**AP** is an SS₂ instruction which is used to add packed decimal fields. Operand 1 is a field in storage which should contain a packed decimal number. The resulting sum develops in this field. The contents of Operand 2, another packed decimal field in storage, is added to the contents of Operand 1 to produce the sum which is stored in Operand 1. The operands are limited to a maximum length of 16 bytes and may have different sizes since this is a SS₂ instruction.

A decimal overflow (condition code = 3) can occur if the generated sum loses a significant digit when it is placed in the target field. A decimal overflow may or may not cause a program interruption (abend). This depends on the setting of a bit in the PSW (See SPM). Otherwise, the condition code is set to indicate whether the result was zero (condition code = 0), negative (condition code = 1), or positive (condition code = 2). You can test these conditions with BZ or BNZ, BM or BNM, and BP or BNP.

Consider the following AP example.

```
AP      APK, BPK
BP      APOSITIV
```

Assume the variables are initially in the following states,

```
APK   DC   PL4'34'  = X'0000034C'
BPK   DC   PL3'22'  = X'00022C'
```

After the AP instruction has executed, the variables have the following values.

```
APK  = X'0000056C'
BPK  = X'00022C'
```

The contents of BPK and APK were added and the result stored in APK. BPK was unaffected by the add operation. The branch would occur and execution would resume at “APPOSITIV” since the condition code was set to positive.

On the other hand, consider the following example,

```
AP      APK, BPK
...
APK   DC   PL2'999'  = X'999C'
BPK   DC   PL2'3'    = X'003C'
```

After the AP instruction has executed, the variables have the following values.

```
APK  = X'002C'
```
BPK = X'003C'

Notice that an overflow occurred in APK with the loss of a significant digit since the APK field was too short to hold the resulting sum.

Examples

Some unrelated AP's:

<table>
<thead>
<tr>
<th>A</th>
<th>DC</th>
<th>P'12345'</th>
<th>X'12345C'</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>DC</td>
<td>P'-32'</td>
<td>X'032D'</td>
</tr>
<tr>
<td>C</td>
<td>DC</td>
<td>Z'11'</td>
<td>X'F1C1'</td>
</tr>
</tbody>
</table>

... Results:

<table>
<thead>
<tr>
<th>AP</th>
<th>A,=P'20'</th>
<th>A = X'12365C'</th>
<th>C.C. = HIGH</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>B,=P'20'</td>
<td>B = X'012D'</td>
<td>C.C. = LOW</td>
</tr>
<tr>
<td>AP</td>
<td>B,=P'999'</td>
<td>B = X'967C'</td>
<td>C.C. = HIGH</td>
</tr>
<tr>
<td>AP</td>
<td>A,=P'-100'</td>
<td>A = X'12245C'</td>
<td>C.C. = HIGH</td>
</tr>
<tr>
<td>AP</td>
<td>A,B</td>
<td>A = X'12313C'</td>
<td>C.C. = HIGH</td>
</tr>
<tr>
<td>AP</td>
<td>B,B</td>
<td>B = X'064D'</td>
<td>A FIELD CAN BE ADDED TO ITSELF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C.C. = LOW</td>
</tr>
<tr>
<td>AP</td>
<td>B,=P'32'</td>
<td>B = X'000C'</td>
<td>C.C. = EQUAL</td>
</tr>
<tr>
<td>AP</td>
<td>B,A</td>
<td>B = X'313C'</td>
<td>AN OVERFLOW OCCURS</td>
</tr>
<tr>
<td>AP</td>
<td>A,C</td>
<td>DATA EXCEPTION - C NOT PACKED</td>
<td></td>
</tr>
</tbody>
</table>

Tips

1. You must be familiar with your data. The best way to prevent overflows is to plan the size of your fields based on the data at hand. There is a rule of thumb that you can follow for additions: If you are adding two packed fields with m and n bytes, then the sum might be as large as max(m, n) + 1 bytes. You may need to construct a work area to handle the maximum values. For instance,

   FIELD A DS PL4
   FIELDB DS PL3
   WORK DS PL5

In planning to add FIELD A and FIELDB, we construct a work field called "WORK". The following code completes the task.

   ZAP WORK,FIELDA
   AP WORK,FIELDB