Nishant Kumar / 2017171

Simulation assignment-2

This assignment asked us to Create and simulate our own Simobject that performs the inverse of a given input and also to create two debug flags that output the input matrix and its size and the resultant matrix.

What are the Debug flags?

- Debug flags are functionality included in gem5 to provide a better-debugging experience. These flags allow every component to have many debug-print statements, without all of them enabled at the same time. When running gem5, we can specify which debug flags to enable from the command line.

How is the inversion of a matrix is done?

- Inversion of a matrix is done by dividing its determinant by its adjoint.
- For this, the determinant of the matrix should be non-zero.
- Also, the matrix should be square.

The basic flow of creation and simulation of a custom SimObject.

- Create the python class of the sim object where you declare its type and its wrapped C++ implementation. Since it's a sim object, it also inherits the rest of its functionality from a base sim object.
- Generating the header file and the C++ file for the sim object. C++ file carries the actual functionality of the sim object
- Registering the C++ file, header file, and the python implementation in gem5. This is done using Sconscript since gem5 uses scons as its build command.
- Finally, rebuilding gem5 to include our sim object and creation of a config script to run the simulation.

Nishant Kumar / 2017171

File wise descriptions.

Attached files are:

Sconsript
invert_mat.cc
invert_mat.hh
run_invert.py
InvertMat.py

InvertMat.py:

- Creates the python class associated with the sim object (invert matrix).
- This file describes the parameters of the SimObject.
- Here we declare the sim objects type and the location of its C++ header file that defines the C++ class for the sim object.
- Here "type" is the C++ class that is being wrapped with this python object and cxx_header contains the declaration of the class which is used as the type parameter.

invert_mat.hh:

- This file creates the header file for the above defined C++ class.
- This file defines the InvertMat sim object and its inheritance as a sim object.
- In this file, we first use *#ifndef* to check if there is already any implementation-defined at invert_mat.hh, if we didn't find any then we define it using #define.
- After all this we include the header files for the sim object from where we have to declare the inheritance since our sim object is inserting all the parameters of its from the sim object in gem5.

invert_mat.cc:

- This is the C++ file that actually has the whole implementation and functioning of the entire object. In short, this controls the whole flow of the sim object functioning.
- According to the task given, I have implemented two DEBUG flags MATRIX and RESULT. MATRIX debug flag prints the size of the input matrix and its elements. RESULT debug flag prints the output matrix's (the inverted matrix's) elements.
- The entire inversion is being done by the function named "INV".
- The create() function that is implemented as a separate function is important for the generation of the sim object. This function returns a new instantiation of the sim object. Not adding this will give a LINKER ERROR.

- Since we have already declared the debug flags here we also have to include their header files to use them in our code. Inversion of the matrix is done by first finding its determinant and its adjoint and then computing "adjoint_of_matrix/float(determinant)".

Sconscript:

- This file registers the C++ implementation and the python class we built for the Simobject with the gem5 build system.
- This file simply generates a sim object with the name as our python file and mentions its source implementation in the next line as source("invert_mat.cc").
- This file also decares the debug file that we are creating in this case I am declaring two debug flags as DebugFlag('MATRIX') and DebugFlag('RESULT').

run_invert.py:

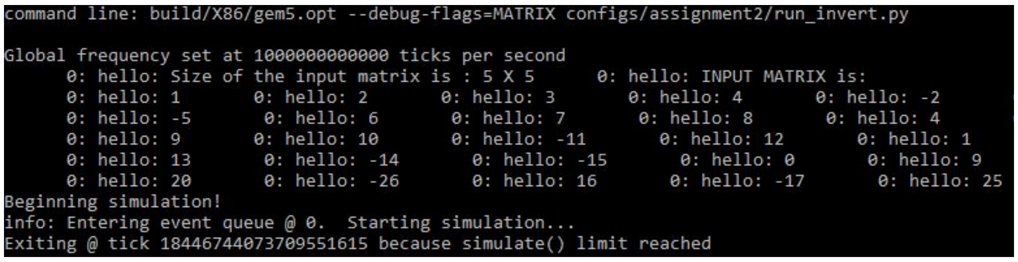
- This file is the python configuration script to instantiate our Simobject. It imports all the sim objects from m5.objects and then instantiates our root object.
- Then it makes our sim object as the child of the root object and instantiates it.
- Finally by calling m5.instantiate, we run the simulation (Simulation is whatever is being implemented in invert_mat.cc).

Steps to run the simulation:

- Copy all files except run_invert.py into a folder in the src directory of the gem5 folder.
- Copy run_invert.py into the configs folder
- Build gem5 to make it include our sim object using the command "scons build/X86/gem5.opt -j7"
- Run the simulation using build/X86/gem5.opt configs<path to the config file>
- To use the debug functionality, before configs in the above command insert --debug-flags=MATRIX/RESULT

Outputs:

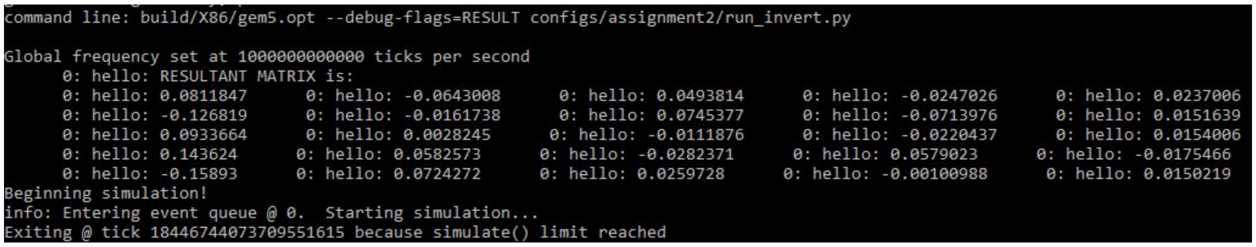
- Using the Debug flag=MATRIX would print the input matrix and its size:



```
command line: build/X86/gem5.opt --debug-flags=MATRIX configs/assignment2/run_invert.py

Global frequency set at 1000000000000 ticks per second
      0: hello: Size of the input matrix is : 5 X 5      0: hello: INPUT MATRIX is:
      0: hello: 1       0: hello: 2       0: hello: 3       0: hello: 4       0: hello: -2
      0: hello: -5       0: hello: 6       0: hello: 7       0: hello: 8       0: hello: 4
      0: hello: 9       0: hello: 10       0: hello: -11       0: hello: 12       0: hello: 1
      0: hello: 13       0: hello: -14       0: hello: -15       0: hello: 0       0: hello: 9
      0: hello: 20       0: hello: -26       0: hello: 16       0: hello: -17       0: hello: 25
Beginning simulation!
info: Entering event queue @ 0.  Starting simulation...
Exiting @ tick 18446744073709551615 because simulate() limit reached
```

- Using the Debug flag=RESULT would print the resultant matrix :



```
command line: build/X86/gem5.opt --debug-flags=RESULT configs/assignment2/run_invert.py

Global frequency set at 1000000000000 ticks per second
      0: hello: RESULTANT MATRIX is:
      0: hello: 0.0811847      0: hello: -0.0643008      0: hello: 0.0493814      0: hello: -0.0247026      0: hello: 0.0237006
      0: hello: -0.126819      0: hello: -0.0161738      0: hello: 0.0745377      0: hello: -0.0713976      0: hello: 0.0151639
      0: hello: 0.0933664      0: hello: 0.0028245      0: hello: -0.0111876      0: hello: -0.0220437      0: hello: 0.0154006
      0: hello: 0.143624      0: hello: 0.0582573      0: hello: -0.0282371      0: hello: 0.0579023      0: hello: -0.0175466
      0: hello: -0.15893      0: hello: 0.0724272      0: hello: 0.0259728      0: hello: -0.00100988      0: hello: 0.0150219
Beginning simulation!
info: Entering event queue @ 0.  Starting simulation...
Exiting @ tick 18446744073709551615 because simulate() limit reached
```