

**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/
```

- 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-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

Parameters	AtomicSimpleCPU	TimingSimpleCPU	MinorCPU	DeriveO3CPU
Ticks	2654500	342272000	30117500	17146000
CPU clock domain	500	500	500	500
Clock Cycles	5310	684544	60235	34293
Instructions simulated	5280	5280	5292	5080
CPI	1.0057	129.64	11.38	6.75
Committed instructions	5280	5280	5292	5080
Host seconds	0.00	0.02	0.03	0.05
Instructions encountered by Fetch Unit				10950

- 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) 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)
```

- b) 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:

```
76  /* Now solve some random equations */  
77  for(a1=1;a1<10;a1+=2) { // EDITED  
78      for(b1=10;b1>0;b1-=1) { // EDITED  
79          for(c1=5;c1<15;c1+=1) { // EDITED  
80              for(d1=-1;d1>-.5;d1+=.5) { // EDITED  
81                  SolveCubic(a1, b1, c1, d1, &solutions, x);  
82                  printf("Solutions:");  
83                  for(i=0;i<solutions;i++)  
84                      printf(" %f",x[i]);  
85                  printf("\n");  
86              }  
87          }  
88      }  
89  }  
90  
91  
92  printf("***** INTEGER SQRT ROOTS *****\n");  
93  /* perform some integer square roots */  
94  for (i = 0; i < 1000; i+=2) // EDITED  
95  {  
96      usqrt(i, &q);  
97      // remainder differs on some machines  
98      // printf("sqrt(%3d) = %2d, remainder = %2d\n",  
99      printf("sqrt(%3d) = %2d\n",  
100         i, q.sqrt);  
101  }  
102  printf("\n");  
103  for (l = 0x3fed0169L; l < 0x3fed4169L; l++)  
104  {  
105      usqrt(l, &q);  
106      //printf("\nsqrt(%lx) = %X, remainder = %X\n", l, q.sqrt, q.frac);  
107      printf("sqrt(%lx) = %X\n", l, q.sqrt);  
108  }  
109  
110  
111  printf("***** ANGLE CONVERSION *****\n");  
112  /* convert some rads to degrees */  
113  /* for (X = 0.0; X <= 360.0; X += 1.0) */  
114  for (X = 0.0; X <= 360.0; X += .01) //EDITED  
115      printf("%.30f degrees = %.12f radians\n", X, deg2rad(X));  
116  puts("");  
117  /* for (X = 0.0; X <= (2 * PI + 1e-6); X += (PI / 180)) */  
118  for (X = 0.0; X <= (2 * PI + 1e-6); X += (PI / 5760))  
119      printf("%.12f radians = %.30f degrees\n", X, rad2deg(X));  
120  
121  
122  return 0;  
123 }
```

- c) Simulate the resulting program using the `gem5` different CPU models and collect the following information:
- Number of instructions simulated
 - Number of CPU Clock Cycles
 - Clock Cycles per Instruction (CPI)
 - 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_large` program behavior on different CPU models

Parameters	AtomicSimpleCPU	TimingSimpleCPU	MinorCPU	DerivO3CPU
Ticks	222416570500	27849853285000	358284775500	142889111500
CPU clock domain	500	500	500	500
Clock Cycles	444833142	5569970650	716569551	285778225
Instructions simulated	444833079	444833079	444833105	436251129
CPI	1.0000	12.5200	1.6108	0.655077
Committed instructions	444833079	444833079	444833105	436251129
Host seconds	225.15	121699	1142.47	1292.11
Prediction ratio				0.0267
BTB hits				46229517
Instructions encountered by Fetch Unit				485513496