**Q1. Convert the hexadecimal number 973D4 to a number with base 15**. **(2 points)**

973D4 / F = A152 R: 6

A152 / F = AC1 R: 3

AC1 / F = B7 R: 8

B7 / F = C R: 3

C / F = 0 R: C

Therefore:

**Q2. Floating point numbers. (2 points)**

Convert the following floating-point numbers to hexadecimal number in IEEE single-precision format. Please give the result as eight

hexadecimal digits.

**1. -69/32 (-69 divide by 32)**

Insert 1 in the first bit to represent the negative

69/32 = 2.15625

Convert whole number part:

Convert decimal part:

0.15625 \* 2 = 0.3125

0.3125 \* 2 = 0.625

0.625 \* 2 = 1.25

0.25 \* 2 = 0.5

0.5 \* 2 = 1.0 (done)

Conversion:

Binary result:

Binary representation (32bit):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1100 |0000 |0000 |1010 |0000 |0000 |0000 |0000

12|0|0|10|0|0|0|0

Final Hex representation:

C00A0000

**2. 13.625**

Convert whole number part:

Convert decimal part:

0.625 \* 2 = 1.25

0.25 \* 2 = 0.5

0.5 \* 2 = 1.0 (done)

Conversion:

Binary result:

Binary representation (32bit):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0100 |0001 |0101 |1010 |0000 |0000 |0000 |0000

4|1|5|10|0|0|0|0

Final Hex representation:

415A0000

Convert the following hexadecimal numbers in IEEE single-precision format to floating-point numbers:

**1. 42E48000**

4|2|14|4|8|0|0|0

Binary representation (32bit):

0100 |0010 |1110 |0100 |1000 |0000 |0000 |0000

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Whole part:

Decimal part:

Final Decimal representation:  
+114.25

**2. C6F00040**

12|6|15|0|0|0|4|0

Binary representation (32bit):

1100 |0110 |1111 |0000 |0000 |0000 |0100 |0000

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Whole part:

Decimal part:

Final Decimal representation:  
-30720.125

**Q3. Error Finding (4 points)**

Each of the following 68K assembly language instructions will cause an assembler error. Examine each instruction and explain why the

assembler would flag it as an error.

1) MOVE.B $A000, A3

Data passed $A000 is larger than a byte, you cannot MOVE.B a word, you should use MOVE.W

2) ADD.B #$1000, D2

Data passed $1000 is larger than a byte, you cannot ADD.B a word, you should use ADD.W

3) MOVEA.W $1234, D0

The MOVEA.W cannot move a number in immediate access. MOVEI.W should be used

Also, I think is the user wants to pass the hex number 1234, they should add the # sign i.e. #$1234

4) ANDI.B #23, #$100

Performing an ANDI operation needs an effective address to store the result. #$100 is not a destination, but an immediate access number. The # sign should be deleted to make the ANDI store the result at memory location $100

**Q4. Create a source file and analyze the results. (4 points)**

**(separate files)**

**Q5. Two’s complement (7 points)**

Assume that we are using a **16-bit system**. Represent a negative integer with two’s complement format.

1. (2 pts) Convert the decimal numbers -102 and -87 into hexadecimal number.

-102

Divide by 2 until you reach 1,

1 for odd, 0 or even: 0 1 1 0 0 1 1

Reverse bits for answer: 0110 0110

One compliment: 1001 1001

Add 1: +1

1001 1010

Answer in hex: 9A

-87

Divide by 2 until you reach 1,

1 for odd, 0 or even: 1 1 1 0 1 0 1

Reverse bits for answer: 0101 0111

One compliment: 1010 1000

Add 1: +1

1010 1001

Answer in hex: A9

2. (1 pt) Add two numbers of the previous question as hexadecimal, and state

1 1 1 1 1

9A 1001 1010

+A9 +1010 1001  
 143 0100 0011

3. whether the sign bit of the result is 1, and

The sign in 0.

4. whether an overflow occurred.

and overflow occurred because the 16bit system was over capacity for these two values.

5. (4 pts) Write a program in assembly language to add the two numbers (-102 and -87). Inputs should be in decimal format.

Store the result as hexadecimal numbers at address $6000. Print out the result in command output window in ***decimal*** format.

Figure 1: Output window

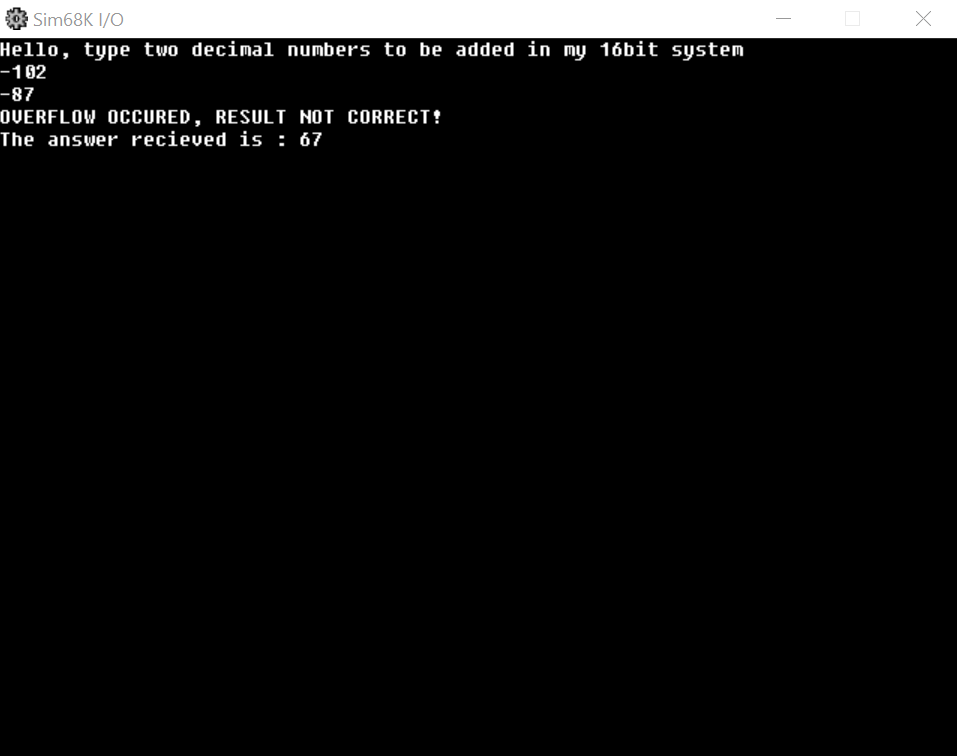


Figure 2: Memory

