Key to Practical 1
First Steps in 68000 Assembly Language

Step 5

1. Without using the assembler and the debugger, determine the result of the following additions as well as the values of the **N**, **Z**, **V** and **C** flags.
   - 8-bit addition: `$B4 + $4C`
     `$B4 + $4C = $`**`100`** (the 8-bit result is `$`**`00`**.)
     N = 0, Z = 1, V = 0 et C = 1

   - 16-bit addition: `$B4 + $4C`
     `$00B4 + $004C = $`**`0100`**
     N = 0, Z = 0, V = 0 et C = 0

   - 16-bit addition: `$4AC9 + $D841`
     `$ 4AC9 + $D841 = $1`**`230A`** (the 16-bit result is `$`**`230A`**.)
     N = 0, Z = 0, V = 0 et C = 1

   - 32-bit addition: `$FFFFFFFF + $00000015`
     `$FFFFFFFF + $00000015 = $1`**`00000014`** (the 32-bit result is `$`**`00000014`**.)
     N = 0, Z = 0, V = 0 et C = 1

- **N** = 1, if the most significant bit of the result is one.
- **Z** = 1, if the result equals zero.
- **C** = 1, if a carry occurs (assuming that the numbers are unsigned).
- **V** = 1, if an arithmetic overflow occurs (assuming that the numbers are signed).
  To determine the value of **V** for an addition, perform the addition assuming that the numbers and the result are signed. Then **V** = 1, if one of the two conditions below is met:
  - The sum of two positive numbers is negative.
  - The sum of two negative numbers is positive.

2. Use the debugger to check your answers. To do so, write a program that performs the four additions above. Assemble it, run it, check the results and the values of the flag.

   - There are many possibilities, you can find one of them below.
   - Execute the code step by step and check your answers.

```
            org     $4

Vector_001  dc.l    Main

            org     $500

Main        ; 8-bit addition.
            move.b  #$b4,d0
            move.b  #$4c,d1
            add.b   d0,d1

            ; 16-bit addition.
            move.w  #$b4,d0
            move.w  #$4c,d1
            add.w   d0,d1

            ; 16-bit addition.
            move.w  #$4ac9,d0
            move.w  #$d841,d1
            add.w   d0,d1

            ; 32-bit addition.
            move.l  #$ffffffff,d0
            move.l  #$15,d1
            add.l   d0,d1
```

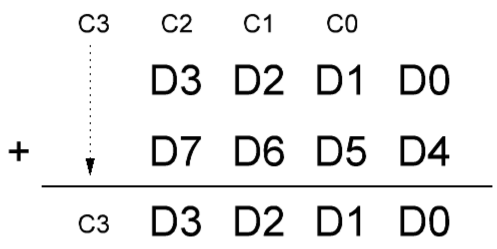
Step 6

Write a program that performs a 128-bit addition.

Inputs : **D3:D2:D1:D0** = 128-bit integer (**D0** contains the 32 least significant bits).

D7:D6:D5:D4 = 128-bit integer (**D4** contains the 32 least significant bits).

Output : **D3:D2:D1:D0** = **D3:D2:D1:D0** + **D7:D6:D5:D4**



```
            add.l   d4,d0         ; D4 + D0     -> D0, C0 -> X
            addx.l  d5,d1         ; D5 + D1 + X -> D1, C1 -> X
            addx.l  d6,d2         ; D6 + D2 + X -> D2, C2 -> X
            addx.l  d7,d3         ; D7 + D3 + X -> D3, C3 -> X
```

Step 7

Write a few rotate instructions that modify **D1** so that it takes the values below. For each case, the initial value of **D1** is `$76543210`.

- **D1** = `$76543120`

```
                                        ; D1 = $ 7654 3210
            ror.w   #4,d1               ; D1 = $ 7654 0321
            ror.b   #4,d1               ; D1 = $ 7654 0312
            rol.w   #4,d1               ; D1 = $ 7654 3120
```

- **D1** = `$75640213`

```
                                        ; D1 = $ 7654 3210
            rol.w   #4,d1               ; D1 = $ 7654 2103
            ror.b   #4,d1               ; D1 = $ 7654 2130
            ror.w   #4,d1               ; D1 = $ 7654 0213
            swap    d1                  ; D1 = $ 0213 7654
            ror.w   #4,d1               ; D1 = $ 0213 4765
            ror.b   #4,d1               ; D1 = $ 0213 4756
            rol.w   #4,d1               ; D1 = $ 0213 7564
            swap    d1                  ; D1 = $ 7564 0213
```

- **D1** = `$54231067`

```
                                        ; D1 = $ 7654 3210
            ror.l   #8,d1               ; D1 = $ 1076 5432
            ror.b   #4,d1               ; D1 = $ 1076 5423
            swap    d1                  ; D1 = $ 5423 1076
            ror.b   #4,d1               ; D1 = $ 5423 1067
```

- **D1** = `$05634127`

```
                                        ; D1 = $ 7654 3210
            ror.l   #4,d1               ; D1 = $ 0765 4321
            ror.b   #4,d1               ; D1 = $ 0765 4312
            ror.l   #8,d1               ; D1 = $ 1207 6543
            ror.b   #4,d1               ; D1 = $ 1207 6534
            ror.l   #8,d1               ; D1 = $ 3412 0765
            ror.b   #4,d1               ; D1 = $ 3412 0756
            ror.l   #8,d1               ; D1 = $ 5634 1207
            ror.b   #4,d1               ; D1 = $ 5634 1270
            ror.l   #4,d1               ; D1 = $ 0563 4127
```