Test 2

You may use a System/370 Reference Summary ("Yellow Card"). You may use a calculator. You may not collaborate with other students.

Scores:

Section I (15 possible) ______ Section VI (33 possible) ______
Section II (6 possible) ______ Section VII (11 possible) ______
Section III (6 possible) ______
Section IV (15 possible) ______
Section V (14 possible) ______ Total (100 possible) ______

Part I. Encoding/Decoding. (12 points total -- 3 points per blank)

Given the following Assembler instruction or object code, encode or decode as requested.

<table>
<thead>
<tr>
<th>Object Code</th>
<th>ASSEMBLER Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>FA94 9015 8011</td>
<td>______________________</td>
</tr>
<tr>
<td>____________________</td>
<td>MVI 20(5), X'6B'</td>
</tr>
<tr>
<td>982B C080</td>
<td>______________________</td>
</tr>
<tr>
<td>____________________</td>
<td>CLC 16(47,12), 34(10)</td>
</tr>
<tr>
<td>4674 9049</td>
<td>______________________</td>
</tr>
</tbody>
</table>
**Part II. Editing.** (6 points total -- 3 points per blank)

Use the given edit patterns and packed decimal fields contents to determine the resulting edited field contents after the following EDIT instruction has been executed. Give the complete hexadecimal contents of each byte. The results are **not** cumulative. Assume original contents of each field for each question.

```
ED   PATTERN(8),FLD1
```

<table>
<thead>
<tr>
<th>FLD1</th>
<th>Pattern</th>
<th>Edited Result (in hexadecimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00495F</td>
<td>XL8'4020206B20212060'</td>
<td>________________________________</td>
</tr>
<tr>
<td>00381B</td>
<td>XL8'5C2120204B202060'</td>
<td>________________________________</td>
</tr>
</tbody>
</table>

**Part III. SRP Instruction.** (6 points total -- 3 points per blank)

Use the following contents of FLD1 and the given SRP instructions to determine the contents of FLD (in hexadecimal) after the execution of the SRP instructions. Specify any error conditions that might occur.

<table>
<thead>
<tr>
<th>FLD1</th>
<th>Instruction</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3456789E</td>
<td>SRP   FLD1(4),(64-3),5</td>
<td>________________________________</td>
</tr>
<tr>
<td>00048B</td>
<td>SRP   FLD1(3),2,0</td>
<td>________________________________</td>
</tr>
</tbody>
</table>
**Part IV. Result of Instructions.** (15 points total -- 3 points per blank)

Given the following field definitions, perform the indicated packed decimal operations. Give the complete contents of each byte of the receiving field (in hexadecimal) after the execution of the instruction. The instructions are not cumulative. Assume original contents of each field for each question.

NUM1 DC X'30310C'
NUM2 DC X'30350B'
NUM3 DC X'0000006008C'
NUM4 DC X'310D'
NUM5 DC X'000030B'
NUM6 DC X'4F3D6C'

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZAP NUM5(4),NUM2(3)</td>
<td>____________________________</td>
</tr>
<tr>
<td>SP NUM1(3),NUM2(3)</td>
<td>____________________________</td>
</tr>
<tr>
<td>DP NUM3(6),NUM5(2)</td>
<td>____________________________</td>
</tr>
<tr>
<td>PACK NUM2(3),NUM6(3)</td>
<td>____________________________</td>
</tr>
<tr>
<td>CP NUM4(3),NUM1(2)</td>
<td>Condition Code _____________</td>
</tr>
</tbody>
</table>
Part V. Dump Interpretation. (14 points -- 2 points for each answer)

Answer the following questions using the PSW AT ABEND, register contents, and memory contents given below. Give 'address' answers in hexadecimal.

PSW AT ABEND FFC5000B D000036
R8 = F4F4F4F4 R14 = FFFE7960 R15 = 0000000

1. What is the address of the instruction that would have been executed next had the program not ABENDed on the current instruction?

2. What is the length (in bytes) of the instruction that caused the ABEND?

3. What is the address of the instruction that caused the ABEND?

4. What is the name of the program interrupt that occurred?

5. Write the object code for the instruction that caused the ABEND.

6. Decode the instruction that caused the ABEND.

7. What is the value of the condition code?
Part VI. Short Coding. (33 points -- 3 points each)

Write assembly instruction(s) to perform the following tasks. Use literals wherever possible (unless otherwise specified).

1. (a) Write code that will store registers 8 through 1 in storage area SAVE. Define SAVE to be the exact size needed.

   (b) Write code that will restore all of the above registers except for registers 12 and 13.

2. Write code that will compare the contents of the storage locations ALPHA DS PL7 and BETA DS PL10 and branch to the label PROBLEM if ALPHA and BETA are equal. You may assume the label PROBLEM already exists somewhere.

3. Write a parameter list that will provide a subroutine access to storage areas labeled MAYNOT, CANNOT and WILLNOT, in that order.

4. Write code that will call the internal subroutine FUNC and place the return address in register 11.
Part VI continued

5. Write code that will set the contents of each byte in the 85-byte field REGION to C'+'.

6. Write code that will copy the contents of the 10-byte packed decimal field SOURCE into the 12-byte packed decimal field TARGET.

7. Without using a literal, increment the value in register 7 by 1 with a single line of code.

For questions 8 through 10, assume that VALU is:

\[
\text{VALU} \quad \text{DC} \quad \text{XL4'0020443B'} \quad \text{Initial Value}
\]

Write a single instruction to change VALU to the given new value. The instruction may refer to a literal unless otherwise indicated. The ZAP instruction may not be used. The changes are not cumulative.

8. X'9820443B'

9. X'0000205B' without using a literal

10. X'010221F1B' without using an immediate byte
**Part VII. Long Coding.** (11 points)

Using the storage areas that are defined below:

- **TABLE** DS 60PL3 Table of numbers
- **EOT** DS A Address of the logical end of table

and assume that values have been properly placed into the table and that EOT has been properly set as well. Write the code for a loop that will display the values that are in the table. For each entry, print one triple-spaced line containing one number. Suppress leading zeroes and print a minus sign to the right of a number if it is negative.

Define any additional storage that is needed.

This does not have to be a complete program or subroutine. Write only what is needed. You do not need to document the code.