Today

- Review representations
- Floating point and hexadecimal

Unsigned integer representation

- With n bits, max value that can be represented: $2^n - 1$

Binary to Decimal conversion

1	1	0	1	0	0
2^5	2^4	2^3	2^2	2^1	2^0

32 + 16 + 4

=52

6 bits, so max number possible is 2^6-1 = 63

Binary to Decimal conversion

0	0	1	1	0	1
2^5	2^4	2^3	2^2	2^1	2^0

8 + 4 + 1

=13

6 bits, so max number possible is 2^6*-1 = 63*

Binary to Decimal conversion (8 bits)

1	1	0	0	1	1	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128+	64 +			8	+ 4		+ 1

=205

8 bits, so max number possible is 2^8-1 = 255

Binary to Decimal conversion (8 bits)

1	1	1	1	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

128+ 64 + 32 + 16+ 8 + 4 +2 + 1

=255

8 bits, so max number possible is 2^8-1 = 255

Decimal to Binary

1. Find number of bits required

$$Floor(\log_2 number) + 1$$

2. For each bit-position, starting from highest, Repeatedly check if number greater or equal to $2^{\text{bit-position}}$,and set bit to 0 or 1 accordingly

Number of bits required

- **52:** $\log_2(52) = 5.7; floor(5.7) = 5; \#\text{ bits} = 6$
  - Check 1: $2^6 - 1 \leq 52; 63 < 52$ (YES)
  - Check 2: $2^5 - 1 > 52; 32 > 52$ (YES)
- **102:** $\log_2(102) = 6.67; floor(6.67) = 6; \#\text{ bits} = 7$
  - Check 1: $2^7 - 1 \leq 102; 127 < 107$ (YES)
  - Check 2: $2^6 - 1 > 102; 63 > 107$ (YES)
- **276:** $\log_2(276) = 8.10; floor(8.10) = 8; \#\text{ bits} =$9
  - Check 1: $2^9 - 1 \leq 276; 511 < 276$ (YES)
  - Check 2: $2^8 - 1 > 276; 255 > 276$ (YES)

Number of bits required

- **64:** $\log_2(64) = 6.0; floor(6.0) = 6; \#\text{ bits} = 7$
  - Check 1: $2^7 - 1 \leq 64; 127 < 64$ (YES)
  - Check 2: $2^6 - 1 > 64; 63 > 64$ (YES)

Decimal to Binary

52; # bits = 6

Bit positions start at 0, so 6 bits means 2^0 to 2^5

Bit position	Power of 2	Number	>=	Remainder	Bit value
5	32	52	Yes	20	1
4	16				
3	8				
2	4				
1	2				
0	1				

We can check

1	1	0	1	0	0
2^5	2^4	2^3	2^2	2^1	2^0

32 + 16 + 4

=52

Decimal to Binary

37; # bits = 6

Bit positions start at 0

Bit position	Power of 2	Number	>=	Remainder	Bit value
5	32	37	Yes	5	1
4	16				
3	8				
2	4				
1	2				
0	1				

We can check

1	0	0	1	0	1
2^5	2^4	2^3	2^2	2^1	2^0
32			+ 4		+1

=37

Decimal to Binary

37; # bits = 6

Bit positions start at 0

Bit position	Power of 2	Number	>=	Remainder	Bit value
5	32	37	Yes	5	1
4	16				
3	8				
2	4				
1	2				
0	1				

2's complement representation

- Allows representing negative numbers
- **Arithmetic operations can be done by operating on individual bits**
- 0 is always all bits 0
- Most negative number: -2^{n-1}
- Most positive number: $+2^{n-1} - 1$

2’s complement range

1 bit	This is weird: -1 to 0		
2 bits	-2	To	+1
3 bits	-4	To	+3
4 bits	-8	To	7
5 bits	-16	To	15
6 bits	-32	To	31
7 bits	-64	To	63
8 bits	-128	To	127
9 bits	-256	To	255
10 bits	-512	To	511

Decimal to Binary (2's comp)

- First get number of bits

$$Floor(\log_2 abs(number)) + 2$$

- If positive number, then use process we developed before and you are done
- If negative number,
  - First get representation of the absolute value
  - Then invert all bits
  - Then add +1 to the inverted bits

Decimal to Binary

- 52

1. # bits = 7
2. Positive number; 0110100

All 7 bits.
Note that the MSB will always be zero in this step

Decimal to Binary

- -52

1. # bits = 7
2. Negative number
   a) Representation of +52 = `0110100`

All 7 bits.
Note that the **MSB will always be zero in this intermediate step**

   b) Invert all bits:
   c) Add +1:

```
 1001011
+0000001
=1001100
```

All 7 bits. Note that the MSB will always be **ONE for negative numbers** at the very end

Decimal to Binary

- -101

1. # bits = 8
2. Negative number
   a) Representation of -101 = 01100101

All 7 bits.
Note that the MSB will always be zero in this step

   b) Invert all bits: 10011010
   c) Add +1: +00000001
   =10011011

All 7 bits. Note that the MSB will always be **ONE for negative numbers** at the very end

Decimal to Binary

- -64

1. # bits = 7
2. Negative number
   a) Representation of +64 = `1000000`

All 7 bits.
Note that the **MSB will always be zero in this intermediate step**

   b) Invert all bits: `0111111`
   c) Add +1: `+0000001`
   `=1000000`

All 7 bits. Note that the MSB will always be **ONE for negative numbers** at the very end

2’s complement binary to decimal

- If MSB is 0, same as unsigned
- If MSB is 1, reverse steps:
  a) Invert all bits
  b) Add +1
  c) Now determine magnitude
     Remember it is a negative number

Binary to decimal

- 1001100
- MSB is 1

a) Invert all bits: 0110011

b) Add +1: +0000001

0110100

$2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

= 32+16+4 = 52

c) -52

d) Go back to 2's complement range and check

Binary to decimal

- `10011011`
- MSB is 1

a) Invert all bits: `01100100`

b) Add +1: +`00000001`

`01100101`

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

= 64+32+4+1 = 101

c) -101

d) Go back to 2's complement range and check

2's complement arithmetic
It's bitwise addition!

- 52 + (-101) = -49

```
  00110100
 +10011011
 ---------
  11001111
```

- Let's check what this value is

Check value

- `11001111`
- MSB is 1

a) Invert all bits: `00110000`

b) Add +1: +`00000001`

`00110001`

$2^7\ 2^6 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

= 32+16+1 = 49

c) -49

d) Go back to 2's complement range and check

2's complement extension

- -52 in 7 bits
- -52 in 8 bits
- -52 in 9 bits
- -52 in 10 bits

- -52

1. # bits = 7
2. Negative number
   a) Representation of +52 = `0110100`
   b) Invert all bits: `1001011`
   c) Add +1: `+0000001`
      `=1001100`

2’s complement extension

- -52 7 bits

a) Representation of +52 = 0110100

b) Invert all bits: 1001011

c) Add +1: +0000001

=1001100

Extension rule: 2s complement

```
  1001100     7 bits
 11001100     8 bits
111001100     9 bits
```

- To take a number represented in X bits can get its representation in Y bits, (Y > X), copy the **MSB** into the “new” bit positions

Fixed point

- After the decimal point negative powers of 2
- 0.001

1	.	1	0	1	0
2^0		2^{-1}	2^{-2}	2^{-3}	2^{-4}

2.625

Conversion from binary to decimal

- 0.43

Power of 2	Weight	Number	>=	Remainder	Bit value
2	0.5	0.43	No	0.43	0
4	0.25	0.43	Yes	0.18	1
8	0.125	0.18	Yes	0.055	1
16	0.0625	0.055	No	0.055	0
32	0.03125	0.055	Yes	0.02375	1
64	0.015625	0.02375	Yes	0.008125	1
128	0.0078125	0.008125	Yes	0.0003125	1
Represented value	**0.4296875**				

Power of 2	Weight	Number	>=	Remainder	Bit value
2	0.5	0.17	No	0.17	0
4	0.25	0.17	No	0.17	0
8	0.125	0.17	Yes	0.045	1
16	0.0625	0.045	No	0.045	0
32	0.03125	0.045	Yes	0.01375	1
64	0.015625	0.01375	No	0.01375	0
128	0.0078125	0.01375	Yes	0.0059375	1
Represented value	**0.1640625**				

Power of 2	Weight	Number	>=	Remainder	Bit value
2	0.5	0.14	No	0.14	0
4	0.25	0.14	No	0.14	0
8	0.125	0.14	Yes	0.015	1
16	0.0625	0.015	No	0.015	0
32	0.03125	0.015	No	0.015	0
64	0.015625	0.015	No	0.015	0
128	0.0078125	0.015	Yes	0.0071875	1
Represented value	**0.1328125**				

Power of 2	Weight	Number	>=	Remainder	Bit value
2	0.5	0.09	No	0.09	0
4	0.25	0.09	No	0.09	0
8	0.125	0.09	No	0.09	0
16	0.0625	0.09	Yes	0.0275	1
32	0.03125	0.0275	No	0.0275	0
64	0.015625	0.0275	Yes	0.011875	1
128	0.0078125	0.011875	Yes	0.0040625	1
Represented value	**0.0859375**				

Power of 2	Weight	Number	>=	Remainder	Bit value
2	0.5	0.01	No	0.01	0
4	0.25	0.01	No	0.01	0
8	0.125	0.01	No	0.01	0
16	0.0625	0.01	No	0.01	0
32	0.03125	0.01	No	0.01	0
64	0.015625	0.01	No	0.01	0
128	0.0078125	0.01	Yes	0.0021875	1
Represented value	**0.0078125**				

Power of 2	Weight	Number	>=	Remainder	Bit value
2	0.5	0.01	No	0.01	0
4	0.25	0.01	No	0.01	0
8	0.125	0.01	No	0.01	0
16	0.0625	0.01	No	0.01	0
32	0.03125	0.01	No	0.01	0
64	0.015625	0.01	No	0.01	0
128	0.0078125	0.01	Yes	0.0021875	1
256	0.00390625	0.002188	No	0.0021875	0
512	0.001953125	0.002188	Yes	0.000234375	1
Represented value	**0.009765625**				

Various conversions

- `1001001`
- Decimal value interpreted as unsigned representation?
- Decimal value interpreted as 2's complement representation?
- `0.1001001 = ?`

Floating point

- Allows representing very large and very small numbers
- Standard used in all machines today
  - Much interesting theory behind it
- It is equivalent to scientific notation but done in binary.
- Think:
  - $0.145 = 1.45 * 10^{-2}$
  - $0.0090897 = 9.0897 * 10^{-3}$
  - $145 = 1.45 * 10^{+2}$
- How do we do this with binary?

Floating Point Standard
IEEE-754 Standard
Single Precision Representation

1 bit	8 bits	23 bits
S	Exponent	Fraction

- $N = -1^{S} * 1.fraction * 2^{exponent-127}$
  $when\ 1 \leq exponent \leq 254$
- $N = -1^{S} * 0.fraction * 2^{-126}$
  $when\ exponent == 0$

Example

```
11000000110010001000000000000000
1   10000001   10010000000000000000000
```

```
S = 1 ; therefore negative number
exponent = 129
fraction = 10010000000000000000000
```

$$N = -1^1 * 1.1001 * 2^{129-127}$$

$$N = -1^1 * 1.1001 * 2^2$$

$$N = -1^1 * 110.01$$

$$N = 6.25$$

Example 2

01000010110010001000000000000000

0 10000101 10010000000000000000000

S = 1 ; therefore negative number

exponent = 133

fraction = 10010000000000000000000

$$N = -1^1 * 1.1001 * 2^{133-127}$$

$$N = -1^1 * 1.1001 * 2^5$$

$$N = -1^1 * 110010$$

$$N = 50$$

Hexadecimal notation

- Would be nice to not write so many bits!
- Simple scheme to represent 16 bits with one symbol
- 16 is a power of 2, in fact it is 2 raised 4.
- So one hexadecimal “symbol” represent 4 bits.
- The symbols are :

  0 through 9, a, b, c, d, e, f

Hexadecimal table

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	C
1110	E
1111	F

0100 0010 1100 1000 1000 0000 0000 0000

?

Unsigned integer representation

- With n bits, max value that can be represented: $2^n - 1$

Decimal to Binary

52; # bits = 6

Bit positions start at 0, so 6 bits means 2^0 to 2^5

Bit position	Power of 2	Number	>=	Remainder	Bit value
5	32	52	Yes	20	1
4	16	20	Yes	4	1
3	8	4	No	4	0
2	4	4	Yes	0	1
1	2	0	No	0	0
0	1	0	No	0	0

Decimal to Binary

- -52

1. # bits = 7
2. Negative number
   a) Representation of +52 = 0110100

All 7 bits. Note that the MSB will always be zero in this intermediate step

   b) Invert all bits: 1001011
   c) Add +1: +0000001
      =1001100

All 7 bits. Note that the MSB will always

Extension rule: 2s complement

```
  1001100     7 bits
 11001100     8 bits
111001100     9 bits
```

- To take a number represented in X bits can get its representation in Y bits, (Y > X), copy the **MSB** into the "new" bit positions

Binary to Decimal conversion

```
1   1   0   1   0   0
2^5 2^4 2^3 2^2 2^1 2^0
32 + 16     + 4
=52
```

6 bits, so max number possible is 2^6-1 = 63

2's complement representation

- Allows representing negative numbers
- **Arithmetic operations can be done by operating on individual bits**
- 0 is always all bits 0
- Most negative number: -2^{n-1}
- Most positive number: $+2^{n-1} - 1$

2's complement binary to decimal

- If MSB is 0, same as unsigned
- If MSB is 1, reverse steps:
  a) Invert all bits
  b) Add +1
  c) Now determine magnitude
     Remember it is a negative number

Fixed point

- After the decimal point negative powers of 2
- 0.001

```
1 . 1   0   1   0
2^0   2^-1 2^-2 2^-3 2^-4
2.625
```

Decimal to Binary

1. Find number of bits required
   $Floor(\log_2 number) + 1$
2. For each bit-position, starting from highest, Repeatedly check if number greater or equal to $2^{\text{bit-position}}$,and set bit to 0 or 1 accordingly

2's complement range

1 bit	This is weird: -1 to 0		
2 bits	-2	To	+1
3 bits	-4	To	+3
4 bits	-8	To	7
5 bits	-16	To	15
6 bits	-32	To	31
7 bits	-64	To	63
8 bits	-128	To	127
9 bits	-256	To	255
10 bits	-512	To	511

Binary to decimal

- 1001100
- MSB is 1
  a) Invert all bits: 0110011
  b) Add +1: +0000001
     0110100
     $2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

= 32+16+4 = 52
c) -52
d) Go back to 2's complement range and check

Conversion from binary to decimal

- 0.43

Power of 2	Weight	Number	>=	Remainder	Bit value
2	0.5	0.43	No	0.43	0
4	0.25	0.43	Yes	0.18	1
8	0.125	0.18	Yes	0.055	1
16	0.0625	0.055	No	0.055	0
32	0.03125	0.055	Yes	0.02375	1
64	0.015625	0.02375	Yes	0.008125	1
128	0.0078125	0.008125	Yes	0.0003125	1
Represented value	**0.4296875**				

Number of bits required

- **52:** $\log_2(52) = 5.7; floor(5.7) = 5; \#\text{ bits} = 6$
  - Check 1: $2^6 - 1 \leq 52; 63 < 52$ (YES)
  - Check 2: $2^5 - 1 > 52; 32 > 52$ (YES)
- **102:** $\log_2(102) = 6.67; floor(6.67) = 6; \#\text{ bits} = 7$
  - Check 1: $2^7 - 1 \leq 102; 127 < 107$ (YES)
  - Check 2: $2^6 - 1 > 102; 63 > 107$ (YES)
- **276:** $\log_2(276) = 8.10; floor(8.10) = 8; \#\text{ bits} = 9$
  - Check 1: $2^9 - 1 \leq 276; 511 < 276$ (YES)
  - Check 2: $2^8 - 1 > 276; 255 > 276$ (YES)

Decimal to Binary (2's comp)

- First get number of bits
  $Floor(\log_2 abs(number)) + 2$
- If positive number, then use process we developed before and you are done
- If negative number,
  – First get representation of the absolute value
  – Then invert all bits
  – Then add +1 to the inverted bits

2's complement arithmetic
It's bitwise addition!

- 52 + (-101) = -49

```
  00110100
 +10011011
 ---------
  11001111
```

- Let's check what this value is

Floating Point Standard
IEEE-754 Standard
Single-Precision Representation

1 bit	8 bits	23 bits
S	Exponent	Fraction

- $N = -1^S * 1.fraction * 2^{exponent-127}$
  $when\ 1 \leq exponent \leq 254$
- $N = -1^S * 0.fraction * 2^{-126}$
  $when\ exponent == 0$

Representation

- Binary
- Integers
  - Sign magnitude representation
  - 2’s complement
- Decimal numbers (fixed point)
- Floating point
- Boolean Logic
- Hexadecimal digits

CS 252

Lecture 12; 2015 Oct 5th; Transcribed Lecture notes

Review Number Representation

With n bit, the max value that can be presented is $2^n - 1$

Binary to Decimal Conversion

1 1 0 1 0 0 = 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 0*2^0 = 32 + 16 + 4 = 52

0 0 1 1 0 1 = 0*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 8 + 4 + 1 = 13

More bits means more weights

1 1 0 0 1 1 0 1 = 1*2^5 + 1*2^4 + 0*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 128 + 64 + 8 + 4 + 1 = 205

Decimal to Binary Conversion

Find the number of bits required

$floor(log_2(number)) + 1$

For each bit-position, starting from highest, and set bit to 1 if greater than that position value

Number of bits required:

52: log base 2 of 52 = 5.7, so need 6 bits

102: log base 2 of 102 = 6.67, so need 7 bits

276: log base 2 of 276 = 8.10, so need 9 bits

Number Conversion:

52, 6 bits, = 32 + 16 + 4 = 1*32 + 1*16 + 0*8 + 1*4 + 0*2 + 0*1 = 1 1 0 1 0 0

37, 6 bits, = 32 + 4 + 1 = 1*32 + 0*16 + 0*8 + 1*4 + 0*2 + 1*1 = 1 0 0 1 0 1

2's complement representation: allows representing negative number

arithmetic operations can be done by operating on individual bits

most negative: -2^{n-1}

most positive: $2^{n-1} - 1$

2's complement range

1 bit: -1 to 0

2 bits: -2 to 1

3 bits: -4 to 3

4 bits: -8 to 7

5 bits: -16 to 15

6 bits: -32 to 31

7 bits: -64 to 53

8 bits: -128 to 127

9 bits: -256 to 255

10 bits: -512 to 511

Decimal to Binary (2's complement)

If not negative

number of bits = $floor(log_2(number)) + 2$

Convert the same method as before

If negative

number of bits = $floor(log_2(number - 1)) + 2$ *(not mentioned in slides)*

Convert by

1. get positive value representation
2. invert all bits
3. add 1

52: 7 bits, 0110100 using the same method as before

-52: 7 bits, start with abs value 0110100; then invert all bits: 1001011; and add 1: 1001100

-101: 8 bits, start with abs value: 01100101; then invert all bits: 10011010; and add 1: 10011011

-64: 7 bits, start with abs value: 1000000; then invert all bits: 10111111; and add 1: 1000000

If the most significant bit (MSB) is 0, then same as unsigned

If MSB is 1, reverse steps to find decimal number

1001100: MSB is 1, so invert all bits 0110011; add 1: 0110100; convert to decimal: -52

In 2's complement, arithmetic is bitwise addition!

52+-101=-49: 00110100 + 10011011 = 11001111

Homework: (-101) + (-163) = -264

10011011 + 10100011 = 1011111000

2's complement sign extension:

-52 in 7 bits: 1001100

-52 in 8 bits: 11001100

52 in 8 bits is 00110100; invert all bits: 11001011; add 1: 11001100

-52 in 9 bits: 111001100

-52 in 10 bits: 1111001100

simply add more of the most significant bits

Fixed Point Representation

After the binary point, it presents negative powers of 2

1.1010 is 1*2^0 + 1*2^-1 + 0*2^-2 + 1*2^-3 + 0*2^-4 = 1.625

Conversion from binary to decimal

0.43 is represented as 0.4296875 = 0.0110111 as shown in the table below

Power of 2	weight	number	>=	remainder	bit value
2	0.5	0.43	no	0.43	0
4	0.25	0.43	yes	0.18	1
8	0.125	0.18	yes	0.055	1
16	0.0625	0.055	no	0.02375	0
32	0.03125	0.055	yes	0.008125	1
64	0.015625	0.02375	yes	0.0003125	1

0.17 is represented as 0.1640625 = 0.0010101

0.01 is represented as 0.0078125 = 0.00000001 or 0.00976525 = 0.000000101

Closer approximation as we increase the number of bits

Floating Point Representation

Allow representing very large and very small numbers

Standard used in all machines today

It is scientific notation done in binary

0.145 = 1.45 * 10^-2

0.0090897 = 9.0897 * 10^-3

145 = 1.45 * 10^2

Format: 1 signed bit, 8 bits exponent, and 23 bits fraction, with an implicit 1 at the left of the binary point

If $exponent == 0$

N = $(-1)^s * 0.fraction * 2^{-126}$

Else

N = $(-1)^s * 1.fraction * 2^{exponent - 127}$

Example 1

1_1000001_10010000000000000000000

S = 1; therefore negative number

exponent = 129

fraction = 10010000000000000000000

N = (-1)^1*1.1001 *2^(129-127) = 6.25

Example 2

0_1000101_10010000000000000000000

S = 1; therefore positive number *(typo in slides)*

exponent = 133

fraction = 10010000000000000000000

N = (-1)^0*1.1001 *2^(133-127) = 50

Example 3

743.2

fraction = 0110 0111

exponent = 136 = 10001000

final representation = 0 1001000 0111 0011 1000 0000 0000 000

Hexadecimal Representation

Convenience sake to avoid writing so many bits

Has nothing to do with how machines interpret numbers

4 bits is represented as 1 symbol: 0-9 and A-F

0100 0010 1100 1000 1000 0000 0000 0000 = 4 2 C 8 8 0 0 0

prefixed with 0x, 0X, 0h, or 0H

side note: binary numbers can be prefixed with 0b or 0B