Coder Army

Lecture 001

Typed By – Nishit Mehta

Introduction To Programming

Decimal Number System (DNS):

In Decimal Number System there are **10** unique digits i.e. {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}. Therefore, DNS is **Base**$_{10}$.

Binary Number System (BNS):

In Binary Number System there are **2** unique digits i.e. {0, 1}. Therefore, BNS is **Base**$_{2}$.
In Binary Number System,

$0 + 0 = 1$; $0 + 1 = 1$; $1 + 0 = 1$; $1 + 1 = 10$;

Octal Number System (ONS):

In Octal Number System there are **8** unique digits i.e. {0, 1, 2, 3, 4, 5, 6, 7}. Therefore, ONS is **Base**$_{8}$. In Octal Number System,

$3 + 4 = 7$; $4 + 5 = 11$; $5 + 6 = 13$; $23 = 27$;

HexaDecimal Number System (HNS):

In HexaDecimal Number System there are **16** unique digits i.e. {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}. Therefore, HNS is **Base**$_{16}$. In HexaDecimal Number System,

$3 + 4 = 7$; $10 = A$; $15 = F$; $15 + 10 = 19$; $15 + 12 = 1B$;

Decimal Numbers	Binary Numbers	Octal Numbers	HexaDecimal Numbers
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	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
16	10000	20	10
17	10001	21	11
18	10010	22	12

To change number in any number system just check the **base** of that number system i.e., for decimal system the base is **10**, for binary system the base is **2**, for octal system the base is **8** and for hexadecimal system the base is **16**.

Decimal System to any Number System Conversion:

Formula:

Base	Quotient	Remainder

We'll divide the **Base** with the **Quotient** till the **Quotient** column becomes **0**, and write the number system form of the decimal number in the reverse order of the **Remainder** column.

- Suppose we need to find **27** in its **Binary** form.

2	**27**	**Remainder (in reverse order)**
2	13	**1**
2	6	**1**
2	3	**0**
2	1	**1**
2	**0**	**1**

Therefore, **27** in binary form is **11011**.

- Suppose we need to find **43** in its **Binary** form.

2	**43**	**Remainder (in reverse order)**
2	21	**1**
2	10	**1**
2	5	**0**
2	2	**1**
2	1	**0**
2	**0**	**1**

Therefore, **43** in binary form is **101011**.

Any Number System to Decimal System Conversion:

Let's consider a number **278**, we can write this number as:

$$2 \times 10^2 + 7 \times 10^1 + 8 \times 10^0 = 278$$

In the above example we can see that the base is **10**. Similarly, a binary number, octal number and hexadecimal number can be converted to its equivalent decimal from with base as **2, 8** and **16** respectively.

- Suppose we need to find **101** in its **Decimal** form.
  We can convert **101** as:
  $$1 \times \mathbf{2^2} + 0 \times \mathbf{2^1} + 1 \times \mathbf{2^0} = 4 + 0 + 1 = \mathbf{5}$$
  Therefore, **101** in decimal form is **5**.

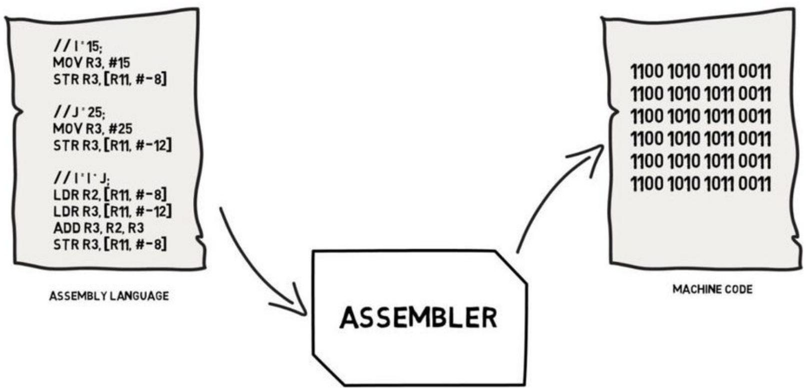
- Suppose we need to find **110101** in its **Decimal** form.
  We can convert **110101** as:
  $$1 \times \mathbf{2^5} + 1 \times \mathbf{2^4} + 0 \times \mathbf{2^3} + 1 \times \mathbf{2^2} + 0 \times \mathbf{2^1} + 1 \times \mathbf{2^0} = 32 + 16 + 0 + 4 + 0 + 1 = \mathbf{53}$$
  Therefore, **110101** in decimal form is **53**.

- Suppose we need to find **AC2** in its **Decimal** form.
  We can convert **AC2** as:
  $$10 \times \mathbf{16^2} + 12 \times \mathbf{16^1} + 2 \times \mathbf{16^0} = 2560 + 192 + 2 = \mathbf{2754}$$

Moore's Law:
It states that, in every 2 years the capacity of transistor will be increased by double.

History of Languages:

A computer only understands **binary form**. The language which computer understands is called **Machine Language**. A Machine Language is that language which consists of either **'0'** or **'1'**.
Eg: 010101000101;



To write Machine Language manually was a tough task and sometimes mistakes might happen. Therefore, to resolve this issue, an **assembler** was introduced which converts the **assembly language** submitted by user to computer into machine language (understandable to computer) i.e., **assembler** worked as a translator between user and computer.

Later on, more advancements were made to make a language which was similar to our speaking language which we termed as **high-level language**.
High-Level Language made our writing code convenient than before.
Eg: Java, Python, etc.

Machine Language is faster than **Assembly Language** and **Assembly Language** is faster than **High-Level Language.** This is because Assembly Language and High-Level Language are converted to Machine Language via some medium whereas Machine Language requires no conversion and is directly understandable to computer.

Data:
Programmers main goal should be to store maximum amount of data in minimum space, fetch data in less amount of time and arrange data in a sorted fashion.

Introduction To Programming (Homework)

1. **Convert Decimal to Binary**
   a. 37
   b. 92
   c. 128
   d. 243

2. **Convert Binary to Decimal**
   a. 1011
   b. 111001
   c. 10011011
   d. 10100100

3. **Convert Decimal to Octal**
   a. 28
   b. 47
   c. 928
   d. 1243

4. **Convert Octal to Decimal**
   a. 41
   b. 207
   c. 124

d. 311

5. **Convert Decimal to HexaDecimal**
   a. 317
   b. 41
   c. 14
   d. 845

6. **Convert HexaDecimal to Decimal**
   a. A11
   b. 49
   c. AE2F
   d. D97

Solutions

1. a.

2	**37**	
2	18	1
2	9	0
2	4	1
2	2	0
2	1	0
2	0	1

37 in binary form is **100101**.

b.

2	**92**	
2	46	0
2	23	0
2	11	1
2	5	1
2	2	1
2	1	0
2	0	1

92 in binary form is **1011100**.

c.

2	**128**	
2	64	0
2	32	0
2	16	0
2	8	0

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

128 in binary form is **10000000**.

d.

2	**243**	
2	121	1
2	60	1
2	30	0
2	15	0
2	7	1
2	3	1
2	1	1
2	0	1

243 in binary form is **11110011**.

2. a.

$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11$

1011 in decimal form is **11**.

b.

$1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 16 + 8 + 0 + 0 + 1 = 57$

111001 in decimal form is **57**.

c.

$1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 128 + 0 + 0 + 16 + 8 + 0 + 2 + 1 = 155$

10011011 in decimal form is **155**.

d.

$1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 128 + 0 + 32 + 0 + 0 + 4 + 0 + 0 = 164$

10100100 in decimal form is **164**.

3. a.

8	**28**	
8	3	4
8	0	3

28 in octal form is **<u>34</u>**.

b.

8	**47**	
8	5	7
8	0	5

47 in octal form is **<u>57</u>**.

c.

8	**47**	
8	5	7
8	0	5

47 in octal form is **<u>57</u>**.

d.

8	**928**	
8	116	0
8	14	4
8	1	6
8	0	1

928 in octal form is **<u>1640</u>**.

4. a.

$4 \times 8^1 + 1 \times 8^0 = 32 + 1 = 33$

41 in decimal form is **<u>33</u>**.

b.

$2 \times 8^2 + 0 \times 8^1 + 7 \times 8^0 = 128 + 0 + 7 = 135$

207 in decimal form is **<u>135</u>**.

c.

$1 \times 8^2 + 2 \times 8^1 + 4 \times 8^0 = 64 + 16 + 4 = 84$
124 in decimal form is **84**.

d.

$3 \times 8^2 + 1 \times 8^1 + 1 \times 8^0 = 192 + 8 + 1 = 201$
311 in decimal form is **201**.

5. a.

16	**317**	
16	19	D
16	1	3
16	0	1

317 in hexadecimal form is **13D**.

b.

16	**41**	
16	2	9
16	0	2

41 in hexadecimal form is **29**.

c.

16	**14**	
16	0	E

14 in hexadecimal form is **E**.

d.

16	**845**	
16	52	D
16	3	4
16	0	3

845 in hexadecimal form is **34D**.

6. a.

$A \times 16^2 + 1 \times 16^1 + 1 \times 16^0 = 2560 + 16 + 1 = 2577$

A11 in decimal form is **<u>2577</u>**.

b.

$4 \times 16^1 + 9 \times 16^0 = 64 + 9 = 73$

49 in decimal form is **<u>73</u>**.

c.

$A \times 16^3 + E \times 16^2 + 2 \times 16^1 + F \times 16^0 = 40960 + 3584 + 32 + 15 = 44591$

AE2F in decimal form is **<u>44591</u>**.

d.

$D \times 16^2 + 9 \times 16^1 + 7 \times 16^0 = 3328 + 144 + 7 = 3479$

D97 in decimal form is **<u>3479</u>**.
