INFO3105 Week 2 Part 2

Review

- PDS
  - JCL
  - LOAD
- JCL
  - JOB
  - EXEC
  - DD

Chapter 17 Notes

- Current Mainframe OS we are using in Marist mainframe = z/OS Version 2.3 (Although version 2.4 is available as of Sept 30, 2019).
- Technologies we will use in this course (descriptions on page 509)
  - JES
  - TSO/E
  - ISPF
  - DB2
  - RACF
- Other Concepts to read over
  - Virtual Storage
  - Multiprogramming
  - Spooling
  - Batch programming (this is what we'll do in this course)

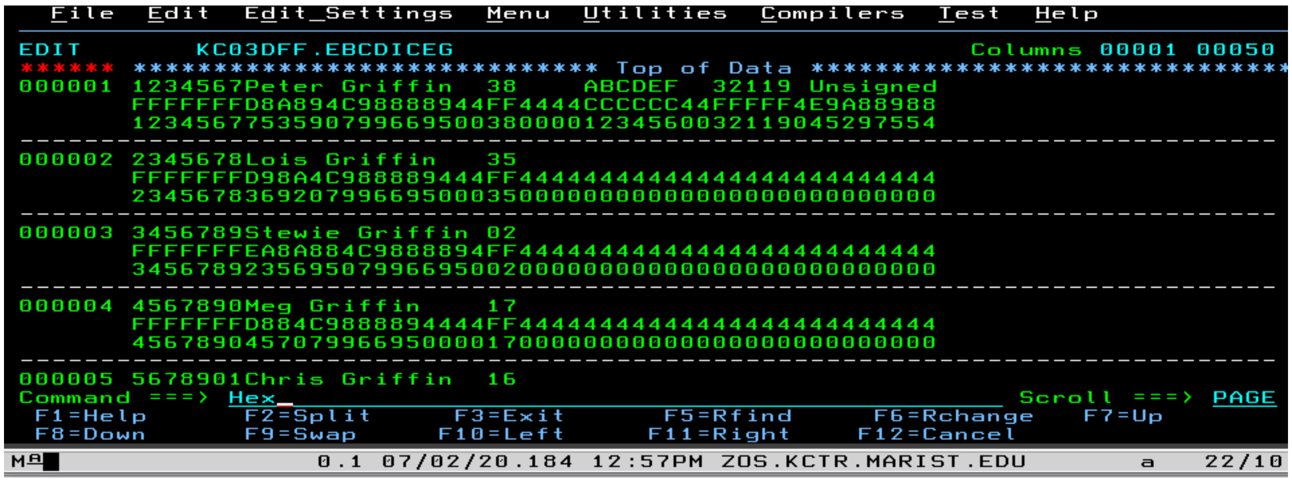
COBOL Data Types

COBOL Data types are **PIC** clause dependent, and come in several broad categories, page 31 of the text shows some of the picture clauses available, below are some more characteristics:

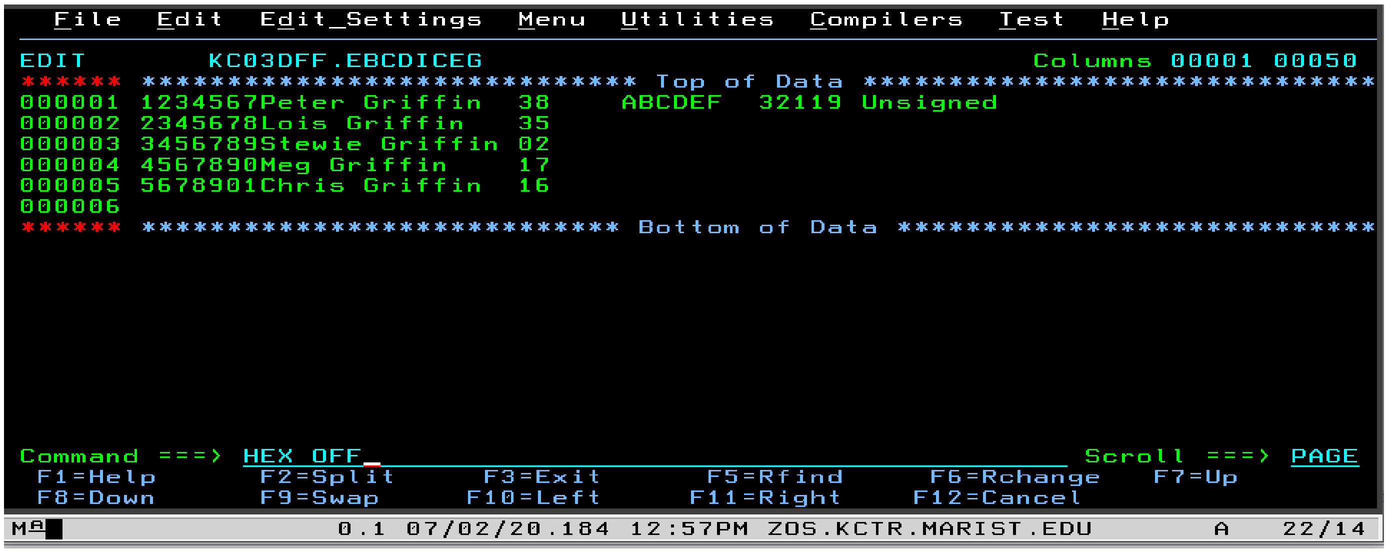
- Character Data:
  - Fixed Length:**PIC X(nn) – or PIC A(nn)** … Note, A ➔ "Alphabetic data"
  - Stored as EBCDIC bytes – examples: **A** ➔ Hex: **C1**, **B** ➔ Hex: **C2**, **9** ➔ Hex: **F9**

- Numeric Data:
  - **Display** numeric **PIC 9(5), PIC S9(5)V99**
  - Stored as EBCDIC bytes, with "assumed" decimal place – one byte per digit
  - Hex values: **F0 ➔ F9**
  - With **V** in declaration – COBOL takes care of decimal alignment in math/**MOVE** operations
  - With **S** (sign) in declaration, the last byte of internal storage holds the sign (C or D):
    - **C** - is positive number:
      - **PIC S9(5)V99 - value: 321.19 ➔ F0 F0 F3 F2 F1 F1 C9**
    - **D** - is negative number
      - **PIC S9(5)V99 - value: 321.19 ➔ F0 F0 F3 F2 F1 F1 D9**
- COMP and COMP-3.
  - You'll see additional picture characteristics with COMP and COMP-3 fields. We'll look into these in an upcoming class.

- Here is an example of the same exact 2 files 1 showing the EBCDIC Hexidecimal representation (ie. How it is stored in HEX) and 1 with HEX OFF :



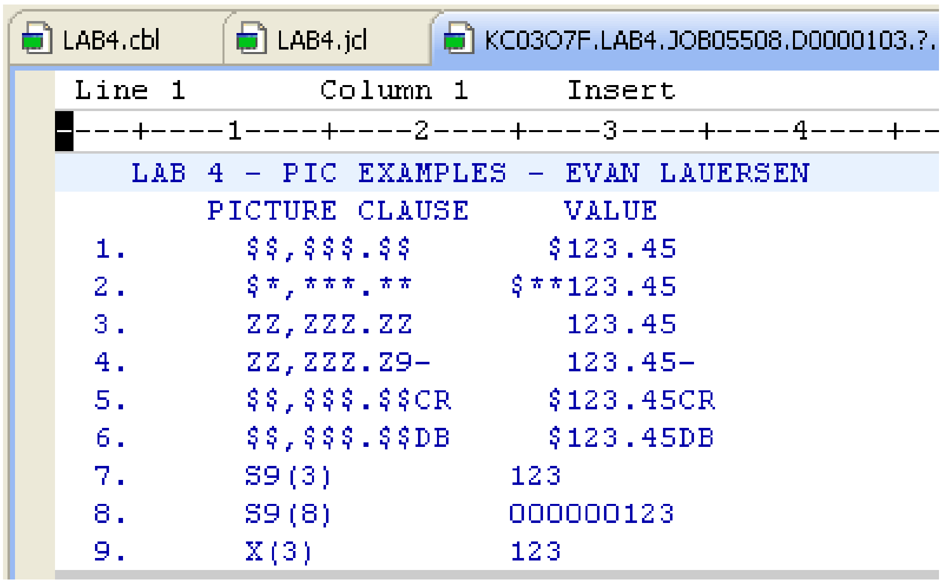
```
   File   Edit   Edit_Settings   Menu   Utilities   Compilers   Test   Help
 EDIT       KC03DFF.EBCDICEG                                Columns 00001 00050
 ****** ***************************** Top of Data ******************************
 000001 1234567Peter Griffin  38     ABCDEF  32119 Unsigned
        FFFFFFFD8A894C98888944FF4444CCCCCC44FFFFF4E9A88988
        12345677535907996695003800001234560032119045297554
 ------------------------------------------------------------------------------
 000002 2345678Lois Griffin   35
        FFFFFFFD98A4C988889444FF44444444444444444444444444
        23456783692079966950003500000000000000000000000000
 ------------------------------------------------------------------------------
 000003 3456789Stewie Griffin 02
        FFFFFFFEA8A884C9888894FF44444444444444444444444444
        34567892356950799669500200000000000000000000000000
 ------------------------------------------------------------------------------
 000004 4567890Meg Griffin    17
        FFFFFFFD884C98888944444FF4444444444444444444444444
        45678904570799669500001700000000000000000000000000
 ------------------------------------------------------------------------------
 000005 5678901Chris Griffin  16
 Command ===> Hex_                                             Scroll ===> PAGE
  F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
  F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
 MA                 0.1 07/02/20.184 12:57PM ZOS.KCTR.MARIST.EDU          a   22/10
```



```
   File   Edit   Edit_Settings   Menu   Utilities   Compilers   Test   Help
 EDIT       KC03DFF.EBCDICEG                                Columns 00001 00050
 ****** ***************************** Top of Data ******************************
 000001 1234567Peter Griffin  38     ABCDEF  32119 Unsigned
 000002 2345678Lois Griffin   35
 000003 3456789Stewie Griffin 02
 000004 4567890Meg Griffin    17
 000005 5678901Chris Griffin  16
 000006
 ****** **************************** Bottom of Data ****************************













 Command ===> HEX OFF_                                         Scroll ===> PAGE
  F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
  F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
 MA                 0.1 07/02/20.184 12:57PM ZOS.KCTR.MARIST.EDU          A   22/14
```

Lab 4 – 2%

- Today you'll create a small program (Lab4.cbl) to exercise some of the picture clauses available in COBOL and shown in chapter 1 of the text. In the report below we have 3 columns using the value of **123.45**. The first column will display a line number, the 2nd column the picture clause we're applying, the 3rd column will display the value once the picture clause has been applied:



```
Line 1          Column 1        Insert
----+----1----+----2----+----3----+----4----+--
    LAB 4 - PIC EXAMPLES - EVAN LAUERSEN
        PICTURE CLAUSE      VALUE
  1.        $$,$$$.$$        $123.45
  2.        $*,***.**      $**123.45
  3.        ZZ,ZZZ.ZZ         123.45
  4.        ZZ,ZZZ.Z9-        123.45-
  5.        $$,$$$.$$CR      $123.45CR
  6.        $$,$$$.$$DB      $123.45DB
  7.        S9(3)          123
  8.        S9(8)          000000123
  9.        X(3)           123
```

- The program uses the following working storage fields to accomplish the task.

```
******************************************************************
*         MISC PIC CLAUSES                                       *
******************************************************************
 01  NUMERIC-EDITED-FIELDS.
     05  DOLLAR-SIGNS       PIC $$,$$$.$$.
     05  ZERO-SUPPRESSED    PIC ZZ,ZZZ.ZZ.
     05  NEGATIVE-VALUE     PIC ZZ,ZZZ.ZZ-.
     05  CREDIT-VALUE       PIC $$,$$$.$$CR.
     05  DEBIT-VALUE        PIC $$,$$$.$$DB.
     05  ASTERISK-VALUE     PIC $*,***.**.

 01 OTHER-FIELDS.
    05  SMALL-ALPHA         PIC X(3)  VALUE SPACES.
    05  SMALL-NUMERIC       PIC S9(3) VALUE ZEROES.
    05  LARGE-NUMERIC       PIC S9(9) VALUE ZEROES.
------------------------------------------------------------------
```

- Your procedure division code will then use a combination of hardcoding and MOVE statements to get the desired output. I have moved 123.45 to each of the fields above and then below is the code used to produce one of the detail lines. You can use this as a template to complete the other 8.

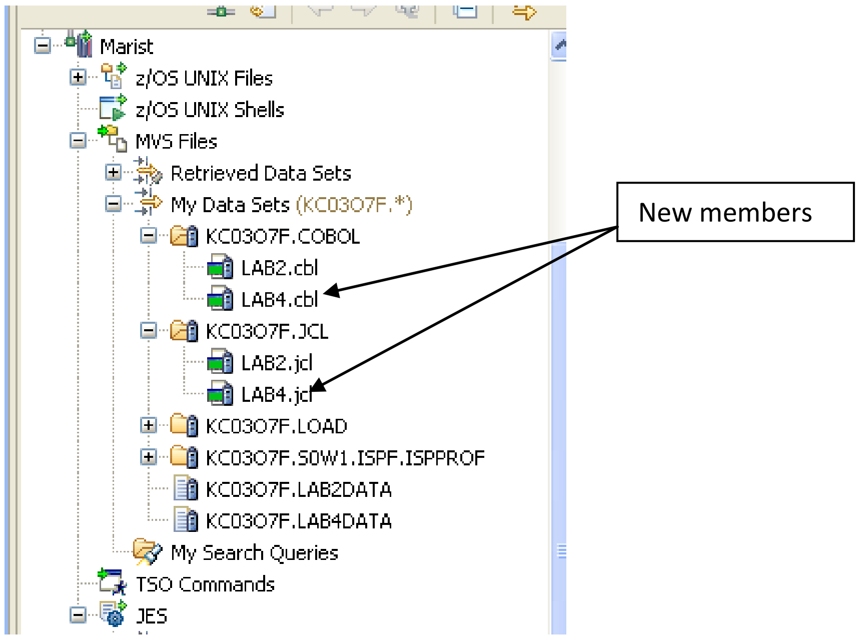
```
MOVE SPACES TO PRNT-CLAUSE.
MOVE '1.' TO PRNT-NUM.
MOVE '$$,$$$.$$' TO PRNT-CLAUSE.
MOVE DOLLAR-SIGNS TO PRNT-VALUE
WRITE PRNT-REC FROM PRNT-DETAIL AFTER ADVANCING 1 LINE.
```

This code outputs the 1st detail line

- And then your output line would be formatted something like this:

```
******************************************************************
*    LAYOUT FOR THE DETAIL LINE OF REPORT PRNTING                *
******************************************************************
 01  PRNT-DETAIL.
     03  FILLER                PIC X(2)      VALUE SPACES.
     03  PRNT-NUM              PIC X(8)      VALUE SPACES.
     03  PRNT-CLAUSE           PIC X(13)     VALUE SPACES.
     03  FILLER                PIC X         VALUE SPACES.
     03  PRNT-VALUE            PIC X(12)     VALUE SPACES.
```

- Using the RDZ environment, copy the lab 2 members over to some new lab 4 counter parts. This will involve setting up a new LAB4 COBOL member and a new LAB4 JCL member, you can use the same LOAD PDS for your machine code:



- You will need to make significant modifications to your LAB2 COBOL code to generate the new output. Note that the LAB4 program does not require an input file this time, just the printer output
- The output uses 3 negative values, so instead of moving 123.45 to the fields, move -123.45 to CREDIT-VALUE, DEBIT-VALUE and NEGATIVE-VALUE.
- To utilize the SMALL-ALPHA field move the string ‘123.45’ to the field or you will receive a compiler error.

<u>Submit to the dropbox:</u>

1. A screen shot of the new printer output (**make sure your name is in the heading**) make sure all 9 of the picture clauses are shown.
2. Lab4 COBOL source code (just do this in a txt file)

Additionally, read Chapter 4 of the text