Computer Organization lab 0

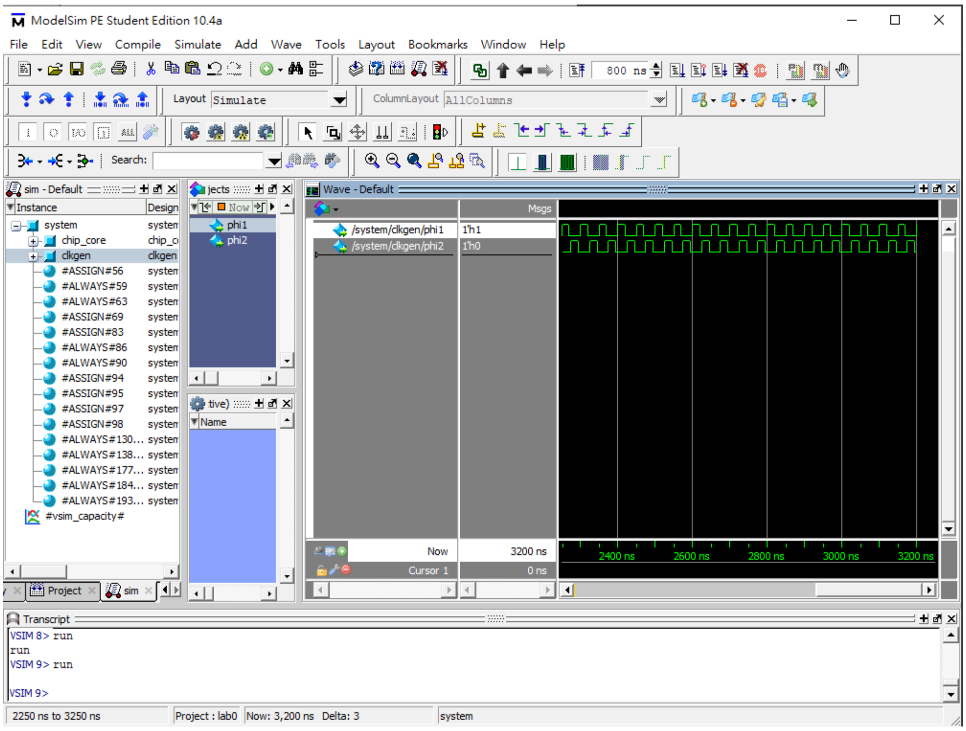
Informatoin

- Author : 0712238, Yan-Tong Lin
- Due date : 3/20
- Online version of this hackmd file (https://hackmd.io/91w4a3v2TtqfwihAk-1ccQ)

Part 1

- Goal
    - The purpose of this Lab is to familiar with ModelSim Simulator Tool.
- The result is as screen shot
- Extra Note
    - avoid using @nctu.edu.tw to prevent not being able to receive email
    - need to fill the form in the end to get liscense
    - the liscense should be put in the modelism folder

Run Result Screenshot



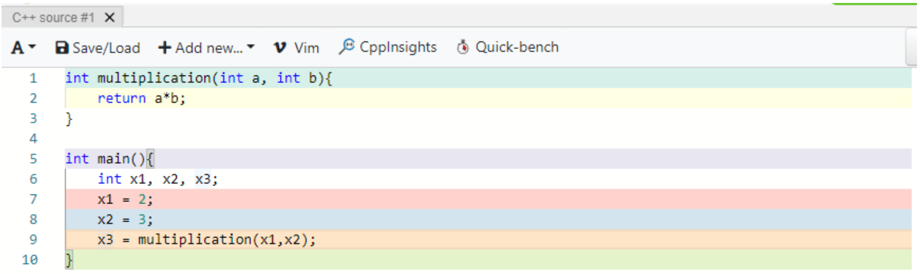
```
VSIM 8> run
run
VSIM 9> run

VSIM 9>
```

2250 ns to 3250 ns | Project : lab0 | Now: 3,200 ns Delta: 3 | system

Part 2

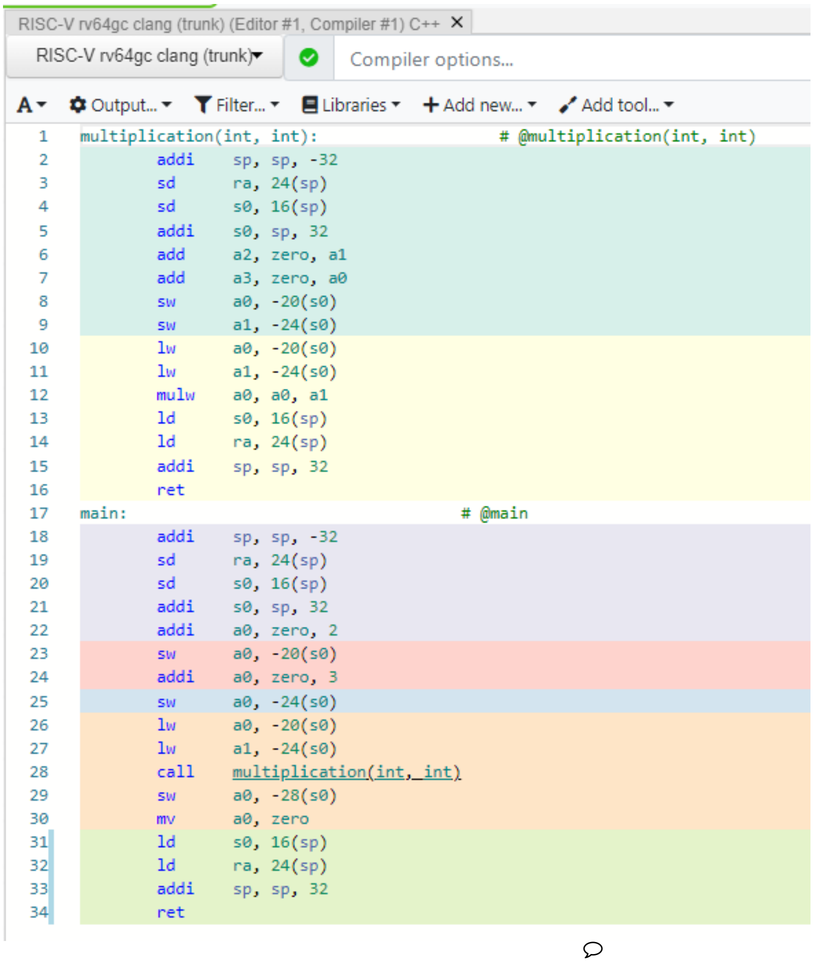
- Objective : try to explain the instructions compiled from C

- Original Code



```
int multiplication(int a, int b){
    return a*b;
}

int main(){
    int x1, x2, x3;
    x1 = 2;
    x2 = 3;
    x3 = multiplication(x1,x2);
}
```

- The Compile Result


RISC-V rv64gc clang (trunk) (Editor #1, Compiler #1) C++ ✕
RISC-V rv64gc clang (trunk)
Compiler options...
A
Output...
Filter...
Libraries
Add new...
Add tool...

```
multiplication(int, int):                          # @multiplication(int, int)
        addi    sp, sp, -32
        sd      ra, 24(sp)
        sd      s0, 16(sp)
        addi    s0, sp, 32
        add     a2, zero, a1
        add     a3, zero, a0
        sw      a0, -20(s0)
        sw      a1, -24(s0)
        lw      a0, -20(s0)
        lw      a1, -24(s0)
        mulw    a0, a0, a1
        ld      s0, 16(sp)
        ld      ra, 24(sp)
        addi    sp, sp, 32
        ret
main:                                   # @main
        addi    sp, sp, -32
        sd      ra, 24(sp)
        sd      s0, 16(sp)
        addi    s0, sp, 32
        addi    a0, zero, 2
        sw      a0, -20(s0)
        addi    a0, zero, 3
        sw      a0, -24(s0)
        lw      a0, -20(s0)
        lw      a1, -24(s0)
        call    multiplication(int, int)
        sw      a0, -28(s0)
        mv      a0, zero
        ld      s0, 16(sp)
        ld      ra, 24(sp)
        addi    sp, sp, 32
        ret
```

- word(example: 32bit int) = 4 byte

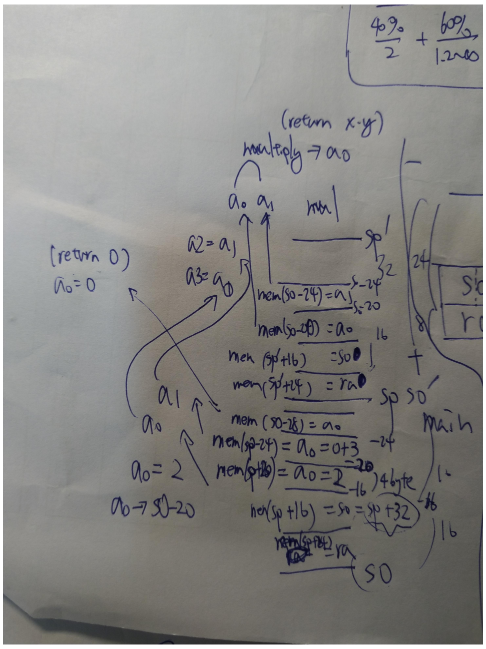
- double word(example: address) = 8 byte

Code Translation + Meaning Explanation

```
function call base code:

addi    sp, sp, -32 // sp = sp-32, stack pointer = stack pointer -32 , reserve m
sd      ra, 24(sp)  // mem[sp+24] = ra, store return address to sp[24]
sd      s0, 16(sp)  // mem[sp+16] = s0, store s0 to sp[16], remember the memory
addi    s0, sp, 32  // s0 = sp+32, s0 = the origin stack pointer position

...

ld      s0, 16(sp) // s0 = mem[sp+16], load the last stack pointer position
ld      ra, 24(sp) // ra = mem[sp+24],load return address
addi    sp, sp, 32 // sp = sp+32, stack pointer go down, give back memory
ret                // return
```

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

    addi    sp, sp, -32
    sd      ra, 24(sp)
    sd      s0, 16(sp)
    addi    s0, sp, 32
    add     a2, zero, a1 //a2 = a1, pass param a1 to a2
    add     a3, zero, a0 //a3 = a0, pass param a0 to a3
    sw      a0, -20(s0)  //mem[s0-20] = a0, param is saved
    sw      a1, -24(s0)  //mem[s0-24] = a1
    lw      a0, -20(s0)  //a0 = mem[s0-20], saved param is used
    lw      a1, -24(s0)  //a1 = mem[s0-24]
    mulw    a0, a0, a1   //a0 = a0*a1
    ld      s0, 16(sp)
    ld      ra, 24(sp)
    addi    sp, sp, 32
    ret

main:
    addi    sp, sp, -32
    sd      ra, 24(sp)
    sd      s0, 16(sp)
    addi    s0, sp, 32
    addi    a0, zero, 2 //a0 = 2, get the value
    sw      a0, -20(s0) //mem[s0-20] = a0, save value
    addi    a0, zero, 3 //a0 = 3
    sw      a0, -24(s0) //mem[s0-24] = a0
    lw      a0, -20(s0) //a0 = mem[s0-20], param to pass
    lw      a1, -24(s0) //a1 = mem[s0-24]
    call    multiplication(int, int) //call function multiplicatio with a0, a1 a
    sw      a0, -28(s0) // mem[s0-28] = a0, a0 is return value
    mv      a0, zero    //a0 = 0, main return 0 as exit code
    ld      s0, 16(sp)
    ld      ra, 24(sp)
    addi    sp, sp, 32
    ret                 //end of program
```

Some drawing about the process of understanding the code



40%/2 + 60%/1.2
(return x-y)
multiply → a0
a0 a1
mul
sp'
32
24
a2 = a1
a3 = a0
(return 0)
a0 = 0
mem(s0-24) = a1
s-24
s-20
mem(s0-20) = a0
16
mem (sp'+16) = s0
mem(sp'+24) = ra
sp s0'
a1
a0
mem (s0-28) = a0
main
mem(sp-24) = a0 = 0+3
-24
a0 = 2
mem(sp+16) = a0 = 2
-20
-16
) 4 byte
a0 → s0-20
mem(sp+16) = s0 = sp+32
16
mem(sp+24) = ra
s0
s
r
8