



## Learn, Share, Build

Each month, over 50 million developers come to Stack Overflow to learn, share their knowledge, and build their careers.

Join the world's largest developer community.

Sign Up

## OpenCL - is it possible to invoke another function from within a kernel?



I am following along with a tutorial located here: <http://openc1.codeplex.com/wikipage?title=OpenCL%20Tutorials%20-%201>

The kernel they have listed is this, which computes the sum of two numbers and stores it in the output variable:

```
__kernel void vector_add_gpu (__global const float* src_a,
                             __global const float* src_b,
                             __global float* res,
                             const int num)
{
    /* get_global_id(0) returns the ID of the thread in execution.
    As many threads are launched at the same time, executing the same kernel,
    each one will receive a different ID, and consequently perform a different
    computation.*/
    const int idx = get_global_id(0);
```

```
/* Now each work-item asks itself: "is my ID inside the vector's range?"
If the answer is YES, the work-item performs the corresponding computation*/
if (idx < num)
    res[idx] = src_a[idx] + src_b[idx];
}
```

1) Say for example that the operation performed was much more complex than a summation - something that warrants its own function. Let's call it `ComplexOp(in1, in2, out)`. How would I go about implementing this function such that `vector_add_gpu()` can call and use it? Can you give example code?

2) Now let's take the example to the extreme, and I now want to call a generic function that operates on the two numbers. How would I set it up so that the kernel can be passed a pointer to this function and call it as necessary?

[opencl](#)

asked Aug 25 '11 at 20:07



[Adam S](#)

**3,366** 12 43 83

- 
- 1 Just a comment. This is OpenCL not CUDA. You are not forced to use multiples of workgroup sizes. I see very often those ugly 'if (idx < num)'. That is not needed in OpenCL since the workgroup size can be perfectly defined to fit the needs of the data to process. – [DarkZeros](#) Aug 2 '13 at 12:10
- 

## 2 Answers

---

Yes it is possible. You just have to remember that OpenCL is based on C99 with some caveats. You can create other functions either inside of the same kernel file or in a separate file and just include it in the beginning. Auxiliary functions do not need to be declared as inline however, keep in mind that OpenCL will inline the functions when called. Pointers are also not available to use when calling auxiliary functions.

### Example

```
float4 hit(float4 ray_p0, float4 ray_p1, float4 tri_v1, float4 tri_v2, float4
tri_v3)
{
    //logic to detect if the ray intersects a triangle
}
```

```
__kernel void detection(__global float4* trilist, float4 ray_p0, float4 ray_p1)
{
    int gid = get_global_id(0);
    float4 hitlocation = hit(ray_p0, ray_p1, trilist[3*gid], trilist[3*gid+1],
    trilist[3*gid+2]);
}
```

answered Aug 26 '11 at 1:29

[Edward Carmack](#)


151 3

---


What do you mean by 'Pointers are also not available to use when calling auxiliary functions?' – [Nigel](#) Aug 29 '11 at 15:09

---

Within an opencl kernel pointers cannot be used anywhere in a kernel or when calling a function from a kernel. Pointers are one of those caveats. One reason you do not need pointer passing is that the functions are always inlined instead of handing off to a separate memory address. – [Edward Carmack](#) Nov 8 '11 at 13:49



Love remote work?  
Find it on a new kind of career site



Get started

You can have auxiliary functions for use in the kernel, see [OpenCL user defined inline functions](#) . You can not pass function pointers into the kernel.

edited May 23 at 11:54



Community ♦

1 1

answered Aug 25 '11 at 20:11

[Adam S.](#)

668 1 8 16

---

3 I like that your name is also Adam S – [Adam S](#) Aug 25 '11 at 20:18