CS203 Winter ’16 - Project 1

Due: Thursday, January 21 before class
All projects will be turned in to iLearn.

1. Objective

The goal of project 1 is to setup the simulation environment. We will be using the ESESC simulator running in a VirtualBox virtual machine. Throughout the course of the quarter, we will be using this simulator to gain hands on design experience and to reinforce concepts covered in lecture. If you experience any difficulties during setup, please post to Piazza as others may be having the same issue.

2. Installing VM

Download and Install VirtualBox

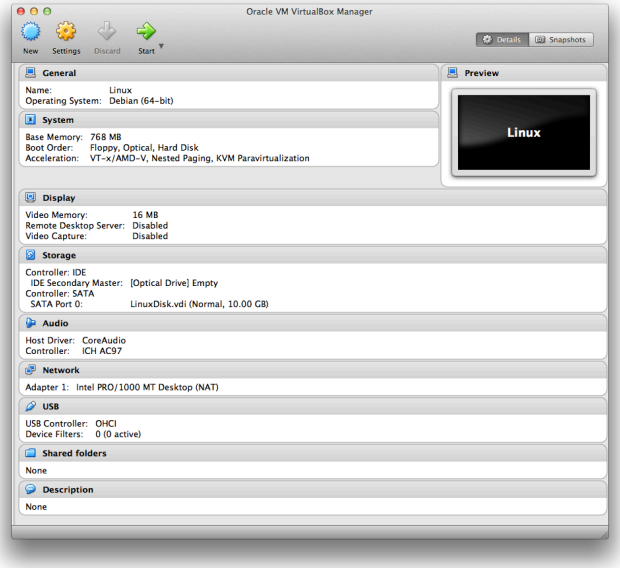
Go to https://www.virtualbox.org/wiki/Downloads and download the appropriate VirtualBos for your host system. The download linkes are under the “VirtualBox platform packages” section.

- **VirtualBox platform packages.** The binaries are released under the terms of the GPL version 2.
  - **VirtualBox 5.0.12 for Windows hosts** x86/amd64
  - **VirtualBox 5.0.12 for OS X hosts** amd64
  - **VirtualBox 5.0.12 for Linux hosts**
  - **VirtualBox 5.0.12 for Solaris hosts** amd64

Import Linux virtual machine

The VM image used is based on Linux Mint 17.3 (built on Ubuntu 14.04 LTS).

Go to http://www.cs.ucr.edu/~danwong/Linux.ova
or http://www.ee.ucr.edu/~dwong/Linux.ova
to download the virtual machine image.
WARNING: File size is ~3GB. Please plan accordingly when you download this.



Open Virtual Box. To import, go to File -> Import Appliance

Browse for downloaded Linux.ova file and click Continue. Then Click continue again.

Once imported, you should see the VM in VirtualBox. If you want, at this point, you can also click Settings to change the resources dedicated for your VM. Allocating more processor cores and memory will enable the simulator to run faster.

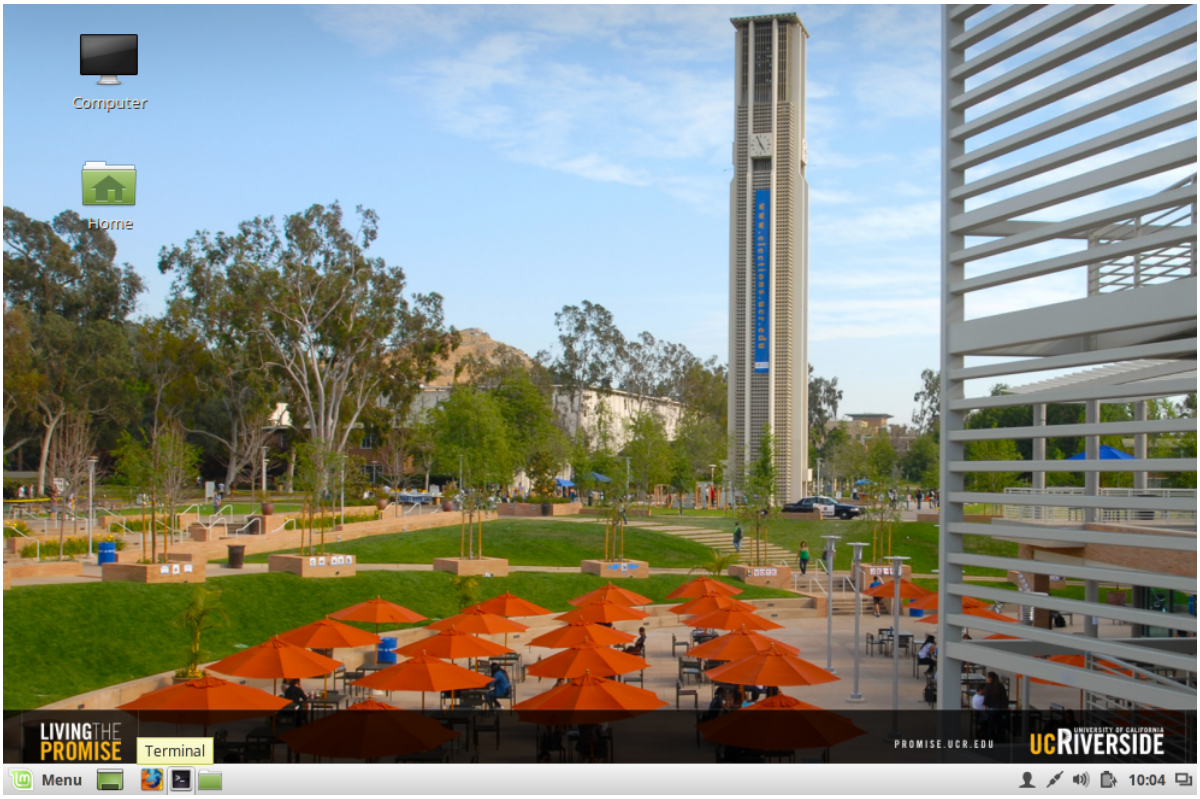
Boot VM

Boot the VM by pressing Start. And login with the following account:

Username: student
Password: UCRiverside

Open terminal

Click on the terminal icon on the lower left of the screen. (A black square).



3. ESESC Setup

Go to esesc folder:

```
cd ~/esesc
```

Make directory to build simulator:

```
mkdir -p build/release

cd build/release
```

Compile simulator:

```
cmake ~/esesc

make
```

Copy config files:

```
cp ~/esesc/conf/* .
```

(no need for recursive copy. It's ok to omit the directories in config/)

Copy binaries to simulate:

```
cp ~/esesc/bins/mips64/* .

cp ~/esesc/bins/inputs/* .
```

4. Running ESESC

Edit configuration to point to binary:

```
vim esesc.conf
```

(use any text editor you want)

Uncommment `benchName = ...` to pick the benchmark binary you want to run.
Make sure only one benchName is uncommented.
NOTE: for `kernel_dhry.mips64` change "`1000`" to "`10000000`". Otherwise, the benchmark is too short for the simulator to extract useful statistics.

Run simulator

```
./main/esesc
```

NOTE: If you're running the `crafty` benchmark, you have to pass the crafty input like so:
`./main/esesc < crafty.in`

Process report file

After running a simulation, a result file is generated (esesc_result.*). Run the following script to process the output into a user-friendly table format.

```
~/esesc/conf/scripts/report.pl -last
```

You will see an output like below:

```
*****************************************************************************************************
# File  : esesc_noname.M4fWnD  : Mon Jan 11 02:19:05 2016
*****************************************************************************************************
Sampler 0 (Procs  0 1)
         Rabbit     Warmup    Detail    Timing    Total  KIPS
  KIPS   44359       1760       757       765      3794
  Time     5.0%      83.3%      1.6%     10.1%             : Sim Time (s) 131.789 Exe   4.739 ms Sim (1700MHz)
  Inst    59.0%      38.7%      0.3%      2.0%           : Approx Total Time 288.533 ms Sim (1700MHz)
*****************************************************************************************************
Proc : Avg.Time :  BPType           :  Total  :          RAS         :  BPred  :         BTB          :  BTAC   : WasteRatio
   0 :   24.993 :  2bit             :  61.11% : (100.00% of 9.58%) :  68.11% : ( 54.37% of 29.30%) :  33.74% :  38.41%
-----------------------------------------------------------------------------------------------------------------------------
--------
Proc :  nCommit   :   nInst   :  AALU   :  BALU   :  CALU   :  LALU   :  SALU   :  LD Fwd :    Replay    : Worst Unit  (clk)
   0 :  10139922  :  10139922 :  48.38% :  18.22% :   0.05% :  25.76% :   7.60% :  0.00%  :      N/A     : MUNIT_AALU  0.22
-----------------------------------------------------------------------------------------------------------------------------
--------
Proc  IPC     uIPC    Active        Cycles    Busy   LDQ   STQ  IWin   ROB  Regs    IO  maxBr  MisBr Br4Clk brDelay
   0   1.26   1.26   1.00         8056756     62.9   0.2   1.8   1.8   0.1   0.7   0.0   0.0   0.0    0.0   13.1
*****************************************************************************************************
Cache           Occ  AvgMemLat MemAccesses  MissRate  (  RD ,    WR,    BUS)
IL1(0)          0.0  2.2       3608033       0.18%    ( 99.8%,  0.0%,  0.0%)
-----------------------------------------------------------------------------------------------------
DL1(0)          0.0  4.9       3347212       0.23%    ( 99.8%, 97.2%,  0.0%)
-----------------------------------------------------------------------------------------------------
ITLB(0)         0.0  2.2       3607979       0.00%    (100.0%,  0.0%,  0.0%)
L2(0)           0.0  43.7      14186        24.61%    ( 75.4%,  0.0%,  0.0%)
Memory(0)       0.0  0.0       0             0.00%    (100.0%,  0.0%,  0.0%)
PTLB(0)         0.0  4.9       3339493       0.16%    ( 99.8%,  0.0%,  0.0%)
*****************************************************************************************************
```

Important statistics relevant to this project includes:

```
Sim Time (s) 131.789 Exe   4.739 ms Sim (1700MHz)
```

This shows how long the simulation took to run (Exe) and how long the processor was simulated for (Sim). In this case, the simulation took 131s to run, and the processor simulated 4.7ms. The `Sim` time is of importance as it represents the program's execution time on the simulated processor.

`nInst` – Number of instructions

`IPC` – Instruction per clock

`Cycles` – Number of clock cycles

5. Assignment Problems

1. Run each of the benchmarks using the default configuration file and fill out the following table. What is the arithmetic mean of the execution time?

Benchmark	Exec Time (sim)	nInst	Cycles	IPC
blackscholes				
crafty				
kernel_dhry				
sjeng				

2. Edit esesc.conf. Edit the following line: `frequency = 1700e6` to `frequency = 3400e6`. Here we are doubling the frequency of the processor from 1.7GHz to 3.4GHz. Fill out the following table using this modified configuration. What is the arithmetic mean of the execution time?

Benchmark	Exec Time (sim)	nInst	Cycles	IPC
blackscholes				
crafty				
kernel_dhry				
sjeng				

3. Does the IPC improve from doubling the processor frequency? Why or why not? How can you further improve the IPC of a processor?

4. Using the configuration from question 1 as the reference machine, what is the speedup for each benchmark? What is the geometric mean of the speedup?

5. Assuming a program has 30% of instructions that uses the ALU, and we developed a technique to speedup ALU instructions, what is the maximum speedup that we can achieve?

6. If 75% of a program can be done in parallel, what is the speedup on a machine with 4 cores vs a machine with 1 core?