|  |  |
| --- | --- |
| **Architetture dei Sistemi di Elaborazione 02GOLOV [M-Z]** | Delivery date:  12/11/2020 |
| **Laboratory**  **4** | Expected delivery of lab\_04.zip must include:   * this document compiled in pdf format. |

1. Getting started with gem5

gem5 is an event-driven simulator freely available at: <http://gem5.org/>

The laboratory version uses the ALPHA CPU model previously compiled.

From Portale della Didattica, download the gem5\_env\_2020.zip. Decompress it in your home directory.

**NOTE: All the commands shown here must be executed from the terminal.**

Preliminarily, set up the environment variables executing the following command:

source start.sh (**NOTE, if you are using the VBox VM, replace this command with** source start\_vbox.sh**)**

**The effects of these scripts are visible only in the current shell.**

labinf@ubuntu-desktop:~/Desktop/gem5\_env\_2020$ source start.sh

Setting up the environment...

* 1. Write a hello world C program (hello.c). Then compile the program, using the ALPHA compiler with the command gem5\_alpha\_compiler. The compiler is the gcc version for the ALPHA ISA, therefore it is used with the same options:

|  |
| --- |
| labinf@ubuntu-desktop:~/Desktop/gem5\_env\_2020$ gem5\_alpha\_compiler -static -o hello hello.c |

* 1. Then simulate the program with the gem5\_sim command as follows:

|  |
| --- |
| labinf@ubuntu-desktop:~/Desktop/gem5\_env\_2020$ gem5\_sim $GEM5\_DEFAULT\_PY -c hello |

In this simulation, gem5 uses *AtomicSimpleCPU* by default.

* 1. Check the results

your simulation output should be similar than the one provided in the following:

|  |
| --- |
| labinf@ubuntu-desktop:~/Desktop/gem5\_env\_2020$ gem5\_sim $GEM5\_DEFAULT\_PY -c hello  gem5 Simulator System. http://gem5.org  gem5 is copyrighted software; use the --copyright option for details.  gem5 compiled Jan 17 2019 11:54:22  gem5 started Oct 30 2020 11:29:05  gem5 executing on ubuntu-desktop, pid 31977  command line: gem5.opt /opt/gem5/configs/example/se.py -c hello  /opt/gem5/configs/common/CacheConfig.py:50: SyntaxWarning: import \* only allowed at module level  def config\_cache(options, system):  Global frequency set at 1000000000000 ticks per second  warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (512 Mbytes)  warn: Breakpoints do not work in Alpha PAL mode.  See PCEventQueue::doService() in cpu/pc\_event.cc.  0: system.remote\_gdb: listening for remote gdb on port 7000  \*\*\*\* REAL SIMULATION \*\*\*\*  info: Entering event queue @ 0. Starting simulation...  info: Increasing stack size by one page.  Hello There!!  Exiting @ tick 2411000 because exiting with last active thread context |

* + - Check the output folder

in your working directory, gem5 creates an output folder (m5out), and saves there 3 files: config.ini, config.json, and stats.txt. In the following, **some examples** of the produced files are reported.

* + - Statistics (stats.txt)

|  |
| --- |
| ---------- Begin Simulation Statistics ----------  sim\_seconds 0.000003 # Number of seconds simulated  sim\_ticks 2623000 # Number of ticks simulated  final\_tick 2623000 # Number of ticks from beginning of simulation  sim\_freq 1000000000000 # Frequency of simulated ticks  host\_inst\_rate 1128003 # Simulator instruction rate (inst/s)  host\_op\_rate 1124782 # Simulator op (including micro ops) rate(op/s)  host\_tick\_rate 564081291 # Simulator tick rate (ticks/s)  host\_mem\_usage 640392 # Number of bytes of host memory used  host\_seconds 0.00 # Real time elapsed on the host  sim\_insts 5217 # Number of instructions simulated  sim\_ops 5217 # Number of ops (including micro ops) simulated  ... ... ...  system.cpu\_clk\_domain.clock 500 # Clock period in ticks  ... ... ... |

* + - Configuration file (config.ini)

|  |
| --- |
| ... ... ...  [system.cpu]  type=AtomicSimpleCPU  children=dtb interrupts isa itb tracer workload  branchPred=Null  checker=Null  clk\_domain=system.cpu\_clk\_domain  cpu\_id=0  default\_p\_state=UNDEFINED  do\_checkpoint\_insts=true  do\_quiesce=true  do\_statistics\_insts=true  dtb=system.cpu.dtb  eventq\_index=0  fastmem=false  function\_trace=false |

1. Simulate the same program using different CPU models.

Help command:

|  |
| --- |
| 1. labinf@ubuntu-desktop:~/Desktop/gem5\_env\_2020$ gem5\_sim $GEM5\_DEFAULT\_PY -h |

List the CPU available models:

|  |
| --- |
| labinf@ubuntu-desktop:~/Desktop/gem5\_env\_2020$ gem5\_sim $GEM5\_DEFAULT\_PY --list-cpu-types |

* 1. *TimingSimpleCPU* simple CPU that includes an initial memory model interaction

|  |
| --- |
| labinf@ubuntu-desktop:~/Desktop/gem5\_env\_2020$ gem5\_sim $GEM5\_DEFAULT\_PY --cpu-type=TimingSimpleCPU -c hello |

* 1. *MinorCPU* the CPU is based on an in order pipeline including caches

|  |
| --- |
| labinf@ubuntu-desktop:~/Desktop/gem5\_env\_2020$ gem5\_sim $GEM5\_DEFAULT\_PY --cpu-type=MinorCPU --caches -c hello |

* 1. *DerivO3CPU* is a superscalar processor

|  |
| --- |
| labinf@ubuntu-desktop:~/Desktop/gem5\_env\_2020$ gem5\_sim $GEM5\_DEFAULT\_PY --cpu-type=DerivO3CPU --caches -c hello |

To practice with the generated statistics, create a table (TABLE1) gathering for each simulated CPU the following statistics (**when available!**):

* sim\_ticks (Number of ticks simulated)
* sim\_insts (Number of instructions simulated)
* system.cpu.numCycles (Number of CPU Clock Cycles)
* system.cpu.cpi (Clock Cycles per Instruction)
* system.cpu.committedInsts (Number of instructions committed)
* host\_seconds (Host time in seconds)
* system.cpu.fetch.Insts (Number of instructions Fetch Unit has encountered)

TABLE1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameters | AtomicSimpleCPU | TimingSimpleCPU | MinorCPU | DeriveO3CPU |
| sim\_ticks | 2755000 | 356448000 | 31495500 | 18533000 |
| sim\_insts | 5477 | 5477 | 5490 | 5278 |
| system.cpu.numCycles | 5511 | 712896 | 62991 | 37067 |
| system.cpu.cpi | 1.006207 | 130.161767 | 11.473770 | 7.022925 |
| system.cpu.committedInsts | 5477 | 5477 | 5490 | 5278 |
| host\_seconds | 0.01 | 0.02 | 0.03 | 0.14 |
| system.cpu.fetch.Insts | - | - | - | 11124 |

**NOTE: When not available compute the CPI using the formula:**

1. Let’s now switch to a slightly more complex benchmark: the computation of a Fast Fourier Transform. The program is in the *benchmarks/fft* subdirectory and can be compiled using the MakeFile with the commands make clean and then make that will produce as output the fft executable file.

Simulate the program using the gem5 CPU models seen before and collect the following information (when available!) filling TABLE 2:

* sim\_ticks (Number of ticks simulated)
* sim\_insts (Number of instructions simulated)
* system.cpu.numCycles (Number of CPU Clock Cycles)
* system.cpu.cpi (Clock Cycles per Instruction)
* system.cpu.committedInsts (Number of instructions committed)
* host\_seconds (Host time in seconds)
* system.cpu.fetch.Insts (Number of instructions Fetch Unit has encountered)
* Prediction ratio for Conditional Branches: system.cpu.branchPred.condIncorrect / system.cpu.branchPred.condPredicted
* system.cpu.branchPred.BTBHits (Number of BTB hits)

TABLE2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameters | AtomicSimpleCPU | TimingSimpleCPU | MinorCPU | DeriveO3CPU |
| sim\_ticks | 10678466000 | 1266139855000 | 11824593000 | 5607559500 |
| sim\_insts | 21356881 | 21356881 | 21356902 | 20972488 |
| system.cpu.numCycles | 21356933 | 2532279710 | 23649186 | 11215120 |
| system.cpu.cpi | 1.000002 | 118.569734 | 1.107332 | 0.534754 |
| system.cpu.committedInsts | 21356881 | 21356881 | 21356902 | 20972488 |
| host\_seconds | 12.17 | 46.91 | 37.68 | 46.69 |
| system.cpu.fetch.Insts | - | - | - | 24132664 |
| Pred. ratio Cond. Branches | - | - | 0.043722 | 0.040075 |
| system.cpu.branchPred.BTBHits | - | - | 1444140 | 1562500 |

1. Compare Table 1 and 2. Why the instructions encountered by the Fetch Unit differ from the instruction committed?

Your Answer: Poichè il tipo di CPU DeriveO3 supporta la branch prediction e il reorder buffer, si continuerà a fare il fetch ed eseguire (pur senza scrivere nei registri e in memoria) tutte le istruzioni dopo il branch. Quando verrà valutato il branch però, se la predizione si rivelasse errata, bisognerebbe fare il flush di tutte quelle istruzioni, che quindi non faranno mai il commit. Il numero di istruzioni fetched e committed sarebbe uguale solo nel caso in cui tutte le branch prediction fossero corrette.

**HINTS:** If you are thinking to use a bash script to automatically run and gather the statistics from the simulations, you might encounter some troubles since the commands listed above are actually aliases of more complex commands (see the start.sh for details). To have the different aliases visible from a bash script, put the following commands at the top of your script:

#!/bin/bash

shopt -s expand\_aliases

source start.sh # or start\_vbox.sh

# Here starts your own script…

**Instructions for importing the VBox VM**

1. Import the virtual machine in VirtualBox (<https://docs.oracle.com/cd/E26217_01/E26796/html/qs-import-vm.html>)
2. The virtual machine can be downloaded using the following link:
   * <https://baltea.polito.it/owncloud/index.php/s/SbJPJb6kQW7mcze>
3. Log in using the following credentials:
   * Account: gem5
   * Password: gem5