Binary Representation

CS 1428
Fall 2011

Jill Seaman

Lecture 8

Binary Number Representation

- Computer memory is a sequence of switches, on or off: ON = 1, OFF = 0.
- All values are stored in the computer as a series of ones and zeroes (binary format).
- All data values have to be converted to and from a binary format.

Binary number system

- The binary number system has 2 digits, 0 and 1.
- We refer to these digits as “bits”
- In any number system, the value of the i^{th} digit d is: $d \times base^{i}$
- Consider the number 57 in the decimal system.

$$57 = 5 \times 10^{1} + 7 \times 10^{0}$$
$$= 5 \times 10 + 7 \times 1$$
$$= 50 + 7 = 57$$

Converting binary to decimal

- the value of the i^{th} digit b is: $b \times 2^{i}$

100110

$= 1 \times 2^{5} + 0 \times 2^{4} + 0 \times 2^{3} + 1 \times 2^{2} + 1 \times 2^{1} + 0 \times 2^{0}$

$= 1 \times 32 + 0 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 0 \times 1$

$= 32 + 0 + 0 + 4 + 2 + 0$

$= 38$

Converting binary to decimal

- Or, you can consider the "value" of each position in the binary number:

```
32  16  8  4  2  1   <-- the value of each position
 1   0  0  1  1  0   <-- the binary number
```

- If there is a one under the value of the position, add that number to the total:

32 + 4 + 2 = 38

Converting decimal to binary

- If the number is divisible by 2,
  - write 0
  - divide by 2 to get the next number
- else
  - write 1
  - subtract 1 from the number
  - divide by 2 to get the next number
- Repeat with the resulting number, until you get to 0
- Add the 1 or 0 to the left of the previous value

Converting decimal to binary

- Example:

```
40
40          is divisible by 2:       write   0:         0
40/2 = 20   is divisible by 2:       write   0:        00
20/2 = 10   is divisible by 2:       write   0:       000
10/2 = 5    is NOT divisible by 2: write   1:      1000
(5-1)/2 = 2 is divisible by 2:         write 0:     01000
2/2 = 1     is NOT divisible by 2:  write 1:    101000
(1-1)/2 = 0 DONE
```

- 40 decimal is 101000 in binary

Binary Arithmetic

- Just like decimal arithmetic:

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10  (carry the 1)
```

- Numbers with multiple digits:

```
0 0 1 1 0
0 1 1 0 1
-----------
1 0 0 1 1
```

Negative numbers
sign and magnitude

- Let's say we have only 4 bits to represent positive and negative numbers
- Let's use the leftmost bit for the sign, 1 is negative, 0 is positive
- What number is:
  - 0 0 0 0
  - 1 0 0 1
  - 1 0 0 0

Negative numbers
sign and magnitude

- There would be 2 values for 0
- Also, addition is broken
  - 3 + (-3) = ?

```
0011
1011
------
1110  =  -6?
```

2's complement representation

- Leftmost bit is still the sign, but the rest of the negative number is converted differently
- only one representation for 0
- binary arithmetic still works!
- one extra negative number

Converting decimal to 2's complement

- If the number is positive:
  - convert decimal to binary in the usual way
- If the number is negative:
  - get the binary representation for the absolute value of the number in the usual way
  - flip all the bits
  - add 1 to the complement
- example:

```
-3
0011   // 4-bit binary for absolute value of -3
1100   // all bits flipped
1101   // 1 added to the complement
```

Converting 2's complement to decimal

- If the left-most bit is 0 then convert the number in the usual way
- Else
  - Subtract 1
  - Flip all bits
  - Convert to decimal in the usual way
  - Affix a minus sign
- Example:

```
1010 // 2's complement binary
1001 // 1 subtracted
0110  // bits flipped
-6       // affixed the negative sign
```


2's complement arithmetic

- Let's add 3 and -3 again

```
0011
1101
------
0000
```

Decimal/sign+magnitude/2's complement

7	0111	0111
6	0110	0110
5	0101	0101
4	0100	0100
3	0011	0011
2	0010	0010
1	0001	0001
0	0000	0000
0	1000	------
-1	1001	1111
-2	1010	1110
-3	1011	1101
-4	1100	1100
-5	1101	1011
-6	1110	1010
-7	1111	1001
-8	------	1000