COMPUTER ORGANIZATION AND ARCHITECTURE

Lecture 4

Data Presentation and Binary Conversion

Dr Victoria Mukami

INTRODUCTION

This lecture is an introduction of the number system used by the computer. Within this unit we will review data representation within the CPU, forms of measurement and coding system. Next, we will review how to convert various number systems to the binary number systems.

Learning objectives

By the end of this topic, you should be able to:

1. Understand how data is represented in a computer and the various measures
2. Identify coding systems for representing data
3. Convert Decimal, Octal and Hexadecimal to Binary

OVERVIEW

Computers were invented to make work easier and to advance the field of science in every sectors. When you think of what scientists do, whether in the environment or in health or in nuclear physics, something common with their research and experiments is the analysis they need to perform. Apart from analysis, these scientists need to collect enormous amounts of information. Before computers, these tasks required a huge number of personnel and many man hours. Now in the advent of computers, it is not only possible to work with a few people but even work collaboratively across various disciplines and continents. This chapter looks at how computers can make this possible. Computers have been able to make work easier based on how their internal components process. We review how computers identify data and how it is able to convert both letters, symbols and numbers into a form readable by the computer.

DATA REPRESENTATION

For us to be able to understand how computers can process data, we hear a term called binary. Binary is the format that a computer can understand and is made up of two numbers, 0 and 1. The smallest data that a computer can process is called a bit and is either a 0 or a 1 [1]. A group of 8 bits is known as a byte. The exact number of bytes that make up a kilobyte is 1024. However, to make calculations easier, approximations are used. Based on that, computers can process more than a byte, and these are grouped based on table 1 below.

Table 1: Approximate Bytes

Term	Approximate Bytes
Kilobyte (KB)	1,000
Megabyte (MB)	1,000,000
Gigabyte (GB)	1,000,000,000
Terabyte (TB)	1,000,000,000,000
Petabyte	1,000,000,000,000,000
Exabyte	1,000,000,000,000,000,000

CODING SYSTEMS FOR TEXT

We just saw the representation of data based on binary. During the next lecture, we will learn how number systems can convert to binary systems. Computers have coding systems for text-based data. There are three coding systems used by computers depending on computer type and language.

ASCII

This stands for the American Standard Code for Information Interchange. This is an older still in use system that was invented for PCs. ASCII primarily uses a 7-bit code although some newer codes use an 8-bit code [4]. An 8-bit code can represent 256 characters (text and symbols) while a 7-bit can only represent 128 characters. An example of the 8-bit code is A that is represented as 0100 0001 while + is represented as 0010 1011.

EBCDIC

This stands for Extended Binary-Coded Decimal Interchange Code. This was developed by IBM and was to be used by their mainframes [4]. Like the 8-bit ASCII, EBCDIC represented 256 characters from an 8-bit code. For instance, A is represented by 1100 0001.

UNICODE

While both ASCII and EBCDIC are limited to the normal alphabet, Unicode can handle languages that have symbols as opposed to the alphabet. When you think of a language like Chinese, Greek, Hebrew, Amharic, they all have many

symbols that represent various letters or characters [4]. For example, there are about 50,000 Chinese characters. Unicode allows for 8 to 32-bit codes thereby able to handle languages with more than 256 characters.

NUMBER SYSTEMS

In our everyday life we are acclimatized to the number system that uses numbers ranging from 0-9. This is known as the decimal system. Computers on the other hand use the binary system that has 0 and 1. Other systems use number systems ranging from 0-7, known as the octal system or numbers ranging from 0-9,A-F, known as the hexadecimal system. We will review these systems before we understand how to convert between these systems.

Decimal System

This system relies on numbers ranging from 0-9. Any other number will contain one or more numbers from the same range i.e., 8347 are all numbers from the range. The decimal system is known as base 10 as it contains 10 digits (0,1,2,3,4,5,6,7,8,9) and numbers that are decimal are shown as follows: 589_{10}.

Binary system

This system relies on numbers ranging from 0-1. Any number in the binary system will contain any of the two numbers within the same range i.e., 1101010 are all numbers from the range. The binary system is known as base 2 as it contains 2 digits (0,1) and numbers that are binary are shown as follows: 101_2.

Octal system

This system relies on numbers ranging from 0-7. Any number in the octal system will contain any of the seven numbers within the same range i.e., 5372 are all numbers from the range. The octal system is known as base 8 as it contains 8 digits (0,1,2,3,4,5,6,7) and numbers that are octal are shown as follows: 267_8.

Hexadecimal system

This system relies on numbers ranging from 0-15. However, since it is not possible to tell the difference from a number with 151412 whether it's 15 or just part of the number. This introduces letters for all the two digit numbers (10-15). These are replaced with letters A-F) The table below shows the numbers and their corresponding letters.

Table 2: Hexadecimal letters

Number	Corresponding letter
10	A
11	B
12	C
13	D
14	E
15	F

Any number will contain one or more numbers from the same range i.e., if I use the previous number - 151412 it may look something of the sort F14C. The hexadecimal system is known as base 16 as it contains 16 digits (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F) and numbers that are hexadecimal are shown as follows: $8F9_{16}$.

DECIMAL TO BINARY CONVERSION

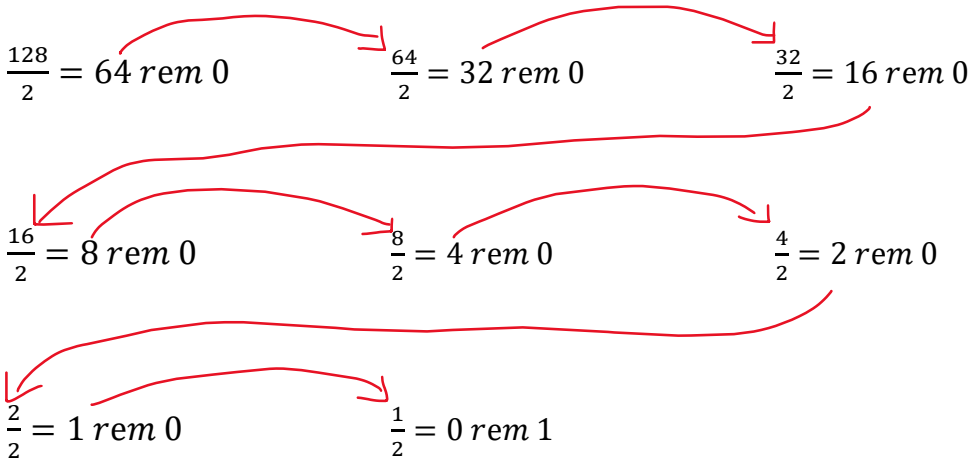
First, we will focus on converting from a binary number to a decimal number. There are two methods to achieve that, and we will look at both methods. The first method is a division by 2 until the number to be divided is a 0. Then the remainder of the numbers are considered as the binary number. The remainders are read from bottom to top. For instance, let us try and convert 5_{10} to binary.

$$\frac{5}{2} = 2\ rem\ 1$$

$$\frac{2}{2} = 1\ rem\ 0$$

$$\frac{1}{2} = 0\ rem\ 1$$

Once the number being divided remains as a zero then the conversion is complete. We read the remainders from the bottom to top. Final binary answer is 101_2. Let us work out one more using this method. Convert 128_{10} to binary.



The numbers when read from bottom to top would be 10000000_2.

Now let us look at the second method. The second method focuses on the multiples of 2 including the number 1. Imagine the number line used then we incorporate the multiples of 2. The numbers would therefore be 2^0 2^1 2^2 2^3 2^4 2^5 2^6 ….. This then gives us the numbers 1,2,4,8,16,32,64….. These numbers can grow to whatever number is needed. Now based on this we create a table which we use to convert our decimal numbers to binary. The least significant number 1 goes to the right while the most significant number is on the left.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Assuming you want to convert a number, you would pick the number to be converted, lets call it X, then follow the following rules:

1. Check whether there is a number from the number line equal or less than it (Closest)
2. Subtract from X the number from 1) above
3. Place a 1 under the subtracted number on the table
4. Repeat from 1 until finally you have a 0 as the subtraction answer.
5. Then place 0 under any number on the table that was not used.

Let's do this practically below. We will convert 5_{10} first. Let us follow the rules:

- Is there a number equal or less to 5 in the number line? Yes 4.
- 5 – 4 = 1
- Place a 1 in the table

128	64	32	16	8	4	2	1
					1		

- Similar number is 1
- 1 – 1 = 0

128	64	32	16	8	4	2	1
					1		1

- Place 0 on any numbers not used

128	64	32	16	8	4	2	1
					1	0	1

We read the number from left to right. Final number is 101_2.

- Lets now convert 128_{10}. Is there a number equal or less? Yes, 128
- 128 – 128 = 0

128	64	32	16	8	4	2	1
1							

- Fill in the 0's within

128	64	32	16	8	4	2	1
1	0	0	0	0	0	0	0

Final answer is 10000000_2

Fraction conversions

When converting from decimal to binary when using fractions, we will review one method. We first convert the fraction to a decimal point number. For instance, ½ would be 0.5. 8/32 would be 0.25 and so on. This method is like the division one just that we go ahead and multiply. We basically multiply the numbers until the final one has 0 for the answer. Sometimes we do not get a 0 after 5 tries and we stop the process there. Our binary number is made up of the whole parts of the product. Let us try two examples to understand what I am talking about.

Let us convert the following number to binary: 0.0625_{10}.

$$0.0625 \times 2 = 0.125 \qquad 0.125 \times 2 = 0.25$$

$$0.25 \times 2 = 0.5 \qquad 0.5 \times 2 = 1.0$$

If you look at the workout above, when we multiply for the first time, our binary number starts with 0.0 and we multiply the .25 on the next step. This goes on until we get a .0 which means our workout is done. Our final binary answer is: 0.0001_2.

Let us now convert another number: 0.584_{10} into binary

$$0.584 \times 2 = 1.168 \qquad 0.168 \times 2 = 0.336$$

$$0.336 \times 2 = 0.672 \qquad 0.672 \times 2 = 1.344$$

$$0.344 \times 2 = 0.688 \qquad 0.688 \times 2 = 1.376$$

As you can see with the number above it may continue to infinity. In this case we pick up to the fifth number and you can place three dots to show that it continues on. Our final binary number is 0.10010_2…

Now what happens when we have a number before the decimal point? Well we convert the whole part before the fraction. Then we work on the fraction part and combine. For instance, to convert 5.0625_{10} to binary we first work on the whole part. We know 5 in binary is 101_2. Then we work on the 0.0625 which in binary is 0.0001_2. Our final answer would be 101.0001_2.

BINARY TO DECIMAL CONVERSION

Here we look at one method of converting from binary to decimal.

The method involves using the number line as seen in method two above. The rules state:

1. Have the table drawn
2. Place the binary number from the least significant to the most significant, right to left onto the table
3. Pick all the numbers that have a 1 underneath
4. Add all the numbers together. This gives you your final answer.

Let us convert 101011_2 to Decimal.

- Step 1 is to place the numbers on the table from Least Significant to Most.

128	64	32	16	8	4	2	1
		1	0	1	0	1	1

- Pick the numbers with a 1 underneath: 1,2,8,32
- Add the number: $1 + 2 + 8 + 32 = 43_{10}$
- Final answer = 43_{10}

When converting from fractions we use a similar number line but instead of 2^1 you have 2^{-1}. I will demonstrate on the table below. The second row in green shows the converted numbers.

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}
0.5	0.25	0.125	0.0625	0.03125

To convert 101.11101_2 to decimal, we convert the whole number 101 which by now we know is 5_{10}. Then we arrange on the number line the fraction part and add the numbers with a 1 underneath.

0.5	0.25	0.125	0.0625	0.03125
1	1	1	0	1

We add the numbers: $0.5 + 0.25 + 0.125 + 0.03125 = 0.90625_{10}$.

We attach the two and our final answer would be 5.90625_{10}.

OCTAL TO BINARY CONVERSION

Remember octal numbers are Base 8 numbers. Octal is made up of 8 digits. Without converting from Octal to decimal then to binary, there is an easy way in which to convert. First, we need to convert all the numbers to 3 bits. Why you ask? First there are 2^3 combinations of numbers. Hence the three bits. Let's make a table that converts the numbers.

Table 3: Octal to binary conversion

Number	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Once all the numbers are converted then it is easy to use the table to convert them. Let us convert 721_8 to binary.

Using Table 3, we pick each corresponding binary number.

7 – 111

2 – 010

1 – 001

Our final answer would be combining all the binary numbers therefore having:

$111\ 010\ 001_2$.

To convert from binary to octal you would divide the number to three bits from the least significant to the most significant then use the table to convert. Let us convert the following number: 1101010100_2 to octal.

We first pick the number and subdivide into three bits and replace each with the table equivalent.

1	1 0 1	0 1 0	1 0 0
1	5	2	4

The final answer would be 1524_8.

HEXADECIMAL TO BINARY CONVERSION

Remember hexadecimal numbers are Base 16 numbers with letters. Hexadecimal is made up of 16 digits and letters. Without converting from hexadecimal to decimal then to binary, there is an easy way in which to convert. First, we need to convert all the numbers to 4 bits. Why you ask? First there are 2^4 combinations of numbers. Hence the four bits. Let's make a table that converts the numbers.

Table 4: Hexadecimal to binary conversion

Number	Binary
0	0000
1	0001
2	0010

3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Once all the numbers are converted then it is easy to use the table to convert them. Let us convert $64FE_{16}$ to binary. Using table 4, we pick each corresponding binary number.

6 – 0110

F – 1111

4 – 0100

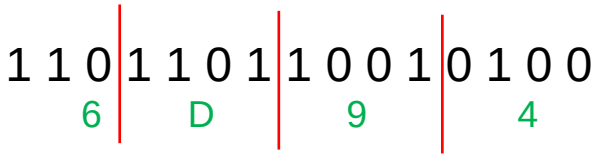
E – 1110

Our final answer would be combining all the binary numbers therefore having:

$0110\ 1111\ 0100\ 1110_2$.

To convert from binary to hexadecimal you would divide the number to four bits from the least significant to the most significant then use the table to convert. Let us convert the following number: 110110110010100_2 to octal.

We first pick the number and subdivide into three bits and replace each with the table equivalent.



The final answer would be $6D94_{16}$.

SUMMARY

We have reviewed and worked out quire a bit of areas in this lecture. We looked at the areas of data representation and data coding. We then reviewed the number systems, converted from decimal to binary and back to decimal. Further, we converted from octal and hexadecimal to binary and back to octal and hexadecimal. The next week we learn how to do binary arithmetic (Addition, Subtraction, Multiplication and Division).

DISCUSSION TOPIC

We have learnt how to calculate and convert from various numbering systems to binary and back. One of the ways we have not looked at is how to convert directly from Octal to Hexadecimal without first converting to binary. Is this possible? Why might it be necessary for such conversions and how do you do the conversion?

REFERENCES

[1] G. Shelly and M. Vermaat, Discovering Computers — Fundamentals: Your Interactive Guide to the Digital World. Boston, MA: Course Technology, 2012.

[2] W. Stallings, Computer Organization and Architecture Designing for Performance. Hoboken, NJ: Pearson Education, Inc, 2016.

[3] A. Evans, K. Martin and A. Poatsy, Technology in Action. New York, NY: Pearson, 2020

[4] D. Morley and C. Parker, Understanding Computers: Today and Tomorrow. Boston, MA: Course Technology, 2017.