

Cache simulator

Language : C++

Implementation

-Basic CPU

CPU class, which have cpu id, cache, bus, dram, cycles, etc..

-Read line(from data files)

Read label and data. Input of cpu execution

-Cache access (if fail, mem access)

Request data to cache if not, target cycles change for mem access

-Bus access

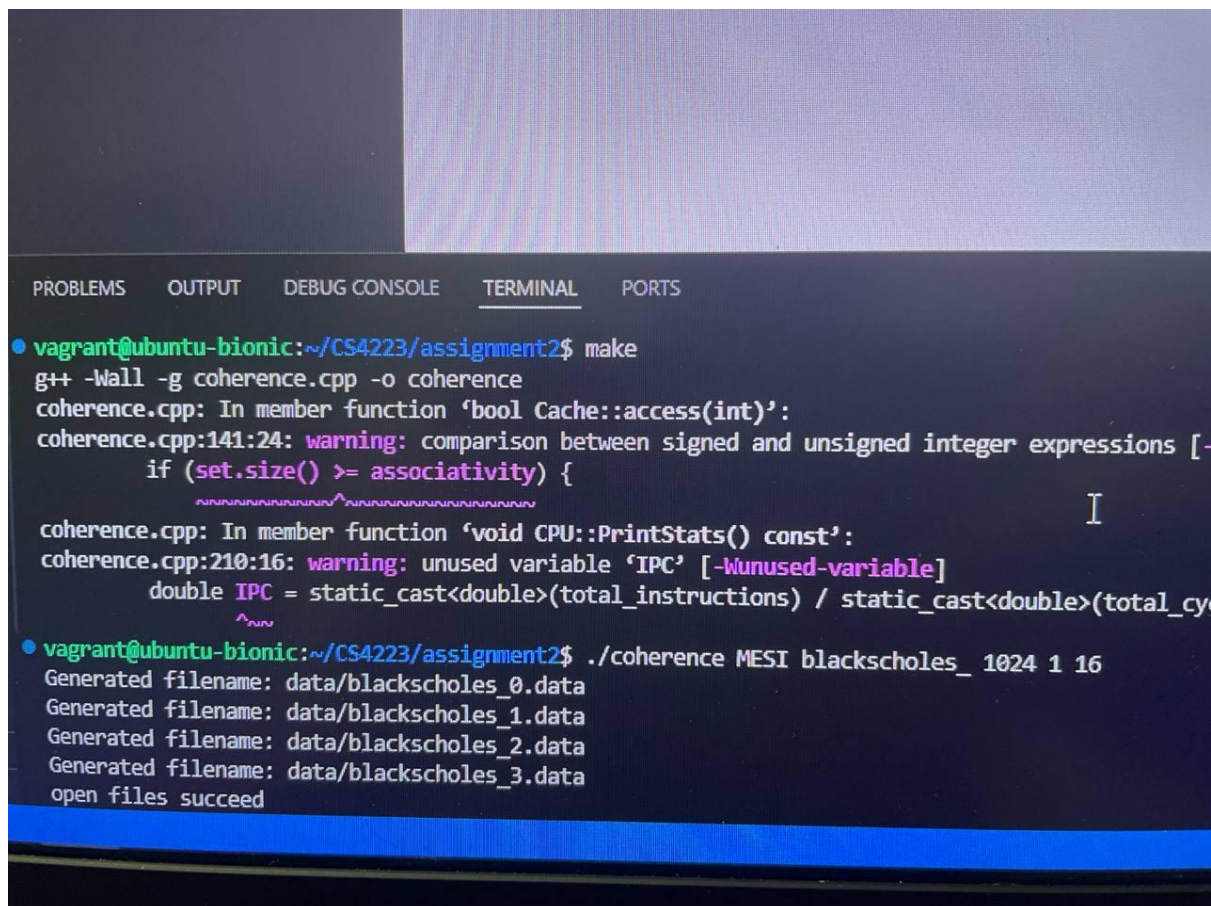
Not implemented yet.

-Compute cycles

Add cycles for an instruction to total cycles

Input example

Just "make" then `"/coherence MESI blackscholes_1024 1 16"`



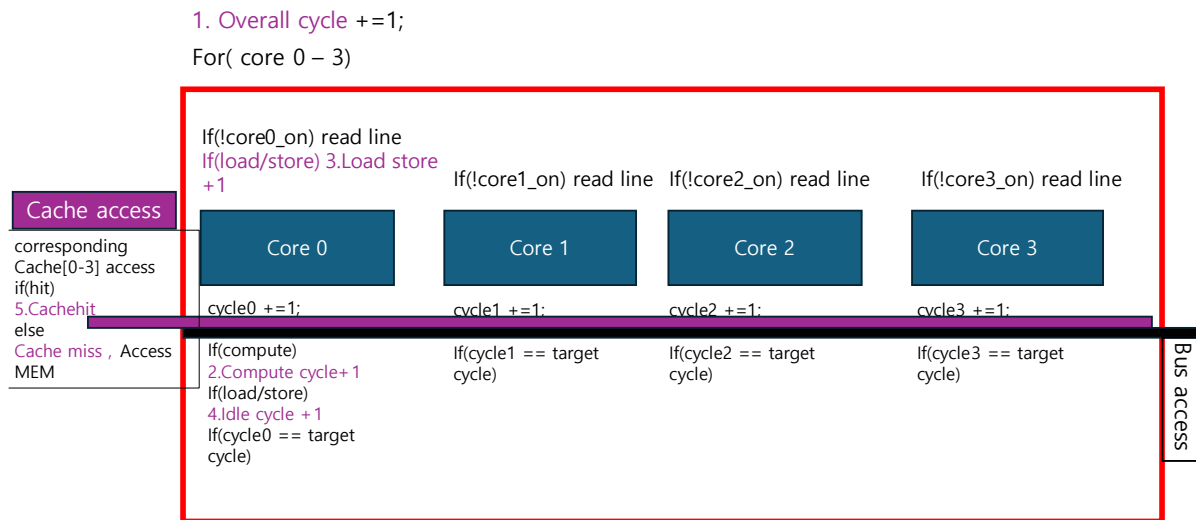
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

● vagrant@ubuntu-bionic:~/CS4223/assignment2$ make
g++ -Wall -g coherence.cpp -o coherence
coherence.cpp: In member function 'bool Cache::access(int)':
coherence.cpp:141:24: warning: comparison between signed and unsigned integer expressions [-Wsign-compare]
    if (set.size() >= associativity) {
                        ^
coherence.cpp: In member function 'void CPU::PrintStats() const':
coherence.cpp:210:16: warning: unused variable 'IPC' [-Wunused-variable]
    double IPC = static_cast<double>(total_instructions) / static_cast<double>(total_cycles);
               ^
● vagrant@ubuntu-bionic:~/CS4223/assignment2$ ./coherence MESI blackscholes_ 1024 1 16
Generated filename: data/blackscholes_0.data
Generated filename: data/blackscholes_1.data
Generated filename: data/blackscholes_2.data
Generated filename: data/blackscholes_3.data
open files succeed
```

Output

- Overall cycles
- Compute cycles
- # of Load/store
- Idle cycles
- Cache hit / miss

Process



But we only implemented Single Core yet.

Cycles detail

Cycles

Total_cycles : timing, count cycles until end of instructions

Compute_cycles: count label==2(compute) cycles.

Total_instructions : total instructions, # of lines in a file

Num_Is : # of load/store inst

Cache_hit : cache hit

Cache_miss : cache miss

You can trace

Cache_miss*100(DRAM latency) + cache hit * 1 (hit takes 1 cycle) + compute_cycles = total_cycles.