README FILE

GEM5 EXECUTION STEPS

Files Found
Linearcomplete.c
Linearoverhead.c
Linear.sh
Randcomplete.c
Randoverhead.c
Random.sh
stat.txt

Understanding the Files
In order to do the cache access time measurements, we have 2 files – xxxcomplete.c files which are the complete code. Xxxoverhead.c files are files that measure only overheads in measuring average access times. Thus, average access ticks are calculated as follows

Access_Time_Ticks = (Ticks_complete – Ticks_overheads)/Array_Size

The xxxx.sh scripts are using to compile various C codes and run the gem5 syscall simulator for multiple array sizes.

Steps to Execute
1. After having unzipped the Assignment zip file, we need to update the paths in all the shell scripts. The instructions are as follows for Gem5 scripts

    I.   Update the path to C code in the Gcc command on line 2 and 24.
        gcc -o Linearcomplete -Wall ==~/Desktop/CAO2_Assignment1/cse520-Trivedi-S_assgn01/Gem5/==Linearcomplete.c

    II.   Update the simulator and source code path to the GEM5 simulator command.

==~/gem5/build/X86/==gem5.opt --outdir===~/gem5==/tests ~/gem5/configs/example/se.py --cpu-clock=1GHz --cpu-type=DerivO3CPU --caches --l1i_size=32kB --l1i_assoc=2 --l1d_size=64kB --l1d_assoc=4 --l2cache --l2_size=256kB --l2_assoc=16 -c ==~/Desktop/CAO2_Assignment1/cse520-Trivedi-S_assgn01/Gem5==/Linearcomplete –options=1
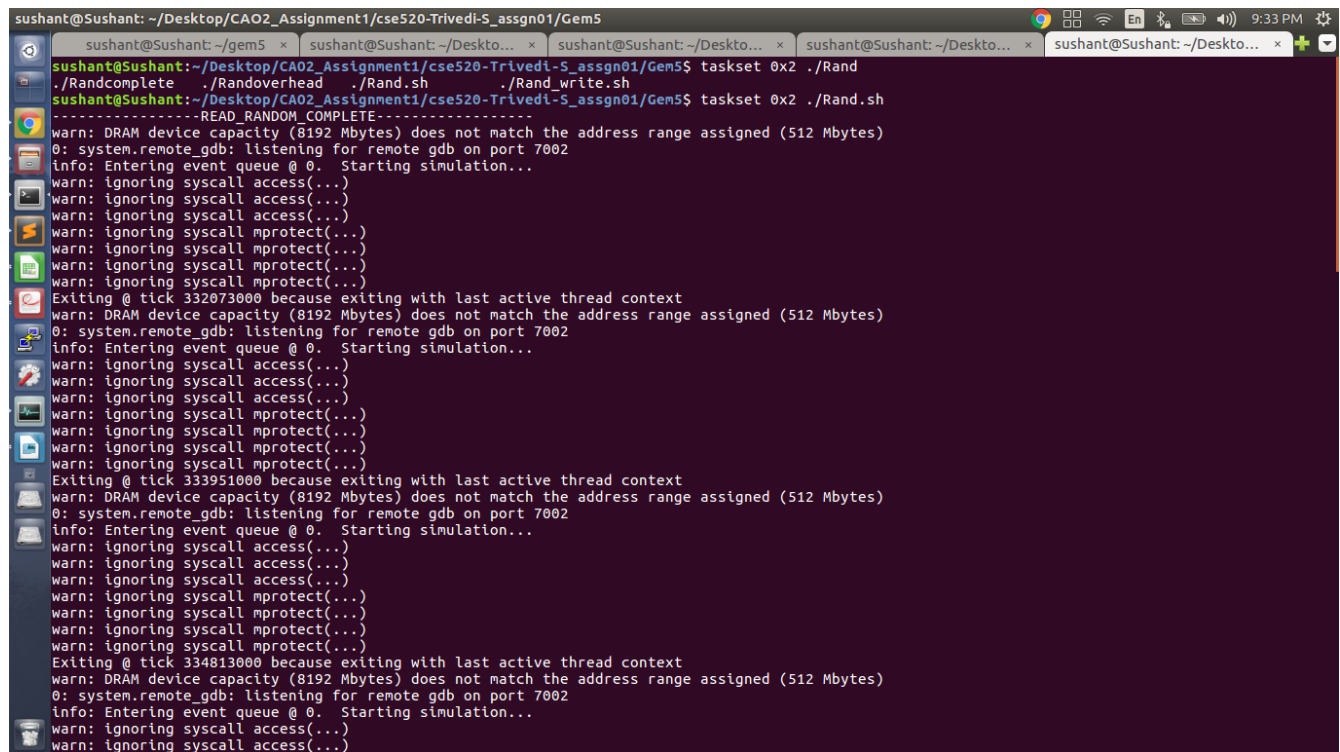
In the above command update the highlighted parts with the correct path. This would be required to be done in all the Gem5 commands, we have multiple commands for various array sizes.

2. Having updated the scripts with correct paths to simulator and source code. Use the following command to call the script from the directory it is in. We need to use the taskset command to make sure it runs on one specific Core.
Assuming we are already in /cse520-Trivedi-S_assign01/Gem5/ directory, run

taskset 0x1 ./Random.sh
or
taskset 0x1 ./Linear.sh

These script would measure the read access times for Random and Linear Access times.

Sample Execution of the file.

GEM5 CONFIGURATION AS PER ASSIGNMENT
The assignment required us to configure GEM5 with the following settings.

CLOCK EXECUTION STEPS

Files Found
Linear.sh → contains script to execute the measurements for linear access read and writes
Random.sh → contains script to execute the measurements for random access read and write
Clock_linear.c → C code for Linear Access
Clock_random.c → C code for Random Access

Understanding the Files

The shell compiles the GCC code to generate the executable and runs it by passing it parameters as follows:
./executable Array_Size (1 – Read or 2 – Write)
eg: ./Clock_linear 65536 1

Linear Access:
We have simulated the linear access using multiple strides to assess the memory mountain.
Please modify stride variable on line 87 of Clock_linear.c

Steps to Execute

1. In order to run, you need to unzip the folder and navigate to Clock folder within the zip.
2. Once within Clock directory, execute Linear.sh or Random.sh to get results.

taskset 0x1 ./Linear.sh
taskset 0x1 ./Random.sh

3. This would print output in the following format. First the read measurements followed by write measurements.

Array Size Time(ns)

eg output:

RANDOM_READ
1 3.000000 ns
2 2.500000 ns
..
..
..
RANDOM_WRITE
1 3.000000 ns
2 2.500000 ns

PERF EXECUTION STEPS

Files Found
PerfLin.sh
PerLin_complete.c
PerfLin_overhead.c
PerfRand.sh
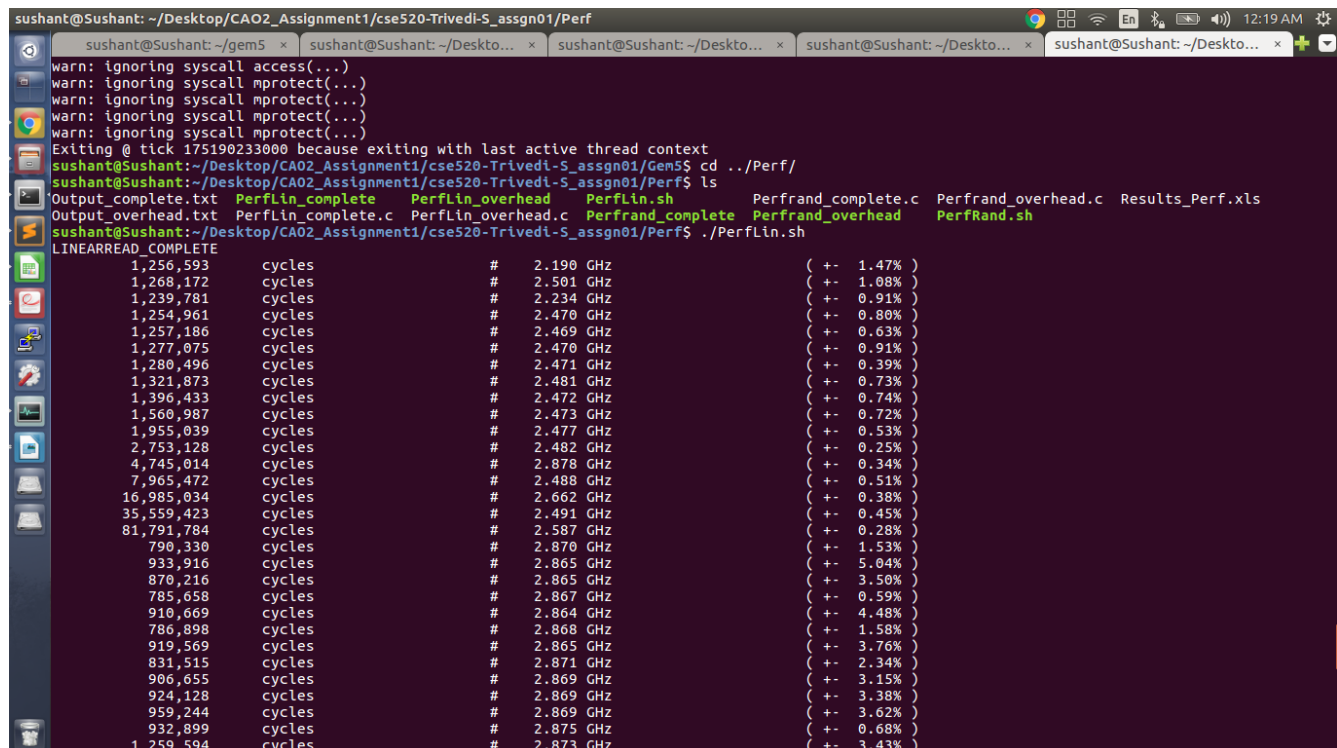PerRand_Complete.c
PerfRand_overhead.c

Understanding the File

The script executed the read complete, then read overhead, write_complete and write_overhead measurements. It just needs to be executed. The second parameter passes to files in argv param is whether it is read→1 or write→2. First parameter is the array size

Steps to Execute
1. Update the paths as explained earlier for the different shell files and then execute the shell files. They would output as follows

Execute each script on a specific CPU using taskset
taskset 0x1 ./Script_name

**NOTE: Sometimes the scripts may give certain permission denied error. Then do the following
chmod +x ./script_name**

GEM5 simulator configuration

CPU type:
L1 instruction cache:
L1 data cache:
L2 unified cache:
DerivO3CPU
2-way set-associate, total size 32KB, LRU
4-way set-associate, total size 64KB, LRU
16-way set-associate, total size 256KB, LRU

cacheline_size=64 bytes.
CPU in the simulator running in 1000MHz

These have been done in the script file by means of various flags as shown below

~/gem5/build/X86/gem5.opt --outdir=~/gem5/tests ~/gem5/configs/example/se.py --cpu-clock=1GHz --cpu-type=DerivO3CPU --caches --l1i_size=32kB --l1i_assoc=2 --l1d_size=64kB --l1d_assoc=4 --l2cache --l2_size=256kB --l2_assoc=16 -c ~/Desktop/CAO2_Assignment1/cse520-Trivedi-S_assgn01/Gem5/Linearcomplete --options=1 1