Assignment #1

Low-level Programming

Due Date:	as specified on Moodle
Topics covered:	Assembly Programming
Deliverables:	One file: `square-root.s`.
Objectives:	Students should learn some basic assembly programming.

Programming Statement

Students are expected to implement a square root calculation library with variable arguments type. You should ensure the submission is compilable with the test cases on the moodle under vpl/Linux (x86-64) environment. Please take note the requirement for the file name and start header as specified in the syllabus.

Assembly Programming - Square Root Calculation

An interesting observation made long ago give us a simple way to calculate square roots. That is: the square root of an integer is equal to the number of successively higher odd numbers that can be subtracted from it. Below are some examples of this:

$$1 + 3 + 5 + 7 + 9 = 25 \quad — \quad 5^2$$

$$1 + 3 + 5 + 7 + 9 + 11 = 36 \quad — \quad 6^2$$

$$1 + 3 + 5 + 7 + 9 + 11 + 13 + 15 + 17 = 81 \quad — \quad 9^2$$

This implies that we can determine the root of a given number by subtracting increasing odd numbers from it. The following procedure shows how the integer square root (rounded down) of 30 can be determined through such a process.

```
30-1=29    partial square root = 1
29-3=26    partial square root = 2
26-5=21    partial square root = 3
21-7=14    partial square root = 4
14-9=5     partial square root = 5
5 < 9      Stop. Nearest square root is 5.
```

Write two functions (in assembly!) that performs the square root calculation with the following C function prototypes:

```
void sq_root_compute_array(int num_of_elements, unsigned int *array_of_elements);
void sq_root_compute_varargs(unsigned int, ...);
```

The only difference between the two prototypes is that in the former all the arguments are passed in an array and an intger while in the latter, by comma-separated arguments. Both functions essentially perform the same thing: use the above algorithm to compute the nearest rounded down integer roots of the given arguments.

The following two function calls should result in exactly the same printout:

```
int a[]={ 25, 169, 9810};
sq_root_compute_array(3, a);
sq_root_compute_varargs(25, 169, 9810, 0);
```

In the variadic prototype, we use 0 to indicate that there are no more arguments. The printout for the above should be (and they should print the same):

```
Square root of 25 is 5.
Square root of 169 is 13.
Square root of 9810 is 99.
```

Write the assembly code for both functions in an assembly file named `square-root.s`. `square-root.s` should be compilable using the command `gcc -std=c99 -pedantic -Wall -Wextra -Wno-deprecated -fno-stack-protector -no-pie -o vpl_execution main.c square-root.s driver-sample.c`. `main.c` and `driver-sample.c` are provided in vpl. Your submitted assembly code should be commented sufficiently. No comments result in zero mark.

Rubrics

Please do not use the instructions that are not taught in the class. Pass the ten tests.