#ifndef DARKCUDA\_H

#define DARKCUDA\_H

#include "darknet.h"

#ifdef \_\_cplusplus

extern "C" {

#endif

extern int cuda\_debug\_sync;

extern int gpu\_index;

#ifdef \_\_cplusplus

}

#endif // \_\_cplusplus

#ifdef GPU

#define BLOCK 512

#define FULL\_MASK 0xffffffff

#define WARP\_SIZE 32

#define BLOCK\_TRANSPOSE32 256

#include <cuda.h>

#include <cuda\_runtime.h>

#include <curand.h>

#include <cublas\_v2.h>

#include <cuda\_runtime\_api.h>

//#include <driver\_types.h>

#ifdef CUDNN

#include <cudnn.h>

#endif // CUDNN

#ifndef \_\_DATE\_\_

#define \_\_DATE\_\_

#endif

#ifndef \_\_TIME\_\_

#define \_\_TIME\_\_

#endif

#ifndef \_\_FUNCTION\_\_

#define \_\_FUNCTION\_\_

#endif

#ifndef \_\_LINE\_\_

#define \_\_LINE\_\_ 0

#endif

#ifndef \_\_FILE\_\_

#define \_\_FILE\_\_

#endif

#ifdef \_\_cplusplus

extern "C" {

#endif // \_\_cplusplus

void check\_error(cudaError\_t status);

void check\_error\_extended(cudaError\_t status, const char \*file, int line, const char \*date\_time);

#define CHECK\_CUDA(X) check\_error\_extended(X, \_\_FILE\_\_ " : " \_\_FUNCTION\_\_, \_\_LINE\_\_, \_\_DATE\_\_ " - " \_\_TIME\_\_ );

cublasHandle\_t blas\_handle();

void free\_pinned\_memory();

void pre\_allocate\_pinned\_memory(size\_t size);

float \*cuda\_make\_array\_pinned\_preallocated(float \*x, size\_t n);

float \*cuda\_make\_array\_pinned(float \*x, size\_t n);

float \*cuda\_make\_array(float \*x, size\_t n);

void \*\*cuda\_make\_array\_pointers(void \*\*x, size\_t n);

int \*cuda\_make\_int\_array(size\_t n);

int \*cuda\_make\_int\_array\_new\_api(int \*x, size\_t n);

void cuda\_push\_array(float \*x\_gpu, float \*x, size\_t n);

//LIB\_API void cuda\_pull\_array(float \*x\_gpu, float \*x, size\_t n);

//LIB\_API void cuda\_set\_device(int n);

int cuda\_get\_device();

void cuda\_free\_host(float \*x\_cpu);

void cuda\_free(float \*x\_gpu);

void cuda\_random(float \*x\_gpu, size\_t n);

float cuda\_compare(float \*x\_gpu, float \*x, size\_t n, char \*s);

dim3 cuda\_gridsize(size\_t n);

cudaStream\_t get\_cuda\_stream();

cudaStream\_t get\_cuda\_memcpy\_stream();

int get\_number\_of\_blocks(int array\_size, int block\_size);

int get\_gpu\_compute\_capability(int i);

void show\_cuda\_cudnn\_info();

#ifdef CUDNN

cudnnHandle\_t cudnn\_handle();

enum {cudnn\_fastest, cudnn\_smallest, cudnn\_specify};

void cudnn\_check\_error\_extended(cudnnStatus\_t status, const char \*file, int line, const char \*date\_time);

#define CHECK\_CUDNN(X) cudnn\_check\_error\_extended(X, \_\_FILE\_\_ " : " \_\_FUNCTION\_\_, \_\_LINE\_\_, \_\_DATE\_\_ " - " \_\_TIME\_\_ );

#endif

#ifdef \_\_cplusplus

}

#endif // \_\_cplusplus

#else // GPU

//LIB\_API void cuda\_set\_device(int n);

#endif // GPU

#endif // DARKCUDA\_H