Example 19.1

Change the following IPv4 addresses from binary notation to dotted-decimal notation.

a. 10000001 00001011 00001011 11101111
b. 11000001 10000011 00011011 11111111

Solution

We replace each group of 8 bits with its equivalent decimal number (see Appendix B) and add dots for separation.

a. 129.11.11.239
b. 193.131.27.255

Example 19.2

Change the following IPv4 addresses from dotted-decimal notation to binary notation.

a. 111.56.45.78
b. 221.34.7.82

Solution

We replace each decimal number with its binary equivalent (see Appendix B).

a. 01101111 00111000 00101101 01001110
b. 11011101 00100010 00000111 01010010

Example 19.3

Find the error, if any, in the following IPv4 addresses.

a. 111.56.045.78
b. 221.34.7.8.20
c. 75.45.301.14
d. 11100010.23.14.67

Solution

a. There must be no leading zero (045).
b. There can be no more than four numbers in an IPv4 address.
c. Each number needs to be less than or equal to 255 (301 is outside this range).
d. A mixture of binary notation and dotted-decimal notation is not allowed.

IPv4 Address Rules (Reminder)

An IPv4 address:

- Consists of **four decimal numbers (octets)** separated by dots.
- Each octet must be **between 0 and 255**.
- Uses **only dotted-decimal notation**.
- **No leading zeros** are allowed in any octet.

a. `111.56.045.78`

❌ **Error:** Leading zero in one octet.

Explanation:

The octet `045` has a leading zero. IPv4 addresses do not allow leading zeros because they can cause ambiguity.

✔ Correct form would be: `111.56.45.78`

b. `221.34.7.8.20`

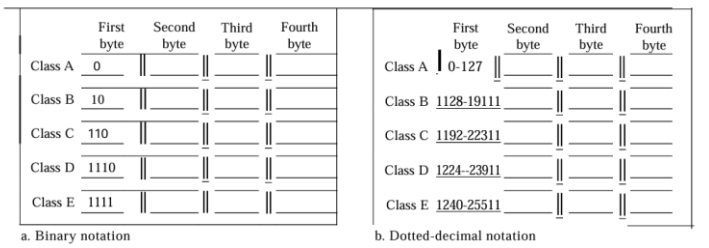
❌ **Error:** Too many octets.

Explanation:

An IPv4 address must contain **exactly four numbers**, but this address has **five**.

Therefore, it is invalid.

Figure 19.2 *Finding the classes in binary and dotted-decimal notation*



See b

Example 19.4

Find the class of each address.

a. 00000001 00001011 00001011 11101111
b. 11000001 10000011 00011011 11111111
c. 14.23.120.8
d. 252.5.15.111

Solution

a. The first bit is 0. This is a class A address.
b. The first 2 bits are 1; the third bit is 0. This is a class C address.
c. The first byte is 14 (between 0 and 127); the class is A.
d. The first byte is 252 (between 240 and 255); the class is E.

class A 0-127
class B 128-191
class C 192-223
class D 224-239
class E 240-255

c. `75.45.301.14`

❌ **Error:** Octet value out of range.

Explanation:

Each octet must be between **0 and 255**.

The value `301` exceeds this limit, making the address invalid.

d. `11100010.23.14.67`

❌ **Error:** Mixing number formats.

Explanation:

The first octet `11100010` is written in **binary**, while the rest are in **decimal**.

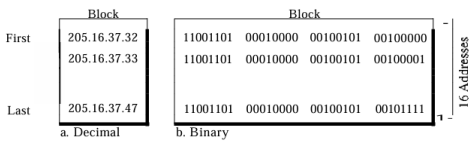
IPv4 addresses must use **only dotted-decimal notation** throughout.

Mixing binary and decimal formats is not allowed.

Example 19.5

Figure 19.3 shows a block of addresses, in both binary and dotted-decimal notation, granted to a small business that needs 16 addresses.

Figure 19.3 *A block of 16 addresses granted to a small organization*



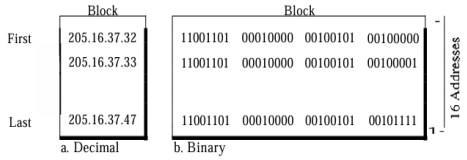
Example 19.6

A block of addresses is granted to a small organization. We know that one of the addresses is 205.16.37.39/28. What is the first address in the block?

Solution

The binary representation of the given address is 11001101 00010000 00100101 00100111. If we set 32 - 28 rightmost bits to 0, we get 11001101 000100000100101 0010000 or 205.16.37.32. This is actually the block shown in Figure 19.3.

Figure 19.3 *A block of* 16 *addresses granted to a small organization*



Last Address The last address in the block can be found by setting the 32 - *n* rightmost bits in the binary notation of the address to Is.

The last address in the block can be found by setting the rightmost 32 - *n* bits to Is.

Example 19.7

Find the last address for the block in Example 19.6.

Solution

The binary representation of the given address is 11001101 00010000001001010010 0111. If we set 32 - 28 rightmost bits to 1, we get 11001101 00010000 00100101001011111 or 205.16.37.47. This is actually the block shown in Figure 19.3.

Number of Addresses The number of addresses in the block is the difference between the last and first address. It can easily be found using the formula 2^{32-n}.

The number of addresses in the block can be found by using the formula 2^{32-n}.

Example 19.8

Find the number of addresses in Example 19.6.

Solution

The value of *n* is 28, which means that number of addresses is 2^{32-28} or 16.

Key Concept (Exam Tip)

- **First address:** all host bits = `0`
- **Last address:** all host bits = `1`
- In a `/28` block, addresses increase in steps of **16**

Example 19.9

Another way to find the first address, the last address, and the number of addresses is to represent the mask as a 32-bit binary (or 8-digit hexadecimal) number. This is particularly useful when we are writing a program to find these pieces of information. In Example 19.5 the /28 can be represented as 11111111 11111111 11111111 11110000 (twenty-eight Is and four Os). Find

a. The first address
h. The last address
c. The number of addresses

Solution

a. The first address can be found by ANDing the given addresses with the mask. ANDing here is done bit by bit. The result of ANDing 2 bits is 1 if both bits are Is; the result is 0 otherwise.

Address:	11001101 00010000 00100101 00100111
Mask:	11111111 11111111 11111111 11110000
First address:	11001101 00010000 00100101 00100000

b. The last address can be found by ORing the given addresses with the complement of the mask. ORing here is done bit by bit. The result of ORing 2 bits is 0 if both bits are Os; the result is 1 otherwise. The complement of a number is found by changing each 1 to 0 and each 0 to 1.

Address:	11001101 00010000 00100101 00100111
Mask complement:	00000000 00000000 00000000 00001111
Last address:	11001101 00010000 *00100101 00101111*

c. The number of addresses can be found by complementing the mask, interpreting it as a decimal number, and adding 1 to it.

Mask complement:	00000000 00000000 00000000 00001111
Number of addresses:	15 + 1 = 16

Example 19.9

Another way to find the **first address**, **last address**, and **number of addresses** is to represent the **subnet mask** as a **32-bit binary number**.

For **/28**, the mask is:

```
11111111 11111111 11111111 11110000
```

(28 ones and 4 zeros)

Given address:

```
205.16.37.39
```

Binary form of the address:

```
11001101 00010000 00100101 00100111
```

a. Finding the first address

The **first address** is obtained by **ANDing** the address with the mask.

AND operation rule:

- 1 AND 1 = 1
- Otherwise = 0

```
Address:      11001101 00010000 00100101 00100111
Mask:         11111111 11111111 11111111 11110000
-----------------------------------------------
First address:11001101 00010000 00100101 00100000
```

Convert back to decimal:

```
205.16.37.32
```

✅ **First address = 205.16.37.32**

b. Finding the last address

The **last address** is found by **ORing** the address with the **complement of the mask**.

Mask complement

(Change 1 → 0 and 0 → 1)

```
Mask:            11111111 11111111 11111111 11110000
Mask complement: 00000000 00000000 00000000 00001111
```

OR operation rule:

- 0 OR 0 = 0
- Otherwise = 1

```
Address:        11001101 00010000 00100101 00100111
Mask complement:00000000 00000000 00000000 00001111
-----------------------------------------------
Last address:   11001101 00010000 00100101 00101111
```

Convert back to decimal:

```
205.16.37.47
```

c. Finding the number of addresses

The number of addresses is found by:

1. Taking the **mask complement**
2. Converting it to decimal
3. Adding **1**

```cpp
Mask complement (last octet): 00001111 = 15
15 + 1 = 16
```

✅ **Number of addresses = 16**

Example 19.6

A block of IPv4 addresses is granted to a small organization. One address in the block is:

205.16.37.39/28

Find the **first address** in the block.

Step 1: Understand the prefix `/28`

- `/28` means **28 bits are for the network.**
- Remaining bits = **32 − 28 = 4 bits** for host addresses.
- A `/28` block contains **2^4 = 16 addresses.**

Step 2: Convert the given IP address to binary

Decimal	Binary
205	11001101
16	00010000
37	00100101
39	00100111

So the full binary form is:

```
11001101 00010000 00100101 00100111
```

Step 3: Set the host bits to 0

Since `/28`, the **last 4 bits** are host bits.

Setting the last 4 bits to **0**:

```
11001101 00010000 00100101 00100000
```

Step 4: Convert back to decimal

Binary	Decimal
11001101	205
00010000	16
00100101	37
00100000	32

✅ Final Answer

The **first address in the block** is:

205.16.37.32

Example 19.7

Find the last address for the block in Example 19.6.

Given address: **205.16.37.39/28**

Step 1: Understand the prefix `/28`

- `/28` means **28 network bits**
- Host bits = **32 − 28 = 4**
- Total addresses in the block = **2^4 = 16**

Step 2: Convert the given IP address to binary

Decimal	Binary
205	11001101
16	00010000
37	00100101
39	00100111

Binary representation:

```
11001101 00010000 00100101 00100111
```

Step 3: Set the host bits to 1

To find the **last address**, set the **last 4 bits (host bits)** to **1**:

```
11001101 00010000 00100101 00101111
```

Step 4: Convert back to decimal

Binary	Decimal
11001101	205
00010000	16
00100101	37
00101111	47

✅ Final Answer

The **last address in the block** is:

205.16.37.47