Assembly Programming

Lecture 2: Number System

Yan Pang
yanpang@gzhu.edu.cn

Binary Numbers

TABLE 2–1

Decimal Number	Binary Number			
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

In general, with n bits you can count up to a number equal to ?

Binary Numbers

In general, with n bits you can count up to a number equal to $2^n - 1$.

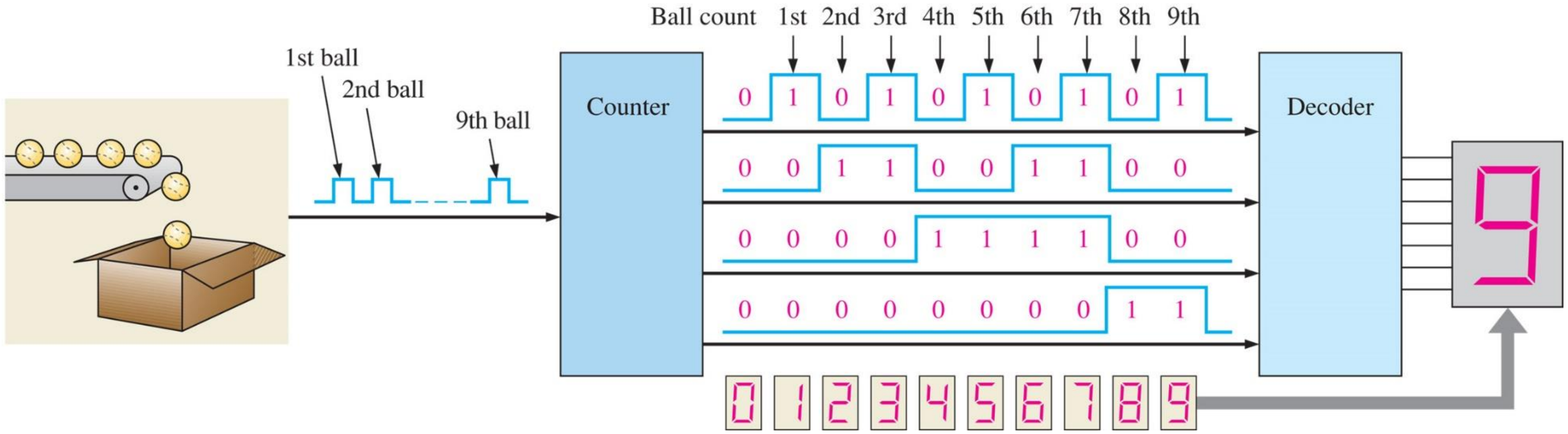
TABLE 2–1

Decimal Number	Binary Number			
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

In general, with n bits you can count up to a number equal to $2^n - 1$.

The largest decimal number is:
$2^4 - 1 = 15$

Binary Counting Application



A example of counting tennis balls going into a box from a conveyor belt.

Weight Structure of Binary

TABLE 2–2

Binary weights.

Positive Powers of Two (Whole Numbers)									Negative Powers of Two (Fractional Number)					
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
256	128	64	32	16	8	4	2	1	1/2 0.5	1/4 0.25	1/8 0.125	1/16 0.625	1/32 0.03125	1/64 0.015625

Binary Point

The largest decimal number

Binary to Decimal Conversion

Binary: 1101101

Decimal: ?

Binary to Decimal Conversion

Binary: 1101101

Decimal: ?

Determine the weight of each bit that is a 1, and then find the sum of the weights to get the decimal number.

Weight:	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Binary number:	1	1	0	1	1	0	1

$$1101101 = 2^6 + 2^5 + 2^3 + 2^2 + 2^0$$
$$= 64 + 32 + 8 + 4 + 1 = \mathbf{109}$$

Binary to Decimal Conversion

Binary: 0.1011

Decimal: ?

Binary to Decimal Conversion

Binary: 0.1011

Decimal: ?

Determine the weight of each bit that is a 1, and then sum the weights to get the decimal fraction.

Weight:	2^{-1}	2^{-2}	2^{-3}	2^{-4}
Binary number:	0 . 1	0	1	1

$$0.1011 = 2^{-1} + 2^{-3} + 2^{-4}$$
$$= 0.5 + 0.125 + 0.0625 = \mathbf{0.6875}$$

Decimal to Binary Conversion

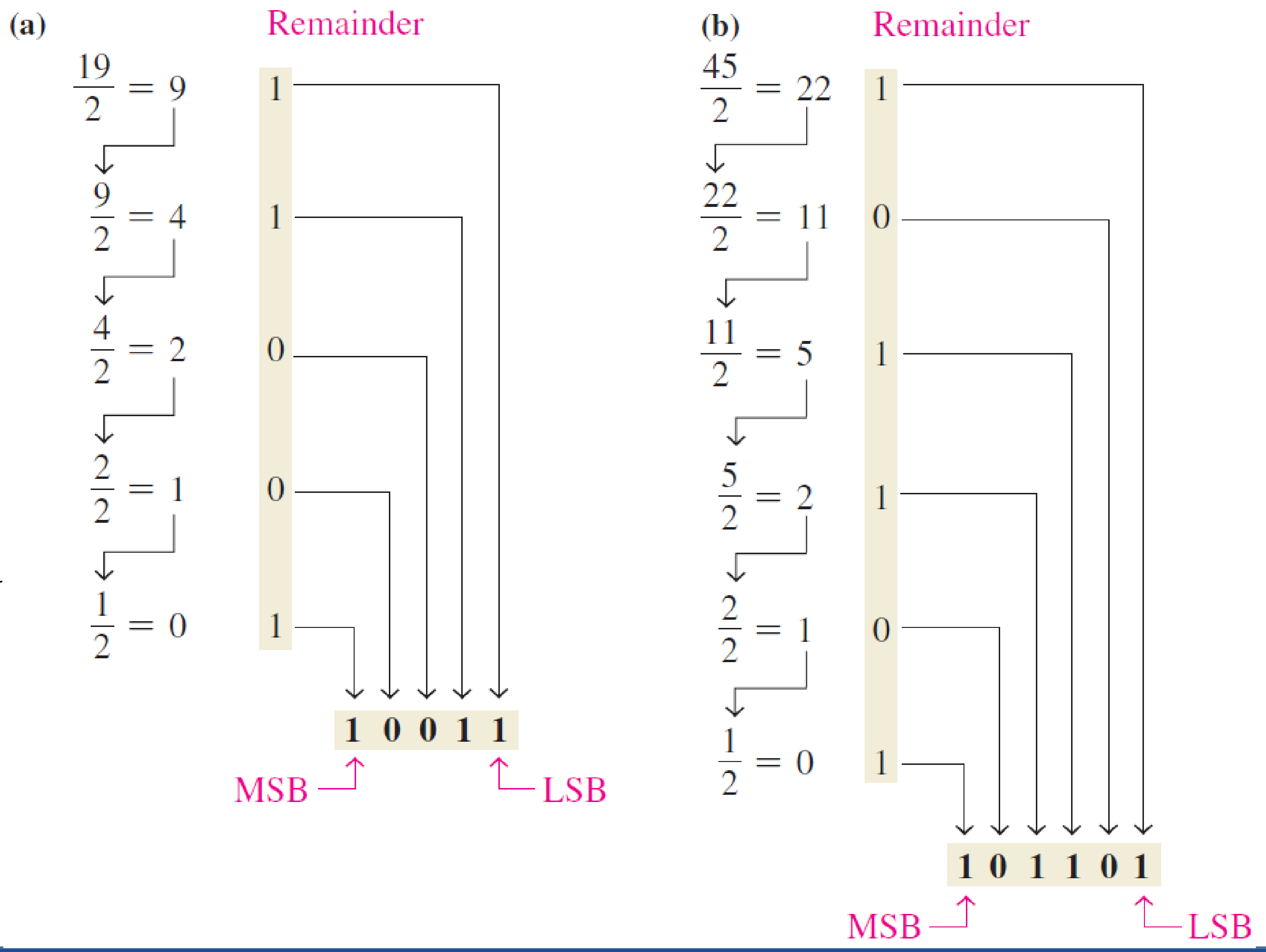
Repeated Division-by-2:

First, divide the decimal number by 2;

Then divide each resulting quotient by 2 until there is a 0 whole-number quotient. The remainders generated by each division form the binary number.

MSB: The most significant bit

LSB: The least significant bit

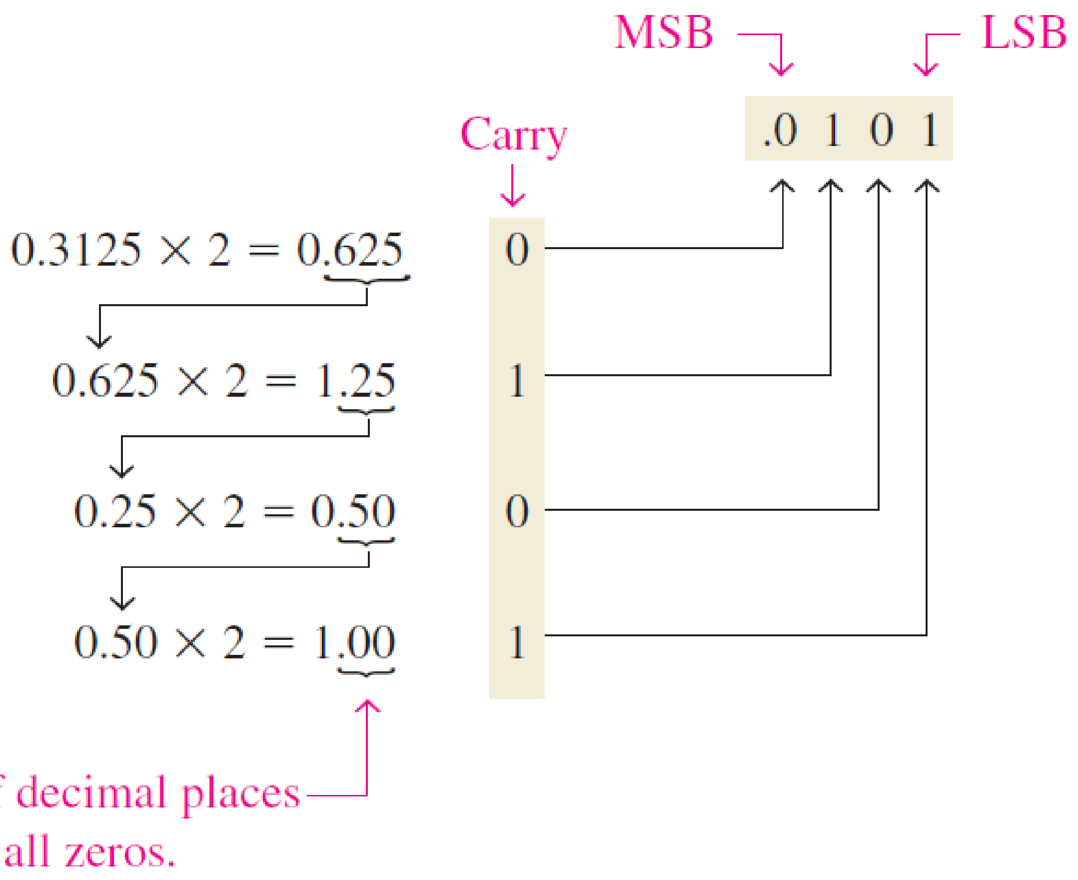


Convert Decimal Fractions to Binary

Repeated Multiplication-by-2 Method:

First, multiply the decimal number by 2;

Then multiply each resulting fraction part of the product by 2 until the fractional product is zero or until the desired number of decimal places is reached.



Practice

$(1100\ 0011)_2 = \qquad _{10}$ $(0111\ 0010)_2 = \qquad _{10}$ $(1111\ 1111)_2 = \qquad _{10}$

$(0010.0100)_2 = \qquad _{10}$ $(1101.0001)_2 = \qquad _{10}$ $(1111.1111)_2 = \qquad _{10}$

$(234)_{10} = \qquad _2$ $(64)_{10} = \qquad _2$ $(111)_{10} = \qquad _2$

$(1.1875)_{10} = \qquad _2$ $(0.375)_{10} = \qquad _2$ $(0.4375)_{10} = \qquad _2$

Practice

$(1100\ 0011)_2=195_{10}$ $(0111\ 0010)_2 = 114_{10}$ $(1111\ 1111)_2=255_{10}$

$(0010.0100)_2=2.25_{10}$ $(1101.0001)_2=13.0625_{10}$ $(1111.1111)_2=15.9375_{10}$

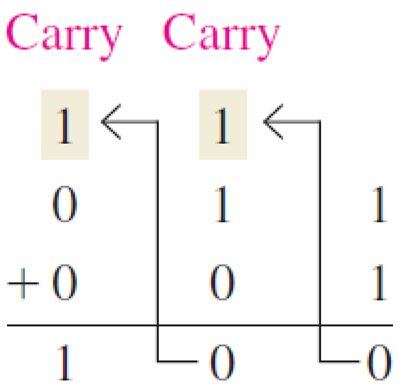
$(234)_{10}=11101010_2$ $(64)_{10}=01000000_2$ $(111)_{10}=01101111_2$

$(1.1875)_{10}=1.0011_2$ $(0.375)_{10}=0.011_2$ $(0.4375)_{10}=0.0111_2$

Binary Arithmetic

Binary Addition:

$0 + 0 = 0$	Sum of 0 with a carry of 0
$0 + 1 = 1$	Sum of 1 with a carry of 0
$1 + 0 = 1$	Sum of 1 with a carry of 0
$1 + 1 = 10$	Sum of 0 with a carry of 1



Add the following binary numbers:

a) 11 + 11
b) 100 + 10
c) 111 + 11
d) 110 + 100

(a)

$$\begin{array}{r} 11 \\ +\ 11 \\ \hline \mathbf{110} \end{array} \qquad \begin{array}{r} 3 \\ +\ 3 \\ \hline 6 \end{array}$$

(b)

$$\begin{array}{r} 100 \\ +\ 10 \\ \hline \mathbf{110} \end{array} \qquad \begin{array}{r} 4 \\ +\ 2 \\ \hline 6 \end{array}$$

(c)

$$\begin{array}{r} 111 \\ +\ 11 \\ \hline \mathbf{1010} \end{array} \qquad \begin{array}{r} 7 \\ +\ 3 \\ \hline 10 \end{array}$$

(d)

$$\begin{array}{r} 110 \\ +\ 100 \\ \hline \mathbf{1010} \end{array} \qquad \begin{array}{r} 6 \\ +\ 4 \\ \hline 10 \end{array}$$

Binary Arithmetic

Binary Subtraction:

$$0 - 0 = 0$$
$$1 - 1 = 0$$
$$1 - 0 = 1$$
$$10 - 1 = 1 \qquad 0 - 1 \text{ with a borrow of } 1$$

$$\text{(a)}\quad \begin{array}{r} 11 \\ -01 \\ \hline \mathbf{10} \end{array} \qquad \begin{array}{r} 3 \\ -1 \\ \hline 2 \end{array} \qquad \text{(b)}\quad \begin{array}{r} 11 \\ -10 \\ \hline \mathbf{01} \end{array} \qquad \begin{array}{r} 3 \\ -2 \\ \hline 1 \end{array}$$

111 – 011:

Left column:
When a 1 is borrowed,
a 0 is left, so 0 − 0 = 0.

Middle column:
Borrow 1 from next column
to the left, making a 10 in
this column, then 10 − 1 = 1.

$$\begin{array}{r} \overset{0}{\cancel{1}}\,{}^{1}0\,1 \\ -0\,1\,1 \\ \hline 0\,1\,0 \end{array}$$

Right column:
1 − 1 = 0

Binary Arithmetic

Binary Multiplication:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

(a)

$$\begin{array}{lr}
 & 11 \\
 & \times\ 11 \\ \hline
\text{Partial} & 11 \\
\text{products} & +11 \\ \hline
 & \mathbf{1001}
\end{array}
\qquad
\begin{array}{r}
3 \\
\times\ 3 \\ \hline
9
\end{array}$$

(b)

$$\begin{array}{lr}
 & 111 \\
 & \times\ 101 \\ \hline
 & 111 \\
\text{Partial} & 000 \\
\text{products} & +111 \\ \hline
 & \mathbf{100011}
\end{array}
\qquad
\begin{array}{r}
7 \\
\times\ 5 \\ \hline
35
\end{array}$$

Binary Division:

(a)

$$\begin{array}{r}
\mathbf{10} \\
11\overline{)110} \\
\underline{11} \\
000
\end{array}
\qquad
\begin{array}{r}
2 \\
3\overline{)6} \\
\underline{6} \\
0
\end{array}$$

(b)

$$\begin{array}{r}
\mathbf{11} \\
10\overline{)110} \\
\underline{10} \\
10 \\
\underline{10} \\
00
\end{array}
\qquad
\begin{array}{r}
3 \\
2\overline{)6} \\
\underline{6} \\
0
\end{array}$$

Both multiplication and division are performed with binary numbers in the same manner as with **decimal** numbers.

Complements of Binary Numbers

Finding the 1's Complement：

1 0 1 1 0 0 1 0	Binary number
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	
0 1 0 0 1 1 0 1	1's complement

Gate?

What is the *simplest* way to obtain the 1's complement of a binary number with a digital circuit?

Complements of Binary Numbers

Finding the 1’s Complement：

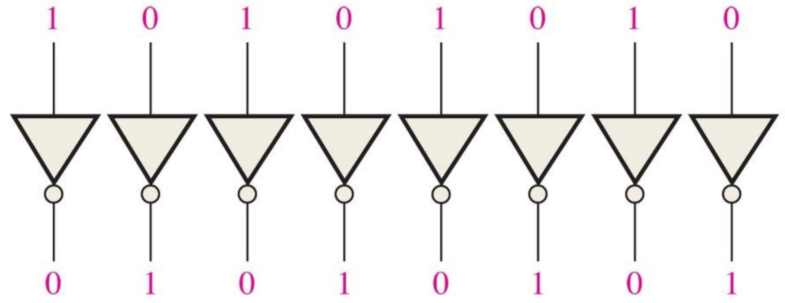
1 0 1 1 0 0 1 0 Binary number

↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

0 1 0 0 1 1 0 1 1’s complement

What is the *simplest* way to obtain the 1’s complement of a binary number with a digital circuit?

Gate?



Signed Numbers

Decimal: -25 and 25

Binary: -0001 1001 and 0001 1001

True? False?

How could we indicate the sign?

Signed Numbers

Decimal: -25 and 25

Binary: -0001 1001 and 0001 1001

False

How could we indicate the sign?

Signed Numbers

How could we indicate the sign?

The **left-most** bit in a signed binary number is the sign bit, which tells you whether the number is positive or negative.

A 0 sign bit indicates a positive number

A 1 sign bit indicates a negative number

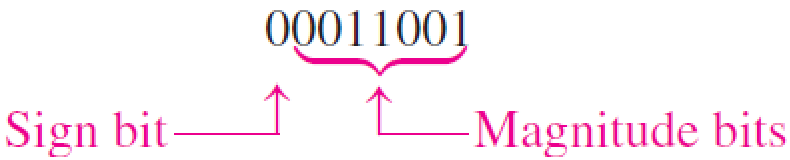
Signed Numbers

How could we indicate the sign?

The **left-most** bit in a signed binary number is the sign bit, which tells you whether the number is positive or negative.

A 0 sign bit indicates a positive number

A 1 sign bit indicates a negative number



positive 25

-25: 1001 1001

Arithmetic Operations with Signed N

Addition

1. Both number positive:

00000111	7	positive
+ 00000100	+ 4	+ positive
00001011	11	positive

Arithmetic Operations with Signed N

Addition

1. Both number positive:

$$\begin{array}{r} 00000111 \\ +\ 00000100 \\ \hline 00001011 \end{array} \qquad \begin{array}{r} 7 \\ +\ 4 \\ \hline 11 \end{array}$$

positive
+ positive
positive

2. Positive number with magnitude larger than negative number:

$$\begin{array}{rr} & 00001111 \\ & +\ 11111010 \\ \hline \boxed{1} & 00001001 \end{array} \qquad \begin{array}{r} 15 \\ +\ -6 \\ \hline 9 \end{array}$$

Discard carry $\longrightarrow$ 1

positive large
+ negative small
positive

Arithmetic Operations with Signed N

Addition

3. Both number negative :

	11111011	-5	negative
	+ 11110111	+ -9	+ negative
Discard carry →	1 11110010	-14	negative

4. Negative number with magnitude larger than positive number:

00010000	16	positive	small
+ 11101000	+ -24	+ negative	large
11111000	-8	negative	

Arithmetic Operations with Signed N

When two numbers are added and the number of bits required to represent the sum exceeds the number of bits in the two numbers, an overflow results as indicated by an *incorrect* sign bit.

Arithmetic Operations with Signed N

When two numbers are added and the number of bits required to represent the sum <u>exceeds</u> the number of bits in the two numbers, an overflow results as indicated by an *incorrect* sign bit.

01111101	125
+ 00111010	+ 58
10110111	183

Sign incorrect

Magnitude incorrect

- 55

183: 0000 1011 0111

Arithmetic Operations with Signed N

Multiplication

add + add + add + add … …

77	Multiplicand
× 4	Multiplier
308	Product

01001101	1st time
+ 01001101	2nd time
10011010	Partial sum
+ 01001101	3rd time
11100111	Partial sum
+ 01001101	4th time
100110100	Product

Multiplication

239	Multiplicand
×123	Multiplier
29397	Product

Arithmetic Operations with Signed N

Multiplication

239	Multiplicand
×123	Multiplier
29397	Product

add + add + add + add
\+ add + add + add + add
\+ add + add + add + add
… …
\+ add + add + add + add
\+ add + add + add + add
… …
\+ add + add + add + add
\+ add + add + add + add
\+ add + add + add + add

Add 123 times

Arithmetic Operations with Signed N

Multiplication

239	Multiplicand
×123	Multiplier
29397	Product

239	Multiplicand
× 123	Multiplier
717	1st partial product (3 × 239)
478	2nd partial product (2 × 239)
+ 239	3rd partial product (1 × 239)
29,397	Final product

Each successive partial product is **shift** **one** place to the **left**.

When all the partial products have been produced, they are added to get the final product.

Arithmetic Operations with Signed N

Multiplication

Unsigned

```
 10011  (19)    5 bits
 01011  (11)    5 bits
───────────────
0000010011
000010011
00000000
0010011
000000
───────────────
cut |0011010001     10 bits
Final product 0011010001 (209)
```

N bits * N bits

The total bits of final product is 2N.

- Starting with the least significant multiplier bit, generate the partial products;
- Shift each successive partial product one bit to the left, and put **0**s ahead to the partial products;
- Add each successive partial product to the sum of the previous partial products to get the final product.

Arithmetic Operations with Signed N

Multiplication

Signed

```
              10011   (-13)
              01011   (11)
      ____________________
         1111110011
         111110011
         00000000
         1110011
     (-) 000000
      ____________________
 cut  |1101110001   10 bits
Final product  1101110001   (-143)
```

```
              101   (-3)
              111   (-1)
      ____________________
          111101
          11101
   (-)    1101
      ____________________
 cut |   000011   6 bits
Final product   000011   (+3)
```

N bits * N bits

The total bits of final product is 2N.

- Starting with the least significant multiplier bit, generate the partial products;
- Shift each successive partial product one bit to the left, and put **same values of the leftmost bit** ahead to the partial products;
- Add the first **2N-1** partial products, and then **delete** the **last** partial product to get the final product.

Arithmetic Operations with Signed N

Division

- **If the signs are the same, the quotient is positive.**
- **If the signs are different, the quotient is negative.**

$$\frac{\text{dividend}}{\text{divisor}} = \text{quotient} \ \text{… remainder}$$

$21 \div 7$

21	Dividend
− 7	1st subtraction of divisor
14	1st partial remainder
− 7	2nd subtraction of divisor
7	2nd partial remainder
− 7	3rd subtraction of divisor
0	Zero remainder

- Do the subtraction (reversed addition) and saved the remainder;
- If (remainder < divisor)
  then break;
  Else
  go to the first step.
- The number of loops is the final quotient.

Hexadecimal Numbers

The hexadecimal number system has a base of sixteen; that is, it is composed of 16 **numeric** and alphabetic **characters**.

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

Binary to Hexadecimal Conversion

Break the binary number into 4-bit groups, starting at the right-most bit and replace each 4-bit group with the equivalent hexadecimal symbol.

(a) $\underbrace{1100}_{C}\underbrace{1010}_{A}\underbrace{0101}_{5}\underbrace{0111}_{7} = \mathbf{CA57_{16}}$

(b) $\underbrace{0011}_{3}\underbrace{1111}_{F}\underbrace{0001}_{1}\underbrace{0110}_{6}\underbrace{1001}_{9} = \mathbf{3F169_{16}}$

Two zeros have been added in part (b) to complete a 4-bit group at the left.

Hexadecimal to Binary Conversion

To convert from a hexadecimal number to a binary number, reverse the process and replace each hexadecimal symbol with the appropriate four bits.

(a) $\overbrace{1}^{1}\overbrace{0000}^{0}\overbrace{1010}^{A}\overbrace{0100}^{4}$

(b) $\overbrace{1100}^{C}\overbrace{1111}^{F}\overbrace{1000}^{8}\overbrace{1110}^{E}$

(c) $\overbrace{1001}^{9}\overbrace{0111}^{7}\overbrace{0100}^{4}\overbrace{0010}^{2}$

In part (a), the MSB is understood to have three zeros preceding it, thus forming a 4-bit group.

Hexadecimal to Decimal Conversion

First convert the hexadecimal number to binary and then convert from binary to decimal.

(a)

$$\overbrace{0001}^{1}\overbrace{1100}^{C} = 2^4 + 2^3 + 2^2 = 16 + 8 + 4 = \mathbf{28}_{10}$$

(b)

$$\overbrace{1010}^{A}\overbrace{1000}^{8}\overbrace{0101}^{5} = 2^{11} + 2^9 + 2^7 + 2^2 + 2^0 = 2048 + 512 + 128 + 4 + 1 = \mathbf{2693}_{10}$$

Hexadecimal to Decimal Conversion

Multiply the decimal value of each hexadecimal digit by its weight and then take the sum of these products.

For a 4-digit hexadecimal number, the weights are

16^3	16^2	16^1	16^0
4096	256	16	1

(a) $E5_{16} = (E \times 16) + (5 \times 1) = (14 \times 16) + (5 \times 1) = 224 + 5 = \mathbf{229_{10}}$

(b)
$$\begin{aligned} B2F8_{16} &= (B \times 4096) + (2 \times 256) + (F \times 16) + (8 \times 1) \\ &= (11 \times 4096) + (2 \times 256) + (15 \times 16) + (8 \times 1) \\ &= 45{,}056 + 512 + 240 + 8 = \mathbf{45{,}816_{10}} \end{aligned}$$

Decimal to Hexadecimal Conversion

fractional part

Hexadecimal remainder

$$\frac{650}{16} = 40.625 \rightarrow 0.625 \times 16 = 10 = \mathrm{A}$$

$$\frac{40}{16} = 2.5 \rightarrow 0.5 \times 16 = 8 = 8$$

$$\frac{2}{16} = 0.125 \rightarrow 0.125 \times 16 = 2 = 2$$

Stop when whole number quotient is zero.

2 8 A Hexadecimal number

MSD

LSD

Octal Numbers

The **octal** number system is composed of eight digits, which are

0, 1, 2, 3, 4, 5, 6, 7

To count above 7, begin another column and start over:

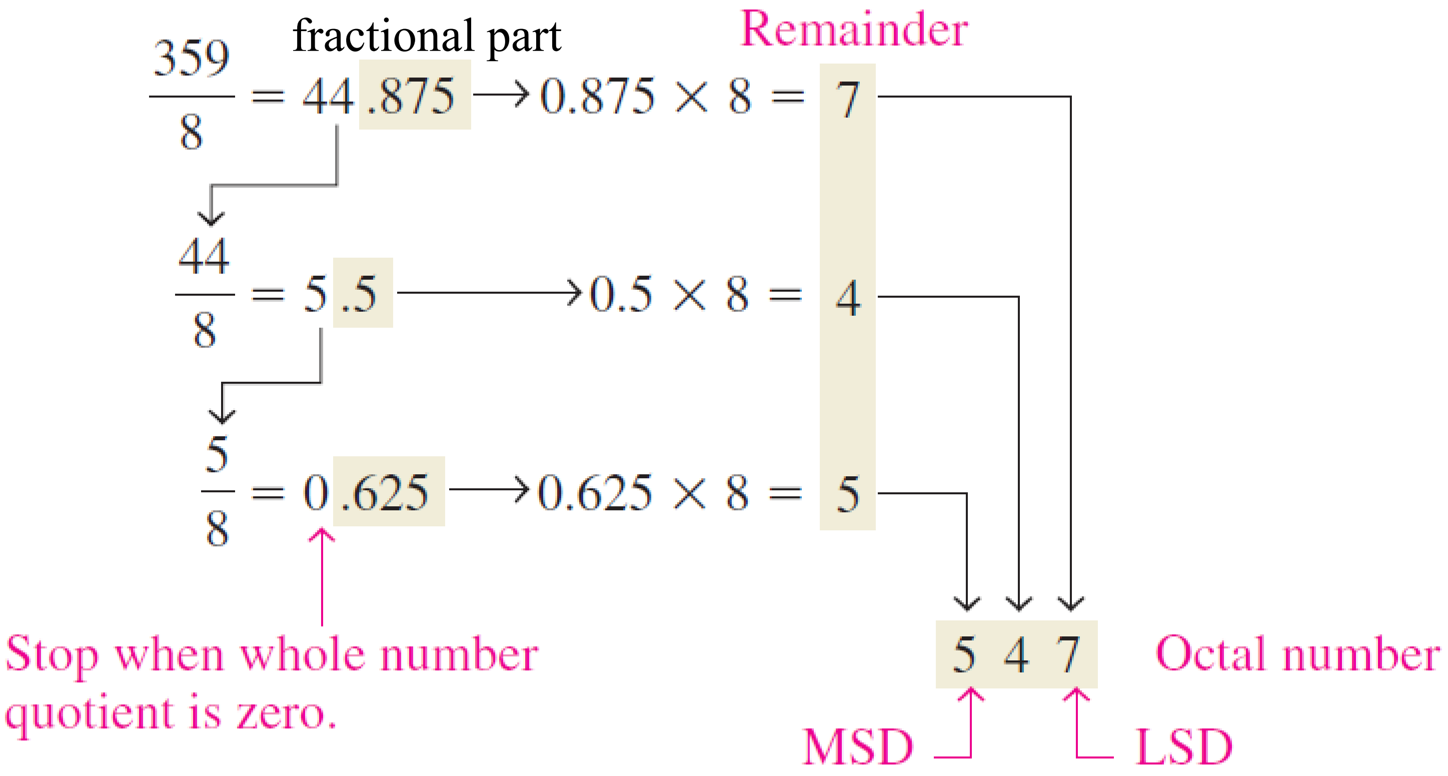
10, 11, 12, 13, 14, 15, 16, 17, 20, 21, . . .

Octal to Decimal Conversion

$$
\begin{array}{rl}
\text{Weight:} & 8^3\ 8^2\ 8^1\ 8^0 \\
\text{Octal number:} & 2\ \ 3\ \ 7\ \ 4
\end{array}
$$

$$
\begin{aligned}
2374_8 &= (2 \times 8^3) + (3 \times 8^2) + (7 \times 8^1) + (4 \times 8^0) \\
&= (2 \times 512) + (3 \times 64) + (7 \times 8) + (4 \times 1) \\
&= 1024 + 192 + 56 + 4 = 1276_{10}
\end{aligned}
$$

Decimal to Octal Conversion



Octal to Binary Conversion

Each octal digit is represented by three bits.

Octal/binary conversion.

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

(a) $\overbrace{001}^{1}\overbrace{011}^{3}$

(b) $\overbrace{010}^{2}\overbrace{101}^{5}$

(c) $\overbrace{001}^{1}\overbrace{100}^{4}\overbrace{000}^{0}$

(d) $\overbrace{111}^{7}\overbrace{101}^{5}\overbrace{010}^{2}\overbrace{110}^{6}$

Binary to Octal Conversion

Octal/binary conversion.

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

(a) $\underbrace{110}_{6}\underbrace{101}_{5} = \mathbf{65_8}$

(b) $\underbrace{101}_{5}\underbrace{111}_{7}\underbrace{001}_{1} = \mathbf{571_8}$

(c) $\underbrace{100}_{4}\underbrace{110}_{6}\underbrace{011}_{3}\underbrace{010}_{2} = \mathbf{4632_8}$

(d) $\underbrace{011}_{3}\underbrace{010}_{2}\underbrace{000}_{0}\underbrace{100}_{4} = \mathbf{3204_8}$

Practice

$(32)_{10} = (\qquad\qquad)_2 = (\quad)_8 = (\quad)_{16}$

$(\quad)_{10} = (11111111)_2 = (\quad)_8 = (\quad)_{16}$

$(\quad)_{10} = (\qquad\qquad)_2 = (1774)_8 = (\quad)_{16}$

$(\quad)_{10} = (\qquad\qquad)_2 = (\quad)_8 = (abc)_{16}$

Practice

$(32)_{10} = (100000)_2 = (40)_8 = (20)_{16}$

$(255)_{10} = (11111111)_2 = (377)_8 = (ff)_{16}$

$(1020)_{10} = (1111111100)_2 = (1774)_8 = (3fc)_{16}$

$(2748)_{10} = (101010111100)_2 = (5274)_8 = (abc)_{16}$

ASCII

ASCII is the abbreviation for American Standard Code for Information Interchange.

Pronounced "askee," ASCII is a universally accepted alphanumeric code used in most computers and other electronic equipment.

Most computer keyboards are standardized with the ASCII.

ASCII

American Standard Code for Information Interchange (ASCII).

Control Characters				Graphic Symbols											
Name	Dec	Binary	Hex	Symbol	Dec	Binary	Hex	Symbol	Dec	Binary	Hex	Symbol	Dec	Binary	Hex
NUL	0	0000000	00	space	32	0100000	20	@	64	1000000	40	′	96	1100000	60
SOH	1	0000001	01	!	33	0100001	21	A	65	1000001	41	a	97	1100001	61
STX	2	0000010	02	”	34	0100010	22	B	66	1000010	42	b	98	1100010	62
ETX	3	0000011	03	#	35	0100011	23	C	67	1000011	43	c	99	1100011	63
EOT	4	0000100	04	$	36	0100100	24	D	68	1000100	44	d	100	1100100	64
ENQ	5	0000101	05	%	37	0100101	25	E	69	1000101	45	e	101	1100101	65
ACK	6	0000110	06	&	38	0100110	26	F	70	1000110	46	f	102	1100110	66
BEL	7	0000111	07	’	39	0100111	27	G	71	1000111	47	g	103	1100111	67
BS	8	0001000	08	(	40	0101000	28	H	72	1001000	48	h	104	1101000	68
HT	9	0001001	09	)	41	0101001	29	I	73	1001001	49	i	105	1101001	69
LF	10	0001010	0A	*	42	0101010	2A	J	74	1001010	4A	j	106	1101010	6A
VT	11	0001011	0B	+	43	0101011	2B	K	75	1001011	4B	k	107	1101011	6B
FF	12	0001100	0C	,	44	0101100	2C	L	76	1001100	4C	l	108	1101100	6C
CR	13	0001101	0D	–	45	0101101	2D	M	77	1001101	4D	m	109	1101101	6D
SO	14	0001110	0E	.	46	0101110	2E	N	78	1001110	4E	n	110	1101110	6E
SI	15	0001111	0F	/	47	0101111	2F	O	79	1001111	4F	o	111	1101111	6F
DLE	16	0010000	10	0	48	0110000	30	P	80	1010000	50	p	112	1110000	70
DC1	17	0010001	11	1	49	0110001	31	Q	81	1010001	51	q	113	1110001	71
DC2	18	0010010	12	2	50	0110010	32	R	82	1010010	52	r	114	1110010	72
DC3	19	0010011	13	3	51	0110011	33	S	83	1010011	53	s	115	1110011	73
DC4	20	0010100	14	4	52	0110100	34	T	84	1010100	54	t	116	1110100	74
NAK	21	0010101	15	5	53	0110101	35	U	85	1010101	55	u	117	1110101	75
SYN	22	0010110	16	6	54	0110110	36	V	86	1010110	56	v	118	1110110	76
ETB	23	0010111	17	7	55	0110111	37	W	87	1010111	57	w	119	1110111	77
CAN	24	0011000	18	8	56	0111000	38	X	88	1011000	58	x	120	1111000	78
EM	25	0011001	19	9	57	0111001	39	Y	89	1011001	59	y	121	1111001	79
SUB	26	0011010	1A	:	58	0111010	3A	Z	90	1011010	5A	z	122	1111010	7A
ESC	27	0011011	1B	;	59	0111011	3B	[	91	1011011	5B	{	123	1111011	7B
FS	28	0011100	1C	<	60	0111100	3C	\	92	1011100	5C	|	124	1111100	7C
GS	29	0011101	1D	=	61	0111101	3D	]	93	1011101	5D	}	125	1111101	7D
RS	30	0011110	1E	>	62	0111110	3E	^	94	1011110	5E	~	126	1111110	7E
US	31	0011111	1F	?	63	0111111	3F	_	95	1011111	5F	Del	127	1111111	7F

ASCII

Determine the binary ASCII codes that are entered from the computer's keyboard when the following C language program statement is typed in. Also express each code in hexadecimal.

$$if\ (x > 5)$$

Symbol	Binary	Hexadecimal
i	1101001	69_{16}
f	1100110	66_{16}
Space	0100000	20_{16}
(	0101000	28_{16}
x	1111000	78_{16}
>	0111110	$3E_{16}$
5	0110101	35_{16}
)	0101001	29_{16}

Q&A