CSCI 360 **Assignment 4** **Name__KEY_____**
Summer 2020 **ABENDs and Dump Reading**
25 points

This programming assignment does not require any further coding or documentation than what is provided. The program will ABEND and your task is to learn how to investigate what happened so that you can debug your own Assembler programs in the future. To begin, run the following program on the Marist mainframe using the ASSIST JCL used previously. Be sure you type it or copy it EXACTLY as shown below:

```
DUMPEX1  CSECT
         USING DUMPEX1,15   ESTABLISH REG 15 AS BASE REG
*
         SR    7,7          CLEAR REG 7
*
         LA    5,VAR1       LOAD ADDRESS OF VAR1 INTO REG 5
         LA    6,VAR2       LOAD ADDRESS OF VAR2 INTO REG 6
*
         A     7,0(,5)      ADD VAR1 TO REG 3
         A     7,0(,6)      ADD VAR2 TO REG 3
*
         LA    8,ADDED      LOAD ADDRESS OF ADDED INTO REG 8
         ST    7,0(,8)      STORE THE SUM OF VAR1 & VAR2 AT ADDED
*
         XDUMP ,            DUMP REGS
*
         LTORG
*
ADDED    DC    F'0'         FULLWORD OF ZERO TO LATER HOLD SUM
*
VAR1     DC    F'1206000000'  VARIOUS FULLWORD INTEGER VALUES
VAR2     DC    F'972460'
VAR3     DC    F'1344335922'
*
         END   DUMPEX1
```

Use the resulting ABEND dump output to answer the questions below. Each is worth 2 points except question 11 which is worth 5 points.

1. Did this error occur (a) while the program was being assembled or (b) when it was being run?

   While it was being executed (an ABEND can only occur during execution).

2. What is the address of the next instruction which would have been executed?

   ___000030_____

3. What is the value of the condition code at the time of the ABEND?

   ___A = B'1010'. The second two bits, the cc, are B'10' which is 2._____

4. What is the length of the instruction that caused the ABEND (number of bytes)?

   The first two bits, the ILC, are B'10' which is 2. 2 * 2 = 4 bytes long.

5. What is the address of the instruction that caused the abend?

   ___000030 - 4 bytes = 00002C_____

6. What type of error occurred (number and name)?

   ___S0C 6 - Specification Exception_____

7. What usually causes this error?

   ___An attempt to load from or store to storage not on a correct boundary.__

8. What does the value in register 3 represent at the time of the ABEND dump?

   ___A branch on condition (BC) instruction._____

9. What instruction needs to be added to fix this ABEND?

   __BCR B'1111',14 or BR 14 placed right before the LTORG._____

10. What does the value stored at location counter value 00002C represent?

    __A store (ST) instruction of reg 2 to 50 bytes (X'32') off of reg 15.____

11. What exactly happened here to cause the ABEND? Be detailed but succinct in your description.

    __Because the BR 14 was missing right before the LTORG, the program continued executing into storage where it happened to find a valid encoded instruction that branched to another valid encoded instruction that actually caused the S0C 6.*_____

    _____

    * From this you should learn that any time you find the abending instruction within storage, the error is probably an issue of an incorrect branching instruction or something that has happened one or more instructions BEFORE the abending instruction.

    While writing this assignment, there had to be enough encoded instruction bytes above the LTORG so that storage area began exactly on a doubleword boundary (DWB). Why so? It's because the LTORG always starts on a DWB. If the last encoded instruction before the LTORG had not ended on the byte just prior to a DWB, there would have been "slack bytes" in between the last instruction and the beginning of the variable SUM. This would have caused a S0C1, an ABEND that would have been far less challenging to solve.