Logic Design

Tutorial 3

Amr Keleg

Faculty of Engineering, Ain Shams University

July 30, 2020

Contact: amr_mohamed@live.com

Outline

Outline

Assume we have 4 bits to represent numbers on our system.

- We can represent 16 unsigned values [0, 15].

Assume we have 4 bits to represent numbers on our system.

- We can represent 16 unsigned values [0, 15].
- What if we want to represent negative values as well?

Assume we have 4 bits to represent numbers on our system.

- We can represent 16 unsigned values [0, 15].
- What if we want to represent negative values as well?

Table 1.3
Signed Binary Numbers

Decimal	Signed Magnitude
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
−0	1000
−1	1001
−2	1010
−3	1011
−4	1100
−5	1101
−6	1110
−7	1111
−8	—

Table 1.3
Signed Binary Numbers

Decimal	Signed-2's Complement	Signed Magnitude
+7	0111	0111
+6	0110	0110
+5	0101	0101
+4	0100	0100
+3	0011	0011
+2	0010	0010
+1	0001	0001
+0	0000	0000
−0	—	1000
−1	1111	1001
−2	1110	1010
−3	1101	1011
−4	1100	1100
−5	1011	1101
−6	1010	1110
−7	1001	1111
−8	1000	—

How to add two numbers in signed two's complement representation?

Example:

$(+7) + (+7) = ??$

How to add two numbers in signed two's complement representation?

Example:

$(+7) + (+7) = ??$

$(0111) + (0111) =$

How to add two numbers in signed two's complement representation?

Example:

$(+7) + (+7) = ??$

$(0111) + (0111) = (1110)$ which is

How to add two numbers in signed two's complement representation?

Example:

$(+7) + (+7) = ??$

$(0111) + (0111) = (1110)$ which is -2 !!

How to add two numbers in signed two's complement representation?

Example:

$(+7) + (+7) = ??$

$(0111) + (0111) = (1110)$ which is -2 !!

Therefore we need to have another bit to handle this overflow:

How to add two numbers in signed two's complement representation?
Example:
$(+7) + (+7) = ??$
$(0111) + (0111) = (1110)$ which is -2 !!
Therefore we need to have another bit to handle this overflow:$(\mathbf{0}0111) + (\mathbf{0}0111) = \mathbf{0}1110$

How to add two numbers in signed two's complement representation?
Example:
$(+7) + (+7) = ??$
$(0111) + (0111) = (1110)$ which is -2 !!
Therefore we need to have another bit to handle this overflow:$(\mathbf{0}0111) + (\mathbf{0}0111) = \mathbf{0}1110$

- The result is positive since the sign bit is equal to zero
- The value is $8 + 4 + 2 = 14$ (Makes sense now)

What about (-7) + (-7)?

What about (-7) + (-7)?

+7 = (00111)
-7 = (11001)

What about (-7) + (-7)?

+7 = (00111)
-7 = (11001)

(11001) + (11001) = 1**1**0010 (Ignore the overflow).

The result is 10010.

What about (-7) + (-7)?

+7 = (00111)
-7 = (11001)

(11001) + (11001) = **1**10010 (Ignore the overflow).

The result is 10010.

- The result is negative since the sign bit is equal to one
- Its two's complement is 01110
- The value is - (8 + 4 + 2) = -14 (Makes sense now)

Outline

Q17) Convert decimal +49 and +29 to binary, using the signed 2's complement representation and enough digits to accommodate the numbers. Then perform the binary equivalent of (+29) + (-49), (-29) + (+49), and (-29) + (-49). Convert the answers back to decimal and verify that they are correct.

Q17) Convert decimal +49 and +29 to binary, using the signed 2's complement representation and enough digits to accommodate the numbers. Then perform the binary equivalent of (+29) + (-49), (-29) + (+49), and (-29) + (-49). Convert the answers back to decimal and verify that they are correct.

- $49_{10} = 110001_2$
- $29_{10} = \mathbf{0}11101_2$

Q17) Convert decimal +49 and +29 to binary, using the signed 2's complement representation and enough digits to accommodate the numbers. Then perform the binary equivalent of (+29) + (-49), (-29) + (+49), and (-29) + (-49). Convert the answers back to decimal and verify that they are correct.

- $49_{10} = 110001_2$
- $29_{10} = \mathbf{0}11101_2$
- Add two bits, one for sign and one for overflow:
- $49_{10} = \mathbf{00}110001_2$
- $29_{10} = \mathbf{00}011101_2$

Q17) Convert decimal +49 and +29 to binary, using the signed 2's complement representation and enough digits to accommodate the numbers. Then perform the binary equivalent of (+29) + (-49), (-29) + (+49), and (-29) + (-49). Convert the answers back to decimal and verify that they are correct.

- $49_{10} = 110001_2$
- $29_{10} = \mathbf{0}11101_2$
- Add two bits, one for sign and one for overflow:
- $49_{10} = \mathbf{00}110001_2$
- $29_{10} = \mathbf{00}011101_2$
- $-49_{10} = \mathbf{11}001111_2$
- $-29_{10} = \mathbf{11}100011_2$

A] $(+29) + (\text{-}49) = 0001{,}1101 + 1100{,}1111$

A] (+29) + (-49) = 0001,1101 + 1100,1111 = 0,1110,1100

A] $(+29) + (\text{-}49) = 0001{,}1101 + 1100{,}1111 = 0{,}1110{,}1100$

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **1**110,1100.

A] (+29) + (-49) = 0001,1101 + 1100,1111 = 0,1110,1100

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **1**110,1100.
- This is a negative value (Sign bit = 1).

A] (+29) + (-49) = 0001,1101 + 1100,1111 = 0,1110,1100

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **1**110,1100.
- This is a negative value (Sign bit = 1).
- Its magnitude is 0001,0100 which is 16 + 4 = 20

A] (+29) + (-49) = 0001,1101 + 1100,1111 = 0,1110,1100

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **1**110,1100.
- This is a negative value (Sign bit = 1).
- Its magnitude is 0001,0100 which is 16 + 4 = 20
- The result is $(-20)_{10}$

B] (-29) + (49) = 1110,0011 + 0011,0001

B] (-29) + (49) = 1110,0011 + 0011,0001 = 1,0001,0100

B] (-29) + (49) = 1110,0011 + 0011,0001 = 1,0001,0100

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **0**001,0100.

B] (-29) + (49) = 1110,0011 + 0011,0001 = 1,0001,0100

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **0**001,0100.
- This is a positive value (Sign bit = 0).

B] (-29) + (49) = 1110,0011 + 0011,0001 = 1,0001,0100

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **0**001,0100.
- This is a positive value (Sign bit = 0).
- Its magnitude is 0001,0100 which is 16 + 4 = 20

B] (-29) + (49) = 1110,0011 + 0011,0001 = 1,0001,0100

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **0**001,0100.
- This is a positive value (Sign bit = 0).
- Its magnitude is 0001,0100 which is 16 + 4 = 20
- The result is $(+20)_{10}$

C] $(-29) + (-49) = 1110{,}0011 + 1100{,}1111$

C] (-29) + (-49) = 1110,0011 + 1100,1111 = 1,1011,0010

C] (-29) + (-49) = 1110,0011 + 1100,1111 = 1,1011,0010

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **1**011,0010.

C] (-29) + (-49) = 1110,0011 + 1100,1111 = 1,1011,0010

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **1**011,0010.
- This is a negative value (Sign bit = 1).

C] (-29) + (-49) = 1110,0011 + 1100,1111 = 1,1011,0010

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **1**011,0010.
- This is a negative value (Sign bit = 1).
- Its magnitude is 0100,1110 which is 64 + 8 + 4 + 2 = 78

C] (-29) + (-49) = 1110,0011 + 1100,1111 = 1,1011,0010

- ALWAYS IGNORE ANY BITS OTHER THAN THE bits that we need (8 bits in this case).
- Result is **1**011,0010.
- This is a negative value (Sign bit = 1).
- Its magnitude is 0100,1110 which is 64 + 8 + 4 + 2 = 78
- The result is $(-78)_{10}$

Outline

Outline

Q18) Represent the decimal number 6,248 in:

- a) BCD

[1] https://en.wikipedia.org/wiki/Aiken_code

Q18) Represent the decimal number 6,248 in:

- a) BCD: Represent each decimal **digit** as a 4-bit binary number.
  $6{,}248_{10} = 0110,\ 0010,\ 0100,\ 1000_{\mathrm{BCD}}$
- b) excess3 code

[1] https://en.wikipedia.org/wiki/Aiken_code

Q18) Represent the decimal number 6,248 in:

- a) BCD: Represent each decimal **digit** as a 4-bit binary number.
  $6{,}248_{10} = 0110, 0010, 0100, 1000_{\text{BCD}}$
- b) excess3 code: Similar to BCD but add three to each digit before converting to binary
  $6{,}248_{10} = 1001, 0101, 0111, 1011_{\text{Excess-3 code}}$
- c) 2421 code[1]

[1] https://en.wikipedia.org/wiki/Aiken_code

Q18) Represent the decimal number 6,248 in:

- a) BCD: Represent each decimal **digit** as a 4-bit binary number.
  $6{,}248_{10} = 0110,\ 0010,\ 0100,\ 1000_{\text{BCD}}$
- b) excess3 code: Similar to BCD but add three to each digit before converting to binary
  $6{,}248_{10} = 1001,\ 0101,\ 0111,\ 1011_{\text{Excess-3 code}}$
- c) 2421 code[1]: Instead of considering the weights of the 4 bits to be 8, 4, 2, 1 make them 2, 4, 2, 1
  $6{,}248_{10} = 1100,\ 0010,\ 0100,\ 1110_{\text{2421 code}}$
- d) 6311 code

[1] https://en.wikipedia.org/wiki/Aiken_code

Q18) Represent the decimal number 6,248 in:

- a) BCD: Represent each decimal **digit** as a 4-bit binary number.
  $6{,}248_{10} = 0110,\ 0010,\ 0100,\ 1000_{\mathrm{BCD}}$
- b) excess3 code: Similar to BCD but add three to each digit before converting to binary
  $6{,}248_{10} = 1001,\ 0101,\ 0111,\ 1011_{\text{Excess-3 code}}$
- c) 2421 code[1]: Instead of considering the weights of the 4 bits to be 8, 4, 2, 1 make them 2, 4, 2, 1
  $6{,}248_{10} = 1100,\ 0010,\ 0100,\ 1110_{\text{2421 code}}$
- d) 6311 code: Instead of considering the weights of the 4 bits to be 8, 4, 2, 1 make them 6, 3, 1, 1
  $6{,}248_{10} = 1000,\ 0011,\ 0101,\ 1011_{\text{6311 code}}$

[1] `https://en.wikipedia.org/wiki/Aiken_code`

Outline

Q1) Convert the following numbers into BCD and hence carry out the BCD addition:

- b) $(398)_{10} + (198)_{10}$

Q1) Convert the following numbers into BCD and hence carry out the BCD addition:

- b) $(398)_{10} + (198)_{10}$
- $(398)_{10} = (0011, 1001, 1000)_{BCD}$
- $(198)_{10} = (0001, 1001, 1000)_{BCD}$

$$\begin{array}{rrr} 0011 & 1001 & 1000 \\ & & + \\ 0001 & 1001 & 1000 \\ \hline \end{array}$$

$$
\begin{array}{r r r}
0011 & 1001 & 1000 \\
 & & + \\
0001 & 1001 & 1000 \\
\hline
 & & 10000
\end{array}
$$

(result is more than 9, Add 6 to the value)

$$
\begin{array}{rrr}
0011 & 1001 & 1000 \\
 & & + \\
0001 & 1001 & 1000 \\
\hline
 & & 10000
\end{array}
$$

(result is more than 9, Add 6 to the value)

$$
\begin{array}{r}
+ \\
0110 \\
\hline
\end{array}
$$

0011	1001	1000
		+
0001	1001	1000
		10000

(result is more than 9, Add 6 to the value)

	+
	0110
1	0110

$$
\begin{array}{rrr}
 & 1 & \\
0011 & 1001 & 1000 \\
 & & + \\
0001 & 1001 & 1000 \\
\hline
\end{array}
$$

$$
\begin{array}{rrr}
 & 1 & \\
0011 & 1001 & 1000 \\
 & & + \\
0001 & 1001 & 1000 \\
\hline
 & 10011 & 0110
\end{array}
$$

(result is more than 9, Add 6 to the value)

$$\begin{array}{rrr}
 & 1 & \\
0011 & 1001 & 1000 \\
 & & + \\
0001 & 1001 & 1000 \\
\hline
 & 10011 & 0110
\end{array}$$

(result is more than 9, Add 6 to the value)

$$\begin{array}{rrr}
 & & + \\
 & 0110 & \\
\hline
\end{array}$$

	1	
0011	1001	1000
		+
0001	1001	1000
	10011	0110

(result is more than 9, Add 6 to the value)

		+
	0110	
1	1001	0110

$$
\begin{array}{rrr}
1 & & \\
0011 & 1001 & 1000 \\
 & & + \\
0001 & 1001 & 1000 \\
\hline
\end{array}
$$

1		
0011	1001	1000
		+
0001	1001	1000
0101	1001	0110

(result is less than 9, Do thing)

1		
0011	1001	1000
		+
0001	1001	1000
0101	1001	0110

(result is less than 9, Do thing)

0101	1001	0110

Outline

Outline

How to perform the following **UNSIGNED binary** multiplications:

- $10011_2 * 1_2 =$

How to perform the following **UNSIGNED binary** multiplications:

- $10011_2 * 1_2 = 10011_2$

How to perform the following **UNSIGNED binary** multiplications:

- $10011_2 * 1_2 = 10011_2$
- $10011_2 * 10_2 =$

How to perform the following **UNSIGNED binary** multiplications:

- $10011_2 * 1_2 = 10011_2$
- $10011_2 * 10_2 = 10011\mathbf{0}_2$
  Explanation:
  $10011_2 = (2^4) + (2^1) + (2^0)$
  $10_2 = (2^1)$

How to perform the following **UNSIGNED binary** multiplications:

- $10011_2 * 1_2 = 10011_2$
- $10011_2 * 10_2 = 10011\mathbf{0}_2$
  Explanation:
  $10011_2 = (2^4) + (2^1) + (2^0)$
  $10_2 = (2^1)$
  $10011_2 * 10_2 = ((2^4) + (2^1) + (2^0)) * (2^1) = 2^5 + 2^2 + 2^1$
- $10011_2 * 100_2 =$

How to perform the following **UNSIGNED binary** multiplications:

- $10011_2 * 1_2 = 10011_2$
- $10011_2 * 10_2 = 10011\mathbf{0}_2$
  Explanation:
  $10011_2 = (2^4) + (2^1) + (2^0)$
  $10_2 = (2^1)$
  $10011_2 * 10_2 = ((2^4) + (2^1) + (2^0)) * (2^1) = 2^5 + 2^2 + 2^1$
- $10011_2 * 100_2 = 10011\mathbf{00}_2$

- $10011_2 * 1_2 = 10011_2$
- $10011_2 * 10_2 = 100110_2$
- $10011_2 * 100_2 = 1001100_2$

- $10011_2 * 1_2 = 10011_2$
- $10011_2 * 10_2 = 100110_2$
- $10011_2 * 100_2 = 1001100_2$
- $10011_2 * 111_2 =$

- $10011_2 * 1_2 = 10011_2$
- $10011_2 * 10_2 = 100110_2$
- $10011_2 * 100_2 = 1001100_2$
- $10011_2 * 111_2 =$
  $10011_2 * 1_2 + 10011_2 * 10_2 + 10011_2 * 100_2$

- $10011_2 * 1_2 = 10011_2$
- $10011_2 * 10_2 = 100110_2$
- $10011_2 * 100_2 = 1001100_2$
- $10011_2 * 111_2 =$
  $10011_2 * 1_2 + 10011_2 * 10_2 + 10011_2 * 100_2 = 10011 + 100110 + 1001100 = 1000,0101_2$

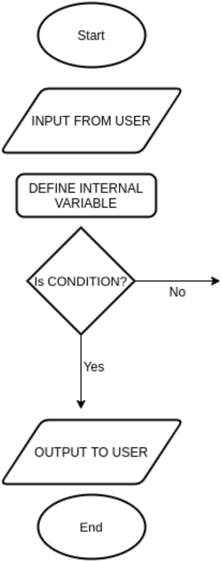
- $10011_2 * 1_2 = 10011_2$
- $10011_2 * 10_2 = 100110_2$
- $10011_2 * 100_2 = 1001100_2$
- $10011_2 * 111_2 = 10011_2 * 1_2 + 10011_2 * 10_2 + 10011_2 * 100_2 = 10011 + 100110 + 1001100 = 1000,0101_2$
- Note: Convert the values to decimal and check that the result is correct.

- $10011_2 * 1_2 = 10011_2$
- $10011_2 * 10_2 = 100110_2$
- $10011_2 * 100_2 = 1001100_2$
- $10011_2 * 111_2 =$
  $10011_2 * 1_2 + 10011_2 * 10_2 + 10011_2 * 100_2 = 10011 + 100110 + 1001100 = 1000,0101_2$
- Note: Convert the values to decimal and check that the result is correct.

Thus, binary multiplication depends on two operations:
Shift-left (multiplication by 2) and Binary-addition.

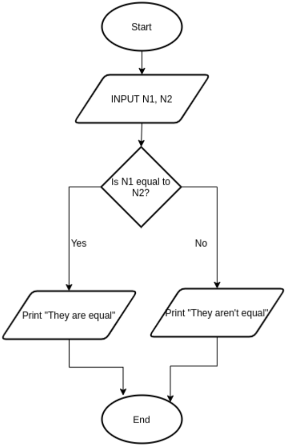
Outline

The main blocks are:



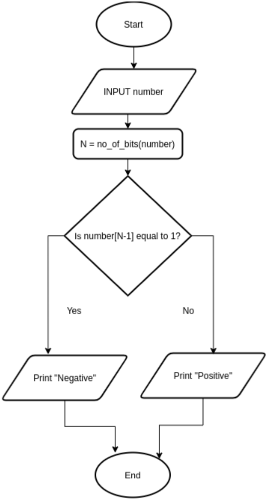
Example 1: Allow the user to input two values and check whether they are equal to not.

Example 1: Allow the user to input two values and check whether they are equal to not.



Example 2: Allow the user to input a **Signed 2's complement binary number** and check whether it's positive or negative.

Example 2: Allow the user to input a **Signed 2's complement binary number** and check whether it's positive or negative.



Outline

Q4) Write an algorithm for unsigned binary numbers multiplication. You can draw a flow chart.

Q4) Write an algorithm for unsigned binary numbers multiplication. You can draw a flow chart.

Note: For binary non-floating numbers:

- A shift left operation adds a zero to the right of the number (e.g: 101 becomes 1010).

Q4) Write an algorithm for unsigned binary numbers multiplication. You can draw a flow chart.

Note: For binary non-floating numbers:

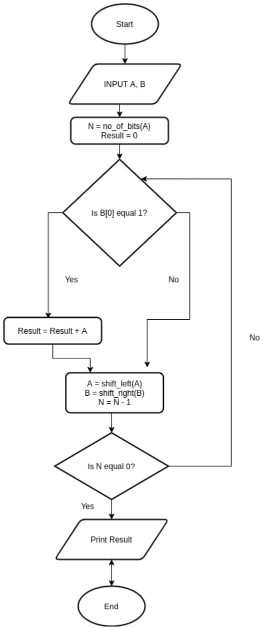
- A shift left operation adds a zero to the right of the number (e.g: 101 becomes 1010).
- A shift right operation drops the least significant bit (e.g: 101 becomes 10).

Q4) Write an algorithm for unsigned binary numbers multiplication. You can draw a flow chart.

Note: For binary non-floating numbers:

- A shift left operation adds a zero to the right of the number (e.g: 101 becomes 1010).
- A shift right operation drops the least significant bit (e.g: 101 becomes 10).

Note: Assume that we want to compute (A * B) where A, B are two binary numbers of the equal length.


Start
INPUT A, B
N = no_of_bits(A)
Result = 0
Is B[0] equal 1?
Yes
No
Result = Result + A
No
A = shift_left(A)
B = shift_right(B)
N = N - 1
Is N equal 0?
Yes
Print Result
End

Outline

Outline

Table 1.6
Gray Code

Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

https://en.wikipedia.org/wiki/Gray_code