**CSCI 360 Assignment 6 – Packed Decimal Summer 2020**

**200 points**

This assignment is an introduction to working with packed decimal data and instructions. Albeit difficult, it is meant to illustrate to you the radically different packed decimal instructions and how packed decimal and zoned decimal numbers are stored and manipulated. For help, refer to Blackboard's Lecture Notes & Slides 17. Packed Decimal Instructions and 18. Immediate Byte Instructions.

To start, copy your Assignment 5 PDSE member and name it PAYROLL2. Change the name of the program to PAYROLL2 (don't forget to change the END statement too).

The following pieces of code are known as Standard Entry and Exit Linkage. We will talk about them in more detail very soon. For now, simply replace the CSECT and USING statements with the following eight lines of code:

PAYROLL2 CSECT

STM 14,12,12(13) SAVE REGS IN CALLER'S SAVE AREA

LR 12,15 COPY CSECT ADDRESS INTO R12

USING PAYROLL2,12 ESTABLISH R12 AS THE BASE REG

LA 14,PR2SAVE R14 POINTS TO THIS CSECT'S SAVE AREA

ST 14,8(,13) STORE ADDRESS OF THIS CSECT'S SAVE AREA

ST 13,4(,14) STORE ADDRESS OF THE CALLER'S SAVE AREA

LR 13,14 POINT R13 AT THIS CSECT'S SAVE AREA

Add the following to your storage ***immediately following*** the LTORG:

PR2SAVE DC 18F'-1' CURRENT REGISTER SAVE AREA

Replace the BCR B'1111',14 or BR 14 at the end of your ***executable code*** with the following:

SR 15,15 SET RETURN CODE TO ZERO

L 13,4(,13) POINT R13 TO CALLER'S SAVE AREA

L 14,12(,13) RESTORE R14

LM 0,12,20(13) RESTORE R0 THRU R12

BR 14 RETURN TO CALLER

This program will be very much like your PAYROLL1 program you did for Assignment 5 but the arithmetic will all change. We are going to use packed decimal arithmetic instead of binary arithmetic. The input data set line is:

//FT05F001 DD DSN=KC02322.CSCI360.DATASU20(DATA6),DISP=SHR

The first record has the following format:

|  |  |  |
| --- | --- | --- |
| **Cols.** | **Field Name** | **Range** |
| 1-5 | Withholding Pct. | 000.01% to 100.00% |

Each of the remaining records has the following format:

|  |  |  |
| --- | --- | --- |
| **Cols.** | **Field Name** | **Range** |
| 1-7 | Employee ID | 0000000 to 9999999 |
| 9-13 | Hourly Pay | $000.01 to $999.99 |
| 14-18 | Hours Worked | 000.01 to 999.99 |
| 19-23 | Deduction Amount | $000.01 to $999.99 |
| 24-28 | Bonus Amount | $000.01 to $999.99 |
| 29-53 | Employee Name | alphanumeric |
| 54-80 | Unused | spaces |

* Withholding Percentage, Employee ID, Hourly Pay, Hours Worked, Deduction Amount and Bonus Amount are all zoned decimal numbers in the data set.
* Withholding Percentage, 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 a single zoned decimal number in the first five bytes. Read this one record and pack it into storage in a packed decimal field defined as shown below in the packed fields provided.
* 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 ZL7 🡪 packs into 4 bytes (7 bytes / 2 rounded = 3 + 1 = 4 bytes)

DS CL1

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 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.

* All of the Deductions, Bonuses, Gross Pay, Withholding and Net Pay Amounts should be accumulated and printed at the top of a new page with headers and a new subheader with '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.
* 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 XPRNTs a blank line will earn 0 points. Always use the carriage control character to double space your lines. Note that you will not see double-spaced lines in SDSF. Press F10 to scroll one byte to the left to see the carriage control character and verify that they are correct.
* It is recommended that you get the ID, name, hourly pay, hours worked, deductions and bonus fully formatted and printing correctly on the print line ***BEFORE*** doing any of the arithmetic and then formatting the gross pay, withholding and net pay on the print line.

**Programming Guidelines**

1. Print exactly 18 double-spaced detail lines per page.
2. 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 is the 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.

1. You MUST correctly preset register 1 prior to ALL EDMKs.  It is not necessary when doing ED.
2. When you XPRNT, you MUST ALWAYS print exactly 133 bytes.
3. DO NOT XPRNT blank lines!
4. 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.
5. You are required to declare each print line with a valid carriage control character in column 1.
6. You must use good clean logic...spaghetti code in Assembler is THE WORST!
7. 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.
8. 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.
9. To increment a register being used as a counter, use LA and ONLY LA.
10. You should no longer be using XDECIs and XDECOs for anything except for maybe using it to check values quickly while building your program.

1. 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 XPRNTing the detail line.
2. Be sure you print the averages for total gross pay, total withholding and total net pay as shown in the exact output provided.
3. To help you get started and to use consistent variable names, here are the packed decimal fields as you can copy them into your program to use:

\*

\* PACKED DECIMAL VARIABLES

\*

PWITHPCT DC PL3'0' PACKED WITHHOLDING PCT FROM FIRST REC

\*

PEMPCTR DC PL3'0' PACKED EMPLOYEE COUNTER (MAX 999)

PPAGECTR DC PL2'0' PACKED PAGE COUNTER (MAX 999)

\*

PEMPLID DC PL4'0' 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

PEMPWITH DC PL6'0' PACKED CALCULATED EMPLOYEE WITHHOLDING

PEMPNPAY DC PL6'0' PACKED CALCULATED EMPLOYEE NET PAY

\*

PCALC DC PL9'0' USED TO CALCULATE WITHHOLDING AND AVGS

\*

\* PACKED TOTALS FIELDS

\*

PTDEDUCT DC PL4'0' PACKED TOTAL DEDUCTIONS

PTBONUS DC PL4'0' PACKED TOTAL BONUSES

PTGRPAY DC PL7'0' PACKED TOTAL GROSS EMPLOYEE PAY

PTWITH DC PL7'0' PACKED TOTAL WITHHOLDING

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 in SDSF 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 selectors.
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 .txt file on Blackboard as before.