VectorOperation.py

In this python file we simply created the python class for our SimObject named VectorOperation, For our SimObject we need to declare it's name in type variable and the C++ class for our SimObject is defined in C++ header file.the python class also tells parameters of ours SimObject can be controlled from the python configuration files.

The "type" in the file is the c++ class that you are wrapping with your python SimObject

The "cxx_header" file that have the declaration of the class used as the type parameter

The "cxx_class" is a declared attribute in the gem5 namespace that specifies the newly created SimObject.

This python file VectorOperation.py is in src/learning_gem5/part2.

VectorOperation.hh

To implement the VectorOperation we needed to create VectorOperation.hh and VectorOperation.cc in src/learning_gem5/part2.

In order to prevent circular includes, gem5 encapsulates all header files in #ifndef /#endif with the name of the file and the directory it is in.

We need to declare our class so, VectorOperation is a simobject so it will inherit the c++ simobject class

We create a constructor for a vectorOperation simobject which takes parameters based on the python class,name of the parameter is "VectorOperation" and the type name is "VectorOperationParams".

This file also contains the three events we need to find the "VectorAddition", "VectorDotProduct", "VectorNormalize" for the bonus part i have also created to extra events for taking inputs for the vectors and for the cycle ticks, for the vector input i created "VectorUserInput" and for the cycles ticks inputs i created "VectorTicks" all these events are declared in private scope.

Every event will execute at a specific tick and the startup function is used to schedule the given events in the assignment.

In this header file we declared these 5 functions in private scope, now we want to execute the EventFunctionWrapper which is used to wrap the events which takes two parameters, a function to execute and a name. These functions take no input parameters and return no value as specified in assignment; these events do not store the resultant values in the code, it will compute the values each time when the events are called.

VectorOperation.cc

All the functions I created in this .cc file, let's start with the constructor for VectorOperation, in which we pass the parameter objects of all the event names.

Now we made two vectors of 4x1 with the names x and y. We capture [this] in the lambda that calls member function instances of the class, and we will call event,event2,event3,event4,event5 defined in the header file under EventFunction Wrapper.

In the startup() function the scheduling events will be done. We use the "cppschedule" function which will initially schedule using the schedule(event,ticks) then we add scheduled events to the VectorOperation class. As given in the assignment VectorAddition will happen at 100 ticks,

VectorDotProduct will at 1000 ticks and VectorNormalize at 10000 ticks. Event 4 for vector input at tick 10 **for bonus part**

In VectorAdditon function i have created a new vector “addr” of 4x1 which gonna store the result of the sum of two global vectors x and y, then i have find out the sum of two vector and using for loop and in end print the values of the “addr” vector using DPRINTF function.

In VectorDotProduct i have variable “res” which gonna store the dot product result of the two global vector x and y, then i had run for loop to find out the dot product of two vector and store it in res and in end print the value of the “res” variable using DPRINTF function.

In VectorNormalize i have created two vector norr1 and norr2 of 4x1 for storing the normalized form of x and y vector and two variable dx and dy to store d for vector x and y ,then i had run 1st for loop for finding the value of dx and dy then square rooting them, then comes the 2nd for loop to find the normalized form and storing in norr1 and norr2, then in the end printing the values of both norr1 and norr2 vector or i will say the normalized form of x and y using DPRINTF.

In this file we also create debug flags and declare them in the SConscript file. We also need to include the header file in this cc file like “debug//name of debug.hh”. I have created 4 debug flags. 1st is “VECTOR” this debug flag is printing the input vector, 2nd is “ADDRESULT” this debug flag prints the result of addition of two vectors stored in “addr”, 3rd is “DOTRESULT” which prints the result “res” of dot product of two vectors x and y. 4th is “NORMALIZE” this prints the normalized form of the vector x and y basically printing the values of norr1 and norr2.

For bonus

1) I have declared 3 global variables a1,a2,a3 which will hold the tick value entered by the user and these ticks value will be used for events. I had taken input for a1,a2,a3 in startup() function using std::cin function then using these entered values in schedule(event,latency) as latency for events like “event,event2,event3” for addition, dot product and normalization.
2) I have declared the function “VectorUserInput” which takes input from the user for both the vector x and y. Using functions like cin, cout and for loops and in end printing both the input vectors.

SConscript

Registering the SimObject and c++ file. In order to compile c++ and python file we need to tell the build system about these files. We declare the simobject “VectorOperation” and .cc file “source(VectorOperation.cc)” and i also include the debug flags in this file (“VECTOR”,”ADDRESULT”,”DOTRESULT”,”NORMALIZE”) like given in the assignment.
This file will be in src/learning_gen5/part2

run_aman.py

To instantiate the object we need to make a run_aman.py file, this file will be in configs/learning_gem5/part2. In this file first we will import m5 and all objects we have compiled,

then we set the root SimObject, then root.Vector creates the instantiation of the VectorOperation SimObject. Then in the end call instantiate in the m5 module and run the simulation.

To compile or run the code:

run git checkout command first then, add VectorOperation.py, VectorOperation.cc, VectorOperation.hh and SConscript in src/learnging_gem5/part2 and the run_aman.py in config/learnging_gem5/part. Then after that re-run the build command which is “scons build/X86/gem5.opt”

Then run this command:(in gem5 using cd gem5)
build/X86/gem5.opt --debug-flags=VECTOR,ADDRESULT,DOTRESULT,NORMALIZE configs/learning_gem5/part2/run_aman.py

Code output sample:

```
aman@ubuntu:~/gem5$ build/X86/gem5.opt --debug-flags=VECTOR,ADDRESULT,DOTRESULT,NORMALIZE config
s/learning_gem5/part2/run_aman.py
gem5 Simulator System.  https://www.gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 22.0.0.2
gem5 compiled Nov  4 2022 06:18:07
gem5 started Nov  4 2022 06:26:50
gem5 executing on ubuntu, pid 3024
command line: build/X86/gem5.opt --debug-flags=VECTOR,ADDRESULT,DOTRESULT,NORMALIZE configs/lear
ning_gem5/part2/run_aman.py

Global frequency set at 1000000000000 ticks per second
warn: No dot file generated. Please install pydot to generate the dot file and pdf.
Beginning simulation!
enter tick for addition
200
enter tick for dotproduct
2000
enter tick for normalization
5000
build/X86/sim/simulate.cc:194: info: Entering event queue @ 0.  Starting simulation...
enter inputs for vector1 1 2 3 4
enter inputs for vector2 9 8 7 6
     10: Vector: vector1 is 1 2 3 4
     10: Vector: vector2 is 9 8 7 6
    200: Vector: Vectors addition result= 10 10 10 10
   2000: Vector: dot product result is 70
   5000: Vector: Vector1 normalize form= 0.182574 0.365148 0.547723 0.730297
   5000: Vector: Vector2 normalize form= 0.593442 0.527504 0.461566 0.395628
Exiting @ tick 18446744073709551615 because simulate() limit reached
aman@ubuntu:~/gem5$
```