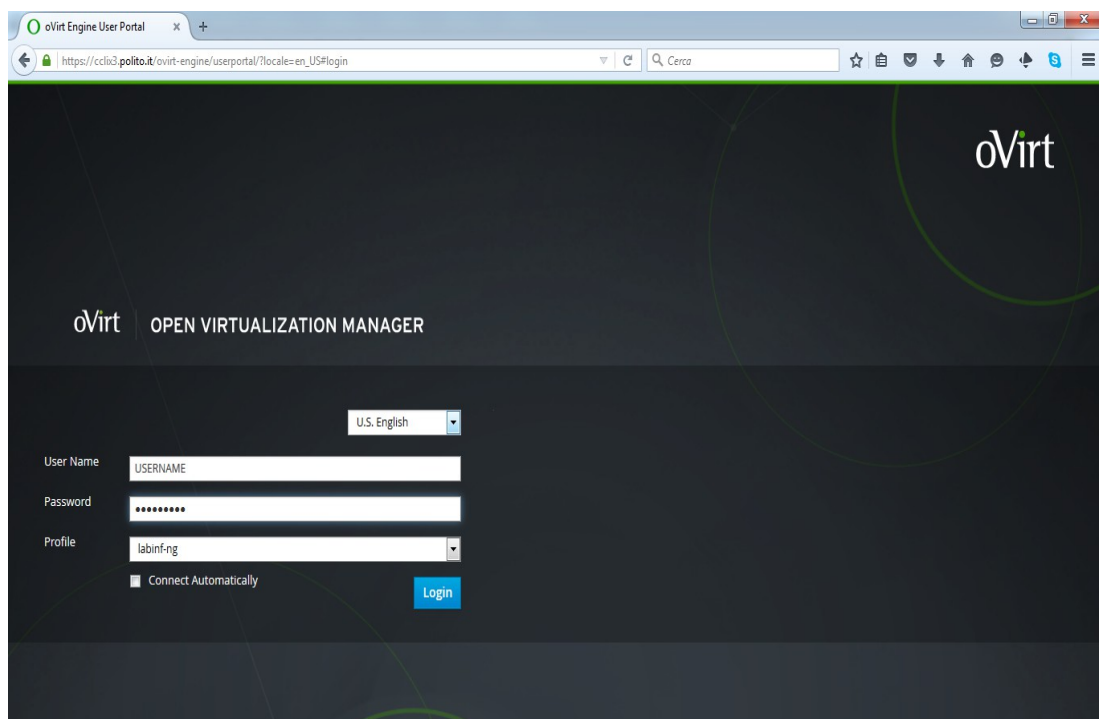


Computer Architectures

2nd part labs – lab 2

Introducing SimpleScalar

- 1) Working with SimpleScalar using the virtualization system:
 - a. Open this site: cclix3.polito.it in your favorite browser
 - b. Click on **UserPortal** and accept the web's security certificate
 - c. log you in using your labinf credentials
 - use the profile: labinf-ng
 - disable the option: Connect Automatically



- d. start the CASLABS3 virtual machine by clicking on “RUN VM”
 - e. then, in the console menu, select Connect.

2) Working with the SimpleScalar virtual machine:

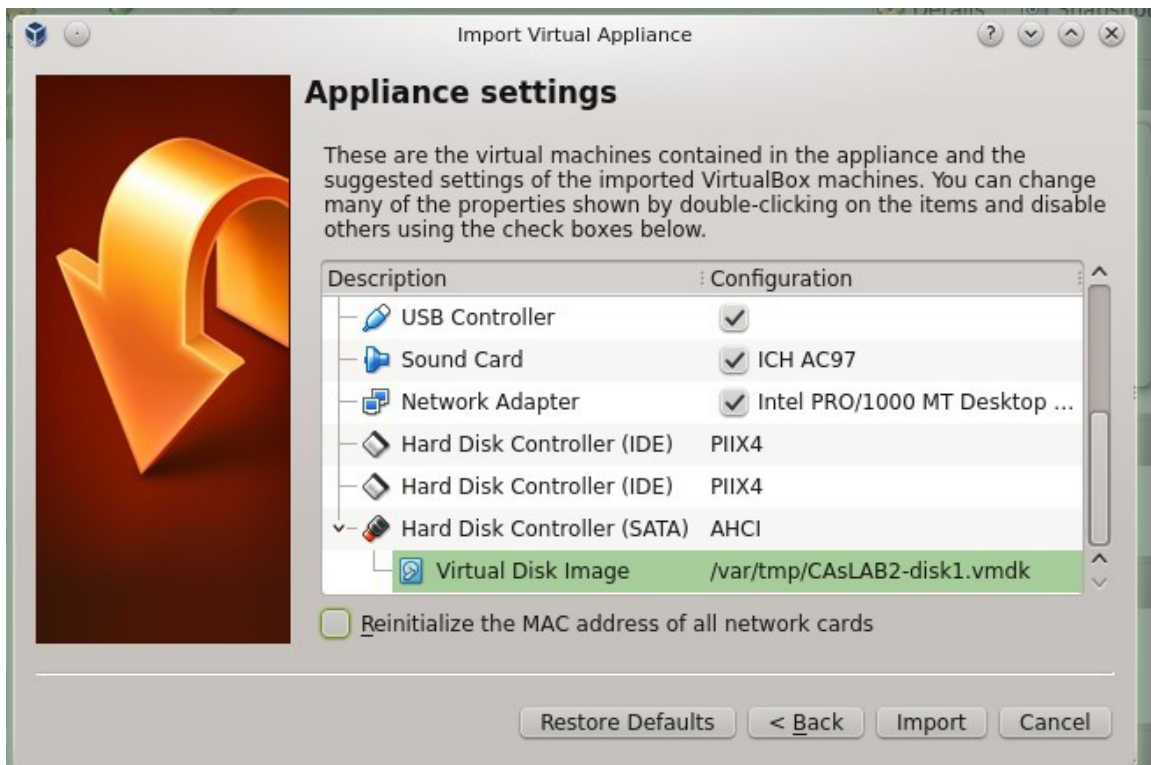
- a. the virtual machine (*CAsLABS2.ova*) for SimpleScalar labs is placed in the folder:

`/var/tmp`

- b. launch the virtual box tool

Oracle VM VirtualBox

- c. import the SimpleScalar virtual machine by clicking on *import appliance* in the *file* menu, and select the virtual machine in the tmp folder
- d. in the appliance settings window, change the path of the *Virtual Disk Image* by the tmp folder used before (`/var/tmp`):



3) start the virtual machine

- username: Student CAS
- password: cas

4) launch a terminal

- notice that the main folder for SimpleScalar is:

```
~/simplescalar/simplesim-3.0/
```

5) Using SimpleScalar:

a. modify the hello.c program placed in:

6)

```
~/simplescalar/
```

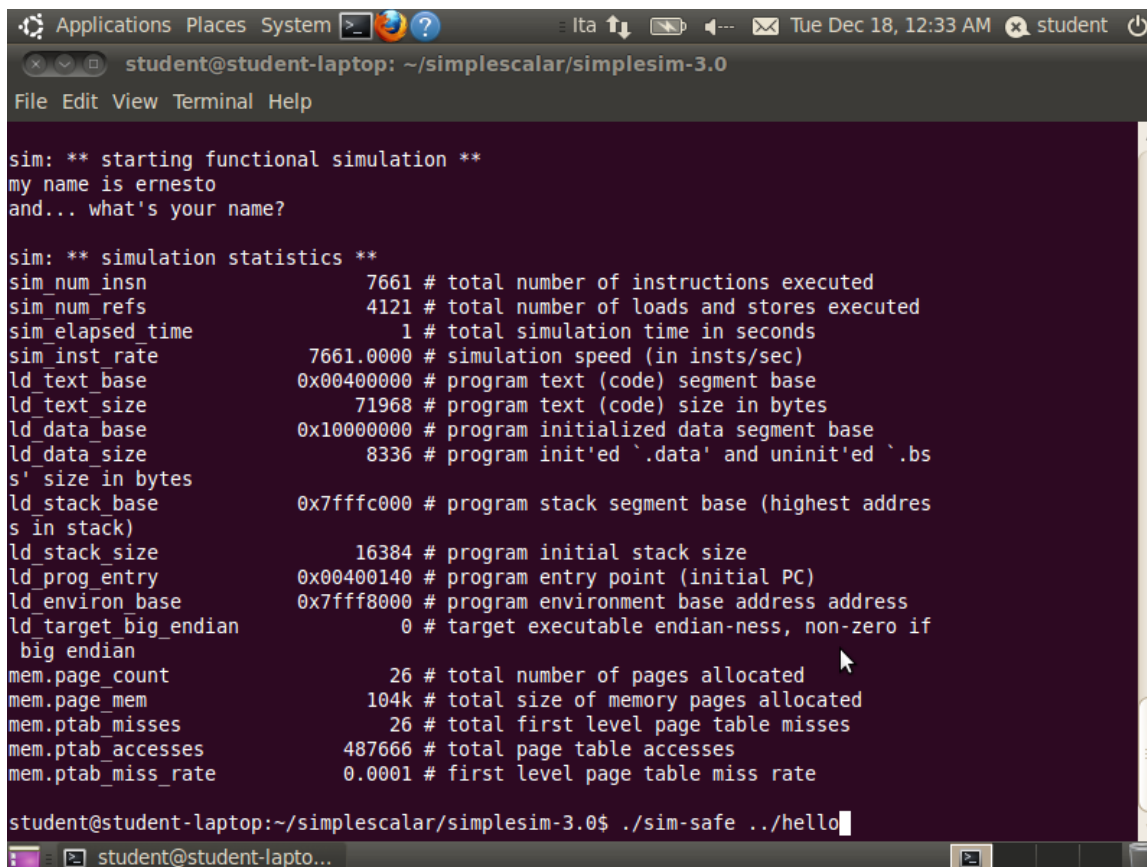
b. compile the program, using gcc, by running this command:

```
student@student-laptop:~/simplescalar$ ./bin/sslittle-na-sstrix-gcc -o  
hello hello.c
```

c. execute the program using sim-safe:

```
student@student-laptop:~/simplescalar/simplesim-3.0$ ./sim-safe ../hello
```

d. check the output with the following:



```
sim: ** starting functional simulation **  
my name is ernesto  
and... what's your name?  
  
sim: ** simulation statistics **  
sim_num_insn          7661 # total number of instructions executed  
sim_num_refs          4121 # total number of loads and stores executed  
sim_elapsed_time       1 # total simulation time in seconds  
sim_inst_rate         7661.0000 # simulation speed (in insts/sec)  
ld_text_base          0x00400000 # program text (code) segment base  
ld_text_size          71968 # program text (code) size in bytes  
ld_data_base          0x10000000 # program initialized data segment base  
ld_data_size          8336 # program init'ed '.data' and uninit'ed '.bs  
s' size in bytes  
ld_stack_base         0x7fffc000 # program stack segment base (highest address  
s in stack)  
ld_stack_size         16384 # program initial stack size  
ld_prog_entry         0x00400140 # program entry point (initial PC)  
ld_environ_base       0x7fff8000 # program environment base address address  
ld_target_big_endian  0 # target executable endian-ness, non-zero if  
big endian  
mem.page_count        26 # total number of pages allocated  
mem.page_mem          104k # total size of memory pages allocated  
mem.ptab_misses       26 # total first level page table misses  
mem.ptab_accesses     487666 # total page table accesses  
mem.ptab_miss_rate    0.0001 # first level page table miss rate  
  
student@student-laptop:~/simplescalar/simplesim-3.0$ ./sim-safe ../hello
```

- 7) Execute the program using the superscalar version (out of order) of the simulator (`sim-outorder`):

```
student@student-laptop:~/simplescalar/simplesim-3.0$ ./sim-outorder ../hello
```

a. compare the output results using different simulators. Notice that the out of order simulator provides more information about processor performance.

8) Running tests-pisa programs

5 testing C programs (`test-*.c`) for the *pisa* version of simplescalar are placed in:

```
~/simplescalar/simplesim-3.0/tests-pisa/src
```

- a. copy the source programs (`test-*.c`) in the folder *CAsLABs*
- b. compile all the C programs (see 2.b)
- c. run the test programs with the “superscalar out of order” version of the simulator using the the configuration file called *default.cfg* available in *CAsLABs*
example:

```
student@student-laptop:~/simplescalar/CAsLABs$ ../simplesim-3.0/sim-outorder  
-config default.cfg test-math
```

- d. check the program output, and collect for every program the following statistics parameters:
 - `sim_num_insn` // total number of instructions committed
 - `sim_cycle` // total simulation time in cycles
 - `sim_IPC` // instructions per cycle
 - `sim_CPI` // cycles per instruction

Check the configuration in the *default.cfg* file, and notice that the provided configuration tries to use a very simplified processor architecture. The main parameters to consider are listed in the following:

# instruction fetch queue size (in insts)	
-fetch:ifqsize	1
# speed of front-end of machine relative to execution core	
-fetch:speed	1
# extra branch mis-prediction latency	
-fetch:mplat	1
# instruction issue B/W (insts/cycle)	
-issue:width	1
# instruction decode B/W (insts/cycle)	
-decode:width	1
# instruction commit B/W (insts/cycle)	
-commit:width	1

# run pipeline with in-order issue	
-issue:inorder	true
# register update unit (RUU) size (this unit is similar to ROB)	
-ruu:size	2
# load/store queue (LSQ) size	
-lsq:size	2
# total number of integer ALU's available	
-res:ialu	1
# total number of integer multiplier/dividers available	
-res:imult	1
# total number of memory system ports available (to CPU)	
-res:memport	1
# total number of floating point ALU's available	
-res:fpalu	1
# total number of floating point multiplier/dividers available	
-res:fpmult	1

9) Performance comparison:

Modify the configuration file `default.cfg`, in order to improve the processor performance while running the `test-pisa` programs.

In particular, for every run change the parameters value following the directions provided in the table below:

conf	FP resources	-fetch: speed	-fetch: ifqsize	-lsq: size	-ruu: size	-issue: width	-commit: width	In order	sim_CPI
1	1	1	1	2	2	1	1	Y	
2	1	1	1	2	4	1	1	N	
3	4	1	1	2	4	1	1	N	
4	4	1	1	2	8	1	1	N	
5	1	4	4	4	8	4	4	N	
6	4	4	4	4	16	4	4	N	
7									

Simulate every *test-pisa* program gathering the `sim_CPI` value for every one of the proposed configurations.

Comment the results justifying the change in the performance.

10) Define your own configuration, (place it on table's row No 7) in order to perform better than in the proposed cases.