CS559: Computer Systems Lab

READ THE FOLLOWING INSTRUCTIONS CAREFULLY. FAILING TO ADHERE TO IT WILL LEAD TO DEDUCTION OF MARKS.

1. Make sure to give the correct information in your submission link. You will receive a response email for every submission. **You must save it for future reference for the whole semester as you will be asked to show it.**

2. Every assignment will have multiple questions. **For every question, you will save your solution (i.e. program) as a .c file.** So if "Assignment 01" has 4 questions (marked as 1, 2, 3 and 4), then you will save your solution program as 1.c, 2.c, 3.c and 4.c.

   Note: Your program will be compiled to check if your solution is correct and you will be marked accordingly. If any solution file is missing or not submitted, the marks for that solution will be zero.

3. Submit your files in a compressed folder (rar or zip) as per "submission instructions" given in your assignment question paper.

   Note: You must make sure to upload in .zip or .rar. **Folders submitted with any other extension or wrong extension will not be evaluated and awarded zero.**

4. The student must ensure that the submitted file is not corrupted and can be unzipped properly. **Corrupted files that cannot be opened will be given zero. Similarly bad filenames having any other extension than ".c" will not be evaluated and given zero.**

5. You will be able to upload your assignment multiple times within the deadline span, but only your last submission will be treated as final and considered for evaluation. **No plea/request to consider intermediate submissions for evaluation will be entertained.** So, make sure to carefully check that you are submitting the correct assignment.

6. The submission link will be automatically deactivated after the deadline, and no request/plea for extension will be accepted. **Non-submission /wrong submission will be automatically awarded zero.**

7. Your code will be checked for similarity and you will be penalized according to the following rule:

   Similarity above 75%: 50% deduction

   Similarity of 100%: full deduction

8. **Compiler Information: Program must be compiled using online gdb compiler. If your program can not be compiled using gdb compiler then marks will be reduced.**
   Compiler Link : `https://www.onlinegdb.com/online_c_compiler`

9. Marks for every assignment (other than practice assignments) will be mailed to you with remarks. **You will be allowed a deadline for rebuttal, after which no rebuttal will be entertained.**

CS 559: Computer Systems Lab
Date: [September 02 - 2023]
Assignment [4]
Submission Deadline: [September 09 - 2023 - 1800 IST]

Submission Instruction: Store your assignments in folder and compress it as a rar/zip file (filename should be in this format: roll-number_assign4.rar or roll-number_assign4.zip). For example, if your roll number is 2211CS01, store your assignment as 2211CS01_assign4.rar or 2211CS01_assign4.zip. Also, save each program in the format given beside each question. Upload the same at the below link:
`https://forms.gle/mJcpVDgS4FWVyXTo6`

Problem 1 (Save as 1.c/cpp) :- Write a program using only c/c++ to subtract two BCD numbers. (**Note:** proper logic should be present in the program and output should be in the format stated below) [10 points]

Sample Input:
Number of test cases: 2

First number: 835
Second number: 274

First number: 429
Second number: 476

Expected Output:

835 (1000 0011 0101) - 274 (1101 1000 1011) = 561 (0101 0110 0001)

429 (0100 0010 1001) - 476 (1011 1000 1001) = -47 (- 0000 0100 0111)

CS559: Computer Systems Lab
Assignment 4 - Evaluation Criteria
Assignment Deadline:[September, 09 2023 - 1800 Hrs. IST]

Problem 1:

- Improper indentation: -1 points
- Compilation error: -1 points
- Input not taken(i.e hard-coded): -2 points
- Wrong logic/output: -6 points
  - Conversion Logic (Decimal to BCD): -3 points
  - BCD Subtraction Logic: -3 points

Supplementary Material
For CS559 Assignment 4 , Dated September 02, 2022

BCD or **Binary Coded Decimal** is that number system or code which has the binary numbers or digits to represent a digital number. A decimal number contains 10 digits (0-9). Now the equivalent binary numbers can be found out of these 10 decimal numbers. In case of **BCD** the binary number formed by four binary digits, will be the equivalent code for the given decimal digits. In **BCD** we can use the binary number from 0000-1001 only, which are the decimal equivalent from 0-9 respectively. Suppose if a number have single decimal digit then it's equivalent **Binary Coded Decimal** will be the respective four binary digits of that decimal number and if the number contains two decimal digits then it's equivalent **BCD** will be the respective eight binary of the given decimal number. Four for the first decimal digit and next four for the second decimal digit. It may be cleared from an example.

Let, $(12)_{10}$ be the decimal number whose equivalent **Binary coded decimal** will be 00010010.
Four bits from L.S.B is binary equivalent of 2 and next four is the binary equivalent of 1.
Table given below shows the binary and **BCD** codes for the decimal numbers 0 to 15.

From the table below, we can conclude that after 9 the decimal equivalent binary number is of four bit but in case of BCD it is an eight bit number. This is the main difference between Binary number and binary coded decimal. For 0 to 9 decimal numbers both binary and BCD is equal but when decimal number is more than one bit BCD differs from binary.

Decimal Number	**Binary Number**	**BCD Equivalent**
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000
11	1011	0001 0001
12	1100	0001 0010
13	1101	0001 0011
14	1110	0001 0100
15	1111	0001 0101

BCD Addition

Like other number system in BCD arithmetical operation may be required. BCD is a numerical code which has several rules for addition. The rules are given below in three steps with an example to make the idea of **BCD Addition** clear.

1. At first the given number are to be added using the rule of binary. For example,
   Case 1:

```
   1010
 + 0101
 ------
   1111
```

   Case 2:

```
   0001
 + 0101
 ------
   0110
```

2. In second step we have to judge the result of addition. Here two cases are shown to describe the rules of **BCD Addition**. In case 1 the result of addition of two binary number is greater than 9, which is not valid for BCD number. But the result of addition in case 2 is less than 9, which is valid for BCD numbers.

3. If the four bit result of addition is greater than 9 and if a carry bit is present in the result then it is invalid and we have to add 6 whose binary equivalent is $(0110)_2$ to the result of addition. Then the resultant that we would get will be a valid binary coded number. In case 1 the result was $(1111)_2$, which is greater than 9 so we have to add 6 or $(0110)_2$ to it.

   $(1111)_2 + (0110)_2 = 00010101 = 15$

   Therefore, you can see the result is valid in BCD.
   But in case 2 the result was already valid BCD, so there is no need to add 6. This is how BCD Addition could be.
   Now a question may arrive that why 6 is being added to the addition result in case BCD Addition instead of any other numbers. It is done to skip the six invalid states of binary coded decimal i.e. from 10 to 15 and again return to the BCD codes.

Now the idea of BCD Addition can be cleared from two more examples.

Example 1:
Let, 0101 is added with 0110.

```
    0101
+   0110
-----------
    1011  → Invalid BCD number
+   0110  → Add 6
-----------
0001 0001 → Valid BCD number
```

Check your self.
$(0101)_2 \rightarrow (5)_{10}$ and $(0110)_2 \rightarrow (6)_{10}$ and $(0001\ 0001)_{BCD} \rightarrow (11)_{10}$
$(5)_{10} + (6)_{10} = (11)_{10}$

Example 2:
Now let 0001 0001 is added with 0010 0110.

```
    0001 0001
+   0010 0110
-------------
    0001 0001 → Valid BCD number
```

Check your self.
$(0001\ 0001)_{BCD} \rightarrow (11)_{10}$ and $(0010\ 0110)_{BCD} \rightarrow (26)_{10}$ and $(0011\ 0111)_{BCD} \rightarrow (37)_{10}$
$(11)_{10} + (26)_{10} = (37)_{10}$
So no need to add 6 as because both $(0011)_2 = (3)_{10}$ and $(0111)_2 = (7)_{10}$ are less than $(9)_{10}$. This is the process of BCD Addition.

BCD Subtraction

There are several methods of **BCD Subtraction**. BCD subtraction can be done by 1's compliment method. We will clear our idea on this method of **BCD Subtraction**.

Method of BCD Subtraction:

We will do BCD Subtraction by 1's compliment method. There are several steps for this method as given below:-

1. At first 1's compliment of the subtrahend is done.

2. Then the complimented subtrahend is added to the other number from which the subtraction is to be done. This is called adder 1.

3. Now in BCD Subtraction there is a term 'EAC(end-around-carry)'. If there is a carry i.e. if EAC = 1 the result of the subtraction is +ve and if EAC = 0 then the result is –ve. A table shown below gives the rules of EAC.

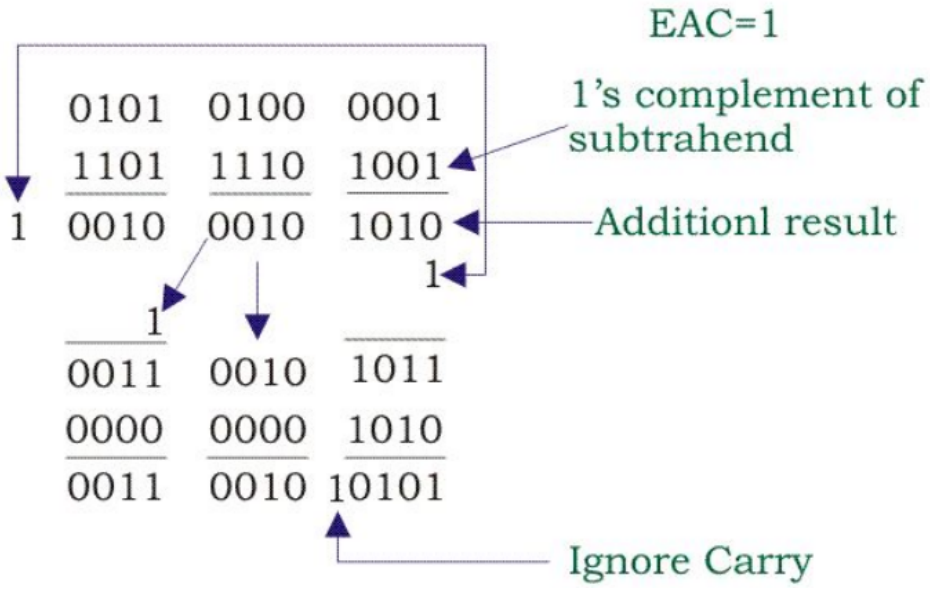
carry of individual groups	**EAC = 1**	**EAC = 0**
1	Transfer real result of adder 1 and add 0000 in adder 2	Transfer 1's compliment result of adder 1 and add 1010 in adder 2
0	Transfer real result of adder 1 and add 1010 in adder 2	Transfer 1's compliment result of adder 1 and add 0000 to adder 2

4. In the final result if any carry bit occurs the it will be ignored.

Examples given below would provide a clear idea of BCD Subtraction.

Example 1:
In this example 0010 0001 0110 is subtracted from 0101 0100 0001.

- At first 1's compliment of the subtrahend is done, which is 1101 1110 1001 and is added to 0101 0100 0001. This step is called adder 1.
- Now after addition if any carry occurs then it will be added to the next group of numbers towards MSB. Then EAC will be examined. Here, EAC = 1. So the result of addition is positive and true result of adder 1 will be transferred to adder 2.
- Now notice from LSB. There are three groups of four bit numbers. 1010 is added 1011 which is the first group of numbers because it do not have any carry. The result of the addition is the final answer.
- Carry 1 will be ignored as it is from the rule.
- Now move to the next group of numbers. 0000 is added to 0010 and gives the result 0010. It is the final result again.
- Now again move to the next group here 0000 is also added to 0011 to give the final result 0011.
- You may have noticed that in this two groups 0000 is added, because result of first adder do not contain any carry. Thus the results of the adder 2 is the final result of BCD Subtraction.

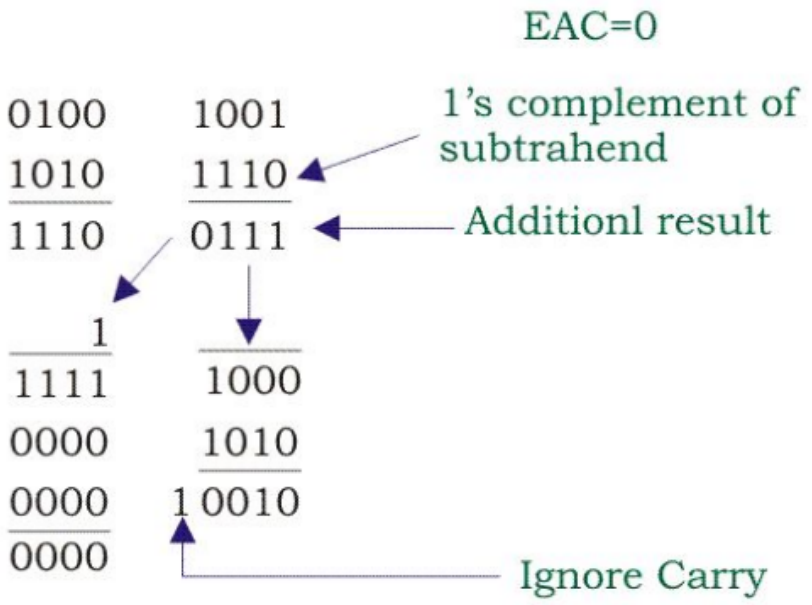


Therefore, $(0101\ 0100\ 0001)_{BCD} - (0010\ 0001\ 0110)_{BCD} = (0011\ 0010\ 0101)_{BCD}$

Check your self.
$(0101\ 0100\ 0001)_{BCD} \rightarrow (541)_{10}$ and $(0010\ 0001\ 0110)_{BCD} \rightarrow (216)_{10}$ and $(0011\ 0010\ 0101)_{BCD} \rightarrow (325)_{10}$
$(541)_{10} - (216)_{10} = (325)_{10}$

Example 2:
In this example let 0101 0001 be subtracted from 0100 1001.

- As per rule firstly 1's compliment of the subtrahend is done. Then the addition is done and the result is checked. Here EAC = 0, so the overall result will be –ve.
- Now see the result of adder 1 from LSB. 1's compliment value of 0111 is transferred to adder 2 and it is added with 1010 since no carry is added with it as per the rule. The answer is the final result.
- Now move to the next result of adder 1 i.e. 1110. Here 1 is added to it which is the carry of the previous result. Then it' value is 1's complimented i.e. 0000 and it is added to 0000. Result of adder 2 is the final result. This is the final result of BCD Subtraction.



Therefore, $(0100\ 1001)_{BCD} - (0101\ 0001)_{BCD} = (-\ 0000\ 0010)_{BCD}$

Check your self.
$(0100\ 1001)_{BCD} \rightarrow (49)_{10}$ and $(0101\ 0001)_{BCD} \rightarrow (51)_{10}$ and $(-\ 0000\ 0010)_{BCD} \rightarrow (-2)_{10}$
$(49)_{10} - (51)_{10} = (-2)_{10}$