COMPUTER LOGIC AND DIGITAL CIRCUIT DESIGN (PHYS306/COSC330): UNIT 1.3

Jordan Hanson February 27, 2018

Whittier College Department of Physics and Astronomy



UNIT 1.3 SUMMARY - WORKING WITH BINARY

Reading: Digital Fundamentals (DF) Ch. 2 (see Moodle)

- 1. Number representation
- 2. Binary conversions
- 3. Binary arithmetic
- 4. The floating-point system
- 5. Hexadecimals, Binary-Coded Decimals (BCD), Gray codes, and ASCII

Homework: exercises 5-16, 19-32, 35-38, 45, 53, 58 Ch. 2 (DF) (two weeks).

Questions:

- A simple question: how many students do we have in this class?
- Una simple pregunta: ¿Cuántos estudiantes tenemos en esta clase?
- Une question simple: Combien d'étudiants est-ce que nous avons dans cette classe?

What languages do computers speak? How can we store and transmit numbers through circuits? (We cannot use voltage magnitudes).

Consider the number 37

Numbers

- · 0d37
- · 0b100101
- · 0x25

Expanded notation

•
$$3 \times 10^{1} + 7 \times 10^{0}$$

•
$$1 \times 2^5 + 1 \times 2^2 + 1 \times 2^0$$

•
$$2 \times 16^1 + 5 \times 16^0$$

Consider the number 412

Numbers

- · 0d412
- · 0b110011100
- · 0x100

Expanded notation

•
$$4 \times 10^2 + 1 \times 10^1 + 2 \times 10^0$$

$$\cdot 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2$$

•
$$1 \times 16^2$$

Number representations - Digits, weights, and a common base

- A number is written with digits that have weights.
- · A weight is a power of a base.
- At right, the base is ten, and the weights are 10², 10¹, and 10⁰.
- The digits are four, one, and two.
- The digits cannot represent numbers larger than the base.

Expanded notation

• $412 = 4 \times 10^2 + 1 \times 10^1 + 2 \times 10^0$

Number representations - (Aside) please use scientific notation, and here is why: digit minimization!

- Scientific notation factors the largest weight.
- Results in weights that are less than one.
- Weights that are less than one go to the right of the decimal point.
- Return here with floating-point representation.

Expanded notation

- 412,000,000 = $4 \times 10^8 + 1 \times 10^7 + 2 \times 10^6 +$ $0 \times 10^5 + 0 \times 10^4 + 0 \times 10^3 +$ $0 \times 10^2 + 0 \times 10^1 + 0 \times 10^0$
- 412,000,000 = $4.12 \times 10^8 = (4 \times 10^0 + 1 \times 10^{-1} + 2 \times 10^{-2}) \times 10^8$

Number representations - (Aside) please use scientific notation, and here is why: arithmetic operations with large numbers!

- 1. $4200 \times 4200 = (4.2 \times 10^3)^2 = (4.2)^2 \times 10^6 \approx 16 \times 10^6$
- 2. $\approx 17.6 \times 10^6$ if you account for the 0.2 ...

- 1. $4000/3000 = 4 \times 10^3 \times \frac{1}{3} \times 10^{-3} = \frac{4}{3}$
- 2. $\frac{4}{3} \approx 1.33$

Expand the following numbers to expanded decimal notation:

- · -10.432
- · 800,000,144

Expand the following numbers to expanded binary notation:

- · 10011010
- · 11110000

Convert the following decimal numbers to binary notation:

- · 260
- 560

Volunteer to board? - Key is explaining how you did the binary conversions

How did you do the conversions to binary? Is there a systematic what to do this?

- Successive Approximation like a number puzzle (Sum of Weights Method)
- · Successive Division Method Example of an algorithm

Successive approximation technique (does this remind you of doing division in your head?) $260...2^8 = 256$. Now we need four more... $4 = 2^2$. So $2^8 + 2^2 = 0b10000100$

What is 328 divided by 3? Ok try 100 because three times one hundred is close...

Successive Division Method

$$0d412 = 0b110011100 = 2^8 + 2^7 + 2^4 + 2^3 + 2^2$$

Algorithm:

- 1. Divide the decimal number by 2, and write down the remainder. This is the *least-significant bit* or LSB.
- 2. Keep dividing and recording the remainders in order, until you reach a dividend of 1.
- 3. 1/2 = 0r1, so the most-significant bit, or MSB, is always 1.

Convert 412 to binary using the successive division method.

Convert the numbers at right to binary.

•
$$2^0 = 1$$

•
$$2^1 = 2$$

•
$$2^2 = 4$$

•
$$2^3 = 8$$

•
$$2^4 = 16$$

•
$$2^5 = 32$$

•
$$2^6 = 64$$

•
$$2^7 = 128$$

Note: how many bits do you need for that last one? For binary, show that the highest representable number with n bits is $2^n - 1$.

Notice that we get a *repeating* pattern when we compare the conversion of 189 and 93. Why?

In decimal notation, we represent numbers $\in [0-1]$ with digits to the right of the decimal point.

In expanded notation:

$$42.42 = 4 \times 10^{1} + 2 \times 10^{0}$$
 . $4 \times 10^{-1} + 2 \times 10^{-2}$

We have a similar notation in other number systems:

$$101.11 = 2^2 + 2^0 \cdot 2^{-1} + 2^{-2}$$

Successive Multiplication Method

$$0d0.48 = 0b0.011110101 = 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} + 2^{-7} + 2^{-9}$$
 Algorithm:

- 1. Multiply the decimal fraction by 2, and write down the *carry* (the 1 to the left of the decimal point, else 0). This is the *most-significant bit* or MSB.
- 2. Keep multiplying and recording the carries in order, to the desired precision.
- 3. Watch for repeating patterns.

Convert 0.48 to binary using the successive multiplication method until you identify a repeating pattern.

Convert the numbers at right to binary.

•
$$2^0 = 1$$

•
$$2^1 = 2$$

•
$$2^2 = 4$$

•
$$2^3 = 8$$

•
$$2^4 = 16$$

•
$$2^5 = 32$$

•
$$2^6 = 64$$

•
$$2^7 = 128$$

- 1. 0.64
- 2. 0.125

Which result should have a smaller number of digits, and why?

Binary arithmetic, like decimal arithmetic, relies upon *carries* and *borrows*. For example in decimal:

•
$$8 + 7 = 5c1 = 15$$

•
$$3 + 4 = 7c0 = 7$$

$$\cdot 8 - 2 = 6b0 = 6$$

$$\cdot$$
 10 - 5 = (0*b*10) - 5 = 5

$$\cdot$$
 13 - 7 = (0*b*10) - 4 = 6

In binary there are limited combinations, so we may form a set of addition and subtraction rules that describe all possibilities:

$$\cdot 0 + 0 = 0$$

$$\cdot 1 + 0 = 1$$

$$\cdot 0 + 1 = 1$$

$$\cdot 1 + 1 = 0c1$$

$$\cdot 0 - 0 = 0$$

$$\cdot 1 - 0 = 1$$

•
$$0-1=0b1$$

$$\cdot 1 - 1 = 0$$

In the fourth rule for the addition set, the carry works the same way as a decimal carry: we add that 1 to the digit corresponding to the next power of 2. Similarly, in the third rule of the subtraction set, we subtract the 1 on the left side of the equation from a 1 from the digit corresponding to the next highest power of 2 available.

Complete the following additions and subtractions:

•
$$10 + 11$$

•
$$111 + 1$$

•
$$111 + 111$$

•
$$1010 + 101$$

•
$$10 - 11$$

Binary multiplication stems from another set of rules:

•
$$0 \times 0 = 0$$

•
$$1 \times 0 = 0$$

•
$$0 \times 1 = 0$$

•
$$1 \times 1 = 1$$

Multiply:

1.
$$10 \times 10$$

2.
$$110 \times 10$$

Binary division is exactly the same as decimal long division. Let's work these examples on the board:

- · 145/12
- · 1101/101
- · 45/3
- · 111/10

Of what logic operation does binary multiplication remind you?

We need a few tools to improve our arithmetic techniques in order to install these functions into circuits. Consider two actions:

- NOT to the bit sequence of a number: 1's compliment
- Add 1 to the 1's compliment: 2's compliment

Logical operator representation of these actions for 8-bits:

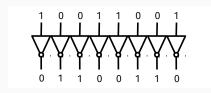


Figure 1: Representation of 1's compliment.

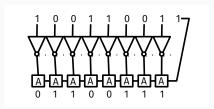


Figure 2: Representation of 2's compliment.

Exercises, and a realization for the purpose of 2's compliment.

Take the 1's compliment of the right-hand number, and complete the addition:

$$\cdot 111 + (000)$$

•
$$111 + (111)$$

Take the 2's compliment of the right-hand number, and complete the addition:

$$\cdot$$
 111 + (000)

•
$$111 + (111)$$

Same, but drop the MSB:

$$\cdot$$
 1010 + (1010)

For what purpose are we going to use the 2's compliment?

The answer: representing negative numbers. For 2's compliment form signed binary numbers, the MSB is the **sign** bit. To convert to decimal, give the MSB a negative weight:

Example: 10101010

This is a negative number because the MSB is a 1. Converting in expanded form: $-2^7 + 2^5 + 2^3 + 2^1 = -128 + 32 + 8 + 2 = -86$.

Consider the effect on the maximum *range* of binary numbers after sacrificing the MSB to flag numbers less than zero. For 8-bits unsigned numbers, the range is [0-255]:

n-bits	8	16
Lowest unsigned	0	0
Highest unsigned	255	65535
Lowest signed	-128	-32768
Highest signed	127	32767

Table 1: The ranges of unsigned and signed numbers using *n* digits.

By changing the role of the MSB, we are not reducing the range of numbers, just its location. Related to conservation of information...

2's compliment, signed number representation and conversion, arithmetic with negative numbers

- 1. Write -278 in binary, 2's compliment form.
- 2. Convert -302 and 65 each to binary, and add them.
- 3. This number is in 2's compliment form: 11001101. Convert it to decimal.

Who has taken software programming?¹

In software code, we allocate memory for variables which could be decimal numbers. In C++, this might look like

```
float var = 15.237;

double var = 1.61E - 19;

unsigned int var = 8;

long int var = 320000000;
```

We will now learn the origin and purpose of these keywords.

¹Normally, this is a pre-requisite for this course, but in this innaugural year we are more flexible.

A floating-point number is a general system for storage of wide ranges of real numbers in binary. The structure of a floating-point number:

N	Binary	Scientific Notation	Sign bit	Exponent	Mantissa
-10.23	-1010.0011101	-1.0100011101×2^3	1	10000010	010001110100000000000000
22.3	10110.01001100	$1.011001001100 \times 2^4$	0	10000011	011001001100000000000000

Table 2: For **float** or **single** precision, the number of bits allowed for the sign, exponent and mantissa is **32**. The exponent is 3 + 127 in binary. We *bias* the exponent by 127 so that the range of exponents can be both negative and positive.

• Float or single precision: 32 bits

· Double precision: 64 bits

The structure of a floating-point number:

N	Binary	Scientific Notation	Sign bit	Exponent	Mantissa
22.3	10110.01001100	$1.011001001100 \times 2^4$	0	10000011	011001001100000000000000
6.825	1010.11010011	1.01011010011×2^3	0	10000010	01011010011000000000000

Table 3: Note that the MSB in scientific notation is **dropped** in the mantissa. Why?

Convert the following numbers into **floats**:

- · 8.125
- -4.0625

Addition and subtraction with signed numbers: either number can be positive/negative, and the larger/smaller number by magnitude. Thus, 4 combinations are possible:

	Negative	Positive
Larger	А	В
Smaller	С	D

Table 4: Four categories, two numbers, so 6 initial pairs: AB AC AD BC BD CD. However, AB and CD aren't real choices, so the list is: AC AD BC BD.

Bottom line: both negative, both positive, one negative one positive (two cases)

The results:

- 1. Both negative → discard the carry
- 2. One positive, one negative, and the negative number has a larger magnitude
- One positive, one negative, and the positive number has a larger magnitude → discard the carry
- 4. Both positive

Example of negative-negative case: -13 - 9 = -22.

- 1. Identify case: both are negative, with magnitudes of 13 and 9.
- 2. Decide *n*-bits. Ensure that $2^n 1$ is much larger than other numbers. Let's choose n = 8.
- 3. Convert the first magnitude (13) to 2's compliment form: 2s(00001101) = 11110011
- 4. Convert the second magnitude (9) to 2's compliment form: 2s(0001001) = 11110111
- 5. Add them, and in this case, drop the final carry: 11101010
- 6. Check: 2s(11101010) = 00010110 = 22

ASCII

Why do we use Hexadecimals?

https://youtu.be/DqfnMzDhi38

In case you get your butt stranded on Mars, and need to devise a rag-tag communication system from a camera-pole, is why.

Astronaut Watney needs a way to cram more information into shorter-length words...hexadecimals

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	1111	7
8	1000	8
9	1001	9
10	1010	Α
11	1011	В
12	1100	С
13	1101	D
14	1110	E
15	1111	F

Table 5: Decimal, binary and Hexadecimal representations of numbers. What do you notice that is different between the representations?

Because $2^4 = 16$, we may group bits in bit sequences in groups of four, and read out the hexadecimal conversion:

$$0b11010010 = 1101\ 0010 = (13)\ (2) = 0xD2$$

On page 71 of DF: "It should be clear that it is much easier to deal with a hexadecimal number than with the equivalent binary number. Since conversion is so easy, the hexadecimal system is widely used for representing binary numbers in programming, printouts, and displays."

It is probably important. What follows is a few examples of how I've encountered hexadecimals.

```
set key at 500,73 box on samplen 2 spacing 1.5 font "Courier,16" width 2
set xrange [1:1e3]
set yrange [0:180]
set format x "10^{%T}"
set rmargin 8
set lmargin 13
set tmargin 2
set bmargin 6
set xlabel "Frequency [MHz]" font "Courier, 28" offset 0, -2,0
set ylabel "Phase (deg)" font "Courier, 28" offset -5,0,0
set vtics scale 2 font "Courier.28"
set xtics scale 2 font "Courier, 28" offset 0,-1,0
set style line 6 linecolor rgb "#000000" linewidth 5.000 dashtype 3 pointtype 2
set style line 7 linecolor rgb "#BBBBBB" linewidth 5.000 dashtype 3 pointtype 2
set linestyle 1 lc rgb "#000000" lw 5
set linestyle 2 lc rgb "#444444" lw 5
set linestyle 3 lc rgb "#888888" lw 5
set linestyle 4 lc rgb "#BBBBBB" lw 5
set linestyle 5 lc rgb "#DDDDDD" lw 5
set terminal postscript color enhance
```

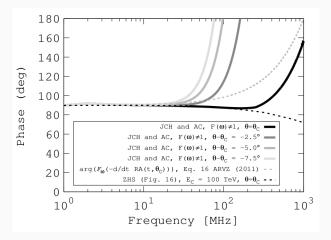


Figure 3: A figure from a recent paper we published about neutrino signals in ice. The hexadecimals were used to encode the line-colors.

A math problem: If the RGB (red, green, blue) codes have the following format 0xA1B2C3, where each hex word (two characters) represents an amount of red, green, and blue (sequentially), how many distinct shades of each are technically possibe? How many colors are therefore possible (even if a standard monitor could not resolve them)?

Volunteer to board?

In graduate school, I was responsible for designing and deploying a housekeeping integrate circuit (HK board). The task of this device was to control which subsystems are activated in an ARIANNA station, managing power consumption.

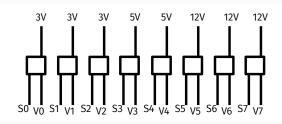


Figure 4: Symbolic representation of a relay bank. *Raising* the various signals S_i to a HIGH state connects the sub-system voltages V_i to power sources at the given voltages, and setting them to LOW state powers down the subsystems.

In software, the C-code interpreted a bit-mask in hexadecimal, which the single-board computer (SBC) would send to the digital IO (input/output), connected to the S_i lines. For example, 0b11100000 activated only the 3V subsystems.

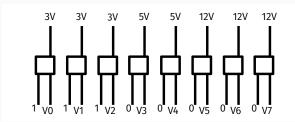


Figure 5: The S_i lines were controlled by the system digital IO. The above bit-mask was sent from the C-code through the digital IO.

However, to save space on a hard drive that only had 4 GB of space, with potentially thousands of defined functions that had to send bit masks, the manufacturer designed the base bitmasking function input as a hex word (two hex characters). In this case, it would have been 0xE0 = 0b11100000.

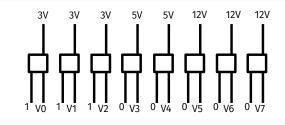


Figure 6: A bitmask of D0.

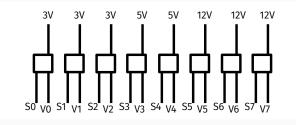


Figure 7: The general case for the relay bank.

- 1. What hex word would activate only the 5V systems?
- 2. What hex word would activate only the 12V sytems?
- 3. What hex word would activate just S_1 , S_4 , and S_7 ?

Other codings with binary and hexadecimals:

- Binary coded decimals
- Gray codes
- ASCII

Binary coded decimals (BCD):

The 8421 BCD system is often used in digital displays, readouts and number pads. It combines the familiarity of decimal digits with the utility of binary.

For example:

 $0d234 = 0010\ 0011\ 0100\ in\ BCD.$

0 0000

1 0001

0010

3 0011

4 0100 5 0101

6 0110

7 0111 8 1000

8 1000 9 1001

Table 6: The 8421 coding for BCD. Each four-digit bit sequence represents one decimal digit.

Quick exercise: Write your first name as a decimal string (A = 1, B = 2, etc.) and convert it to BCD.

Decimal digit	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Table 7: The 8421 coding for BCD. Each four-digit bit sequence represents one decimal digit.

The Gray Code:

The Gray code is a non-arithmetic code word system, related to binary, that minimizes transition errors by restricting one bit change per transition.

Binary to Gray Code conversion

- 1. The MSB is conserved.
- Add consecutive binary bits starting with the MSB to obtain Gray Code bits.
- 3. Discard carries.

Binary	Gray Code
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101

Table 8: The 4-bit Gray Code.

The Gray Code:

The Gray code is a non-arithmetic code word system, related to binary, that minimizes transition errors by restricting one bit change per transition.

Gray Code to Binary conversion

- 1. The MSB is conserved.
- Add newest binary bit to next Gray Code bit to obtain next new binary bit.
- 3. Discard carries.

Binary	Gray Code
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101

Table 9: The 4-bit Gray Code.

Other codings with binary and hexadecimals:

- Binary coded decimals
- Gray codes
- ASCII American Standard Code for Information Interchange

0x01 1 SoM Start of heading 0x21 33 1 0x41 65 A 0x61 97 0x02 23 TX Start of text 0x22 34 0x43 67 C0x63 99 0x04 4 ENT End of text 0x23 35 # 0x43 67 C0x63 99 0x05 5 ENV Enquiry 0x25 37 8 0x44 68 D 0x64 10 0x06 6 ACX Akmowledge 0x25 37 8 0x44 67 F 0x66 102 0x07 7 BEL Bell 0x27 39 0x47 71 0x66 102 0x08 8 BS Backspace 0x28 40 0 0x48 72 H 0x66 102 0x09 9 AB Novilia 19 Vertical tab 0x28 42 0x44 74<			Char		Hex		Char	Hex		Char			Char
0x02 2 STX Start of text		0					Space			6		96	
0x02 2 StA Staff Of text		1	SOH							A		97	a
0x04 4 007 End of transmission 0x24 36 5 0x44 68 D 0x64 100 0x05 5 END Enquiry 0x25 37 3 0x45 69 5 0x65 0x65 0x05 0x06 0x00 0x25 37 3 0x45 69 5 0x65 0x65 0x06 0x00 0x07 7 0x17 0x18 0x27 39 0 0x46 70 F 0x66 100 0x09 9 7 8 End End 0x28 41 0 0x48 73 F 0x68 100 0x00 0	0x02	2	STX		0x22	34		0x42	66	В	0x62	98	b
0x95 5 ENQ Enquiry 0x25 37 1 0x45 69 E 0x65 10 0x06 6 ACX Acknowledge 0x26 38 4 0x46 70 0x66 10 0x07 7 BELL Bell 0x27 39 0x47 71 G 0x67 103 0x08 9 78 Borlander 0x22 40 0 0x48 72 I 0x68 104 0x08 9 78 Borlander 0x22 44 0 0x47 71 G 0x68 104 0x00 10 LF Nev line 0x20 44 0x40 74 J 0x6a 10 0x00 12 FF Form Feed 0x20 44 0x40 76 L 0x60 10 0x00 14 80 Shift out 0x20 45 0x40 77 0x66 11 0x10	0x03	3	ETX	Bild 01 00110	0x23		#	0x43	67	C		99	C
0x06 6 ACK Acknowledge	0×04	4	EOT	End of transmission	0x24	36	\$	0x44	68	D	0x64	100	d
0x07 7 BELL Pell	0x05	5	ENQ		0x25	37	8	0x45	69	E	0x65	101	е
0.00 0.00												102	f
0x09 9 7AB Norizontal tab		7	BELL				1			G		103	g
0x0A 10 LP New line	80x0	8	BS		0x28	40	(0x48	72		0x68	104	h
0x00 11 0x1 VP Vertical tab 0x28 43 4 0x48 75 K 0x68 107 0x0C 12 FF Form Feed 0x2C 44 , 0x46 75 K 0x68 107 0x0C 12 FF Form Feed 0x2C 44 , 0x46 75 K 0x66 107 0x0C 13 FF Form Feed 0x2C 44 , 0x46 75 K 0x66 108 0x0C 14 SO Shift out 0x2E 46 . 0x46 78 N 0x66 110 0x0F 15 ST Shift in 0x10 16 DLE Data link escape 0x3D 48 0 0x5D 80 P 0x7D 112 0x10 16 DLE Data link escape 0x3D 48 0 0x5D 80 P 0x7D 112 0x12 18 DC2 Device control 1 0x31 49 1 0x51 81 0 0x71 17 0C 112 0x12 18 DC2 Device control 2 0x32 50 2 0x52 82 R 0x72 114 0x13 19 DC2 Device control 3 0x33 51 0x53 83 S 0x73 151 0x14 20 DC4 Device control 4 0x34 52 4 0x54 84 T 0x74 116 0x15 21 XM Regative ack 0x35 53 5 0x55 85 0x55 85 0x75 185 0x16 22 SYN Synchronous idle 0x36 54 6 0x56 86 V 0x75 112 0x16 22 SYN Synchronous idle 0x36 54 6 0x56 86 V 0x75 112 0x16 22 SYN Synchronous idle 0x36 54 6 0x56 86 V 0x75 112 0x16 22 SYN Synchronous idle 0x36 54 6 0x56 86 V 0x75 112 0x16 22 SYN Synchronous idle 0x36 54 6 0x56 86 V 0x75 112 0x16 27 SYN 0x75 125 0x18 27 SYN 0x75 1	0x09	9	TAB		0x29	41)	0x49	73	I	0x69	105	i
0x0c 12 PF Form Feed			LF				*					106	j
0xD0 1 CR Carriage return 0xD 45 - 0x4D 77 M 0xED 10 0x0E 14 50 Shift out 0x2E 46 . 0x4E 78 M 0x6E 10 0x0F 15 Shift in 0x2E 46 . 0x4E 78 M 0x6E 11 0x10 16 DLE Data 1nh escape 0x3D 48 0 0x5D 80 P 0x7D 112 0x11 17 DC Device control 1 0x31 49 0x51 81 0 0x71 13 0x13 19 Device control 2 0x33 50 2 0x52 82 R 0x72 14 0x13 49 Device control 2 0x33 51 0x58 83 0x73 15 0x15 21 Ax8 Regativa eac 0x35 53 0	0x0B		VT			43	+	0x4B		K	0x6B	107	k
0x0E 14 80 Shift out 0x2E 46 . 0x4E 78 N 0x6E 10 0x10 14 80 Shift out 0x2E 47 . 0x4F 79 0 0x6F 11 0x10 16 DLE Data link escape 0x2E 47 . 0x4F 79 0 0x6F 11 0x10 16 DLE Data link escape 0x30 48 0 0x50 80 P 0x70 11 0x12 18 DC2 Device control 1 0x31 49 1 0x51 81 0x51 81 0 0x71 131 0x12 18 DC2 Device control 2 0x3E 50 2 0x52 82 R 0x72 114 0x13 19 DC3 Device control 3 0x33 51 3 0x53 83 83 0x73 115 0x14 20 TM NR Negative ack 0x35 53 5 0x55 85 U 0x75 117 0x15 22 STM Synchronous side 0x35 53 5 0x55 85 U 0x75 117 0x17 116 0x17 23 ETB End transmission block 0x37 55 7 0x57 87 W 0x75 117 0x18 22 STM Cancel 0x39 55 7 0x55 87 U 0x75 117 0x17 118 0x18 22 STM Substitute 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 27 FSC Escape 0x38 59 ; 0x56 90 0x50 80 1 (0x78 123 0x16 28 File separator 0x30 61 0x56 85 0x50 93 1 (0x76 123 0x16 28 5 File separator 0x30 61 0x56 95 0x56 93 1 0x76 123 0x18 29 0x58 90 87 0x70 125 0x18 29 0x58 90 87 0x70 125 0x18 20 0x18 20 8 Record separator 0x30 61 0x56 93 0x56 94 0x70 125 0x18 20 0x56 90 0x50 94 0x70 125 0x50 94 0x70 125 0x56 94 0x70 125 0x70 125 0x56 94 0x70 125 0x56 94 0x70 125 0x56 94 0x70 125 0x56 94 0x70 125 0x	0x0C	12	FF		0x2C	44		0x4C	76	L	0x6C	108	1
0x0F 15 31 Shift in	$0 \times 0 D$	13	CR		0x2D	45	-	0x4D	77	M	0x6D	109	m
0x10 16 DLE Data link escape 0x30 48 0 0x50 80 P 0x70 112 0x11 17 0cl Device control 1 0x21 49 1 0x51 81 0c 0x71 113 0x12 18 DCZ Device control 2 0x32 50 2 0x52 82 R 0x72 114 0x13 19 DC3 Device control 3 0x33 51 3 0x52 82 R 0x72 114 0x15 21 NAK Negative ack 0x33 51 3 0x54 84 T 0x74 116 0x15 21 NAK Negative ack 0x35 53 5 0x55 85 U 0x74 116 0x15 23 ETB End transmission block 0x37 55 7 0x57 87 W 0x75 117 0x12 25 EM End of medium 0x39 55 7 0x57 87 W 0x77 118 0x12 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM End of medium 0x30 60 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 60 0x50 92 0x78 120 0x10 25 EM End of medium 0x30 92 0x78 120 0x10 25 EM End of medium 0x30 92 0x10 120 0x10 25 EM End of medium 0x30 92 0x10 120 0x10 25 EM End of medium 0x30 92 0x10 120 0x10 25 EM End of medium 0x30 92 0x10 120 0x10 25 EM End of medium 0x30 92 0x10 120 0x10 25 EM End of medium 0x30 92 0x10 120	0x0E	14	so		0x2E	46		0x4E	78	N	0x6E	110	n
0x11 17 DCL Device control 1 0x31 49 1 0x51 81 Q 0x71 13 0x12 18 DC2 Device control 2 0x32 50 2 0x52 82 82 R 0x72 14 0x13 19 DC3 Device control 3 0x33 51 3 0x53 83 S 0x73 115 0x15 21 NAR NR Negative ack 0x35 53 5 0x55 85 0 0x55 85 0 0x55 85 0 0x74 116 0x16 22 SYN Synchronous idle 0x36 54 6 0x56 86 V 0x76 118 0x76 118 0x17 32 EFF End transmission block 0x37 55 7 0x57 85 7 0x57 85 7 0x77 112 0x18 24 CAN Cancel 0x38 56 80 0x88 88 X 0x78 125 0x18 27 0x78 125 0x12 25 EFF End transmission block 0x38 58 1 0x8 88 X 0x78 125 0x18 27 0x78 125 0x12 25 EFF End of Synchronous 0x38 58 1 0x8 88 X 0x78 125 0x18 27 0x78 125 0x12 26 SUB Substitute 0x38 59 1 0x8 88 X 0x78 125 0x18 27 0x78 125 0x12 27 FSC Escape 0x38 59 7 0x58 81 1 0x78 125 0x78 129 0x78 120 0x12 28 FS File separator 0x38 59 7 0x58 94 0x58 94 0x78 125 0x12 30 50 0x1 0	0x0F	15	SI		0x2F	47	/	0x4F	79	0	0x6F	111	0
0x12 18 DC2 Device control 2 0x32 50 2 0x52 82 R 0x72 14 0x13 19 DC3 Device control 3 0x33 51 3 0x53 83 3 0x53 83 83 85 873 318 0x14 20 DC4 Device control 4 0x33 51 3 0x54 84 84 T 0x74 116 0x15 21 NAX Negative ack 0x35 53 5 0x55 85 U 0x75 17 0x17 23 ETB End transmission block 0x37 55 7 0x57 87 W 0x77 119 0x18 24 CAR Cancel 0x39 55 80 W 0x78 177 119 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 120 0x18 25 EM Substitute 0x34 58 1 0x56 98 Y 0x78 123 0x18 26 Excape 0x38 55 9 0x59 91 [0x78 123 0x12 28 FF File separator 0x36 60 0 0x56 93 1 0x78 123 0x12 29 GS Group separator 0x38 62 0 0x56 93 1 0x70 125 0x12 30 5 7 9 0x59 93 1 0x70 125 0x56 93 1 0x70 125	0x10	16	DLE	Data link escape	0x30	48	0	0x50	80	P	0x70	112	p
0x13 19 DC3 Device control 3 0x33 51 3 0x53 83 83 80 0x3 118 0x14 20 DC4 Device control 4 0x34 52 4 0x54 84 84 T 0x74 116 0x15 21 NAK Negative ack 0x35 53 5 50 0x55 85 U 0x75 117 0x16 22 SYN Synchronous idle 0x36 54 6 0x56 86 V 0x76 118 0x17 23 ETB End transmission block 0x37 55 7 0x57 87 W 0x77 119 0x18 24 CAN Cancel 0x38 56 80 0x88 88 X 0x78 120 0x19 25 EM End of medium 0x39 57 9 0x59 89 Y 0x79 120 0x12 25 FSC Escape 0x38 58 : 0x58 90 2 0x72 123 0x12 28 FS File separator 0x30 56 0 0x56 92 \ 0x58 91 [0x70 123 0x12 29 GS Group separator 0x38 61 0x38 61 0x56 93] 0x57 93 0x77 125 0x12 30 57 0x58 87 0x50 93] 0x77 125 0x12 30 57 0x50 93] 0x77 125		17	DC1				1	0x51		Q		113	q
0x14 20 DCA Device control 4 0x34 52 4 0x54 84 7 0x74 16 0x15 21 NAR Negative ack 0x35 53 5 0x55 85 U 50 x75 11 0x16 22 SYN Synchronous idle 0x35 53 5 6 0x55 85 U 50 x75 11 0x17 23 ETB End transmission block 0x37 55 7 0x57 87 W 0x77 11 0x18 24 CAR Cancel 0x38 56 8 0x58 88 X 0x78 12 0x18 25 EM End of medium 0x39 57 9 0x59 89 Y 0x78 12 0x18 26 EXB Substitute 0x33 56 8 0x58 88 X 0x78 12 0x18 27 FSC Escape 0x38 59 7 0x58 90 Y 0x78 12 0x10 28 F File separator 0x36 60 0x50 93 1 0x78 12 0x10 29 GS Group separator 0x36 61 8 0x56 93 1 0x70 125 0x18 30 8 Record separator 0x38 62 0x56 93 0x56 93 0x78 12 0x18 30 8 Record separator 0x38 62 0x56 93 0x56 93 0x6 9x6 0x18 30 8 Record separator 0x38 62 0x56 9x6 0x56 9x6 0x18 30 8 Record separator 0x38 62 0x56 9x6 0x56 9x6 0x18 30 8 Record separator 0x38 62 0x56 9x6 0x56 9x6 0x18 30 8 Record separator 0x38 62 0x56 9x6 0x18 20 8 x6 0x6 0x56 9x6 0x18 20 8 x6 0x6 0x56 9x7 0x18 20 8 x6 0x7 0x56 9x7 0x18 20 8 x6 0x56 9x7 <	0x12	18	DC2	Device control 2	0x32	50	2	0x52	82	R	0x72	114	r
0x15 21 NAK Negative ack 0x35 53 5 0x55 85 U 0x75 117 0x16 22 SYN Synchronous idle 0x36 54 6 0x56 86 V 0x76 118 0x18 22 SYN Synchronous idle 0x38 56 8 0x76 118 0x18 24 CAN Cancel 0x38 56 8 0x88 88 x 0x78 12 0x19 25 EW End of medium 0x39 57 9x59 89 7x0 7x9 7x79 7x79 7x79 7x72 7x72 <td>0x13</td> <td>19</td> <td>DC3</td> <td>Device control 3</td> <td>0x33</td> <td>51</td> <td>3</td> <td>0x53</td> <td>83</td> <td>S</td> <td>0x73</td> <td>115</td> <td>s</td>	0x13	19	DC3	Device control 3	0x33	51	3	0x53	83	S	0x73	115	s
0x16 22 SYM Synchronous idle 0x36 54 6 0x56 86 V 0x76 118 0x17 23 FB End transmission block 0x37 55 7 0x57 87 0x58 88 X 0x79 12 0x19 25 EM End of medium 0x39 57 9 0x58 88 X 0x79 12 0x18 27 FSC Escape 0x38 59 0x58 90 Z 0x78 12 0x10 28 File separator 0x36 60 0x55 9 0x57 0x70 12 0x12 29 GS Group separator 0x30 60 0x55 93 1 0x70 12 0x1E 30 8 Record separator 0x38 61 - 0x55 93 0x59 93 1 0x70 126	0x14	20	DC4	Device control 4	0x34	52	4	0x54	84	T	0x74	116	t
0x17 22 ETB End transmission block 0x37 55 7 0x57 87 W 0x77 119 0x18 24 CAN Cancel 0x38 56 80 x8 80 x 0x78 120 0x19 25 EM End of medium 0x39 57 9 0x58 80 x 0x78 120 0x12 25 SUB Substitute 0x33 58 : 0x54 90 2 0x7A 120 0x16 28 PS File separator 0x38 59 0x58 92 0x7C 124 0x10 28 PS File separator 0x30 61 = 0x5C 92 \tag{0x7b} 124 0x12 29 GS Group separator 0x3E 62 > 0x5E 94 ^ 0x7E 126	0x15	21	NAK	Negative ack	0x35	53	5	0x55	85	U	0x75	117	u
0x18 24 CAN Cancel 0x38 56 8 0x58 88 X 0x78 12 0x19 25 EM End of medium 0x39 57 9 0x59 89 0x78 12 0x1A 26 SUB Substitute 0x3A 58 : 0x5A 90 2 0x7A 12 0x1B 27 FSC Escape 0x3B 59 ; 0x5B 91 (0x7B 123 0x1C 28 FS F1Le separator 0x3C 60 0x5C 92 0x6C 92 0x7D 125 0x1B 30 RS Record separator 0x3B 62 0x5B 93 0x7D 125 0x5C 94 0x5C 92 0x7D 125	0x16	22	SYN	Synchronous idle	0x36	54	6	0x56	86	v	0x76	118	v
0x19 25 EM End of medium 0x39 57 9 0x59 89 Y 0x79 121 0x1A 26 SUB Substitute 0x3A 58 : 0x5A 90 2 0x7A 122 0x1B 27 FSC Escape 0x3B 59 0x5B 91 0x5B 92 0x7C 124 0x1C 28 FS File separator 0x3C 60 0x5C 92 0x7C 124 0x1D 29 GS Group separator 0x3D 61 = 0x5D 93 0x7D 125 0x1E 28 Record separator 0x3E 62 > 0x5E 94 ^ 0x7E 126	0x17	23	ETB	End transmission block	0x37	55	7	0x57	87	W	0x77	119	W
OxlA 26 SUB Substitute 0x3A 58 0x5A 90 2 0x7A 12 0x1B 27 FSC Escape 0x3B 59 ; 0x5B 91 [0x7B 123 0x1C 28 FS File separator 0x3C 60 < 0x5C 92	0x18	24	CAN	Cancel	0x38	56	8	0x58	88	X	0x78	120	×
0x1B 27 FSC Escape 0x3B 59 ; 0x5B 91 [0x7B 123 0x1C 28 FS File separator 0x3C 60 < 0x5C 92 \ 0x7C 124	0x19	25	EM	End of medium	0x39	57	9	0x59	89	Y	0x79	121	У
0x1C 28 FS File separator 0x3C 60 0x5C 92 0x7C 124 0x1D 29 GS Group separator 0x3D 61 = 0x5D 93 0x7D 128 0x1E 30 RS Record separator 0x3E 62 > 0x5E 94 0x7E 126	0x1A	26	SUB	Substitute	0x3A	58		0x5A	90	Z	0x7A	122	z
0x1D 29 GS Group separator 0x3D 61 = 0x5D 93] 0x7D 125 0x1E 30 RS Record separator 0x3E 62 > 0x5E 94 ^ 0x7E 126	0x1B	27	FSC	Escape	0x3B	59	;	0x5B	91	1	0x7B	123	{
0x1E 30 RS Record separator 0x3E 62 > 0x5E 94 ^ 0x7E 126	0x1C	28	FS	File separator	0x3C	60	<	0x5C	92	N.	0x7C	124	- i
UXIE 30 RS RECOIL SEPARATOR UX3E 62 UX3E 94 UX7E 126	0x1D	29	GS	Group separator	0x3D	61	-	0x5D	93	1	0x7D	125	}
0018 021 Umc Hait assesses 042 2 0458 05 0478 127	0x1E	30	RS	Record separator	0x3E	62	>	0x5E	94	^	0x7E	126	-
AXIE OTTION OUT REPUTATOR OX3E 93 1 OX3E 32 OX7E 12/	0x1P	031	jos	Unit separator	0x3F	63	?	0x5F	95	_	0x7F	127	DEL

Figure 8: An ASCII table, used in computer science for encoding text from standard input.

Example: compiling the following C++ code on some machines still *rings the bell*.

```
#include <stdio.h>
using namespace std;
int main()
{
  char y = 7;
  printf("%c\n",y);
  return 0;
}
```

Often used in systems and code involving exchanging strings of text.



UNIT 1.3 SUMMARY - WORKING WITH BINARY

Reading: Digital Fundamentals (DF) Ch. 2 (see Moodle)

- 1. Number representation
- 2. Binary conversions
- 3. Binary arithmetic
- 4. The floating-point system
- 5. Hexadecimals, Binary-Coded Decimals (BCD), Gray codes, and ASCII

Homework: exercises 5-16, 19-32, 35-38, 45, 53, 58 Ch. 2 (DF) (two weeks).