**BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH**

**Lab 7 : Lệnh gọi chương trình con, truyền tham số sử dụng ngăn xếp**

|  |  |
| --- | --- |
| **Họ tên** | **MSSV** |
| Phạm Minh Hiển | 20235705 |

**Assignment 1:**

Tạo project để thực hiện Home Assignment 1. Dịch và chạy mô phỏng. Thay đổi các tham số chương trình (thanh ghi **a0**) và quan sát kết quả thực hiện. Chạy chương trình ở chế độ từng dòng lệnh và chú ý sự thay đổi của các thanh ghi, đặc biệt là thanh ghi **pc** và **ra**.

**A screenshot of a computer code

AI-generated content may be incorrect.**

* Quan sát thanh ghi:

+ a0 được lưu giá trị cần tính trị tuyệt đối, ở đây là -45 (0xffffffd3).

+ Sau lệnh **jal abs**, chương trình lưu giá trị địa chỉ của lệnh kế tiếp là lệnh **li a7, 10** vào thanh ghi **ra** (0x00400008) và rẽ nhánh xuống chương trình con **abs.**

+ Các lệnh tiếp theo thực hiện tính giá trị tuyệt đối của a0, cụ thể nếu a0 < 0 thì số âm nên giá trị tuyệt đối là s0 = 0 – a0, *done.* Nếu a0 > 0 thì giá trị tuyệt đối là chính nó nên s0 = a0 + 0.

+ Thanh ghi **pc** tăng giá trị thêm 4 câu lệnh **li** (0x00400004). Khi thực hiện lệnh **jal**, lúc này **pc** có giá trị 0x00400010 tăng 12 byte so với giá trị trước đó do lúc này thanh ghi **pc** đã trỏ xuống lệnh **sub**. Sau các câu lệnh trong *abs* có giá trị 0x0040001c, và khi thực hiện lệnh **jr ra** thì lập tức đổi thành giá trị đã lưu trong **ra** đó là 0x00400008 và chương trình thực hiện câu lệnh **li a7** để kết thúc chương trình.

* Kết quả:

+ Thêm lệnh hiển thị kết quả ta có được:



+ Thử với a0 = 100 :

A black text on a white background

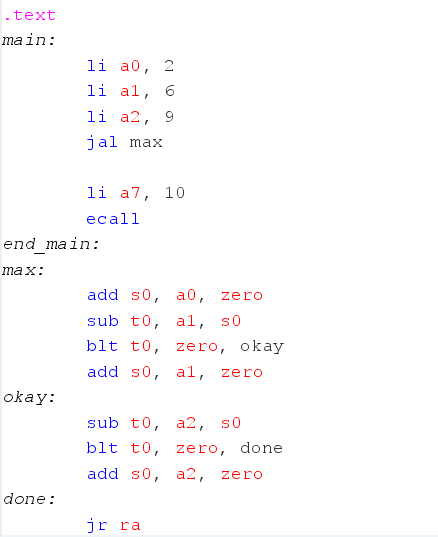
AI-generated content may be incorrect.

+ Các trường hợp khác cũng cho kết quả tương tự

* Vậy chương trình hoạt động chính xác.

**Assignment 2:**

Tạo project để thực hiện Home Assignment 2. Dịch và chạy mô phỏng. Thay đổi các tham số chương trình (thanh ghi **a0**, **a1**, **a2**) và quan sát kết quả thực hiện. Chạy chương trình ở chế độ từng dòng lệnh và chú ý sự thay đổi của các thanh ghi, đặc biệt là thanh ghi **pc** và **ra**.



* Quan sát thanh ghi:

+ Sau 3 câu lệnh **li** các thanh ghi a0, a1, a2 lần lượt chứa giá trị 2, 6, 9.

+ Khi thực hiện lệnh **jal max** thanh ghi **ra** lưu giá trị 0x00400010 là địa chỉ của câu lệnh **li a7, 10.** Đồng thời câu lệnh **pc** có giá trị 0x00400018 tăng thêm 12 byte so với giá trị 0x0040000c sau khi gán giá trị a2 do con trỏ đã chỉ đến lệnh **add** trong chương trình con *max.*

+ Thanh ghi pc lần lượt tăng 4 byte với mỗi lệnh sau đó và có giá trị 0x00400034 khi thực hiện lệnh **add s0, a2, zero**.

+ Khi thực hiện lệnh **jr ra** thì giá trị thanh ghi **pc** lập tức thay đổi thành giá trị đã lưu trong **ra** trước đó và trỏ vào lệnh **li**, thực hiện ecall và kết thúc chương trình.

* Kết quả:

+ Ta thêm lệnh để hiển thị kết quả:

A black and white text

AI-generated content may be incorrect.

+ Thử với bộ số -2, 6, -9:

A black text on a white background

AI-generated content may be incorrect.

+ Thử với các trường hợp khác cũng cho kết quả tương tự.

* Vậy chương trình hoạt động đúng.

**Assignment 3:**

Tạo project để thực hiện Home Assignment 3. Dịch và chạy mô phỏng. Thay đổi tham số chương trình (thanh ghi **s0**, **s1**), quan sát quá trình và kết quả thực hiện. Chú ý sự thay đổi giá trị của thanh ghi **sp**. Quan sát vùng nhớ được trỏ bởi thanh ghi **sp** trong cửa sổ Data Segment.

**Source Code:**

.text

main:

li s0, 1

li s1, 2

push:

addi sp, sp, -8

sw s0, 4(sp)

sw s1, 0(sp)

work:

add a0, s0, zero

li a7, 1

ecall

add a0, s1, zero

li a7, 1

ecall

pop:

lw s0, 0(sp)

lw s1, 4(sp)

addi sp, sp, 8

add a0, s0, zero

li a7, 1

ecall

add a0, s1, zero

li a7, 1

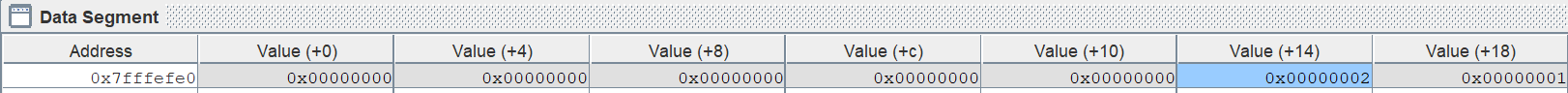
ecall

* ***Để dễ quan sát kết quả, chương trình đã được bổ sung thêm lệnh để hiển thị kết quả ra Run I/O.***
* Quan sát thanh ghi:

+ Ta khởi tạo s0 = 1, s1 = 2.

+ Ban đầu thanh ghi **sp** có giá trị 0x7fffeffc, sau khi thực hiện lệnh **addi sp, sp, -8**, ta đã cấp phát bộ nhớ 8 byte cho 2 giá trị s0 và s1 và thay đổi giá trị thanh ghi **sp** thành 0x7fffeff4.

+ Sau lệnh **sw** ta đã lưu giá trị 1 và 2 vào địa chỉ 0x7fffeff8 và 0x7fffef4.



+ Sau đó ta load giá trị 2 tại 0x7fffef4 ( 0(sp) ) vào s0 và load giá trị 1 tại 0x7fffef8 vào s1, từ đó ta đã đổi được giá trị 2 thanh ghi s1 và s0 cho nhau.

+ Thực hiện khôi phục **sp** về giá trị ban đầu.

* Kết quả:

A black text on a white background

AI-generated content may be incorrect.

+ Ta thấy giá trị s0 = 1 và s1 = 2 đã được đổi vị trị thành 2 và 1.

+ Thử với bộ số khác cũng cho kết quả tương tự.

* Vậy chương trình hoạt động chính xác.

**Assignment 4:**

Tạo project để thực hiện Home Assignment 4. Dịch và chạy mô phỏng. Thay đổi tham số ở thanh ghi **a0** và kiểm tra kết quả ở thanh ghi **s0**. Chạy chương trình ở chế độ từng dòng lệnh và quan sát sự thay đổi giá trị của các thanh ghi **pc**, **ra**, **sp**, **a0**, **s0**. Liệt kê các giá trị trong vùng nhớ ngăn xếp khi thực hiện chương trình với n = 3.

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer code

AI-generated content may be incorrect.**

* Quan sát thanh ghi:

+ Khi thực hiện **jal WARP** địa chỉ lệnh **add** trong *print* là 0x00400004 được lưu trong thanh ghi **ra,** thanh ghi **pc** lập tức nhận giá trị và trỏ đến địa chỉ 0x00400020, là vị trí lệnh **addi** đầu tiên trong *WARP.*

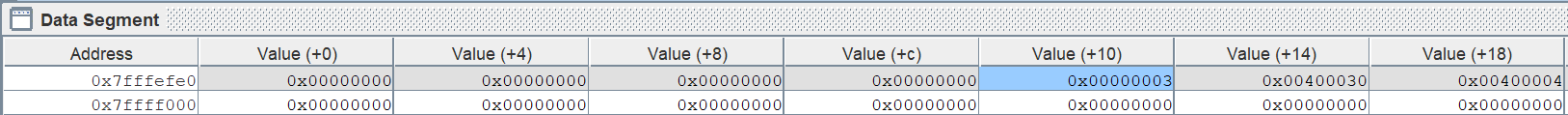
+ Thanh ghi sp có giá trị ban đầu là 0x7fffeffc đã trở thành 0x7fffeff8 do được cấp phát 4 byte sau lệnh **addi** đầu tiên trong *WARP*.

+ Sau **sw** trong *WARP*, giá trị trong **ra** được lưu trong địa chỉ 0x7fffeff8.

+ Khai báo a0 = n = 3, **jal FACT** ghi đè địa chỉ 0x00400030 của lệnh **lw ra, 0(sp)** vào thanh ghi **ra,** thanh ghi **pc** ngay lập tức nhận giá trị và trỏ đến địa chỉ 0x0040003c, chuẩn bị thực hiện lệnh **addi sp, sp, -8.**

+ Thanh ghi sp được cấp phát thêm 8 byte thành 0x7fffeff0.

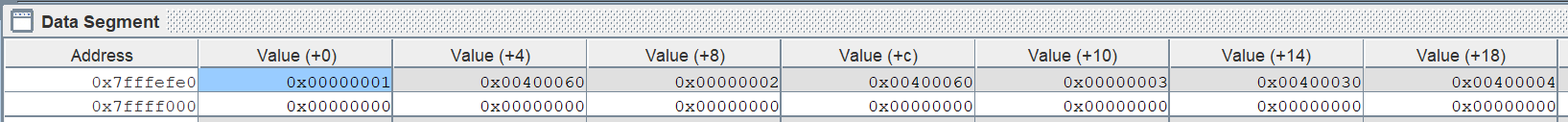
+ Hai lệnh **sw** sau đó lưu giá trị trong thanh ghi **ra** hiện tại vào 0x7fffeff4 và a0 vào 0x7fffeff0.



+ Chạy đên lệnh **bge**, do a0 = 3 > 2 nên rẽ nhánh *recursive* và thực hiện **addi.** Sau đó **jal FACT** được thực hiện, ghi đè địa chỉ của **lw s1, 0(sp)** vào thanh ghi **ra** và thay đổi giá trị thanh ghi **pc** và vị trí con trỏ thành địa chỉ của **addi** trong *FACT*.

+ Tiếp tục vòng lặp cho đến khi a0 = 1 < 2 thì **j done.**

+ Lúc này **sp** có giá trị 0x7fffefe0, **ra** có giá trị 0x00400060.



+ Ở hình trên là các giá trị được lưu trong thanh ghi **ra** từ trái sang phải lần lượt là a0, ra, a0, ra, a0, ra, ra.

+ Lấy giá trị tại 0x7fffefec là 0x00400060 và load vào **ra**, và giá trị 1 tại 0x7fffefe0 vào a0. Khôi phục thanh ghi **sp** thành 0x7fffefe8 và thực hiện **jr ra** để trở lại địa chỉ 0x00400060.

+ Thực hiện lấy giá trị 2 tại 0x7fffefe8 vào s1 và tính giai thừa.

+ Lặp lại 2 bước trên cho đến khi lấy ra hết giá trị lưu trong thanh ghi **sp.**

+ Cuối cùng thanh ghi **sp** được trả về giá trị ban đầu, thanh ghi **ra** trở về giá trị 0x00400004 và chương in kết quả chính xác ra màn hình:

A screenshot of a computer

AI-generated content may be incorrect.

* Thử với n = 5:

A screenshot of a computer

AI-generated content may be incorrect.

* Thử với các trường hợp khác cũng cho kết quả tương tự.
* Vậy chương trình hoạt động.

**Assignment 5:**

Viết chương trình con tìm giá trị lớn nhất, nhỏ nhất và vị trí tương ứng trong danh sách gồm 8 số nguyên được lưu trữ trong các thanh ghi từ a0 đến a7. Ví dụ:

Largest: 9, 3 => Giá trị lớn nhất là 9 được lưu trữ trong a3

Smallest: -3, 6 => Giá trị nhỏ nhất là -3 được lưu trữ trong a6

Gợi ý: Sử dụng ngăn xếp để truyền tham số.

**Source Code:**

.data

largest: .asciz "Largest: "

smallest: .asciz "Smallest: "

space: .asciz ", "

newline: .asciz "\n"

.text

main:

addi sp, sp, -32 # Cap phat 32 bytes cho 8 so

li a0, 0

sw a0, 0(sp)

li a1, 1

sw a1, 4(sp)

li a2, 2

sw a2, 8(sp)

li a3, 3

sw a3, 12(sp)

li a4, 15

sw a4, 16(sp)

li a5, 4

sw a5, 20(sp)

li a6, 6

sw a6, 24(sp)

li a7, 7

sw a7, 28(sp)

lw t0, 0(sp) # temp\_max

lw t1, 0(sp) # temp\_min

li t2, 0 # Vi tri max

li t3, 0 # Vi tri min

li t5, 1 # Bo dem

addi t4, sp ,4 # Tro den vi tri ke tiep

li s8, 8

jal loop

jal print

li a7, 10

ecall

loop:

bge t5, s8, end\_loop # Neu da duyet het 8 so thi out

lw t6, 0(t4) # Load gia tri tu stack

bgt t6, t0, max

min\_check:

blt t6, t1, min

update\_vitri:

addi t4, t4, 4 # Tang con tro

addi t5, t5, 1 # Tang bo dem

j loop

max:

addi t0, t6, 0 # Cap nhat max

addi t2, t5, 0 # Cap nhat vi tri max

j min\_check

min:

addi t1, t6, 0 # Cap nhat min

addi t3, t5, 0 # Cap nhat vi tri min

j update\_vitri

end\_loop:

jr ra

print:

la a0, largest

li a7, 4

ecall

addi a0, t0, 0

li a7, 1

ecall

la a0, space

li a7, 4

ecall

addi a0, t2, 0

li a7, 1

ecall

la a0, newline

li a7, 4

ecall

la a0, smallest

li a7, 4

ecall

addi a0, t1, 0

li a7, 1

ecall

la a0, space

li a7, 4

ecall

addi a0, t3, 0

li a7, 1

ecall

addi sp, sp, 32 # Khoi phuc sp

jr ra

* Chương trình lưu giá trị của 8 số nguyên trong các thanh ghi a0 – a7 vào ngăn xếp, sau đó duyệt ngăn xếp để tìm số lớn nhất, bé nhất và vị trí của chúng, sau đó in ra màn hình Run I/O.
* Kết quả khi chạy thử với bộ số 0, 1, 2, 3, 15, 4, 6, 7 :

A screen shot of a computer

AI-generated content may be incorrect.

* Bộ số -99, 18, -5, 0, 3, 2, -7, 8:

A blue and black text

AI-generated content may be incorrect.

* Thử với vài bộ số khác cũng cho kết quả chính xác.
* Vậy có vẻ chương trình hoạt động như mong muốn.