

## **CS 52 SMC Lab 3**

### **Due Date: Please see due date on Canvas**

#### **Assignment Rules:**

- 1. If you are looking at this assignment on ANY device other than a computer, please immediately close this document. Go to a computer and print a copy. The visibility of whole document is NOT guaranteed unless it is printed from a computer. You are fully responsible for any loss to you, if procedure given is not followed.**
- 2. If your submission is so late that Canvas does not accept it, then it is TOO LATE for a credit. In that case please DO NOT Email me your program for any credit. If you want feedback on your late work, then that can only be done during office hours. No feedbacks on late assignments are done through emails.**
- 3. Program submitted with Compile errors will automatically get zero. Please do not submit programs with compile errors. They are not feedback worthy. On the other hand, your compile errors can be fixed in may be no more than 1 minute, if you show me your program during office hours.**
- 4. Your assignment must pass ALL tests given in the test run portion of this document. You will lose 10 points for each failing test. Four-failed test erases the whole credit for the assignment. There will be points taken off for other bugs (for example bad output formatting). But most key bugs will be removed if the program passes test runs.**

#### **Statement of Work**

In this work you will write a menu driven C++ program that uses arrays and does conversions from Octal to decimal numbers and binary to decimal numbers. Appendix two introduces non-decimal number systems. You can read it to know key details of non-decimal number systems. Table below gives values of numbers up to 15, as reflected in various numbering systems.

Decimal	Binary	Octal	Hexadecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Your program must have at least two significant user defined functions, each using C++ arrays. **Programs done using only main function, and done without using arrays, will get no credit.** The program you will submit will have a menu driven system as below:

**Caution! No validation is done on numbers entered. They must conform to required input form.**

**\*\*\*\*\*Main Menu\*\*\*\*\***

**1. Convert binary to decimal:**

**2. Convert Octal to decimal:**

**3. Exit:**

**1**

**Enter the binary number**

**:10101**

**Original binary Number: 10101**

**Its decimal conversion:21**

**Caution! No validation is done on numbers entered. They must conform to required input form.**

**\*\*\*\*\*Main Menu\*\*\*\*\***

- 1. Convert binary to decimal:**
- 2. Convert Octal to decimal:**
- 3. Exit:**

**Enter the octal number**

**:177**

**Original octal Number: 177**

**Its decimal conversion:127**

**Caution! No validation is done on numbers entered. They must conform to required input form.**

**\*\*\*\*\*Main Menu\*\*\*\*\***

- 1. Convert binary to decimal:**
- 2. Convert Octal to decimal:**
- 3. Exit:**

**3**

Other test runs are shown below. The test runs only show conversion values. Your actual output from your program may differ in look and feel. Thus, use below outputs only for testing your program and ascertain that conversion math is done correctly. Slight variations in output format are acceptable.

Test Runs from My program for Octal to Decimal conversion

Enter the octal number

:777

Original octal Number: 777

Its decimal conversion:511

.....

Enter the octal number

:176

Original octal Number: 176

Its decimal conversion:126

.....

Enter the octal number

:765

Original octal Number: 765

Its decimal conversion:501

.....  
Enter the octal number  
:7777  
Original octal Number: 7777

Its decimal conversion:4095

.....  
Enter the octal number  
:76765  
Original octal Number: 76765

Its decimal conversion:32245

Test Runs from My program for binary to Decimal conversion

Enter the binary number  
:1111  
Original binary Number: 1111

Its decimal conversion:15

.....  
Enter the binary number  
:101010  
Original binary Number: 101010

Its decimal conversion:42

.....  
Enter the binary number  
:111111  
Original binary Number: 111111

Its decimal conversion:63

.....  
Enter the binary number  
:1011110  
Original binary Number: 1011110

Its decimal conversion:94

.....  
Enter the binary number  
:11111111111111  
Original binary Number: 1111111111111111

Its decimal conversion:524287

## Appendix B: Non-decimal Number systems

### Non-Decimal Number Systems

The first computer ENIAC used decimal numbering system to crunch numbers. Engineers soon realized that design of computers based on decimal numbering system is too complicated. With discovery of transistors it was easy to build circuit switches and logic gates which in on state would represent digit one and in off state digit zero. Thus, it is easy to build a numbering system just using digits zero and one, and have computer hardware function on that number logic. Practically all computers in the world today use binary numbering system that we discuss next.

#### Binary Numbers

A pair of discrete values can be said to form a binary set<sup>1</sup>. The word binary is derived from Latin word binarius, which literally means two by two. When we feed data into computers they are fed as digital signals<sup>2</sup> and signal can only have two values, which form a pair. Imagine we ask computer to add numbers three and five. First we need to feed numbers three and five into computer as digital signals, and then the computer would add them up and create a digital signal for sum of three and five. Finally computer would display the sum of three and five in the form in which we can understand it. Since computers can only process zero or one, it would not be able to understand the decimal numbering system<sup>3</sup> that we are used to. Rather it understands a numbering system that is based on numbers one and zero only. This number system is called a binary system, as it has only two digits – zero and one. The system we human use is called a decimal system as we have ten digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. In order to understand how the binary numbering system works, we can start with by understanding as how our decimal numbering system works. Imagine we have a number say 1534 or one thousand five hundred and thirty four. We re-write the number 1534 along with the position index of each digit from right to left (Table 1.2). (In computer science indexing always starts with zero and proceeds towards higher numbers).

Index of the location of each digits from extreme right →	3	2	1	0
The number ----- →	1	5	3	4

Table 1.2

We can then re-write number 1534 as follows (Table 1.3):

Contribution from location with index 0	= Digit at location zero*10 <sup>0</sup>	= 4*10 <sup>0</sup> = 4*1	= 4
Contribution from location with index 1	= Digit at location one*10 <sup>1</sup>	= 3*10 <sup>1</sup> = 3*10	= 30
Contribution from location with index 2	= Digit at location two*10 <sup>2</sup>	= 5*10 <sup>2</sup> = 5*100	=500
Contribution from location with index 3	= Digit at location three*10 <sup>3</sup>	= 1*10 <sup>3</sup> = 1*1000	=1000

<sup>1</sup> A set is a collection of numbers or things, in which no two members can be same.

<sup>2</sup> It is also possible to build computers that can work based on analogue signal. The technology of building digital computers however is more advanced and is commonplace.

<sup>3</sup> ENIAC however was designed on decimal numbering system.

Total of last column	1534
----------------------	------

**Table 1.3**

We are so used to decimal system that we do not actually break down to see that how number 1534 is made of its parts from thousands, hundreds, tens, and single digits (as in last column of Table 1.3). This is not so with the binary system, as we are not used to it. In binary system each digit in a number can only be zero or one. Then how much each digit contributes depends on its location. In the manner similar to decimal system described above, the extreme right digit (at index zero) gets multiplied by  $2^0$ , the digit at index one with  $2^1$ , the digit with index two with  $2^2$  and so on. We first take a binary number 101011 and convert it to its decimal equivalent to show the whole process (Table 1.4). *Understand that binary number 101011 is read as one-zero-one-zero-one-one and not as one hundred one thousand and eleven!*

Index of the location of each digits from extreme right →	5	4	3	2	1	0
The number →	1	0	1	0	1	1
Contribution of the particular digit to overall number →	$1*2^5$ ↓	$0*2^4$ ↓	$1*2^3$ ↓	$0*2^2$ ↓	$1*2^1$ ↓	$1*2^0$ ↓
	32	0	8	0	2	1
Total of contributions from all locations →	32 + 0 + 8 + 0 + 2 + 1 = 43					

**Table 1.4**

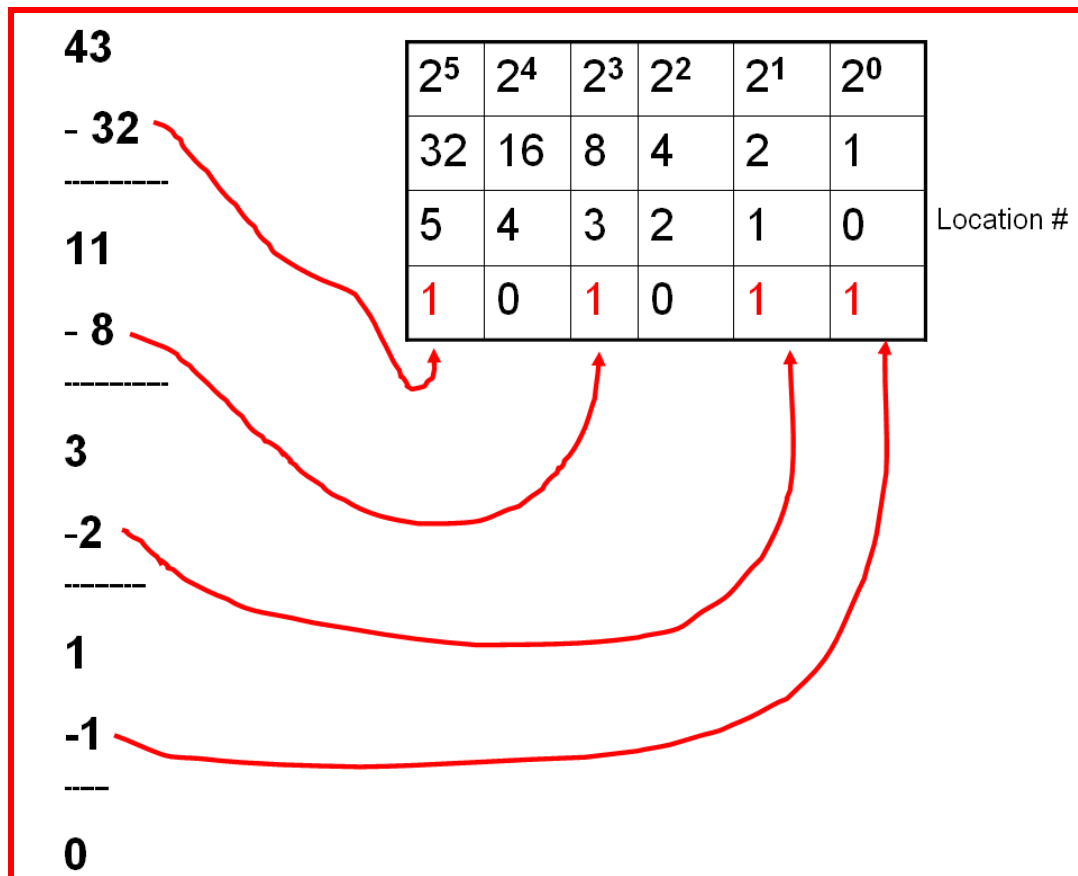
The process shown in Table 1.3 indicates that number 101011 in binary system is same as number 43 in decimal system! Often people would represent a binary number such as 101011 as 101011base<sub>2</sub> to indicate that the number represented is a binary number.

The process of converting from decimal to binary number is a bit different. Here the task is to determine digits (either zero or one) at various locations, latter being identified by their index. We will show the process that is reverse of what we showed in Table 1.4 and convert number decimal number 43 to binary form. Table 1.5 shows the layout of index for each digit and the multiplication factor for that location in the binary system.

Index of the location of each digits from extreme right	6	5	4	3	2	1	0
Multiplication factor to the digit at each location	$2^6$ ↓	$2^5$ ↓	$2^4$ ↓	$2^3$ ↓	$2^2$ ↓	$2^1$ ↓	$2^0$ ↓
Value of the multiplication factor	64	32	16	8	4	2	1

**Table 1.5**

The overall process is shown in Figure 1.10.



**FIG. 1.10**

We find the largest number that can be subtracted from 43 such that the number is some power of two. Top two rows in the table in Figure 1.10 show the numbers that are powers of two from zero to five. Notice that  $2^6$  which is 64 cannot be subtracted from 43 as that will cause a negative remainder. Thus the largest number that can be gotten by an integer power of two and can be subtracted from 43 is 32. Since  $32 = 2^5$ , we put a 1 in the location 5 in the bottom most row of table in the Figure 1.10. The remainder of  $(43-32)$  is 11. The next largest number that can be subtracted from 11 from second row of table is 8. Since  $8 = 2^3$ , we put a 1 in the location 3. Remainder of  $(11-8) = 3$ . The largest power of two number that can be subtracted from 3 is 2. Since  $2 = 2^1$ , we put one in location #1. Remainder of  $(3-2) = 1$ . The largest power of two that can be subtracted from 1 is  $2^0$  which is one. Therefore we also put 1 in the location # 0. Remainder of  $(1-1)$  is zero. We then put zero in empty locations. Thus decimal 43 is 101011 in binary system. Figure 1.11 shows the conversion of decimal numbers zero to 15 into binary system.

Decimal Equivalent	Binary System			
	$2^3$	$2^2$	$2^1$	$2^0$
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

**FIG. 1.11**

### Another Technique to Convert Decimal Numbers to Binary Form

If system of remembering powers of two seems too complicated, there is another way to convert decimal numbers to binary form. The process<sup>4</sup> can be described as follows:

1. Divide the number by two.
2. Place the remainder at the location zero in the binary digit.
3. Take the quotient from step one and repeat the step one.
4. Place the remainder to the left of previous location in the binary digit.
5. Continue until remainder is zero or one. The last remainder becomes the first (leftmost) digit in the binary number.

Let us apply this technique to convert 43 to binary form. The sequence of conversion is shown in Figure 1.12

---

<sup>4</sup> Such step by step processes are called algorithms in Computer Science.



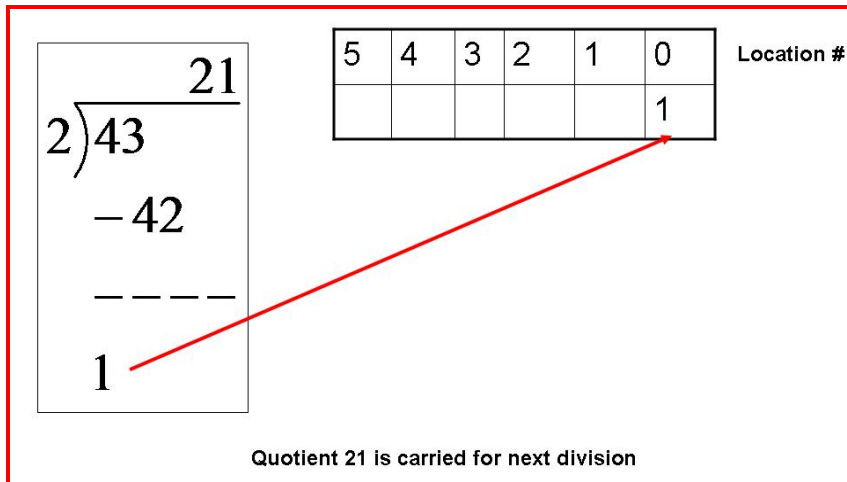


FIG. 1.12

Number 43 divided by two results into a remainder of one and quotient of 21. Thus one becomes the right most digit in the binary number. Quotient 21 is carried for next division by two (Figure 1.13A).

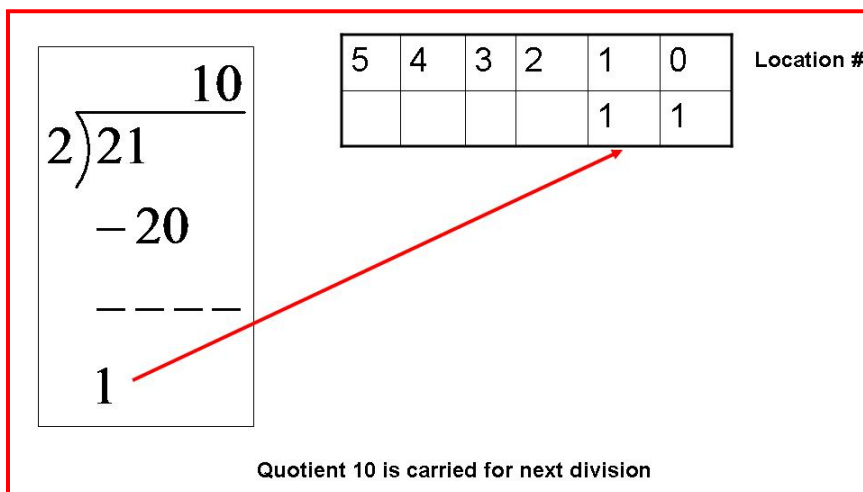
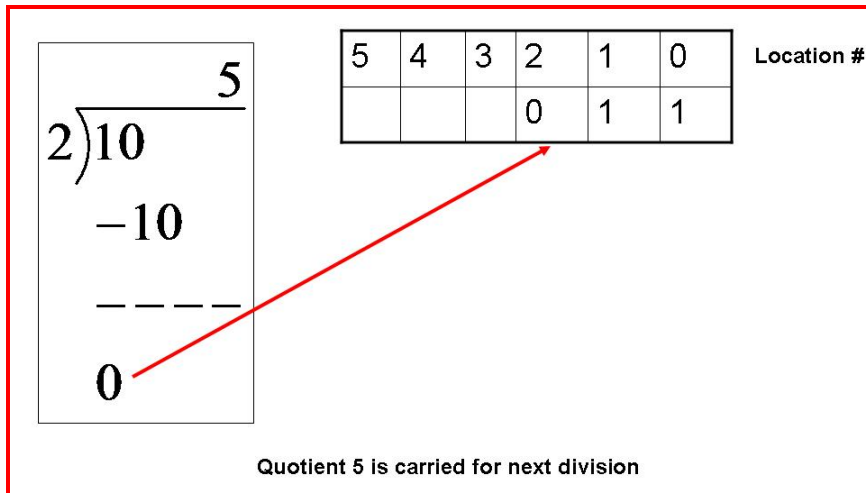


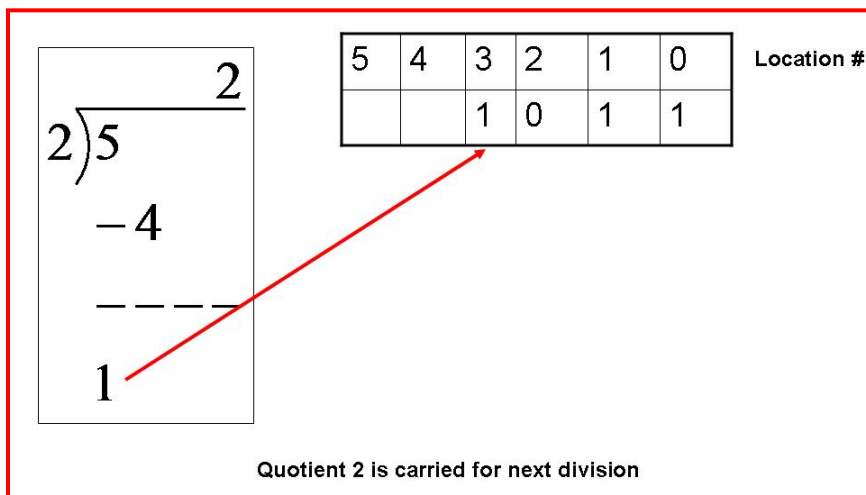
FIG. 1.13 A

Number 21 divided by two results into a remainder of one and quotient of ten. Thus one becomes the second right most digit in the binary number. Quotient ten is carried for next division by two (Figure 1.13B).



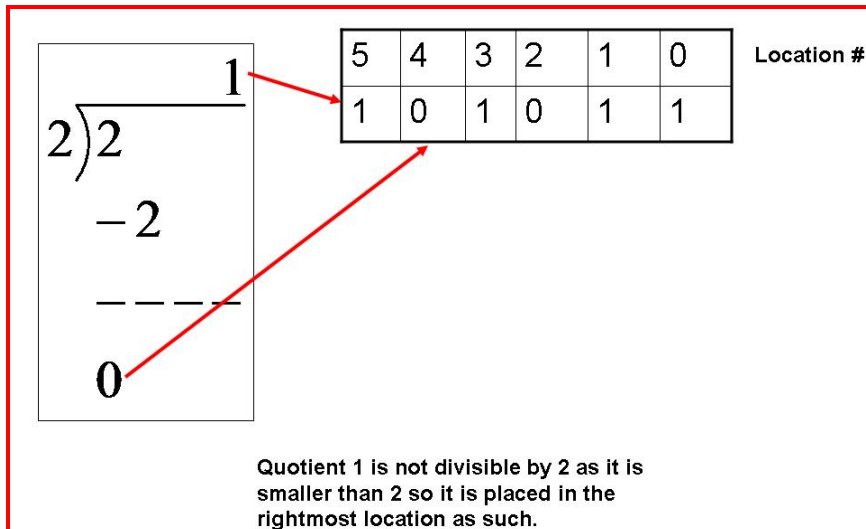
**FIG. 1.13 B**

Number 10 divided by two results into a remainder of zero and quotient of five. Thus zero becomes the third right most digit in the binary number. Quotient five is carried for next division by two (Figure 1.13C).



**FIG. 1.13 C**

Number five divided by two results into a remainder of one and quotient of two. Thus one becomes the fourth right most digits in the binary number. Quotient two is carried for next division by two (Figure 1.13D).



**FIG. 1.13 D**

Number two divided by two results into a remainder of zero and quotient of one. Thus zero becomes the fifth right most digit in the binary number. Since quotient one is less than two and is not divisible by two further, it becomes the leftmost digit in binary number. You can see that Figures 1.10 and 1.13D have identical binary number!

## **Hexadecimal Numbers**

Binary numbers are excellent in representing small numbers. Problems come when one has to store large numbers. For example we all know that in today's computers, it is not uncommon to have 2 gigabyte of RAM or 500 gigabyte of hard disk space. Now the way data or instruction are stored in RAM or in hard disk is that the storage areas are broken down in sectors of bytes. Each sector has a physical address. This address is similar in nature to the street address that you have for your residence. There is one difference however. The street addresses will never run into billion! Imagine that you have 1 gigabyte of RAM in your computer. That is 1 billion bytes. If a sector of 4 bytes must have an address, then total number of sectors would be 0.25 billion or 250,000,000 sectors. So this would be as if house numbers will run from 1 to 250,000,000. So what is 250,000,000 in binary? It is 101111101011100001000000base 2. The nine digits that we had in decimal system now need 25 digits in binary system. The strings representing large numbers in binary system thus get too large. Therefore, engineers designed another numbering system, called hexadecimal, which has 16 digits. Well how can a numbering system have 16 digits, when actual digits are only 10 (digits 0 to 9). What happens is that alphabets A, B, C, D, E, and F are used as six additional digits. Table 1.6 shows the 16 digits in hexadecimal and their equivalent values in decimal and binary systems.

Hexadecimal Digits	Representation in Decimal System	Representation in Binary System
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111
<b>Table 1.6</b>		

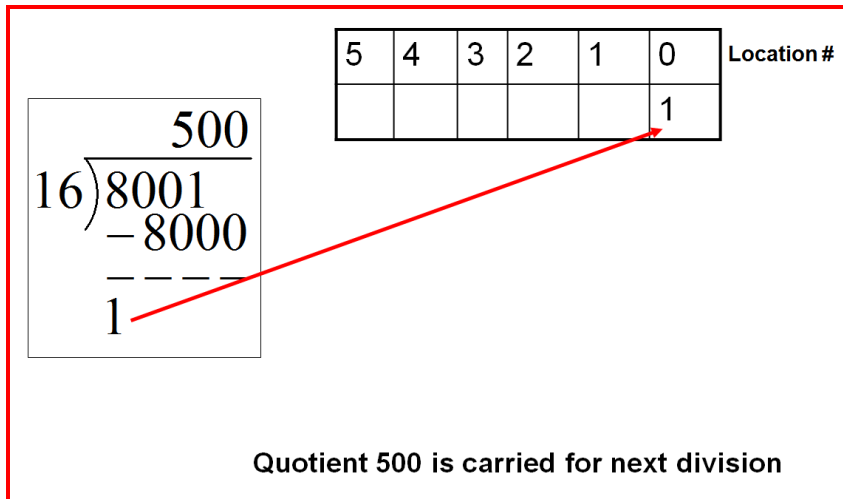
Without much ado, now we will tell you what 250,000,000 is in system based on hexadecimal. It is EE6B280. It is incredible that how number of characters in the representation shrunk to mere seven instead of nine for decimal system and 25 for binary. Thus hexadecimal numbers are very efficient in condensing the representation of large numbers. The procedure for converting hexadecimal to decimal and vice versa is identical to the similar procedures shown for binary and decimal inter-conversions. However, this time we would like to choose somewhat larger number. Imagine that you bought a new 500 megabyte flash drive. If each sector is 4 byte then number of sectors would be 125 megabyte. Let us say that you put some files already in first 2000 sectors. That means that you have used up 8000 bytes or 8 kilobytes. Now next file added would get written at location 8001th byte. So what would this number be in hexadecimal? Thus problem reduces to converting number 8001 to hexadecimal base. We follow the procedure shown in Figures 1.12 and 1.13, except this time we must divide the decimal number by 16, because hexadecimal system has 16 digits. We would also need the table 1.6 as this table would tell us to convert remainders into proper hex<sup>5</sup> digits. The algorithm of the process is as follows:

1. Divide the number by 16.
2. Place the remainder at the location zero in the hex digit. If remainder is more than nine then use table 1.6 to find the alphabet corresponding to that number.
3. Take the quotient from step one and repeat the step one.
4. Place the remainder to the left of previous location in the hex digit.
5. Continue until remainder is zero or one. The last remainder becomes the first (leftmost) digit in the hex number.

Figure 1.14 shows conversion of decimal 8001 to hex, step-by-step.

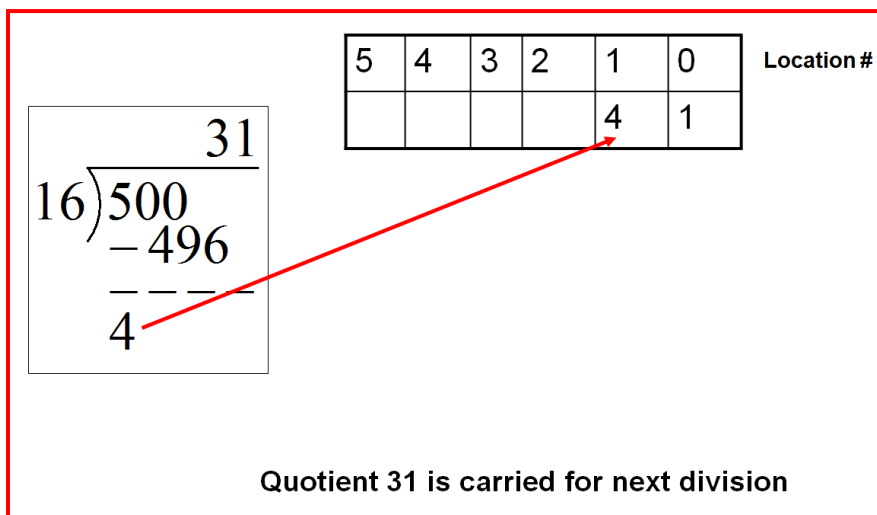
---

<sup>5</sup> Hexadecimal numbers are also affectionately called hex.



**FIG. 1.14A**

Number 8001 divided by 16 results in a quotient of 500 and remainder of one. Thus we place 1 in the location number zero (Figure 1.14A), and carry the quotient 500 for next division by 16.



**FIG. 1.14B**

Number 500 divided by 16 results in quotient of 31 and remainder of 4. Number 4 gets places in location #1(Figure 1.14B) and quotient 31 gets carried for next division by 16.

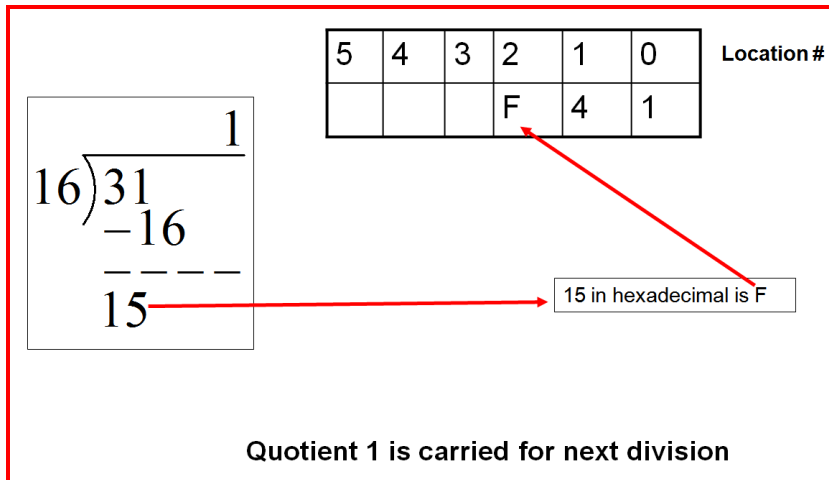


FIG. 1.14C

Number 31 divided by 16 yields a quotient of one and remainder of 15. On base hexadecimal, 15 is F. Thus F gets placed in location #2 (Figure 1.14C), and quotient 1 gets carried over for next division by 16.

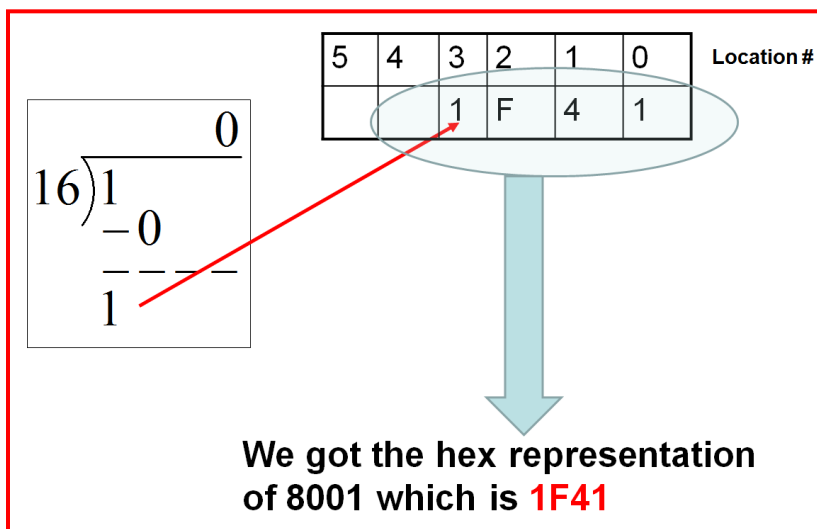






FIG. 1.14D

The numerator of 1 is smaller than 16, so in 1/16, the quotient is zero and remainder is one. Thus, 1 gets placed at the location # 3. This gives us the conversion of decimal 8001 as 1F41<sub>base 16</sub>.

### Converting Hexadecimal to Decimal

The algorithm for converting hexadecimal to decimal is identical to the one we used for converting binary to decimal earlier. To keep matters simple we convert hex 1F41 back to decimal to ascertain that we get 8001 back. Table 1.7 shows the conversion.

Index of the location of each digits from extreme right →	3	2	1	0
The hexadecimal number →	1	F	4	1
Contribution of the particular digit to overall number →	1*16 <sup>3</sup>	15*16 <sup>2</sup>	4*16 <sup>1</sup>	1*16 <sup>0</sup>
	 4096	 3840	 64	 1
Total of contributions from all locations →	4096 + 3840 + 64 + 1 = 8001			
Table 1.7				

In hex system, the digit at each location will get multiplied by the  $16x$ , where  $x$  is the location number starting with location zero for extreme right digit. Therefore, for 1F41base 16, rightmost 1 contributes 1, 4 contributes 64, F contributes 3840, and extreme left 1 contributes 4096. Sum of all these would be 8001, the decimal form of hex 1F41.

### **Converting Hex to Binary and Vice Versa Using Short Procedure**

If I asked you convert a hex number to binary right now, you may be tempted to convert hex to decimal first and then convert decimal to binary. That is the long procedure. There is a shorter procedure to convert hex to binary directly. Since hexadecimal has 16 digits and number 16 is simply 24, when converting from hex to binary we just convert each hex digit to binary and if there are less than 4 binary digits after conversion, then we place leading zeros, until there are 4 digits in it. Table 1.8 shows the procedure.

The hexadecimal number →	1	F	4	1
Binary value for each digit →	0001	1111	0100	0001
The binary form of 1F41 ----->	0001111101000001			
The binary form of 1F41 after removing leading zeros ----->	1111101000001			
<u>Table 1.8</u>				

To convert a hex number to binary we start in set of four digits from extreme right and convert each set to hexadecimal independent of other. For example, the number 1111101000001 divided in set of fours starting from extreme right would be as follows:

1	1111	0100	0001	Binary Number
1	15	4	1	
1	F	4	1	Corresponding hexadecimal number
1F41				

The conversion to hex of each set of four is also shown. This conversion from binary back to hex is

The conversion to hex of each set of four is also shown. Thus conversion from binary back to hex is just the exact reverse of converting hex to binary.

### **Octal Numbers**

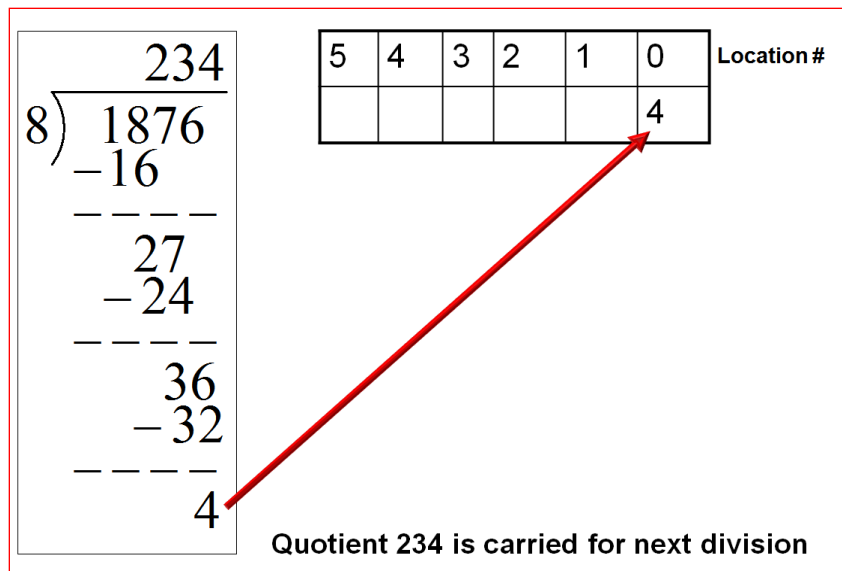
Former Computer Company Digital Equipment Corporation or DEC used a number system to the base eight, called octal. The octal system has digits zero to seven or overall eight digits in it. This numbering system worked out just about right for the minicomputer systems they pioneered. Octal

system has also been used in UNIX file permissions in a command called chmod, which literally means change modes. We re-write the Table 1.6 to add octal numbers in it (Table 1.9).

Octal Digits and Numbers	Representation in Hexadecimal digits	Representation in Decimal System Digits and Numbers	Representation in Binary System Digits and Numbers
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
10	8	8	1000
11	9	9	1001
12	A	10	1010
13	B	11	1011
14	C	12	1100
15	D	13	1101
16	E	14	1110
17	F	15	1111
Digits in each system		Table 1.9	

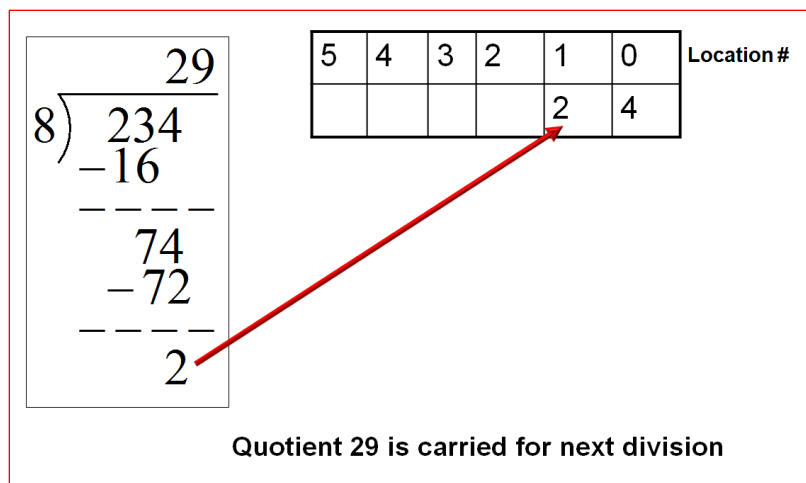
Representing a large numbers in octal system will take more digits than hexadecimal but less compared to binary system. Year 1876 was 100 years after the declaration of independence was written by Thomas Jefferson. Let us see what 1876 would be in octal system. Since octal system has 8 digits, in converting from decimal to octal we divide by 8 until the quotient is zero. The procedure is identical to the one shown for hexadecimal (Figure 1.14), and for binary (Figure 1.13). We show the procedure in Figure 1.15.





**FIG. 1.15A**

Number 1876 divided by 8 yields a quotient of 234 and remainder of 4. The remainder 4 is placed in the extreme right location at the location number zero (Figure 1.15A). The quotient 234 is carried for next division.



**FIG. 1.15B**

Number 234 divided by 8 yields a quotient of 29 and remainder of 2. The remainder 2 gets placed location number 1 (Figure 1.15B) as part of emerging octal number. The quotient 29 is carried for next division by 8.

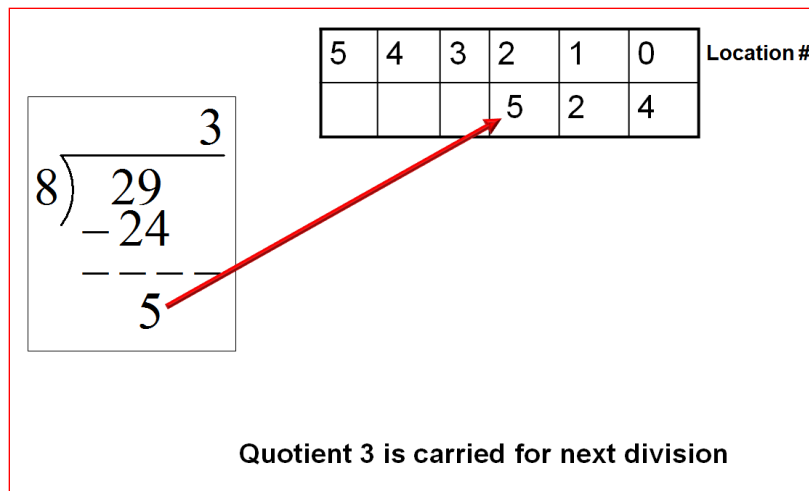


FIG. 1.15C

The number 29 divided by 8 results in a quotient of 3 and remainder of 5. The remainder 5 gets placed in the location number 2 for the emerging octal number (Figure 1.15C). The quotient 3 is carried over for next division by 8.

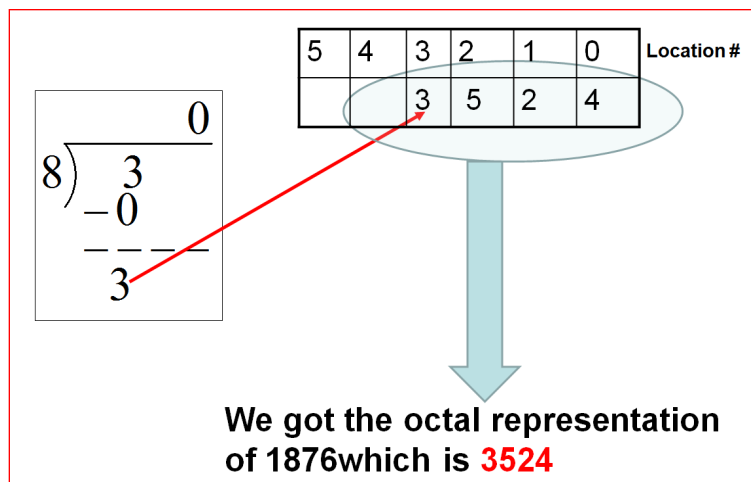






FIG. 1.15D

Finally number 3 divided by 8 yields a quotient of zero and remainder of 3. Thus 3 is placed leftmost (in location #3) giving us  $3524_{\text{base octal}}$  as the number that is equivalent to decimal 1876.

### Conversion from Octal to Decimal

In the interest of brevity, we just take number  $3524_{\text{base octal}}$  and convert it back to base decimal. Table 1.10 shows the conversion steps. Since octal system has 8 digits, the digit at each location will get multiplied by the  $8^x$ , where  $x$  is the location number starting with location zero for extreme right digit.

Index of the location of each digits from extreme right →	3	2	1	0
The octal number →	3	5	2	4
Contribution of the particular digit to overall number →	$3 \cdot 8^3$	$5 \cdot 8^2$	$2 \cdot 8^1$	$4 \cdot 8^0$
	 1536	 320	 16	 4
Total of contributions from all locations →	1536 + 320 + 16 + 4 = 1876			
Table 1.10				

Therefore, for  $3524_{\text{base } 8}$ , rightmost 4 contributes 4, 2 contributes 16, 5 contributes 320, and extreme left 3 contributes 1536. Sum of all these would be 1876, the decimal form of octal 3524.

### **Converting Octal to Binary and Vice Versa Using Short Procedure**

There is also a shorter procedure to convert octal to binary directly and vice versa. Since octal has 8 digits and number 8 is simply 2<sup>3</sup>; when converting from octal to binary we just convert each octal digit to binary and if there are less than 3 binary digits after conversion, then we place leading zeros, until there are 3 digits in it. Table 1.8 shows the procedure.

The octal number →	3	5	2	4
Binary value for each digit →	011	101	010	100
The binary form of octal 3524 ----->	011101010100			
The binary form of octal 3524 after removing leading zeros ----->	11101010100			
Table 1.8				

To convert a binary number to octal we start in set of three digits from extreme right and convert each set to octal, independent of others. For example, the number 111010100 divided in set of three starting from extreme right would be as follows:

11	101	010	100	Binary Number
3	5	2	4	
3524				Corresponding octal number

Second row shows the octal value for each set of three (or less for extreme left group). The binary 111010100 would result into octal 3524.