Variables and Data Representation

Data and Memory

- Each piece of data used in a program must be stored in the computer's memory.
- The term variable is used to refer to the place in memory where a data item is stored.
- Each variable has a name, a datatype and a fixed size.

Data and Memory

- Computer memory is composed of electronic circuits.
- Each circuit is either on or off.
- All data must be encoded in some way to be represented in a series of “on”s and “off”s.

Binary Numbers

- The binary digits 1 and 0 can be used to represent a circuit that is on or off.
- Each binary digit is called a bit.
- The positional number system that uses 0 and 1 as digits and base 2 is called the binary number system.

Unsigned Integers

Value of Each Position								
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	In Base 2 Exponential Notation
128	64	32	16	8	4	2	1	In Decimal Notation
Binary Examples								Decimal Equivalent
1	1	0	0	0	1	1	1	128 + 64 + 4 + 2 + 1 = 199
0	0	0	1	0	1	0	1	16 + 4 + 1 = 21
1	0	1	0	1	0	0	0	128 + 32 + 8 = 168

Unsigned Integers

- Practice Examples
  - 5 = ?
  - 13 = ?
  - 27 = ?
  - 31 =?
  - 63 = ?
  - 00010011 = ?
  - 00111100 = ?
  - 00001111 = ?
  - 01111111 = ?
  - 01010101 = ?

Unsigned Integers

- 255 is the largest unsigned integer that can be represented in 8 bits.
  - Can you develop a formula that uses an exponent to express this?
  - Can you generalize this formula to give you the largest unsigned integer that can be represented in 16 bits? 32 bits? N bits?
  - Why does this matter to you as a programmer?

Signed Integers

- By definition, -1 is the number that can be added to 1 to get 0.

```
  00000001
+ ????????
  --------
  00000000
```

- Before you can answer that question you have to be able to add binary numbers.

Binary Addition

- 1 + 1
  - Does not equal 2
  - It equals 10

$$\begin{array}{rr} & 000000^{1}01 \\ + & 000000\ \ 01 \\ \hline & 000000\ \ 10 \end{array}$$

Binary Addition

$$
\begin{array}{r}
 {}^{}\,0\,0\,0\,{}^{1}0\,{}^{1}1\,{}^{1}0\,{}^{1}1\;1 \\
+\;0\,0\,0\;\;0\;\;0\;\;1\;\;1\;\;1 \\
\hline
 0\,0\,0\;\;1\;\;0\;\;0\;\;1\;\;0
\end{array}
$$

$$
\begin{array}{r}
 {}^{}\,0\,0\;\;0\;\;{}^{1}1\,{}^{1}0\,{}^{1}1\,{}^{1}1\;1 \\
+\;0\,0\;\;1\;\;0\;\;0\;\;1\;\;0\;\;1 \\
\hline
 0\,0\;\;1\;\;1\;\;1\;\;1\;\;0\;\;0
\end{array}
$$

Binary Addition

- Practice Examples
  - 00010101 + 00001101 = ?
  - 00111110 + 00101001 = ?
  - 00011111 + 00000001 = ?
  - 01010101 + 00111111 = ?
  - 00000001 + 11111111 = ?

Signed Integers

- The left most bit is the sign bit.
- 11111111 = -1
  - It's the largest negative integer.
- 10000000 = - 128
  - It's the smallest negative integer that can be represented in 8 bits.
  - Can you generalize that to 16 bits? 32 bits? N bits?
  - What's the formula for the largest positive signed integer?

Signed Integers

- The notation for negative binary numbers is called 2's complement.
- To represent a negative # in 2's complement
  1. Convert the absolute value of the # to its binary representation
  2. Flip each of the bits in the number from step 1 to the "opposite" binary digit.
  3. Add 1 to the result from step 2.
- Let's verify that with –1 and – 128.

Signed Integer

- Practice Problems
  - -3 = ?
  - -18 = ?
  - -25 = ?
  - 11001000 = ?
  - 10111101 = ?
  - 11111110 = ?

Hexadecimal Numbers

Value of Each Position				
16^3	16^2	16^1	16^0	In Base 16 Exponential Notation
4096	256	16	1	In Decimal Notation
Hex/Binary Examples				Decimal Equivalent
0	7	A	3	(7*256) + (10*16) + 3 = 1955
0000	0111	1010	0011	Each hex digit is 4 binary digits!
0	F	C	1	(15*256) + (12 * 16) + 1 = 4033
0000	1111	1100	0001	
1	B	4	D	(1*4096) + (11*256) + (4*16) + 13 = 6989
0001	1011	0100	1101	

Hexadecimal Numbers

- Practice Problems
  - 01011101 = ? hex
  - 01110111 = ?
  - 19 = ? binary
  - 89 = ?
  - A6 = ?
  - 23 = ?
  - 51 = ?
  - CB = ?

Real Numbers

Sign (1 bit)	Exponent (8 bits)	Mantissa (23 bits)	Sign * Mantissa * 2 Exponent
0	00001010	00000000000000000011010	$1*26*2^{10} = 26*1024 = 26624$
0	11110110	00000000000000000011010	$1*26*2^{-10} = 26*(1/1024) = 26 * 0.0009765625 = 0.025390625$

Real Numbers

- It’s not necessary to be able to know the details of the representation!
- It is important to know that
  - Real numbers stored in a computer are only approximations
  - Double precision floating point numbers are closer approximations than single precision numbers.
  - Calculations involving real numbers can result in round off errors.

Characters

- A variety of character sets can be used to encode character data
  - ASCII used in DOS programs
  - Unicode used in modern programming languages
- Chart in packet shows subset of ASCII character set
- A has value of 65
  - 00100001 is stored in memory when you type A at the keyboard