Data Representation

Using binary numbers to represent information

Data Representation

- Goal: Store numbers, characters, sets, database records in the computer.
- What we got: Circuit that stores 2 voltages, one for logic 0 (0 volts) and one for logic 1 (ex: 3.3 volts).

Storing Information

Value	Representation
H	0
T	1

Value	Representation
False	0
True	1

Value	Representation
1e-4	0
5	1

- Use more bits for more items
- Three bits can represent 8 things: 000,001,…,111
- N bits can represent 2^N things

N bits	Can represent	Which is approximately
8	256	256
16	65,536	65 thousand (64K where K=1024)
32	4,294,967,296	4 billion
64	$1.8446\ldots\times10^{19}$	20 billion billion

Storing Information

Byte is a unit of information. Remember 1 byte = 8 bits

Most computers today use:

Type	**# of bits**
Character	8-16
Integers	32-64
Addresses	32-64

Integer Representation

Usual answers:

1. Represent 0 and consecutive positive integers
   - Unsigned integers
2. Represent positive and negative integers
   - Signed magnitude
   - One's complement
   - Two's complement

Unsigned and two's complement the most common

Unsigned Integers

- Integer represented is binary value of bits:

0000 -> 0, 0001 -> 1, 0010 -> 2, …

- Encodes only positive values and zero
- Range: 0 to $2^n - 1$, for n bits

Unsigned Integers

If we have 4 bit numbers:

To find range make n = 4. Thus $2^4 – 1$ is 15
Thus the values possible are 0 to 15
[0:15] = 16 different numbers

7 would be 0111

17 not represent-able
-3 not represent-able

For 32 bits:

Range is 0 to 2^{32} - 1 = [0: 4,294,967,295]
Which is 4,294,967,296 different numbers

Signed Magnitude Integers

- A human readable way of getting both positive and negative integers.

Signed Magnitude Integers

Representation:

- Use 1 bit of integer to represent the sign of the integer.

  - Sign bit is msb: 0 is “+”, 1 is ”–”

- Rest of the integer is a magnitude, with same encoding as unsigned integers.

- To get the additive inverse of a number, just flip (invert, complement) the sign bit.
- Range: $-(2^{n-1} – 1)$ to $2^{n-1} -1$

Signed Magnitude - Example

If 4 bits then range is:

$-2^3 + 1$ to $2^3 - 1$

which is -7 to +7

Given only 4 **bits** to represent signed magnitude, what integer the following binary numbers represent:

- 0101 is 0 101 = + 5
- -3 is ? 1 011 = 1011
- +12 is ? • Not possible range is -7 …. +7
- [-7,…, -1, 0, +1,…,+7] = 7 + 1 + 7 = 15 < 16 =

- What problems does this cause?

Signed Magnitude - Example

If 4 bits then range is:

$-2^3 + 1$ to $2^3 – 1$

which is -7 to +7

Questions:

- 0101 is 0 101 = + 5

- -3 is ? is 1 011 = 1011
- +12 is ? Not possible range is -7 …. +7

- [-7,…, -1, 0, +1,…,+7] = 7 + 1 + 7 = 15 < 16 =

- What problems does this cause?

  0 = 1 000 (negative zero)

  0 = 0 000 (positive zero)

You are wasting two unique binary numbers (1000 and 0000) in representing the same integer value!

One's Complement

- Historically important (in other words, not used today!!!)
- Early computers built by Semour Cray (while at CDC) were based on 1's complement integers.

- Positive integers use the same representation as unsigned.
  - 0000 is 0
  - 0111 is 7, etc

- Negation is done by taking a bitwise complement of the positive representation.
  - Complement = Invert = Not = Flip = {0 -> 1, 1 -> 0}
  - A logical operation done on a single bit
- Top bit is sign bit

One's Complement Representation

To get 1's complement of –1

- Take +1: 0001
- Complement each bit: 1110
- Don't add or take away any bits.

Another example:

- 1100 ->flip 0011:3 -> -3

- This must be a negative number. To find out which, find the inverse.

- 0011 is +3
- 1100 in 1's Complement must be?

Properties of 1's complement:

- Any negative number will have a 1 in the MSB
- What is 0000? 0
- What is 1111? -0

Two's Complement

- Variation on 1's complement that does not have 2 representations for 0.
- This makes the hardware that does arithmetic simpler and faster than the other representations.
- The negative values are all "slid" by one, eliminating the –0 case.
- How to get 2's complement ***representation***:
  - Positive: just as if unsigned binary
  - Negative:
    - Take the positive value
    - Take the 1's complement of it
    - Add 1

Two's Complement

Example, what is -5 in 2SC?

1. What is 5? 0101
2. Invert all the bits: 1010 (basically find the 1SC)
3. Add one: 1010 + 1 = 1011 which is -5 in 2SC

To get the additive inverse of a 2's complement integer

1. Take the 1's complement
2. Add 1

What value is my negative number?

- Assume 4-bit number…
  - 1100 is negative, but what number is it?
  - Take 2SC again using same method!
    - Invert all bits
    - Add 1


```
1100 > 0011
     + 0001
       0100
```

Visualizing Signed Numbers

- Signed Magnitude
- One's Complement
- And… Two's Complement

	unsigned	signed magnitude	1's Complement	2's Complement
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1

Two's Complement

Number of integers representable is -2^{n-1} to $2^{n-1}-1$

So if 4 bits:

$[-8,\ldots,-1,0,+1,\ldots,+7] = 8 + 1 + 7 = 16 = 2^4$ numbers

Interesting observation about 2SC representation

- Assume you use n bits to represent your 2SC number
- The integer 0 is always represented as 0….0 (n times)
- The integer -1 is always represented as 11…..111 (n times)

Sign Extension

How to change a number with a smaller number of bits into the same number (same representation) with a larger number of bits?

This must be done frequently by arithmetic units

0010 = 2 (4 bits)
0000 0010 = 2 (8 bits)

Sign Extension - unsigned

Unsigned representation:

Copy the original integer into the LSBs, and put 0's elsewhere.

Thus for 5 bits to 8 bits:

xxxxx -> 000xxxxx

Sign Extension – signed magnitude

Signed magnitude:

Copy the original integer's magnitude into the LSBs & put the original sign into the MSB, put 0's elsewhere.

Thus for 6 bits to 8 bits

sxxxxx -> s00xxxxx

Sign Extension – 1SC and 2SC

1’s and 2’s complement:

1. Copy the original n-1 bits into the LSBs
2. Take the MSB of the original and copy it elsewhere

Thus for 6 bits to 8 bits:

sxxxxx -> sssxxxxx

Sign Extension

What is -12 in 8-bit 2's complement form
12
0000 1100 -> Binary
1111 0011 -> flip
1111 0100 -> +1

Arithmetic and Logical Operations

Logical Operations

- Operate on raw bits with 1 = true and 0 = false

		AND	OR	NAND	NOR	XOR	XNOR
In1	In2	&	|	~(&)	~(|)	^	~(^)
0	0	0	0	1	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	1	0	0	0	1

Logical Operations

- "bit-wise" logical operations are done in parallel for corresponding bits
  - Example & (AND):
    - X = 0011
    - Y = 1010
    - X AND Y = ?

Logical Operations

- “bit-wise” logical operations are done in parallel for corresponding bits
  - Example & (AND):
    - X = 0011
    - Y = 1010
    - X AND Y = X & Y = 0010