Binary Math and Boolean Logic

Lecture Contents

- Number Systems
- Binary Numbers
- Binary Operations
- Two’s Complement
- Conversion Between Bases

Canberra Institute of Technology

aie

SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Number Systems

- The number system most frequently used in society is the decimal number system, also referred to as the base ten number system.
- There are other number systems
  - Base 2 : Binary
  - Base 8 : Octal
  - Base 16 : Hexadecimal

Number Systems

- Computer memory is made up of bits – binary digits – represented by the base 2 number system.
- These can be hard for humans to understand – so we can use the other numbering systems to represent binary:
  - Base 8 : Octal
  - Base 10 : Decimal
  - Base 16 : Hexadecimal

Common Theme

- Numerals in these number systems are organised into columns.
- Each column is the base number raised to a power, the far right column represents $base^0$.

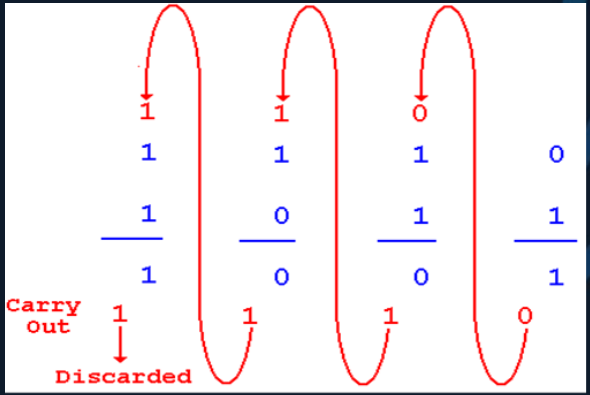
Thousands	Hundreds	Tens	Ones
10^3	10^2	10^1	10^0
1	2	3	4

Decimal – Base 10

- Uses the numerals 0 - 9
- If a numeral exceeds 9 in a given column that value will roll over to zero and one is added to the column to the immediate left.

Binary – Base 2

- Numerals 0 and 1
- If a numeral exceeds one in a given column that value rolls over to zero, and one is added to the column to the left.



8's	4's	2's	1's
2^3	2^2	2^1	2^0
1	0	1	0

Hexadecimal – Base 16

- Numerals 0 – 9 then A, B, C, D, E, F
- If a numeral exceeds nine then letters are used
- If a numeral exceeds F in a given in a given column that value rolls over to zero, and one is added to the column to the left

4096's	256's	16's	1's
16^3	16^2	16^1	16^0
7	A	F	F

Base 8

- Numerals 0 - 8
- If a numeral exceeds seven in a given column that value rolls over to zero, and one is added to the column to the left

512's	64's	8's	1's
8^3	8^2	8^1	8^0
7	5	1	7

Conversion Between Bases

- We can manually convert between the different bases using some simple maths
  - Decimal -> Binary
  - Binary -> Decimal
  - Binary -> Hex

Decimal -> Binary

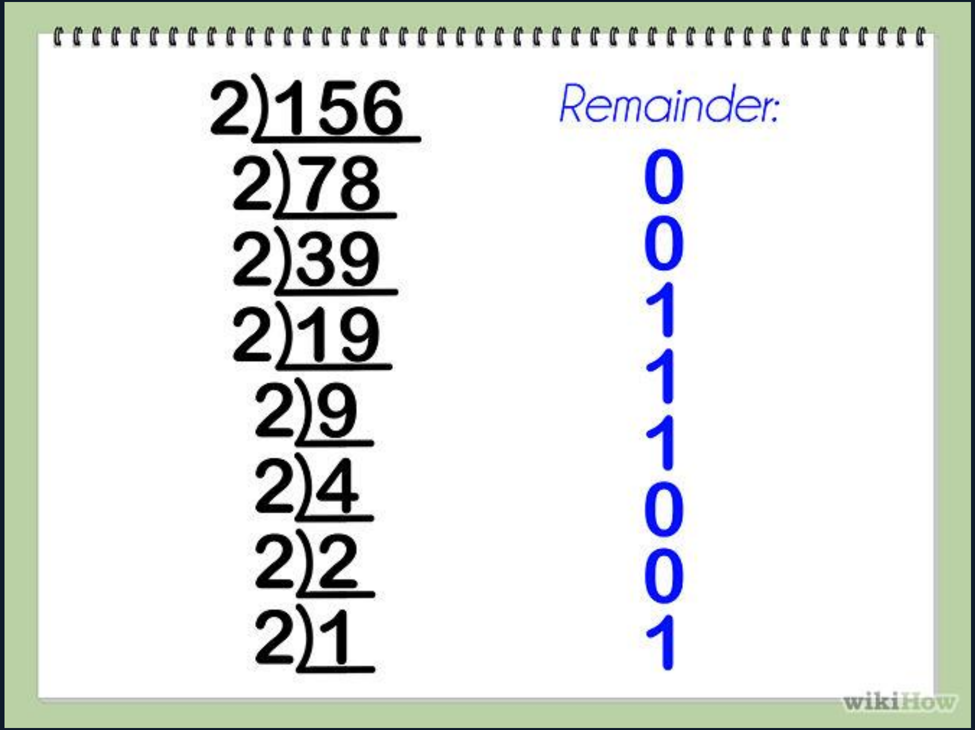
- There are two main methods to convert from Decimal to Binary:
  - Short division by two with remainder
  - Comparing powers of two and subtraction

Decimal -> Binary (Divide by two method)

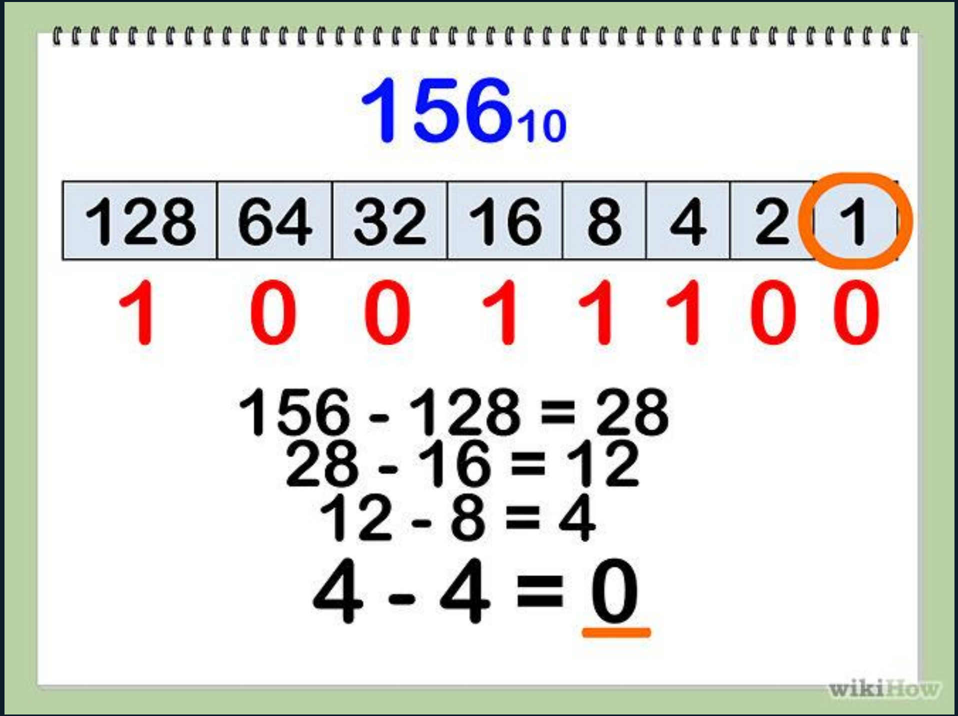
1. Take the number we need to convert.
2. Divide it by two.
3. Note down the remainder (0 or 1).
4. Take the result from 2 and repeat steps 2 and 3 until the result is one or zero.
5. The remainders when read backwards form the binary number.

Decimal -> Binary (divide by two method)

- In this case the number is: 1001 1100



Decimal->Binary (Powers of two)



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Different Base, Same Math

- Mathematical operators applied to decimal numbers can also be applied to binary numbers:

Base 10	Operator	128's	64's	32's	16's	8's	4's	2's	1's
170		1	0	1	0	1	0	1	0
85	+	0	1	0	1	0	1	0	1

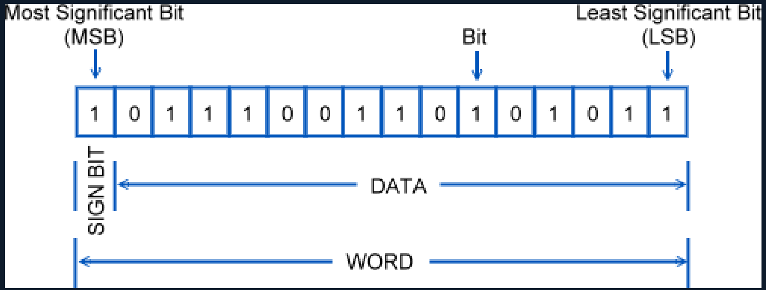
Truncation

- Consider the value 255 from the previous slide. If 1 is added to it a chain reaction of carries occurs.
- All bits flip and 1 is carried to the left and lost.
- The number that was meant to be 256 is now zero.

Operator	128's	64's	32's	16's	8's	4's	2's	1's
	1	1	1	1	1	1	1	1
+	0	0	0	0	0	0	0	1
=	0	0	0	0	0	0	0	0

Negative Binary Numbers

- So far all the binary numbers we've looked at have been positive, or unsigned.
- How do we represent negative numbers in binary?
- Sign and Magnitude - the most significant bit (left-most) represents the sign on the number, leaving 15 bits (in an integer, for example) to store the magnitude. 1 for negative, and 0 for positive.



One’s Complement

- A simple method to represent negative binary numbers.
- Flip all the bits in a binary number – that’s it!

Base 10	sign	64’s	32’s	16’s	8’s	4’s	2’s	1’s
10	0	0	0	0	1	0	1	0
-10	1	1	1	1	0	1	0	1

Two’s Complement

- To represent negative values computers today use a *two’s complement* notation.
- The two’s complement of a bit pattern is the inversion of all bits followed by the addition of one.

Base 10	Step	sign	64’s	32’s	16’s	8’s	4’s	2’s	1’s
5		0	0	0	0	0	1	0	1
	invert	1	1	1	1	1	0	1	0
-5	add 1	1	1	1	1	1	0	1	1

Two's Complement

- The most significant bit (MSB) is a sign bit.
- If set the number is negative.
- The reverse works in the same fashion.

Base 10	Step	sign	64's	32's	16's	8's	4's	2's	1's
-5		1	1	1	1	1	0	1	1
	invert	0	0	0	0	0	1	0	0
5	add 1	0	0	0	0	0	1	0	1

Two's Complement Works With Zero

- Consider the two's complement of Zero:

Base 10	Step	sign	64's	32's	16's	8's	4's	2's	1's
0		0	0	0	0	0	0	0	0
	invert	1	1	1	1	1	1	1	1
0	add 1	0	0	0	0	0	0	0	0

- The inversion sets all bits, the addition of one results in a truncation; which results in a value of zero

Bitwise Operators

- Bitwise operators operate on integral types, treating them as a collection of bits

Operator	Function	Use
&	bitwise AND	operand & operand
|	bitwise OR	operand | operand
~	bitwise NOT	~operand
^	bitwise XOR	operand ^ operand
<<	left shift	operand << operand
>>	right shift	operand >> operand

Bitwise AND

- & compares two bits
- Yields a value of 1 if both bits are set, 0 otherwise.

Base 10	Operator	128's	64's	32's	16's	8's	4's	2's	1's
255		1	1	1	1	1	1	1	1
170	&	1	0	1	0	1	0	1	0
170	=	1	0	1	0	1	0	1	0

```
//What will be output by the following?
int result = 5 & 6;
std::cout << result << std::endl;
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Bitwise OR

- | compares two bits
- Yields a value of 1 if any bits are set, 0 otherwise.

Base 10	Operator	128's	64's	32's	16's	8's	4's	2's	1's
204		1	1	0	0	1	1	0	0
170	|	1	0	1	0	1	0	1	0
238	=	1	1	1	0	1	1	1	0

```
//What will be output by the following?
int result = 5 | 6;
std::cout << result << std::endl;
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Bitwise NOT

- ~ flips each bit
- A value of 1 become a value of 0 and a value of 0 becomes a 1.

Base 10	Operator	128's	64's	32's	16's	8's	4's	2's	1's
204	~	1	1	0	0	1	1	0	0
51	=	0	0	1	1	0	0	1	1

```
//What will be output by the following?
int result = ~6;
std::cout << result << std::endl;
```

Bitwise XOR

- ^ (exclusive or) compares two bits
- Yields a value of 1 if bits are different, 0 if same.

Base 10	Operator	128's	64's	32's	16's	8's	4's	2's	1's
204		1	1	0	0	1	1	0	0
170	^	1	0	1	0	1	0	1	0
102	=	0	1	1	0	0	1	1	0

```
//What will be output by the following?
int result = 5 | 6;
std::cout << result << std::endl;
```

Bitmasks

- Bitmasks are used to perform a single bitwise operation on a set of bits. This is often used to represent multiple Boolean values.

- We can use bitmasks to turn on or off various bits within a set of bits. Also to check the state of a bit within a set of bits.

Bitmasks reduce data storage

- In Doom there were a limited number of weapons.

- A player either has a weapon in their inventory or they don’t

- We could use a boolean array to store this info
  - 8 weapons * 1 (size of bool) = 8 bytes

- Using a bitmask makes more sense
  - 8 bits (one for each weapon) = 1 byte

Bit Shifting

- The << and >> operators shift bits to the left or right, by a value given after the operator.
- For example: 6 << 1
  - This means that the bits making up the number six will be shifted one space to the left
  - 0000 0110 becomes 0000 1100 (12)

Bit Shifting

- Digits that are shifted off the end do not wrap around – they are lost

Operation	Result
1 << 6	Shifts the value 1 six bits to the left, yields 64
66 >> 1	Shifts the value 66 one bit to the right, yields 33
67 >> 1	Shifts the value 67 one bit to the right, yields 33 due to truncation

Shift Examples (continued)

Base 10	Operation	128's	64's	32's	16's	8's	4's	2's	1's
1	<< 1	0	0	0	0	0	0	0	1
2	=	0	0	0	0	0	0	1	0
26	<< 4	0	0	0	1	1	0	1	0
160	=	1	0	1	0	0	0	0	0
160	>> 7	1	0	1	0	0	0	0	0
1	=	0	0	0	0	0	0	0	1

Conclusion

- For more information on binary maths and Boolean logic refer to :
- C++ Primer 6th Edition : Appendix E
- http://dev.tutsplus.com/tutorials/number-systems-an-introduction-to-binary-hexadecimal-and-more--active-10848

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX