Number Systems

Computer Mathematics I

Jiraporn Pooksook
Department of Electrical and Computer Engineering
Naresuan University

What is a bit?

- A bit stands for binary digit.
- A bit is merely true or false.
- It can be considered as 0 or 1.
- Each bit represents a unit of information.
- Most computers use blocks of 8 bits or bytes as the smallest addressable unit of memory.

Number Notations

Most computers count in binary while we can understand as decimal numbers.
$301.75 = 3 \times 10^2 + 0 \times 10^1 + 1 \times 10^0 + 7 \times 10^{-1} + 5 \times 10^{-2}$

- 301.75 is on base-10.
- 301 is integer part.
- 75 is fractional part.
- 3 is the most significant digit.
- 5 is the least significant digit.

Binary Notations

Most computers count in binary while we can understand as decimal numbers.
$22 = 2 \times 10^1 + 2 \times 10^0$

Binary has only 2 digits so the base becomes 2.

$$\begin{aligned} 10110 &= 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 16{+}0{+}4{+}2{+}0 = 22 \end{aligned}$$

$$\begin{aligned} 10.110 &= 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} \\ &= 2.75 \end{aligned}$$

Other Notations

Higher bases make for shorter numbers that are easier to manipulate.
Octal is base-8 ($8=2^3$ digits, which means 3 binary bits per digit)
$22_{10} = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 10\ 110$
$10\ 110 = 26_8$ ($2 \times 8^1 + 6 \times 8^0$ =22)

Hexadecimal is base-16 ($16=2^4$ digits, which means 4 binary bits per digit) $22_{10} = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$
$= 1\ 0110$
$1\ 0110 = 16_{16}$ ($1 \times 16^1 + 6 \times 16^0$ =22)

Bytes

A single byte consists of 8 bits. In binary notation its values ranges from 00000000_2 to 11111111_2 which is from 0_{10} to 255_{10}.

We write bit patterns as base-16 or hexadecimal numbers which use digits 0-9 and letters A-F.

In C programming language, numeric constants start with ’0x’ or ’0X’ are interpreted as being in hexadecimal.

Hex digit	0	1	2	3	4	5	6	7
Decimal value	0	1	2	3	4	5	6	7
Binary value	00	01	10	11	100	101	110	111
Hex digit	8	9	A	B	C	D	E	F
Decimal value	8	9	10	11	12	13	14	15
Binary value	1000	1001	1010	1011	1100	1101	1110	1111

Practice

Convert 12 based-10 to based-2 ($12_{10} = ?_2$)

Answer:

base	2^4	2^3	2^2	2^1	2^0	
multiply	16	8	4	2	1	
binary	0	1	1	0	0	
sum	0	8	4	0	0	$= 12$

$12_{10} = 8{+}4 = 1100_2$

Practice

Convert 12 based-10 to based-8 ($12_{10} = ?_8$)

Answer:

base	8^2	8^1	8^0	
multiply	64	8	1	
binary	0	1	4	
sum	0	8	4	$= 12$

$12_{10} = 8{+}4 = 14_8$

Practice

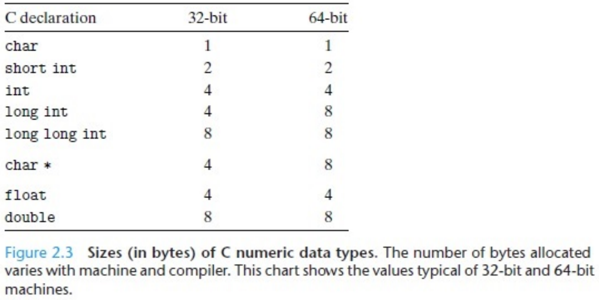
Convert 27 based-10 to based-16 ($27_{10} = ?_{16}$)

Answer:

base	16^2	16^1	16^0	
multiply	256	16	1	
binary	0	1	B	
sum	0	16	11	$= 27$

$27_{10} = 16+11 = 1B_{16}$

Data Size

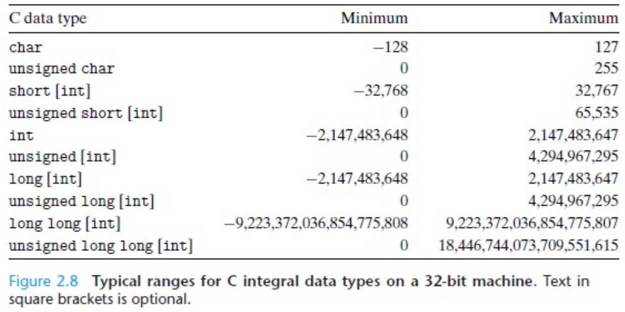


C declaration	32-bit	64-bit
`char`	1	1
`short int`	2	2
`int`	4	4
`long int`	4	8
`long long int`	8	8
`char *`	4	8
`float`	4	4
`double`	8	8

Figure 2.3 **Sizes (in bytes) of C numeric data types.** The number of bytes allocated varies with machine and compiler. This chart shows the values typical of 32-bit and 64-bit machines.

Figure: Retrieved from Computer systems : a programmer's perspective / Randal E. Bryant, David R. O'Hallaron.-2nd ed.

Data Representations

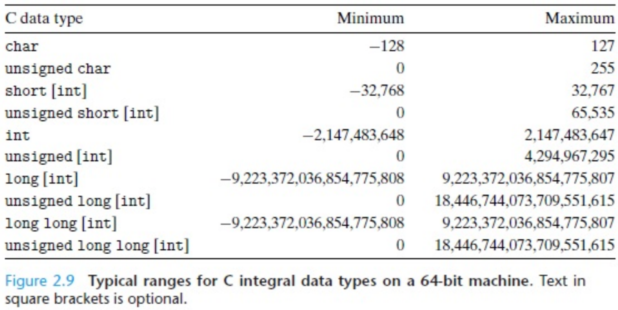


C data type	Minimum	Maximum
`char`	−128	127
`unsigned char`	0	255
`short [int]`	−32,768	32,767
`unsigned short [int]`	0	65,535
`int`	−2,147,483,648	2,147,483,647
`unsigned [int]`	0	4,294,967,295
`long [int]`	−2,147,483,648	2,147,483,647
`unsigned long [int]`	0	4,294,967,295
`long long [int]`	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
`unsigned long long [int]`	0	18,446,744,073,709,551,615

Figure 2.8 **Typical ranges for C integral data types on a 32-bit machine.** Text in square brackets is optional.

Figure: Retrieved from Computer systems : a programmer's perspective / Randal E. Bryant, David R. O'Hallaron.-2nd ed.

Data Representations

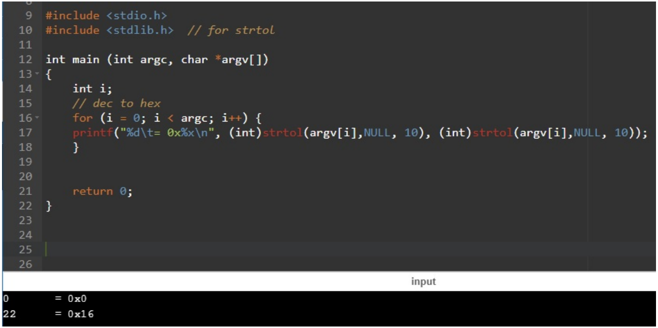


C data type	Minimum	Maximum
char	−128	127
unsigned char	0	255
short [int]	−32,768	32,767
unsigned short [int]	0	65,535
int	−2,147,483,648	2,147,483,647
unsigned [int]	0	4,294,967,295
long [int]	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long [int]	0	18,446,744,073,709,551,615
long long [int]	−9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long long [int]	0	18,446,744,073,709,551,615

Figure 2.9 **Typical ranges for C integral data types on a 64-bit machine.** Text in square brackets is optional.

Figure: Retrieved from Computer systems : a programmer's perspective / Randal E. Bryant, David R. O'Hallaron.-2nd ed.

Practice: Base Conversion

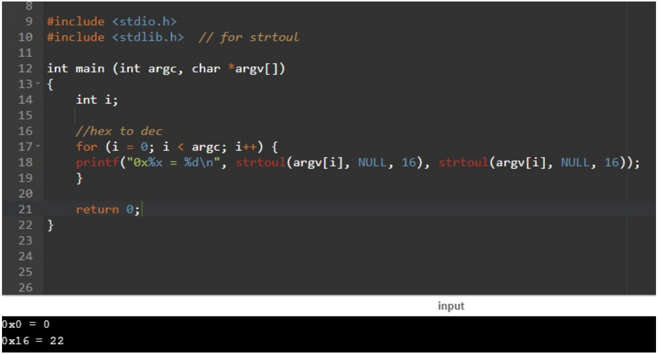


```c
#include <stdio.h>
#include <stdlib.h>  // for strtol

int main (int argc, char *argv[])
{
    int i;
    // dec to hex
    for (i = 0; i < argc; i++) {
    printf("%d\t= 0x%x\n", (int)strtol(argv[i],NULL, 10), (int)strtol(argv[i],NULL, 10));
    }


    return 0;
}
```

```
0        = 0x0
22       = 0x16
```

Figure: Slightly changed from Computer systems : a programmer's perspective / Randal E. Bryant, David R. O'Hallaron.-2nd ed.

Practice: Base Conversion



```c
#include <stdio.h>
#include <stdlib.h>  // for strtoul

int main (int argc, char *argv[])
{
    int i;

    //hex to dec
    for (i = 0; i < argc; i++) {
    printf("0x%x = %d\n", strtoul(argv[i], NULL, 16), strtoul(argv[i], NULL, 16));
    }

    return 0;
}
```

```
0x0 = 0
0x16 = 22
```

Figure: Slightly changed from Computer systems : a programmer's perspective / Randal E. Bryant, David R. O'Hallaron.-2nd ed.

Exercise

Question 1:
Convert 53 based-10 to based-2 ($53_{10} = ?_2$)

Question 2:
Convert 53 based-10 to based-8 ($53_{10} = ?_8$)

Question 3:
Convert 53 based-10 to based-16 ($53_{10} = ?_{16}$)

Exercise

Question 4:
Convert 100101 based-2 to based-10 ($100101_2 = ?_{10}$)

Question 5:
Convert 105 based-8 to based-10 ($105_8 = ?_{10}$)

Question 6:
Convert 45 based-16 to based-10 ($45_{16} = ?_{10}$)