# ECE2049 Homework #1

**Submitted by:** Jack C. Bergin

**Date:** 02/07/21

| Question | Grade |
|---|---|
| 1-- 25 | |
| 2 -- 15 | |
| 3 -- 10 | |
| 4 -- 10 | |
| 5 -- 25 | |
| 6 – 15 | |
| Total: 100 | |

**YOU MUST ATTACH THIS COVER SHEET TO YOUR HW!!**

**<u>Problem 1:</u>**

**Part A:**

I. The loop iterated through 10,922 times. The loop will be exited when kk exceeds the 32767 int range limit and goes back to 0 thus exiting with the while loop logic.

First Loop:

- Kk Value:
    - Start: 1
    - End: 4
- Mm Value:
    - Start: 32767
    - End: 16383
- In-Val Value:
    - Start: 0
    - End: 1

Final Loop:

- Kk Value:
    - Start:32766
    - End: 0
- Mm Value:
    - Start: 15
    - End: 15
- In-Val Value:
    - Start: 1
    - End:  1

II. The last 3 loops are going to have the values: 252, 254, 0 because char hits its overflow at 255.

**Problem 1 (Continued):**

**Part B:**

```
unsigned char trial, tP_history[175];
while(trial&lt;175){
        unsigned char tP=alphaTouchPad();
        tP_history[trial]=tP;

        if(tP==’A’){
                led_on(0);
        }
        if(trial==’B’){
                led_on(1);
        }
        if(trial==’C’){
                led_on(2);
        }
        if(trial==’D’){
                led_on(3);
        }
        if(trial==’E’){
                led_on(4);
        }
        else{
                led_all_off();
        }
        trial++;
}
```

**Problem 1 (Continued):**

**Part C:**

```
int main(){
  //Declare arrays and needed variables
  int sizeCnst = 500;
  int alphaCnst = 0.9;
  int rawArray[500];
  int expoArray[500];

  //Give values to both arrays
  for (int i = 0; i < sizeCnst; i++){
    rawArray[i] = (rand()%2001)-2000
  }
  for (int j = 0; j <sizeCnst; j++){
    if(j=0){
      expoArray[j] = rawArray[j]
    }
    else{
      expoArray [j] = (1-alphaConst) * rawArray[j] + alphaConst * expoArray[j-1];
    }
  }

  //Print out the array values
  for (int k = 0; k < sizeCnst(); k++){
    printf("Raw Data: ", rawArray[k], "\n");
  }
  for (int g = 0; g < sizeCnst(); g++){
    printf("Exponential Average: ", expoArray[g], "\n");
  }
return 0;
}
```

**Problem 2:**

a) Express the following numbers as 16 bit unsigned integers 1652, 11000, 256.Show all your work.

**1652:**

- 1652/2 = 826 - 0
- 826/2 = 413 - 0
- 413/2 = 206 R1 - 1
- 206/2 = 103 - 0
- 103/2 = 51 R1 - 1
- 51/2 = 25 R1 - 1
- 25/2 = 12 R1 - 1
- 12/2 = 6 - 0
- 6/2 = 3 - 0
- 3/2 = 1 R1 - 1
- ½ = R1 - 1

0000 0110 0111 0100b

**11000:**

- 11000/2 = 5500 - 0
- 5500/2 = 2750 - 0
- 2750/2 = 1375 - 0
- 1375/2 = 687 R1 - 1
- 687/2 = 343 R1 - 1
- 343/2 = 171 R1 - 1
- 171/2 = 85 R1 - 1
- 85/2 = 42 R1 - 1
- 42/2 = 21 - 0
- 21/2 = 10 R1 - 1
- 10/2 = 5 - 0
- 5/2 = 2 R1 - 1
- 2/2 = 1 - 0
- ½ = 1 R1 - 1

0001 0101 0111 1000b

**<u>Problem 2 (Continued):</u>**

**256:**

- 256/2 = 128 - 0
- 128/2 = 64 - 0
- 64/2 = 32 - 0
- 32/2 = 16 - 0
- 16/2 = 8 - 0
- 8/2 = 4 - 0
- 4/2 = 2 - 0
- 2/2 = 1 - 0
- ½ = 1 R1 - 1

0000 0001 0000 0000b

b) Express the following number as 16 bit signed (two's complement)integers -7, 6200, -32750. Show all your work.

-7 = 1000 0000 0000 0111b
0111 1111 1111 1000 +1
= 0111 1111 1111 1001

6200 = 0001 1000 0011 1000b

- Because the most significant number is positive the 2's complement is the same as the original 16 bit representation: 0001 1000 0011 1000.

-32750 = **1**111 1111 1110 1110b
0000 0000 0001 0001 + 1
= 0000 0000 0001 0010

You are given the following 16-bit numbers 2A56h, 0C45h, and E25Ah. Each of these values could be interpreted as.

c) An unsigned number.

1. 2A56h → 10838
2. 0C45h → 3141
3. E25Ah → 57946

d) Sign-magnitude numbers.

1. 10838d → 0010 1010 0101 0110
2. 3141d → 0000 1100 0100 0101
3. 57946d → 1110 0010 0101 1010

e) A two's-complement number.

1. 10838d → 0010 1010 0101 0110 = 10838d
2. 3141d → 0000 1100 0100 0101 = 3141d
3. 57946d → 1110 0010 0101 1010
   0001 1101 1010 0110 +1 = -7590d

**Problem 3:**

a) Express the base 10 integer Y = 32141169 in BCD?
   - 32141169/2 = 16,070,584 R - 1
   - 16,070,584/2 = 8,035,292 - 0
   - 8,035,292/2 = 4,017,646 - 0
   - 4,017,646/2 = 2008823 - 0
   - 2008823/2 = 1,004,411 R - 1
   - 1,004,41/2 = 502,205 R - 1
   - 502,205/2 = 251,102 R - 1
   - 251,102/2 = 125,551 - 0
   - 125,551 = 62,775 R - 1
   - 62,775/2 = 31,387 R - 1
   - 31,387/2 = 15,693 R - 1
   - 15693/2 = 7,846 R - 1
   - 7,846/2 = 3,923 - 0
   - 3,923/2 = 1961 R - 1
   - 1961/2 = 980 R - 1
   - 980/2 = 490 - 0
   - 490/2 = 245 - 0
   - 245/2 = 122 R - 1
   - 122/2 = 61 - 0
   - 61/2 = 30 R - 1
   - 30/2 = 15 - 0
   - 15/2 = 7 R - 1
   - 7/2 = 3 R - 1
   - 3/2 = 1 R - 1
   - ½ = 1 R - 1
   - 0001 1110 1010 0110 1111 0111 0001b

b) Assume that the variable X is encoded in BCD. What is the decimal equivalent value of X if X = 0011 1001 0111 0110b?
   2^0+2^4+2^5+2^6+2^8+2^9+2^10+2^11+2^13+2^14+2^17+2^19+2^21+2^22+2^23+2^24 = 14710

**Problem 4:**

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

- 0 = 0x30
- 1 = 0x31
- 2 = 0x32
- 3 = 0x33
- 4 = 0x34
- 5 = 0x35
- 6 = 0x36
- 7 = 0x37
- 8 = 0x38
- 9 = 0x39

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

- char decimalVal;
- char asciiVal = decimalVal + 30;

c) Assume the integer value D = 56987 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 your reasoning. Hint: Check the Lab 0 demo code.

- D = 56987:
  - 5d → +48 (to properly set up hex conversion) → 53d → 35h
  - 6d → +48 (to properly set up hex conversion) → 54d → 36h
  - 9d → +48 (to properly set up hex conversion) → 57d → 39h
  - 8d → +48 (to properly set up hex conversion) → 56d → 38h
  - 7d → +48 (to properly set up hex conversion) → 55d → 37h

  D_asc[7] = 0x20
  D_asc[6] = 0x20
  D_asc[5] = 0x20
  D_asc[4] = 0x35
  D_asc[3] = 0x36
  D_asc[2] = 0x39
  D_asc[1] = 0x38
  D_asc[0] = 0x37

**Problem 5:**

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

a) 5.5d = 101.1b
- S = 0
- E = 127+2 = 129
    - 10000001
- F = 1000
- SEF = 01000000101100000000000000000000
- 0x40b00000h

b) -8.75d = 1000.11b
- S = 1
- E = 127+3 = 130
    - 10000010
- F = 1100
- SEF = 11000001000011000000000000000000
- 0xc10c0000h

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

c) C0E40000h = 11000000111001000000000000000000
- -7.125d

d) 42C82000h = 1000010110010000010000000000000
- 100.0625d

**Problem 6:**

The MSP430 is little endian so the least significant byte is stored first.

| Value | Address | Little Endian | Big Endian |
|---|---|---|---|
| “Tst” | 02000h | 74h | 54h |
|  | 02001h | 73h | 73h |
|  | 02002h | 54h | 74h |
| -17.75 | 02003h | C1h | 00h |
|  | 02004h | 8Eh | 00h |
|  | 02005h | 00h | 8Eh |
|  | 02006h | 00h | C1h |
| ser_num | 02007h | A9h | 59h |
|  | 02008h | 11h | 68h |
|  | 02009h | 97h | ABh |
|  | 0200Ah | 5Ah | 89h |
|  | 0200Bh | 89h | 5Ah |
|  | 0200Ch | ABh | 97h |
|  | 0200Dh | 68hh | 11h |
|  | 0200Eh | 59h | A9h |
| -3 | 0200Fh | FDh | FFh |
|  | 02010h | FFh | FDh |
| 32766 | 02011h | FEh | 7Fh |
|  | 02012h | 7Fh | FEh |
| 5 | 02013h | 05h | 05h |
| A95E13C4 | 02014h | C4h | A9h |
|  | 02015h | 13h | 5Eh |

|  |  |  |  |
|---|---|---|---|
|  | 02016h | 5Eh | 13h |
|  | 02017h | A9h | C4h |