FA	L_1L_2	B_1D_1	D_1D_1	B_2D_2	D_2D_2

AP is an SS_2 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 an SS_2 instruction and the length of both operands is stored in the instruction as L_1 and L_2.

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 "APOSITIV" since the condition code was set to positive.

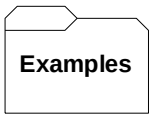
On the other hand, consider the following example,

```
          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.



Some unrelated AP's:

```
A       DC      P'12345'      = X'12345C'
B       DC      P'-32'        = X'032D'
C       DC      Z'11'         = X'F1C1'
        ...
                              Results:
        AP    A,=P'20'    A = X'12365C'  C.C. = HIGH
        AP    B,=P'20'    B = X'012D'    C.C. = LOW
        AP    B,=P'999'   B = X'967C'    C.C. = HIGH
        AP    A,=P'-100'  A = X'12245C'  C.C. = HIGH
        AP    A,B         A = X'12313C'  C.C. = HIGH
        AP    B,B         B = X'064D'    A FIELD ADDED TO ITSELF
                                         C.C. = LOW
        AP    B,=P'32'    B = X'000C'    C.C. = EQUAL
        AP    B,A         B = X'313C'    AN OVERFLOW OCCURS
        AP    A,C                        DATA EXCEPTION - C NOT PACKED
```

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,

```
FIELDA   DS    PL4
FIELDB   DS    PL3
WORK     DS    PL5
```

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

```
ZAP    WORK,FIELDA
AP     WORK,FIELDB
```