Number Representation Notes

CS61C Note 1

Cesar

1 Representation Scheme

Representation Scheme: A method that tells us how to store information (e.g., numbers, addresses).

Example: Number representation

Number Representation: A set of rules for how numbers are interpreted and encoded.

Example: Base systems

Base: A number system that defines the number of unique digits used to represent an integer.

Examples of Common Base Systems

1. Binary (Base-2) = 0, 1
2. Octal (Base-8) = 0, 1, 2, 3, 4, 5, 6, 7
3. Decimal (Base-10) = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
4. Hexadecimal (Base-16) = 0–9 and A–F, where A = 10, B = 11, ..., F = 15

All of these are number systems.

We index starting from 0 meaning 10^0 beginning from the rightmost (the one's place).

Number Value:	1	3	2	9
Place Value:	thousand's	hundred's	ten's	one's
Power of 10 Index:	10^3	10^2	10^1	10^0

Number Value:	1	0 .	2	9
Place Value:	ten's	one's	tenth's	hundredth's
Power of 10 Index:	10^1	10^0	10^{-1}	10^{-2}

2 Decimal (Base-10)

Decimal: A positional number system that represents any integer.

Digits for base-10 system: 0–9

Example: When we write

$$6 = (6 \times 10^0) = 6$$

we are showing how the decimal value 6 can be represented using the base-10 system. We use powers of 10 to multiply because there are 10 distinct numbers. In this example, the digit 6 is in the one's place falling in the range of 0-9. After the 9 digit, we roll over to the next place value:

$$9 \rightarrow 10$$

Meaning you roll over to the ten's place in the number 10. It will roll over to 10-99 digits until the value reaches a new roll over of 100-999 digits, etc.

Number:	1	0
Place value:	ten's	one's
Power of 10 index:	10^1	10^0

Here is a number that rolls over the digits 0-9

$$15 = (1 \times 10^1) + (5 \times 10^0) = 15$$

The digit 15 falls between 10-99 digits.

3 Binary (Base-2)

Binary: A positional number system that represents two digits.

Digits: 0 and 1.

Binary prefixes: `0b`, subscript 2, or just binary digits.

Example: We have 0b0100

Binary Values:	0	1	0	0
Power of 2 Index:	2^3	2^2	2^1	2^0

$$= (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 4$$

Binary rolls over after 1, just like decimal rolls over after 9.

$$0b0, \text{0b1}, 0b10, \text{0b11}, 0b100, 0b101, 0b110, \text{0b111}, 0b1000$$

After the binary 0b1, it rolls over to 0b10, because there is a binary digit of 1 at the one's place with no zero's. Similar, after 0b11, we roll over to 0b100, because we have a binary digit of 1 at the one's and two's place with no zero's, etc.

4 Storing Information: Bits

A **bit** is a binary digit (0 or 1). A single bit can represent anything, e.g., 0 = False, 1 = True.

Example: The binary `0b1010` has 4 bits, it is just the total number of 0's and 1's.

Common groupings:

- Nibble = 4 bits = 0b1000
- Byte = 8 bits = 0b1000 0000
- 2 Bytes = 16 bits = 0b1000 0000 0000 0000
- 4 Bytes = 32 bits = 0b1000 0000 0000 0000 0000 0000 0000 0000

Formulas

$$2^n = \text{Total number of combination values of 0 and 1 representable with } n \text{ bits}$$

$$2^n - 1 = \text{Maximum decimal value with } n \text{ bits}$$

$$2^{(n-1)} = \text{Value of the leftmost (most significant) bit}$$

Example 1: With 4 bits:

$$2^4 = 16 \quad \Rightarrow \quad \text{16 total combinations of 0 and 1, example 0b0000, 0b0001, ...0b1111}$$

$$2^4 - 1 = 15 \quad \Rightarrow \quad \text{Max decimal value} = 15 \text{ representable with 4 bits}$$

$$2^{(4-1)} = 2^3 = 8 \quad \Rightarrow \quad \text{Significant bit value} = 8 = \text{0b1000}$$

Example 2: With 8 bits:

$$2^8 = 256$$

$$2^8 - 1 = 255$$

$$2^{(8-1)} = 128$$

5 Examples of Binary and Decimal

Binary to Decimal

Example 1:
`0b11001011`

Binary Value:	1	1	0	0	1	0	1	1
Digit Index:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Thus, `0b11001011`

$$(1 \times 128) + (1 \times 64) + (0 \times 32) + (0 \times 16) + (1 \times 8) + (0 \times 4) + (1 \times 2) + (1 \times 1)$$

$$= 128 + 64 + 8 + 2 + 1 = 203$$

Example 2:
`0b00111111`

Binary Value:	0	0	1	1	1	1	1	1
Digit Index:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Thus, `0b00111111`:

$$(0 \times 128) + (0 \times 64) + (1 \times 32) + (1 \times 16) + (1 \times 8) + (1 \times 4) + (1 \times 2) + (1 \times 1)$$

$$= 32 + 16 + 8 + 4 + 2 + 1 = 63$$

Decimal to Binary

Example 1:
`143, using 8 bits, our leftmost digit is 128 which is` 2^{8-1}

Digit Index:	128	64	32	16	8	4	2	1
Binary Value:	1	0	0	0	1	1	1	1

Thus, `143` = 0b10001111

Example 2:
`9, using 4 bits, our leftmost digit is 8 which is` 2^{4-1}

Digit Index:	8	4	2	1
Binary Value:	1	0	0	1

Thus, $9 =$ 0b1001

Example 3:
`3.14, using 4 integer bits for the decimal 3, and 4 fractional bits for .14 decimal`

`3`

Digit Index:	8	4	2	1
Binary Value:	0	0	1	1

Thus, 3 = 0b0011

```
.14, for fractions, multiply by 2
```

fraction:	.14	.28	.56	.12
$fraction * 2$:	.28	.56	1.12	.24
Binary:	0	0	1	0

So, the binary of .14 using 4 bits fractional is

$$.14 \approx 0b0.0010.$$

We put a 1 at the third step because $0.56 \times 2 = 1.12 \geq 1$. That means the 2^{-3} place (the 1/8 place) is filled with a 1. The earlier steps gave 0s in the 2^{-1} and 2^{-2} places.

Thus, the binary of 3.14 using 4 bits and 4 fractional bits is

$$3.14 \approx 0b0011.0010.$$

Example 4:

```
.36
```

fraction:	.36	.72	.44	.88
$fraction * 2$:	.72	1.44	.88	1.76
Binary:	0	1	0	1

So,

$$.36 \approx 0b0.0101.$$

6 Hexadecimal (Base-16)

Hexadecimal: A positional number system that includes 16 digits.

Digits: 0-9, and A-F, where A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.

Prefix: 0x

Example: "0x14"

$$0x14 = (1 * 16^1) + (4 * 16^0) = 20$$

What we did is convert hexadecimal to decimal using the power of 16 because we have 16 different digits. Additionally, we know that $16^1 = 2^4 = 16$, and $16^0 = 2^0 = 1$ thus we can write,

$$0x14 = (1 * 16^1) + (4 * 16^0)$$

$$0x14 = (0b0001) * 2^4 + (0b0100) * 2^0 (\textit{Converted the values 1 and 4 to binary})$$

$$0x14 = (0b0001)0000 + 0b0100 (\textit{concatenate})$$

$$0x14 = 0b00010100 (\textit{Binary})$$

Now we converted hexadecimal to binary. We converted 1 and 4 to binary because 2^4 is 16^1, and each hex digit maps to a nibble, then we concatenate.

Steps from hexadecimal to decimal.

1. Multiply the individual values by 16 with its respective position.
2. Then add those values to obtain the decimal.

Steps from decimal to hexadecimal.

1. Convert the whole decimal to binary.
2. Then split the binary into nibbles, resulting to hex form.

Steps from hexadecimal to binary.

1. Convert the individual values to binary.
2. Then concatenate both binary values to obtain the final binary.

Steps from binary to hexadecimal.

1. Split the binary into two nibbles.
2. Then match it to the right digit.

7 Examples of Hexadecimal

Hexadecimal to Decimal

Example 1: $0x5A$

$$0x5A = (5 * 16^1) + (10 * 16^0) = (5 * 16) + 10 = 90$$

Example 2: $0xF1$

$$0xF1 = (15 * 16^1) + (1 * 16^0) = (15 * 16) + 1 = 241$$

Decimal to Hexadecimal

Example 1: 21 using 8 bits because we want two nibbles to represent a hexadecimal, so our leftmost digit is $2^{8-1} = 2^7 = 128$.

Digit Index:	128	64	32	16	8	4	2	1
Binary Value:	0	0	0	1	0	1	0	1

So,

$$23 = 0b00010101 = 0x15$$

Example 2: 9

Digit Index:	128	64	32	16	8	4	2	1
Binary Value:	0	0	0	0	1	0	0	1

So,

$$9 = 0b00001001 = 0x09$$

Example 3: 233

Digit Index:	128	64	32	16	8	4	2	1
Binary Value:	1	1	1	0	1	0	0	1

So,

$$233 = 0b11101001 = 0xE9$$

Hexadecimal to Binary

Example 1: $0x45$

$$0x45 = (4 * 16^1) + (5 * 16^0)$$

$$0x45 = (0b0100 * 2^4) + 0b0101$$

$$0x45 = (0b0100 * 0000) + 0b0101 = 0b01000101$$

Example 2: $0x6B$

$$0x6B = (6 * 16^1) + (11 * 16^0)$$

$$0x6B = (0b0110 * 2^4) + 0b1011$$

$$0x6B = (0b0110 * 0000) + 0b1011 = 0b01101011$$

Binary to Hexadecimal

Example 1: $0b11001100$

Binary in two nibbles:	1100	1100
Hexadecimal value:	C	C

So,

$$0b11001100 = 0xCC$$

Example 2: $0b10100111$

Binary in two nibbles:	1010	0111
Hexadecimal value:	A	7

So,

$$0b10100111 = 0xA7$$

8 Octal (Base-8)

Octal: A positional number system that includes 8 digits.

Digits: 0, 1, 2, 3, 4, 5, 6 ,7.

Prefix: 0o.

Example 1: "0o12"

$$0o12 = (1 * 8^1) + (2 * 8^0) = 8 + 2 = 10$$

We converted octal to decimal. Similar to hexadecimal, if you want to go from decimal to octal, you follow the same steps.

Example 2: 83

Digit Index:	256	128	64	32	16	8	4	2	1
Digit Index:	256	128	64	32	16	8	4	2	1
Binary Value:	0	0	1	0	1	0	0	1	1

So, 1 octal = 3 bits, we can group the binary to 3 bits, because $8^1 = 2^3$, where n = 3, number of bits.

$$83 = 0b001010011 = 0o123$$

Example 3: $0b101101$ to octal form

$$0b101101 = 0b101 \quad 0b101 = 5 \quad 5 = 0o55.$$

9 Ways to Represent Signed Integers to Binary

Bias

Bias: A fixed number added to a value so that the signed value is represented as a unsigned binary value.

General formulas for bias

$$Signed : value(min) = stored(signed) - bias$$

$$Unsigned : stored(max) = value(unsigned) + bias$$

Bias is stated as

$$Bias = 2^{n-1} - 1$$

Ranges: We can find the ranges for a given bit size using

$$[-(2^{n-1} - 1), 2^{n-1}]$$

Where $-(2^{n-1} - 1)$ is our signed value aka min and 2^{n-1} is our unsigned value aka max.

For example, a 4 bit representation value range is:

$$Min = stored - bias = 0 - bias = 0 - (2^{n-1} - 1) = 0 - (2^{4-1} - 1) = 0 - 7 = -7$$

$$Max = value + bias = (2^n - 1) + (-2^{n-1} - 1) = (2^4 - 1) + (-7) = 15 - 7 = 8$$

So the range is $[-7, 8]$.

For a 8 bit representation value range is:

$$Min = stored - bias = 0 - bias = 0 - (2^{n-1} - 1) = 0 - (2^{8-1} - 1) = 0 - 127 = -127$$

$$Max = value + bias = (2^n - 1) + (-2^{n-1} - 1) = (2^8 - 1) + (-127) = 255 - 127 = 128$$

So the range is $[-127, 128]$ or in short $[-(2^{n-1} - 1), 2^{n-1}]$.

Examples of Bias

Example 1: We have a 8 bit number 0b0000 1001 = 9. What is the stored value of the value 9.

This a unsigned integer so,

$$stored = value + bias = 9 - 127 = -118$$

So what we did was convert a unsigned binary to a signed integer. Which is not the goal of a bias representation. The objective is to translate a signed integer into a unsigned binary representation.

Example 2: Store the value -9 to a unsigned binary representation using 8 bits.

This a signed integer so,

$$value = stored - bias = -9 - (-127) = 118 = 0b01110100$$

So we have successfully converted a signed integer to a unsigned binary represenation, so $-9 = 0b01110100$ using bias representation.

In summary, bias representation turns signed integers into unsigned binary representation, and turns unsigned integers into signed values.

Two's Complement

Two's Complement: A method of representing signed integer to unsigned binary representation, using two distinct steps. This is how computers store sign integers.

Ranges: The range for two's complement is stated as,

$$[-(2^{n-1}), 2^{n-1} - 1]$$

Where -2^{n-1} tells us the most negative number(aka MSB sign), and $2^{n-1} - 1$ tells us the most positive number(aka magnitude).

The range of a 4 bit size: $[-(2^{n-1}), 2^{n-1} - 1] = [-2^{4-1}, 2^{4-1} - 1] = [-8, 7]$

The range of a 8 bit size: $[-(2^{n-1}), 2^{n-1} - 1] = [-2^{8-1}, 2^{8-1} - 1] = [-128, 127]$

Steps for Two's Complement:

1.) First, find the binary of the unsigned format of the signed integer.

2.) Second, get the one's complement of each bit, meaning $1- > 0$, and $0- > 1$.

3.) Finally, get the two's complement by adding one to that binary.

Examples of Two's Complement

Example 1: Convert -12 using the two's complement.

First, we know that -12 falls between a 8 bit size, $[-128, 127]]$:

$$12 = 0b00001100$$

Second,

$$0b00001100 = 0b11110011$$

Finally,

$$0b11110011 + 0b00000001 = 0b11110100$$

Proof, our MSB is our sign bit, where $1 = -$ and $0 = +$.

$$0b11110100 = -128 + 64 + 32 + 16 + 4 = -12$$

Example 2: Convert -6 using the two's complement.

$$6 = 0b0110$$

$$0b0110 = 0b1001$$

$$0b1001 + 0b0001 = 0b1010 = -8 + 2 = -6$$

Sign Magnitude

Sign Magnitude: A method of representing signed integers as unsigned binary.

Ranges: The is range is given as,

$$[-(2^{n-1} - 1), 2^{n-1} - 1]$$

The range for a 4 bit size: $[-(2^{4-1} - 1), 2^{4-1} - 1] = [-7, 7]$

The range for a 8 bit size: $[-(2^{8-1} - 1), 2^{8-1} - 1] = [-127, 127]$

Example: Represent -7 using sign magnitude.

We know that the signed integer falls between the range of a 4 bit size representation. So we can use 4 bits.
We always dedicate a bit the sign and the rest is our magnitude that make up the value 7.

Signs

$$- = 0b1$$

$$+ = 0b0$$

Magnitude of 7 using 3 bits

$$7 = 0b111$$

Combine both the sign bit and magnitude

$$-7 = 0b1111$$

However, this has a lot of shortcomings like the lack of arthmetic logic in computing, it introduces a lot of zero representations.

10 Mathematical Operations on Binary

Addition and Subtraction

General rule for addition is.

$1 + 1 = 0$ means 0, you carry over a 1 to the next place

$$1 + 1 = 0$$

$$1 + 0 = 1$$

$$0 + 1 = 1$$

$$0 + 0 = 0$$

Example: Involving 4 bit Two's Complement: $0b1001 + 0b0011$

First check the range if the operation is possible, we know that the range for a bit in two's complement is [-8,7]. Where $0b1001 = 9$ and $0b0011 = 3$, 9 is out of scope so we can not do this operation.

Example 2: Involving 8 bit Two's Complement: $0b00001001 + 0b00000011$

The range of 8 bit: $[-127, 128]$, both integers 9 and 3 falls between the range, so

$$0b00001001 + 0b00000011 = 0b00001100$$

Example 3: Involving 6 bit Two's Complement: $0b011000 - 0b000011$

The range of 6 bits: $[-32, 31]$, both integers 24 and 3 fall in that range. First lets find the two's complement of -3.

$$-3 = 3 = 0b000011 = 0b111100 = 0b111100 + 1 = 0b111101$$

Then

$$0b011000 + 0b111101 = 0b010101$$

Example 4: Involving 6 bits Two's Complement: $0x3B - 0x06$

Both values falls between the range. Lets find the two's complment of -6. Lets find the binary of -6 using two's complement.

$$-6 = 6 = 0b000110 = 0b111001 = 0b111001 + 1 = 0b111010$$

Then

$$0x3B - 0x06 = 0b111011 + 0b111010 = 0b110101$$

Multiple

General rule for multiplying binaries,

$$1 * 1 = 1$$

$$1 * 0 = 0$$

$$0 * 1 = 0$$

$$0 * 0 = 0$$

Example: $0b001001 * 0b000011$, it is very similar to multiplying real numbers

$$0b001001 * 0b000011 = 0b011011$$