處理器設計與實作

CS250 組合語言與計算機組織

LAB2: Describe C code in MIPS assembly

實驗目的

1. 了解C程式與組合語言程式的關係
2. MIPS定址法
3. 比較LAB1進階題與自己所想的Assembly Code有何不同 (探討LAB1進階題)
4. 了解Assembly Code執行流程&結果

Tool used

實驗環境:

1. Linux
2. MIPS Cross compiler(Compile program)
3. Modelsim (Run CPU simulator)
4. Java

Example: Fibonacci數列

探討LAB1進階題 - Fibonacci數列

```
1    #include<stdio.h>
2    main()
3    {
4       int n, first = 0, second = 1, next, c;
5       n = 10 ;
6       volatile  int * base = (int*) 0x20000000;
7       for ( c = 0 ; c < n ; c++ )
8       {
9          if ( c <= 1 )
10            next = c;
11         else
12         {
13            next = first + second;
14            first = second;
15            second = next;
16         }
17         base[c]= next ;
18      }
19      return 0;
20   }
```

Fibonacci Assembly Code from gcc Compiler

```
main:
        .frame  $sp,0,$31               # vars= 0, regs= 0/0, args= 0, gp= 0
        .mask   0x00000000,0
        .fmask  0x00000000,0
        .set    noreorder
        .set    nomacro

        move    $6,$0
        li      $5,1                    # 0x1
        li      $8,10                   # 0xa
        li      $7,536870912                    # 0x20000000
        move    $4,$0
.L7:
        slt     $2,$4,2
        bne     $2,$0,.L6
        move    $3,$4

        addu    $3,$6,$5
        move    $6,$5
        move    $5,$3
.L6:
        sll     $2,$4,2
        addu    $2,$2,$7
        sw      $3,0($2)
        addiu   $4,$4,1
        slt     $2,$4,$8
        bne     $2,$0,.L7
        move    $2,$0

        j       $31
        nop
```

表示整個FOR迴圈(line:7-18)

表示IF判斷式(line:9-10)

表示ELSE判斷式(line:11-16)

表示每次把記憶體位址+4並store value(line:17)

Compile C Code to Assembly Code

將Ubuntu開機後，打開Terminal並輸入以下指令

```
[cs250a] sde-gcc -S fib.c -o fib.s
```

① -S: 僅將C code轉為Assembly code，不做後續的assemble跟link

```
ubuntu@ubuntu-VirtualBox:~/cs250a/share$ sde-gcc -S fib.c -o fib.s
ubuntu@ubuntu-VirtualBox:~/cs250a/share$ ls
bonus.c  call_function.c  converter  data_struct.c  fib.c  fib.s  MIPS_cpu_test
ubuntu@ubuntu-VirtualBox:~/cs250a/share$
```

Compile Assembly Code

輸入以下指令產生fib.conv

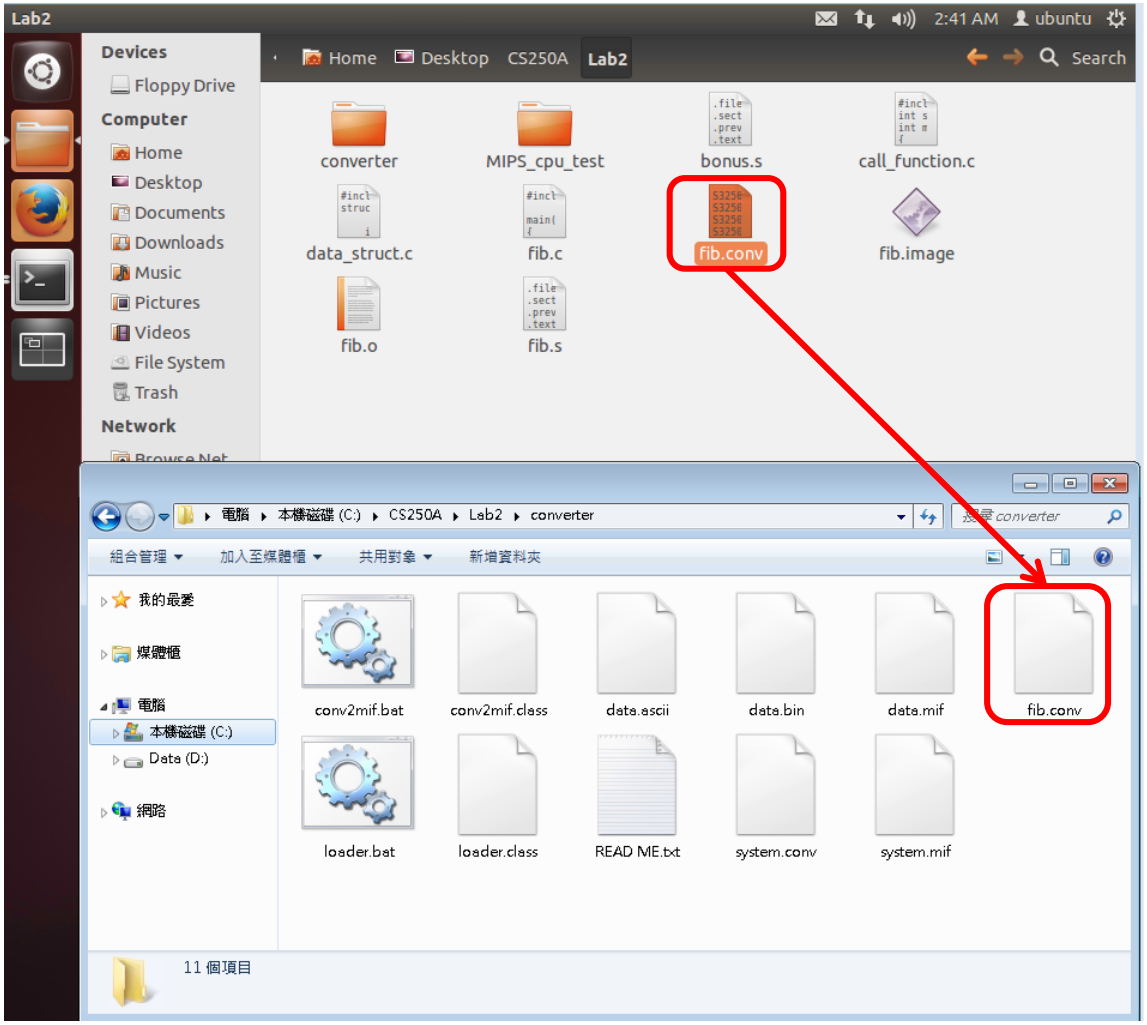
```
[cs250a] sde-as fib.s -g -o fib.o
[cs250a] sde-ld fib.o -Ttext 0x00008000 -o fib.image
[cs250a] sde-conv fib.image -o fib.conv
```

```
ubuntu@ubuntu-VirtualBox:~/cs250a/share$ sde-gcc -S fib.c -o fib.s
ubuntu@ubuntu-VirtualBox:~/cs250a/share$ ls
bonus.c  call_function.c  converter  data_struct.c  fib.c  fib.s  MIPS_cpu_test
ubuntu@ubuntu-VirtualBox:~/cs250a/share$ sde-as fib.s -g -o fib.o
ubuntu@ubuntu-VirtualBox:~/cs250a/share$ sde-ld fib.o -Ttext 0x00008000 -o fib.image
sde-ld: warning: cannot find entry symbol __start; defaulting to 0000000000008000
ubuntu@ubuntu-VirtualBox:~/cs250a/share$ sde-conv fib.image -o fib.conv
ubuntu@ubuntu-VirtualBox:~/cs250a/share$ ls
bonus.c          converter      fib.c     fib.image  fib.s
call_function.c  data_struct.c  fib.conv  fib.o      MIPS_cpu_test
ubuntu@ubuntu-VirtualBox:~/cs250a/share$
```

Copy conv file

將編好的fib.conv複製到Windows中以下路徑:

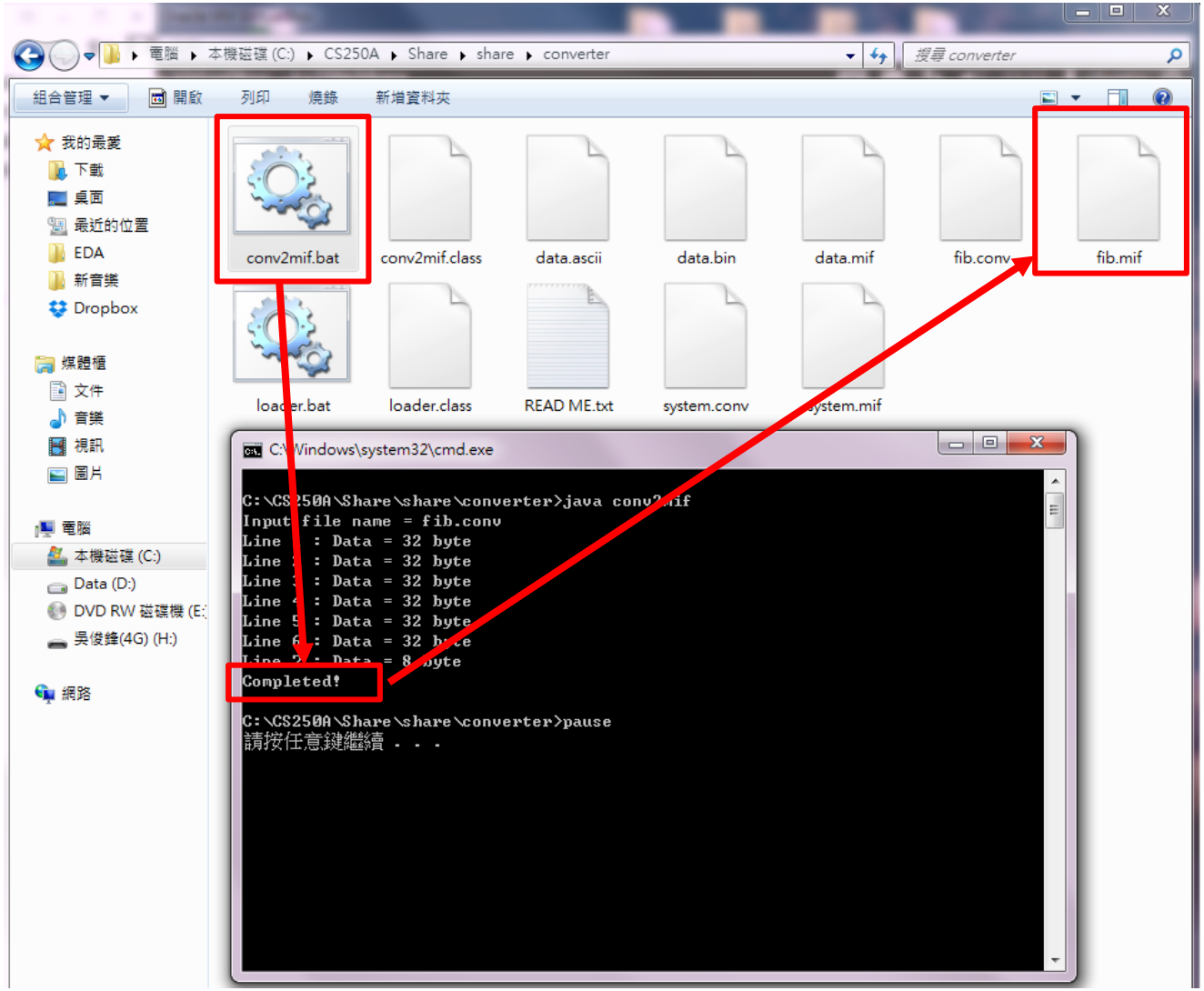
C:\CS250A\Lab2\converter



Conv2mif

在windows底下執行conv2mif.bat，輸入檔名fib.conv按下Enter

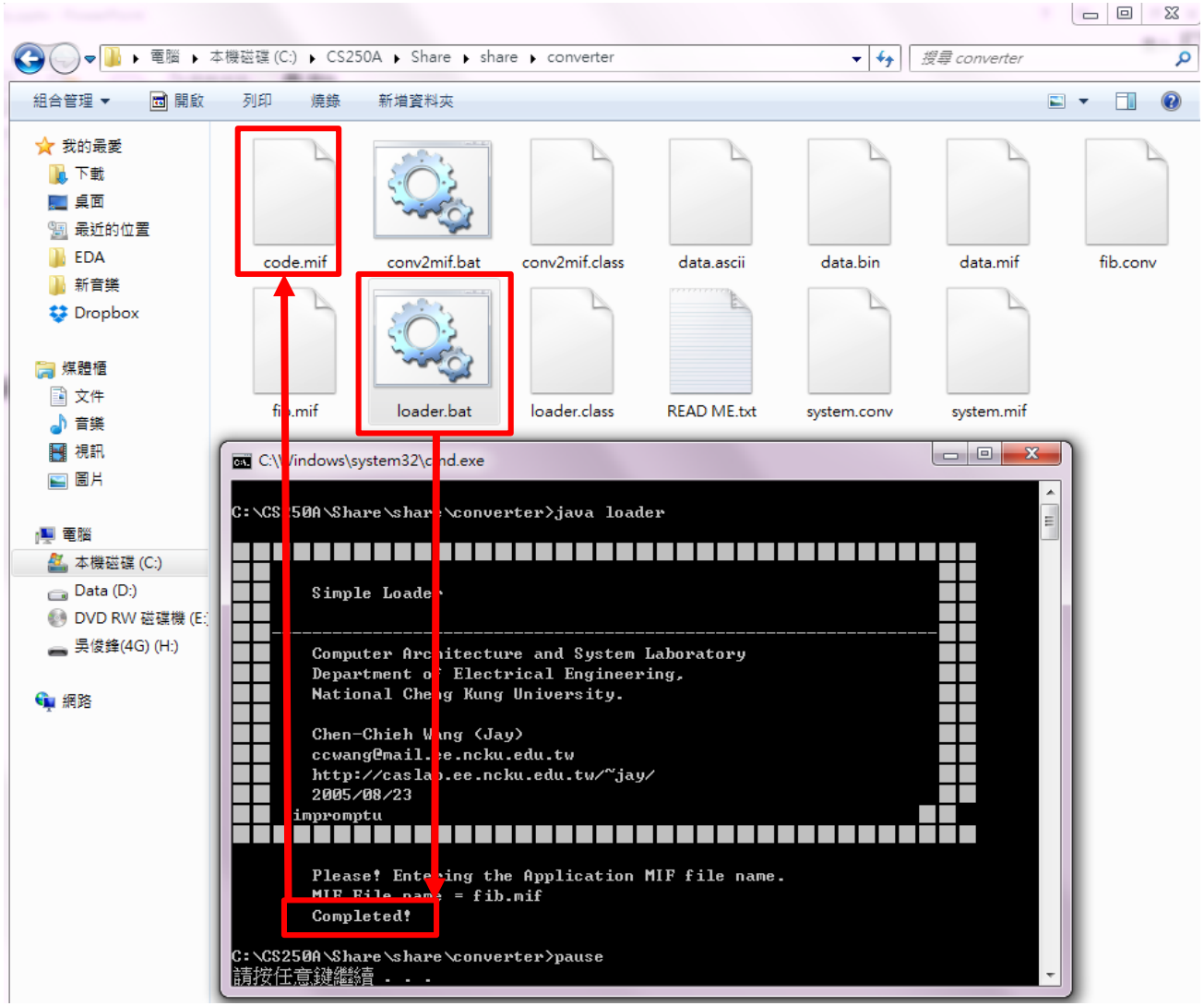
產生fib.mif檔



Loader

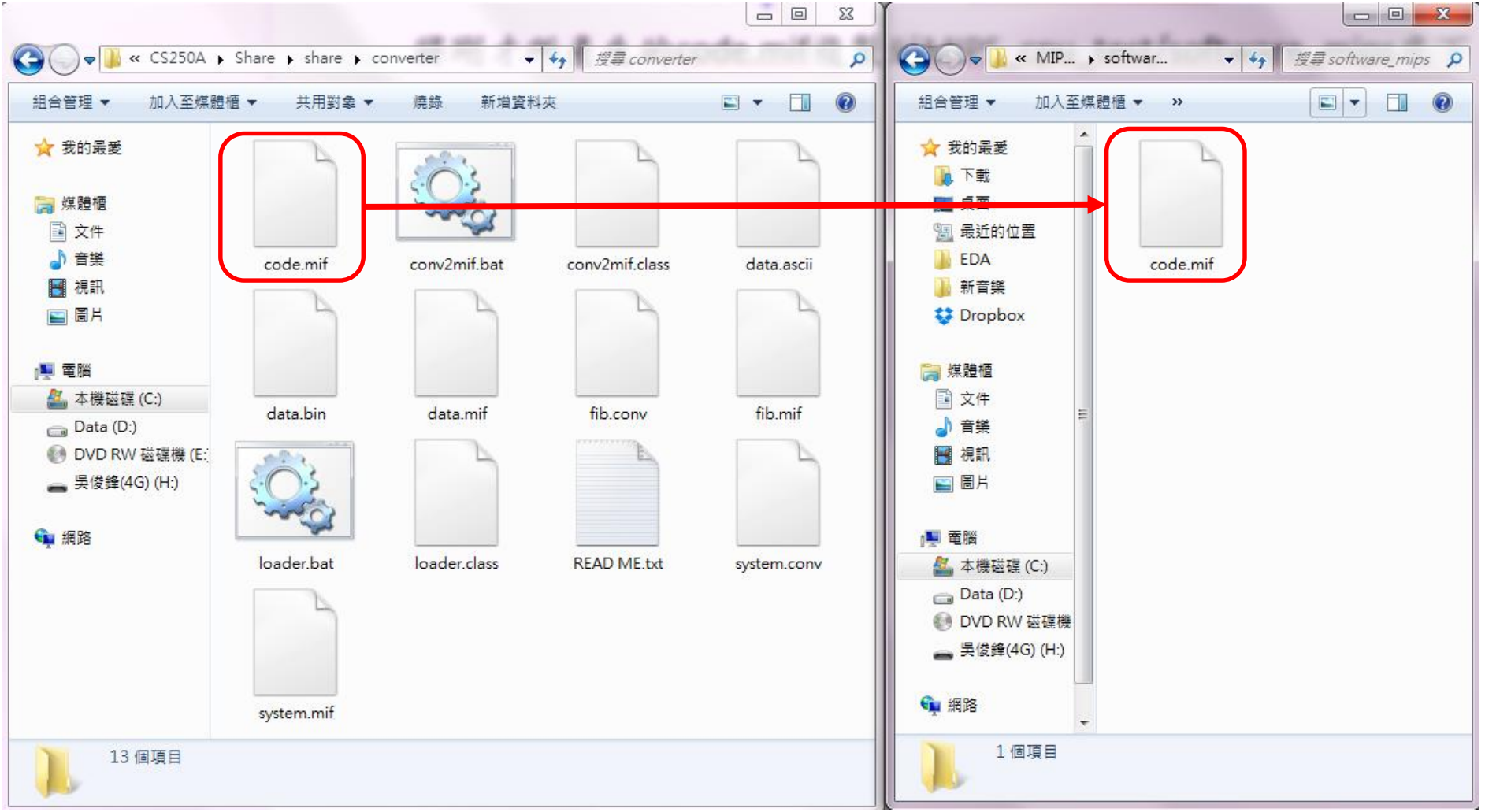
執行loader.bat，輸入剛才所產生的fib.mif按下Enter

產生code.mif檔



Prepare for ModelSim

將剛才所產生的code.mif複製到MIPS_cpu_test/software_mips底下

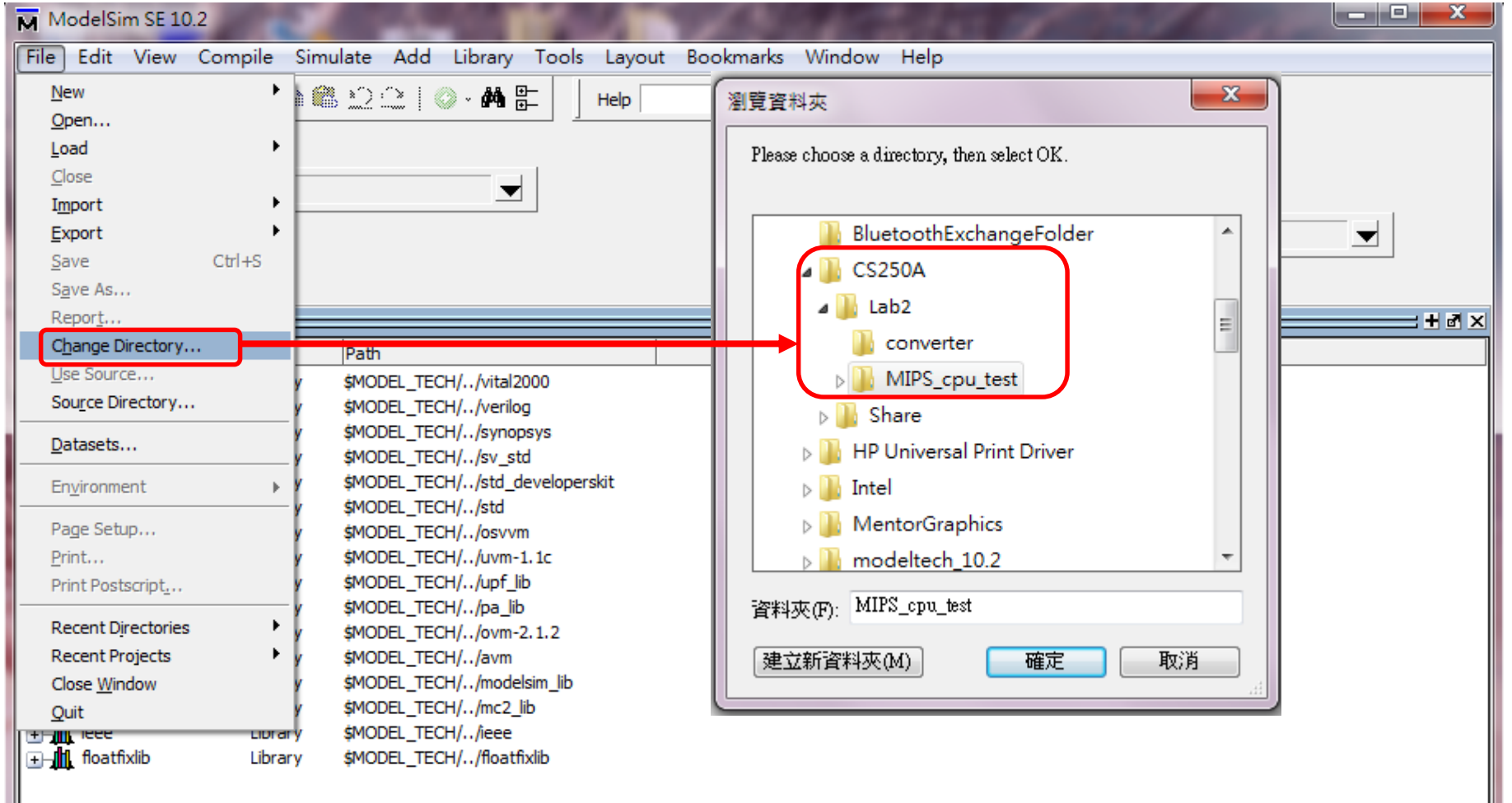


ModelSim 驗證教學

Step.1: change to LAB1 file location

打開modelsim後,在File下選擇change directory到你放\...\MIPS_cpu_test的地方

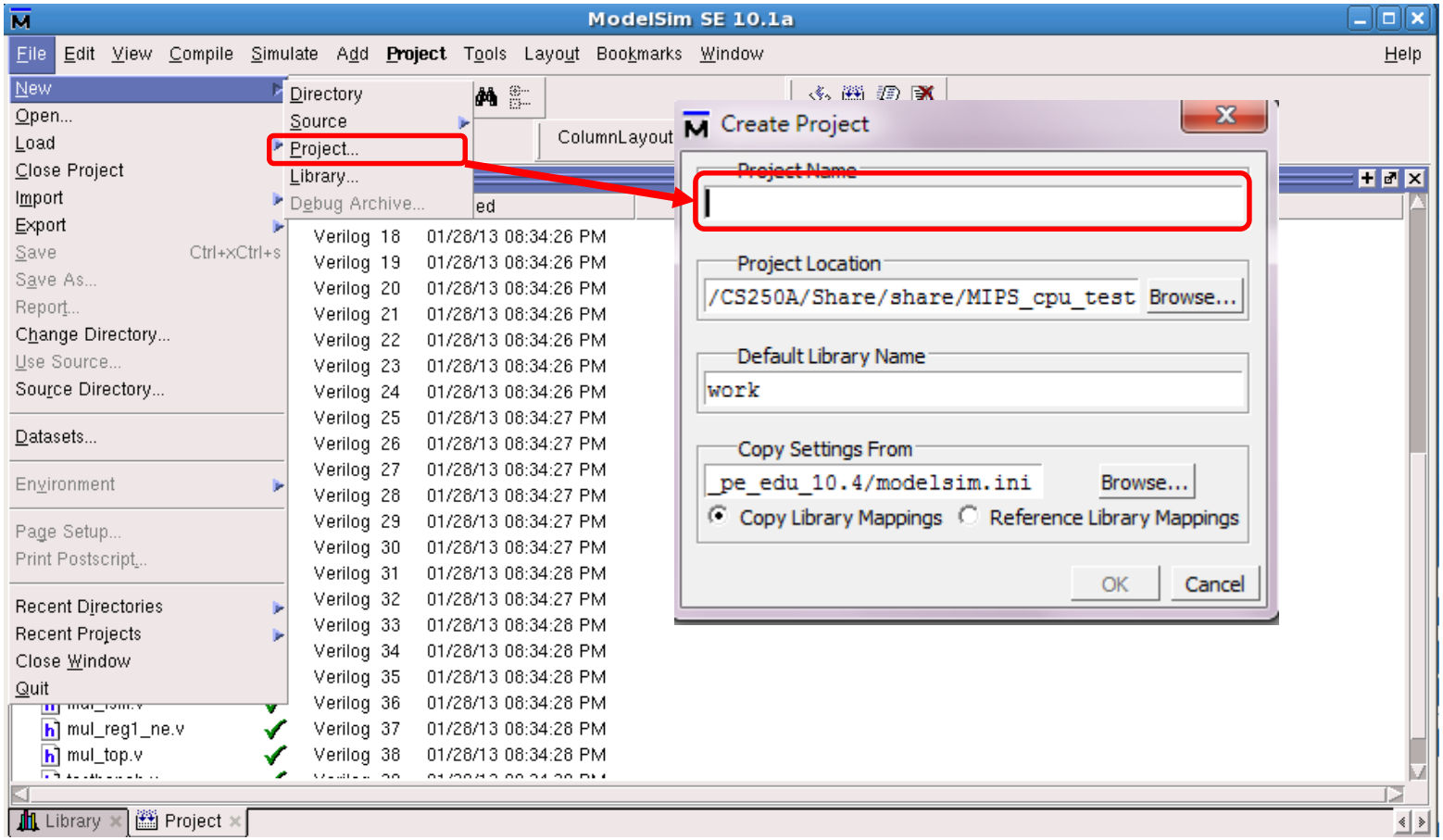
注意:路徑不可以有中文



ModelSim 驗證教學

Step.2: new project

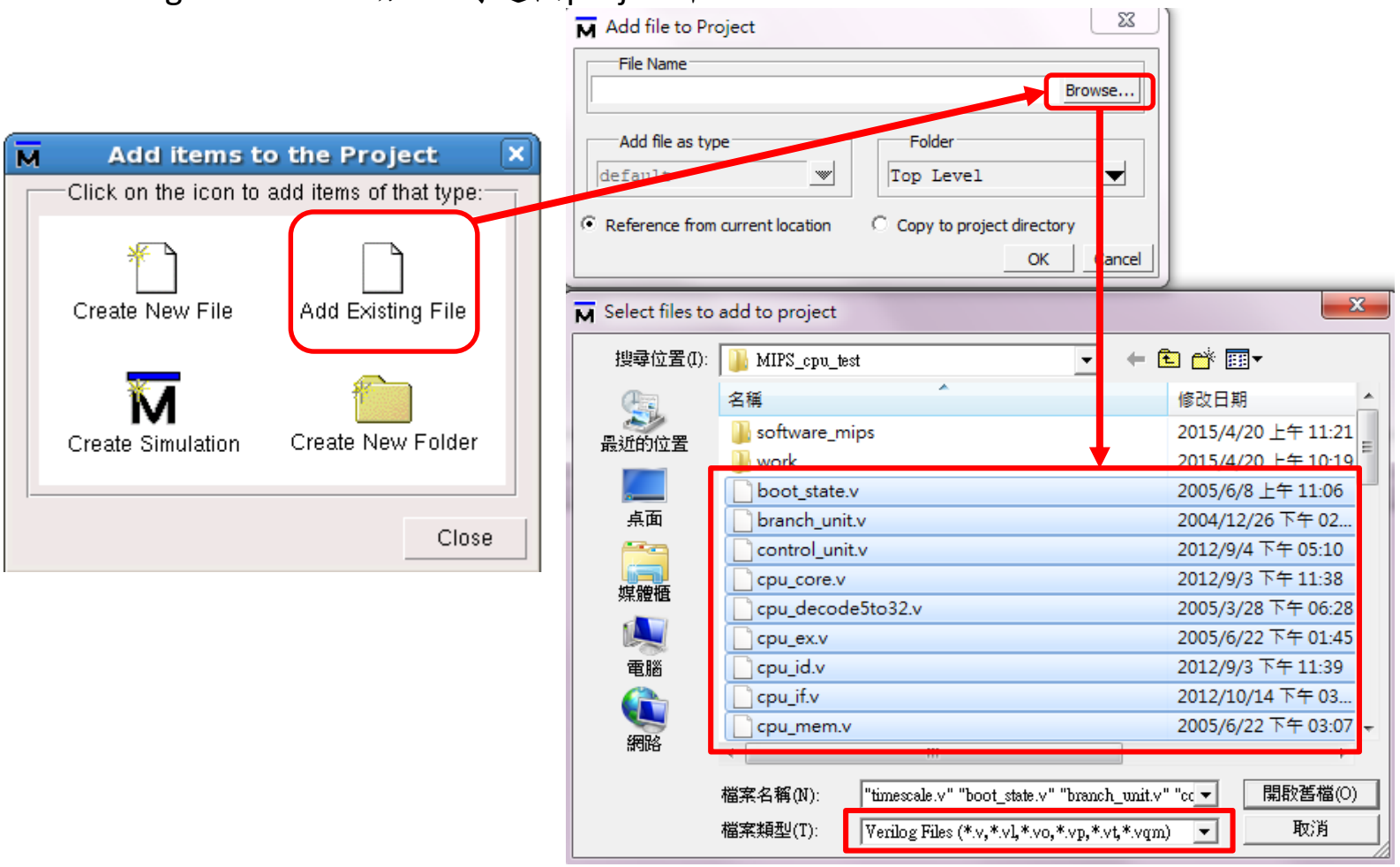
接著new一個project(名稱自訂)



ModelSim 驗證教學

Step.3: add source code

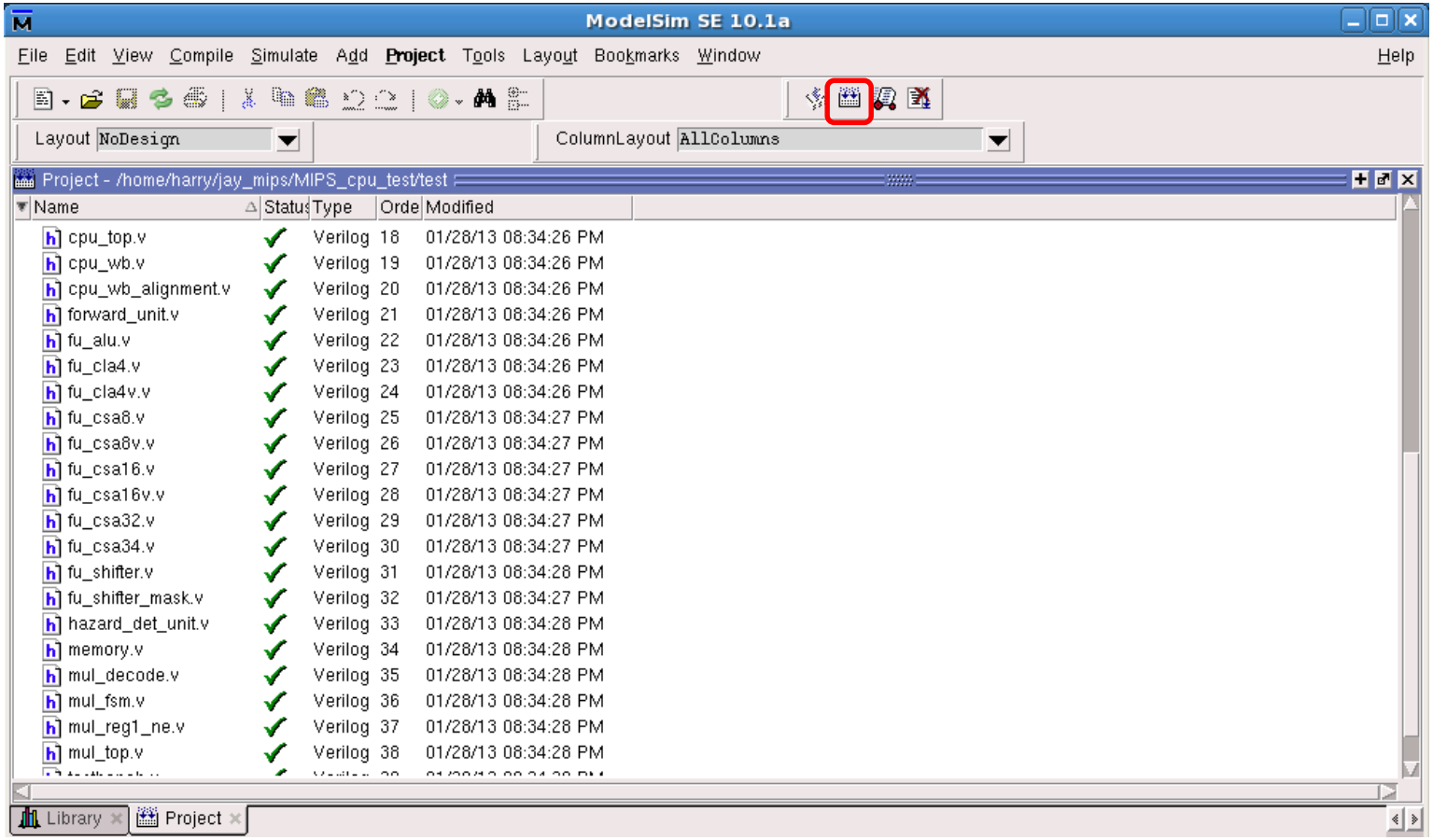
新增完project後,會跳出一個視窗(如左圖),點選Add Existing File將所有的Verilog source code加入到這個project中



ModelSim 驗證教學

Step.4: compile source code

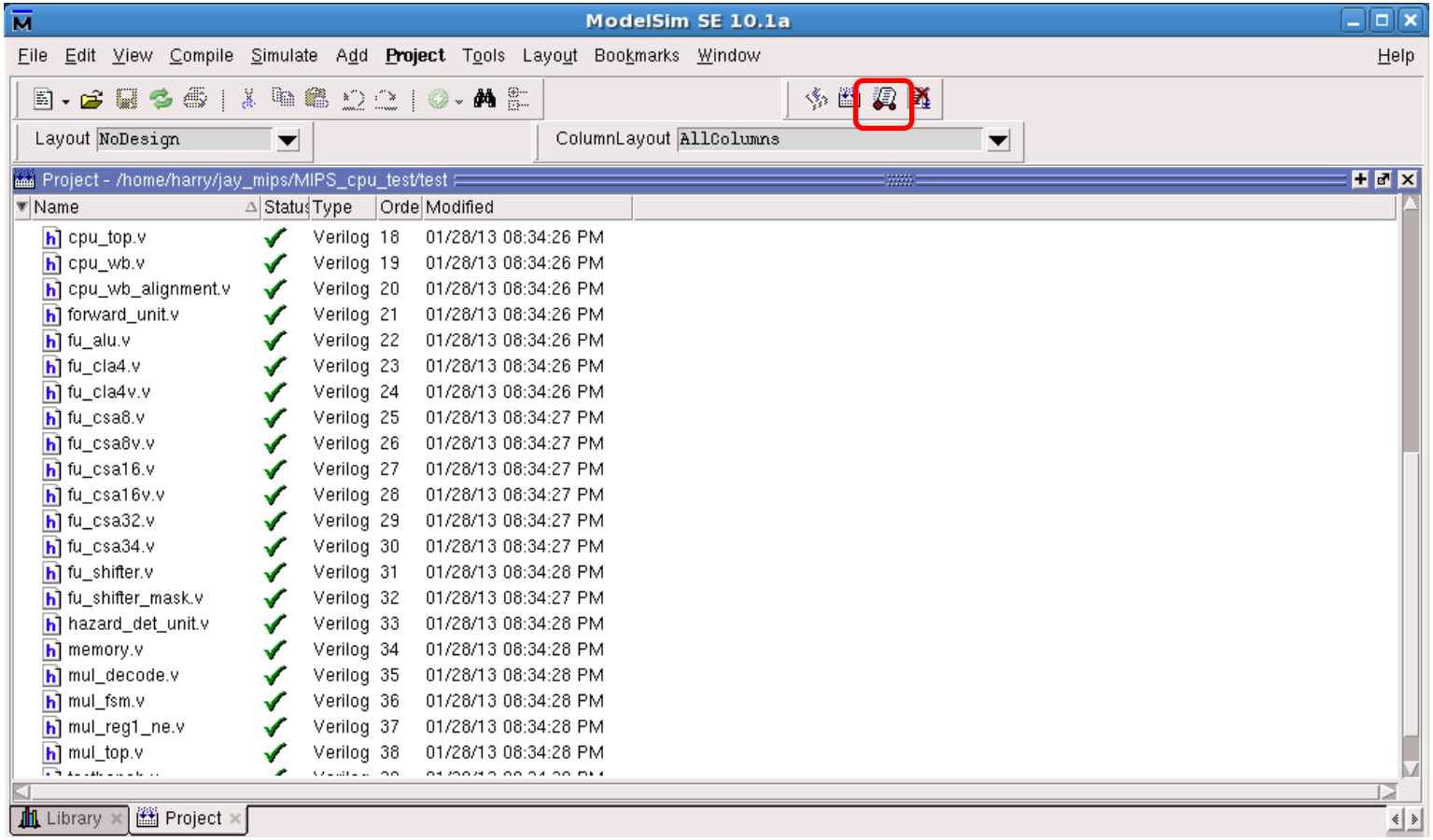
點擊Compile All



ModelSim 驗證教學

Step.5: simulate source code

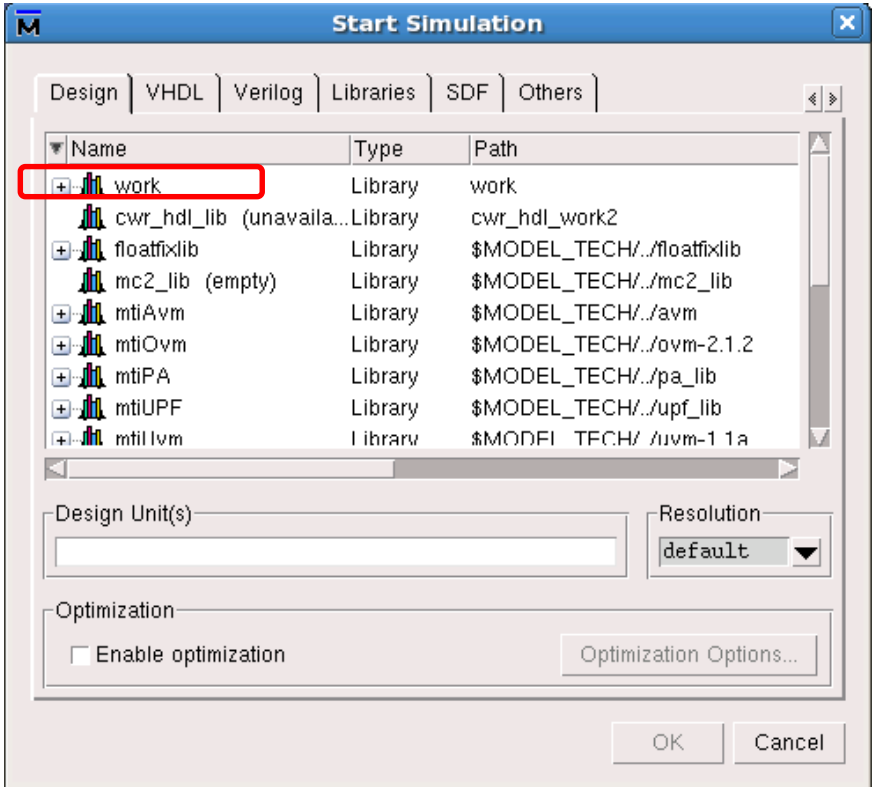
Compile成功後點擊simulate

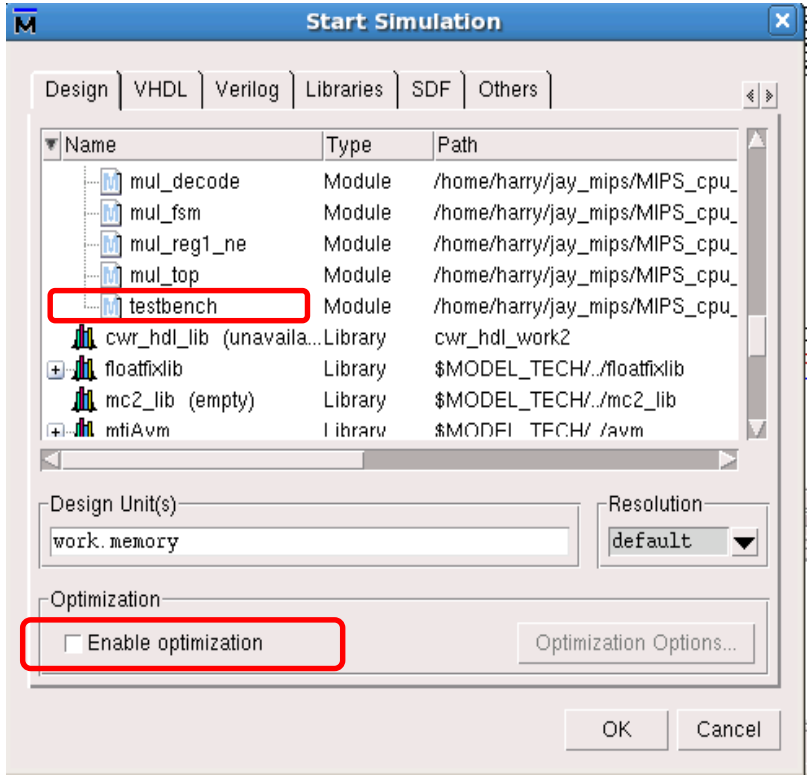


ModelSim 驗證教學

Step.6: choose testbench and enable optimization

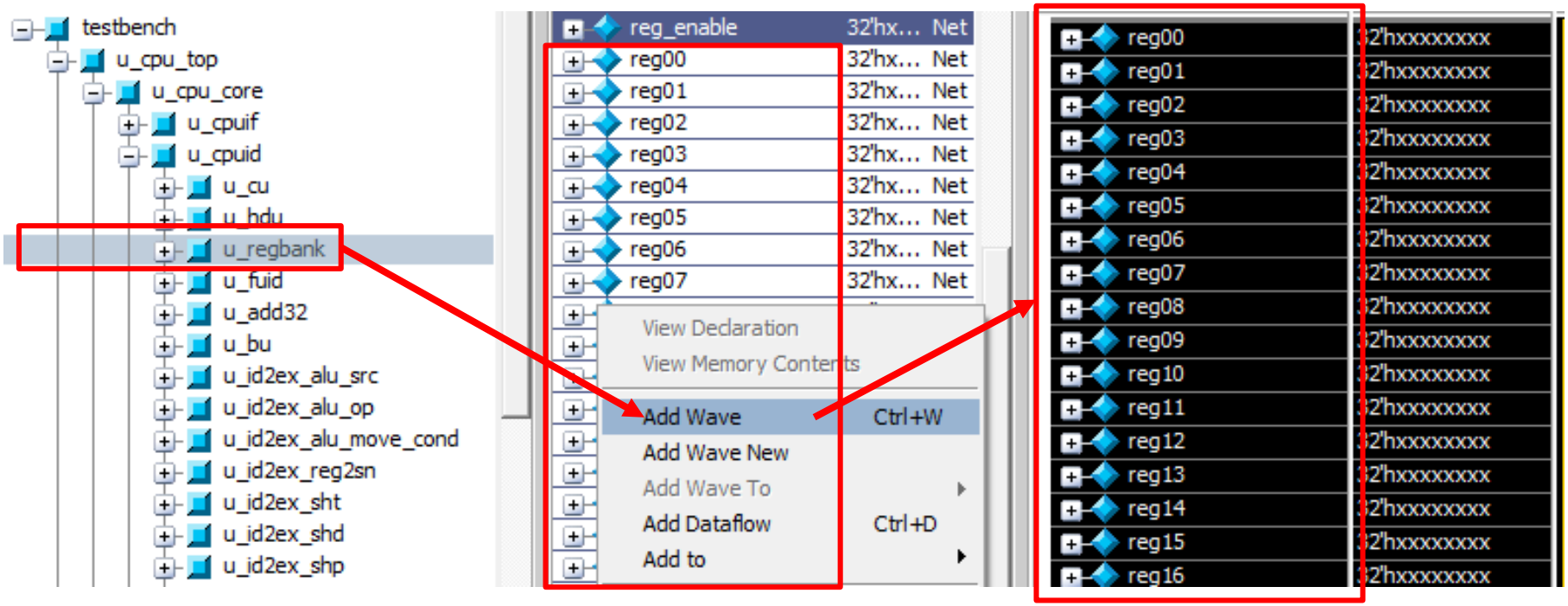
打開work後選testbench並取消最佳化





Add Wave of Registers

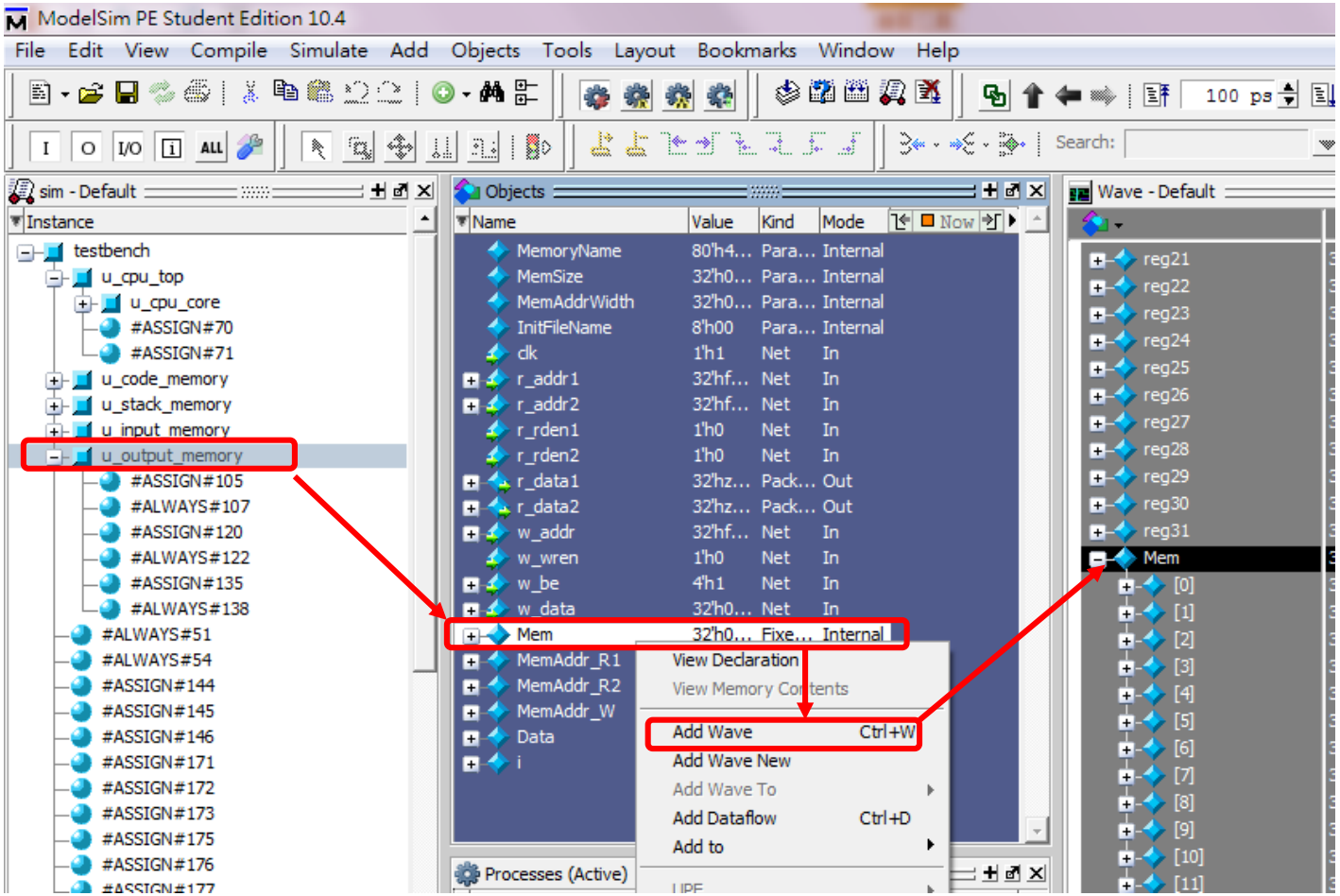
將你要看的訊號線按右鍵add wave,在這裡我們選擇所有的register
(u_cpu_top→u_cpu_core → u_cpuid → u_regbank)



Add Wave of Memory

在這邊要新增與記憶體相關的wave

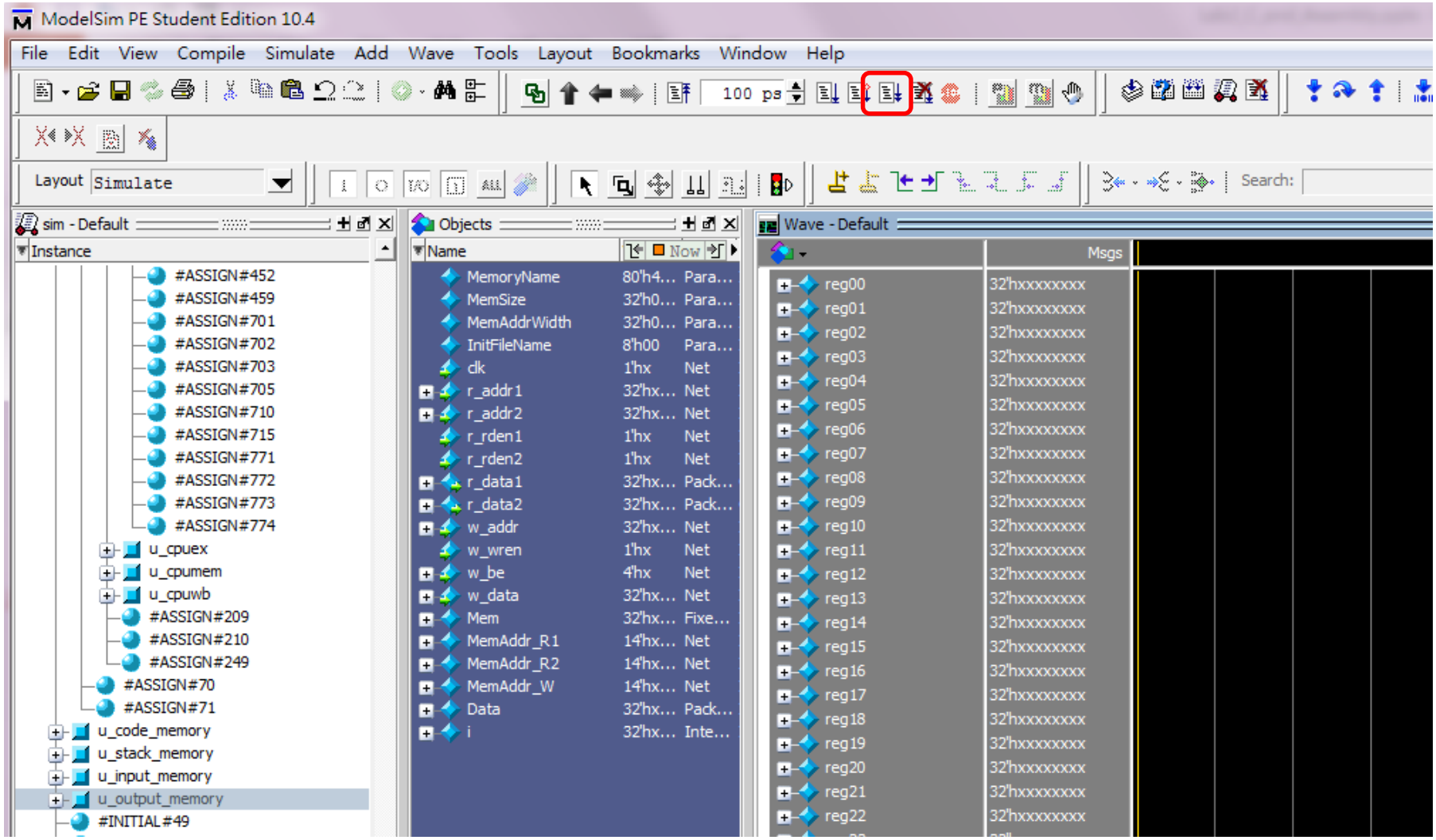
（u_output_memory→ Mem）



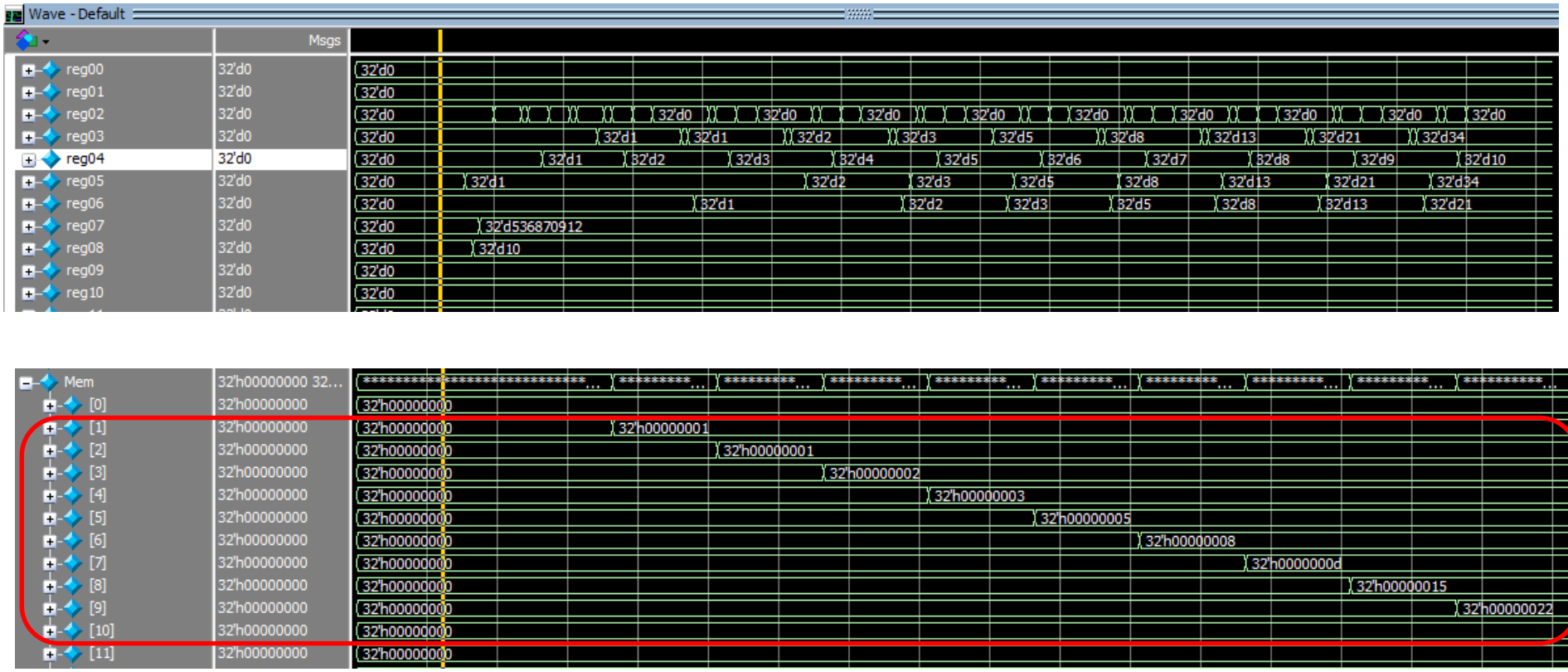
ModelSim 驗證教學

Step.7: run simulation

點擊Run-all



ModelSim – 波形結果



Project Assignment

實作(一) – C/C++ struct

```
1   #include <stdio.h>
2
3   struct student{
4       int mathGrade;
5       int csGrade;
6       int englishGrade;
7   } ;
8   main(){
9
10  volatile struct student* A =(struct student*) 0x20000000;
11  volatile struct student* B =(struct student*) 0x20000020;
12  struct student s1 = {60,70,70};
13  struct student s2 = {70,50,80};
14
15  *A = s1;
16  *B = s2;
17  }
```

實作(一) – assembly code

```
.LC0:
    .word   60
    .word   70
    .word   70
    .align  2
.LC1:
    .word   70
    .word   50
    .word   80
    .text
    .align  2
    .globl  main
    .set    nomips16
    .ent    main
```

```
main:
    .frame  $sp,32,$31      # vars= 32, regs= 0/0, args= 0, gp= 0
    .mask   0x00000000,0
    .fmask  0x00000000,0
    .set    noreorder
    .set    nomacro

    addiu   $sp,$sp,-32
    li  $4,536870912    # 0x20000000
    ori $6,$4,0x20
    lui $2,%hi(.LC0)
    lw  $9,%lo(.LC0)($2)
    addiu   $2,$2,%lo(.LC0)
    lw  $7,4($2)
    lw  $8,8($2)
    lui $2,%hi(.LC1)
    lw  $5,%lo(.LC1)($2)
    addiu   $2,$2,%lo(.LC1)
    lw  $3,4($2)
    lw  $2,8($2)
    sw  $5,16($sp)
    sw  $3,20($sp)
    sw  $2,24($sp)
    sw  $9,0($4)
    sw  $7,4($4)
    sw  $8,8($4)
    lw  $3,20($sp)
    lw  $4,24($sp)
    lw  $2,16($sp)
    sw  $2,0($6)
    sw  $3,4($6)
    sw  $4,8($6)
    j   $31
    addiu   $sp,$sp,32
```

實作(二) – function call

```
1   #include<stdio.h>
2
3   int sum(int,int,int);
4   int main()
5   {
6       int a=44,b=87,c=2;
7       volatile int* n = (int*) 0x20000000;
8       *n=sum(a,b,c);
9       return 0;
10  }
11
12  int sum(int a,int b,int c)
13  {
14      int n;
15      n=a+b+c;
16      return n;
17  }
```

實作(二) – assembly code

```
main:
    .frame  $sp,24,$31      # vars= 0, regs= 1/0, args= 16, gp= 0
    .mask   0x80000000,-8
    .fmask  0x00000000,0
    .set    noreorder
    .set    nomacro

    addiu   $sp,$sp,-24
    sw  $31,16($sp)
    li  $4,44           # 0x2c
    li  $5,87           # 0x57
    jal sum
    li  $6,2            # 0x2

    li  $3,536870912            # 0x20000000
    sw  $2,0($3)
    move    $2,$0
    lw  $31,16($sp)
    j   $31
    addiu   $sp,$sp,24

    .set    macro
    .set    reorder
    .end    main
    .size   main, .-main
    .align  2
    .globl  sum
    .set    nomips16
    .ent    sum
sum:
    .frame  $sp,0,$31       # vars= 0, regs= 0/0, args= 0, gp= 0
    .mask   0x00000000,0
    .fmask  0x00000000,0
    .set    noreorder
    .set    nomacro

    addu    $2,$4,$5
    j   $31
    addu    $2,$2,$6
```

Project Assignment

- 請同學完成上述兩個實作，將C code轉為Assembly code，最後完成在ModelSim的驗證，將結果寫成一份報告上傳至Portal
  - 檔案格式: s+你的學號_hw2.docx
    (ex: s1234567_hw2.docx)

- 報告須包含以下幾個部分
  - 編譯後的Assembly code
  - 實驗結果
    - 波形圖
    - 根據你得到的波形圖，解釋Assembly code與C code之間的關係
  - 實驗心得

Bonus – 進階題

請分析Lab2底下bonus.s這份assembly code

1. 寫出每次迴圈執行後memory的變化(共四次)
2. 簡單描述此code意義為何

```
main:
        .frame  $sp,0,$31
        .mask   0x00000000,0
        .fmask  0x00000000,0
        .set    noreorder
        .set    nomacro

        li      $8,536870912
        li      $9,5
        li      $2,10
        sw      $2,0($8)
        li      $2,92
        sw      $2,4($8)
        li      $2,55
        sw      $2,8($8)
        li      $2,1
        sw      $2,12($8)
        li      $2,46
        sw      $2,16($8)
        move    $7,$0
        addiu   $10,$9,-1
.L10:
        subu    $2,$9,$7
        addiu   $2,$2,-1
        blez    $2,.L14
        move    $5,$0

        subu    $2,$9,$7
        addiu   $6,$2,-1
        sll     $2,$5,2
.L15:
        addu    $4,$2,$8
        lw      $2,4($4)
        lw      $3,0($4)
        slt     $2,$2,$3
        beq     $2,$0,.L7
        nop

        lw      $3,0($4)
        lw      $2,4($4)
        sw      $2,0($4)
        sw      $3,4($4)
.L7:
        addiu   $5,$5,1
        slt     $2,$5,$6
        bne     $2,$0,.L15
        sll     $2,$5,2

.L14:
        addiu   $7,$7,1
        slt     $2,$7,$10
        bne     $2,$0,.L10
        move    $2,$0

        j       $31
        nop
```