CSCI 360 **Assignment 8: Packed Decimal**
200 points

This assignment is a continuation of working with packed decimal data and instructions. Albeit very difficult, it is meant to provide you in-depth experience with packed decimal instructions and how packed decimal and zoned decimal numbers are stored and manipulated. Although this program is very different, copy your `ASSIGN7` PDSE and name it `ASSIGN8` to get started and do not remove the standard entry and exit linkage. The logic of this program will be very much like your `ASSIGN7` program you did for Assignment 7 but a little more complicated. Name this program `PAYROLL1`. The input data set is:

```
//FT05F001 DD DSN=KC02322.CSCI360.DATASP24(DATA8),DISP=SHR
```

The first record of the input data set has the following format:

Cols.	Field Name	Range
1-6	Federal Withholding Pct.	000.000% to 100.000%
7-12	State Withholding Pct.	000.000% to 100.000%

Each of the remaining records has the following format:

Cols.	Field Name	Range
1-8	Employee ID	00000000 to 99999999
9-13	Hourly Pay	$000.00 to $999.99
14-18	Hours Worked	000.00 to 999.99
19-23	Deduction Amount	$000.00 to $999.99
24-28	Bonus Amount	$000.00 to $999.99
29-53	Employee Name	alphanumeric
54-80	Unused	spaces

- The input can be seen in the data set `360 Assign 8 Input Data.txt`.
- The exact output can be seen in the data set `360 Assign 8 Exact Output.txt`.
- The Withholding Percentages, Employee ID, Hourly Pay, Hours Worked, Deduction Amount and Bonus Amount are all zoned decimal numbers in the data set.
- The Withholding Percentages have three implied decimal places and the Hourly Pay, Hours Worked, Deduction Amount and Bonus Amount all have two ***implied*** decimal places (dollars and cents for Hourly Pay, Deduction Amount and Bonus Amount, NOT Hours Worked).
- The first record has two zoned decimal numbers in the first 12 bytes. Read this one record and pack both percentages into storage in packed decimal fields defined as shown below in the packed fields provided. The first is the federal withholding rate and the second is the state withholding rate.
- The definition of the remaining input records should be defined in storage as follows. Please copy this into your program's storage and use the names of the fields provided.

```
RECORD   DS    0H
IEMPID   DS    ZL8   →  packs into 5 bytes (8 bytes / 2 rounded = 4 + 1 = 5 bytes)
IHRPAY   DS    ZL5   →  packs into 3 bytes (5 bytes / 2 rounded = 2 + 1 = 3 bytes, etc.)
IHOURS   DS    ZL5   →  packs into 3 bytes
```

```
IDEDUCT  DS    ZL5   →  packs into 3 bytes
IBONUS   DS    ZL5   →  packs into 3 bytes
IEMPNME  DS    CL25
         DS    CL27
```

- Remember that, when you work with packed decimal numbers, it is your responsibility to keep track of where the decimal point is implied in the packed decimal field. Decimal points are ***NEVER*** stored in numeric storage!

  A good example of where this is especially important is the result of multiplication. If you will remember from your elementary or early middle school math, when you multiply two numbers with decimal places, the product has as many decimal places as the number of decimal places of the multiplier ***added to*** the number of decimal places of the number being multiplied. For example, if you multiply a number with four decimal places by one with two, the product will have six decimal places.

  Hint: This becomes important when you multiply the Hourly Pay by the Hours Worked (described next).

- You are going to be multiplying a number that can be as large as `99999F` (3-byte packed hourly pay rate) by a number that can be as large as `99999F` (3-byte packed hours worked). The largest result, or `99999 * 99999`, would be `9999800001F` which fits in a minimum of 6 bytes with a leading 0:

  `09 99 98 00 00 1C` ← the result with four decimal places

  This 6-byte field needs to be defined in storage and will hold the result, i.e., gross pay, when you're done with all of the arithmetic.

  BEFORE you subtract the deduction and add the bonus, you need to round the result of the multiplication from four to two decimal places. Use `SRP` to shift off the two rightmost digits with rounding.

- The calculations are as follows:

  ```
  Gross Pay = Pay Rate * Hours Worked - Deductions + Bonus

  Federal Withholding = Gross Pay * Federal Withholding Percent

  State Withholding = Gross Pay * State Withholding Percent

  Net Pay = Gross Pay - Federal Withholding - State Withholding
  ```

  Note that Gross Pay should remain the same between the calculation for Federal Withholding and State Withholding.

- As you process employee records, accumulate running totals for Gross Pay, Federal Withholding, State Withholding and Net Pay. After the employee records are all processed, the employee counter value and the totals fields should be printed at the top of a new page with headers and a new subheader with 'REPORT TOTALS' in the middle of the print line as shown in the exact output provided. Do not print the column headers or hyphens on this totals page.

- Print the Employee ID in this form with a hyphen between the third and fourth digits: `123-45678`

(just an example). **Be sure that, if the ID has a leading `0` or more, that you print them!** You must use `ED` to print the Employee ID. The Employee ID output field must ***begin in the column immediately following the detail line's carriage control character***.

- Make sure that your output looks as close to the example output as possible. The TA cannot take off points if yours matches the example.

- Any program that `XPRNT`s a blank line will earn 0 points. Always use the carriage control character to double space your lines.

- It is recommended that you get the ID, name, hourly pay and hours worked fully formatted and printing correctly on the print line ***BEFORE*** doing any of the arithmetic and then formatting the gross pay, federal and state withholding and net pay on the print line.

- You ARE allowed to declare and use `DSECT`s in your program but they are not required.

Programming Guidelines

1) Be sure to code a length on all packed decimal operands or you will earn a 0 on your assignment.

2) Print exactly 17 double-spaced detail lines per page.

3) Use the company name, sub-header and column headers shown in the exact output.

4) As before, put the following two lines in your code near the top of your loop, perhaps immediately following your increment of the employee counter which should be the very first thing you do in your loop body:

```
MVI   empldtl+1,C' '
MVC   empldtl+2(131),empldtl+1
```

where *`empldtl`* must be replaced with the actual name of your output employee detail line. These two lines reset your detail line to all spaces so that you are ready to `MVC`, `ED` or `EDMK` data into the detail line prior to printing it.

Do this ***ONLY*** on the detail line and not any other of your print lines.

5) You MUST correctly preset register 1 prior to ALL `EDMK`s. It is not necessary when doing `ED`.

6) When you `XPRNT`, you MUST ALWAYS print exactly 133 bytes.

7) DO NOT `XPRNT` blank lines!

8) You MUST separate out lines of code with asterisks in column 1 to make it readable...REQUIRED! Put lines of code that are related, i.e., the five instructions to `EDMK` a packed field into the print line.

9) You are required to declare each print line with a valid carriage control character in column 1.

10) You must use good clean logic...spaghetti code in Assembler is THE WORST!

11) NEVER put even a single instruction between the `XREAD` and the `BC`. The only instruction allowed in

between is the unconditional branch back to the top of the loop where you check to see if you successfully read another record.

12) You won't be using many but DO NOT clear every register before you use it. Clear a register only when it will be used as a counter or accumulator. You really should only have to use two registers in this program, one is the employee counter and the other is the line counter.

13) To increment a register being used as a counter, use `LA` and ONLY `LA`.

14) You should no longer be using `XDECI`s and `XDECO`s for anything except for maybe using it to check values quickly while building your program. Any program implementing an `XDECI` and/or `XDECO` will earn 0 points.

15) To help you with paging, use a register for your line counter. Set the line counter register to a high value like 99 before the loop begins. Immediately before you `XPRNT` one of the detail lines, check to see if it is time to print new headers. If so, add 1 to the page counter, `ED` it into the output page counter field at the end of the first header line, `XPRNT` it, the subheader, the column headers and the hyphens line. Then, zero out the line counter register. If it's not time, simply branch over printing headers and `XPRNT` the detail line. Be sure to increment the line counter by 1 after `XPRNT`ing the detail line.

16) Be sure you print the averages for total gross pay, total federal withholding, total state withholding and total net pay as shown in the exact output provided.

17) To help you get started and use consistent variable names, here are the ***required*** packed decimal fields. Please copy them into your program and use them:

```
*
* PACKED DECIMAL VARIABLES
*
PFWHPCT  DC    PL4'0'         PACKED FEDERAL WITHHOLDING PERCENTAGE
PSWHPCT  DC    PL4'0'         PACKED STATE WITHHOLDING PERCENTAGE
*
PEMPCTR  DC    PL3'0'         PACKED EMPLOYEE COUNTER (MAX 999)
PPAGECTR DC    PL2'0'         PACKED PAGE COUNTER (MAX 999)
*
PEMPID   DC    PL5'0'         PACKED EMPLOYEE ID
PHRPAY   DC    PL3'0'         PACKED EMPLOYEE HOURLY PAY RATE
PHOURS   DC    PL3'0'         PACKED EMPLOYEE HOURS WORKED
PDEDUCT  DC    PL3'0'         PACKED EMPLOYEE DEDUCTION
PBONUS   DC    PL3'0'         PACKED EMPLOYEE BONUS
PEMPGPAY DC    PL6'0'         PACKED CALCULATED EMPLOYEE GROSS PAY
PFEDWITH DC    PL6'0'         PACKED CALCULATED FEDERAL WITHHOLDING
PSTWITH  DC    PL6'0'         PACKED CALCULATED STATE WITHHOLDING
PEMPNPAY DC    PL6'0'         PACKED CALCULATED EMPLOYEE NET PAY
*
PCALC    DC    PL10'0'        USED TO CALCULATE WITHHOLDING AND AVGS
*
PTGRPAY  DC    PL7'0'         PACKED TOTAL GROSS EMPLOYEE PAY
PTFWITH  DC    PL7'0'         PACKED TOTAL WITHHOLDING
```

```
PTSWITH  DC    PL7'0'
PTNETPAY DC    PL7'0'          PACKED TOTAL NET EMPLOYEE PAY
*
```

Note that the names of the output fields should be the same with the first letter `P` replaced with an `O` (the letter O for output, not zero).

Missing Output After Downloading and Formatting

Those who can see output on Marist but cannot see it or part of it after downloading it to their own laptop or PC to submit it must do the following to fix the problem:

1) Make sure the lengths of ALL output line definitions in storage are exactly 133.

2) Make sure each of them have a valid carriage control of a space (single spacing), a zero (double spacing) or a 1 (top-of-page).

3) Make sure that each edit patterns moved into a receiving field prior to `ED` or `EDMK` is exactly correct with exactly the right number of digit selector.

4) Make sure that there is a `DC` of spaces in between each of the receiving fields defined in each of the output line definitions.

The loss of output data is caused by one or more bad (and probably hidden) characters being found in an output line. You must count carefully and pay attention to details.

Document your program completely and submit your single ASSIGN8.txt file on Blackboard as before.