|  |  |
| --- | --- |
| **Architetture dei Sistemi di Elaborazione** | Delivery date:  November 5th 2021 |
| **Laboratory**  **4** | Expected delivery of lab\_04.zip must include:   * this document compiled possibly in pdf format. |

1. Introducing gem5

gem5 is freely available at: <http://gem5.org/>

the laboratory version uses the ALPHA CPU model previously compiled and placed at:

|  |
| --- |
| /opt/gem5/ |

the ALPHA compilation chain is available at:

|  |
| --- |
| /opt/alphaev67-unknown-linux-gnu/bin/ |

* 1. Write a hello world C program (hello.c). Then compile the program, using the ALPHA compiler, by running this command:

|  |
| --- |
| ~/my\_gem5Dir$ /opt/alphaev67-unknown-linux-gnu/bin/alphaev67-unknown-linux-gnu-gcc -static -o hello hello.c |

* 1. Simulate the program

|  |
| --- |
| ~/my\_gem5Dir$ /opt/gem5/build/ALPHA/gem5.opt /opt/gem5/configs/example/se.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:

|  |
| --- |
| ~/my\_gem5Dir$ /opt/gem5/build/ALPHA/gem5.opt /opt/gem5/configs/example/se.py -c hello  gem5 Simulator System. http://gem5.org  gem5 is copyrighted software; use the --copyright option for details.  gem5 compiled Sep 20 2017 12:34:54  gem5 started Jan 19 2018 10:57:58  gem5 executing on this\_pc, pid 5477  command line: /opt/gem5/build/ALPHA/gem5.opt /opt/gem5/configs/example/se.py -c hello  Global frequency set at 1000000000000 ticks per second  warn: DRAM device capacity (8192 Mbytes) does not match the address range assigned (512 Mbytes)  0: system.remote\_gdb.listener: listening for remote gdb #0 on port 7000  warn: ClockedObject: More than one power state change request encountered within the same simulation tick  \*\*\*\* REAL SIMULATION \*\*\*\*  info: Entering event queue @ 0. Starting simulation...  info: Increasing stack size by one page.  hola mundo!  Exiting @ tick 2623000 because target called exit() |

* + - 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 extracts 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. ~/my\_gem5Dir$ /opt/gem5/build/ALPHA/gem5.opt /opt/gem5/configs/example/se.py -h |

List the CPU available models:

|  |
| --- |
| ~/my\_gem5Dir$ /opt/gem5/build/ALPHA/gem5.opt /opt/gem5/configs/example/se.py --list-cpu-types |

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

|  |
| --- |
| ~/my\_gem5Dir$ /opt/gem5/build/ALPHA/gem5.opt /opt/gem5/configs/example/se.py --cpu-type=TimingSimpleCPU -c hello |

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

|  |
| --- |
| ~/my\_gem5Dir$ /opt/gem5/build/ALPHA/gem5.opt /opt/gem5/configs/example/se.py --cpu-type=MinorCPU --caches -c hello |

* 1. *DerivO3CPU* is a superscalar processor

|  |
| --- |
| ~/my\_gem5Dir$ /opt/gem5/build/ALPHA/gem5.opt /opt/gem5/configs/example/se.py --cpu-type=DerivO3CPU --caches -c hello |

Create a table gathering for every simulated CPU the following information:

* Ticks
* Number of instructions simulated
* Number of CPU Clock Cycles
  + Number of CPU clock cycles = Number of ticks / CPU Clock period in ticks (usually 500)
* Clock Cycles per Instruction (CPI)
  + CPI = CPU Clock Cycles / instructions simulated
* Number of instructions committed
* Host time in seconds
* Number of instructions Fetch Unit has encountered (this should be gathered for the out-of-order processor only).

TABLE1: Hello program behavior on different CPU models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CPU  Parameters | AtomicSimpleCPU | TimingSimpleCPU | MinorCPU | DeriveO3CPU |
| Ticks |  |  |  |  |
| CPU clock domain |  |  |  |  |
| Clock Cycles |  |  |  |  |
| Instructions simulated |  |  |  |  |
| CPI |  |  |  |  |
| Committed instructions |  |  |  |  |
| Host seconds |  |  |  |  |
| Instructions encountered by Fetch Unit |  |  |  |  |

1. Download the test programs related to the **automotive** sector available in MiBench: basicmath, bitcount, qsort, and susan. These programs are freely available at <http://vhosts.eecs.umich.edu/mibench/>
2. compile the program basicmath using the provided *Makefile* using the ALPHA compiler

*hint:*

|  |
| --- |
| add a variable to the Makefile in order to use the ALPHA compiler:  CROSS\_COMPILE = /opt/alphaev67-unknown-linux-gnu/bin/alphaev67-unknown-linux-gnu  CC=$(CROSS\_COMPILE)-gcc  and substitute all the gcc occurrences with the new variable as follows:  *gcc* → $(CC) |

1. Simulate the program **basicmath** using the ***large***set of inputs and the default processor (*AtomicSimpleCPU)*, saving the output results. In the case the simulation time is higher than a couple of minutes, modify the program in order to reduce the simulation time; for example, in the case of basicmath, it is necessary to reduce the number of iterations the program executes in order to reduce the computational time.

TODO: To reduce the simulation time of *basicmath\_large.c*, modify the number of iterations of the for loops as follows:

Text

Description automatically generated

1. Simulate the resulting program using the gem5 different CPU models and collect the following information:
   1. Number of instructions simulated
   2. Number of CPU Clock Cycles
   3. Clock Cycles per Instruction (CPI)
   4. Number of instructions committed
   5. Host time in seconds
   6. Prediction ratio for Conditional Branches (Number of Incorrect Predicted Conditional Branches / Number of Predicted Conditional Branches)
   7. BTB hits
   8. Number of instructions Fetch Unit has encountered.

Parameters *f* , *g and h* should be gathered exclusively for the out-of-order processor.

TABLE2: basicmath\_large program behavior on different CPU models

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CPUs  Parameters | AtomicSimpleCPU | TimingSimpleCPU | MinorCPU | DerivO3CPU |
| Ticks |  |  |  |  |
| CPU clock domain |  |  |  |  |
| Clock Cycles |  |  |  |  |
| Instructions simulated |  |  |  |  |
| CPI |  |  |  |  |
| Committed instructions |  |  |  |  |
| Host seconds |  |  |  |  |
| Prediction ratio |  |  |  |  |
| BTB hits |  |  |  |  |
| Instructions encountered by Fetch Unit |  |  |  |  |