實驗環境Platform架設與實作

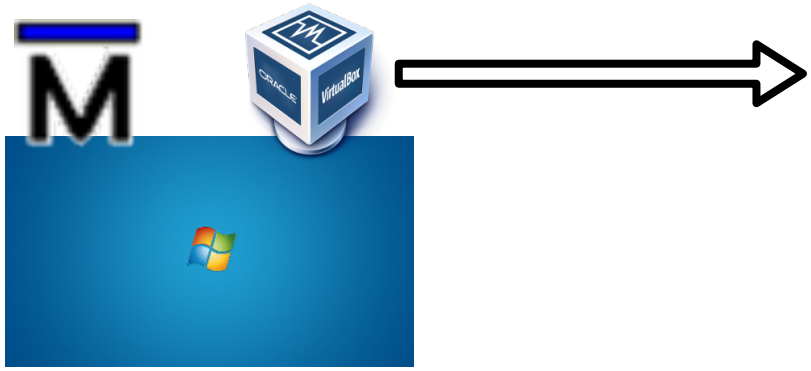
OUTLINE

1. Overview
2. Virtualbox Install
3. Ubuntu
4. RISC-V Cross Compiler
5. Modelsim
6. Example

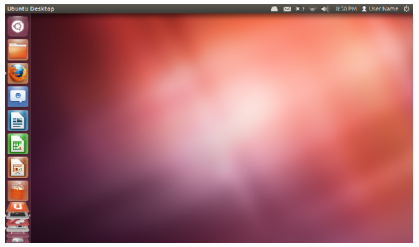
實驗環境Overview

- Operating System : Windows, Ubuntu (on Virtualbox)
- Software: Modelsim, riscv-gnu-toolchain(cross compiler), Python3
- Code: RISC-V RTL code

2. Windows is installed with Modelsim and Virtualbox(Virtual machine)



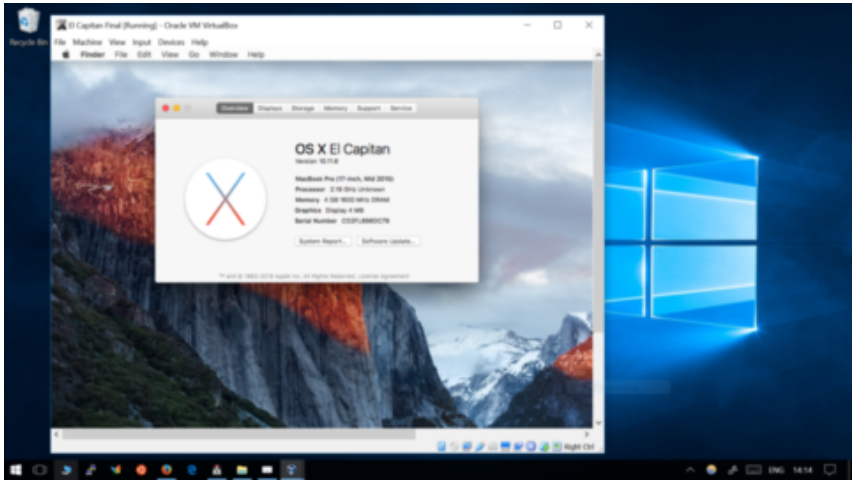
1. Enter windows



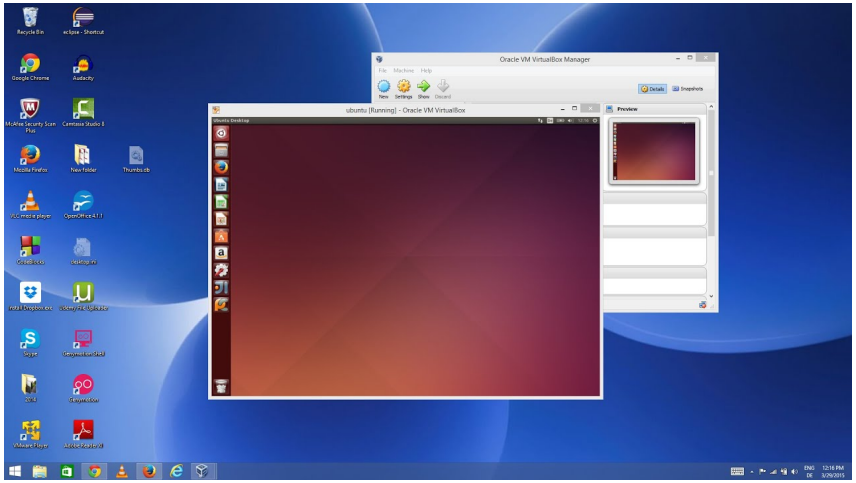
Ubuntu 18.04 Use Virtualbox to execute system.

Virtualbox

- Mac OS X on Windows 10



- Ubuntu on Windows 10

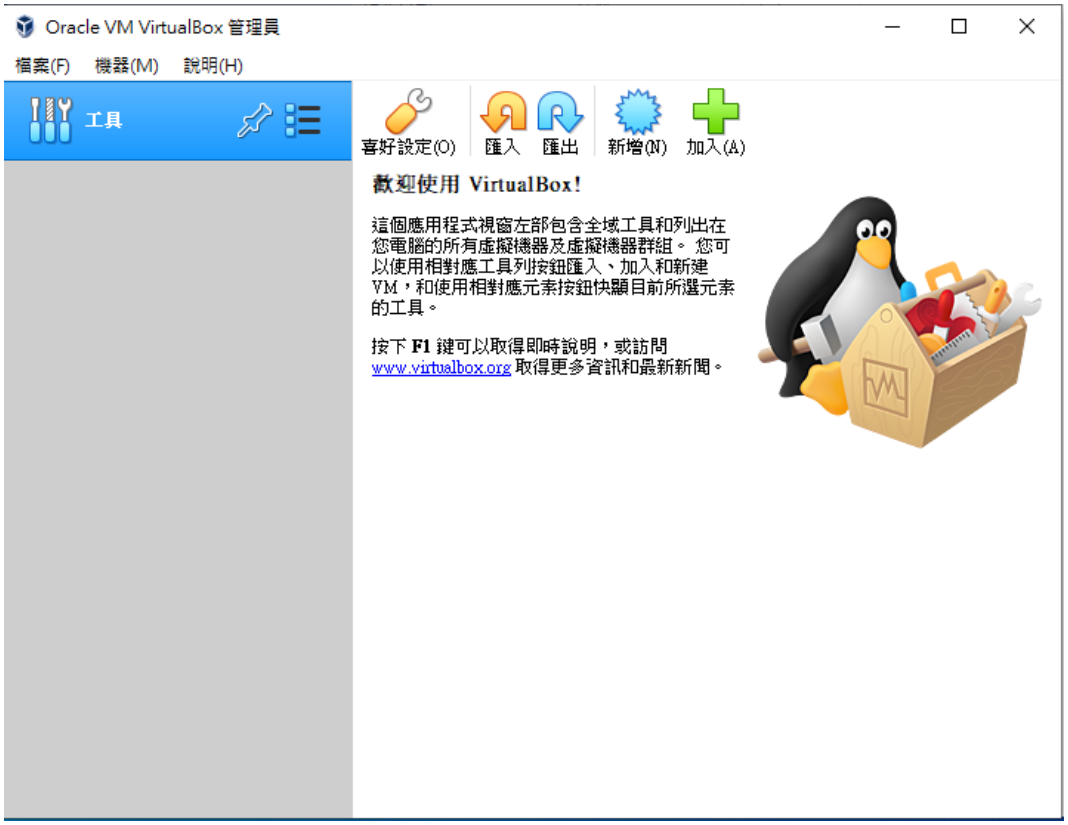


Virtualbox Install

實驗室電腦已安裝完成，可跳過

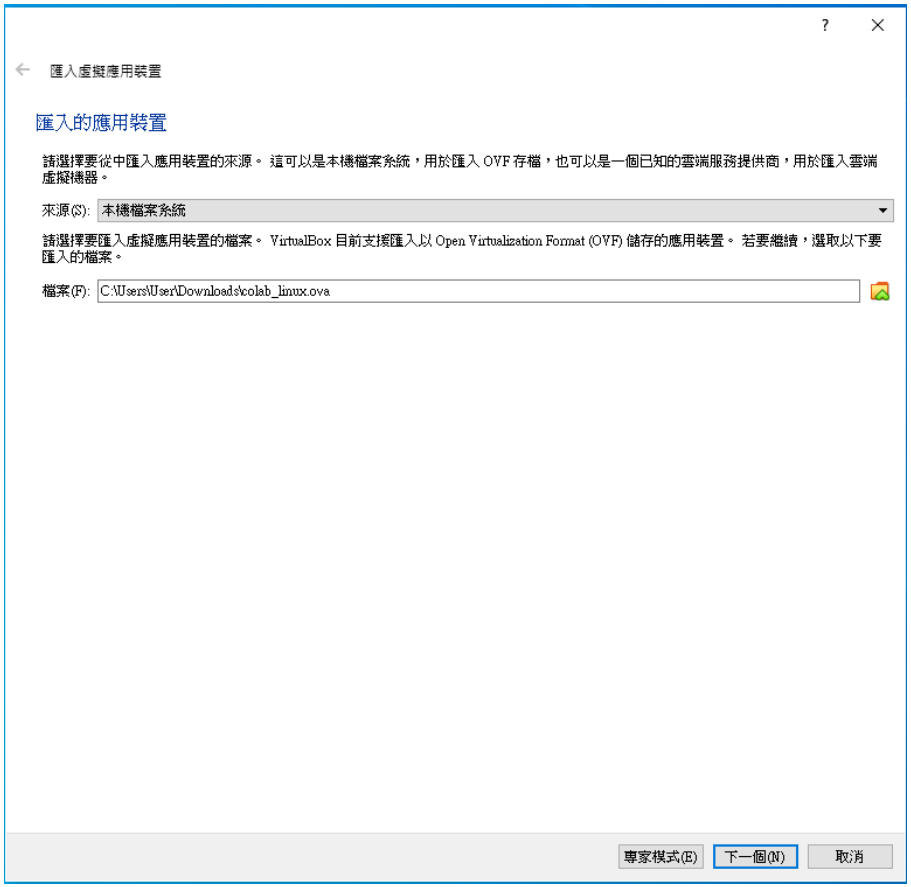
Virtualbox Install

- Download : https://www.virtualbox.org/
- 點選”匯入”



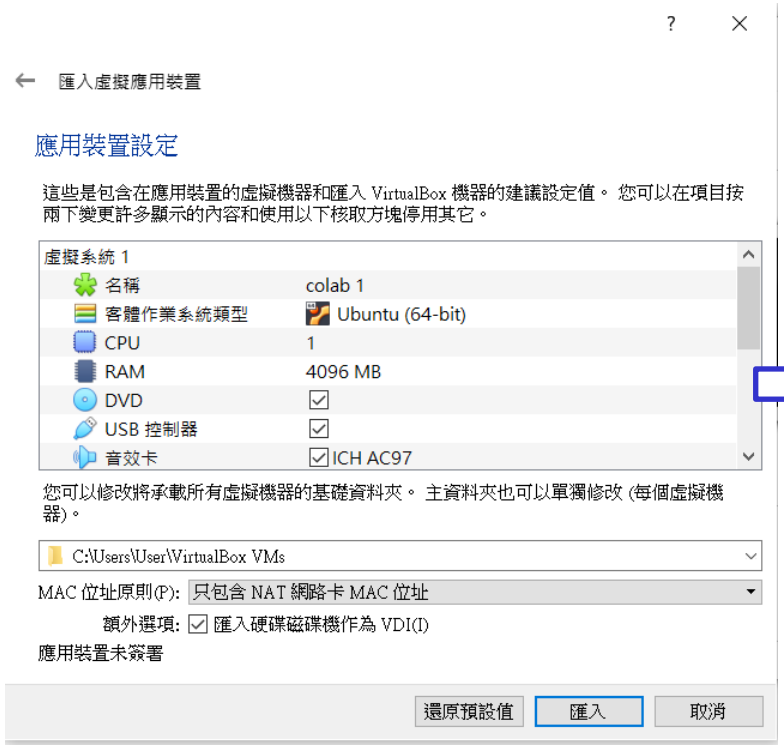
Virtualbox Install

- 檔案選擇colab_linux.ova
  - 下載
  - colab_linux.ova：已經安裝riscv-toolchain的ubuntu 18.04作業系統



Virtualbox Install

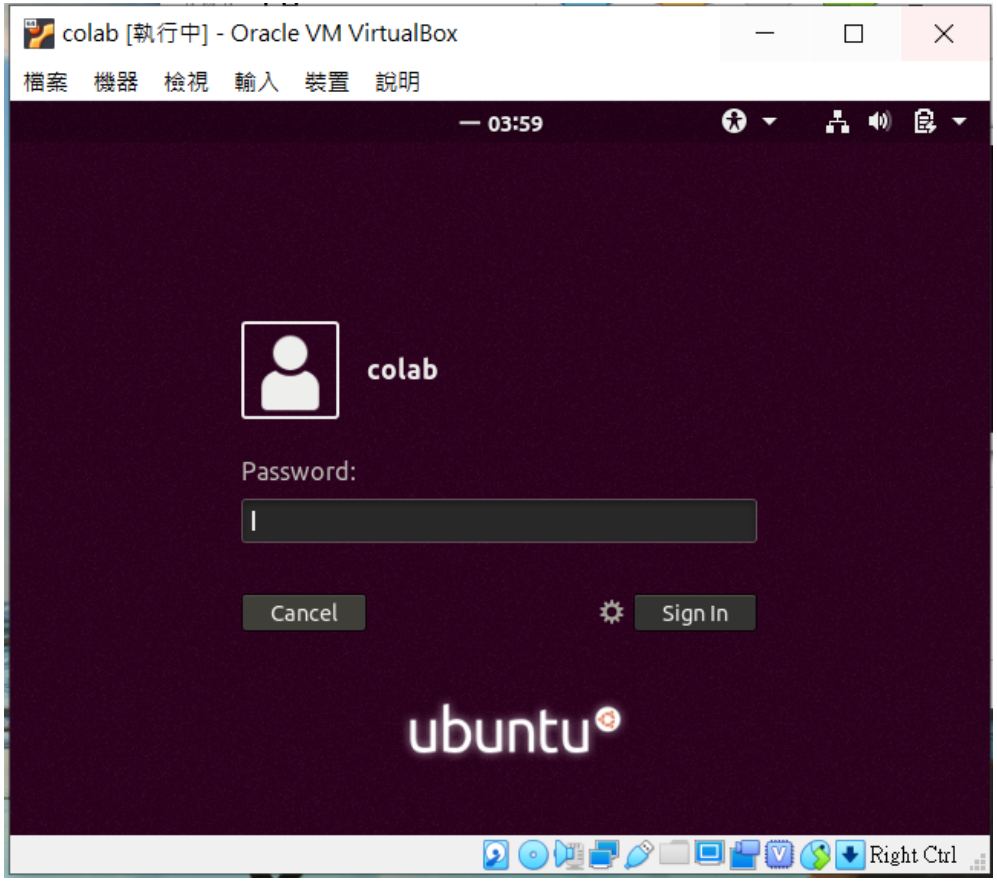
- 直接點選“匯入”，等待幾分鐘 -> 完成



Ubuntu

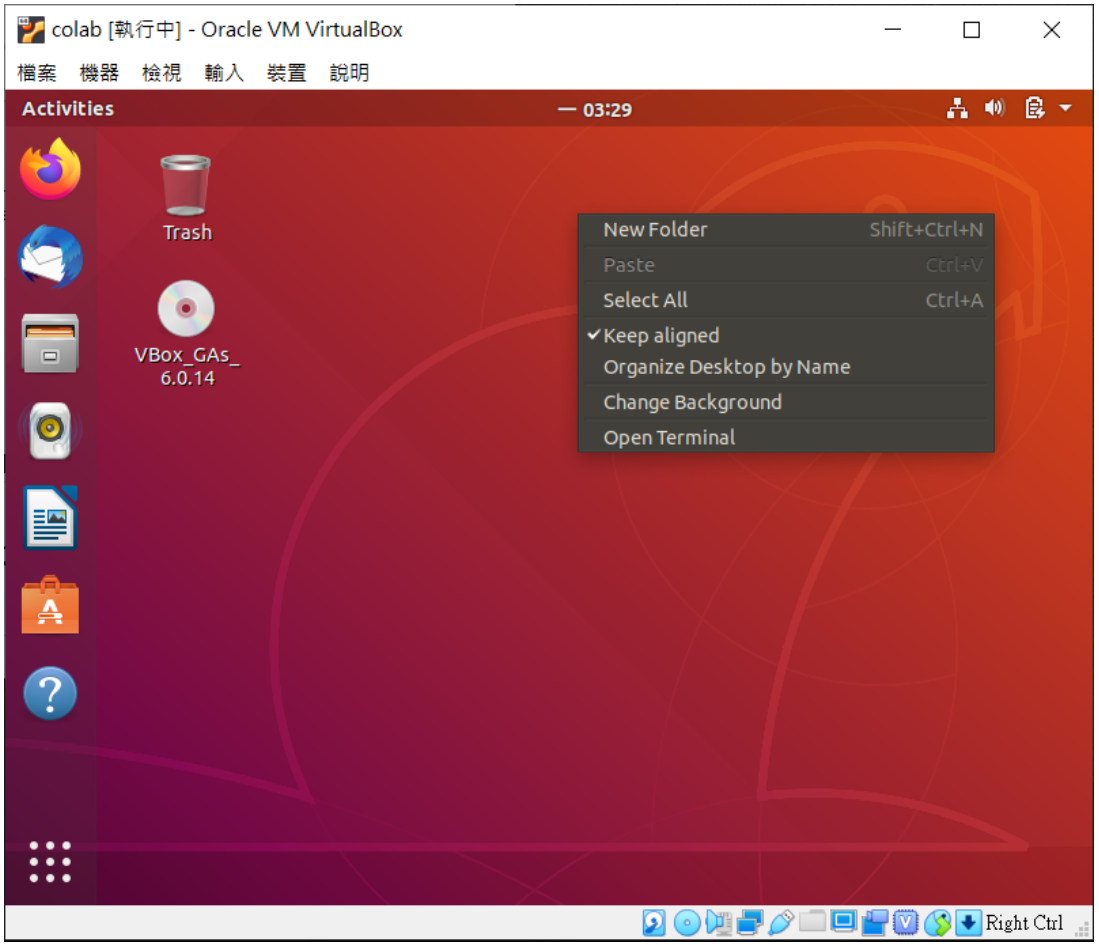
Ubuntu

- 密碼：co2020



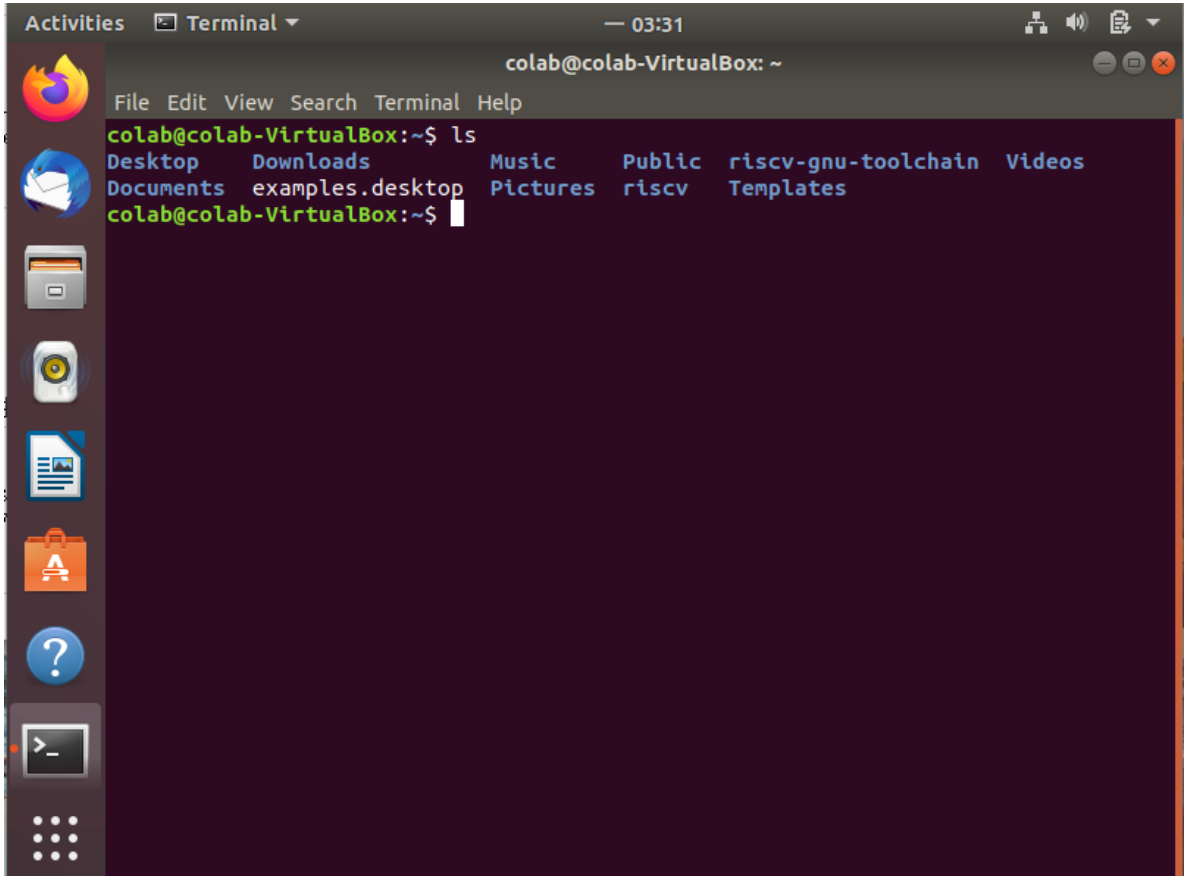
Terminal

- 打開終端機：右鍵 -> Open Terminal



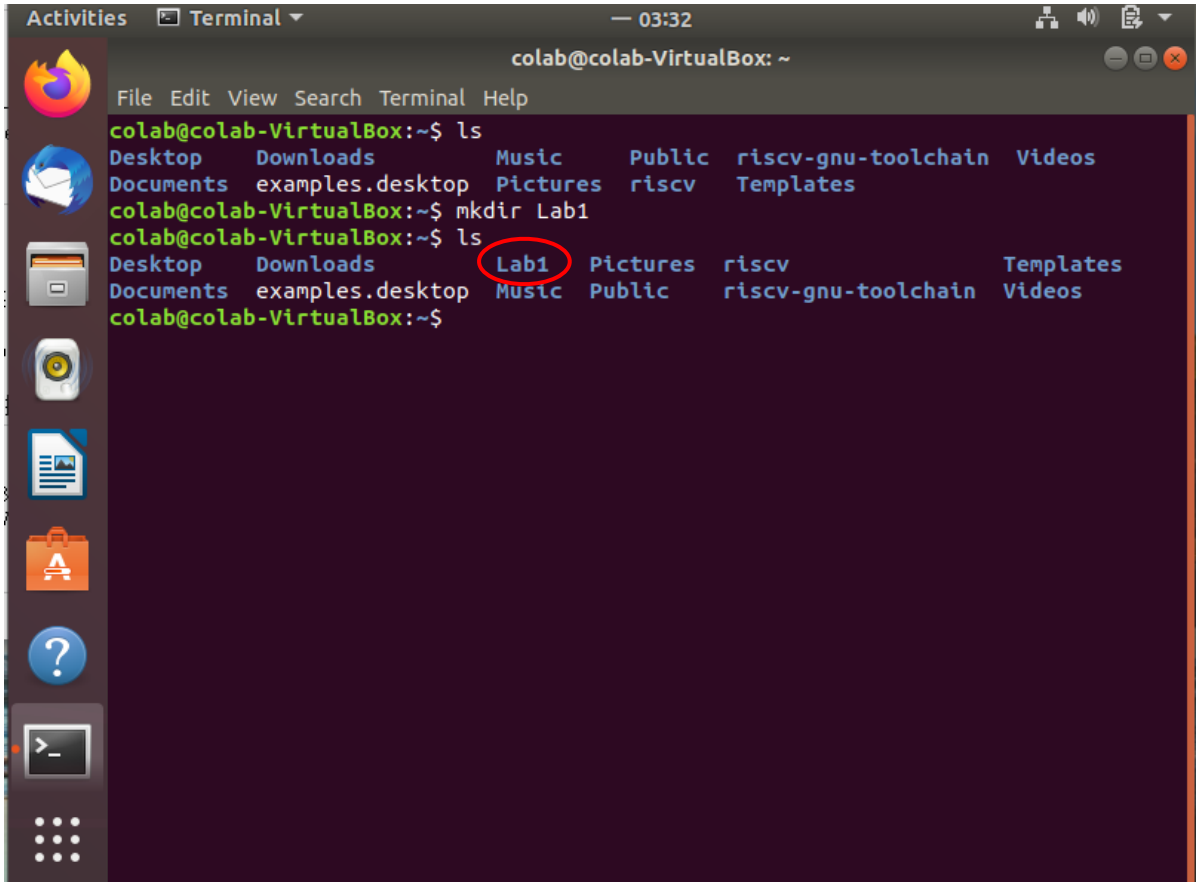
Linux 指令

- ls
  - ➢ 顯示檔案名稱 (藍字為資料夾，白字為檔案)



Linux 指令

- mkdir
  - ➢ 建立新的目錄



Linux 指令

- cd Lab1
  - ➢ 變換工作路徑

```
colab@colab-VirtualBox:~$ cd Lab1
colab@colab-VirtualBox:~/Lab1$
```

- cd ..

```
colab@colab-VirtualBox:~/Lab1$ cd ..
colab@colab-VirtualBox:~$
```

- cd ~

```
colab@colab-VirtualBox:~/Lab1$ cd ~
colab@colab-VirtualBox:~$
```

```
colab@colab-VirtualBox:~/Lab1$ cd /home/colab
colab@colab-VirtualBox:~$
```

Linux 指令

cp

- 就是 copy 的意思
- 語法： cp 來源檔 目的地

```
colab@colab-VirtualBox:~$ ls
Desktop     examples.desktop  Pictures  riscv-gnu-toolchain  Videos
Documents   Lab1              Public    Templates
Downloads   Music             riscv     test.c
colab@colab-VirtualBox:~$ cp test.c Lab1/
colab@colab-VirtualBox:~$ cd Lab1
colab@colab-VirtualBox:~/Lab1$ ls
test.c
colab@colab-VirtualBox:~/Lab1$
```

cp –r Lab1 Lab2

- -r : 複製整個資料夾

```
colab@colab-VirtualBox:~$ ls
Desktop     examples.desktop  Pictures  riscv-gnu-toolchain  Videos
Documents   Lab1              Public    Templates
Downloads   Music             riscv     test.c
colab@colab-VirtualBox:~$ cp -r Lab1 Lab2
colab@colab-VirtualBox:~$ ls
Desktop     examples.desktop  Music     riscv                test.c
Documents   Lab1              Pictures  riscv-gnu-toolchain  Videos
Downloads   Lab2              Public    Templates
colab@colab-VirtualBox:~$ cd Lab2
colab@colab-VirtualBox:~/Lab2$ ls
test.c
colab@colab-VirtualBox:~/Lab2$
```

Linux 指令

- Ctrl + C
  - ➢ 強制中斷執行程式
- Tab
  - ➢ 自動補齊指令
- ↑
  - ➢ 顯示前一個命令

Share folder - Virtualbox

From Windows to Ubuntu

- Windows :

  將檔案放進桌面的Shared資料夾

- Linux :

  ```
  cp /home/colab/shared/{filename} {PATH}
  ```

From Ubuntu to Windows

- Linux :

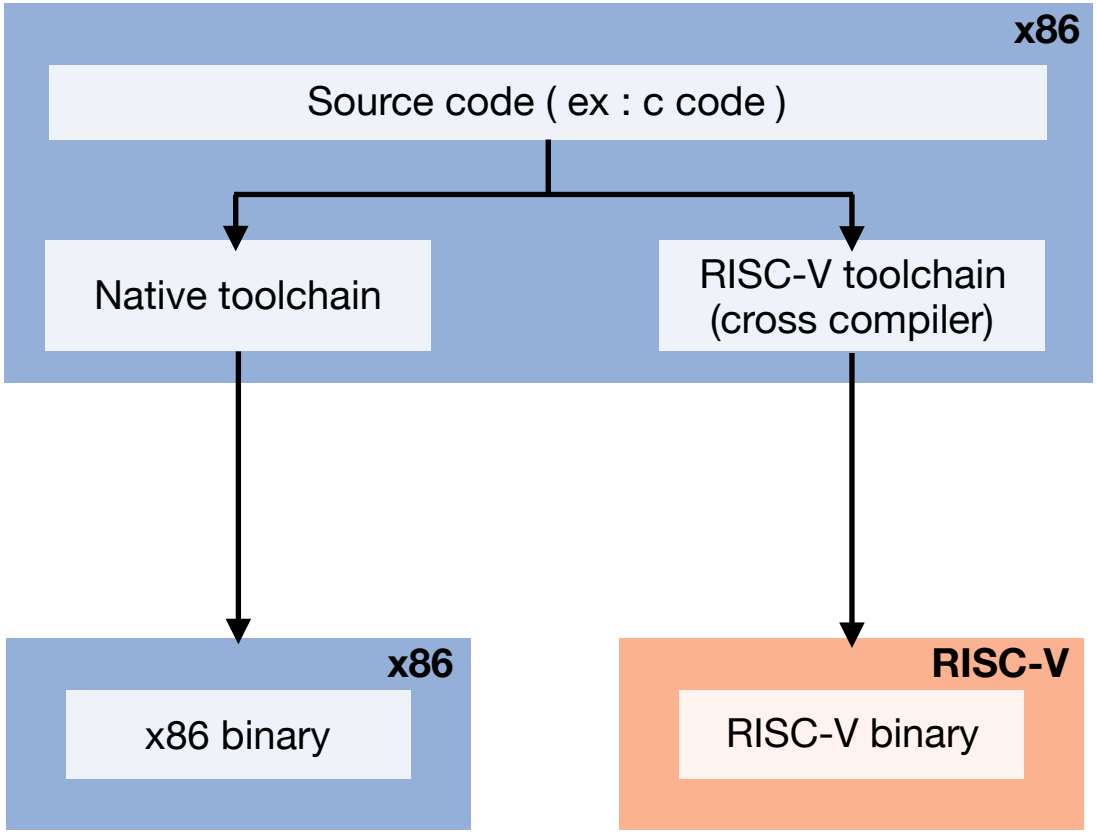
  ```
  cp {PATH}/{filename} /home/colab/shared/
  ```

- Windows :

  至桌面Shared資料夾拿檔案

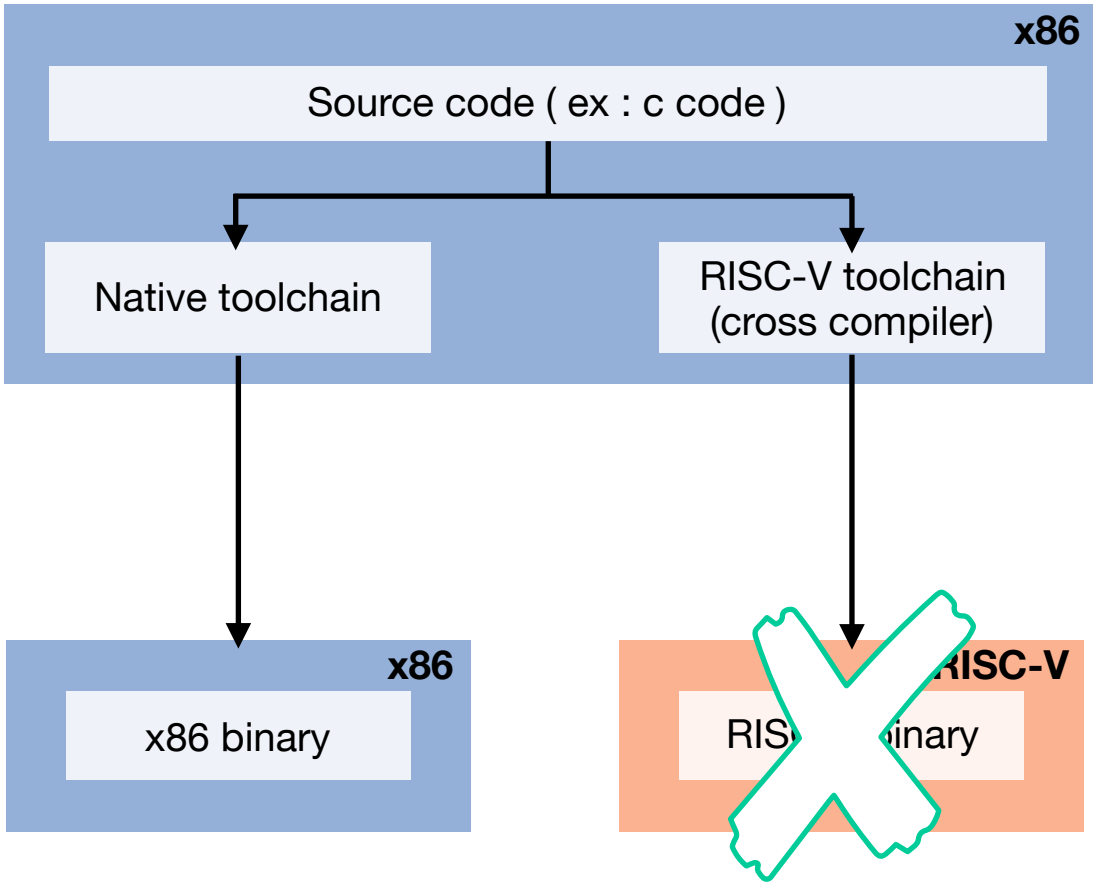
RISC-V Cross Compiler

RISC-V Cross compiler

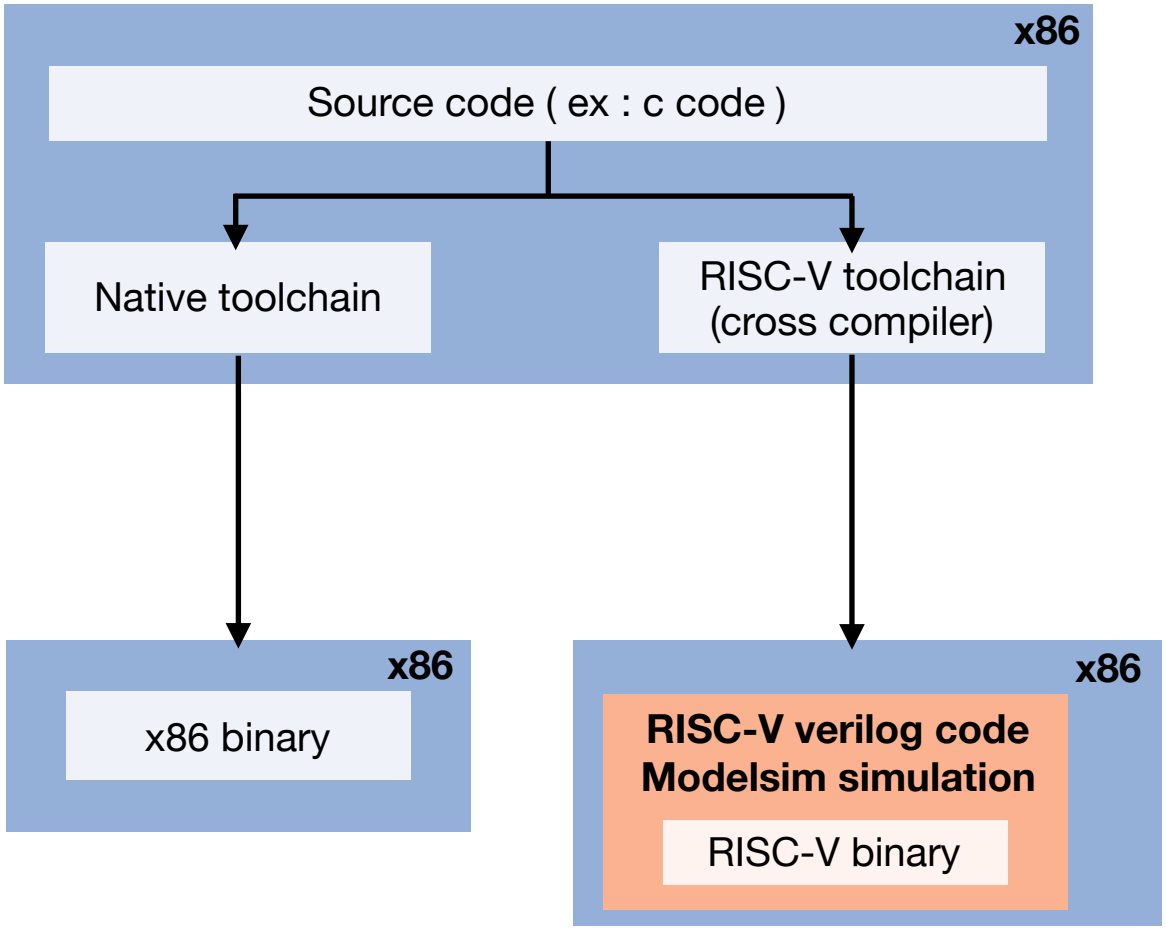


Modelsim

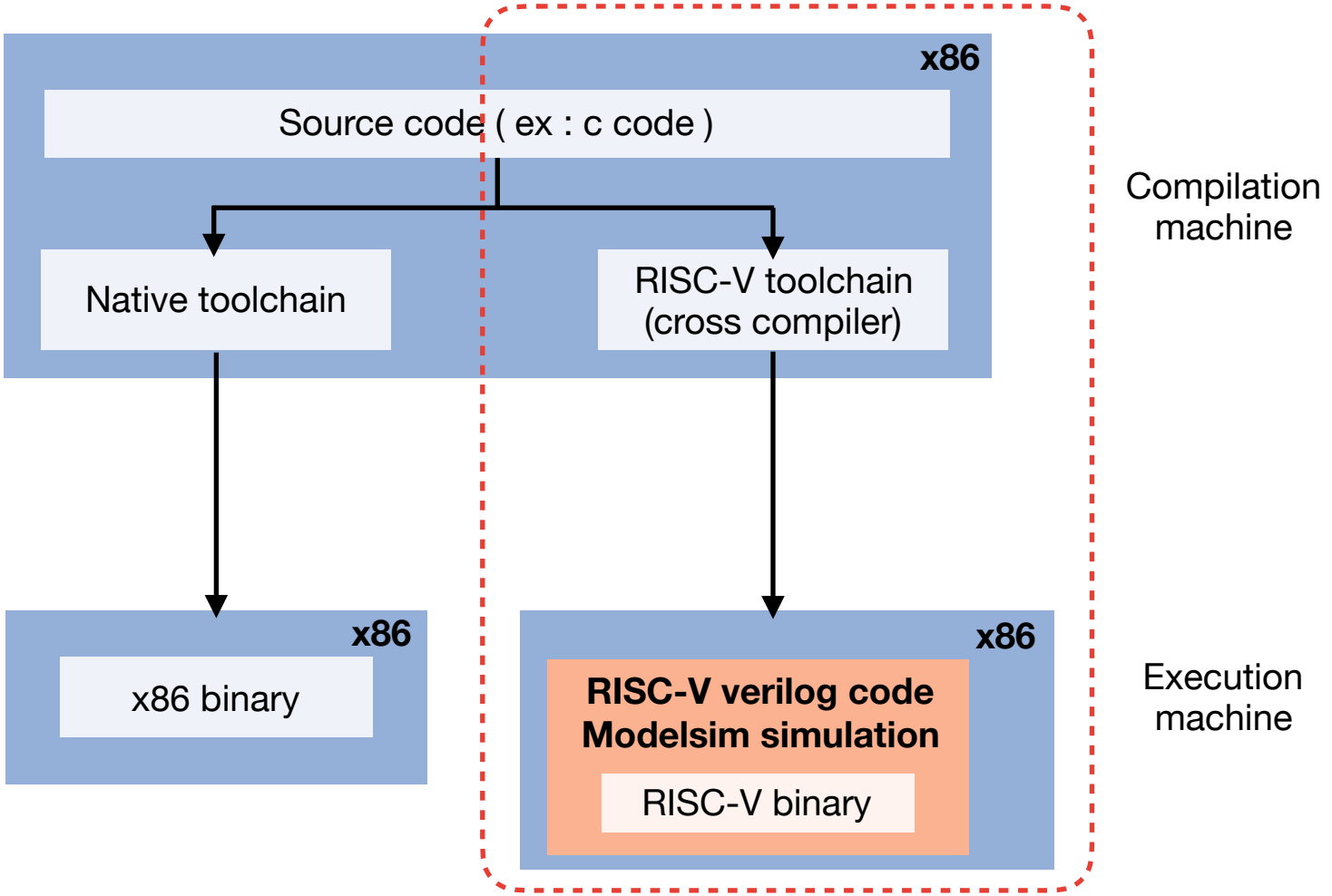
Modelsim


x86
Source code (ex : c code)
Native toolchain
RISC-V toolchain
(cross compiler)
x86
x86 binary
RISC-V
Compilation
machine
Execution
machine

Modelsim


x86
Source code (ex : c code)
Native toolchain
RISC-V toolchain
(cross compiler)
x86
x86 binary
x86
RISC-V verilog code
Modelsim simulation
RISC-V binary
Compilation
machine
Execution
machine

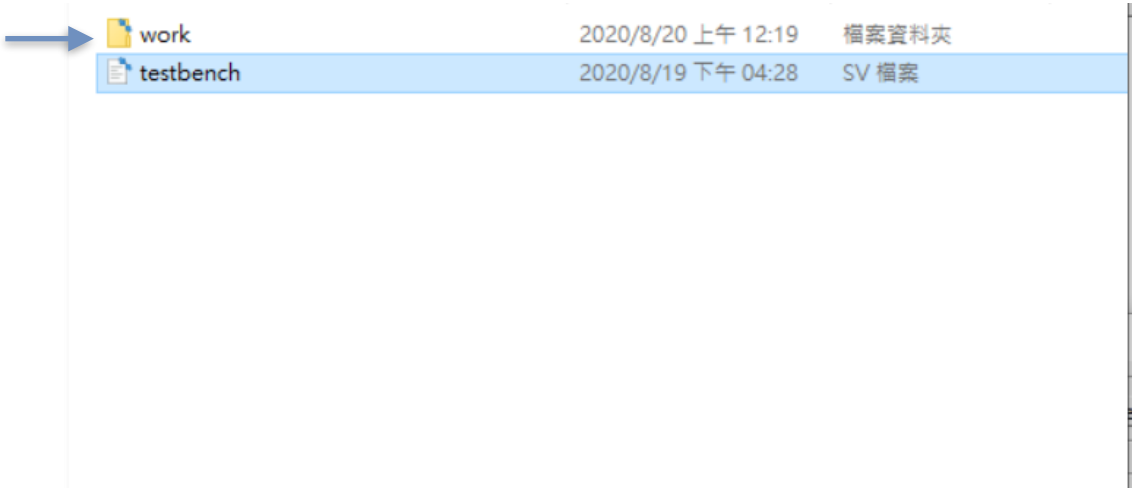
Modelsim


x86
Source code (ex : c code)
Native toolchain
RISC-V toolchain
(cross compiler)
Compilation
machine
x86
x86 binary
x86
RISC-V verilog code
Modelsim simulation
RISC-V binary
Execution
machine

Modelsim Install

- 實驗室電腦已安裝Modelsim

- 個人電腦安裝Modelsim，請參考”modelsim安裝.pdf”
  - 因版本不同，請將要跑的Lab資料夾裡面的work/替換成”modelsim安裝.zip”底下的 work/



Example

使用範例程式 Lab1 / example / add.s

Transform assembly code to binary code

Step.1: transform assembly code to object file

在**linux**下打開**Terminal**接著輸入以下指令

riscv32-unknown-elf-as -mabi=ilp32 add.s -o add.o

① riscv32-unknown-elf-as：讓 Assembly code 轉成 Object file

② -mabi=ilp32：ABI函數調用規則

③ add.s：範例檔案(assembly code)

④ -o：指定產生的目標檔名稱

add.s

```
1  nop
2  li      x12, 1
3  li      x13, 3
4  add     x12, x12, x13
5  ret
```

Transform assembly code to binary code

Step.2: link object file to elf file

輸入

riscv32-unknown-elf-ld -b elf32-littleriscv -T link.ld add.o -o add.elf

① riscv32-unknown-elf-ld：可將多個Object file link (add.o) 壓縮成為一個可執行檔(add.elf)

② -b elf32-littleriscv：指定執行檔格式

③ -T link.ld：使用助教提供的link腳本(link.ld)，指定code在memory的起始位置

④ -o：指定產生的可執行檔名稱

Transform assembly code to binary code

Step.3: generate a file for debugging

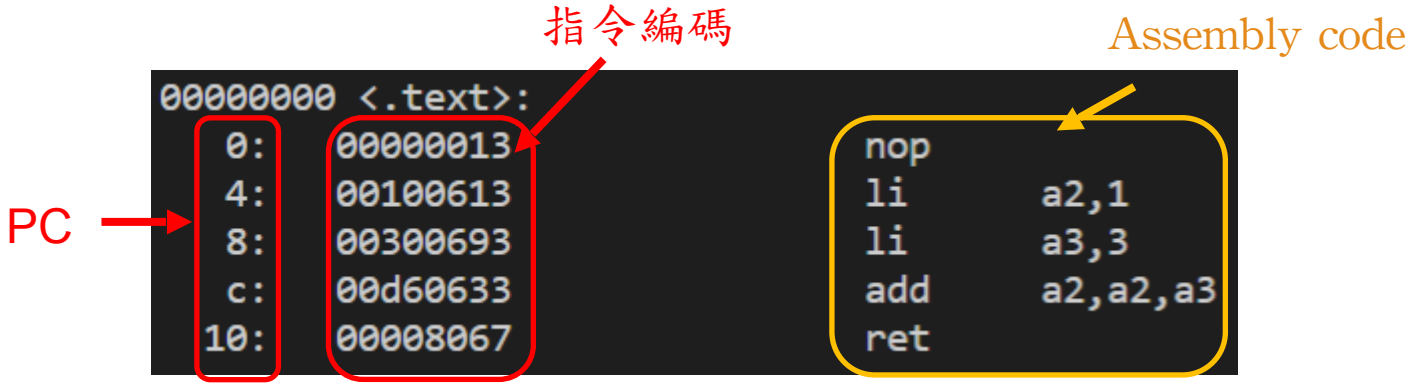
輸入

riscv32-unknown-elf-objdump -dC add.elf > add.dump

① riscv32-unknown-elf-objdump：將 elf 可執行檔轉換成容易debug 的格式，如下圖

② -dC：拆開指令

- 打開 add.dump 可看到如下圖



Transform assembly code to binary code

Step.4: transform elf file to binary

輸入

riscv32-unknown-elf-objcopy -O binary add.elf add.bin

① riscv32-unknown-elf-objcopy：轉換格式，將輸入文件內容拷貝到輸出的目標檔案中。

② -O binary：指定輸出目標的格式，這裡指定為 binary

Transform assembly code to binary code

Step.5: Convert binary to Memory Initialization File

輸入

python3 bin2mem.py --bin add.bin

① bin2mem.py：助教提供的 python 程式，將輸入 binary 檔案輸出為

add.mem (待會 modelsim 驗證要使用)

② add.mem：初始化CPU的 instruction memory

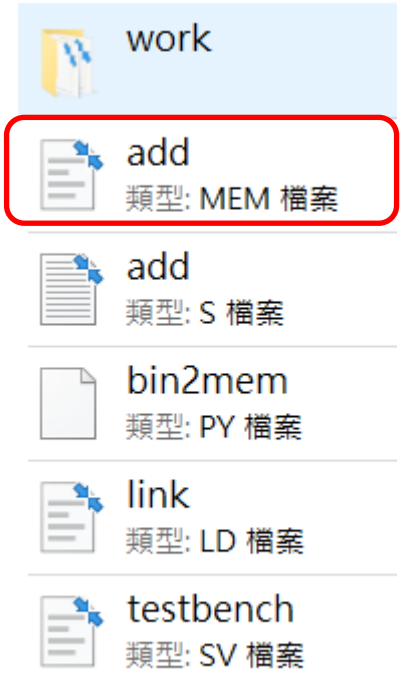
- 輸出結果：

```
processing add.bin
program size: 20 bytes
Memory file: add.mem
```

透過 share folder 將 mem 檔案傳到windows 並放入對應的lab資料夾

- Lab1/example

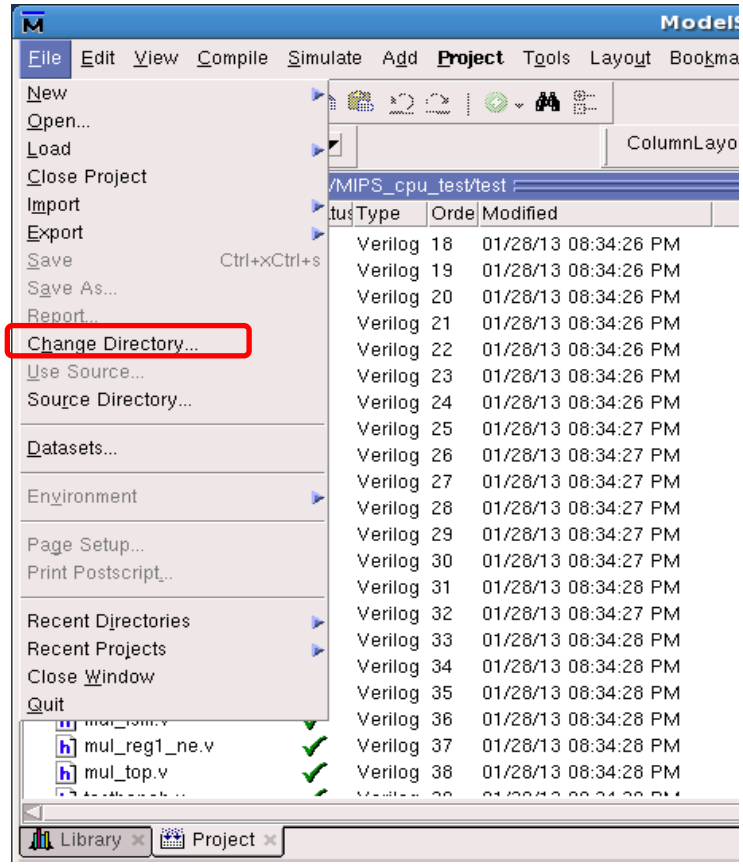
```
caslab-sh:~/colab/lab1_2020/example$ ll
al 44
xrwxr-x 2 sh sh 4096 Sep  1 14:42 ./
xrwxr-x 5 sh sh 4096 Sep  1 14:32 ../
xrwxr-x 1 sh sh   20 Sep  1 14:41 add.bin*
-rw-r-- 1 sh sh  257 Sep  1 14:36 add.dump
xrwxr-x 1 sh sh 4484 Sep  1 14:36 add.elf*
-rw-r-- 1 sh sh   45 Sep  1 14:42 add.mem
-rw-r-- 1 sh sh  564 Sep  1 14:35 add.o
-rw-r-- 1 sh sh   49 Sep  1 14:33 add.s
-r--r-- 1 sh sh  935 Sep  1 14:33 bin2mem.py
-r--r-- 1 sh sh   80 Sep  1 14:33 link.ld
```

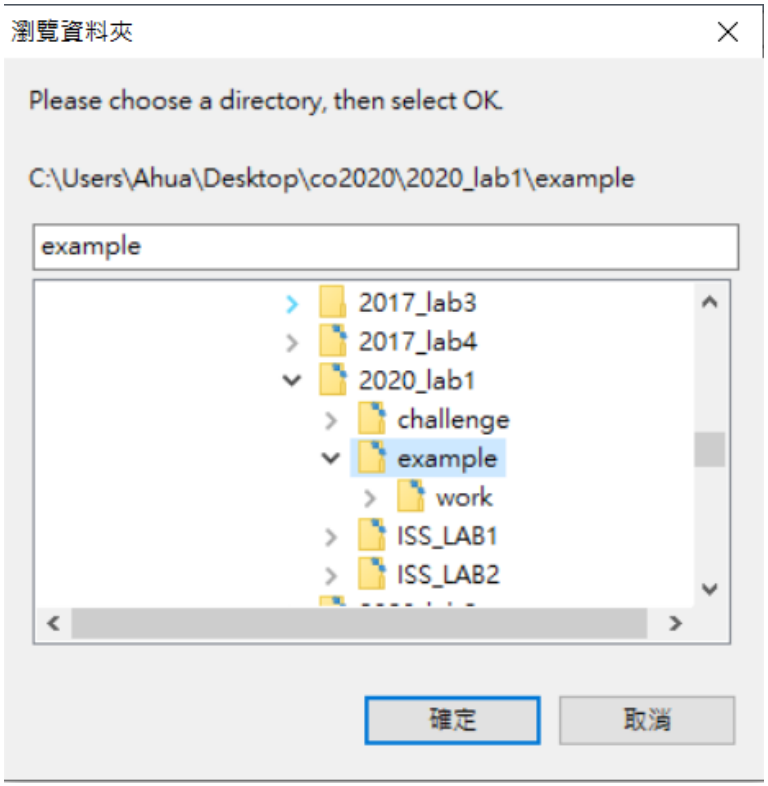


Modelsim 驗證教學

Step.1: change to your file location

打開modelsim後,在File下選擇change directory到你放*work*資料夾的地方

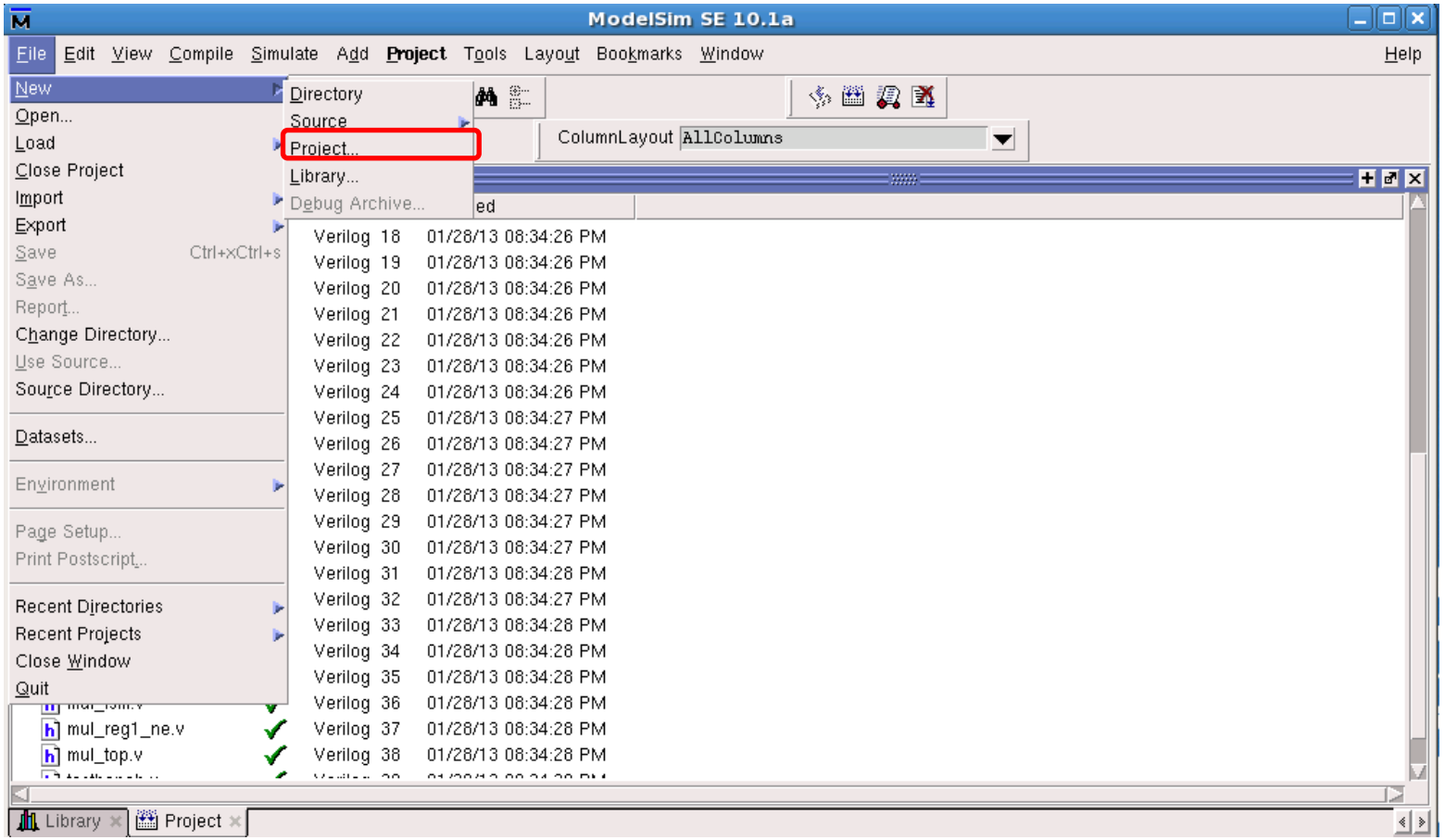




Modelsim 驗證教學

Step.2: new project

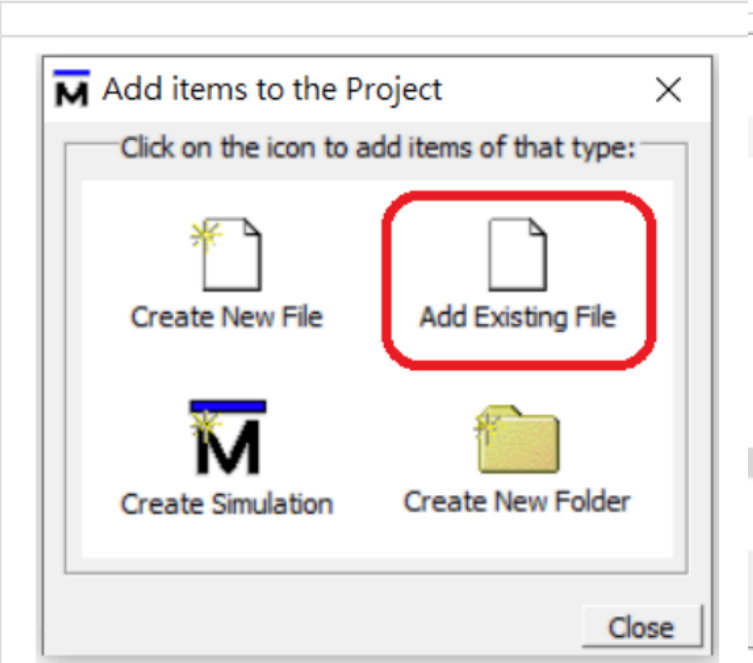
接著new一個project(名稱自訂)

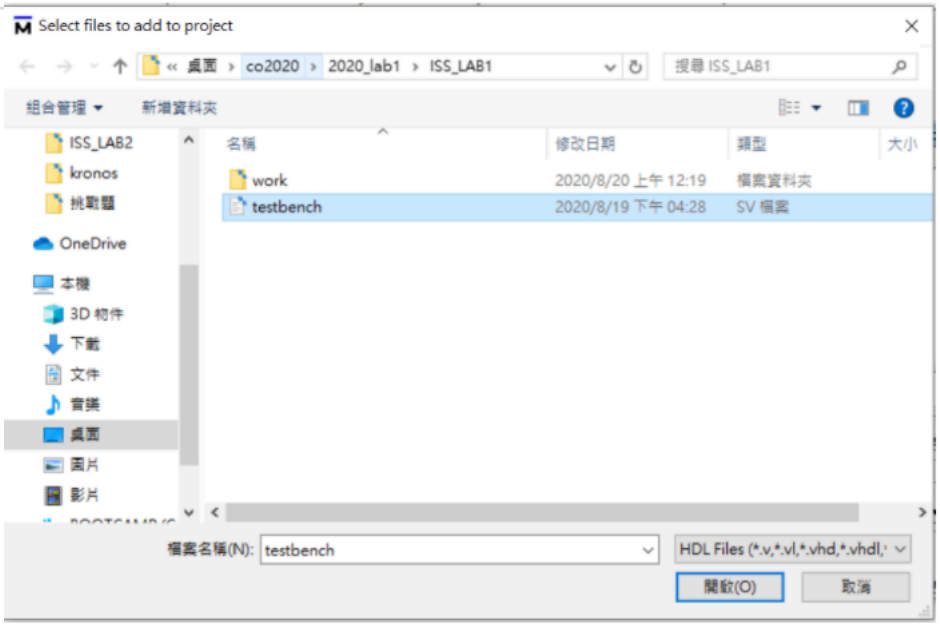


Modelsim 驗證教學

Step.3: add source code

新增完project後,會跳出一個視窗(如圖),點選*Add Existing File*將 source code加入到這個project中





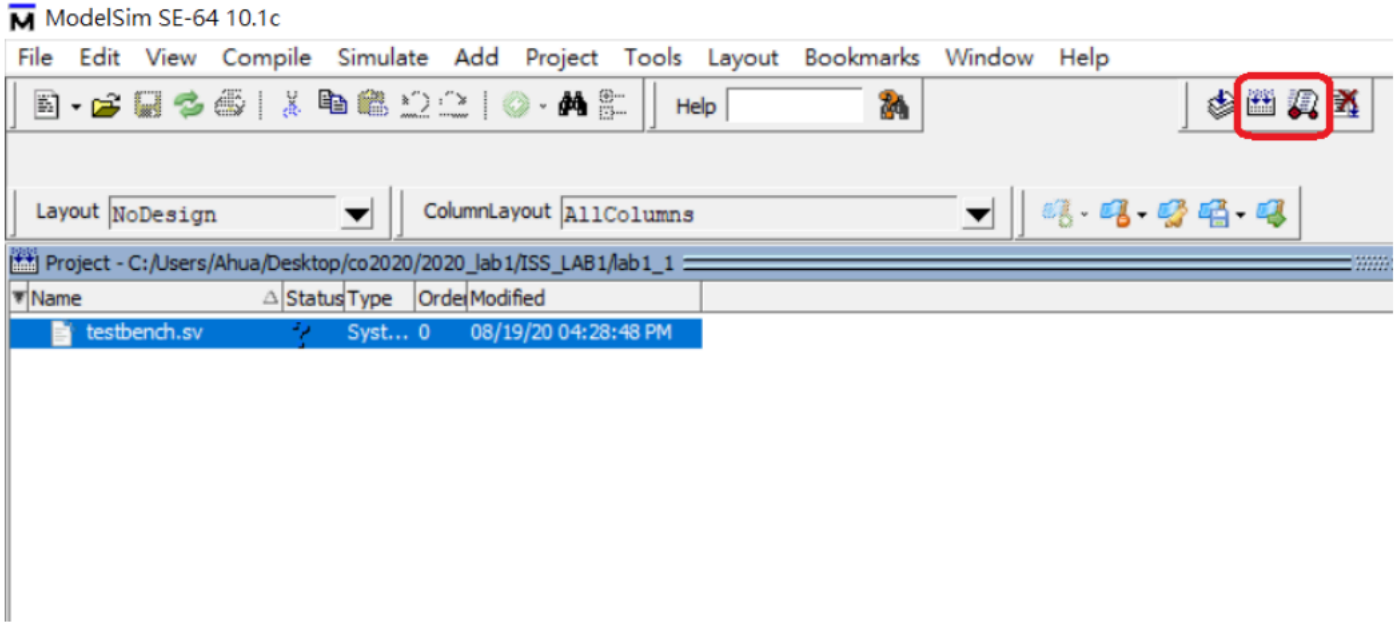
Modelsim 驗證教學

Step.4: compile and simulate source code

點擊testbench.sv

點擊compile 

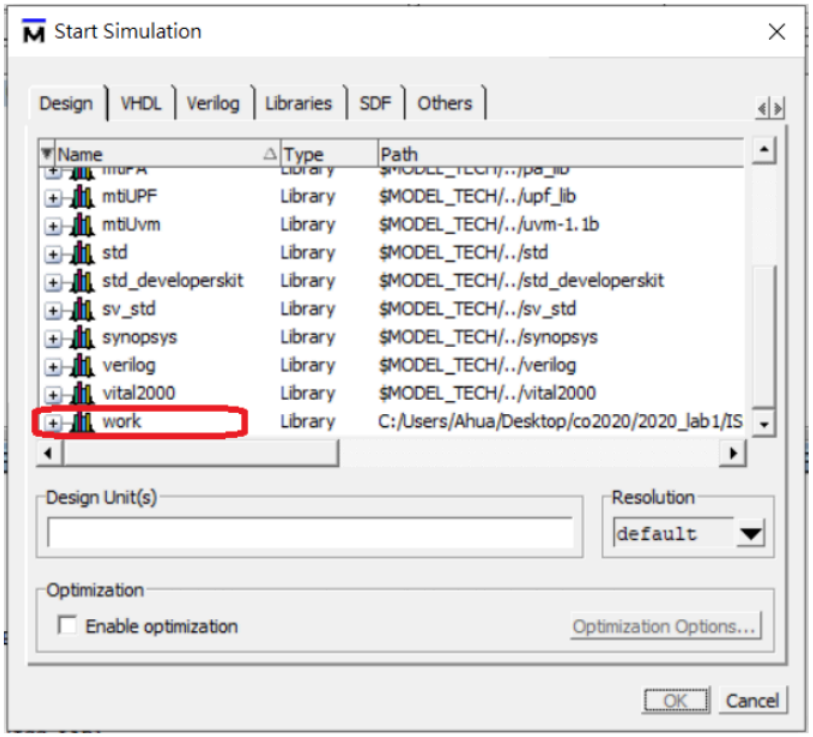
接著點擊*simulate* 

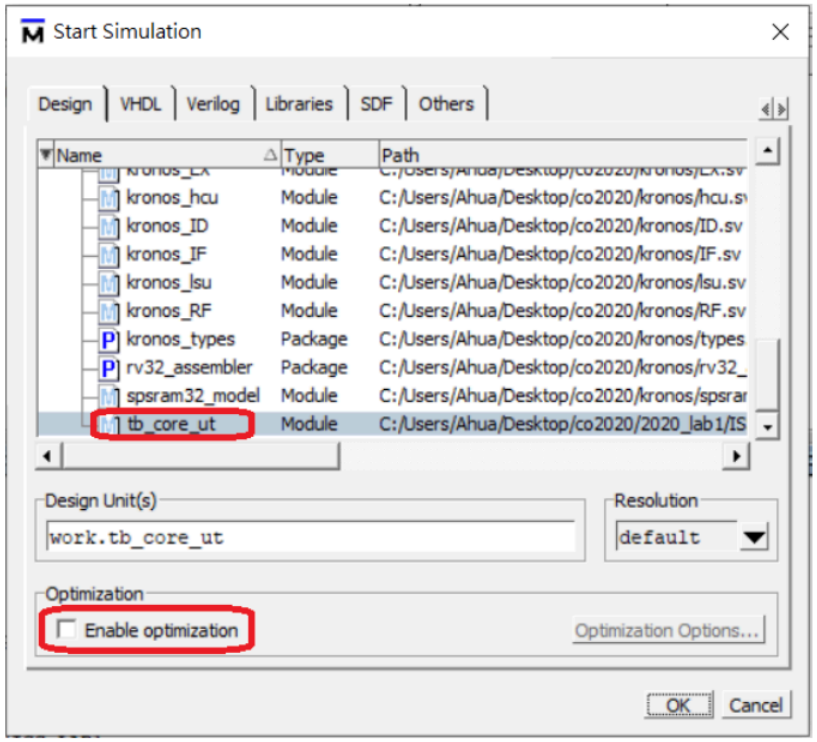


Modelsim 驗證教學

Step.5: choose testbench

打開work後選tb_core_ut並取消最佳化(Enable optimization)



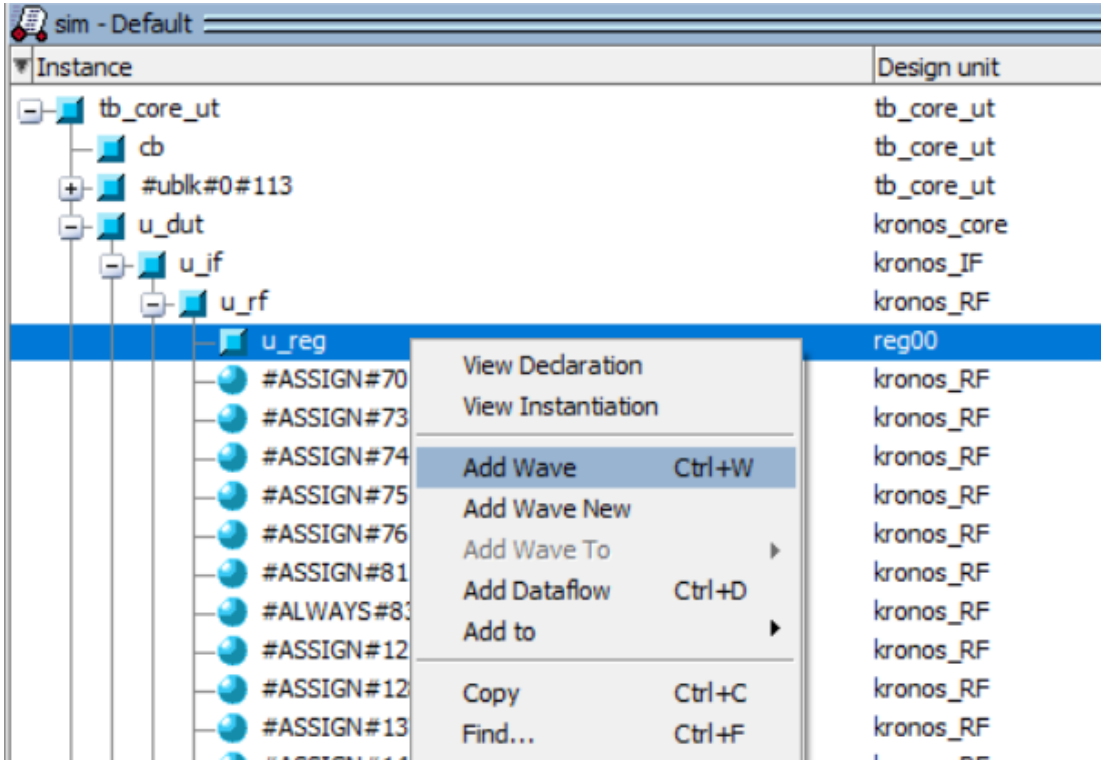


Modelsim 驗證教學

Step.6: add signal to wave

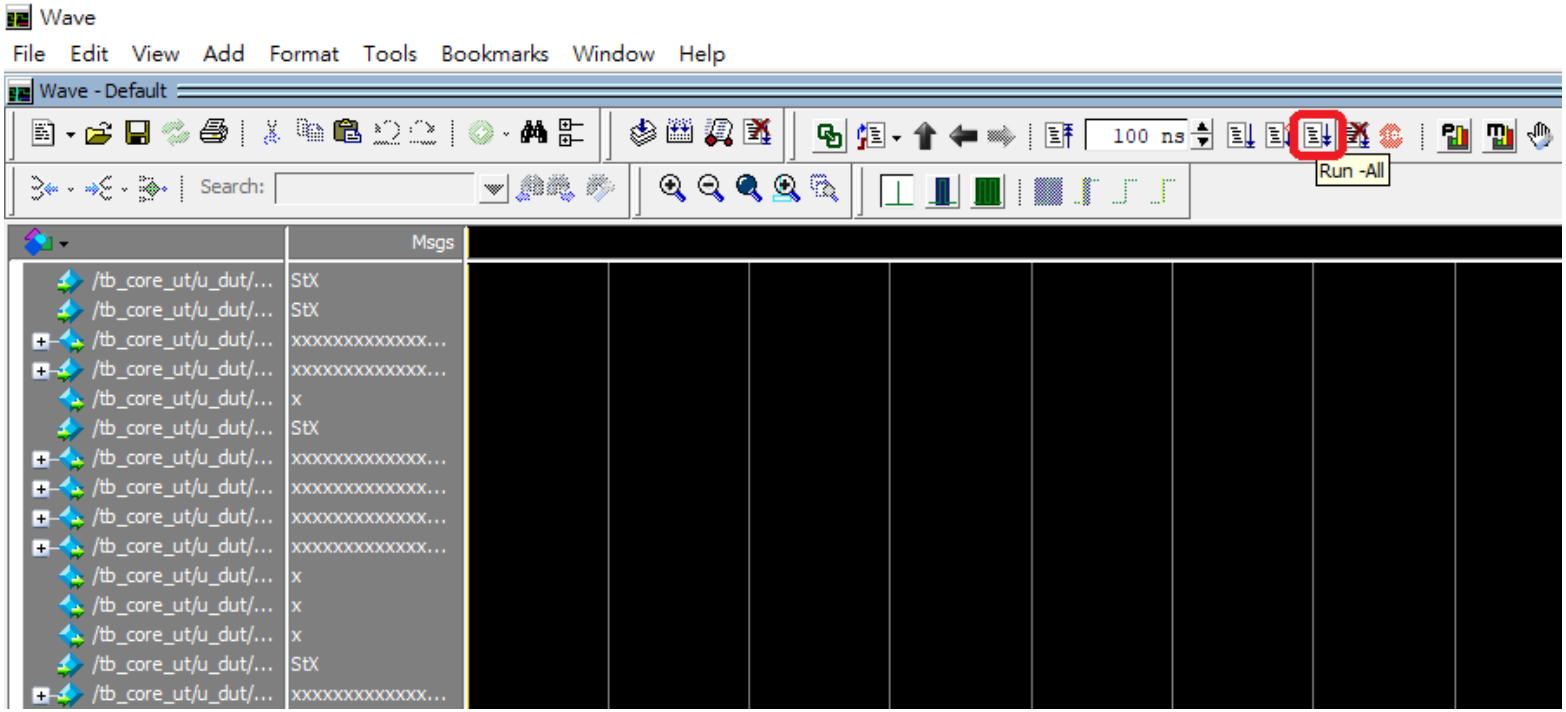
將你要看的訊號線按右鍵add wave

(後面的練習結果需要看u_reg的波形，路徑如下圖tb_core_ut > u_dut > u_if > u_rf > u_reg)



Modelsim 驗證教學

Step.7: run all

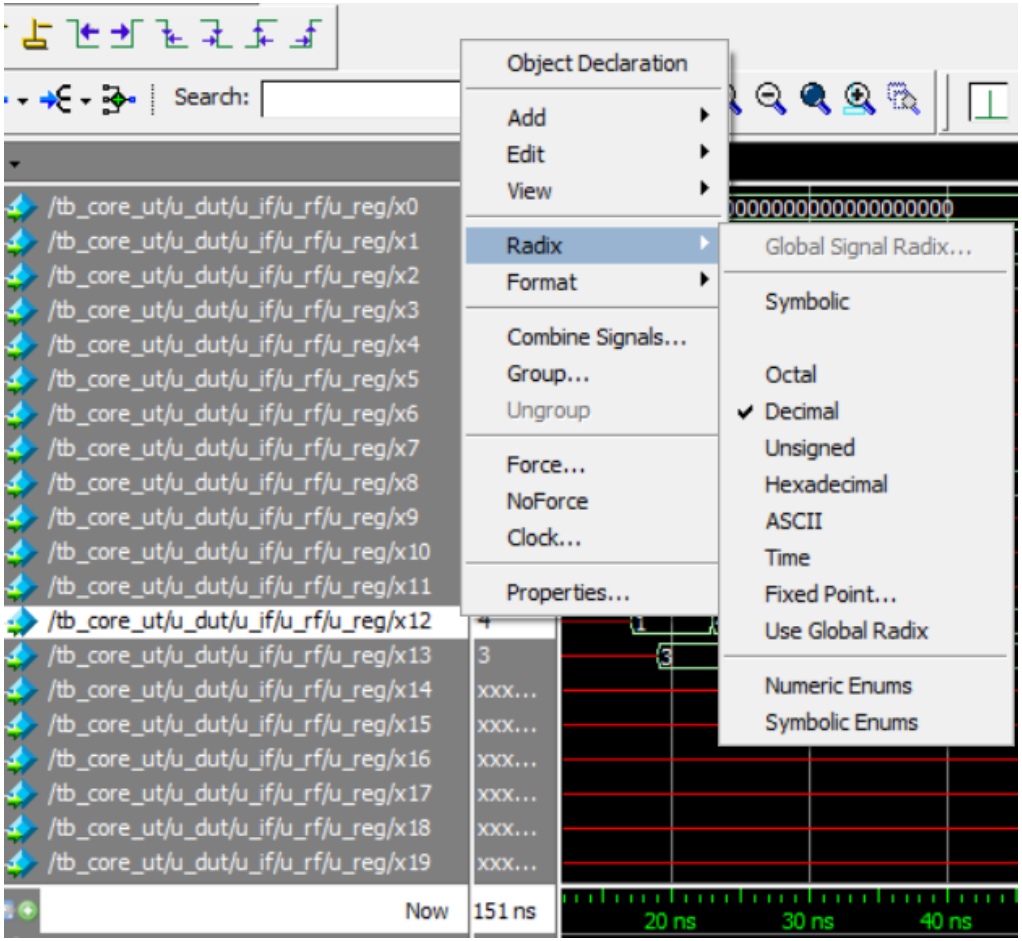


Modelsim 驗證教學

Step.8: check wave result

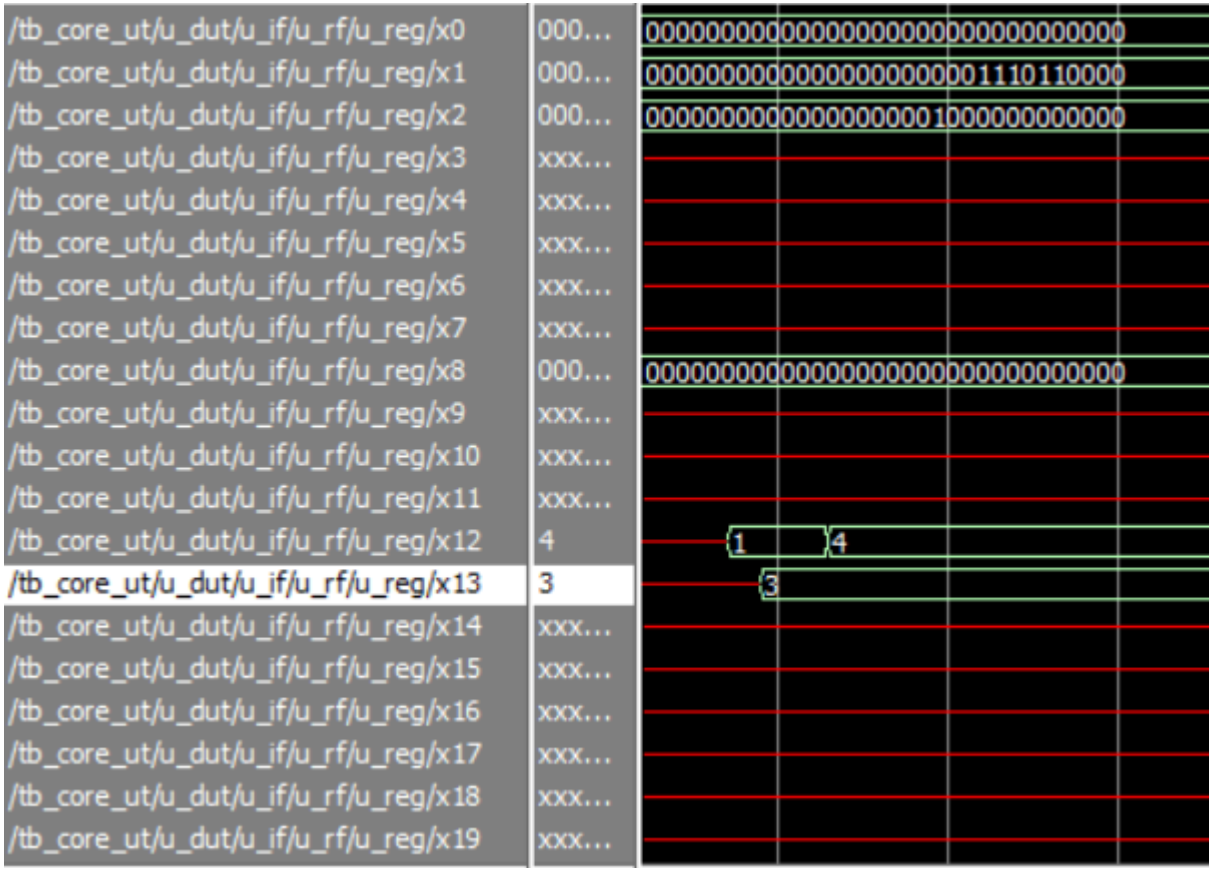
改進位顯示:對訊號線按右鍵 -> Radix

(Decimal:有號10進位; Unsigned:無號10進位; Hex:16進位)



Modelsim 驗證教學

可看到結果(x12, x13暫存器)與add.s程式預期結果符合

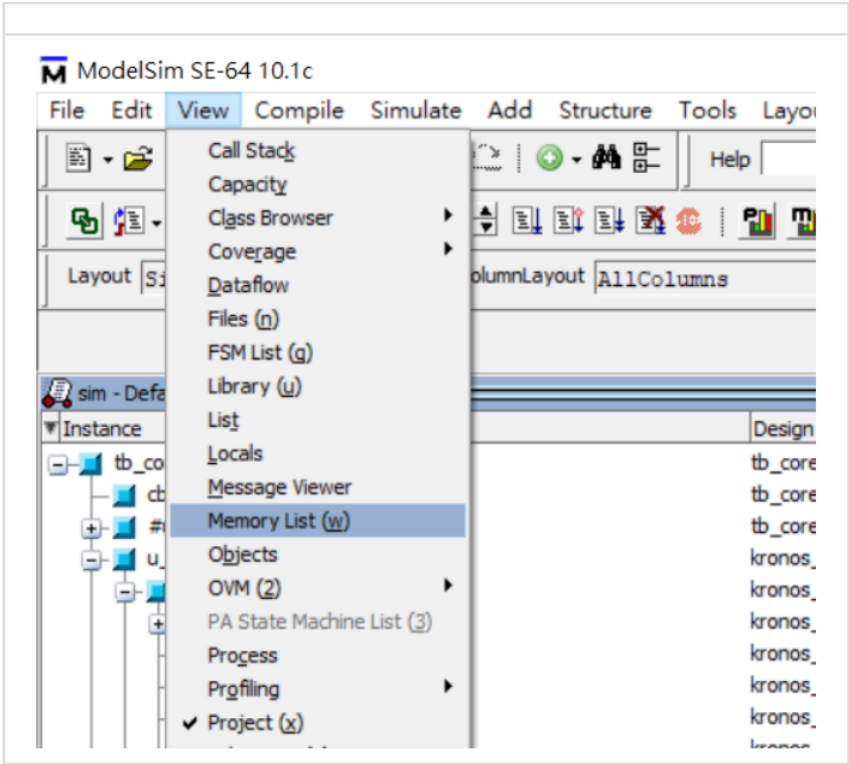


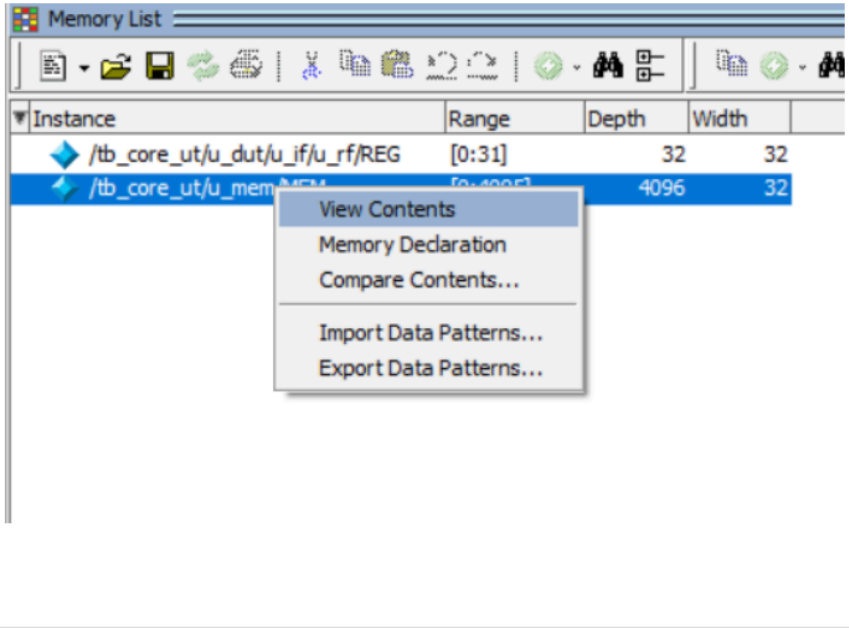
Modelsim 驗證教學

Step.9: check memory result (實作二之後會用到)

View -> Memory List

選擇 MEM 按右鍵 -> View Contents





Modelsim 驗證教學

Step.10: check memory result

Goto：要看的memory位置

(View -> Properties 選擇進位)