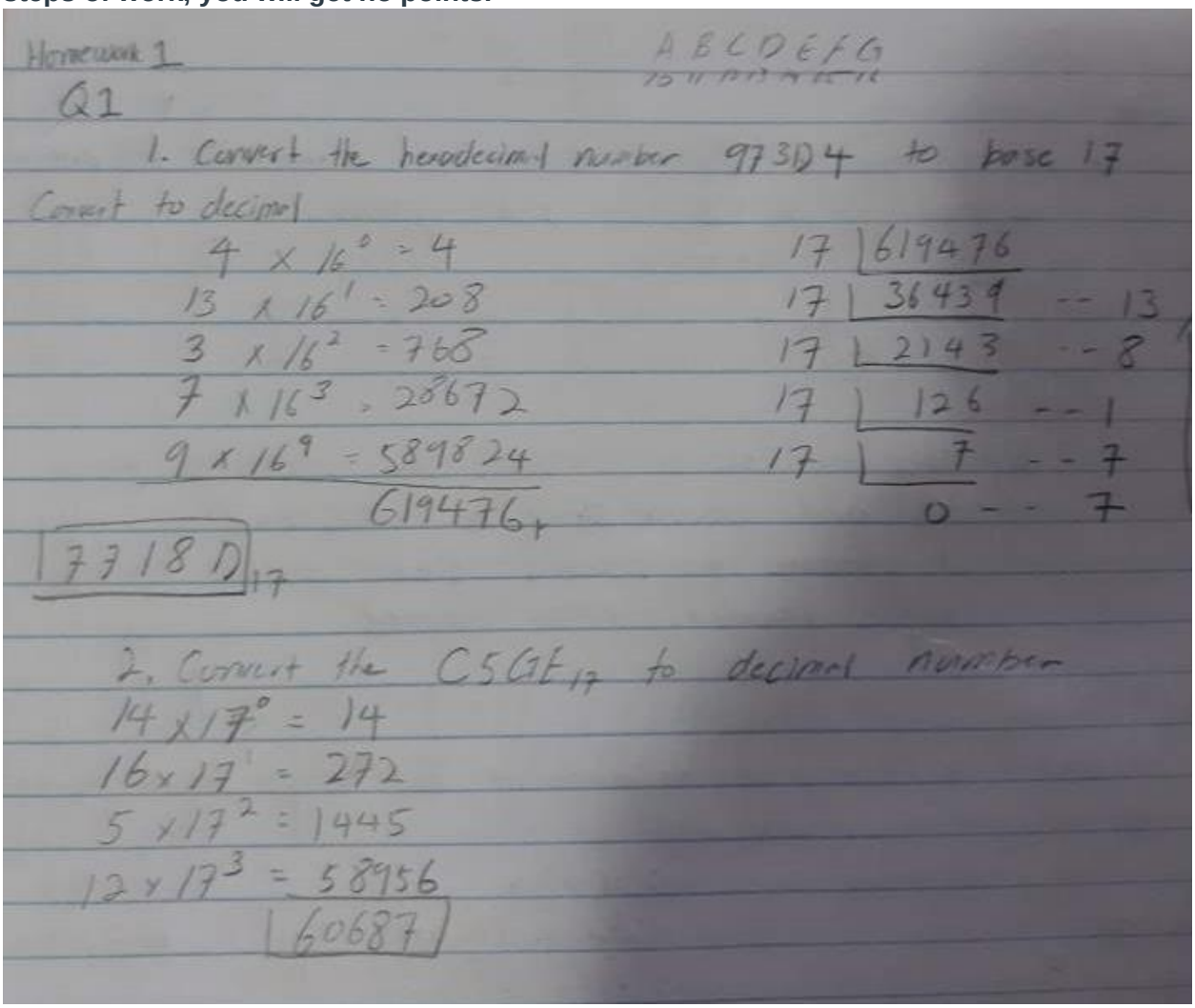
Homework 1

CSS 422

Jun Hyun Park

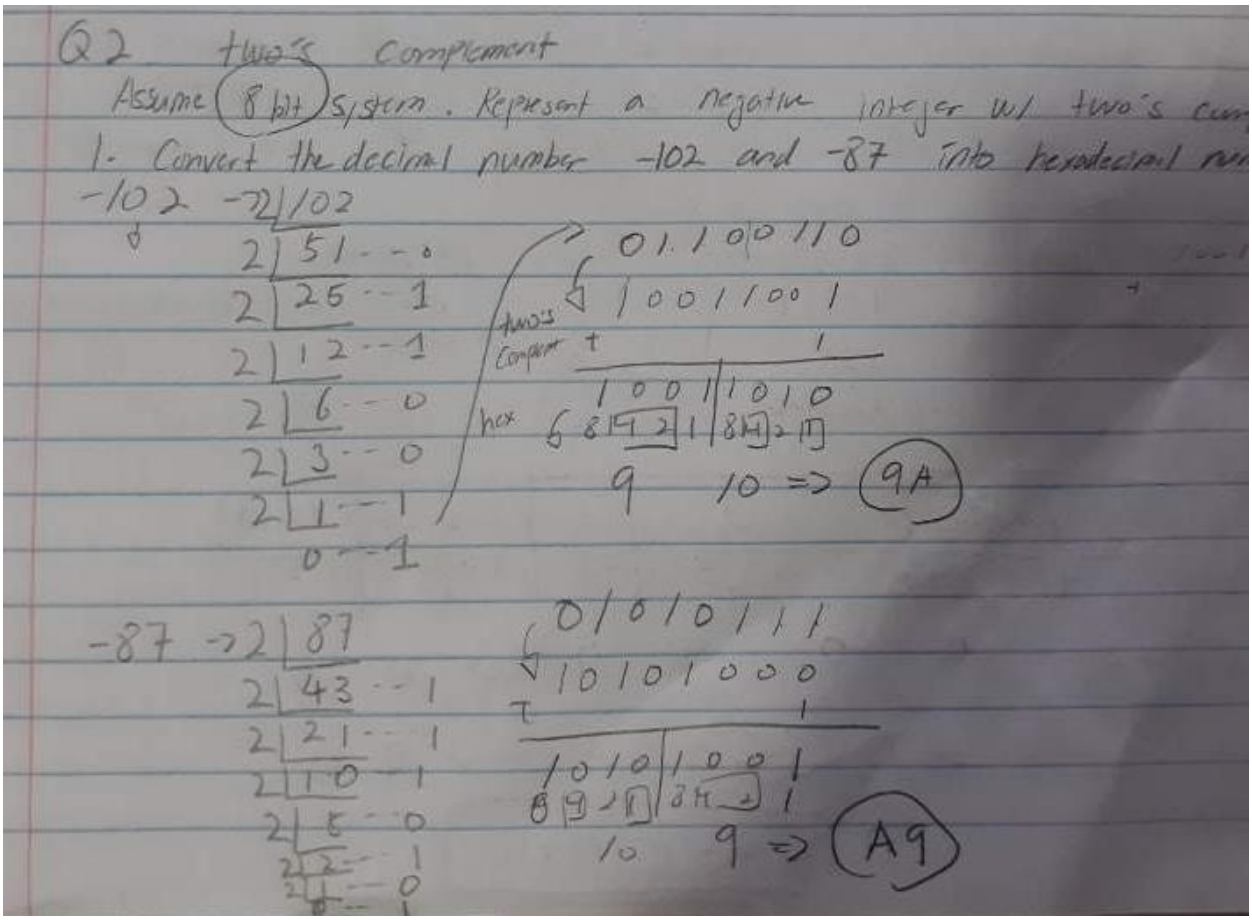
Q1. (4 pts) Number conversion. You must show your work. An answer without the full steps of work, you will get no points.



Q2. (4 pts) Two's complement

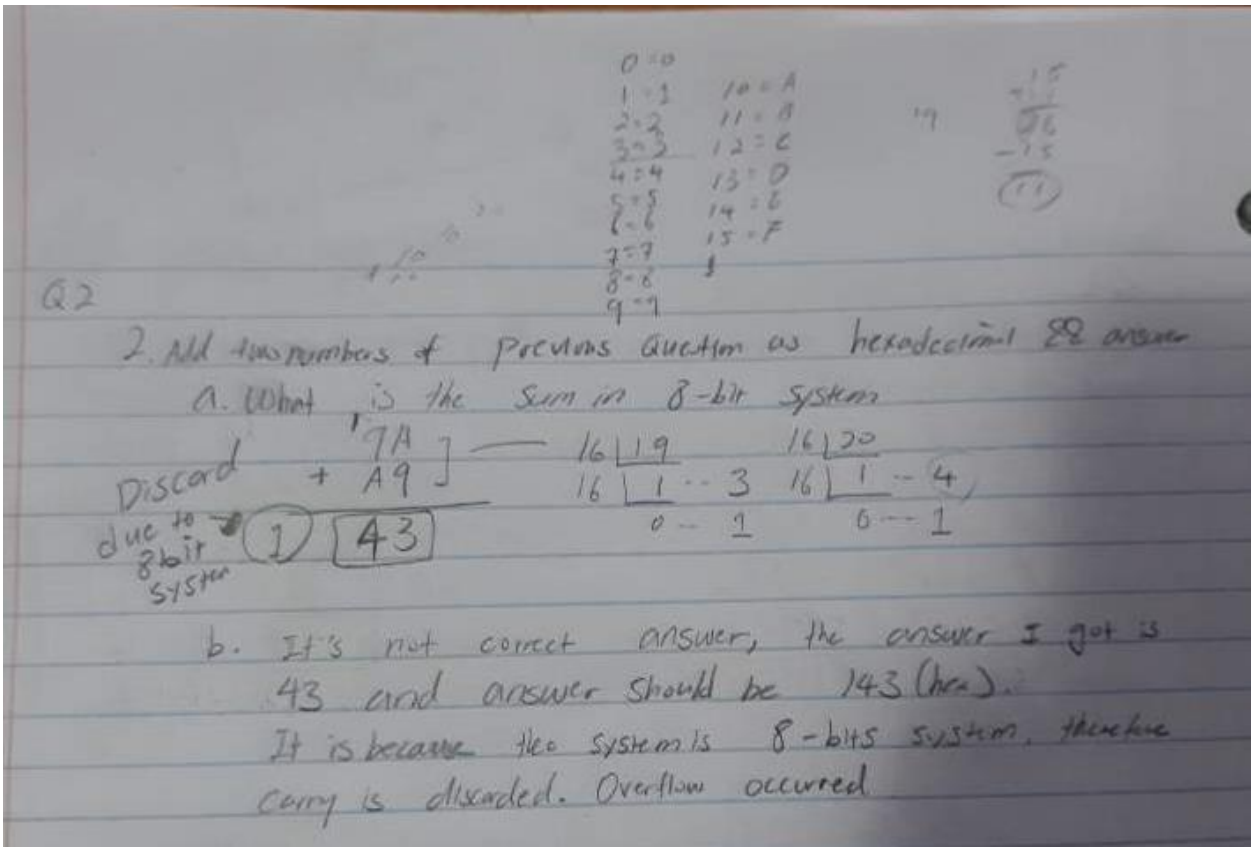
Assume that we are using **8-bits system**. Represent a negative integer with two's complement format.

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



2. Add two numbers of the previous question as hexadecimal, and answer,

   a. What is the sum in 8-bits system?
   b. Is it a correct answer? If it is not, explain why.



And even if I added in binary,
-102 => 0110 0110 => two's complement => 1001 1010
- 87 => 0101 0111 => two's complement => 1010 1001
Adding these two will result out
1 0100 0011 and we are dealing with 8 bits, => 0100 0011.

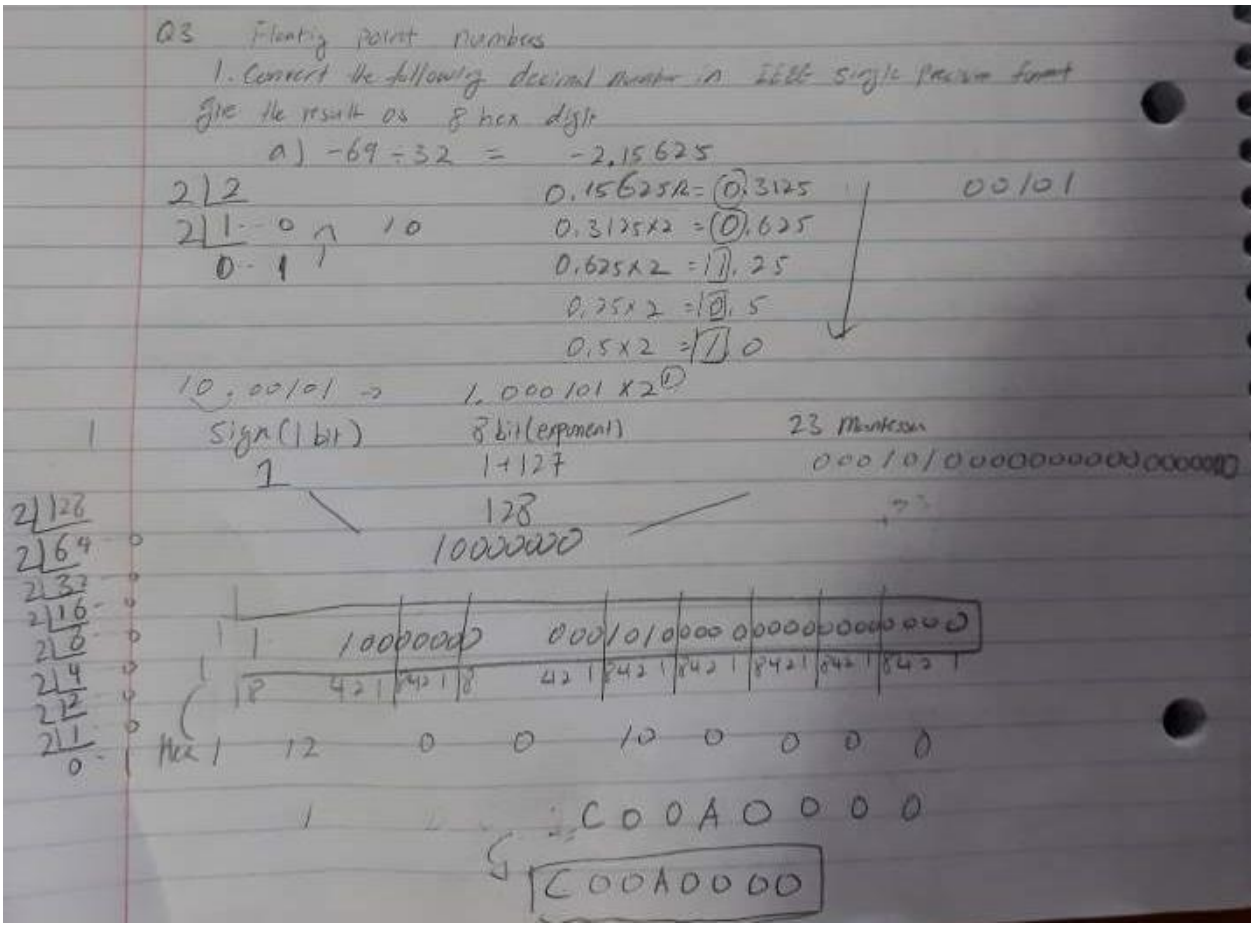
b. Simply it returns wrong answer due to overflow. (since we are in 8-bits system). Both hex and binary, when I convert to it, it results out 67. So wrong.

Q3. (8 pts) Floating point numbers

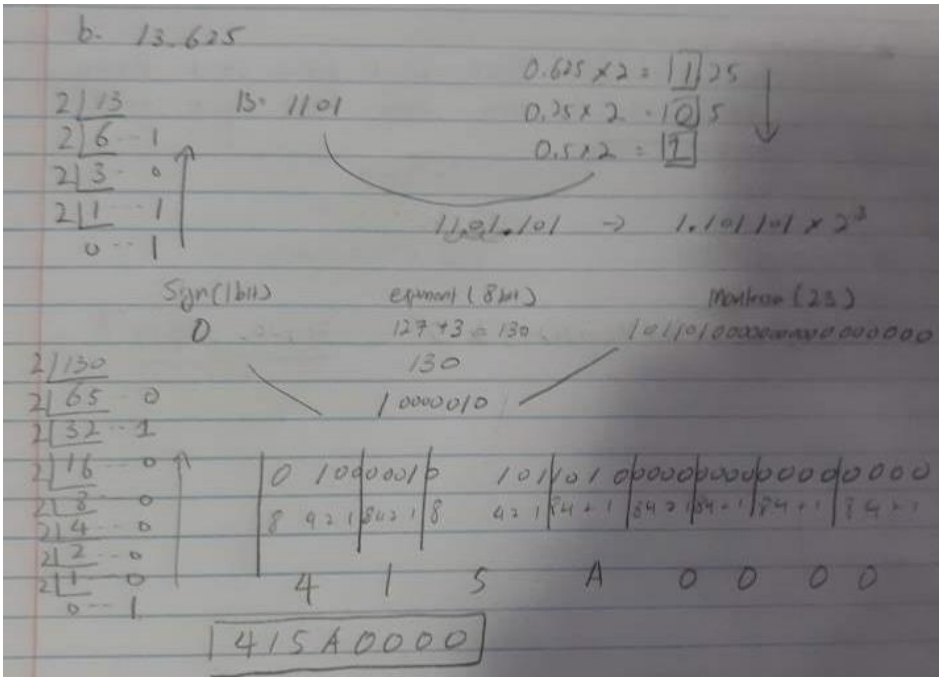
You have to show the steps (works). An answer without work will get no points.

1. Convert the following decimal numbers in IEEE single-precision format. Give the result as eight hexadecimal digits.

a) -69/32 (-69 divide by 32)

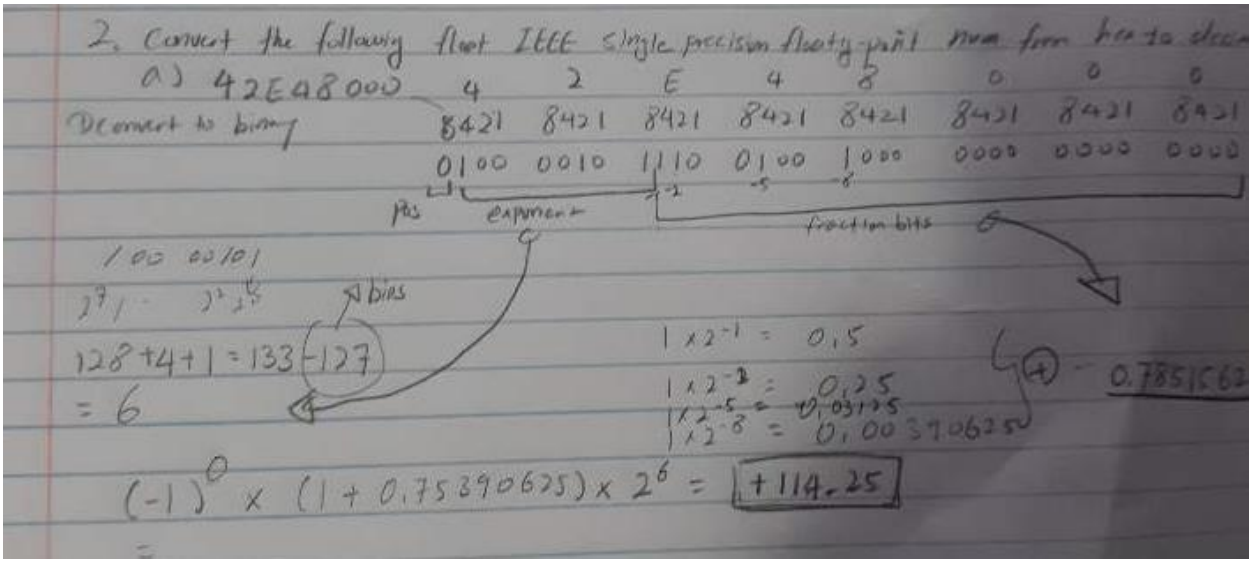

Q3 Floating point numbers
1. Convert the following decimal number in IEEE single precision format
give the result as 8 hex digit
a) -69 ÷ 32 = -2.15625
00101
10.00101 → 1.00010 1 x 2^1
Sign (1 bit)
8 bit (exponent)
23 Mantissa
1
1+127
128
10000000
00010100000000000000000
1 10000000 00010100000000000000000
Hex 1 12 0 0 10 0 0 0 0
C00A00000

b) 13.625

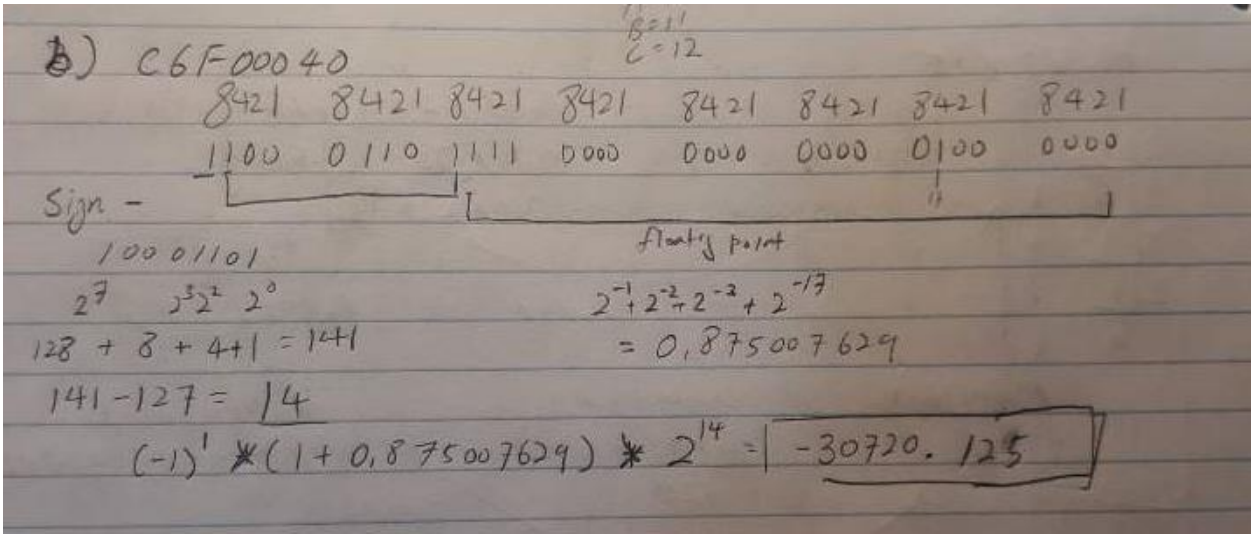

b. 13.625
13: 1101
1101.101 → 1.10110 1 x 2^3
Sign (1 bit)
exponent (8 bit)
Mantissa (23)
0
127+3 = 130
130
10000010
01000001 01010000000000000000000
4 1 5 A 0 0 0 0
41 5A0000

2. Convert the following floating IEEE single-precision floating-point numbers from hex to decimal:

a) 42E48000



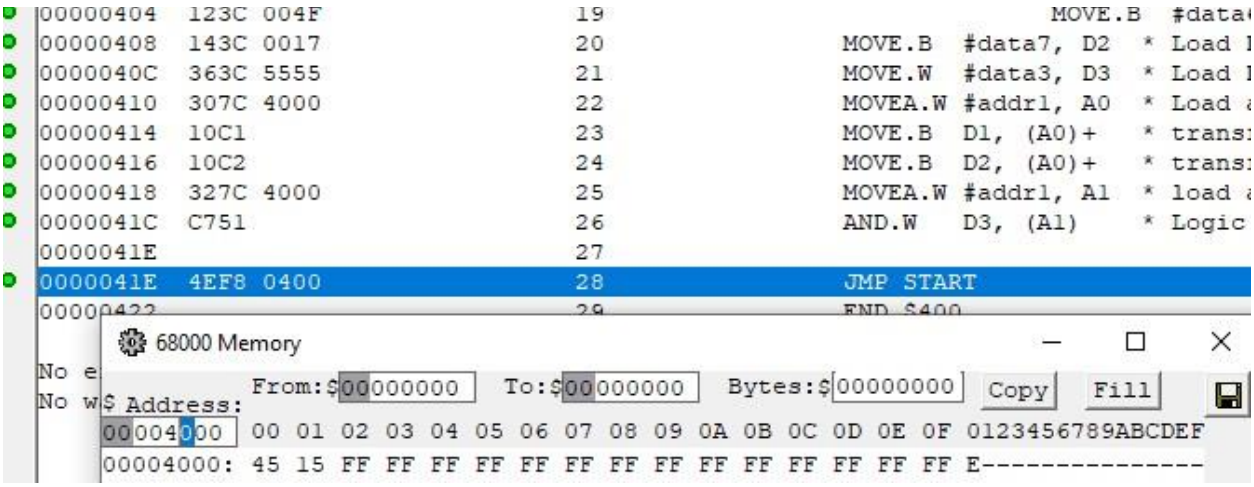
b) C6F00040



Q4. (4 pts) Introduction to 68K

Create a source file and analyze the results. Submit the listfile and answer the question after the code segments. For simplicity, name your source file as HW1-Q4.X68. Then your listfile will be named as HW1-Q4.L68.

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 beginning and start over again? 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)



Answer is
45 15

Describe how I got the answer: I wrote down the code and clicked the Assemble Source (F9) and it will prompt the .S68. From .S68, clicked view -> memory.

Since, the question mentioned, "WORD VALUE (not byte, or longword) of the data in memory location $4000 when the program is just about to loop back to the beginning and start over again?" so, I clicked trace info (F7) till it points at JMP START and change the $Address: to 4000 in Memory.

Since, question was asking 16 bits (word) data and I know each 00, 01 represent 8 bits, therefore I came up with answer 45 15.

Describe what I understand

I understood that because we had code

MOVEA.W #addr1, A0 // A0 has 4000

MOVE.B D1, (A0)+ // which moves D1 to memory 4000 because A0 has content of 4000 with ( ). And increase A0 to 4001 because of +

So, now at $4000, we have D1, which is data6 or 4F in hex

MOVE.B D2, (A0)+ //which moves D2 to memory 4001, and apply same logic. So, at memory 4001, it has D2 (data7) or 17 in hex.

Then,

MOVEA.W #addr1, A1 // now A1 has 4000.

And then, we are using AND.W D3, (A1)

Which is Grab D3 (it has data3, hex 5555) and apply AND Logic with data in memory 4000~4001 (because W is 16 bits). Which is 4F 17.

So, for 00, we must do and logic 55 and 4F.

To do AND logic, convert it to binary

55 => 01010101

4F => 01001111

Apply AND => 01000101 => 45 in hex

For 01, convert 55 and 17 to binary.

55 => 01010101

17 => 00010111

Apply And => 00010101 => 15 in hex

Therefore, 4000~4001 => 45 15.