Slide 1

Administrivia

- Homework 2 on the Web. Due next Wednesday. Exact requirements for OpenCL program to be added, but you should be able to do the other three now.
- Notice that reading assignments have been modified — you should read the updated versions of the appendices (though if you read the ones in the book, much will be review).

Slide 2

GPGPU

- Recall from overview/introduction that the SIMD (Single Instruction, Multiple Data) model was popular in the relatively early days of parallel programming, fell of favor, and is now making a comeback as “GPGPU” (General-Purpose computing on Graphics Processing Units).
- Typically SIMD is a good fit for GPU hardware — but it’s worth noting that they usually(?) have their own memory, not shared with “host” CPU, which makes programming more complicated and has implications for performance.

Slide 3

OpenCL

- Early work on shared-memory and message-passing programming resulted in many competing programming environments — but eventually, OpenMP and MPI emerged as standards.
- Similarly, initially many different programming environments for GPGPU, but OpenCL might be emerging as a standard.
- In both cases, idea was to come up with a single standard, then allow many implementations. For MPI, standard defines concepts and library. For OpenMP, standard defines concepts, library, and compiler directives. For OpenCL, concepts and library.
- First release 2008; evolving fairly rapidly. Meant to address not just GPGPU but more-general problem of “heterogeneous computing” (computing using mix of computational resources).

Slide 4

What’s an OpenCL Program Like?

- Source code in C/C++, with calls to OpenCL functions.
- Typically includes source to be compiled at runtime for whatever device is to be used. “Device”? yes, many new terms/concepts . . .

Slide 5

OpenCL Terms and Concepts

- *Compute device* — something capable of doing computations (CPU, GPU, etc.).
- *Kernel* — computation to execute on device.
- *Index-space* — range of indices (1D or more) on which to execute kernel.
- *Work-item* — one execution of kernel. Grouped into *work-groups*.
- *Compute context*, *program object*, *command queue* — various aspects of setting up environment and assigning work to devices.
- Several *memory regions* — host memory, local memory, etc.

Slide 6

Simple(?) Examples / Compiling and Executing

- Compile with C/C++ compiler, with flags to pick up additional files from OpenCL implementation.
- Execute like regular program — but may need access to GPU beyond what's always available.
- Maybe worth noting that you can't really write a "hello world" program, since compute device doesn't necessarily have access to standard output! (Look at semi-simple examples `semi-hello.c` and `vector-add.c` on sample programs page.)

Minute Essay

- None — sign in.