Assignment 4

100 points

Overview

In this assignment, you will write a program that will read integers from a file of 80-byte records, with zero or more numbers per record, and store the integers in a table.

Read the file and process the integers on each line. Store each one in the table. The table may or may not be entirely full. Store the address of the first entry not in use.

To verify the loading of the table, print out the entire table with 5 numbers per line. The last line may not have as many as 5.

After that, print a list of all of the odd integers with 6 numbers per line. (An integer M is odd if the remainder of dividing M by 2 is not 0.) The last line may not have as many as 6 numbers on it so part of it may be blank.

We will use internal subroutines to structure this program. (See the instructions listed below.)

Input

The input to the program will be a file with an unknown number of records. Each record contains zero or more integers, separated by spaces.

Use the following JCL statement to specify the input file:

```
//FT05F001  DD  DSN=KC02314.SPRING19.CSCI360.HW4DATA,DISP=SHR
```

Internal Subroutines

You will need several internal subroutines:

- BUILD is a subroutine that will read the input file and build the table. It will store the address of the last entry in a fullword passed in as a parameter.
- PRINT is a subroutine that will print the entire table (up to the last entry in use).
- ODDS is a subroutine that will print only the odd values in the table.

There are several requirements for using an internal subroutine:

- You need a label with the name of the subroutine, as in:

  ```
  BUILD    DS    0H
  ```

- You need to create a parameter list for the subroutine (a set of consecutive fullwords, each containing the address of a parameter).

  What parameters should each subroutine have? Each of these three subroutines needs to receive two items: the address of the table itself, and the address of a fullword in which we store the address of the first unused entry. List them in that order.

- You need to call the subroutine, as in:

```
LA      1,BPARMS    Parameter list for BUILD
BAL     11,BUILD    Branch to BUILD
```

  Here the BAL instruction will set register 11 = the address of the next instruction after the BAL.

- You need to return from the subroutine, as in:

```
BR      11
```

- You will need to use some registers in each subroutine. When you exit from the subroutine, these registers should have the same values as they had before the subroutine was called. To accomplish this, set up a save area (a set of consecutive fullwords) and save the register values using STM, and later, just before the subroutine ends, restore the original values using LM.

Assorted Requirements and Notes

- The JCL for this assignment is the same as the JCL used in Assignment 3 except for the line given above to provide the data.
- Define the table to hold 56 values. Each entry is one fullword. Initialize entries in the table to the value -12.
- It is possible that the input file might contain more than 56 values. We do not want to over-fill the table. (Nothing will protect us from this.) As you put numbers into the table, count them. If you reach 56, stop, even if there are more numbers in the file.
- The main program mostly consists of calls to subroutines and the definitions of variables.
- In BUILD, you will need two loops. The outer loop reads the records using XREAD and stops when it detects the end of the file. The inner loop processes one record, using XDECI to obtain each integer in turn. Be sure to stop at the end of the record. Each of these should be a *top-driven* loop.
- You will need to blank out your print lines. You can do this with a standard trick called destructive overlap. For a line of length 61, you can do the following:

```
MVI   PLINE+1,C' '
MVC   PLINE+2(59),PLINE+1
```

  Here we are not changing the first byte in PLINE because it is the carriage-control character.

- Use a non-numeric marker at the end of the input storage area to stop the XDECI scan, like this:

```
BUFFER   DS    CL80
         DC    C'*'
```

- You may use register equates if you like. You can read about these in our textbook, page 65. This is not required.
- You may use extended mnemonics such as BH, BL, BNE, etc. for branch instructions. This is not required.
- In this assignment, avoid using labeled constants in storage; use instead LA instructions or literals.
- In PRINT, you will need print lines of at least 61 bytes each (including the carriage-control character). Double-space between lines of numbers. Each list of numbers should start on a new page and have a heading, centered, stating "List of numbers" or "List of even numbers". Leave two lines blank after the heading.

- Your program needs to have adequate documentation. We need a box of documentation for each subroutine, including its name and a statement of what it does, along with a list of what registers have been used and how. We also need line documentation.

- A problem here is how to print 5 numbers on a line. One way to do this is to treat the print line as a table of 5 values of 12 bytes each:
    - Load the address of LINE+1 into a register (pointer into the line).
    - Counter = 0.
    - While the Counter is < 5:
        - Use XDECO to put a number from the table on the line at the address in the register.
        - Advance the pointer into the line by 12.
        - Increment the table pointer.
        - Increment the counter.
    - Use XPRNT to print the line.
    - Use destructive overlap to restore the line to its original condition.

  and then repeat.