Architetture dei Sistemi di Elaborazione **[GRB-ZZZ]**	Delivery date: November 8th 2022
Laboratory **4**	Expected delivery of lab_04.zip must include: - this document compiled possibly in pdf format.

1) Introducing gem5d

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:

```
/opt/gem5/~/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 = 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	2754000	398041000	34304500	20223500
CPU clock domain	500	500	500	500
Clock Cycles	5508	796082	68609	40447
Instructions simulated	5475	5475	5488	5276
CPI	1.006	145.4	12.5	7.666
Committed instructions	5475	5475	5488	5474
Host seconds	0.01	0.02	0.02	0.03
Instructions encountered by Fetch Unit	x	x	x	11250

3) Download the test programs related to the **automotive** sector available in MiBench: `basicmath`, `bitcount`, `qsort`, and `susan`. These programs are freely available at https://github.com/embecosm/mibench

Commentato [AF1]: Old link does not work

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
```

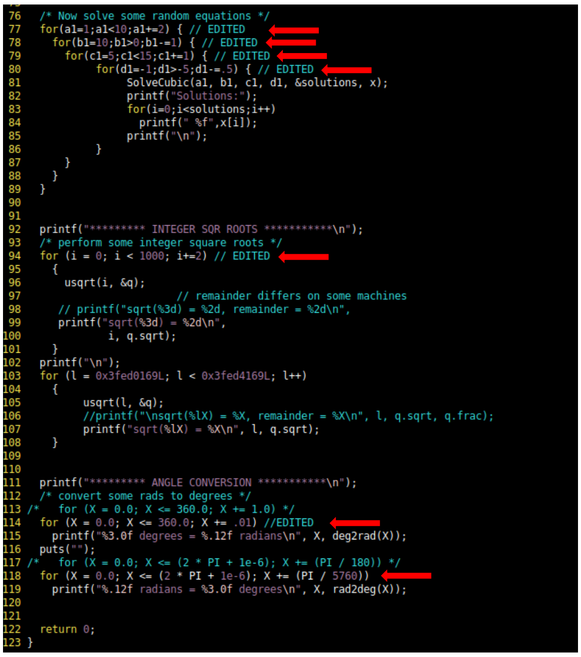
and substitute all the `gcc` occurrences with the new variable as follows:

gcc → $(CC)

then compile:

```
~/my_mybench_dir/automotive/basicmath/ make
```

b) Simulate the program **`basicmath`** using the ***large*** set of inputs (i.e., compile basicmath_large.c) and the default processor (*AtomicSimpleCPU)*, saving the output results. In the case the simulation time is higher than a couple of minutes (it is host-dependent!), 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 (in case of long simulation time): To reduce the simulation time of *basicmath_large.c*, modify the number of iterations of the for loops as follows (RED arrow):



```
  /* Now solve some random equations */
  for(a1=1;a1<10;a1+=2) { // EDITED
    for(b1=10;b1>0;b1-=1) { // EDITED
      for(c1=5;c1<15;c1+=1) { // EDITED
           for(d1=-1;d1>-5;d1-=.5) { // EDITED
                SolveCubic(a1, b1, c1, d1, &solutions, x);
                printf("Solutions:");
                for(i=0;i<solutions;i++)
                  printf(" %f",x[i]);
                printf("\n");
           }
      }
    }
  }


  printf("********* INTEGER SQR ROOTS ***********\n");
  /* perform some integer square roots */
  for (i = 0; i < 1000; i+=2) // EDITED
    {
      usqrt(i, &q);
                      // remainder differs on some machines
     // printf("sqrt(%3d) = %2d, remainder = %2d\n",
     printf("sqrt(%3d) = %2d\n",
             i, q.sqrt);
    }
  printf("\n");
  for (l = 0x3fed0169L; l < 0x3fed4169L; l++)
    {
         usqrt(l, &q);
         //printf("\nsqrt(%lX) = %X, remainder = %X\n", l, q.sqrt, q.frac);
         printf("sqrt(%lX) = %X\n", l, q.sqrt);
    }


  printf("********* ANGLE CONVERSION ***********\n");
  /* convert some rads to degrees */
/*   for (X = 0.0; X <= 360.0; X += 1.0) */
  for (X = 0.0; X <= 360.0; X += .01) //EDITED
    printf("%3.0f degrees = %.12f radians\n", X, deg2rad(X));
  puts("");
/*   for (X = 0.0; X <= (2 * PI + 1e-6); X += (PI / 180)) */
  for (X = 0.0; X <= (2 * PI + 1e-6); X += (PI / 5760))
    printf("%.12f radians = %3.0f degrees\n", X, rad2deg(X));


  return 0;
}
```

c) Simulate the resulting 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_large` program behavior on different CPU models

CPUs				
Parameters	AtomicSimpleCPU	TimingSimpleCPU	MinorCPU	DerivO3CPU
Ticks	222416559500	31158520203000	364964286500	144932423500
CPU clock domain	500	500	500	500
Clock Cycles	444833119	62317040406	729928573	289864849
Instructions simulated	444833057	444833057	444833083	436251113
CPI	1.000	140	1.640904	0.664445
Committed instructions	444833057	444833057	444833083	436251113
Host seconds	497.81	2819.49	1387.63	1323.42
Prediction ratio	x	x	96.8%	97.2%
BTB hits	x	x	43954068	46229129
Instructions encountered by Fetch Unit	x	x		485507542