CSCI 465 **Assignment 3** **Fall 2021**

COBOL Compiler, Assembler and Binder
100 points

Overview

This assignment is in two parts, each worth 50 points. The first part consists of assembling and executing an Assembler program and the second consists of compiling and executing a COBOL program.

You will turn in two (2) text files of output from this assignment. One of the text files will be for Part A, the COBOL Compiler part, and the other will be for Part B, the Assembler part. This assignment's objective is to give you more experience writing and running basic JCL jobs on the Marist mainframe.

Note: Although you will not be required to document either of the programs given to you, please review the JCL documentation guidelines in the *Documentation and Coding Guidelines* in Course Documents on Blackboard and ***document your own JCL*** adequately and appropriately. Inadequate documentation will result in deducted points. Be sure to include the double forward slashes by themselves at the bottom of each job file too!

Part A - COBOL (50 points)

For Part A of the assignment, the source code of a simple complete COBOL program is provided for you as `COBOL3.txt`. This file is included in the assignment folder on Blackboard. Use the COBOL Compiler for the first step of the job you are writing.

Using IDz, you can copy and paste the entire COBOL program provided to you as a text document into a new member of your `ASSIGNS` PDSE named `ASSIGN3A`. Once you get the program copied into IDz's editor, you can build the JCL ***around it***. Your task is to create a complete JCL job stream to 1) compile, 2) bind, and 3) fetch and execute this small COBOL program ***as an in-stream program***.

Programming Notes:

- The first step of both this job will be the COBOL Compiler step, the second will be the Binder step, and the third will be a fetch and execute step.

- Use the following COBOL compile parameters on the COBOL compile step:

  `PARM=APOST,REGION=0M` See the course notes for an explanation of these parms.

  These parameters will be necessary ***throughout the semester*** on any COBOL Compiler steps of your JCL.

- Due to some changes this past summer at Marist, the COBOL Compiler step will require a STEPLIB of:

  ```
  //STEPLIB  DD DSN=IGY630.SIGYCOMP,DISP=SHR
  ```

- Write the object module from the COBOL Compiler step to a temporary data set that will be passed to the Binder step. Again, use the characteristics for this temporary data set provided in the COBOL Compiler part of the notes titled 3. COBOL Compiler, Assembler and Binder.

- Write the program object (load module) that comes out of the Binder step to your own PDSE named `LOADLIB` that you created when you ran your Assignment 1 JCL. The name of the program object member should be exactly the same as the name of the program itself.

- Follow the Binder step with a third step that fetches and executes your program object.

- Do not execute either the Binder step or the fetch and execute step unless the return code from the above steps is zero (0).

- The name of the input file is:

  `KC02322.CSCI465.DATAFA21(DATA3)`

  The input `DD` card for this file for the COBOL program must be named `INDATA`.

- The output `DD` card for the COBOL program must be named `RPTDATA`. This output should be written to standard output.

- The output you turn in should have correct output, and all three return codes should be 0000. The return codes are in the center of about lines 6, 7, and 8 of the JES2 Job Log (second printed page). You should see return codes of 0000 for each of the COBOL Compiler, Binder and fetch and execute steps.

Part B - High-Level Assembler (50 points)

For Part B of the assignment, the source code of a simple complete Assembler program is provided for you as `ASSEMBL3.txt`. This file is included in the assignment folder on Blackboard. Use the High-Level Assembler for the first step of the job you are writing.

Once again, using IDz, you can copy and paste the entire program into a new member of your `ASSIGNS` PDSE named `ASSIGN3B`. Once you get the program copied into IDz's editor, you can build the JCL ***around it***. Your task is to create a complete JCL job stream to 1) assemble, 2) bind, and 3) fetch and execute this small Assembly Language program ***as an in-stream program***.

Programming Notes:

- The first step of this job will be the High-Level Assembler step, the second will be the Binder step, and the third will be a fetch and execute step.

- Be sure to add `PARM=ASA` to the `EXEC` card of the Assembler step. This parameter will be necessary ***throughout the semester*** on any Assembler steps of your JCL.

  Note: `PARM=ASA` tells the Assembler to produce the assembly listing using American National Standard (ANSI) printer-control characters. (If `NOASA` is specified the Assembler uses machine printer-control characters. We want the `ASA` version.)

- Write the object module from the Assembler step to a temporary data set that will be passed to the Binder step. Use the characteristics for this temporary data set provided in the Assembler part of the notes titled 3. COBOL Compiler, Assembler and Binder.

- Write the load module that comes out of the Binder step to PDSE named `LOADLIB` that you created when you ran your Assignment 1. The name of the load module member should be exactly the same as the name of the program itself.

- Follow the Binder step with a third that fetches and executes your program object.

- Do not execute either the Binder step or the fetch and execute step unless the return code from all previous step(s) steps is zero (0).

- The name of the input file is the same as that for the COBOL program above:

  `KC02322.CSCI465.DATAFA21(DATA3)`

  The input `DD` card for this file for the Assembler program must be named `INDATA`.

- The output `DD` card for the Assembler program must be named `RPTDATA`. This output should be written to standard output.

- Like in Part A, the output you turn in should have correct output, and all three return codes should be 0000. The return codes are in the center of about lines 6, 7, and 8 of the JES2 Job Log (second printed page). You should see return codes of 0000 for each of the Assembler, Binder and fetch and execute steps.

IMPORTANT

Be sure to follow ALL of the JCL Documentation Standards as described in the *Documentation and Coding Guidelines*. (Hint: Get one of the two jobs working and completely documented first, then use it as a guide for the second part.)

What To Turn In

Use Retrieve Jobs to get your output down from the Marist output queue onto your PC. Submit the **two (2)** files named `ASSIGN3A.txt` and `ASSIGN3B.txt` on Blackboard.