**Assign1**

Run

Graphical user interface, application, table, Excel

Description automatically generated

Lệnh đầu tiên là 0x20103007: 0010 0000 0001 0000 0011 0000 0000 0111

001000 00000 10000 0011000000000111

Op: 8 (I)(addi)

Rs: 0 ($zero)

Rt: 16 ($s0)

Imm: 12295

Đúng với: addi $s0, $zero, 0x3007

Debug

Ở lần debug đầu tiên, $pc mang giá trị 

Chính là địa chỉ của dòng lệnh đầu tiên *addi $s0, $zero, 0x3007* mang ý nghĩa là pc đang trỏ đến dòng lệnh 1. Khi này $s0 chưa được set giá trị và vẫn là 0x000000

Debug tiếp, $pc mang giá trị



Là địa chỉ của dòng lệnh tiếp *add $s0, $zero, $0 và s0 lúc này đã được gán thêm giá trị 0x3007 và mang giá trị 0x00003007*

*Sau khi sửa lệnh addi $s0, $zero, 0x3007 thành addi $s0,$zero, 0x2110003d có nghĩa là chúng ta đang đòi hỏi việc cộng s0 với số 32bit. Tuy nhiên, lệnh addi chỉ phục vụ cộng số 16bit do đó sau khi chạy chương trình hiện lên như sau:*

Graphical user interface, application, Word

Description automatically generated

Chương trình sẽ chia thành nhiều lệnh nhỏ hơn vd: lui để cộng 16 bit đầu của *0x2110003d vào $1, sau đó cộng tiếp 16 bit cuối bằng lệnh ori. Sau đó cộng $1 vào $s0 bằng lệnh add$16,$0,$1.*

***Assign2:***

Graphical user interface, application, Word

Description automatically generated

Tương tự như đã giải thích ở assign1, 2 lệnh lui và ori sẽ giúp cộng một số 32 bit dần dần vào biến $s0

Bằng cách cộng 16bit đầu rồi cộng tiếp 16 bit cuối

lui



ori

**Assign3:**

Graphical user interface, text, application, Word

Description automatically generated

Li là lệnh gán ngay lập tức 1 số 16bit hoặc 32 bit vào biến . Ở đây, lệnh li muốn gán số 32bit 0x2110003d vào biến $s0 nên đã tách ra làm 2 lệnh là lui và ori.

Còn lệnh thứ 3 là li $s1,0x2 chỉ gán 1 số <32bit nên nó sẽ gán luôn và tương đương với addiu $s1, $s0,0x2

**Assign4;**

Graphical user interface, text, application, Word, email

Description automatically generated

Khi debug, kết quả thanh ghi $s0 thay đổi sau dòng lệnh add $s0,$t1,$t1 là 0x0000000a tương ứng vs 10 và sau dòng lệnh add $s0,$s0,,$t2 là 0x00000009 tương ứng vs 9

* Kết quả đúng = 2\*10-1=9

Addi $t2, $zero, -1

0x200affff : 0010 0000 0000 1010 1111 1111 1111 1111

001000 00000 01010 1111111111111111

Op: addi

Rs: $0

Rt: 10 ($t2)

Imm: -1

Trùng với khuôn mẫu kiểu lệnh I

Add $s0, $t1, $t1 : 0x01298020

0000 0001 0010 1001 1000 0000 0010 0000

000000 01001 01001 10000 00000 100000

Op: R

Rs: 9 ($t1)

Rt: 9 ($t1)

Rd: 16 ($s0)

Shift amount: 0

Function: 32 (add)

Trùng với khuôn mẫu kiểu lệnh R

**Assign 5:**

Graphical user interface, text, application, table

Description automatically generated

2 lệnh addi để gán giá trị cho 2 biến t1 và t2

Mỗi lệnh mul nhân 2 biến và lưu kết quả vào 2 thanh ghi hi và lo, hi sẽ lưu 32bit đầu, lo lưu 32 bit cuối.

Vd như lệnh mul $s0,$t1,$t2 chạy sẽ ghi giá trị hi là 0x00000000 và lo là 0x00000014 tương ứng với 20

Lệnh mul $s0,$s0,3 ghi hi là 0x00000000 và lo là 0x0000003c tương ứng vs 60

Điều bất thường là trong lệnh mul, 3 kphai là 1 biến và khi debug có xuất hiện biến at lưu giá trị của 3



Cuối cùng lệnh mflo giúp sao lưu giá trị của biến s0 sang biến s1.

**Assign 6:**

Table

Description automatically generated

Lệnh la được biên dịch thành 2 lệnh là lui và ori nhằm lưu địa chỉ của các biến X,Y vào biến t8,t9.

Giá trị của X và Y tương ứng vs giá trị khởi tạo, được hiện thị trong Data Segment lần lượt là 0x00000005 và 0xffffffff (5 và -1).

Debug:

Với câu lệnh la, việc lưu địa chỉ của biến X hay Y được lưu qua 1 biến trung gian là $at rồi sẽ được lưu vào &t8 và &t9

Với câu lệnh lw, lưu giá trị của biến X vào biến t1 bằng việc trỏ tới địa chỉ của X qua 0($t8).

Với câu lệnh sw, lưu giá trị của thanh ghi s0 vào biến có địa chỉ t7 (là Z).