


5-Sep-06 (1)




## CSC2510 - Computer Organization

### Lecture 2: Numbers and Arithmetic Operations

Philip Leong


5-Sep-06 (2)



## Counting to 1

- Binary numbers (0, 1) are used in computers as they are easily represented as off/on electrical signals
- Different number systems are used in computers
- Numbers represented as binary vectors
- $B = b_{n-1} \dots b_1 b_0$
- Unsigned numbers are in range 0 to  $2^{n-1}$  and are represented by  $V(B) = b_{n-1} \times 2^{n-1} + \dots + b_1 \times 2^1 + b_0 \times 2^0$
- MSB=Most significant bit (leftmost digit in a binary vector)
- LSB=Least significant bit (rightmost digit in a binary vector)


5-Sep-06 (3)



## Negative Numbers

- Sign-and-magnitude
  - Most significant bit determines sign, remaining unsigned bits represent magnitude
- 1's complement
  - Most significant bit determines sign. To change sign from unsigned to negative, invert all the bits
- 2's complement
  - Most significant bit determines sign. To change sign from unsigned to negative, invert all the bits and add 1
  - This is equivalent to subtracting the positive number from  $2^n$


5-Sep-06 (4)



## Number Systems

| B       | Values represented                      |                |                |
|---------|---|----------------|----------------|
|         | Sign and magnitude<br>$b_3 b_2 b_1 b_0$ | 1's complement | 2's complement |
| 0 1 1 1 | +7                                      | +7             | +7             |
| 0 1 1 0 | +6                                      | +6             | +6             |
| 0 1 0 1 | +5                                      | +5             | +5             |
| 0 1 0 0 | +4                                      | +4             | +4             |
| 0 0 1 1 | +3                                      | +3             | +3             |
| 0 0 1 0 | +2                                      | +2             | +2             |
| 0 0 0 1 | +1                                      | +1             | +1             |
| 0 0 0 0 | +0                                      | +0             | +0             |
| 1 0 0 0 | -0                                      | -7             | -8             |
| 1 0 0 1 | -1                                      | -6             | -7             |
| 1 0 1 0 | -2                                      | -5             | -6             |
| 1 0 1 1 | -3                                      | -4             | -5             |
| 1 1 0 0 | -4                                      | -3             | -4             |
| 1 1 0 1 | -5                                      | -2             | -3             |
| 1 1 1 0 | -6                                      | -1             | -2             |
| 1 1 1 1 | -7                                      | -0             | -1             |

5-Sep-06 (5)




## Addition (1-bit)

|       |       |       |       |
|-------|-------|-------|-------|
| 0     | 1     | 0     | 1     |
| + 0   | + 0   | + 1   | + 1   |
| ----- | ----- | ----- | ----- |
| 0     | 1     | 1     | 1 0   |

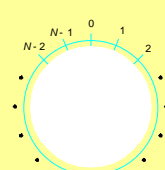
↑  
Carry-out

5-Sep-06 (6)

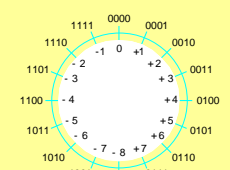


## 2's Complement

- 2's complement numbers actually make sense since they follow normal modulo arithmetic except when they overflow
- Range is  $-2^{n-1}$  to  $2^{n-1}-1$




(a) Circle representation of integers mod  $N$



(b) Mod 16 system for 2's-complement numbers


5-Sep-06 (7)



## Add/sub

- X+Y – use 1-bit addition propagating carry to the next most significant bit
- X-Y – add X to the 2's complement of Y


5-Sep-06 (8)



## Add/Sub (2's comp)

|  |   |
|--|---|
| <p>(a) <math display="block">\begin{array}{r} 0010 \\ +0011 \\ \hline 0101 \end{array}</math></p> <p>(c) <math display="block">\begin{array}{r} 1011 \\ +1110 \\ \hline 10001 \end{array}</math></p> <p>(e) <math display="block">\begin{array}{r} 1101 \\ -1001 \\ \hline 0100 \end{array}</math></p> <p>(f) <math display="block">\begin{array}{r} 0010 \\ -0100 \\ \hline 0110 \end{array}</math></p> <p>(g) <math display="block">\begin{array}{r} 0110 \\ -0011 \\ \hline 0101 \end{array}</math></p> <p>(h) <math display="block">\begin{array}{r} 1001 \\ -1011 \\ \hline 0110 \end{array}</math></p> <p>(i) <math display="block">\begin{array}{r} 1001 \\ -0001 \\ \hline 1000 \end{array}</math></p> <p>(j) <math display="block">\begin{array}{r} 0010 \\ -1101 \\ \hline 0101 \end{array}</math></p> | <p>(b) <math display="block">\begin{array}{r} 0100 \\ +1010 \\ \hline 1110 \end{array}</math></p> <p>(d) <math display="block">\begin{array}{r} 0111 \\ +1101 \\ \hline 0100 \end{array}</math></p> <p><math>\Rightarrow</math></p> <p><math display="block">\begin{array}{r} 1101 \\ +0111 \\ \hline 0100 \end{array}</math></p> <p><math>\Rightarrow</math></p> <p><math display="block">\begin{array}{r} 0010 \\ +1100 \\ \hline 1110 \end{array}</math></p> <p><math>\Rightarrow</math></p> <p><math display="block">\begin{array}{r} 0110 \\ +1101 \\ \hline 0011 \end{array}</math></p> <p><math>\Rightarrow</math></p> <p><math display="block">\begin{array}{r} 1001 \\ +0101 \\ \hline 1110 \end{array}</math></p> <p><math>\Rightarrow</math></p> <p><math display="block">\begin{array}{r} 1001 \\ +1111 \\ \hline 0000 \end{array}</math></p> <p><math>\Rightarrow</math></p> <p><math display="block">\begin{array}{r} 0010 \\ +0011 \\ \hline 0101 \end{array}</math></p> |
|--|---|


5-Sep-06 (9)



## Sign Extension

- Suppose I have a 4-bit 2's complement number and I want to make it into an 8-bit number
  - Positive number – add 0's to LHS e.g. 0111 -> 00000111
  - Negative number – add 1's to LHS e.g. 1010 -> 11111010
  - c.f. circle representation


5-Sep-06 (10)



## Overflow

- In 2's complement arithmetic
  - addition of opposite sign numbers never overflow
  - If the numbers are the same sign and the result is the opposite sign, overflow has occurred
  - E.g. 0111+0100=1011 (but 1011 is -5)
- Unsigned
  - Carry out signals an overflow

5-Sep-06 (11)




## Characters

- Typically represented by 8-bit numbers

| ASCII Code: Character to Binary |           |   |           |
|---------------------------------|-----------|---|-----------|
| 0                               | 0011 0000 | 0 | 0000 1011 |
| 1                               | 0011 0001 | 1 | 0000 1010 |
| 2                               | 0011 0010 | 2 | 0000 1001 |
| 3                               | 0011 0011 | 3 | 0000 1000 |
| 4                               | 0011 0100 | 4 | 0000 0111 |
| 5                               | 0011 0101 | 5 | 0000 0110 |
| 6                               | 0011 0110 | 6 | 0000 0101 |
| 7                               | 0011 0111 | 7 | 0000 0100 |
| 8                               | 0011 1000 | 8 | 0000 0011 |
| 9                               | 0011 1001 | 9 | 0000 0010 |
| A                               | 0011 1010 | A | 0000 0001 |
| B                               | 0011 1011 | B | 0000 0000 |
| C                               | 0010 1000 | c | 0001 1011 |
| D                               | 0010 1001 | d | 0001 1010 |
| E                               | 0010 1010 | e | 0001 1001 |
| F                               | 0010 1011 | f | 0001 1000 |
| G                               | 0010 1100 | g | 0001 0111 |
| H                               | 0010 1101 | h | 0001 0110 |
| I                               | 0010 1110 | i | 0001 0101 |
| J                               | 0010 1111 | j | 0001 0100 |
| K                               | 0010 1000 | k | 0001 0011 |
| L                               | 0010 1001 | l | 0001 0010 |
| M                               | 0010 1010 | m | 0001 0001 |
| N                               | 0010 1011 | n | 0001 0000 |
| O                               | 0001 1000 | o | 0010 1011 |
| P                               | 0001 1001 | p | 0010 1010 |
| Q                               | 0001 1010 | q | 0010 1001 |
| R                               | 0001 1011 | r | 0010 1000 |
| S                               | 0001 1100 | s | 0010 0111 |
| T                               | 0001 1101 | t | 0010 0110 |
| U                               | 0001 1110 | u | 0010 0101 |
| V                               | 0001 1111 | v | 0010 0100 |
| W                               | 0000 1000 | w | 0011 1011 |
| X                               | 0000 1001 | x | 0011 1010 |
| Y                               | 0000 1010 | y | 0011 1001 |
| Z                               | 0000 1011 | z | 0011 1000 |
| [                               | 0000 1100 | [ | 0011 0111 |
| \                               | 0000 1101 | \ | 0011 0110 |
| ]                               | 0000 1110 | ] | 0011 0101 |
| ^                               | 0000 1111 | ^ | 0011 0100 |
| _                               | 0000 1000 | _ | 0011 0011 |
| `                               | 0000 1001 | ` | 0011 0010 |
| {                               | 0000 1010 | { | 0011 0001 |
| }                               | 0000 1011 | } | 0011 0000 |
| space                           | 0010 0000 |   |           |

5-Sep-06 (12)



## Quiz

- Assuming 4-bit numbers
  - What is the binary for -2?
  - Calculate 2+3
  - Calculate -2-3
  - Calculate 5+5
- Assuming 5-bit numbers
  - What is the largest 2's complement number?
  - What is the smallest 2's complement number?
- Convert 56 to unsigned binary
- What is the decimal value of 10110101 in 2's complement? What is the unsigned value of the same binary number?