

## 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.**

Homework 1

Q1

1. Convert the hexadecimal number 973D4 to base 17

Convert to decimal

	A	B	C	D	E	F	G
	10	11	12	13	14	15	16
$4 \times 16^0 = 4$							
$13 \times 16^1 = 208$							
$3 \times 16^2 = 768$							
$7 \times 16^3 = 28672$							
$9 \times 16^4 = 589824$							
<u>619476</u>							

7718D<sub>17</sub>

2. Convert the C5GE<sub>17</sub> to decimal number

$14 \times 17^0 = 14$				
$16 \times 17^1 = 272$				
$5 \times 17^2 = 1445$				
$12 \times 17^3 = 58956$				
<u>60687</u>				

## 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

The image shows a handwritten solution for converting decimal numbers to hexadecimal using two's complement. It is divided into two main parts, one for -102 and one for -87.

**For -102:**

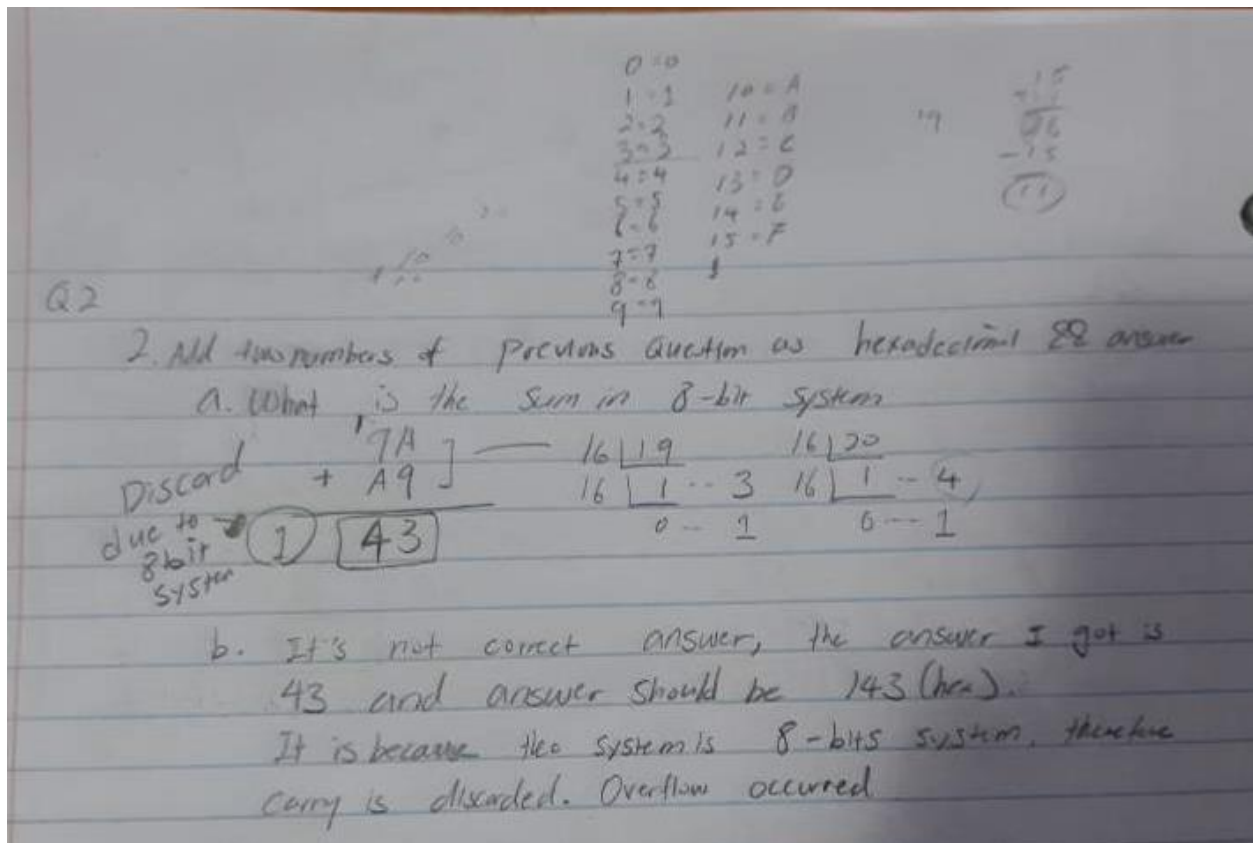
- Division by 2:  $-102 \div 2 = -51$  (remainder 0),  $-51 \div 2 = -25$  (remainder 1),  $-25 \div 2 = -12$  (remainder 1),  $-12 \div 2 = -6$  (remainder 0),  $-6 \div 2 = -3$  (remainder 0),  $-3 \div 2 = -1$  (remainder 1),  $-1 \div 2 = 0$  (remainder 1). The remainders read from bottom to top are 1, 0, 1, 1, 0, 1, 0.
- Two's complement: The binary representation is  $0110110$ . To find the two's complement, invert the bits to get  $1001001$  and add 1 to get  $1001010$ .
- Hexadecimal conversion: Group the bits into pairs:  $1001$  and  $0101$ . These correspond to hex digits 9 and A respectively. The final result is **9A**.

**For -87:**

- Division by 2:  $-87 \div 2 = -43$  (remainder 1),  $-43 \div 2 = -21$  (remainder 1),  $-21 \div 2 = -10$  (remainder 1),  $-10 \div 2 = -5$  (remainder 0),  $-5 \div 2 = -2$  (remainder 1),  $-2 \div 2 = -1$  (remainder 0),  $-1 \div 2 = 0$  (remainder 1). The remainders read from bottom to top are 1, 0, 1, 1, 0, 1, 1.
- Two's complement: The binary representation is  $0101101$ . To find the two's complement, invert the bits to get  $1010010$  and add 1 to get  $1010011$ .
- Hexadecimal conversion: Group the bits into pairs:  $1010$  and  $0111$ . These correspond to hex digits A and 7 respectively. The final result is **A7**.

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

- What is the sum in 8-bits system?
- 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 digits

a)  $-69.32 = -2.15625$

$2 \overline{) 2} \rightarrow 0.15625 \times 2 = 0.3125$   
 $2 \overline{) 1} \rightarrow 0.3125 \times 2 = 0.625$   
 $0 \cdot 1 \rightarrow 0.625 \times 2 = 1.25$   
 $0 \cdot 1 \rightarrow 0.25 \times 2 = 0.5$   
 $0 \cdot 1 \rightarrow 0.5 \times 2 = 1.0$

$10.00101 \rightarrow 1.000101 \times 2^3$

Sign (1 bit)

8 bit (exponent)

23 Mantissa

1

1127

00010100000000000000000

128

10000000

$2 \overline{) 26}$   
 $2 \overline{) 64}$   
 $2 \overline{) 32}$   
 $2 \overline{) 16}$   
 $2 \overline{) 8}$   
 $2 \overline{) 4}$   
 $2 \overline{) 2}$   
 $2 \overline{) 1}$   
 $0 \cdot 1$

$1 \cdot 10000000 \cdot 00010100000000000000000$   
 $1 \cdot 10000000 \cdot 00010100000000000000000$

$1 \cdot 10000000 \cdot 00010100000000000000000$   
 $1 \cdot 10000000 \cdot 00010100000000000000000$

$1 \cdot 10000000 \cdot 00010100000000000000000$

$1 \cdot 10000000 \cdot 00010100000000000000000$

b) 13.625

b. 13.625

$2 \overline{) 13} \rightarrow 13.1101$   
 $2 \overline{) 6} \rightarrow 0.625 \times 2 = 1.25$   
 $2 \overline{) 3} \rightarrow 0.25 \times 2 = 0.5$   
 $2 \overline{) 1} \rightarrow 0.5 \times 2 = 1.0$

$11.01101 \rightarrow 1.101101 \times 2^3$

Sign (1 bit)

exponent (8 bit)

Mantissa (23)

0

127 + 3 = 130

10110100000000000000000

130

10000010

$2 \overline{) 130}$   
 $2 \overline{) 65}$   
 $2 \overline{) 32}$   
 $2 \overline{) 16}$   
 $2 \overline{) 8}$   
 $2 \overline{) 4}$   
 $2 \overline{) 2}$   
 $2 \overline{) 1}$   
 $0 \cdot 1$

$0 \cdot 10000010 \cdot 10110100000000000000000$   
 $0 \cdot 10000010 \cdot 10110100000000000000000$

$0 \cdot 10000010 \cdot 10110100000000000000000$   
 $0 \cdot 10000010 \cdot 10110100000000000000000$

$0 \cdot 10000010 \cdot 10110100000000000000000$

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

a) 42E48000

2. Convert the following float IEEE single precision floating-point num from hex to dec

a) 42E48000

Convert to binary

4	2	E	4	8	0	0	0
8421	8421	8421	8421	8421	8421	8421	8421
0100	0010	1110	0100	1000	0000	0000	0000

sign: 0, exponent: 10, fraction bits: 23

100 00101  
 $2^7 + 2^2 + 2^0 = 128 + 4 + 1 = 133$   
 $133 - 127 = 6$

1  $\times 2^{-1} = 0.5$   
 1  $\times 2^{-2} = 0.25$   
 1  $\times 2^{-5} = 0.03125$   
 1  $\times 2^{-8} = 0.00390625$

$(-1)^0 \times (1 + 0.75390625) \times 2^6 = +114.25$

b) C6F00040

b) C6F00040

C	6	F	0	0	0	4	0
8421	8421	8421	8421	8421	8421	8421	8421
1100	0110	1111	0000	0000	0000	0100	0000

Sign: 1, exponent: 10, fraction bits: 23

100 01101  
 $2^7 + 2^3 + 2^2 + 2^0 = 128 + 8 + 4 + 1 = 141$   
 $141 - 127 = 14$

1  $\times 2^{-1} = 0.5$   
 1  $\times 2^{-2} = 0.25$   
 1  $\times 2^{-3} = 0.125$   
 1  $\times 2^{-4} = 0.0625$   
 1  $\times 2^{-5} = 0.03125$   
 1  $\times 2^{-6} = 0.015625$   
 1  $\times 2^{-7} = 0.0078125$

$(-1)^1 \times (1 + 0.875007629) \times 2^{14} = -30720.125$

#### 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)

```

00000404 123C 004F 19 MOVE.B #data6, D1 * Load 1
00000408 143C 0017 20 MOVE.B #data7, D2 * Load 1
0000040C 363C 5555 21 MOVE.W #data3, D3 * Load 1
00000410 307C 4000 22 MOVEA.W #addr1, A0 * Load 1
00000414 10C1 23 MOVE.B D1, (A0)+ * trans:
00000416 10C2 24 MOVE.B D2, (A0)+ * trans:
00000418 327C 4000 25 MOVEA.W #addr1, A1 * load 1
0000041C C751 26 AND.W D3, (A1) * Logic
0000041E 4EF8 0400 28 JMP START
00000422 29 FMD $400

```

68000 Memory

From: \$00000000 To: \$00000000 Bytes: \$00000000 Copy Fill

\$ Address: 00004000

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 0123456789ABCDEF

00004000: 45 15 FF FF FF FF FF FF FF FF FF FF FF FF FF E-----

**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.