AAA (ASCII adjust after addition)

The ASCII addition procedure below is used to add ASCII decimal values with implied decimal points.

It works by iterating through the two operands, adding each corresponding digit and carrying over any carry from the previous iteration.

The procedure uses the AAA instruction to adjust the sum after each addition to ensure that it is in a valid ASCII decimal format.

Here is a more detailed explanation of the procedure:

```
36 mov esi, SIZEOF decimal_one - 1
37 mov edi, SIZEOF decimal_one
38 mov ecx, SIZEOF decimal_one
39 mov bh, 0
40
41 L1:
42 mov ah, 0
43 mov al, decimal_one[esi]
44 add al, bh
45 aaa
46 mov bh, ah
47 or bh, 30h
48 add al, decimal_two[esi]
49 aaa
50 or bh, ah
51 or bh, 30h
52 or al, 30h
53 mov sum[edi], al
54 dec esi
55 dec edi
56 loop L1
57
58 mov sum[edi], bh
```

The esi register points to the last digit position of the first operand, and the edi register points to the last digit position of the sum. The ecx register contains the length of the first operand.

The loop at L1 iterates through the two operands, adding each corresponding digit and carrying over any carry from the previous iteration.

The AAA instruction is used to adjust the sum after each addition to ensure that it is in a valid ASCII decimal format.

The carry digit is saved in the bh register and then converted to ASCII. The ASCII carry digit is then added to the sum.

After the loop has finished iterating, the last carry digit is saved in the sum.

The following example shows how to use the ASCII addition procedure to add the numbers **1001234567.89765** and **9004020765.02015:**

```
63 mov
64 esi,OFFSET decimal_one
65 mov
66 edi,OFFSET sum
67 mov
68 ecx,SIZEOF decimal_one
69
70 call
71 ASCII_add
72
73 mov
74 edx,OFFSET sum
75 call
76 WriteString
77 call
78 Crlf
```

This code will produce the following output:

1000525533291780.

As you can see, the ASCII addition procedure correctly adds the two numbers, even though they have implied decimal points.