CSE490C Heterogeneous Parallel Computing

Lab assignment 1 - CUDA Vector Addition

This lab is based on the "GPU Teaching Kit Labs". The kit and associated lab are produced jointly by NVIDIA and University of Illinois (UIUC).

System and Software Requirements

You must use an NVIDIA CUDA-enabled GPU to use the compiled binaries.
The labs in the Teaching Kit require a CUDA supported operating system,
C compiler, and the CUDA Toolkit version 8 or later.
The CUDA Toolkit can be downloadedfrom the CUDA Download page.
Instructions on how to install the CUDA Toolkit are available in the
Quick Start page.
Aside from a C compiler and the CUDA Toolkit, CMake 3.17 or later is required
to generate build scripts for your target IDE and compiler.

I strongly recommend that you use the computing cluster at the department of computer science and/or the graduate school or artificial intelligence.
If you do not have access to either, you can request an account using the following form:
CSE cluster
AIGS cluster

Compile and running the lab

If you compile and run the lab, all the software required is already installed on the cluster.
Otherwise, the CUDA Toolkit and CMake must be installed.

1. Build libgputk
   The following procedure will build `libgputk` (the support library) that will be linked with your

template file for processing command-line arguments, logging time, and checking the correctness of your solution.

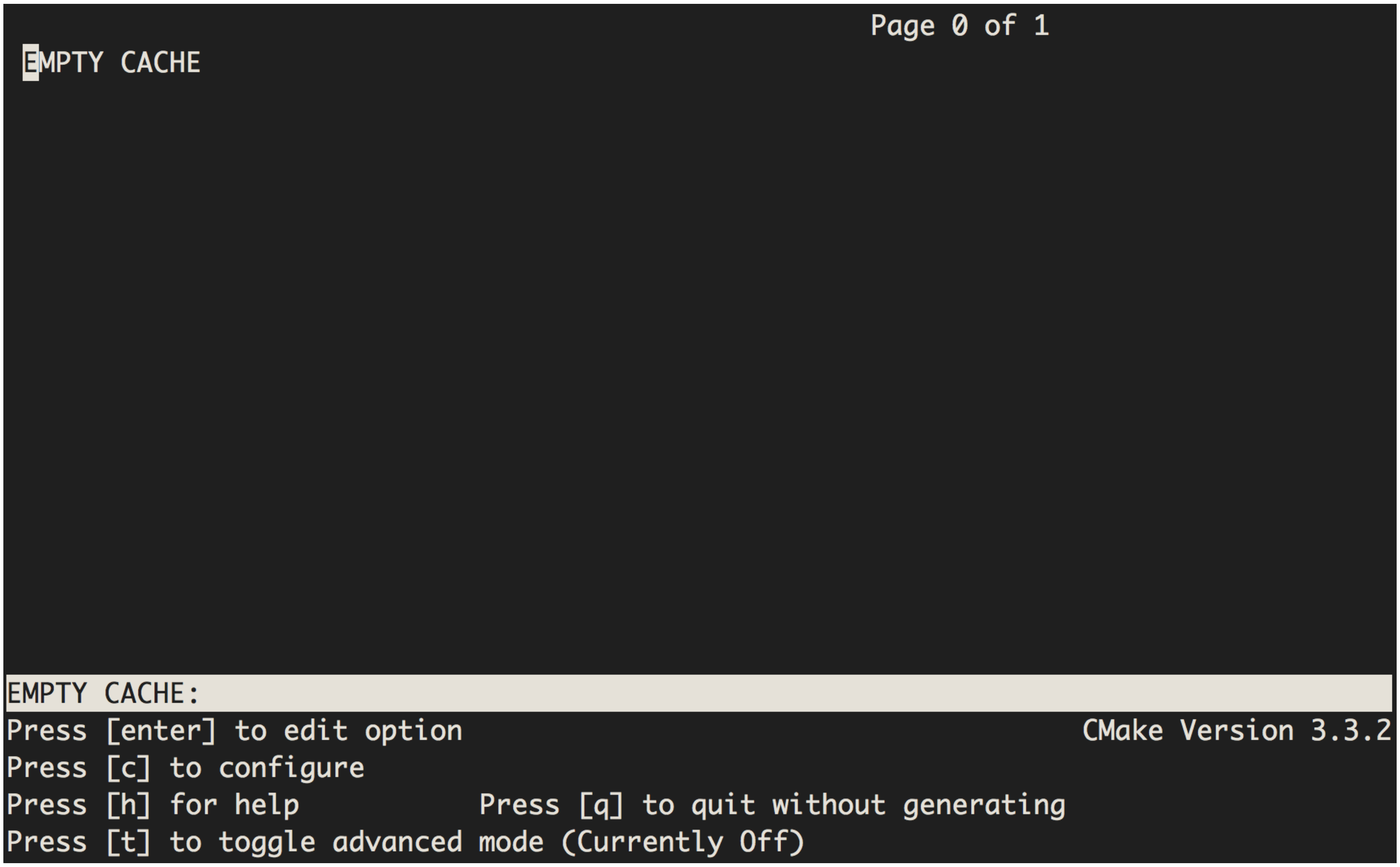
Create the target build directory

```
mkdir build
cd build
```

We will use `ccmake`

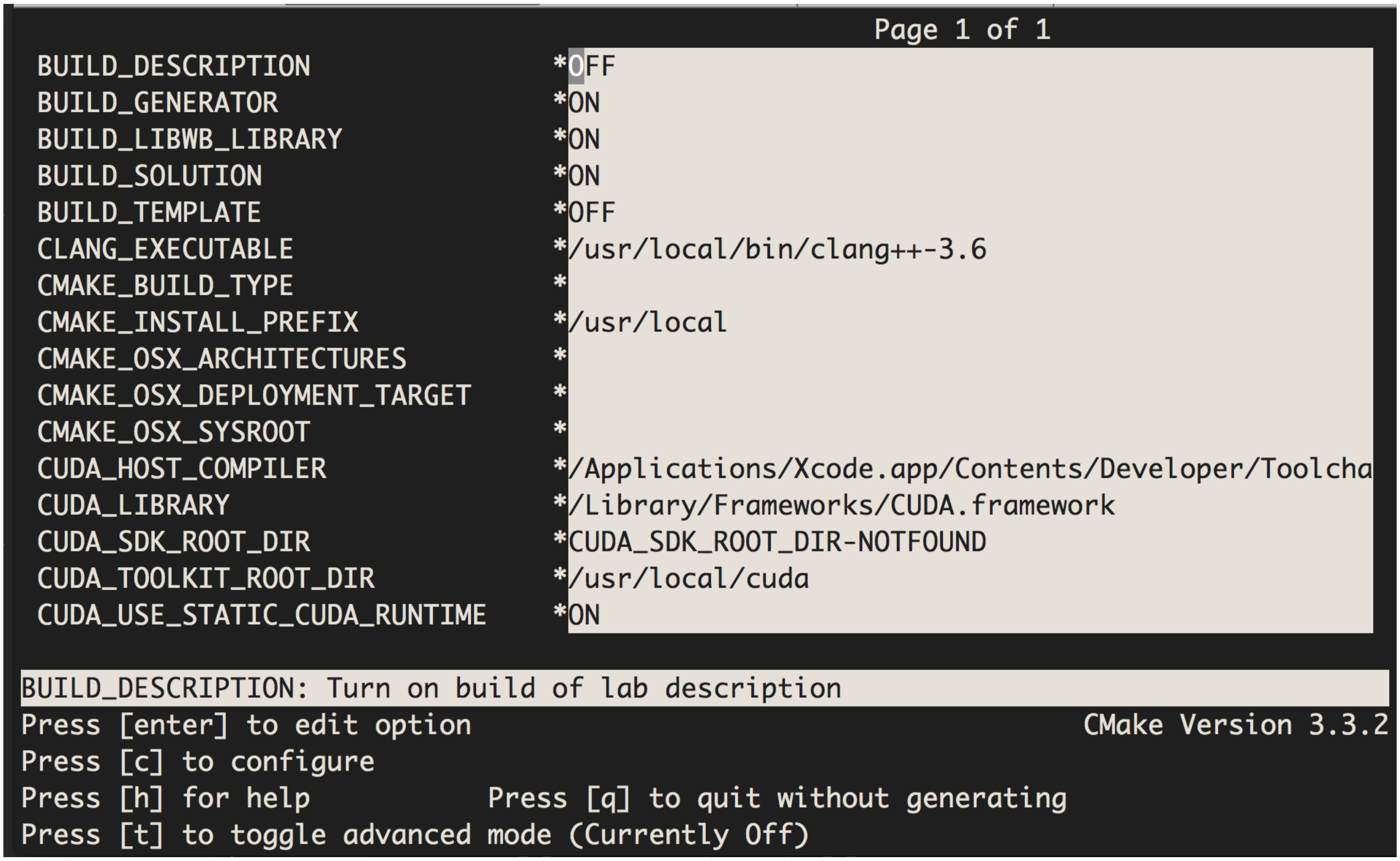
```
ccmake /path/to/Lab1
```

You will see the following screen



```
                                                      Page 0 of 1
 EMPTY CACHE










EMPTY CACHE:
Press [enter] to edit option                                    CMake Version 3.3.2
Press [c] to configure
Press [h] for help           Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

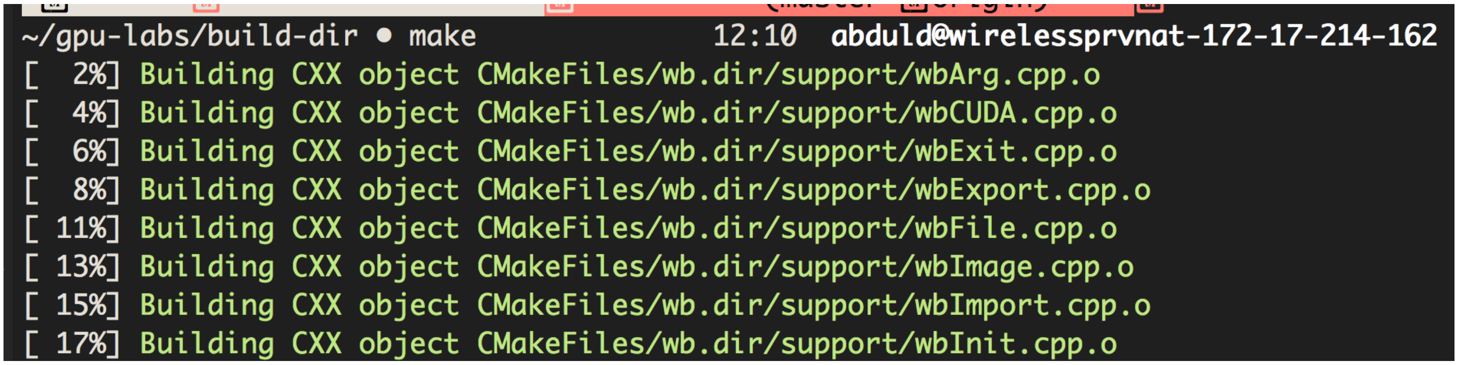
Pressing `c` would configure the build to your system (in the process detecting the compiler, the CUDA Toolkit location, etc...).



```
                                                         Page 1 of 1
 BUILD_DESCRIPTION                *OFF
 BUILD_GENERATOR                  *ON
 BUILD_LIBWB_LIBRARY              *ON
 BUILD_SOLUTION                   *ON
 BUILD_TEMPLATE                   *OFF
 CLANG_EXECUTABLE                 */usr/local/bin/clang++-3.6
 CMAKE_BUILD_TYPE                 *
 CMAKE_INSTALL_PREFIX             */usr/local
 CMAKE_OSX_ARCHITECTURES          *
 CMAKE_OSX_DEPLOYMENT_TARGET      *
 CMAKE_OSX_SYSROOT                *
 CUDA_HOST_COMPILER               */Applications/Xcode.app/Contents/Developer/Toolcha
 CUDA_LIBRARY                     */Library/Frameworks/CUDA.framework
 CUDA_SDK_ROOT_DIR                *CUDA_SDK_ROOT_DIR-NOTFOUND
 CUDA_TOOLKIT_ROOT_DIR            */usr/local/cuda
 CUDA_USE_STATIC_CUDA_RUNTIME     *ON

BUILD_DESCRIPTION: Turn on build of lab description
Press [enter] to edit option                                   CMake Version 3.3.2
Press [c] to configure
Press [h] for help            Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

```
 BUILD_LIBgpuTK_LIBRARY          *ON
 BUILD_LOGTIME                   *ON
```

If you have modified the above, then you should type `g` to regenerate the Makefile and then `e` to quit out of `ccmake` .
You can then use the `make` command to build the labs.



```
~/gpu-labs/build-dir ● make                    12:10  abduld@wirelessprvnat-172-17-214-162
[  2%] Building CXX object CMakeFiles/wb.dir/support/wbArg.cpp.o
[  4%] Building CXX object CMakeFiles/wb.dir/support/wbCUDA.cpp.o
[  6%] Building CXX object CMakeFiles/wb.dir/support/wbExit.cpp.o
[  8%] Building CXX object CMakeFiles/wb.dir/support/wbExport.cpp.o
[ 11%] Building CXX object CMakeFiles/wb.dir/support/wbFile.cpp.o
[ 13%] Building CXX object CMakeFiles/wb.dir/support/wbImage.cpp.o
[ 15%] Building CXX object CMakeFiles/wb.dir/support/wbImport.cpp.o
[ 17%] Building CXX object CMakeFiles/wb.dir/support/wbInit.cpp.o
```

2. Build the data generator and template

The following will compile the template file that you will modify to implement vector addition in CUDA, and the data generator that will generate input files.

```
cd sources
make template
make dataset_generator
```

You can generate input data with

~~

./dataset_generator

~~

This will create a directory that contains multiple pairs of input data. You can modify the file to generate input data of different sizes.