OpenCL bevezetés II.

OpenCL infrastruktúra

- OpenCL platform
- Számító eszközök
- OpenCL kontextusok
- Parancs sorok
- Szinkronizáció
- Memória objektumok
- OpenCL programok
  - OpenCL függvények

OpenCL platform

- Platformok lekérdezése

```
cl_int clGetPlatformIDs(cl_int num_entries,
                        cl_platform_id *platforms,
                        cl_uint *num_platforms)
```

  - num_entries: lekérdezendő platformok száma
  - platforms: platformok azonosítója
  - num_platforms: lekérdezett platformok száma

OpenCL platform

- Platformok lekérdezése

```
cl_uint num_platforms = 0;
clGetPlatformIDs(0, NULL, &num_platforms)
cl_platform_id *platforms = new cl_platform_id[num_platforms];
clGetPlatformIDs(num_platforms, platforms, NULL);
```

  - lekérdezzük a paraméterek számát
  - lefoglaljuk a a megfelelő méretű memóriát
  - lekérdezzük a paramétereket

OpenCL platform

- Platform információk

```
cl_int clGetPlatformInfo(cl_platform_id platform,
                         cl_platform_info param_name,
                         size_t param_value_size,
                         void *param_value,
                         size_t *param_value_size_ret)
```

  - platform: a kiválasztott platform azonosítója
  - param_name: a lekérdezendő információ
  - param_value_size: információ max mérete (byte)
  - param_value: az információ tárolója
  - param_value_size_ret: a visszaadott információ mérete

OpenCL platform

- Platform információk
  - CL_PLATFORM_PROFILE
    - FULL_PROFILE
    - EMBEDDED_PROFILE
  - CL_PLATFORM_VERSION
  - CL_PLATFORM_NAME
  - CL_PLATFORM_VENDOR
  - CL_PLATFORM_EXTENSIONS

OpenCL számító eszközök

- A platformon elérhető eszközök

```
cl_int clGetDeviceIDs(cl_platform_id platform,
                      cl_device_type device_type,
                      cl_uint num_entries,
                      cl_device_id *devices,
                      cl_uint *num_devices)
```

  - device_type: a lekérdezendő eszközök típusa
    - CL_DEVICE_TYPE_CPU
    - CL_DEVICE_TYPE_GPU
    - CL_DEVICE_TYPE_ACCELERATOR
    - CL_DEVICE_TYPE_DEFAULT
    - CL_DEVICE_TYPE_ALL

OpenCL számító eszközök

- Eszköz információk

```
cl_int clGetDeviceInfo(cl_device_id device,
                       cl_device_info param_name,
                       size_t param_value_size,
                       void *param_value,
                       size_t *param_value_size_ret)
```

  - device: a lekérdezendő eszköz azonosítója
  - param_ name: a lekérdezendő információ
  - …

OpenCL számító eszközök

- Általános eszköz információk
  - CL_DEVICE_TYPE
  - CL_DEVICE_COMPILER_AVAILABLE
  - CL_DEVICE_NAME
  - CL_DEVICE_VENDOR
  - CL_DEVICE_VERSION
  - CL_DRIVER_VERSION
  - CL_DEVICE_EXTENSIONS

OpenCL számító eszközök

- Számítási információk
  - CL_DEVICE_MAX_COMPUTE_UNITS
  - CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS
  - CL_DEVICE_MAX_WORK_ITEM_SIZES
  - CL_DEVICE_MAX_WORK_GROUP_SIZE
  - CL_DEVICE_AVAILABLE

OpenCL számító eszközök

- Memória információk
  - CL_DEVICE_MAX_PARAMETER_SIZE
  - CL_DEVICE_GLOBAL_MEM_SIZE
  - CL_DEVICE_MAX_MEM_ALLOC_SIZE
  - CL_DEVICE_LOCAL_MEM_SIZE
  - CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE
  - CL_DEVICE_MAX_CONSTANT_ARGS
  - CL_DEVICE_IMAGE_SUPPORT

OpenCL kontextusok

- Kontextus létrehozása

```
cl_context clCreateContext(const cl_context_properties *props,
                           cl_uint num_devices,
                           const cl_device_id *devices,
                           void (*pfn_notify)(...),
                           void *user_data,
                           cl_int *errcode_ret)
```

  - props: tulajdonság – érték lista
    - CL_CONTEXT_PLATFORM – cl_platform_id
  - num_devices: a kontextushoz tartozó eszközök száma
  - device: a kontextushoz tartozó eszközök listája

OpenCL kontextusok

- Kontextus létrehozása

```
cl_context clCreateContext(const cl_context_properties *props,
                           cl_uint num_devices,
                           const cl_device_id *devices,
                           void (*pfn_notify)(...),
                           void *user_data,
                           cl_int *errcode_ret)
```

  - pfn_notify: callback függvény a hibakezeléshez
    - aszinkron hibajelzés a kontextusban
    - szál biztos implementáció szükséges (thread-safe)
    - paraméterei
      - const char* errinfo: a hiba szövege
      - const void* private_info: implementáció függő hibakereső adat
      - size_t cb: a private_info mérete (byte)
      - void *user_data: felhasználói információ

OpenCL kontextusok

- Kontextus létrehozása

```
cl_context clCreateContext(const cl_context_properties *props,
                           cl_uint num_devices,
                           const cl_device_id *devices,
                           void (*pfn_notify)(...),
                           void *user_data,
                           cl_int *errcode_ret)
```

  - user_data: felhasználói információ
    - a pfn_notify függvény kapja meg
  - errcode_ret: hiba információ

OpenCL kontextusok

- Kontextus számláló növelése

```
cl_int clRetainContext(cl_context context)
```

- Kontextus számláló csökkentése

```
cl_int clReleaseContext(cl_context context)
```

- Kontextus információk

```
cl_int clGetContextInfo(cl_context context,
                        cl_context_info param_name,
                        size_t param_value_size,
                        void *param_value,
                        size_t *param_value_size_ret)
```

  - param_value
    - CL_CONTEXT_REFERENCE_COUNT
    - CL_CONTEXT_DEVICES
    - CL_CONTEXT_PROPERTIES

OpenCL parancs sorok

- Parancs sor létrehozása

```
cl_command_queue
clCreateCommandQueue(cl_context context,
                     cl_device_id device,
                     cl_command_queue_properties properties,
                     cl_int *errcode_ret)
```

  - properties
    - CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE
    - CL_QUEUE_PROFILIN_ENABLE

OpenCL parancs sorok

- Parancs sor számláló növelése

```
cl_int clRetainCommandQueue(cl_command_queue queue)
```

- Parancs sor számláló csökkentése

```
cl_int clReleaseCommandQueue(cl_command_queue queue)
```

- Parancs sor információk

```
cl_int clGetCommandQueueInfo(cl_command_queue queue,
                             cl_command_queue_info param_name,
                             size_t param_value_size,
                             void *param_value,
                             size_t *param_value_size_ret)
```

  - param_value
    - CL_QUEUE_CONTEXT
    - CL_QUEUE_DEVICE
    - CL_QUEUE_PROPERTIES

Szinkornizáció

- Eseményre várakozás

```
cl_int clEnqueueWaitForEvents(cl_command_queue queue,
                              cl_uint num_events,
                              const cl_event *event_list)
```

  - implicit szinkronizáció
- Barrier

```
cl_int clEnqueueBarrier(cl_command_queue queue)
```

  - explicit szinkronizáció

Szinkornizáció

- Parancs sor ürítés
  - minden hívás eljutott az eszközig

```
cl_int clFlush(cl_command_queue queue)
```

  - minden hívás lefutott

```
cl_int clFinish(cl_command_queue queue)
```

Memória objektumok

- Buffer objektum
  - lineáris memória terület
  - skalár, vektor, struktúra típus
  - pointeren keresztül címezhető
- Image objektum
  - 2D, 3D memória terület
  - előre definiált textúra formátumok
  - samplereken keresztül címezhető

Memória objektumok

- Memória objektum létrehozása

```
cl_mem clCreateBuffer(cl_context context,
                      cl_mem_flags flags,
                      size_t size,
                      void *host_ptr,
                      cl_int *errcode_ret)
```

  - size: a buffer mérete (byte)
  - host_ptr: előre lefoglalt memória terület

Memória objektumok

- Memória objektum létrehozása

```
cl_mem clCreateBuffer(...
                      cl_mem_flags flags,
                      ...)
```

  - flags:
    - CL_MEM_READ_WRITE
    - CL_MEM_WRITE_ONLY
    - CL_MEME_READ_ONLY
    - CL_MEM_USE_HOST_PTR
    - CL_MEM_ALLOC_HOST_PTR
    - CL_MEME_COPY_HOST_PTR

Memória objektumok

- Memória objektum olvasása

```
cl_int clEnqueueReadBuffer(cl_command_queue command_queue,
                           cl_mem buffer,
                           cl_bool blocking_read,
                           size_t offset,
                           size_t cb,
                           void *ptr,
                           cl_uint num_events_in_wait_list,
                           const cl_event *event_wait_list,
                           cl_event *event)
```

  - buffer: az olvasandó memória objektum
  - blocking_read: bevárja-e az olvasás befejeztét

Memória objektumok

- Memória objektum olvasása

```
cl_int clEnqueueReadBuffer(cl_command_queue command_queue,
                           cl_mem buffer,
                           cl_bool blocking_read,
                           size_t offset,
                           size_t cb,
                           void *ptr,
                           cl_uint num_events_in_wait_list,
                           const cl_event *event_wait_list,
                           cl_event *event)
```

  - offset: az olvasandó memória ofszete
  - cb: az olvasandó méret
  - ptr: az olvasás célja

Memória objektumok

- Memória objektum olvasása

```
cl_int clEnqueueReadBuffer(cl_command_queue command_queue,
                           cl_mem buffer,
                           cl_bool blocking_read,
                           size_t offset,
                           size_t cb,
                           void *ptr,
                           cl_uint num_events_in_wait_list,
                           const cl_event *event_wait_list,
                           cl_event *event)
```

  - num_events_in_wait_list: bevárandó események száma
  - event_wait_list: bevárandó események
  - event: az olvasás végét jelző esemény

Memória objektumok

- Memória objektum írása

```
cl_int clEnqueueWriteBuffer(cl_command_queue command_queue,
                            cl_mem buffer,
                            cl_bool blocking_write,
                            size_t offset,
                            size_t cb,
                            void *ptr,
                            cl_uint num_events_in_wait_list,
                            const cl_event *event_wait_list,
                            cl_event *event)
```

  - A paraméterek hasonlóak az olvasáshoz

Memória objektumok

- Memória objektum másolása

```
cl_int clEnqueueCopyBuffer(cl_command_queue command_queue,
                           cl_mem src_buffer,
                           cl_mem dst_buffer,
                           size_t src_offset,
                           size_t dst_offset,
                           size_t cb,
                           cl_uint num_events_in_wait_list,
                           const cl_event *event_wait_list,
                           cl_event *event)
```

  - src_buffer, src_offset: a másolandó memória
  - dst_buffer, dst_offset: a cél memória
  - cb: a másolandó memória mérete

Memória objektumok

- Memória objektum számláló növelése

```
cl_int clRetainMemObject(cl_mem memobj)
```

- Memória objektum számláló csökkentése

```
cl_int clReleaseMemObject(cl_mem memobj)
```

- Memória objektum információk

```
cl_int clGetMemObjectInfo(cl_mem memobj,
                          cl_mem_info param_name,
                          size_t param_value_size,
                          void *param_value,
                          size_t *param_value_size_ret)
```

  - param_value
    - CL_MEM_TYPE
    - CL_MEM_SIZE
    - CL_MEM_MAP_COUNT

Memória objektumok

- Memória objektum mappelése

```
void* clEnqueueMapBuffer(cl_command_queue command_queue,
                         cl_mem buffer,
                         cl_bool blocking_map,
                         cl_map_flags map_flags,
                         size_t offset,
                         size_t cb,
                         cl_uint num_events_in_wait_list,
                         const cl_event *event_wait_list,
                         cl_event *event,
                         cl_int *errcode_ret)
```

  - map_flags
    - CL_MAP_READ
    - CL_MAP_WRITE

Memória objektumok

- Memória objektum mappelése

```
cl_int clEnqueueUnmapBuffer(cl_command_queue command_queue,
                            cl_mem buffer,
                            void *mapped_ptr,
                            cl_uint num_events_in_wait_list,
                            const cl_event *event_wait_list,
                            cl_event *event)
```

  - buffer: a kimappelendő memória objektum
  - mapped_ptr: host pointer a mappelt memóriára

Program objektumok

- Program objektum
  - Kontextus információk
  - Program forrása vagy bináris reprezentáció
  - Az utolsó sikeresen lefordított program
  - Fordítási információk
    - fordítási opciók
    - log
  - Kernel objektumok száma

Program objektumok

- Program objektum létrehozása

```
cl_program clCreateProgramWithSource(cl_context context,
                                     cl_uint count,
                                     const char **strings,
                                     const size_t *lengths,
                                     cl_int *errcode_ret)
```

  - strings: a program forrása
  - lengths: a források hossza
  - count: a források száma

Program objektumok

- Program objektum létrehozása

```
cl_program
clCreateProgramWithBinary(cl_context context,
                          cl_uint num_devices,
                          const cl_device_id *device_list,
                          const size_t *lengths,
                          const unsigned char **binaries,
                          cl_int *binary_status,
                          cl_int *errcode_ret)
```

  - device_list: a program álltal használt eszközök
  - binaries: az eszközökhöz tartozó bináris kód
  - binary_status: sikeres volt-e a betöltés

Program objektumok

- Program objektum számláló növelése

```
cl_int clRetainProgram(cl_program program)
```

- Program objektum számláló csökkentése

```
cl_int clReleaseProgram(cl_program program)
```

- Program objektum információk

```
cl_int clGetProgramInfo(cl_program program,
                        cl_program_info param_name,
                        size_t param_value_size,
                        void *param_value,
                        size_t *param_value_size_ret)
```

  - param_value
    - CL_PROGRAM_DEVICES
    - CL_PROGRAM_SOURCE
    - CL_PROGRAM_BINARIES

Program objektumok

- Program fordítása

```
cl_int clBuildProgram(cl_program program,
                      cl_uint num_devices,
                      const cl_device_id *device_list,
                      const char* options,
                      void (*pfn_notify)(...),
                      void *user_data)
```

  - program: a fordítandó program objektum
  - num_devices: az eszközök száma a fordításhoz
  - device_list: a céleszközök listája

Program objektumok

- Program fordítási opciók
  - Preprocesszor
    - -D name
    - -D name = definition
    - -I dir
  - Matematikai viselkedés
    - -cl-single-precision-constant
    - -cl-denorms-are-zero
  - Fordító viselkedés
    - -w
    - -Werror

Program objektumok

- Program fordítási opciók
  - Optimalizáció
    - -cl-opt-disable
    - -cl-strict-aliasing
    - -cl-mad-enable
    - -cl-no-signed-zeros
    - -cl-unsafe-math-optimizations
    - -cl-finite-math-only
    - -cl-fast-relaxed-math

Program objektumok

- Fordítási információk

```
cl_int clGetProgramBuildInfo(cl_program program,
                             cl_device_id device,
                             cl_program_build_info param_name,
                             size_t param_value_size,
                             void *param_value,
                             size_t *param_value_size_ret)
```

  - CL_PROGRAM_BUILD_STATUS
  - CL_PROGRAM_BUILD_OPTIONS
  - CL_PROGRAM_BUILD_LOG

- Fordító törlése

```
cl_int clUnloadCompiler(void)
```

  - A fordító törölhető a memóriából
  - Csak tanács, új program esetén betöltődik ismét

Kernel objektumok

- Kernel objektum létrehozása

```
cl_kernel clCreateKernel(cl_program program,
                         const char *kernel_name,
                         cl_int *errcode_ret)
```

  - kernel_name: a létrehozandó kernel neve
- Kernel objektum számláló növelése

```
cl_int clRetainKernel(cl_kernel kernel)
```

- Kernel objektum számláló csökkentése

```
cl_int clReleaseKernel(cl_kernel kernel)
```

Kernel objektumok

- Kernel objektum információk

```
cl_int clGetKernelInfo(cl_kernel kernel,
                       cl_kernel_info param_name,
                       size_t param_value_size,
                       void *param_value,
                       size_t *param_value_size_ret)
```

  - param_name
    - CL_KERNEL_FUNCTION_NAME
    - CL_KERNEL_NUM_ARGS
    - CL_KERNEL_PROGRAM

Kernel objektumok

- Kernel paraméterek

```
cl_int clSetKernelArg(cl_kernel kernel,
                      cl_uint arg_index,
                      size_t arg_size,
                      const void *arg_value)
```

  - arg_index: a beállítandó paraméter indexe
  - arg_size: a beállítandó paraméter mérete
  - arg_value: a beállítandó paraméter értéke

Kernel objektumok

- Kernel paraméterek
  - Kernel

```
__kernel
void sampleKernel(const float count,
                  __global float* data){
  ...
}
```

  - Paraméter beállítás

```
clSetKernelArg(sampleKernel, 0, sizeof(float), fData);
clSetKernelArg(sampleKernel, 1, sizeof(dataPtr), &dataPtr);
```

Kernel objektumok

- Munkacsoport paraméterek

```
cl_int clGetKernelWorkGroupInfo(cl_kernel kernel,
                                cl_device_id device,
                                cl_kernel_work_group_info pname,
                                size_t param_value_size,
                                void *param_value,
                                size_t *param_value_size_ret)
```

  - pname
    - CL_KERNEL_WORK_GROUP_SIZE
    - CL_KERNEL_COMPILE_WORK_GROUP_SIZE
    - CL_KERNEL_LOCAL_MEM_SIZE

Kernel objektumok

- Kernel futtatás

```
cl_int clEnqueueNDRangeKernel(cl_command_queue queue,
                              cl_kernel kernel,
                              cl_uint work_dim,
                              const size_t *global_work_offset,
                              const size_t *global_work_size,
                              const size_t *local_work_size,
                              cl_uint num_events_in_wait_list,
                              const cl_event *event_wait_list,
                              cl_event *event)
```

  - work_dim: az NDRange dimenziója
  - global_work_offset: NULL
  - global_work_size: a teljes problématér mérete
  - local_work_size: egy munka csoport mérete

OpenCL példa

- Host program

```
void square(){
  cl_kernel squareKernel = createKernel(program, "square");

  const int data_size = 1024;

  float* inputData = (float*)malloc(sizeof(float) * data_size);
  for(int i = 0; i < data_size; ++i){
    inputData[i] = i;
  }

  cl_mem clInputData = clCreateBuffer(context, CL_MEM_READ_ONLY,
                          sizeof(float) * data_size, NULL, NULL);
  clEnqueueWriteBuffer(commands, clInputData,
                       CL_TRUE, 0, sizeof(float) * data_size,
                       inputData, 0, NULL, NULL) );
// ...
```

OpenCL példa

- Host program

```
float* data = (float*)malloc(sizeof(float)*data_size);
cl_mem clData = clCreateBuffer(context, CL_MEM_WRITE_ONLY,
                        sizeof(float) * data_size, NULL, NULL);

clSetKernelArg(squareKernel, 0, sizeof(cl_mem), &clInputData);
clSetKernelArg(squareKernel, 1, sizeof(cl_mem), &clData);
clSetKernelArg(squareKernel, 2, sizeof(int), &data_size);

size_t workgroupSize = 0;
clGetKernelWorkGroupInfo(squareKernel, device_id,
                         CL_KERNEL_WORK_GROUP_SIZE,
                         sizeof(workgroupSize),
                         &workgroupSize, NULL);
clEnqueueNDRangeKernel(commands, squareKernel, 1, NULL,
                       &workSize, &workgroupSize, 0, NULL, NULL) );

clFinish(commands);
```

OpenCL példa

- Host program

```
 clEnqueueReadBuffer(commands, clData, CL_TRUE, 0,
                     sizeof(float) * data_size, data, 0, NULL, NULL);

  int wrong = 0;
  for(int i = 0; i < data_size; ++i){
    if(data[i] != inputData[i] * inputData[i]){
      wrong++;
    }
  }
  std::cout << "Wrong squares: " << wrong << std::endl;

  clReleaseKernel(squareKernel);
  free(data);
  free(inputData);
}
```

OpenCL példa

- OpenCL program

```
__kernel void square(__global float* inputData,
                     __global float* outputData,
                     const int data_size){
  __private int id = get_global_id(0);
  outputData[id] = inputData[id] * inputData[id];
}
```