**Name: \_\_\_\_\_\_Ananya\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ECE2049 HW #1-- C programming and Binary Number Representations**

***Problem #1 (30 pts)***

a) How many times would each of the following loop actually iterate on the MSP430F5529? **The loop will iterate 255 times**

unsigned int inVal, out, kk=1, mm=32767;

. . .

while (kk > 0){

inVal = (mm + kk)/kk;

out = myFunction(inVal);

mm /= 4;

kk = kk << 1;

}

Also, what are the values of kk, mm and inVal (in decimal) at the start and end of the first loop and at the start and end of the final loop?? *Note*: Code does nothing useful.

**Before First loop: kk= 1, mm= 32767, inVal = 1**

**After First loop: kk= 2, mm= 8191, inVal = 32768**

**Start of final loop: kk= 254, mm= 0, inVal = 1**

**End of final loop: kk= 255, mm= 0, inVal = 1**

-----------------------------------------------------------------

What is the value of count in the last 3 loops?

unsigned char count = 1;

while (count > 0) {

/\* Do some things

. . . . \*/

count++;

}

**Third to last loop: 253**

**Second to last loop: 254**

**Third to last loop: 255**

b) Rewrite this code segment using a while loop and if else statements instead of the for loop and switch case statements. *Note*: The functions alphaTouchPad() and led\_on() are NOT one from our demo lab.

**unsigned char trial, tP\_history[175];**

**....**

**trial = 0;**

**while (trial < 175)**

**{**

**tP = alphaTouchPad(); // returns letter A-E for touch pad**

**tP\_history[trial] = tP;**

**if(tP=='A'){**

**led\_on(0);**

**break;**

**}**

**else if(tP=='B'){**

**led\_on(1);**

**break;**

**}**

**else if (tP=='C'){**

**led\_on(2);**

**break;**

**}**

**else if (tP=='D'){**

**led\_on(3);**

**break;**

**}**

**else if (tP=='E'){**

**led\_on(4);**

**break;**

**}**

**else{**

**led\_all\_off();**

**break;**

**}**

**trial++;**

**}**

***Problem #2 (15 pts)***

1. Express the following numbers as *16 bit unsigned integers* 2015, 35000, 111.
   1. 2015 🡪 0x07Df 🡪 00000111 11011111
   2. 35000 🡪 0x88B8 🡪10001000 10111000
   3. 111 🡪 0x006F = 00000000 01101111
2. Express the following number as *16 bit signed (two's complement)* integers -8, 5600, -32755.
   1. 8 🡪 0000 0000 0000 1000

Complement 🡪 1111 1111 1111 0111

+1 🡪 0000 0000 0000 0001

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-8 🡪 1111 1111 1111 1000

b.

Division by 2 Quotient Remainder Bit #

(5600)/2 2800 0 0

(2800)/2 1400 0 1

(1400)/2 700 0 2

(700)/2 350 0 3

(350)/2 175 0 4

(175)/2 87 1 5

(87)/2 43 1 6

(43)/2 21 1 7

(21)/2 10 1 8

(10)/2 5 0 9

(5)/2 2 1 10

(2)/2 1 0 11

(1)/2 0 1 12

5600 🡪 1010111100000

c.

Division by 2 Quotient Remainder Bit #

(32755)/2 16377 1 0

(16377)/2 8188 1 1

(8188)/2 4094 0 2

(4094)/2 2047 0 3

(2047)/2 1023 1 4

(1023)/2 511 1 5

(511)/2 255 1 6

(255)/2 127 1 7

(127)/2 63 1 8

(63)/2 31 1 9

(31)/2 15 1 10

(15)/2 7 1 11

(7)/2 3 1 12

(3)/2 1 1 13

(1)/2 0 1 14

32755 🡪 0111 1111 1111 0011

Complement 🡪 1000 0000 0000 1100

+1 🡪 0000 0000 0000 0001

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

-32755 🡪 1000 0000 0000 1101

You are given the following 16-bit numbers 1FE4h, 0A57h, and F742h. Each of these values could be interpreted as

1. An *unsigned* number
   1. 1FE4 🡪 0001 1111 1110 0100

Base 10 equivalent 🡪 (1 × 16³) + (15 × 16²) + (14 × 16¹) + (4 ×16⁰) = 8164

* 1. 0A57 🡪 0000 1010 0101 0111

Base 10 equivalent 🡪 (0 × 16³) + (10 × 16²) + (5 × 16¹) + (7 × 16⁰) = 2647

1. A *two's-complement* number.
   1. F742 🡪 1111 0111 0100 0010

Base 10 equivalent 🡪 (15 × 16³) + (7 × 16²) + (4 × 16¹) + (2 × 16⁰) = 63298

***Problem #3 (15 pts)***

a) What are the ASCII codes (in hex) for the characters '0', '1', '2', '3' ….'9'?

0. 00

1. 01

2. 02

3. 03

4. 04

5. 05

6. 06

7. 07

8. 08

9. 09

b) In lab you will regularly need to display numbers on the LCD screen. Therefore  
 you will need to convert between integer digits and their ASCII representations and the reverse. What C code (variable declarations and 1 line of code) would you use convert a single decimal digit to its ASCII code?

char var1;

char var2;

var = var2 + 0x30;

c) Assume the integer value D = 35652 has been converted for display into an array of ASCII values, declared as char D\_asc[8];

What value (in hex) should be stored in each array location so that the number would display properly (i.e. *right justified with digits in left to right order*) if printed on our Sharp LCD screen? Explain you reasoning.

\*To find the ASCII value of an integer, add 48 with every digit of the number

D = 35652

=> 48+3 48+5 48+6 48+5 48+2

=> 51 53 54 53 50 (in decimal)

=> 0x33 0x35 0x36 0x35 0x32 (in hexa decimal)

D\_asc[7] = 0x0

D\_asc[6] = 0x0

D\_asc[5] = 0x0

D\_asc[4] = 0x33

D\_asc[3] = 0x35

D\_asc[2] = 0x36

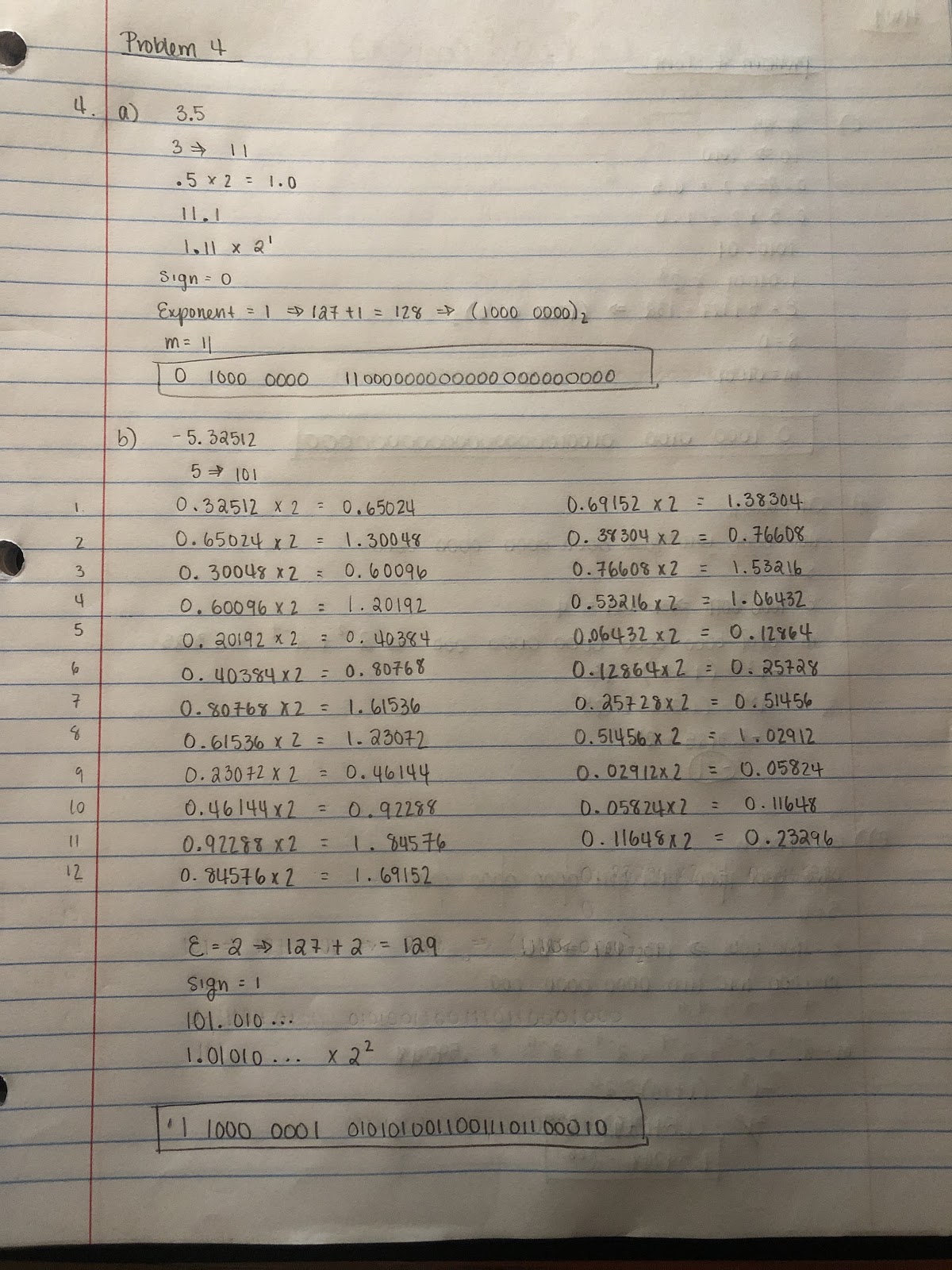
D\_asc[1] = 0x35

D\_asc[0] = 0x32

***Problem #4 (25 pts)***

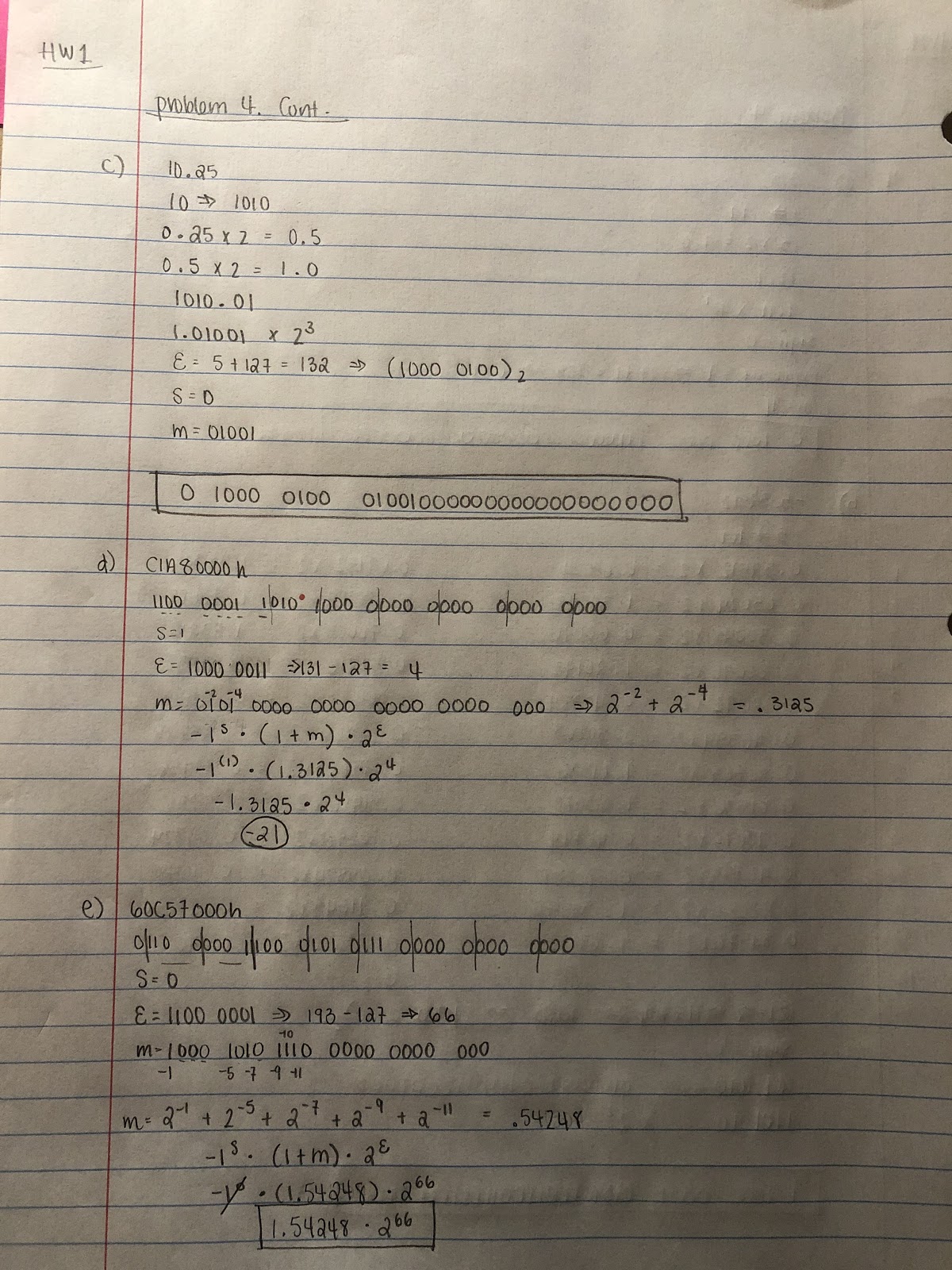
Convert the following numbers from decimal to IEEE 32-bit floating point format

a) 3.5 b) -5.32512 c) 10.25



The following numbers are encoded using 32-bit IEEE floating point format. Find the decimal values that they represent.

d) C1A80000h e) 60C57000h

***Problem #5 (15 pts)***

In a table like the one below, show how the following variables would be stored successively in memory by

(a) a Little-Endian microprocessor

(b) a Big-Endian microprocessor

Show the values in hex (not binary) starting at address 02000 h. That is, array str is stored beginning at address 02000h. Label each address in your table. Remember each address in memory holds 1 byte.

*Is the MSP430F5529 big or little endian*? Little Endian

char str[4] = “Tst:”; // array of ASCII text (No NULL terminator)

\* The character array will be stored in the same order in both little endian and big endian as character is just a byte. The first 4 addresses from 2000h to 2003h are occupied by four characters in the array as each character is one byte.

🡪 “Tst:” in hex is 54 73 74 3A

float ss = -128.75; // IEEE 32-bit floating point

\*float is 4 bytes. In little endian, least significant byte is stored first (lowest memory address) and vice versa in big endian

🡪 -128.75 🡪 11000011000000001100000000000000 🡪 c3 00 c0 00

long long unsigned int ser\_num = Ox3EF5CBDF009CB6A4; //unsigned 64-bit integer

🡪3E F5 CB DF 00 9C B6 A4

int jj = -5; // a two's comp 16 bit integer

🡪 -5 in 2's complement 16 bits 🡪 1111 1111 1111 1011 🡪 FF FB

int arr[2] = {32767, 4}; // 2 element array of integers

\*convert both integer values in the array to hex

long unsigned int trp = 0x00182092;

|  |  |  |  |
| --- | --- | --- | --- |
| ***Address*** | Little Endian | Big Endian | Variable |
| 02000 h | 54 | 54 | str |
| 02001 h | 73 | 73 | str |
| 02002 h | 74 | 74 | str |
| 02003 h | 3A | 3A | str |
| 02004 h | 00 | c3 | ss |
| 02005 h | c0 | 00 | ss |
| 02006 h | 00 | c0 | ss |
| 02007 h | c3 | 00 | ss |
| 02008 h | A4 | 3E | ser\_num |
| 02009 h | B6 | F5 | ser\_num |
| 0200A h | 9C | CB | ser\_num |
| 0200B h | 00 | DF | ser\_num |
| 0200C h | DF | 00 | ser\_num |
| 0200D h | CB | 9C | ser\_num |
| 0200E h | F5 | B6 | ser\_num |
| 0200F h | 3E | A4 | ser\_num |
| 02010 h | FB | FF | jj |
| 02011 h | FF | FB | jj |
| 02012 h | FF | 7F | arr |
| 02013 h | 7F | FF | arr |
| 02014 h | 04 | 00 | arr |
| 02015 h | 00 | 04 | arr |
| 02016 h | 92 | 00 | trp |
| 02017 h | 20 | 18 | trp |
| 02018 h | 18 | 20 | trp |
| 02019 h | 00 | 92 | trp |
| 0201A h |  |  |  |
| 0201B h |  |  |  |
| 0201C h |  |  |  |
| 0201D h |  |  |  |
| 0201E h |  |  |  |
| 0201F h |  |  |  |