計算機組織

實驗二

結果報告

第十二組

組員:

|  |  |
| --- | --- |
| 學號 | 姓名 |
| E24099025 | 李宇洋 |
| E24099059 | 陳旭祺 |
| E24099033 | 蔡晏齊 |
| F74044046 | 蕭佑永 |

日期:2020/10/05

一、實驗內容

(1)實作一

1.題目:用組合語言描述下列C語言

*#include<stdio.h>*

*int sum(int,int,int);*

*int main(){*

*int a=44, b=87, c=2;*

*volatile int\* n = (int\*) 0x00000800;*

*\*n=sum(a,b,c); //函式sum的運算結果會回傳存到記憶體位置2048*

*return 0;*

*}*

*int sum(int a,int b,int c){*

*int n;*

*n=a+b+c;*

*return n;*

*}*

2.實現方式(組合語言)

main:

#### 剛進入main function ####

addi sp,sp,-32

#進入main function,空出一個stack的空間(sp作為main stack的底部)

sw ra,28(sp)

#將ra存起來(owner:caller -> 之後return 0要用)

sw s0,24(sp)

#將s0存起來(owner:callee -> callee要負責維持save register的值)

addi s0,sp,32

#拿s0作為frame pointer(s0作為main stack的頂部)

#### 宣告 int a ####

addi t1,zero,44 #

sw t1,-20(s0) # int a 被我放到(s0-20)這個記憶體位址

#### 宣告 int b,c ####

addi t2,zero,87

sw t2,-24(s0)

addi t3,zero,2

sw t3,-28(s0)

#### 宣告 int \*n ####

li s11,2048

#### 準備function arguments ####

add a1,t1,x0

add a2,t2,x0

add a3,t3,x0

#### 進入sum function (使用跳轉指令) ####

jal ra,sum

#### 將拿到的回傳值(放在a0)放進\*n ####

sw a0,0(s11)

#### 結束main function ####

lw ra,28(sp)

# 拿回ra(owner:caller -> 拿回自己保存的值)

lw s0,24(sp)

# 拿回s0(owner:callee -> callee要負責維持save register的值)

addi sp,sp,32 # 釋出main的stack

jalr zero,0(ra) # 跳出main(使用跳轉指令)

sum:

#### 剛進入sum function ####

addi sp,sp,-32

#進入main function,空出一個stack的空間(sp作為main stack的底部)

sw ra,28(sp)

#將ra存起來(owner:caller -> 之後return 0要用)

sw s0,24(sp)

#將s0存起來(owner:callee -> callee要負責維持save register的值)

addi s0,sp,32

#拿s0作為frame pointer(s0作為main stack的頂部)

#### 儲存function arguments ####

add s1,a1,x0

add s2,a2,x0

add s3,a3,x0

#### 跑 sum function (n = a+b+c), 並將回傳值n放到a0 ####

add s4,s1,s2

add a0,s3,s4

#### 結束sum function ####

lw ra,28(sp)

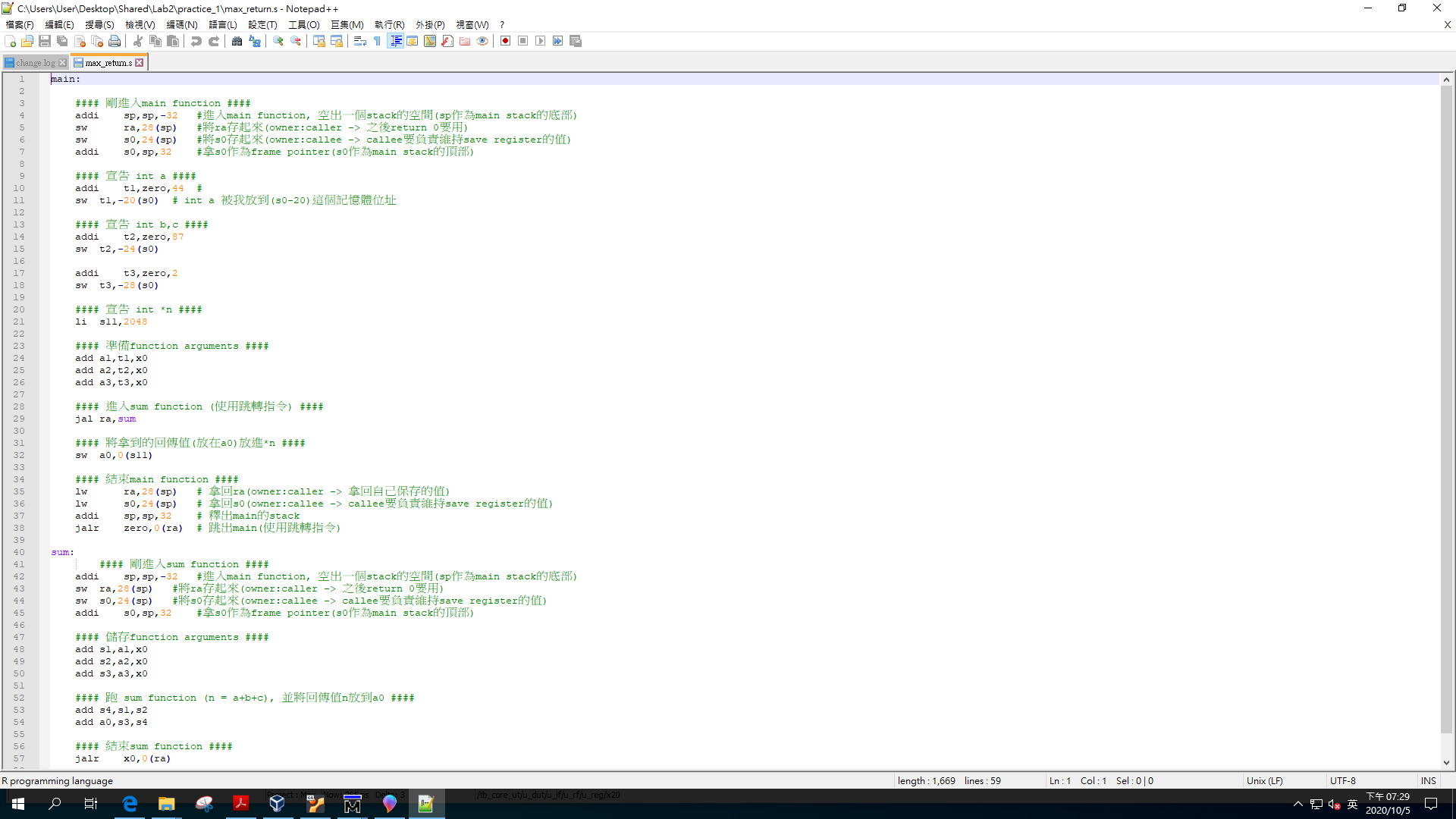
# 拿回ra(owner:caller -> 拿回自己保存的值)

lw s0,24(sp)

# 拿回s0(owner:callee -> callee要負責維持save register的值)

addi sp,sp,32 # 釋出sum的stack(在課堂上我們忘記打這段了)

jalr x0,0(ra) # 跳出sum(使用跳轉指令)



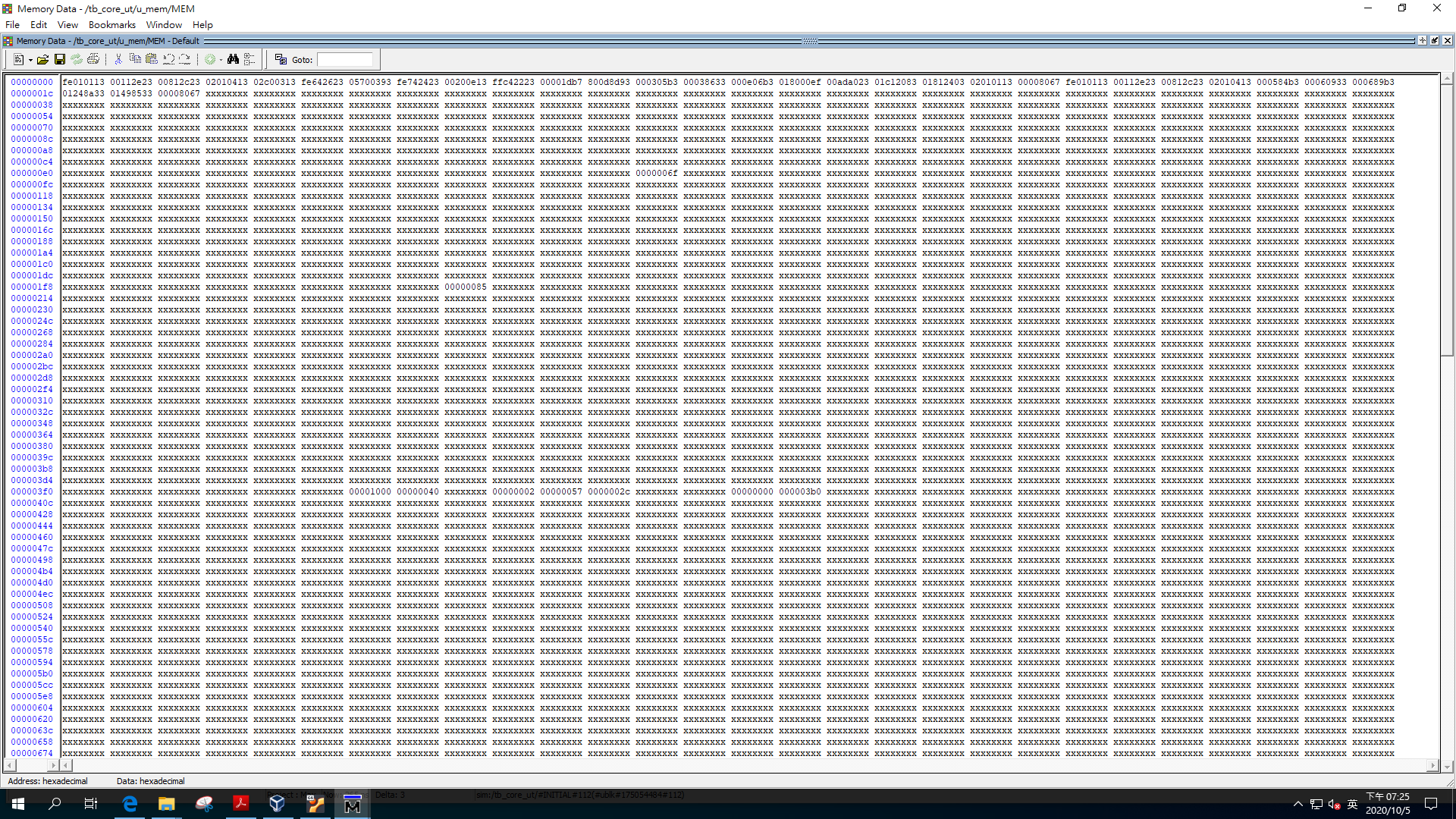
3.結果分析(modelsim)

Memory

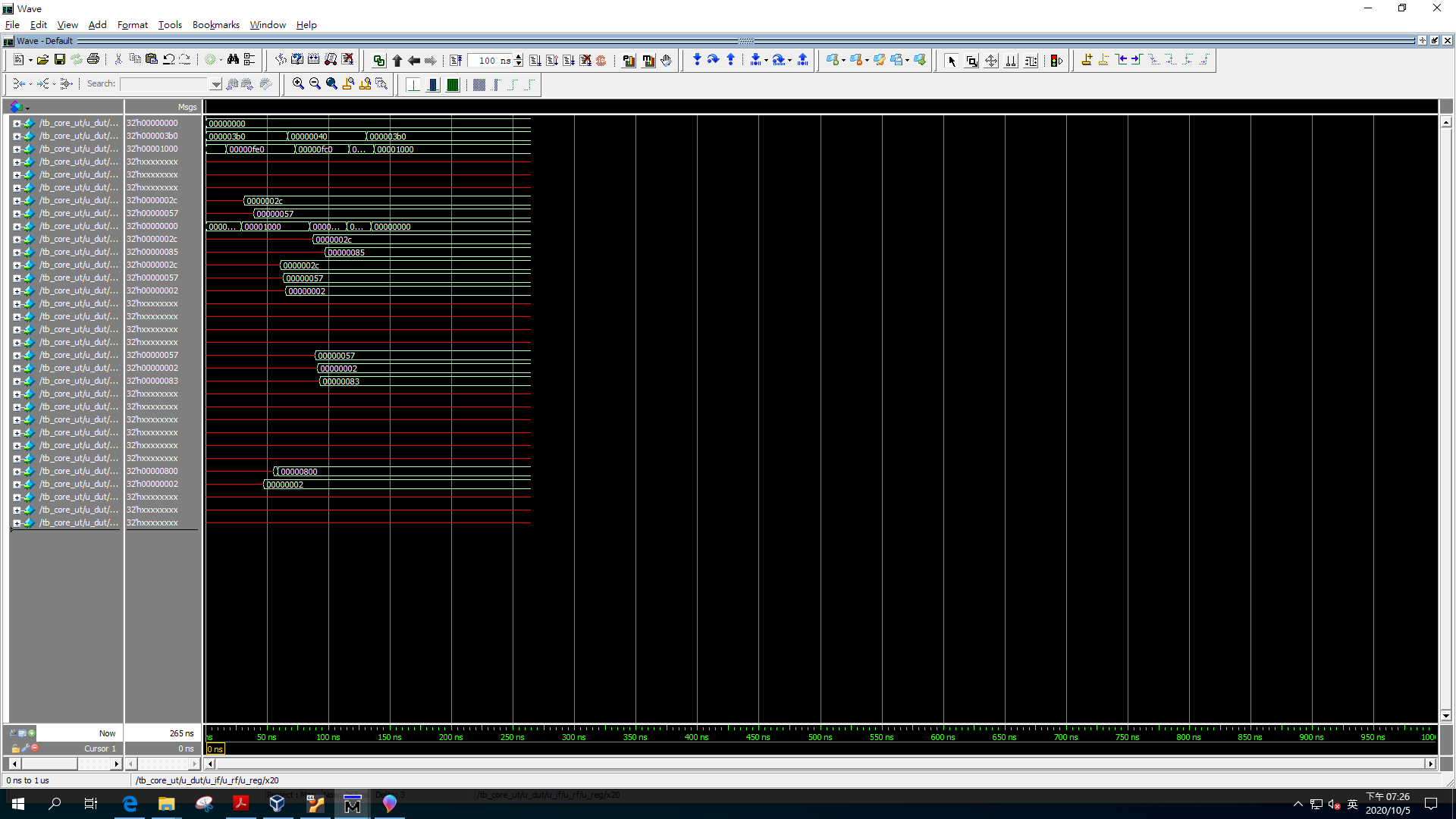
assembly code計算記憶體位址 是以byte(8bits)為單位，而Modelsim中 的memory是以word(32bits)為單位。而因此在判斷位址的時候要除以4

位址：0x00000200(512)

數值：133=27+22+20



Register



(2)實作二

1.題目：用組合語言描述下列存取記憶體的C語言

*#include <stdio.h>*

*struct student{*

*int mathGrade;*

*int csGrade;*

*int englishGrade;*

*};*

*int main(){*

*volatile struct student\* A =(struct student\*) 0x00000800;*

*volatile struct student\* B =(struct student\*) 0x00000820;*

*struct student s1 = {60,70,70};*

*struct student s2 = {70,50,80};*

*\*A = s1;*

*\*B = s2;*

*//將宣告的兩個struct分別放 在記憶體位址2048和2080*

*return 0;*

*}*

2.實現方式(組合語言)

main:

#### 剛進入main function(注意stack的大小)####

addi sp,sp,-40

#進入main function,空出一個stack的空間(sp作為main stack的底部)

sw ra,16(sp)

#將ra存起來(owner:caller -> 之後return 0要用)

addi s0,sp,40

#拿s0作為frame pointer(s0作為main stack的頂部)

#### 宣告struct student\* A ####

addi a5,zero,1024

addi a5,a5,1024

sw a5,-20(s0)

#### 宣告struct student\* B ####

addi a6,zero,1040

addi a6,a6,1040

sw a6,-4(s0)

#### 宣告struct student s1 ####

addi a5,zero,60

sw a5,-36(s0)

addi a5,zero,70

sw a5,-32(s0)

addi a5,zero,70

sw a5,-28(s0)

#### 宣告struct student s2 ####

addi a6,zero,70

sw a6,-16(s0)

addi a6,zero,50

sw a6,-12(s0)

addi a6,zero,80

sw a6,-8(s0)

#### \*A = s1 ####

lw a5,-20(s0) #拿到\*A

lw a4,-36(s0) #將s1的值一個一個丟進去

sw a4,0(a5)

lw a4,-32(s0)

sw a4,4(a5)

lw a4,-28(s0)

sw a4,8(a5)

#### \*B = s2 ####

lw a6,-4(s0) #拿到\*B

lw a4,-16(s0) #將s2的值一個一個丟進去

sw a4,0(a6)

lw a4,-12(s0)

sw a4,4(a6)

lw a4,-8(s0)

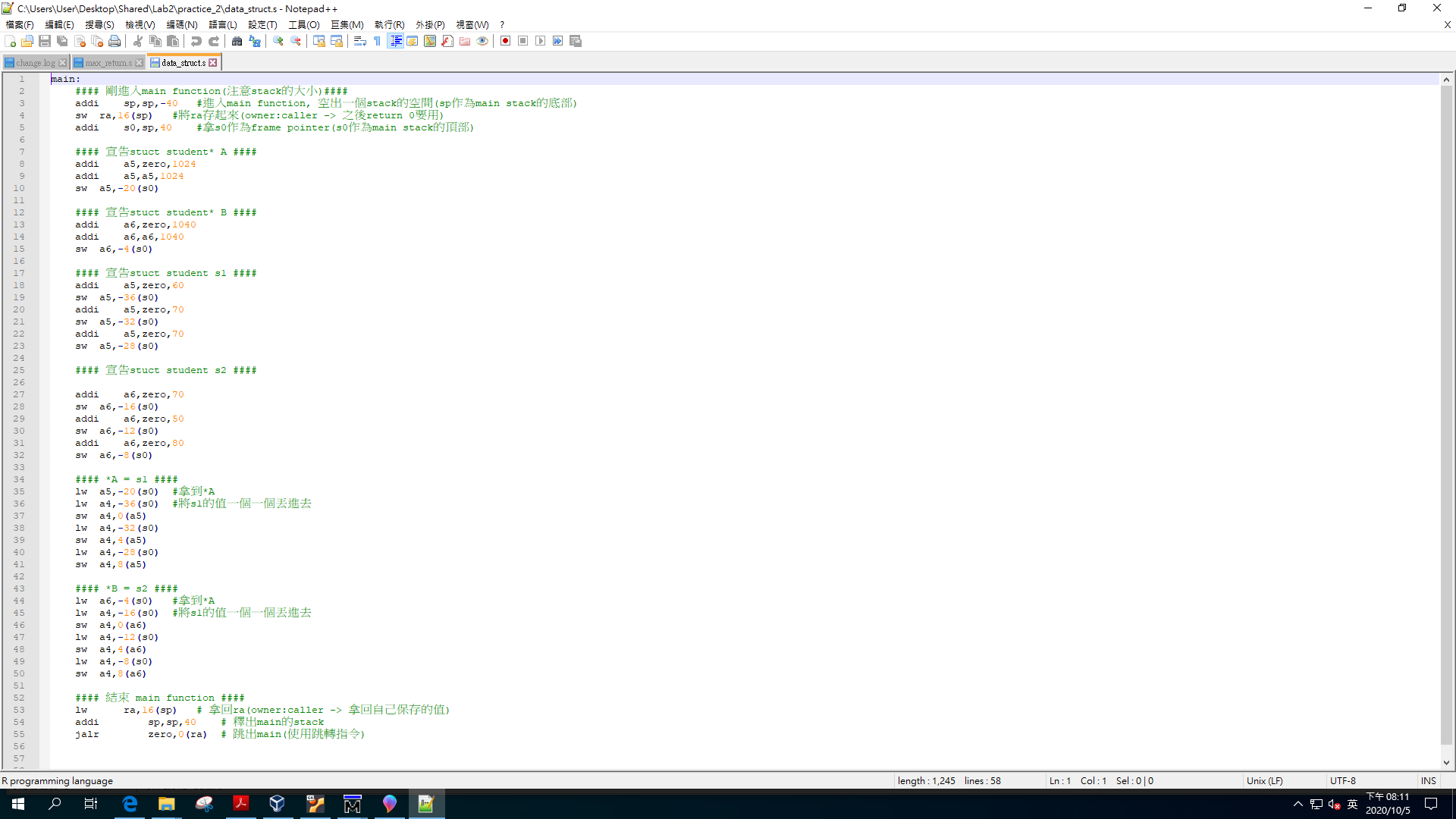
sw a4,8(a6)

#### 結束 main function ####

lw ra,16(sp) # 拿回ra(owner:caller -> 拿回自己保存的值)

addi sp,sp,40 # 釋出main的stack

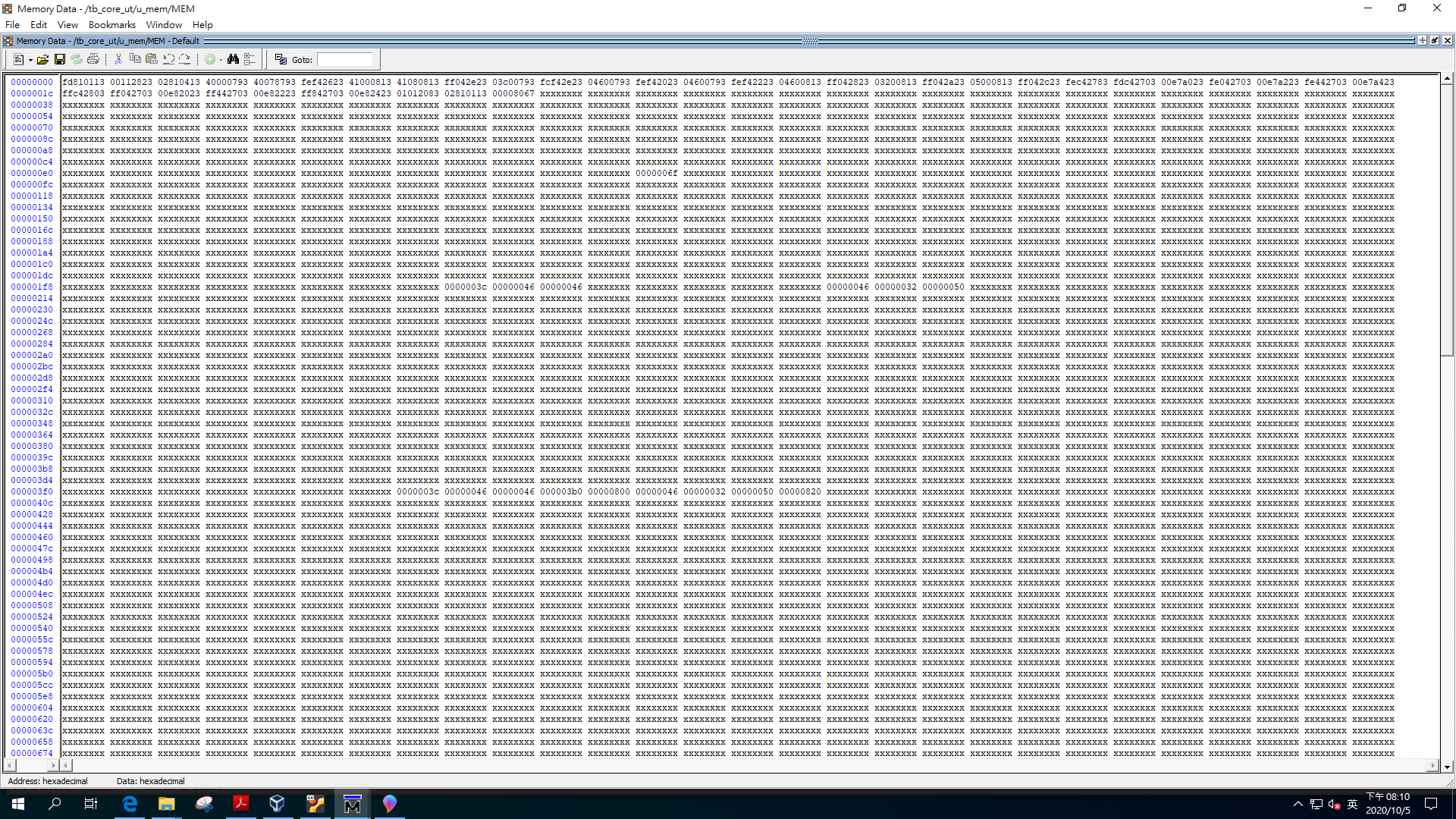
jalr zero,0(ra) # 跳出main(使用跳轉指令)



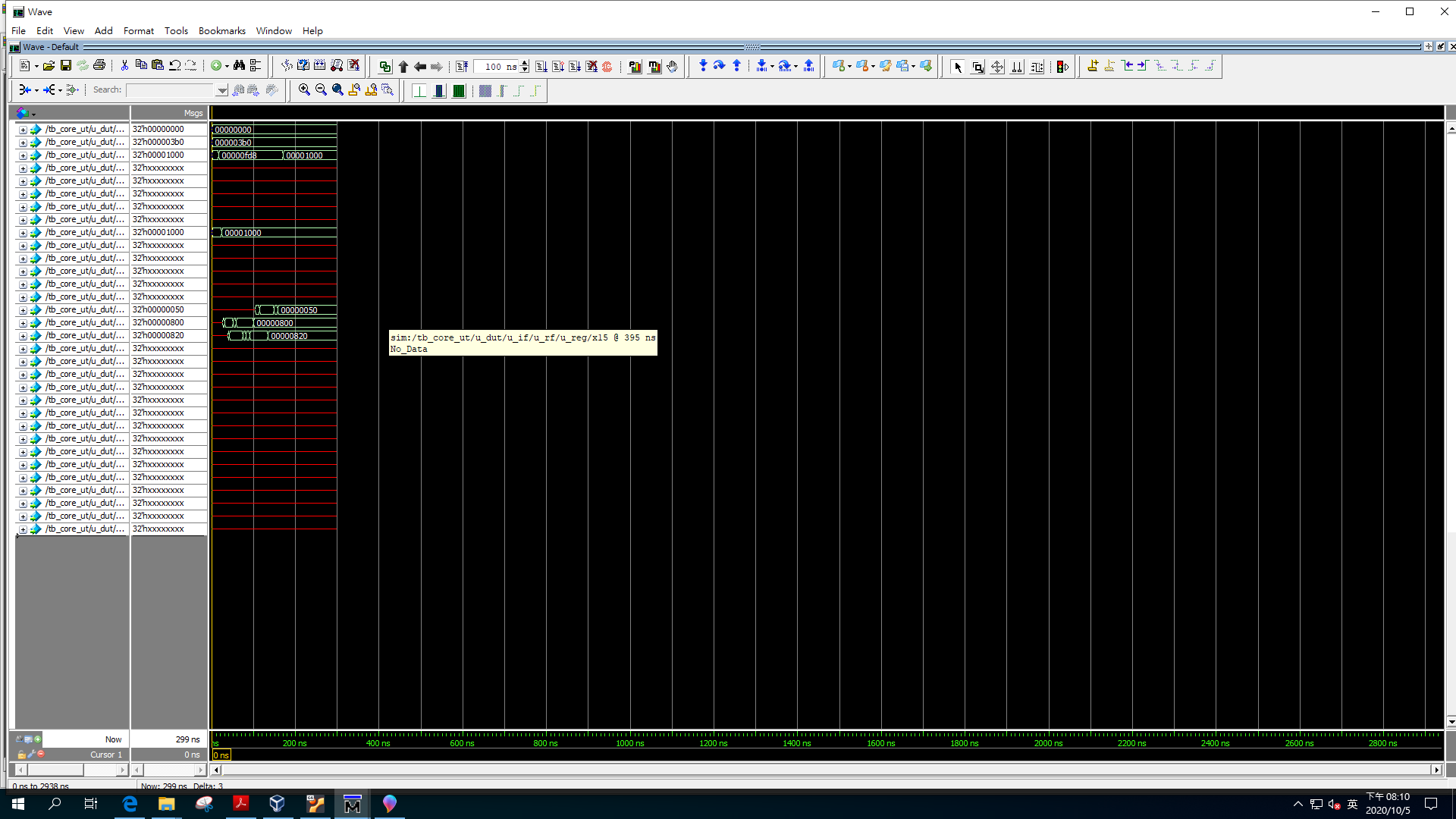
3.結果分析(modelsim)：

Memory

將宣告的兩個struct分別放 在記憶體位址2048(512)和2080(520)



Register



(3)挑戰題

1.題目: 請追蹤以下組合語言，紀錄每次迴圈執行後memory變化並此程式意義

*main:*

*addi sp,sp,-48*

*sw s0,44(sp)*

*addi s0,sp,48*

*li a5,4096*

*addi a5,a5,-2048*

*sw a5,-28(s0)*

*li a5,5*

*sw a5,-32(s0)*

*lw a5,-28(s0)*

*li a4,10*

*sw a4,0(a5)*

*lw a5,-28(s0)*

*addi a5,a5,4*

*li a4,92*

*sw a4,0(a5)*

*lw a5,-28(s0)*

*addi a5,a5,8*

*li a4,55*

*sw a4,0(a5)*

*lw a5,-28(s0)*

*addi a5,a5,12*

*li a4,1*

*sw a4,0(a5)*

*lw a5,-28(s0)*

*addi a5,a5,16*

*li a4,46*

*sw a4,0(a5)*

*sw zero,-20(s0)*

*j .L2*

*.L6:*

*sw zero,-24(s0)*

*j .L3*

*.L5:*

*lw a5,-24(s0)*

*addi a5,a5,1*

*slli a5,a5,2*

*lw a4,-28(s0)*

*add a5,a4,a5*

*lw a4,0(a5)*

*lw a5,-24(s0)*

*slli a5,a5,2*

*lw a3,-28(s0)*

*add a5,a3,a5*

*lw a5,0(a5)*

*bge a4,a5,.L4*

*lw a5,-24(s0)*

*slli a5,a5,2*

*lw a4,-28(s0)*

*add a5,a4,a5*

*lw a5,0(a5)*

*sw a5,-36(s0)*

*lw a5,-24(s0)*

*addi a5,a5,1*

*slli a5,a5,2*

*lw a4,-28(s0)*

*add a4,a4,a5*

*lw a5,-24(s0)*

*slli a5,a5,2*

*lw a3,-28(s0)*

*add a5,a3,a5*

*lw a4,0(a4)*

*sw a4,0(a5)*

*lw a5,-24(s0)*

*addi a5,a5,1*

*slli a5,a5,2*

*lw a4,-28(s0)*

*add a5,a4,a5*

*lw a4,-36(s0)*

*sw a4,0(a5)*

*.L4:*

*lw a5,-24(s0)*

*addi a5,a5,1*

*sw a5,-24(s0)*

*.L3:*

*lw a5,-32(s0)*

*addi a4,a5,-1*

*lw a5,-20(s0)*

*sub a5,a4,a5*

*lw a4,-24(s0)*

*blt a4,a5,.L5*

*lw a5,-20(s0)*

*addi a5,a5,1*

*sw a5,-20(s0)*

*.L2:*

*lw a5,-32(s0)*

*addi a5,a5,-1*

*lw a4,-20(s0)*

*blt a4,a5,.L6*

*li a5,0*

*mv a0,a5*

*lw s0,44(sp)*

*addi sp,sp,48*

*jr ra*

2.觀察結果與分析

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| step/memory address | 512 | 513 | 514 | 515 | 516 |
| 1 | 10 | 92 | 55 | 1 | 46 |
| 2 | 10 | 55 | 92 | 1 | 46 |
| 3 | 10 | 55 | 1 | 92 | 46 |
| 4 | 10 | 55 | 1 | 46 | 92 |
| 5 | 10 | 1 | 55 | 46 | 92 |
| 6 | 10 | 1 | 46 | 55 | 92 |
| 7 | 1 | 10 | 46 | 55 | 92 |

Caption:數字=改變值

推測為bubble sort，因為上述觀察結果符合bubble sort之演算流程:

1.當我們比較過所有元素一次後，可以確保數值最大的元素在最後面

2.接著扣掉陣列中的最後一個元素（因為已經確定它是最大的），接著重複上面的步驟進行兩兩比較

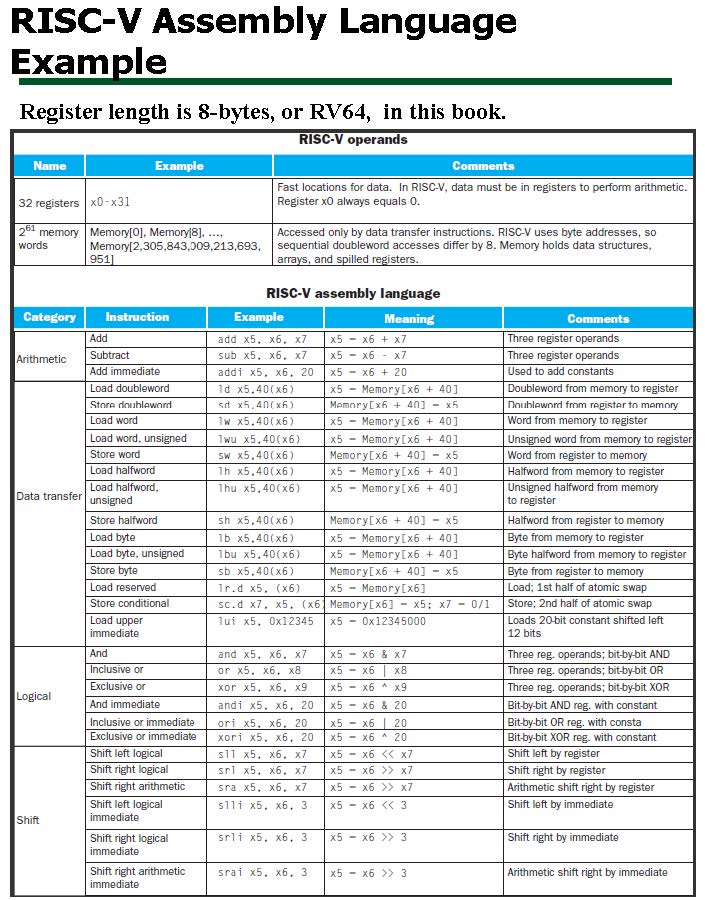
3.重複上述動作，直到排序完畢。

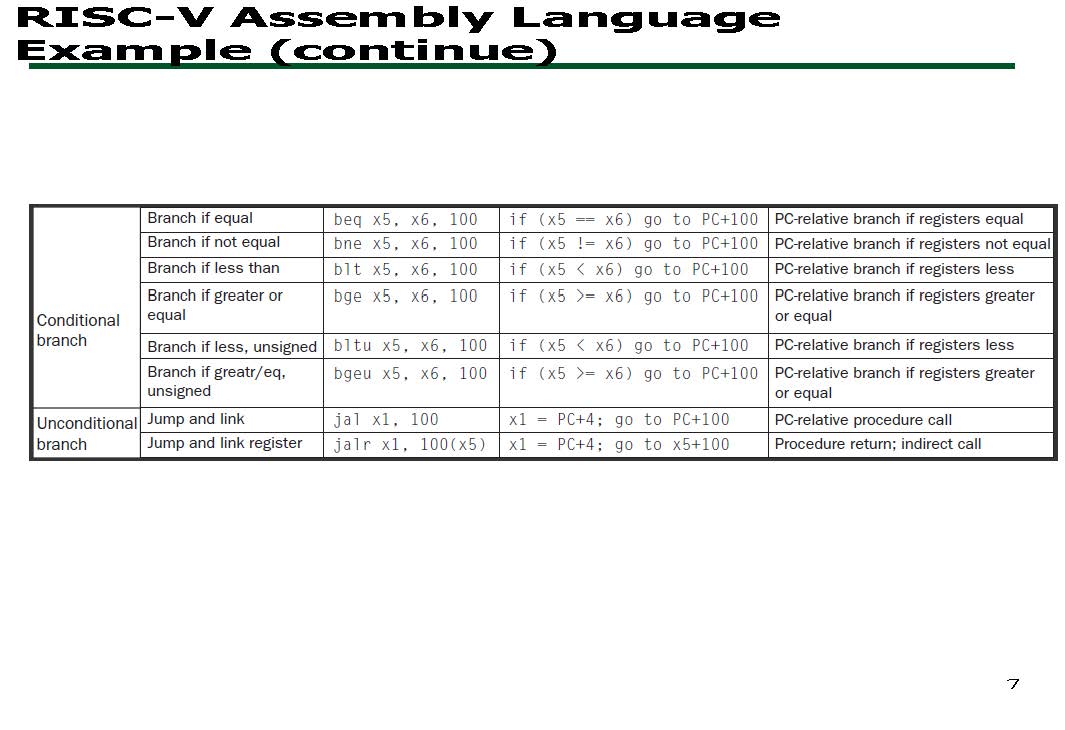
推測實際完整比較流程為:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| step/memory address | 512 | 513 | 514 | 515 | 516 |
| 1 | 10 | 92 | 55 | 1 | 46 |
| 2 | 10 | 55 | 92 | 1 | 46 |
| 3 | 10 | 55 | 1 | 92 | 46 |
| 4 | 10 | 55 | 1 | 46 | 92 |
| 5 | 10 | 55 | 1 | 46 | 92 |
| 6 | 10 | 1 | 55 | 46 | 92 |
| 7 | 10 | 1 | 46 | 55 | 92 |
| 8 | 1 | 10 | 46 | 55 | 92 |
| 9 | 1 | 10 | 46 | 55 | 92 |
| 10 | 1 | 10 | 46 | 55 | 92 |
| 11 | 1 | 10 | 46 | 55 | 92 |

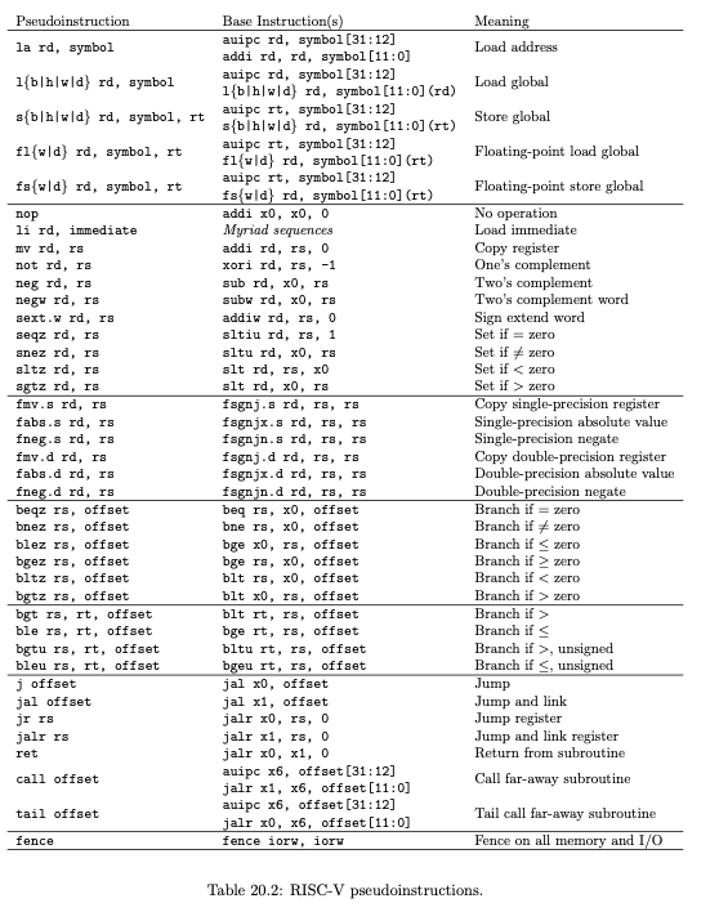
Caption:數字=兩兩相鄰數字比較、數字=整理後數字、棕色為imaginary wall

PS:指令集





偽指令(組譯器除了產生機器語言外，還可以翻譯一些擴展指令，以方便組語的Coding)



二、實驗心得(心得可隨意發揮，但是內容會做工程認證保留檔案)

李宇洋:

此次實驗我們學習到了各種暫存器指令對應的用法以及設定stack的方法。

尤其像是temporary register和saved register的差別，以及用來在函式間互相傳遞的function argument的使用方法。

還有用來設定stack上下限的x2/sp(Stack pointer)和x8/s0(frame pointer)使用方法。

陳旭祺:

本次實驗一樣是寫組合語言，不過相比起來，這次是有照著規定去編程，像RV32I暫存器個別都有規定--x1/ra(Return address)、x2/sp(Stack pointer)、x8/s0(frame pointer)……哪顆暫存器真的不能亂存，像目標暫存器不能為x0，因為x0要恆為0，值得我好好去熟悉他的語法。

蔡晏齊:學到有32種registers，每一個register都有各自的規定。設定好stack空間，預判上下限，以及位置的先前選定，之後的位置，只要使用往下或往上移的累加方式，即可達成指令，最終使用主副迴圈的使用，完成需要的指令，一剛開始學時，真的摸不著頭緒，近日有在研讀相關性知識，及高階語言換成組合語言的轉換，才能夠理解，須待好好加強。

蕭佑永:

本次實驗讓我能夠把stack跟caller, callee的指令及詳細的緣由都學清楚，搭配正課讓我能更不容易忘記。