CSE 351 Section 1

Binary, C
Fall 2022

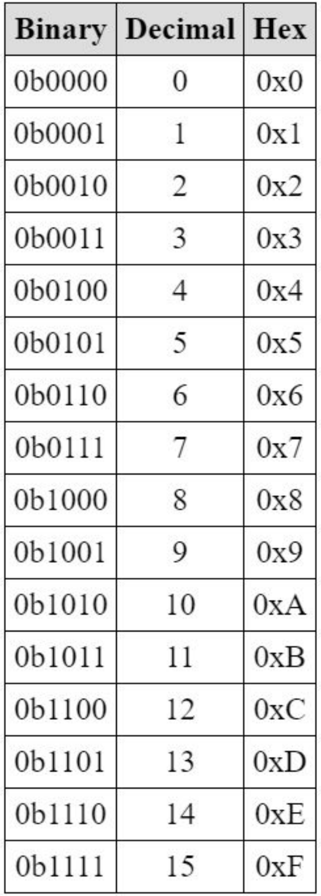
Introductions

Icebreaker Time!

- Let’s get to know each other!
- <activity description and instructions>

Binary and Hexadecimal

- The (decimal) value of the digit *d* in position `i` in base *b* is: $d \text{ x } b^{i}$
  - Digits are numbered starting from 0 from right-to-left
- Pay special attention to base indicators
  - Subscripts: 8, 10_2, BA_{16}
  - Prefixes: `0b` (binary), `0x` (hex)
- Common pitfalls
  - Arithmetic in hex
  - Digit widths and leading zeros



Binary	Decimal	Hex
0b0000	0	0x0
0b0001	1	0x1
0b0010	2	0x2
0b0011	3	0x3
0b0100	4	0x4
0b0101	5	0x5
0b0110	6	0x6
0b0111	7	0x7
0b1000	8	0x8
0b1001	9	0x9
0b1010	10	0xA
0b1011	11	0xB
0b1100	12	0xC
0b1101	13	0xD
0b1110	14	0xE
0b1111	15	0xF

Converting TO Decimal

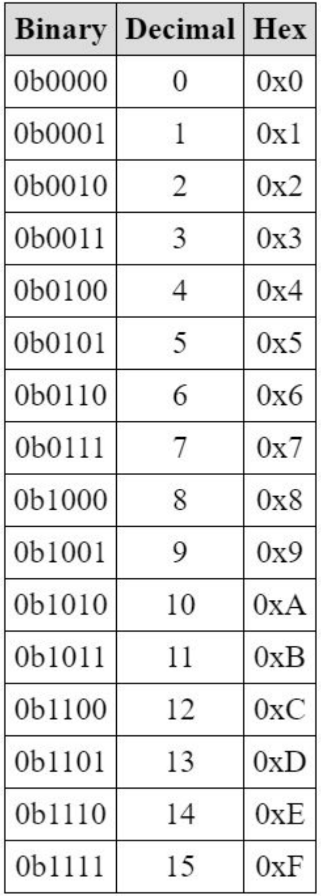
- Use the formula: $\boldsymbol{d \times b^i}$
- Let's try it: Convert 345_8 into decimal:

Converting FROM Decimal

- Remember: write down powers of the base, it’s like long-division
- Let’s try it: Convert 234 into base 7 (powers of 7 are 1, 7, 49):

Converting Binary TO Hexadecimal

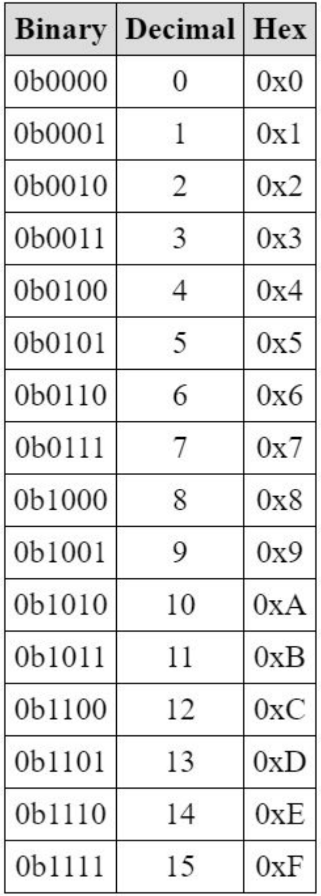
- Convert each group of 4 binary digits into one hex digit
- Let’s try it: Translate 0b111100 into hex:



Binary	Decimal	Hex
0b0000	0	0x0
0b0001	1	0x1
0b0010	2	0x2
0b0011	3	0x3
0b0100	4	0x4
0b0101	5	0x5
0b0110	6	0x6
0b0111	7	0x7
0b1000	8	0x8
0b1001	9	0x9
0b1010	10	0xA
0b1011	11	0xB
0b1100	12	0xC
0b1101	13	0xD
0b1110	14	0xE
0b1111	15	0xF

Converting Binary FROM Hexadecimal

- Convert each hex digit into binary
- Let’s try it: Translate 0x1AB into binary:



Binary	Decimal	Hex
0b0000	0	0x0
0b0001	1	0x1
0b0010	2	0x2
0b0011	3	0x3
0b0100	4	0x4
0b0101	5	0x5
0b0110	6	0x6
0b0111	7	0x7
0b1000	8	0x8
0b1001	9	0x9
0b1010	10	0xA
0b1011	11	0xB
0b1100	12	0xC
0b1101	13	0xD
0b1110	14	0xE
0b1111	15	0xF

Binary Practice Slide (Worksheet)

Binary	**Decimal**	**Hexadecimal**
0b10010011		
		0x16
	63	
0b100100		
		0xC30
	0	
		0xBAD
	437	

Binary	Decimal	Hexadecimal
0b10010011	$2^7 + 2^4 + 2^1 + 2^0 = 147$	0x93
0b10110	$116^1 + 616^0 = 22$	0x16
0b111111	63	0x3F
0b100100	$2^5 + 2^2 = 36$	0x24
0b110000110000	$1216^2 + 316^1 = 3120$	0xC30
0b0	0	0x0
0b101110101101	$1116^2 + 1016^1 + 1316^0 = 2989$	0xBAD
0b110110101	437	0x1B5

Number Representation

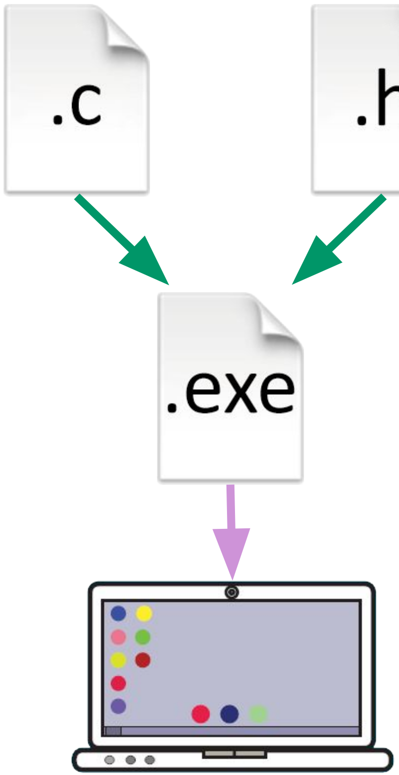
- A single numeral can *represent* many different values/things as long as you know the proper *encoding scheme*
  - The encodings may be arbitrarily chosen by the designer
- Representation limits: need to use a sufficient number of bits to cover the entire range of values/things to be represented
- Some encoding schemes we will cover in this class:
  - Unsigned and signed integers
  - Floating point numbers
  - Characters
  - Data locations

C Workflow

1) Edit source file(s)

2) Build executable

3) Run process



Text editor
(*e.g.*, `vim`, `emacs`)

Compiler
(*e.g.*, `gcc`)

Command line
(*e.g.*, `./a.out`)

Compilation Options

Compilation command:

```
gcc -Wall -g -std=c18 -o foo foo.c
```

- `-W` turns on compiler warnings (all of them)
- `-g` turns on debugging symbols
- `-std` specifies which “standard” of C we are using
- `-o` changes the name of the resulting executable
- `foo.c` is the source file being compiled

Compiling and Executing Slide (Ed Lessons)

printf Format Specifiers

The printf function prototype:

```
    int printf(const char* format, ... );
```

- %d for signed integers
- %u for unsigned integers
- %f for floating point numbers
- %s for "string"
- %x for hexadecimal
- %p for pointer

printf Slide (Ed Lessons)