Computer Organization

605.204

Module Two

Part One

Negative and other Numbers

Summer 2014

Module Two

- Part One
- This week:
- Computer Numbers
- Integers and Integer Arithmetic
- Floating Point Numbers and Calculations

Computer Numbers

- Digits are just digits (no inherent meaning)
  - conventions define relationship between digits and numbers
- Decimal: 0 1 2 3 4 5 6 7 8 9 10 11 …
- Binary: 0 1 10 11 100 101 110 111 1000 1001 ...
- Octal: 0 1 2 3 4 5 6 7 10 11 12 …
- Hexadecimal: 0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 …

Computer Numbers

- Of course it gets more complicated:
  - a finite number of digits
  - fractions and real numbers
  - negative value numbers
- How do we represent number values?

Which bit patterns will represent which numbers?

Sign	Number value

Possible Representations

- INTEGERS
- Sign Magnitude
- One’s Complement
- Two's Complement
- Issues: Balance, Number of zeros, Ease of operations
- Is one representation better than the others?

Sign Magnitude

- Sign + Magnitude = value

Sign	Magnitude		value
0	00	=	+0
0	01	=	+1
0	10	=	+2
0	11	=	+3
1	00	=	- 0
1	01	=	- 1
1	10	=	- 2
1	11	=	- 3

Sign Magnitude:

000 = +0
001 = +1
010 = +2
011 = +3
100 = -0
101 = -1
110 = -2
111 = -3

Issues: Balance, Number of zeros

One’s Complement

- One's Complement

000 = +0
001 = +1
010 = +2
011 = +3

Invert the bit pattern for negative

111 = - 0
110 = - 1
101 = - 2
100 = - 3

One's Complement

000 = +0
001 = +1
010 = +2
011 = +3

111 = -0
110 = -1
101 = -2
100 = -3

Issues: Balance, Number of zeros

Two’s Complement

- Two's Complement

000 = +0
001 = +1
010 = +2
011 = +3

Invert the bit pattern, add 1 for negative

111 = -1
110 = -2
101 = -3
100 = -4

Two's Complement

000 = +0
001 = +1
010 = +2
011 = +3

111 = -1
110 = -2
101 = -3
100 = -4

Issues: Balance, Number of zeros

Possible Representations

- Sign Magnitude
- One’s Complement
- Two's Complement

- All use the left most bit as the sign indicator.

- Issues: Balance, Number of zeros,

- Which representation is better? Why?

- What about: Ease of operations

Ease of operations

- Addition is easy:
  - Add the digits and carry to the next column
  - Continue from right to left
- The challenge: Subtract
  - How do you make the calculation?
  - Well … just ‘take from’ the digit, and ‘borrow’ if necessary.
  - Question: How do you make a computer ‘take from’ or ‘borrow’ ?
- Solution: Use algebra
  - Change the sign and add.

Ease of operations

- Sign Magnitude

```
  0110    + 6
  1011    - 3   add
 10001     ??
```

Not so easy

- One's complement
- 'End around carry' addition

```
  0110    + 6
  1100    - 3   add
 10010
     1    + carry
  0011
```

Ease of operations

- Two's complement
- 'Carry ignore' addition

```
  0110    + 6
  1101    - 3   add
 1 0011
```

- Summary:
- Today
- All modern processors use two's complement integer representation.
- Ease of operation is the reason.

Two's Complement Values

- 32 bit signed numbers:

$$
\begin{array}{l}
0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten} \\
0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = +\ 1_{ten} \\
0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = +\ 2_{ten} \\
\ldots \\
0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = +\ 2{,}147{,}483{,}646_{ten} \\
0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = +\ 2{,}147{,}483{,}647_{ten} \\
\boxed{1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = -\ 2{,}147{,}483{,}648_{ten}} \\
1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = -\ 2{,}147{,}483{,}647_{ten} \\
1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_{two} = -\ 2{,}147{,}483{,}646_{ten} \\
\ldots \\
1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_{two} = -\ 3_{ten} \\
1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_{two} = -\ 2_{ten} \\
1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = -\ 1_{ten}
\end{array}
$$

Two's Complement Number Value

- 8 bit signed numbers
- $\boxed{-1 \mathrm{x} b_7 \mathrm{x} 128} + b_6 \mathrm{x} 64 + b_5 \mathrm{x} 32 + b_4 \mathrm{x} 16 + b_3 \mathrm{x} 8 + b_2 \mathrm{x} 4 + b_1 \mathrm{x} 2 + b_0$
- 32 bit signed numbers:
- $\boxed{-1 \mathrm{x} b_{31} \mathrm{x} 2^{31}} + b_{30} \mathrm{x} 2^{30} + b_{29} \mathrm{x} 2^{29} + \ldots + b_3 \mathrm{x} 2^3 + b_2 \mathrm{x} 2^2 + b_1 \mathrm{x} 2^1 + b_0 \mathrm{x} 2^0$

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = 0_{ten}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_{two} = +\ 1_{ten}$
...

$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = +\ 2{,}147{,}483{,}647_{ten}$
$\boxed{1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_{two} = -\ 2{,}147{,}483{,}648_{ten}}$
...
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_{two} = -\ 1_{ten}$

Two's Complement Operations

- Negating a two's complement number: invert all bits and add 1
  - Remember: “negate” and “invert” are quite different!
- 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

    0000 0010 <- 0010 = 2_{10}

    1111 1010 <- 1010 = -6_{10}
  - "sign extension"

Summary

- Computer numbers fixed number of digits.
- Digits by themselves have no meaning.
- Patterns are required to assign values.
- Two’s complement for integers.

- Next: Integer Arithmetic