This assignment is in three parts. Each part is worth 25 programming points. WARNING: Do not use a previous semester's solution!

Part A

Write an assembler program that does the following:

- Change the program's name to the left of the `CSECT` statement to `ASSIGN3A`.
- Change the `USING` statement to `USING ASSIGN3A,15`
- Change the program's name to the right of the `END` statement at the end of the program to `ASSIGN3A`.
- Declare a fullword variable in storage with label `NUM1` with the value 173.
- Declare a fullword variable in storage with label `NUM2` with the value 93.
- Subtract `NUM2` from `NUM1`. To do this, load these `NUM`s into two separate registers using the Load instruction (`L`) for each and then use Subtract Register (`SR`).
- Add `NUM1` to `NUM2`. To do this, load the two `NUM`s into registers (a different pair of registers than for the previous bullet point) and use Add Register (`AR`).
- Finally, use an `XDUMP` to dump out just the registers and look at the register values to see your answers and verify that they are correct. Remember that `XDUMP` with or without a comma in column 16 will only dump the registers. The comma in col. 16 is necessary if you want to do line documentation on the line with the `XDUMP`.

Part B

Copy your program from Part A and name it `ASSIGN3B`.

Modify the new assembler program as follows:

- Change the program's name to the left of `CSECT` to `ASSIGN3B`.
- Change the `USING` statement to `USING ASSIGN3B,15`
- Change the program's name to the right of the `END` statement at the end of the program to `ASSIGN3B`.
- Declare two new consecutive fullwords in storage with labels `ANSWER1` and `ANSWER2`.
- Store (`ST`) the `NUM1` answer (`NUM2` subtracted from `NUM1`) into `ANSWER1`.
- Store (`ST`) the `NUM2` answer (`NUM1` added to `NUM2`) into `ANSWER2`.
- `XDUMP` that area of storage (containing `ANSWER1` and `ANSWER2`) using `XDUMP`. You can use a **single** `XDUMP` to display the two fullwords, `ANSWER1` and `ANSWER2`, in storage by using the label followed by the length of the field like this:

  ```
  XDUMP ANSWER1,8    DUMP 8 BYTES BEGINNING AT LABEL ANSWER1
  ```

- Go into the `XDUMP` output and verify your math is correct and stored at the correct locations in memory.

Part C

Now copy your program from Part B and name it `ASSIGN3C`. Modify the new assembler program as follows:

- Change the program's name to the left of `CSECT` to `ASSIGN3C`.
- Change the `USING` statement to `USING ASSIGN3C,15`
- Change the program's name to the right of the `END` statement at the end of the program to `ASSIGN3C`.
- You may not use the label names `NUM1`, `NUM2`, `ANSWER1` and `ANSWER2` in this version of the Part B program.
- Rewrite the instructions referencing the labels using *explicit addressing*. This means on the `L` and `ST`, remove the labels as the `NUM2` operands and fill in the `D(X,B)` addresses of the various fullword variables.
- Remove the labels from storage too! Note that you can get the information to form a proper `D(X,B)` address for the `NUM2` operand of each of the instructions by reviewing your program listing from Part B.
- Be sure you still have the **single** `XDUMP` that dumps the contents of what were `ANSWER1` and `ANSWER2`.

Other Notes

To get started on this assignment, copy the code – from the top to the bottom of the file – of your `ASSIGN1` member in your `ASSIGNS` PDSE and then add and open a new member of that PDSE named `ASSIGN3A`. In the IDz editor, paste into the new member what you copied from `ASSIGN1`. When you have that done, begin editing the new member, `ASSIGN3A`, by removing the following three lines of instructions **but *nothing more than these three lines*** from the code in your new `ASSIGN3A` member:

```
LA    4,13       LOAD 13 INTO REG 4
LA    8,6        LOAD 6 INTO REG 8
AR    4,8        ADD REG 8'S CONTENTS TO REG 4'S
```

Now, begin Part A described above.

Once again, you are encouraged to complete Part A as a PDSE member named `ASSIGN3A` in your own `ASSIGNS` PDSE. Then, for Part B, copy the code from `ASSIGN3A` into a new member named `ASSIGN3B` and, when you are ready, copy the code from `ASSIGN3B` into a new member named `ASSIGN3C`.

Be sure to follow the documentation standards described in *Documentation and Coding Guidelines* found in Course Documents on Blackboard.

Submit your ***three*** output .txt files representing your successful runs of your jobs for Parts A, B and C on Blackboard.