Лекция 4

- Уровни компиляции *nvcc*.
- *.cubin, .fatbin, .gpu* и *.ptx* файлы.
- PTX (*Parallel Thread eXecution*) ISA (*Instruction Set Architecture*).
- CUDA Driver API.

```
__global__ void gTest0(float* a, float* b){
    int tid=threadIdx.x+blockIdx.x*blockDim.x;
    a[tid]=a[tid]+b[tid];
}
int main(){
...............................................
    gTest0<<<N/128, 128>>>(a_d,b_d);
    cudaDeviceSynchronize();
.............................................
}
```

```
tests/test0> nvcc -g -G test0m.cu -o test0m
/tests/test0> cuda-gdb test0m
```

```
(cuda-gdb) list gTest0
1        #include <stdio.h>
2        #include <malloc.h>
3
4        __global__ void gTest0(float* a, float* b){
5            int tid=threadIdx.x+blockIdx.x*blockDim.x;
6            a[tid]=a[tid]+b[tid];
7        }
8
9        int main(){
10           int N=1024;
```

```
(cuda-gdb) break 5
………………………………………………………………………………………………………….
```

```
(cuda-gdb) run
..........................................................................................................................................................
CUDA thread hit Breakpoint 1, gTest0<<<(8,1,1),(128,1,1)>>>
(a=0x7fffc3200000, b=0x7fffc3201000)
   at test0m.cu:5
5            int tid=threadIdx.x+blockIdx.x*blockDim.x;
```

```
(cuda-gdb) disassemble
Dump of assembler code for function _Z6gTest0PfS_:
  0x00007fffc8e3f800 <+0>:     MOV R1, c[0x0][0x28]
  0x00007fffc8e3f810 <+16>:    MOV R2, RZ
  0x00007fffc8e3f820 <+32>:    LDC.64 R2, c[0x0][R2+0x160]
  0x00007fffc8e3f830 <+48>:    MOV R7, R2
  0x00007fffc8e3f840 <+64>:    MOV R8, R3
  0x00007fffc8e3f850 <+80>:    MOV R7, R7
  0x00007fffc8e3f860 <+96>:    MOV R8, R8
  ............................................................
  0x00007fffc8e3f8e0 <+224>:   MOV R8, R8
  0x00007fffc8e3f8f0 <+240>:   MOV R5, R5
  0x00007fffc8e3f900 <+256>:   MOV R6, R6
=> 0x00007fffc8e3f910 <+272>:   S2R R0, SR_TID.X
  0x00007fffc8e3f920 <+288>:   MOV R0, R0
  0x00007fffc8e3f930 <+304>:   S2R R2, SR_CTAID.X
  0x00007fffc8e3f940 <+320>:   MOV R2, R2
...........................................................................
  0x00007fffc8e3fca0 <+1184>:  BRA 0x4a0
  0x00007fffc8e3fcb0 <+1200>:  NOP
  0x00007fffc8e3fcc0 <+1216>:  NOP
```

Раздельная компиляция

test2.cu

```
extern "C"
__global__ void gTest2(float* a, float* b){
        int tid=threadIdx.x+blockIdx.x*blockDim.x;
        a[tid]=a[tid]+b[tid];
}
```

test2m.cu

```
#include <stdio.h>
#include <malloc.h>

extern "C" __global__ void gTest2(float*, float*);

int main(){
        int N=1024;
        float* a=(float*)calloc(N, sizeof(float));
        float* b=(float*)calloc(N, sizeof(float));

        for(int i=0; i<N; i++){
        a[i]=2*i;
        b[i]=2*i+1;
        }
```

```
    float *a_d, *b_d;

    cudaMalloc((void**)&a_d, N*sizeof(float));
    cudaMalloc((void**)&b_d, N*sizeof(float));

    cudaMemcpy(a_d, a, N*sizeof(float), cudaMemcpyHostToDevice);
    cudaMemcpy(b_d, b, N*sizeof(float), cudaMemcpyHostToDevice);

    gTest2<<<N/128,128>>>(a_d,b_d);
    cudaDeviceSynchronize();

    cudaMemcpy(a, a_d, N*sizeof(float), cudaMemcpyDeviceToHost);

    for(int i=0; i<N; i+=N/16)
    printf("%g\n",a[i]);
}
```

```
tests/test2> nvcc -c  test2.cu
tests/test2> nvcc -c  test2m.cu
tests/test2> nvcc   test2.o   test2m.o   -o   test2e
```

```
#include <stdio.h>
#include <malloc.h>
#include <cuda.h>

extern "C" void hLauncherTest2(float* a, float* b, int N);

int main(){
………………………………………………
     hLauncherTest2(a_d,b_d,N);
………………………………………………
}
```

```
__global__ void gTest2(float* a, float* b){
      int tid=threadIdx.x+blockIdx.x*blockDim.x;
      a[tid]=a[tid]+b[tid];
}

extern "C"
void hLauncherTest2(float* a, float* b, int N){
 gTest2<<<N/128,128>>>(a,b);
 cudaDeviceSynchronize();
}
```

```
nvcc -Xcompiler -fPIC -shared  test2.cu -o libtest2.so
```

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:.
```

```
nvcc test2m.cu  -L. -ltest2 -o test2m
```

PTX

PTX (Parallel Thread eXecution) определяет виртуальную машину и набор инструкций (ISA - Instruction Set Architecture). PTX программа транслируется во время загрузки в команды соответствующего GPU и машинный код загружается на GPU драйвером.

```
tests/test2> nvcc -ptx test2.cu
```

```
.version 8.5
.target sm_52
.address_size 64

        // .globl       gTest2

.visible .entry gTest2(
        .param .u64 gTest2_param_0,
        .param .u64 gTest2_param_1
)
{
        .reg .f32       %f<4>;
        .reg .b32       %r<5>;
        .reg .b64       %rd<8>;
```

test2.ptx

```
    ld.param.u64    %rd1, [gTest2_param_0];
    ld.param.u64    %rd2, [gTest2_param_1];
    cvta.to.global.u64      %rd3, %rd2;
    cvta.to.global.u64      %rd4, %rd1;
    mov.u32         %r1, %tid.x;
    mov.u32         %r2, %ctaid.x;
    mov.u32         %r3, %ntid.x;
    mad.lo.s32      %r4, %r2, %r3, %r1;
    mul.wide.s32    %rd5, %r4, 4;
    add.s64         %rd6, %rd4, %rd5;
    ld.global.f32   %f1, [%rd6];
    add.s64         %rd7, %rd3, %rd5;
    ld.global.f32   %f2, [%rd7];
    add.f32         %f3, %f1, %f2;
    st.global.f32   [%rd6], %f3;
    ret;
}
```

mst.cu

```
#include <stdio.h>
#include <malloc.h>

 extern "C" {  __global__ void gStub(float* a, float* b){ }  }

int main(){
    int N=2048;
    float* a=(float*)calloc(N, sizeof(float));
    float* b=(float*)calloc(N, sizeof(float));

    for(int i=0; i<N; i++){
        a[i]=2*i;
        b[i]=2*i+1;
    }
float *a_d, *b_d;
    cudaMalloc((void**)&a_d, N*sizeof(float));
    cudaMalloc((void**)&b_d, N*sizeof(float));

    cudaMemcpy(a_d, a, N*sizeof(float), cudaMemcpyHostToDevice);
    cudaMemcpy(b_d, b, N*sizeof(float), cudaMemcpyHostToDevice);

    gStub<<<N/128,128>>>(a_d,b_d);
    cudaDeviceSynchronize();

    cudaMemcpy(a, a_d, N*sizeof(float), cudaMemcpyDeviceToHost);

    for(int i=0; i<N; i+=N/16)
        printf("%g\n",a[i]);
}
```

```
.version 8.5
.target sm_52
.address_size 64

    // .globl    gStub

.visible .entry gStub(
    .param .u64 gStub_param_0,
    .param .u64 gStub_param_1
)
{

    ret;

}
```

```
tests/mystub> nvcc -dryrun -arch=sm_52  mst.cu -o mst
--keep 2> dryrun_mst.out


tests/mystub> ll
total 12
-rw-r--r-- 1 malkov users 5053 авг 21 15:18
dryrun_mst.out
-rw-r--r-- 1 malkov users  815 авг 21 15:09 mst.cu
```

```
tests/mystub> ls -ltr
total 4260
drwxr-xr-x 2 malkov users       21 авг 21 13:42 spare
-rw-r--r-- 1 malkov users      815 авг 21 15:09 mst.cu
-rw-r--r-- 1 malkov users     5053 авг 21 15:18 dryrun_mst.out
-rw-r--r-- 1 malkov users  1091325 авг 21 15:21 mst.cpp4.ii
-rw-r--r-- 1 malkov users       24 авг 21 15:21 mst.module_id
-rw-r--r-- 1 malkov users  1010983 авг 21 15:21 mst.cudafe1.cpp
-rw-r--r-- 1 malkov users  1184988 авг 21 15:21 mst.cpp1.ii
-rw-r--r-- 1 malkov users       13 авг 21 15:21 mst.cudafe1.c
-rw-r--r-- 1 malkov users      312 авг 21 15:21 mst.ptx
-rw-r--r-- 1 malkov users     1237 авг 21 15:21 mst.cudafe1.stub.c
-rw-r--r-- 1 malkov users    10401 авг 21 15:21 mst.cudafe1.gpu
-rw-r--r-- 1 malkov users     1960 авг 21 15:21 mst.sm_52.cubin
-rw-r--r-- 1 malkov users     6647 авг 21 15:21 mst.fatbin.c
-rw-r--r-- 1 malkov users     2248 авг 21 15:21 mst.fatbin
-rw-r--r-- 1 malkov users     8984 авг 21 15:21 mst.o
-rw-r--r-- 1 malkov users      872 авг 21 15:21 mst_dlink.sm_52.cubin
-rw-r--r-- 1 malkov users       32 авг 21 15:21 mst_dlink.reg.c
-rw-r--r-- 1 malkov users     3190 авг 21 15:21 mst_dlink.fatbin.c
-rw-r--r-- 1 malkov users      952 авг 21 15:21 mst_dlink.fatbin
-rw-r--r-- 1 malkov users     2904 авг 21 15:21 mst_dlink.o
-rwxr-xr-x 1 malkov users   975856 авг 21 15:21 mst
```

nvcc mst.cu –keep -o mst

```
tests/mystub> ./mst
0
256
512
768
1024
1280
1536
1792
2048
2304
……. .
```

```
 extern "C"{
__global__ void gStub(float* a, float* b){
    int tid=threadIdx.x+blockIdx.x*blockDim.x;
    a[tid]=a[tid]+b[tid];
   }
}
```

```
.version 8.5
.target sm_52
.address_size 64

       // .globl       gStub

.visible .entry gStub(
       .param .u64 gStub_param_0,
       .param .u64 gStub_param_1
)
{
       .reg .f32       %f<4>;
       .reg .b32       %r<5>;
       .reg .b64       %rd<8>;
```

```
        ld.param.u64    %rd1, [gStub_param_0];
        ld.param.u64    %rd2, [gStub_param_1];
        cvta.to.global.u64      %rd3, %rd2;
        cvta.to.global.u64      %rd4, %rd1;
        mov.u32         %r1, %tid.x;
        mov.u32         %r2, %ctaid.x;
        mov.u32         %r3, %ntid.x;
        mad.lo.s32      %r4, %r2, %r3, %r1;
        mul.wide.s32    %rd5, %r4, 4;
        add.s64         %rd6, %rd4, %rd5;
        ld.global.f32   %f1, [%rd6];
        add.s64         %rd7, %rd3, %rd5;
        ld.global.f32   %f2, [%rd7];
        add.f32         %f3, %f1, %f2;
        st.global.f32   [%rd6], %f3;
        ret;
}
```

```
#$ cicc --c++17 --gnu_version=110200 --display_error_number --orig_src_file_name
"mst.cu" --orig_src_path_name "/home/malkov/tests/mystub/mst.cu"
--allow_managed   -arch compute_52 -m64 --no-version-ident -ftz=0 -prec_div=1
-prec_sqrt=1 -fmad=1 --include_file_name "mst.fatbin.c" -tused
--module_id_file_name "mst.module_id" --gen_c_file_name "mst.cudafe1.c"
--stub_file_name "mst.cudafe1.stub.c" --gen_device_file_name "mst.cudafe1.gpu"
"mst.cpp1.ii" -o "mst.ptx"
#$----------------------------------------------------------------------------------------
ptxas -arch=sm_52 -m64  "mst.ptx"  -o "mst.sm_52.cubin"
```

```
fatbinary --create="mst.fatbin" -64 --cicc-cmdline="-ftz=0 -prec_div=1 -prec_sqrt=1
-fmad=1 " "--image3=kind=elf,sm=52,file=mst.sm_52.cubin"
"--image3=kind=ptx,sm=52,file=mst.ptx" --embedded-fatbin="mst.fatbin.c"

"/home/malkov/anaconda3/bin"/x86_64-conda-linux-gnu-c++
-D__CUDA_ARCH__=520 -D__CUDA_ARCH_LIST__=520
-D__NV_LEGACY_LAUNCH -c -x c++  -DCUDA_DOUBLE_MATH_FUNCTIONS
-Wno-psabi "-I/usr/local/cuda-12.5/bin/../targets/x86_64-linux/include"   -m64
"mst.cudafe1.cpp" -o "mst.o"

nvlink -m64 --arch=sm_52 --register-link-binaries="mst_dlink.reg.c"
"-L/usr/local/cuda-12.5/bin/../targets/x86_64-linux/lib/stubs"
"-L/usr/local/cuda-12.5/bin/../targets/x86_64-linux/lib" -cpu-arch=X86_64 "mst.o"
-lcudadevrt  -o "mst_dlink.sm_52.cubin" --host-ccbin
"/home/malkov/anaconda3/bin/x86_64-conda-linux-gnu-c++"
```

```
fatbinary --create="mst_dlink.fatbin" -64 --cicc-cmdline="-ftz=0 -prec_div=1
-prec_sqrt=1 -fmad=1 " -link "--image3=kind=elf,sm=52,file=mst_dlink.sm_52.cubin"
--embedded-fatbin="mst_dlink.fatbin.c"

"/home/malkov/anaconda3/bin"/x86_64-conda-linux-gnu-c++
-D__CUDA_ARCH_LIST__=520 -D__NV_LEGACY_LAUNCH -c -x c++
-DFATBINFILE="\"mst_dlink.fatbin.c\""
-DREGISTERLINKBINARYFILE="\"mst_dlink.reg.c\"" -I.
-D__NV_EXTRA_INITIALIZATION= -D__NV_EXTRA_FINALIZATION=
-D__CUDA_INCLUDE_COMPILER_INTERNAL_HEADERS__  -Wno-psabi
"-I/usr/local/cuda-12.5/bin/../targets/x86_64-linux/include"
-D__CUDACC_VER_MAJOR__=12 -D__CUDACC_VER_MINOR__=5
-D__CUDACC_VER_BUILD__=82 -D__CUDA_API_VER_MAJOR__=12
-D__CUDA_API_VER_MINOR__=5 -D__NVCC_DIAG_PRAGMA_SUPPORT__=1
-m64 "/usr/local/cuda-12.5/bin/crt/link.stub" -o "mst_dlink.o
```

```
"/home/malkov/anaconda3/bin"/x86_64-conda-linux-gnu-c++
-D__CUDA_ARCH_LIST__=520 -D__NV_LEGACY_LAUNCH -m64 -Wl,--start-group
"mst_dlink.o" "mst.o"   "-L/usr/local/cuda-12.5/bin/../targets/x86_64-linux/lib/stubs"
"-L/usr/local/cuda-12.5/bin/../targets/x86_64-linux/lib"  -lcudadevrt  -lcudart_static  -lrt
-lpthread  -ldl  -Wl,--end-group -o "mst"
```

```
tests/mystub> ./dryrun_mst.out
```

```
tests/mystub> ./mst
1
513
1025
1537
2049
2561
3073
3585
4097
4609
……. .
```

```
tests/cudrapi> g++ -I/usr/local/cuda/include
-L/usr/local/cuda/lib64 -lcudart -lcuda cda.cpp -o cda
```

```
#include <cuda.h>
#include <cuda_runtime.h>
#include <stdio.h>
#include <malloc.h>

int main(){
    cuInit(0);
    CUdevice cuDevice;
    CUresult res = cuDeviceGet(&cuDevice, 0);
    if (res != CUDA_SUCCESS){
        printf("cannot acquire device 0\n");
        exit(1);
    }

    CUcontext cuContext;
    res = cuCtxCreate(&cuContext, 0, cuDevice);
    if (res != CUDA_SUCCESS){
        printf("cannot create context\n");
        exit(1);
    }
```

```
int N=2048;
float* a=(float*)calloc(N, sizeof(float));
float* b=(float*)calloc(N, sizeof(float));

for(int i=0; i<N; i++){
    a[i]=2*i;
    b[i]=2*i+1;
}

float *a_d, *b_d;
cudaMalloc((void**)&a_d, N*sizeof(float));
cudaMalloc((void**)&b_d, N*sizeof(float));

cudaMemcpy(a_d, a, N*sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(b_d, b, N*sizeof(float), cudaMemcpyHostToDevice);

//gStub<<<N/128,128>>>(a_d,b_d);
//cudaDeviceSynchronize();
```

```
 CUmodule cuModule = (CUmodule)0;
 cuModuleLoad(&cuModule, "cda.ptx");
 CUfunction gStub;
 cuModuleGetFunction(&gStub, cuModule, "gStub");

 void* args[] = {&a_d, &b_d};
 cuLaunchKernel(gStub, N/128, 1, 1, 128, 1, 1, 0, 0, args, 0);

  cudaMemcpy(a, a_d, N*sizeof(float), cudaMemcpyDeviceToHost);

  for(int i=0; i<N; i+=N/16)
        printf("%g\n",a[i]);

    cuCtxDestroy(cuContext);
    return 0;
}
```