

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/
```

- a. 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
```

- b. 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.

- c. 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

```

## 2) Simulate the same program using different CPU models.

### Help command:

```
~/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
```

#### a. *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
```

#### b. *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
```

#### c. *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)

- $CPI = \text{CPU Clock Cycles} / \text{instructions simulated}$
- 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	2653000	383438000	33502500	18813000
CPU clock domain	500	500	500	500
Clock Cycles	5306	766876	67005	37626
Instructions simulated	5277	5277	5289	5077
CPI	$5306/5277=1.006$	145.32	12.67	7.41
Committed instructions	5277	5277	5289	5077
Host seconds	0.01	0.03	0.04	0.03
Instructions encountered by Fetch Unit				10780

3) 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/>

- a) Modify the program `basicmath_small`, reducing the number of iterations and therefore the computational time (line 44):

```
/* Now solve some random equations */
for(a1=1;a1<5;a1++) {
    for(b1=5;b1>0;b1--) {
        for(c1=5;c1<8;c1+=0.5) {
            for(d1=-1;d1>-3;d1--) {
                SolveCubic(a1, b1, c1, d1, &solutions, x);
                printf("Solutions:");
                for(i=0;i<solutions;i++)
                    printf(" %f",x[i]);
                printf("\n");
            }
        }
    }
}
```

- b) compile the program `basicmath_small` 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:

$gcc \rightarrow \$(CC)$

- c) Simulate the program `basicmath_small` and the default processor (*AtomicSimpleCPU*), saving the output results.
- d) Simulate the program using the gem5 different CPU models and collect the following information:
  - a) Number of instructions simulated
  - b) Number of CPU Clock Cycles
  - c) Clock Cycles per Instruction (CPI)
  - d) Number of instructions committed
  - e) Host time in seconds
  - f) Prediction ratio for Conditional Branches (Number of Incorrect Predicted Conditional Branches / Number of Predicted Conditional Branches)
  - g) BTB hits
  - h) Number of instructions Fetch Unit has encountered.Parameters  $f$ ,  $g$  and  $h$  should be gathered exclusively for the out-of-order processor.

TABLE2: `basicmath_small` program behavior on different CPU models

Parameters	CPUs	AtomicSimpleCPU	TimingSimpleCPU	MinorCPU	DerivO3CPU
Ticks		5170330500	712106412000	9858595500	4266611500
CPU clock domain		500	500	500	500
Clock Cycles		10340662	1424212824	19717191	8533228
Instructions simulated		10340600	10340600	10340626	10139315
CPI		1	137.73	1.90	
Committed instructions		10340600	10340600	10340626	10340599
Host seconds		4.80	57.21	35.97	38.54
Prediction ratio				0.045	0.038
BTB hits				1003814	1068696
Instructions encountered by Fetch Unit					