Running an Example Vector Assembly Code

***Note: This is meant to be a follow-up on the RISC-V Vector Toolchain document and we will be using version 0.9 of the vector extension along with the corresponding tools.**

In the RISC-V “V” vector extension (v0.9) manual, there are a few vector assembly code examples given in Appendix A. One of these examples is “vector-vector add example” (whose code is not shown in the manual and exists in the riscv-v-spec repository under the examples folder). We will be using this code to test our installed toolchain and the spike simulator.

The code given in the file “vvaddint32.s” is illustrated below:

```
    .text
    .balign 4
    .global vvaddint32
    # vector-vector add routine of 32-bit integers
    # void vvaddint32(size_t n, const int*x, const int*y, int*z)
    # { for (size_t i=0; i<n; i++) { z[i]=x[i]+y[i]; } }
    #
    # a0 = n, a1 = x, a2 = y, a3 = z
    # Non-vector instructions are indented
vvaddint32:
    vsetvli t0, a0, e32, ta,ma  # Set vector length based on 32-bit vectors
    vle32.v v0, (a1)            # Get first vector
      sub a0, a0, t0            # Decrement number done
      slli t0, t0, 2            # Multiply number done by 4 bytes
      add a1, a1, t0            # Bump pointer
    vle32.v v1, (a2)            # Get second vector
      add a2, a2, t0            # Bump pointer
    vadd.vv v2, v0, v1          # Sum vectors
    vse32.v v2, (a3)            # Store result
      add a3, a3, t0            # Bump pointer
      bnez a0, vvaddint32       # Loop back
      ret                       # Finished
```

Note that in the comments, that have the prefix ‘#’ before them, the equivalent C function for this code is given. So this is only the vector addition function written in RISC-V assembly. Now we need to create a main function that will call this function on two stored vectors and will save the result in another vector. This can be done in two ways: writing a main function in assembly or writing it as a C program. For the purpose of simplicity, I have used C language to write the main function in this tutorial.

The main function “main.c” is illustrated below:

```c
#include <stddef.h>
#include <stdio.h>

void *vvaddint32(size_t n, const int*x, const int*y, int*z);
//{for(size_t i=0; i<n; i++){z[i]=x[i]+y[i];}}

static const int x1[] = {1, 4, 7, 24, 51, 63, 86, 125};    // first vector
static const int y1[] = {3, 6, 0, 42, 77, 29, 54, 120};    // second vector

int main()
{
      int vlen = sizeof(x1)/sizeof(int);

      printf("\n############################################################");
      printf("\nProgram start:\n");

      printf("\nFirst vector:\n");
      for(int i=0; i<vlen; i++){printf("%d\t", x1[i]);}

      printf("\n\nSecond vector:\n");
      for(int i=0; i<vlen; i++){printf("%d\t", y1[i]);}

      int z1[sizeof(x1)/sizeof(int)] = {0};
      vvaddint32(vlen, x1, y1, z1);                        // function call

      printf("\n\nResultant vector:\n");
      for(int i=0; i<vlen; i++){printf("%d\t", z1[i]);}

      printf("\n\nProgram finished!\n");
      printf("############################################################\n\n");
      return 0;
}
```

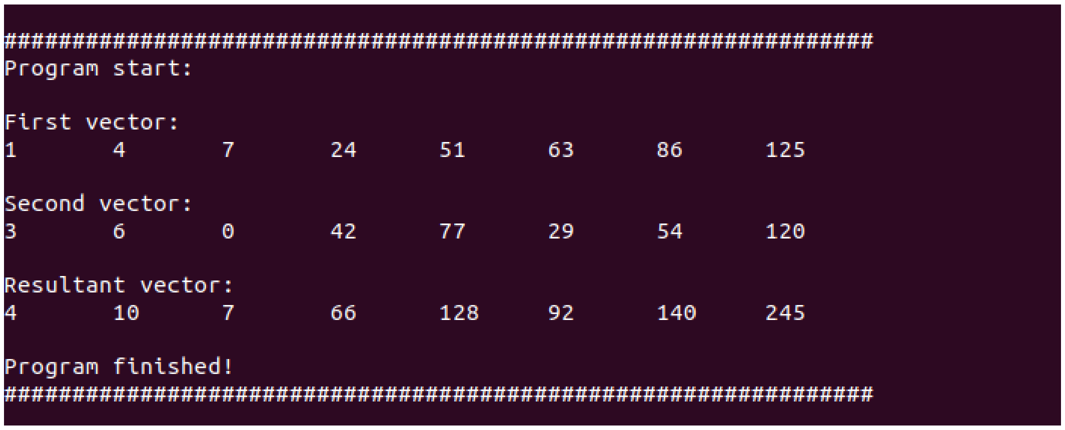
Now open terminal in the location of these two files and run these commands one-by-one:

```
$HOME/riscvv/riscvv09-tools/gnu/bin/riscv64-unknown-elf-as -march=rv64gcv \
-o vvaddint32.o vvaddint32.s

$HOME/riscvv/riscvv09-tools/gnu/bin/riscv64-unknown-elf-gcc -Wall \
main.c -o main vvaddint32.o

$HOME/riscvv/riscvv09-tools/spike/bin/spike --isa=RV64gcV \
$HOME/riscvv/riscvv09-tools/pk/riscv64-unknown-elf/bin/pk main
```

The output of the last command is shown in the screenshot below:



```
################################################################
Program start:

First vector:
1       4       7       24      51      63      86      125

Second vector:
3       6       0       42      77      29      54      120

Resultant vector:
4       10      7       66      128     92      140     245

Program finished!
################################################################
```

Illustration 1: Output of main function

References:

1. RISC-V Vector Spec – Github Repository:
   https://github.com/riscv/riscv-v-spec

2. Programming with RISC-V Vector Instructions by Georg Sauthoff:
   https://gms.tf/riscv-vector.html