbers are *represented* inside the computer.**

Basic Issues

- How are positive and negative numbers represented?
- What is the largest and smallest number that can be represented by a computer word?
- What happens if an operation creates a number bigger or smaller than can be represented?
- How to represent fractions and real numbers?

Numbers

- **In any number base, the value of the *i*th digit *d* is:**

$$d \times base^i$$

Example:

$1011_{two} = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) = 11_{ten}$

- **The above number is placed in a MIPS word as follows:**

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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

Bit 31 is the *most significant bit* and bit 0 is *the least significant bit*.

- **Since the MIPS word is 32 bits, it can represent 2^{32} different 32-bit patterns.**

- If we use a MIPS to just represent positive numbers, they will be in the range from 0 to $2^{32} - 1$.
- However, computer programs need both positive numbers as well as ***negative*** numbers. Thus, the 32 bits should represent an equal number of positive and negative numbers.

Negative Numbers

- **How do we represent negative numbers?**

  i.e., which bit patterns will represent which numbers?

Possible Representations

Sign Magnitude:	One's Compliments	Two's Complement
000 = +0	000 = +0	000 = +0
001 = +1	001 = +1	001 = +1
010 = +2	010 = +2	010 = +2
011 = +3	011 = +3	011 = +3
100 = -0	100 = -3	100 = -4
101 = -1	101 = -2	101 = -3
110 = -2	110 = -1	110 = -2
111 = -3	111 = -0	111 = -1

- **The most significant bit shows the sign of the number.**

MIPS uses 2's compliments

- **32 bit signed numbers:**

0000 0000 0000 0000 0000 0000 0000 0000$_{two}$ = 0 $_{ten}$

0000 0000 0000 0000 0000 0000 0000 0001$_{two}$ = +1$_{ten}$

0000 0000 0000 0000 0000 0000 0000 0010$_{two}$ = +2$_{ten}$

.....

maxint

0111 1111 1111 1111 1111 1111 1111 1110$_{two}$=+2,147,483,646$_{ten}$

0111 1111 1111 1111 1111 1111 1111 1111$_{two}$=+2,147,483,647$_{ten}$

1000 0000 0000 0000 0000 0000 0000 0000$_{two}$ = -2,147,483,648$_{ten}$

1000 0000 0000 0000 0000 0000 0000 0001$_{two}$= -2,147,483,647$_{ten}$

1000 0000 0000 0000 0000 0000 0000 0010$_{two}$= -2,147,483,646$_{ten}$

minint

1111 1111 1111 1111 1111 1111 1111 1101$_{two}$ = -3ten

1111 1111 1111 1111 1111 1111 1111 1110$_{two}$ = -2ten

1111 1111 1111 1111 1111 1111 1111 1111$_{two}$ = -1 ten

- **This representation for signed binary numbers is called *two's complement* representation.**

TWO'S complement Operations

- **Negating a two's complement number:**

  To *negate* a binary number represented in two's complement, simply invert every bit and then add 1 to the result.

Remember: Negate and Invert are quite different

- Example: Negate 000101

Invert of 000101 is 111010

111010 + 1 = 111011

- Example: Negate 1110

Invert of 1110 is 0001

0001 + 1 = 0010

Example

Negate 2_{ten}, and then check the result by negating -2_{ten}.

Answer

2_{ten} = 0000 0000 0000 0000 0000 0000 0000 0010_{two}

Negating this number by inverting the bits and adding 1:

1111 1111 1111 1111 1111 1111 1111 1101_{two}

\+ 1_{two}

= 1111 1111 1111 1111 1111 1111 1111 1110_{two}

= -2_{ten}

Going the other direction (inverting -2 and adding 1):

0000 0000 0000 0000 0000 0000 0000 0001_{two}

\+ 1_{two}

= 0000 0000 0000 0000 0000 0000 0000 0010_{two}

Another Way

- **In two's complement representation, all negative numbers have a 1 in the most significant bit - called *sign* bit - it is easy for the hardware to test whether a number is positive or negative.**
- **Any number in two's complement can be read as follows (*xi* means the *i*th bit of the number):**

$$(x31 \times -2^{31}) + (x30 \times 2^{30}) + \ldots + (x1 \times 2^{1}) + (x0 \times 2^{0})$$

Example

$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100_{two} = (1 \times -2^{31}) + (1 \times 2^{30}) + \ldots + (1 \times 2^{2}) + (0 \times 2^{1}) + (0 \times 2^{0}) = -4_{ten}$

Signed and Unsigned Numbers

- The two's complement representation is a ***signed*** integer representation. If we want a representation to deal just with positive integers, then it is called an ***unsigned*** integer representation.
- In MIPS, `slt` and `slti` instructions work with signed integers (as do other instructions).
- Unsigned integers are compared using `sltu` (`set on less than unsigned`) and `sltiu` (`set on less than immediate unsigned`).

Example

Suppose register $16 has the binary number

`1111 1111 1111 1111 1111 1111 1111 1111`$_{two}$

and that register $17 has the binary number

`0000 0000 0000 0000 0000 0000 0000 0001`$_{two}$

what are the values of registers $8 and $9 after these two instructions?

```
slt $8, $16, $17     # signed comparison

sltu $9, $16, $17  # unsigned comparison
```

Answer

Register $8 has the value 1 since $-1_{ten} < 1_{ten}$

Register $9 has the value 0 since $4,294,967,295_{ten} > 1_{ten}$.

Sign Extension

- Converting n- bit numbers into numbers with more than n bits:
  - MIPS 8 bit constants are converted to 32 bits for arithmetic automatically
  - replicate the most significant bit (the sign bit)

    0010 **0000 0**010

    1010 1111 1010
  - known as "sign extension" (lbu vs. lb)
  - lbu loads an unsigned byte
  - lb loads a signed byte and sign extends it.

Exercises

- What does 111101 represents?

  Ans: Can’t answer this without knowing what number system is being used.

- What does 111101 represents in 2’s complement?

  Ans: -3. (Negate of 111101 is 000011 = +3)

- What does 111101 represents as an unsigned integer?

  Ans: 61.

- What does 111101 represents as an ASCII character?

  Ans: go find out yourself from page 142.

Exercises

- How to represent -5 ?

  Ans: can't answer, rep. scheme is not specified.

- How to represent -5 with 8 bits in 2's complement?

  Ans: +5 is 00000101 so, negate to get 11111011

- How to represent -5 with 7 -bit 2's complement?

  Ans: +5 is 0000101 so, negate to get 1111011

- How to represent -5 with 6 -bit 2's complement?

  Ans: +5 is 000101 so, negate to get 111011

- How to represent -5 with 3-bit 2's compliment?

  Ans: too large to represent !

Summary

- All ***signed*** numbers in MIPS are represented using two's complement.
- There are other signed representation of binary numbers such as ***1's complement*** and ***sign and magnitude*** representations. But, as we will see, they have their drawbacks when compared to two's complement.
- If we are dealing with an ***unsigned*** number, then the MIPS instructions have to specify that (e.g.; `addu, subu, addiu, sltu, sltiu`).