**計算機組織**

**Lab 0**

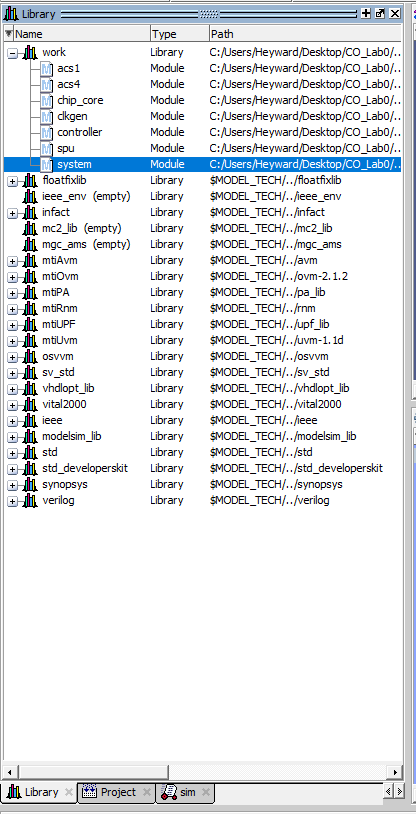
**系級：土木4B**

**學號：0511330**

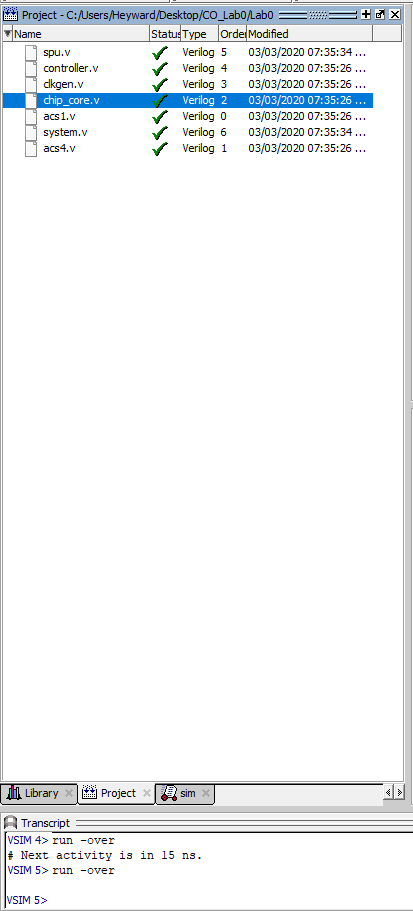
**姓名：劉紘華**

**Part1**

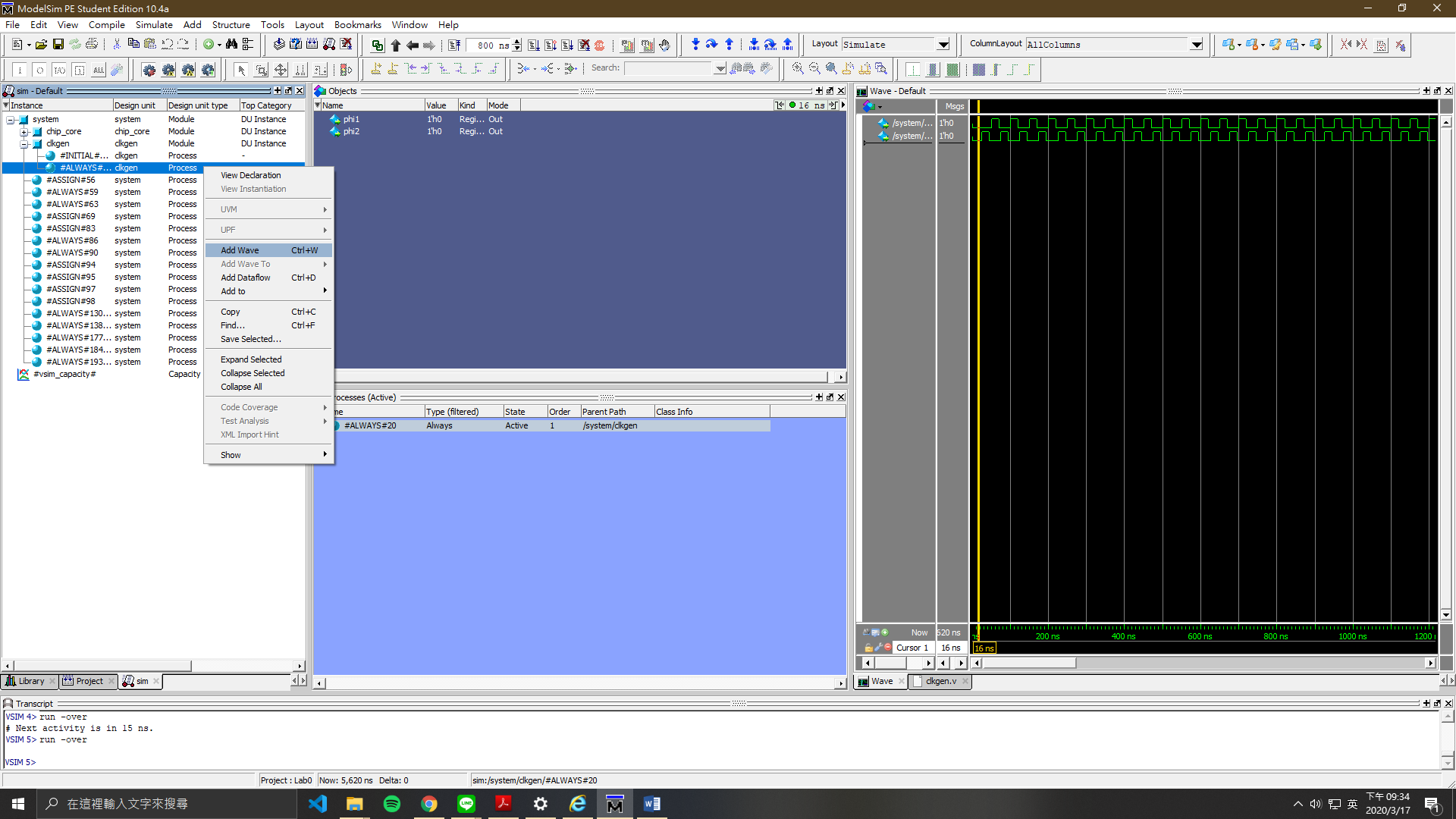
**1.Add Files into work directory.**



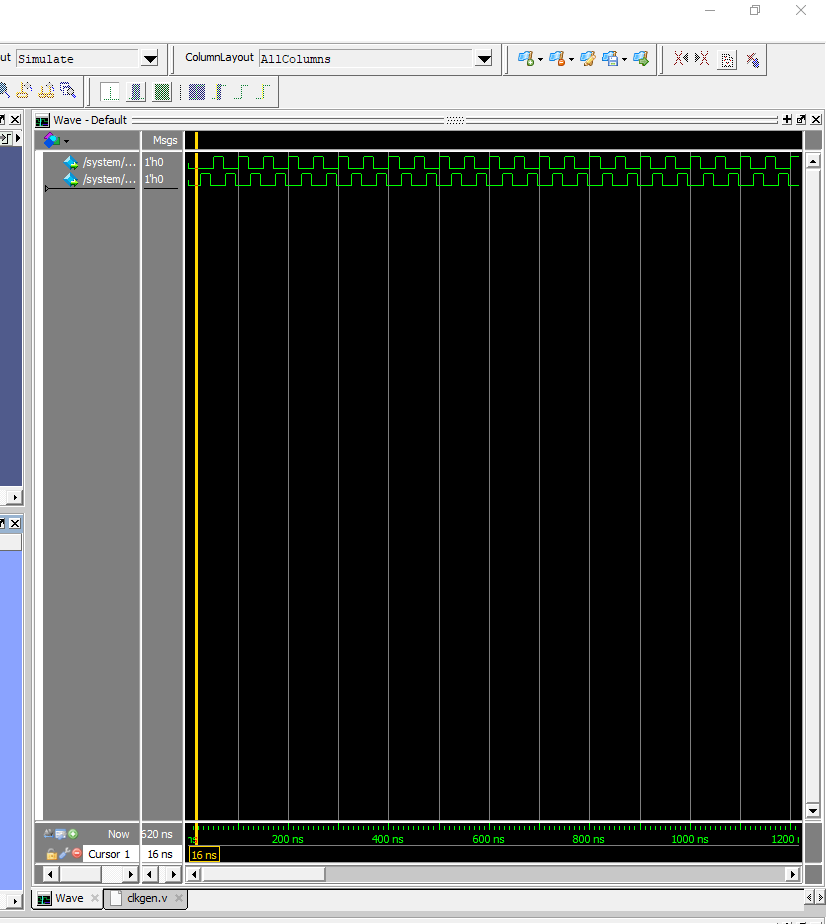
**2.Compile All Project Files.**



**3.Add Wave in Simulation windows.**



**4.Press F9 to Run the simulation and get a Wave diagram.**



**Part2**

**C Code:**

int multiplication(int a, int b){

    return a\*b;

}

int main(){

    int x1, x2, x3;

    x1 = 2;

    x2 = 3;

    x3 = multiplication(x1,x2);

}

**RISC-V 64 Code:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(1) multiplication(int, int):  # @multiplication(int, int)

(2) addi  sp, sp, -32 // Add immediate value -32, sp = sp + (-32)

(3)  sd    ra, 24(sp) // Store doubleword, Memory [sp + 24] = ra

(4)  sd    s0, 16(sp) // Store doubleword, Memory [sp +16] = s0

(2)把stack pointer往下移4個doubleword用來儲存，

(3)和(4)分別把return address (ra)和stack pointer (s0)先存起來，避免執行multiplication( )時，覆蓋掉main( )原本存的的ra和s0。

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(5)  addi  s0, sp, 32 // Add immediate value 32, s0 = sp + 32

(6)  add   a2, zero, a1 // Add zero, a2 = 0 + a1

(7)  add   a3, zero, a0 // Add zero, a3 = 0 + a0

(8)  sw    a0, -20(s0) // Store word, Memory[s0 - 20] = a0

(9)  sw    a1, -24(s0) // Store word, Memory[s0 – 24] = a1

(5) 利用frame pointer (s0)指向一開始的sp當作end\_ptr使用，類似陣列中存在兩個指標start\_ptr和end\_ptr。

(6)和(7)先把function arguments先傳遞給其他Reg.，a1給a2、a0給a3。

(8)和(9)再把a0 & a1存進Stack中，因為a0和a1用來存放**function arguments**和 **return value**，先存放是為了避免被後面產生的**return value**覆蓋掉。

註：a0 = x10, a1 = x11

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(10) lw    a0, -20(s0) // Load word, a0 = Memory[s0 - 20]

(11) lw    a1, -24(s0) // Load word, a1 = Memory[s0 – 24]

(12) mulw  a0, a0, a1 // a0[0:63] = Sign\_Extend ( a0[0:31] \* a1[0:31] )

// RISCv 64才有的指令，將a0的0~32位 乘以 a1的0~32位，

// 再把算出來的結果Sign\_Extend成64-bit，存進a0。

(10)和(11)從Stack中把傳進來a0, a1存取回來（a0 = a = 2, a1 = b = 3）

(12)則是把a0\*a1的結果存進a0當成**return value**（return a\*b in C code）

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(13) ld    s0, 16(sp) // Load doubleword, a0 = Memory[sp + 16]

(14) ld    ra, 24(sp) // Load doubleword, ra = Memory[sp + 24]

(15) addi  sp, sp, 32 // Add immediate value, sp = sp + 32

(16) ret

(13)和(14)將存在Stack中的s0和ra存取回來

(15)把stack pointer指回原來的sp，release(2)占用的4個doubleword

(16) return回main

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(17) main:  # @main

(18) addi  sp, sp, -32 // Add immediate value, sp = sp - 32

(19) sd    ra, 24(sp) // Store doubleword, Memory[sp + 24] = ra

(20) sd    s0, 16(sp) // Store doubleword, Memory[sp + 16] = s0

(18)~(20)意同multiplication函式裡面的(2)~(5)，把ra和s0存進Stack中。

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(21) addi  s0, sp, 32 // Add immediate value, s0 = sp + 32

(22) addi  a0, zero, 2 // Add immediate value, a0 = 0 + 2

(23) sw    a0, -20(s0) // Store word, Memory[s0 – 20] = a0

(24) addi  a0, zero, 3 // Add immediate value, a0 = 0 + 3

(25) sw    a0, -24(s0) // Store word, Memory[s0 – 24] = a0

(21)意同(5)，建立end\_ptr。

(22)~(24)先把值存進Reg.中，再把Reg.中的值存進Stack中。

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(26) lw    a0, -20(s0) // Load word, a0 = Memory[s0 -20]

(27) lw    a1, -24(s0) // Load word, a1 = Memory[s0 -24]

(28) call  multiplication(int, int)

(26)、(27) 把Stack中的值存進a0, a1，透過(28)把參數a0, a1傳遞到function中。

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(29) sw    a0, -28(s0) // Store word, Memory[s0 -28] = a0

(30) mv    a0, zero // Move from zero to a0, a0 = 0

// mv is similar to addi a0, zero, 0 ( a0 = zero + 0 )

(29) 把function的return value存進Stack中，(30)再把a0初始化成0。

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(31) ld    s0, 16(sp) // Load doubleword, s0 = Memory[sp + 16]

(32) ld    ra, 24(sp) // Load doubleword, ra = Memory[sp + 24]

(33) addi  sp, sp, 32 // Add immediate value, sp = sp + 32

(34) ret

(31)、(32) 把main一開始存的s0 & ra從Stack中存回來，(33)將sp指回top of Stack，釋放main一開始占用的4個doubleword。

(34) int main()的結尾（return 0 in C code）。