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

**LAB 7**

Họ và tên: Phạm Vân Anh

MSSV: 20214988

Mã lớp: 139365

***ASSIGNMENT 1:***

1. **Code**

**Text

Description automatically generated**

*Thực hiện gõ chương trình vào công cụ MARS*

1. **Giải thích code**

* Gán $a0 = -20214988 và kết quả trị tuyệt đối của nó được lưu vào thanh ghi $v0
* Dòng 6: jal abs (nhảy đến thủ tục có nhãn abs)
* Lưu thanh ghi $ra = pc = 0x004000c (địa chỉ lệnh tiếp theo là lệnh nop trong main)
* pc thay đổi từ 0x004008 (địa chỉ jal hiện tại) thành địa chỉ đầu tiên của abs 0x0040001c
* Sau khi thực hiện xong abs, lưu kết quả vào $v0
* Thoát abs bằng lệnh jr $ra 🡪 lưu thanh ghi pc = $ra để quay lại main. pc thay đổi từ 0x0040002x (địa chỉ của jr hiện tại) thành 0x004000c ($ra)

1. Kết quả

Table

Description automatically generated

***ASSIGNMENT 2:***

1. **Code**

**Text

Description automatically generated**

*Thực hiện gõ chương trình vào công cụ MARS*

1. **Giải thích**

* Gán giá trị $a0 = 5, $a1 = 4, $a2 = 7. Tìm max trong 3 số và lưu vào thanh ghi $v1
* Dòng 8: jal max (nhảy đến thủ tục có nhãn max)
* Thanh ghi $ra = pc nên $ra = 0x0040000c (giá trị hiện tại của pc) và thay đổi thành 0x004001c là địa chỉ của nhãn max
* Thủ tục *max:*

*-* Dòng 19: gán giá trị max tạm thời vào $v1 = $a0 = 5

*-*Dòng 20: gán $t0 = $a1 - $v1 = 4 – 5 = -1

*-* Dòng 21: so sánh $t0 và 0

*+* Nếu $t0 > 0 🡪 $a1 > $v1 = $a0 🡪 cập nhật max = $v1 = $a1

*+* Nếu $t0 < 0 🡪 $a1 < $v1 = $a0 = 5 🡪 okay

*-* Dòng 24: okay

*+* Xét $t0 = $a2 - $a0

*+* So sánh $t0 và 0

* Nếu $t0 > 0 🡪 cập nhật max = $a2
* Nếu $t0 < 0 🡪 max = $a0 🡪 done

- Dòng 29: *done:* jr $ra 🡪 pc = $ra, để kết thúc trở về main nên pc đổi từ 0x00400040 (địa chỉ jr hiện tại) thành 0x00400010 (địa chỉ tiếp theo tức là lệnh nop)

1. **Kết quả**

* Gán giá trị:

**Background pattern

Description automatically generated**

* max = $v1

****

* Địa chỉ pc và $ra

****

***ASSIGNMENT 3:***

* + - 1. **Code**

**Text

Description automatically generated**

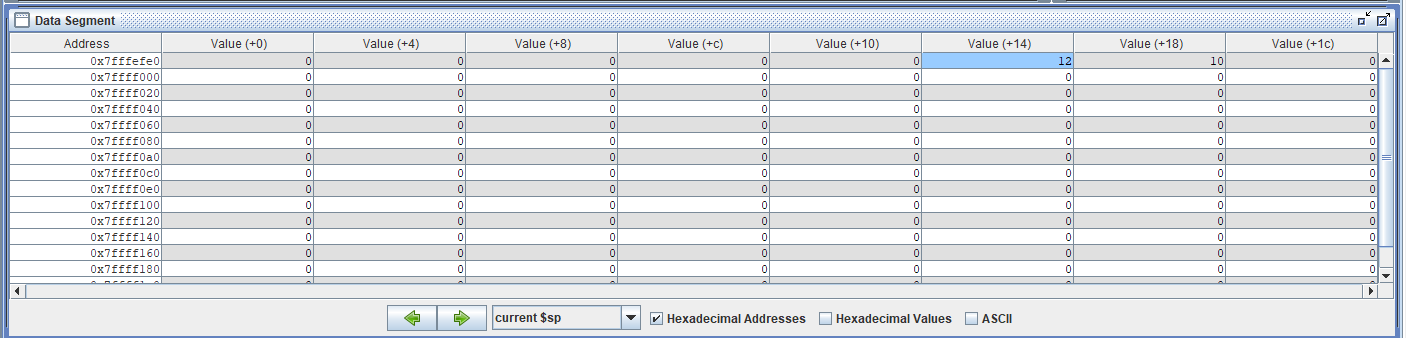
*Thực hiện gõ chương trình vào công cụ MARS*

1. **Giải thích**

* Tạo ngăn xếp để push và pop 2 tham số 10,12 được lưu vào thanh ghi $s0, $s1
* Hàm push:
* Dòng 7: thanh ghi $sp lùi 8 byte để đủ chỗ cho 2 phần tử 10, 12 của stack vì stack có đáy ở dưới và chiều địa chỉ là từ trên xuống 🡪 trừ đi 8.
* $sp chứa địa chỉ của các phần tử trên cùng của stack

🡪 sp (ban đầu) = 0x7fffeffc thay đổi thành $sp = 0x7fffeff4

* Dòng 8: sw để push lần lượt tham số vào thanh ghi

****

* Hàm pop:
* pop lần lượt các giá trị ở trong stack ra ngoài 🡪 pop 12 ra trước và lưu vào $s0, tương tự 10 lưu vào $s1
* Vì pop cả 2 phần tử ra ngoài nên phải +8 (kích thước 1 phần tử là 4 byte) 🡪 $sp = 0x7fffeffc là địa chỉ ban đầu 🡪 stack rỗng

1. **Kết quả:**

* Giá trị của thanh ghi $s0 và $s1

****

* Địa chỉ của sp sau khi chạy xong chương trình:

****

***ASSIGNMENT 4:***

* + - 1. Code

Text

Description automatically generated

Text

Description automatically generated

*Thực hiện gõ chương trình vào công cụ MARS*

* + - 1. **Giải thích code**
* Thanh ghi $fp lưu con trỏ tới frame pointer, $sp – stack pointer
* Thanh ghi $a0 lưu kết quả tính giai thừa
* Hàm WARP:
* Khai báo 2 vị trí trong stack cho frame pointer *$fp* và địa chỉ return *$ra 🡪* Giá trị thanh ghi $fp = 0x7fffeffc; lưu địa chỉ $ra ra bộ nhớ ở địa chỉ 0($sp) bằng lệnh sw



* $fp = $sp
* Dòng 23: tạo ra 2 ngăn nhớ của stack
* Gán giá trị của $ra vào ngăn $sp trỏ vào
* Lưu N vào $a0 🡪 $a0 = 0x00000003



* Restore giá trị từ 0($sp) vào lạ $ra để gọi lệnh jr $ra trở về địa chỉ chính xác của lệnh tiếp theo trong main.
* Hàm FACT:
* Khai báo 3 vị trí trong stack để trả về các giá trị $fp, $ra và $a0

🡪 $sp -12 để tạo ra 3 vị trí

🡪 $sp = 0x7fffeff4 thành 0x7fffefe8, lưu lần lượt giá trị vào stack.

* So sánh tham số đưa vào với 2

- Nếu <2 🡪 kết quả = 1

- Nếu >2 🡪 hàm RECURSIVE

* Nhảy đến done để Restore lại các giá trị return vào thanh ghi $ra để lệnh jr $ra được thực thi chính xác
* Hàm RECURSIVE:
* Giảm a0 đi 1 đơn vị và tiếp tục quay lại load đưa vào stack
  + - 1. **Kết quả**

**Graphical user interface, application

Description automatically generated**

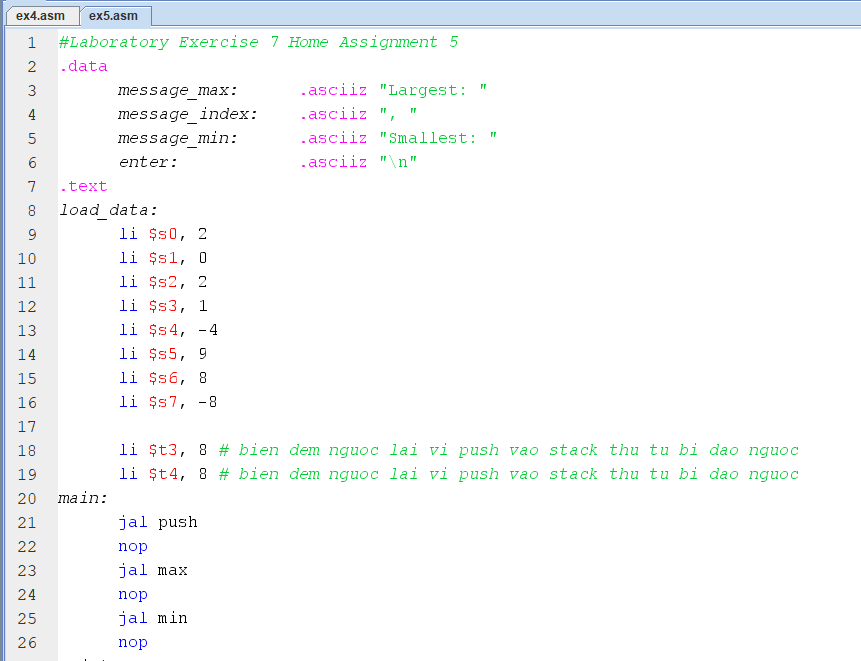
* Bảng stack

**Table, timeline

Description automatically generated with medium confidence**

***ASSIGNMENT 5:***

**Code**

****

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Giải thích**

* Chương trình tìm max, min bằng các đặt max = 1 giá trị cho trước và dịch chuyển $sp để pop lần lượt ra các phần tử ở đầu stack và đem so sánh với max và min, nếu thay đổi thì cập nhật max & min luôn.
* Chương trình tìm max, min và vị trí thanh ghi lưu trữ của nó, được lưu từ $s0 🡪 $s7.

**Table

Description automatically generated**

* Thanh ghi $a1, $a2 lưu max và min

****

* Thanh ghi $t1, $t2 lưu index max, min tương ứng

****

* Thanh ghi $t3, $t4 lưu giá trị biến index để đếm sau các vòng lặp tìm max, min.
* Hàm main:
* Nhảy đến push để lưu các giá trị dãy số vào lần lượt vào stack
* Nhảy đến max để tìm max
* Nhảy đến min để tìm min
* Hàm MAX:
* Pop từ stack và lưu vào $v1 nên giá trị sẽ thay đổi trở thành phần tử được pop ra
* $sp tăng lên 4 để trỏ đến phần tử tiếp theo của stack
* Giảm biến đếm thanh ghi $t3
* Kiểm tra xem có kết thúc dãy chưa, nếu $t3 = 0 🡪 kết thúc
* So sánh $v1 (vừa được pop ra) và $a0 (max hiện tại) để xem cập nhật hay tiếp tục vòng lặp
* Sau khi hàm end thì đưa stack về dãy ban đầu
* Hàm MIN:
* Hoạt động tương tự max
* Giá trị min lưu vào thanh ghi $a2
* Index lưu vào thanh ghi $t4
* Vì các thủ tục không lồng nhau nên thanh ghi $ra không phải lưu lại ra bộ nhớ để restore nên nó thay đổi theo địa chỉ con trỏ pc.

**Kết quả:**

**Graphical user interface, text, application

Description automatically generated**

**Table

Description automatically generated**