Jeremy DeHaan

CSS422

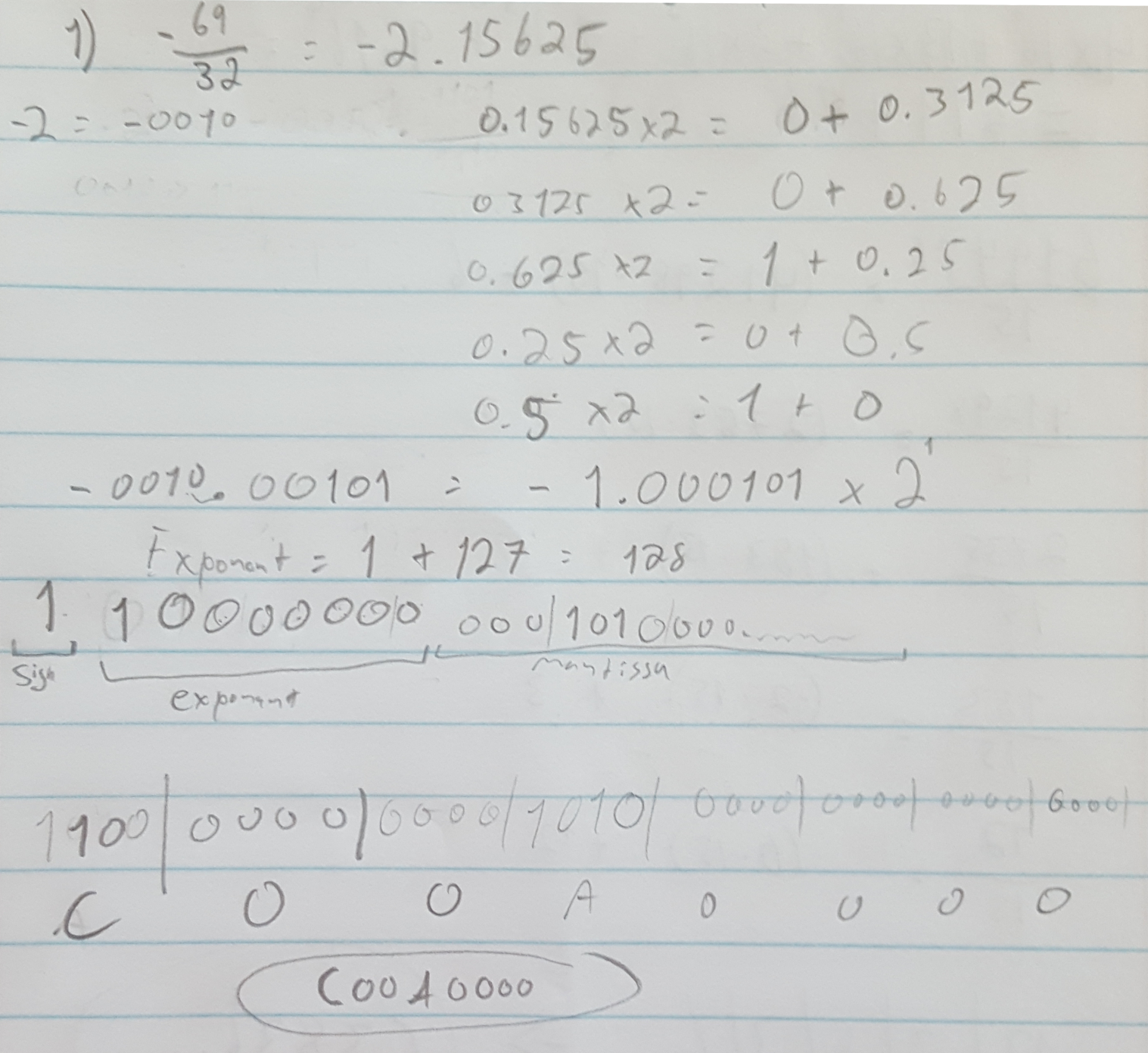
Homework 1

# Q1. Convert the hexadecimal number 973D4 to a number with base 15.

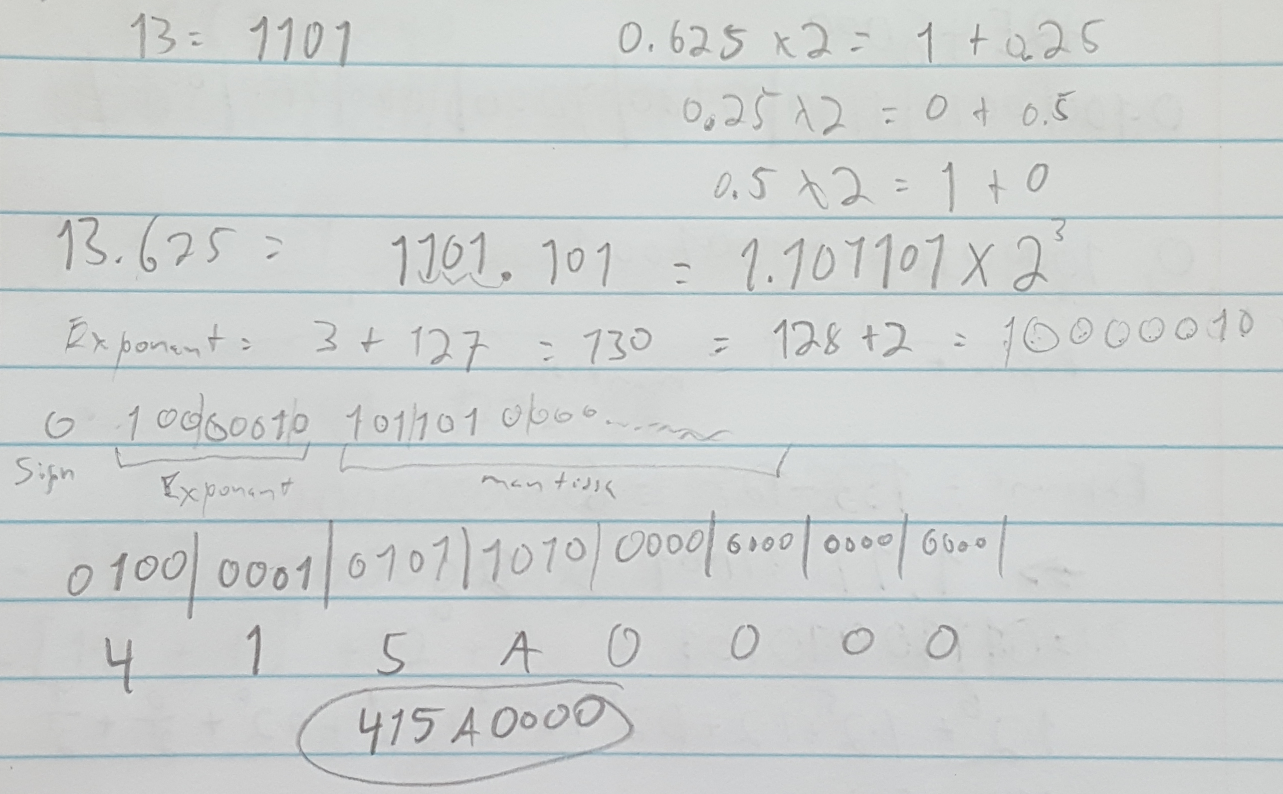


# Q2. Floating point numbers.

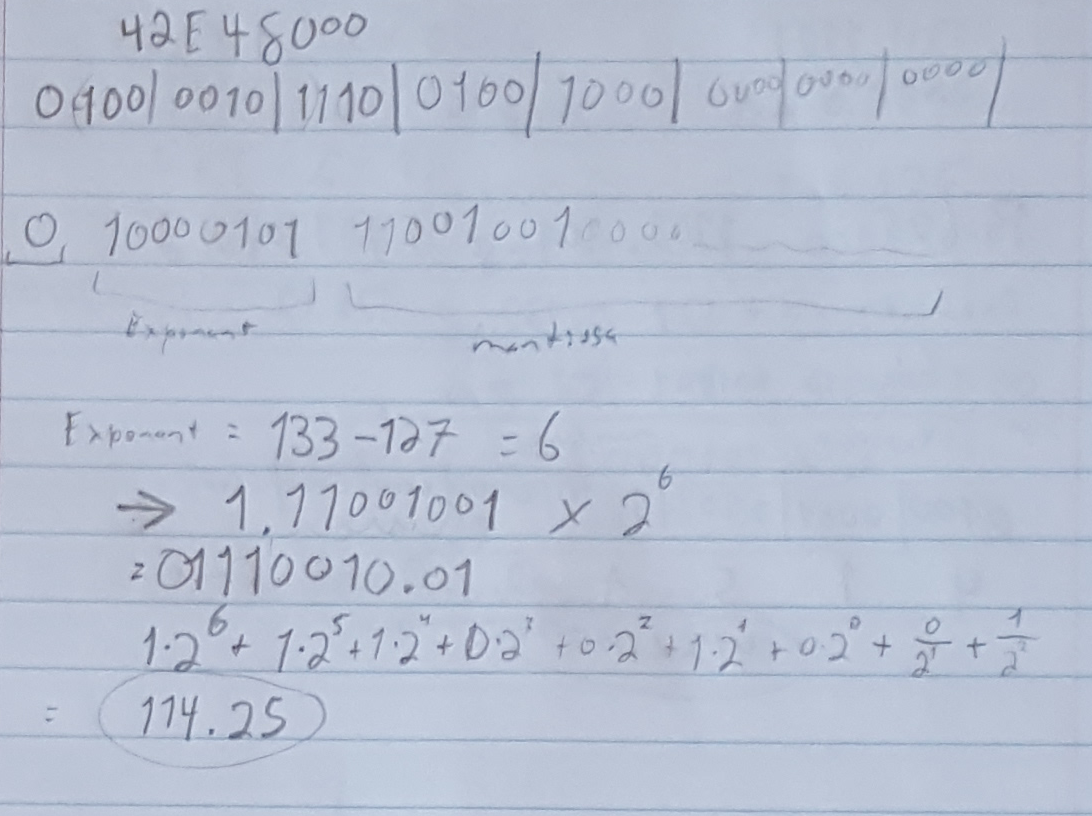
* Convert the following floating-point numbers to hexadecimal number in IEEE single-precision format. Please give the result as eight hexadecimal digits.
  1. -69/32 (-69 divide by 32)



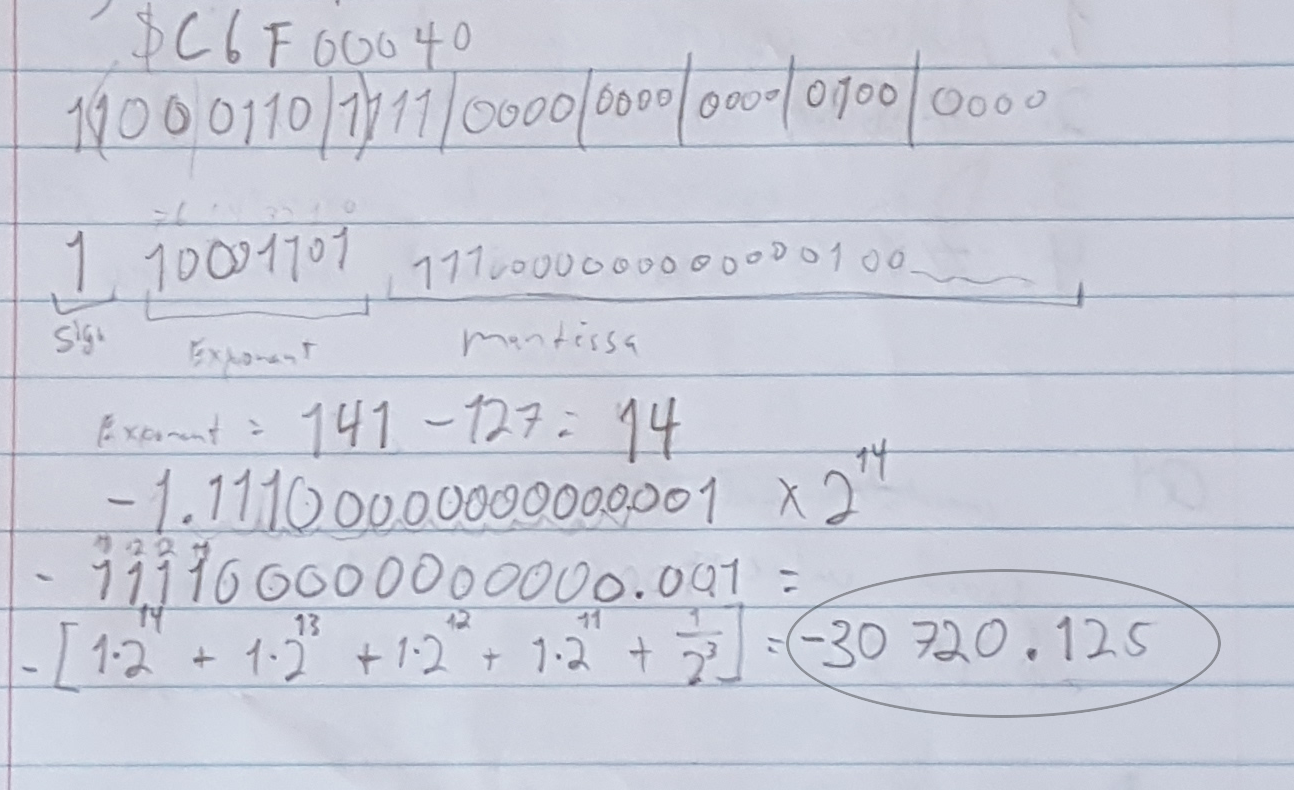
* 1. 13.625



* Convert the following hexadecimal numbers in IEEE single-precision format to floating-point numbers
  1. 42E48000



* 1. C6F00040



# Q3. Error Finding

1. MOVE.B $A000, A3

It is invalid for an address register to be the destination of a MOVE operation.

1. ADD.B #$1000, D2

The #$1000 literal is a total of 2 bytes, and cannot be used in an ADD.B operation (it only works on bytes).

1. MOVEA.W $1234, D0

It is invalid for a data register to be the destination of a MOVEA operation.

1. ANDI.B #23, #$100

The #$100 literal is not a valid effective address.

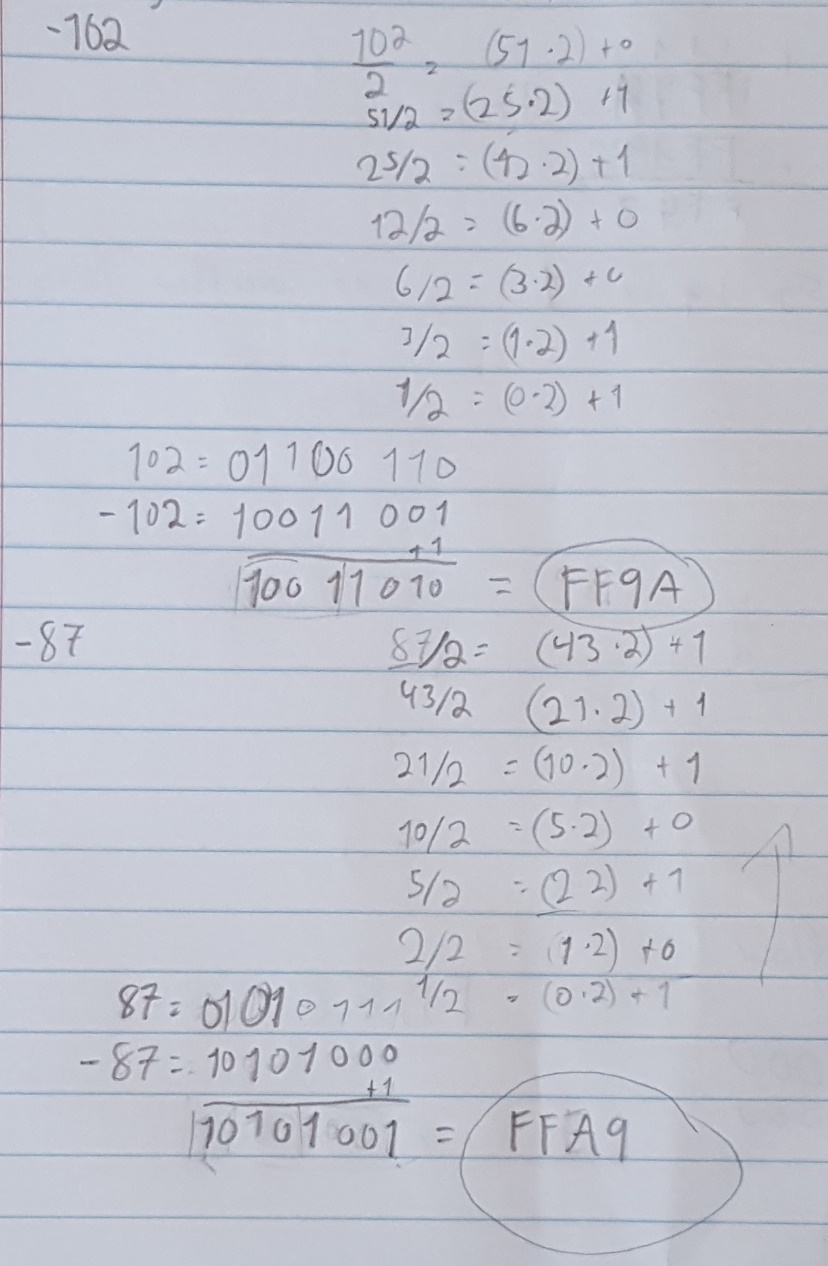
# Q4. Create a source file and analyze the results.

QUESTION: What is the WORD VALUE (not byte, or longword) of the data in memory location $4000, when the program is just about to loop back to the place where "start" is labeled? Please describe how you got the answer as well. (For example, you can describe how you analyzed the code segments, or how you traced the code segments with debug tools)

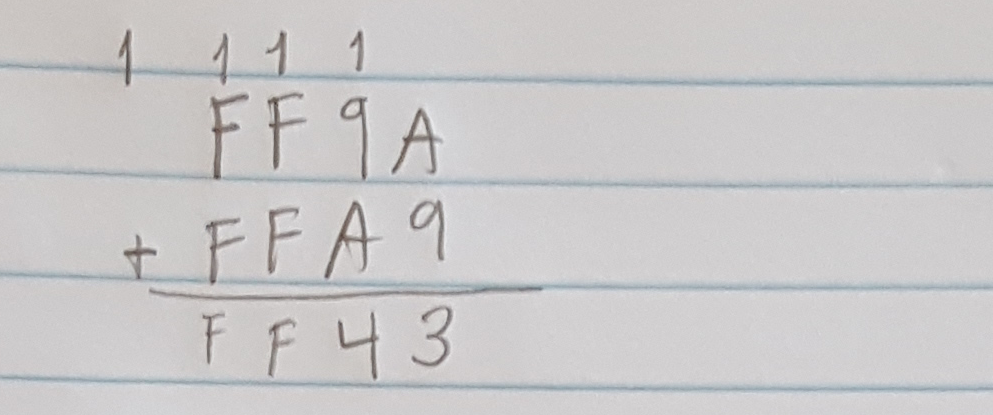
The word value of the data at the memory location $4000 is $4515, or 17685(base 10). I found this value by using the breakpoints in the debugging view and the memory view window. I placed a breakpoint on the line with the “JMP start” instruction. When I ran the simulator, it broke at this instruction, and then I viewed what was at address $4000 at this time.

# Q5. Two’s complement.

1. Convert the decimal numbers -102 and -87 into hexadecimal number.



1. Add two numbers of the previous question as hexadecimal, and state
   1. whether the sign bit of the result is 1
   2. whether an overflow occurred.



The sign bit of the result is 1 since the most significant bit of $F is 1. While there is a carry, there is no overflow since the result of the addition is still within the range of a 16bit signed integer.

1. 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. (Hint: use the trap function task #3). If an error (overflow) happens, you should also print out the error (overflow) message as well.

Copy and paste the screen shot of the output(command) window into your submission file. Do not submit an individual file for the image!



Copy and paste the screen shot of your memory system at address $6000 into your submission file. Do not submit an individual file for the image!

