Лекция 5

- PTX (*Parallel Thread eXecution*) ISA (*Instruction Set Architecture*).
- CUDA Driver API.

```
#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;
}
```

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