What is the 16-bit hexadecimal representation of the binary number 16’b0100111001101011?

* 4E6B
* 16’h4e6b

What is the 16-bit binary representation of the hexadecimal number 16’he6C7?

* 16'b1110011011000111

What is the 16-bit two’s complement binary representation of the decimal number 16’d17364?

* 16'b0100001111010100

What is the 16-bit two’s complement binary representation of the decimal number -16’d9702? (note that this is a negative number)

* 16'b1101101000011010

What is the decimal number represented by the two’s complement hexadecimal number 16’h53fc?

* 0101001111111100
  + 16’d21500

What is the decimal number represented by the two’s complement hexademical number 16’9fb0?

* 1001111110110000
  + -16’d24656

Use two’s complement sign extension to extend the two’s complement number 8’h38 to a 16-bit two’s complement hexadecimal number.

* 00111000
  + 56
    - 0000000000111000
      * 16’h0038

Use two’s complement sign extension to extend the two’s complement number 8’hf4 to a 16-bit two’s complement hexadecimal number.

* 11110100
  + -12
    - 1111111111110100
      * 16’hfff4

Determine the two’s complement addition of each of the following:

* 16’h2f37 + 16’h3ef7
  + 0010111100110111
  + 0011111011110111
    - 110111000101110
      * 16’h6e2e
* 16’h9742 + 16’h4fe3
  + 1001011101000010
  + 0100111111100011
    - 1110011100100101
      * 16’he725
* 16’hfb1a + 16’hb16c
  + 1111101100011010
  + 1011000101101100
    - 11010110010000110
    - overflow

Determine the two’s complement subtraction of each of the following:

* 16’h3ef7 - 16’h2f37
  + 0011111011110111
  + 0010111100110111
    - 0000111111000000
    - 16’h0fc0
* 13.
* 16’h9742 - 16’h4fe3
  + 1001011101000010
  + 0100111111100011
    - 0100011101011111
    - 16’h475f
    - **overflow**
* 16’hfb1a - 16’hb16c
  + 1111101100011010
  + 1011000101101100
    - 100100110101110
    - **16'hac86**

*Two’s Complement Operation* Two’s complement numbers can be negated by determining the “two’s complement” of the number. Determine the “two’s complement” of each of the following numbers:

* 16’h3ef7
  + 0011111011110111
  + 1100000100001000
  + 1100000100001001
    - 16’hc109
* 16’h9742
  + 1001011101000010
  + 1001011101000010
  + 1001011101000010
  + 0110100010111101
  + 0110100010111110
    - * 16’h68be
* 16’hfb1a
  + 1111101100011010
    - 0000010011100101
      * 0000010011100110
      * 16’h0e46

*Logical Operations*

Demonstrate your understanding of the binary logical operations of AND (&), OR (|), and XOR (^) by determining the result of each logical operation. Provide all of your answers as SystemVerilog constants.

* 16’h3ef7 & 16’h2f37
  + 0011111011110111
  + 0010111100110111
    - 0010111000110111
    - 16’h2e37
* 16’h9742 | 16’h4fe3
  + 1001011101000010
  + 0100111111100011
    - 1101111111100011
    - 16’hdfe3
* 16’hfb1a ^ 16’hb16c
  + 1111101100011010
  + 1011000101101100
    - 0100101001110110
    - 16’h4a76

*Shift Operations*

Demonstrate your understanding of the shift operations by determining the result of each shift operation listed below. Provide all of your answers as SystemVerilog constants.

* 16’h3ef7 << 2
  + 0011111011110111
    - 1111101111011100
    - 16’hfbdc
    - **16'hfbdc**
* 16’h9742 >> 5
  + 1001011101000010
    - 10010111010.0001
    - 000001001011101
    - 16’h025d
    - **16'h04ba**
* 16’h9742 >>> 3
  + 1001011101000010
    - 1111001011101000
    - 16’hf2e8
    - **16'hf2e8**

Assuming ‘op1’=32’hf3212f37 and ‘op2’=32’h621c3ee7, determine the value of ALU result for the given value of ‘alu\_op’:

* 4’b0000
  + 32'h62002e27
* 4’b0001
  + 32'hf33d3ff7
* 4’b0010
  + 32'h553d6e1e
* 4’b0011
  + 32'h553d6e1e
* 4’b0110
  + 32'h9104f050
* 4’b0111
  + 32'h00000001
* 4’b1000
  + 32'h01e6425e
* 4’b1001
  + 32'h90979b80
* 4’b1010
  + **32'hffe6425e**
* 4’b1101
  + 32'h913d11d0

