**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

🞠◊🞠◊🞠◊🞠◊🞠

****

**BÁO CÁO MÔN HỌC**

**(Môn học: Kiến trúc máy tính và hợp ngữ)**

**NHÓM : CAAL230180\_20\_2\_02CLC**

**BUỔI : CHIỀU THỨ 2, Tiết 7-10**

**HỌC KỲ : 2**

**NĂM HỌC : 2020-2021**

**GVHD : ThS Nguyễn Đăng Quang**

**THỰC HIỆN :**

**HỌ TÊN : Nguyễn Hoàng Nhật**

**MSSV : 19110031**

**Thành phố Hồ Chí Minh, tháng 5 năm 2021**

**LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành đến Khoa Đào tạo Chất lượng cao, Trường Đại học Sư phạm kỹ thuật TP.HCM đã tạo điều kiện thuận lợi cho chúng em học tập và hoàn thành đề tài cuối kỳ này. Đặc biệt, em xin bày tỏ lòng biết ơn sâu sắc đến thầy ThS. Nguyễn Đăng Quang đã truyền đạt kiến thức và hướng dẫn em trong quá trình hoàn thành đề tài.

Em đã cố gắng vận dụng những kiến thức đã học được trong học kỳ qua để hoàn thành bài báo cáo. Nhưng do kiến thức hạn chế và không có nhiều kinh nghiệm thực tiễn nên khó tránh khỏi những thiếu sót trong quá trình nghiên cứu và trình bày. Rất kính mong sự góp ý của quý thầy để bài báo cáo của em được hoàn thiện hơn.

Một lần nữa, em xin trân trọng cảm ơn sự quan tâm giúp đỡ của Thầy đã giúp đỡ em trong quá trình thực hiện bài báo cáo này.

Xin trân trọng cảm ơn!

**MỤC LỤC**

[PHẦN 1. YÊU CẦU ĐỀ BÀI 1](#_Toc73565631)

[PHẦN 2. XÂY DỰNG CHƯƠNG TRÌNH 1](#_Toc73565632)

[**1.** **Xác định chi tiết yêu cầu đề bài** 1](#_Toc73565633)

[**2.** **Thuật toán và xây dựng các Program label** 1](#_Toc73565634)

[- Sử dụng stack để in giá trị 32bit: 1](#_Toc73565635)

[+ Program label xác định vị trí các local variables: 3](#_Toc73565636)

[+ Program label xác định vị trí EBP: sau khi kết thúc vòng lặp để tính tất cả các biến trong hàm main thì ta cộng giá trị thanh ghi EBX với 4 để trỏ tới địa chỉ EBP 4](#_Toc73565637)

[+ Program label xác định vị trí return address hàm main: 5](#_Toc73565638)

[+ Program label xác định vị trí argument hàm main: 5](#_Toc73565639)

[3. Biên dịch chương trình 6](#_Toc73565640)

[PHẦN 3. KIỂM THỬ CHƯƠNG TRÌNH 7](#_Toc73565641)

[- Các trường hợp khai báo biến local vars khác nhau 7](#_Toc73565642)

[- Thay đổi số lượng tham số khác nhau, số lượng biến cục bộ khai báo 7](#_Toc73565643)

[- Thử thêm vòng lặp for để kiểm tra giá trị tại địa chỉ của các argument 8](#_Toc73565644)

[TÀI LIỆU THAM KHẢO 9](#_Toc73565645)

# DANH MỤC CÁC HÌNH

[Hình 1.1 Yêu cầu đề bài 1](#_Toc73565460)

[Hình 2.2.1. Program label in giá trị 32 bit write\_hex 2](#_Toc73565461)

[Hình 2.2.2. Stack [1] 3](#_Toc73565462)

[Hình 2.2.3. ESP đang trỏ vào vị trí biến y 3](#_Toc73565463)

[Hình 2.2.4. Program label xác định các biến trong hàm main 4](#_Toc73565464)

[Hình 2.2.5 Xác định vùng chứa các biến 4](#_Toc73565465)

[Hình 2.2.4 Xác định vị trí EBP 5](#_Toc73565466)

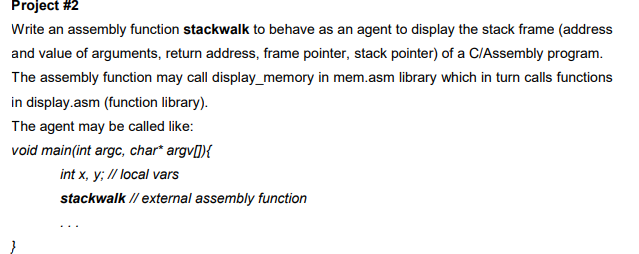
[Hình 2.2.5 Xác định vị trí return address hàm main 5](#_Toc73565467)

[Hình 2.2.6. Xác định số argument truyền vào và vị trí tất cả argument truyền vào 6](#_Toc73565468)

[Hình 3.1 Hoàn thành đề tài 7](#_Toc73565469)

# 

# PHẦN 1. YÊU CẦU ĐỀ BÀI



Hình 1.1 Yêu cầu đề bài

Mục tiêu :

* Viết hàm **stackwalk** hoạt động như một nhân tố để in ra Stack Frame của một chương trình C, lấy đúng vị trí của các thành phần trong Stack Frame (gồm địa chỉ, giá trị của các tham số, địa chỉ trả về, frame pointer, stack pointer).
* Chương trình phải tổng quát trong các trường hợp.
* Chương trình dễ đọc, dễ hiểu, dễ phát triển.

**PHẦN 2. XÂY DỰNG CHƯƠNG TRÌNH**

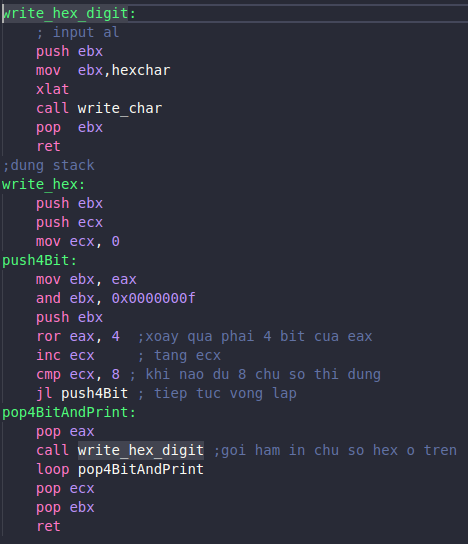
1. **Xác định chi tiết yêu cầu đề bài**

* In giá trị 32 bit (write\_hex – in 8 chữ số hex)
* Lấy chính xác vị trí thành phần trong stack frame

1. **Thuật toán và xây dựng các Program label**

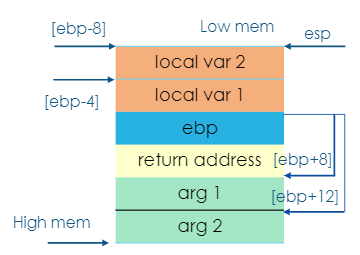
* Sử dụng stack để in giá trị 32bit:

Dùng stack, cắt 32 bit làm 8 phần mỗi phần 4 bit, {28 bit 0} {4 bit 1} and với ebx để ta lấy 4 bit cuối. Khi lấy được ta tiến hành push vào stack, push vào stack từ trái qua phải từng phần một, sau đó pop ra gán vào thanh ghi để tra cứu ( call write\_hex\_digit) sau đó in được.



Hình 2.2.1. Program label in giá trị 32 bit write\_hex

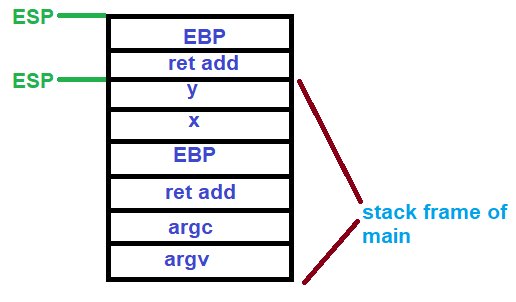
* Truy xuất đến các phần tử trong Stack: Bộ nhớ stack được cấp phát ở vùng nhớ cao, sau đó mỗi lần có phần tử vào trong stack thì con trỏ stack sẽ giảm địa chỉ dần đi. Tất cả các truy xuất đến tham số và biến riêng đều truy xuất thông qua con trỏ nền EBP ( base pointer) - con trỏ lấy làm mốc để tính ra, truy xuất đến các địa chỉ của các tham số, các biến riêng. ( hệ thống 32bit thì mỗi phần tử trong stack là 32bit-4byte) . [1]



Hình 2.2.2. Stack [1]

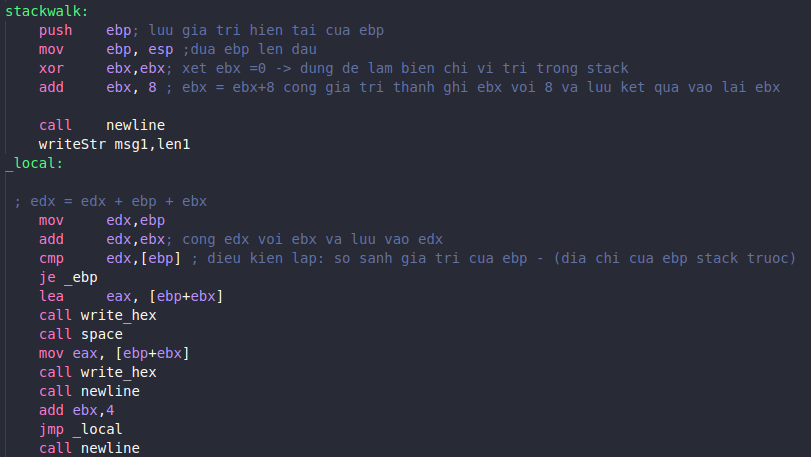
+ Program label xác định vị trí các local variables:

Từ ESP tính được địa chỉ của biến cuối cùng được đưa vào bằng cách sử dụng con trỏ nền EBP (hệ thống 32bit, mỗi phần tử trên stack là 4 byte), sử dụng thanh ghi EBX làm địa chỉ bắt đầu và cộng giá trị thanh ghi EBX với 8 và lưu kết quả vào lại EBX.

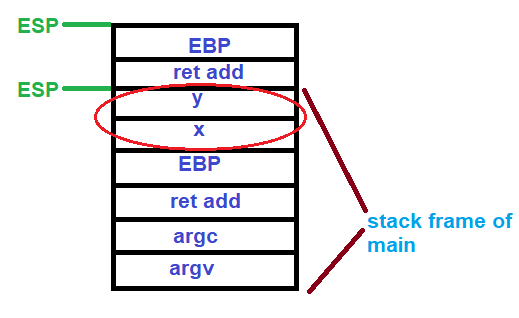


Hình 2.2.3. ESP đang trỏ vào vị trí biến y

Để tính được tất cả các địa chỉ của biến còn lại ta đưa vào vòng lặp với điều kiện kết thúc vòng lặp là dừng khi gặp địa chỉ của EBP trong stack trước.

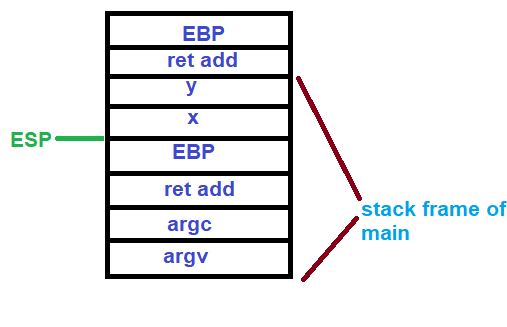


Hình 2.2.4. Program label xác định các biến trong hàm main



Hình 2.2.5 Xác định vùng chứa các biến

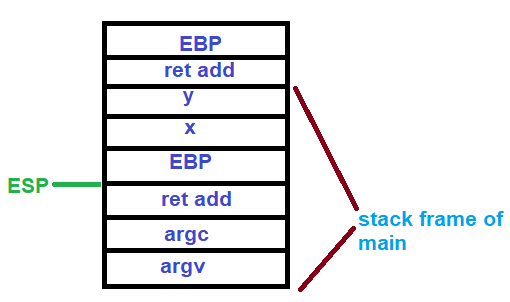
+ Program label xác định vị trí EBP: sau khi kết thúc vòng lặp để tính tất cả các biến trong hàm main thì ta cộng giá trị thanh ghi EBX với 4 để trỏ tới địa chỉ EBP



Hình 2.2.4 Xác định vị trí EBP

+ Program label xác định vị trí return address hàm main:

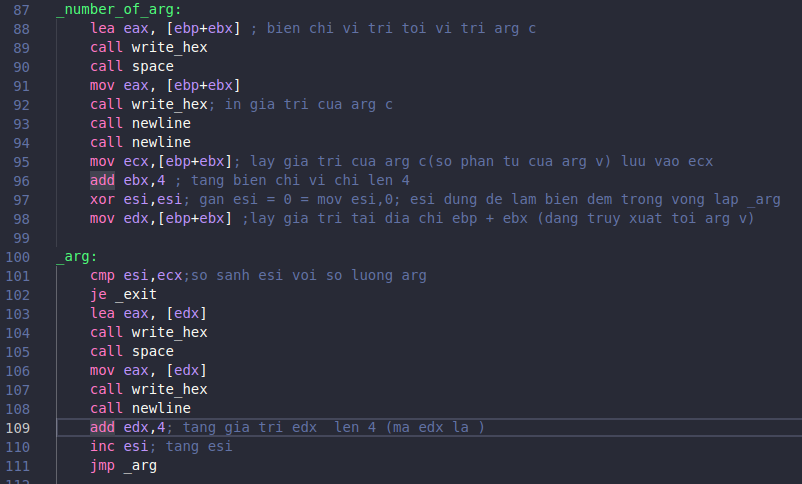
Xác định được vị trí EBP thì ra tiếp tục cộng giá trị thanh ghi EBX với 4 để trỏ tới địa chỉ return address hàm main



Hình 2.2.5 Xác định vị trí return address hàm main

+ Program label xác định vị trí argument hàm main:

Sau khi xác định được vị trí của return address trong hàm main, ta xác định số lượng argument truyền vào bằng cách cộng giá trị thanh ghi EBX với 4 để trỏ tới argc (chính là số lượng phần tử argument truyền vào) và sử dụng vòng lặp tương tự như Program label xác định vị trí các local variables ở trên để xác định được tất cả các argument được truyền vào bằng cách sử dụng thanh ghi đếm cho vòng lặp ESI, khi bắt đầu ta cho ESI = 0 bằng với vị trí bắt đầu của argument v, lấy giá trị tại địa chỉ [ebp+ebx] (đang truy xuất tới arg v) lưu vào thanh ghi EDX. Vào vòng lặp ta thực hiện cộng giá trị thanh ghi EDX cho 4 và tăng giá trị thanh ghi đếm ESI lên 1 và kết thúc vòng lặp khi thanh ghi đếm ESI lớn hơn số lượng arg.

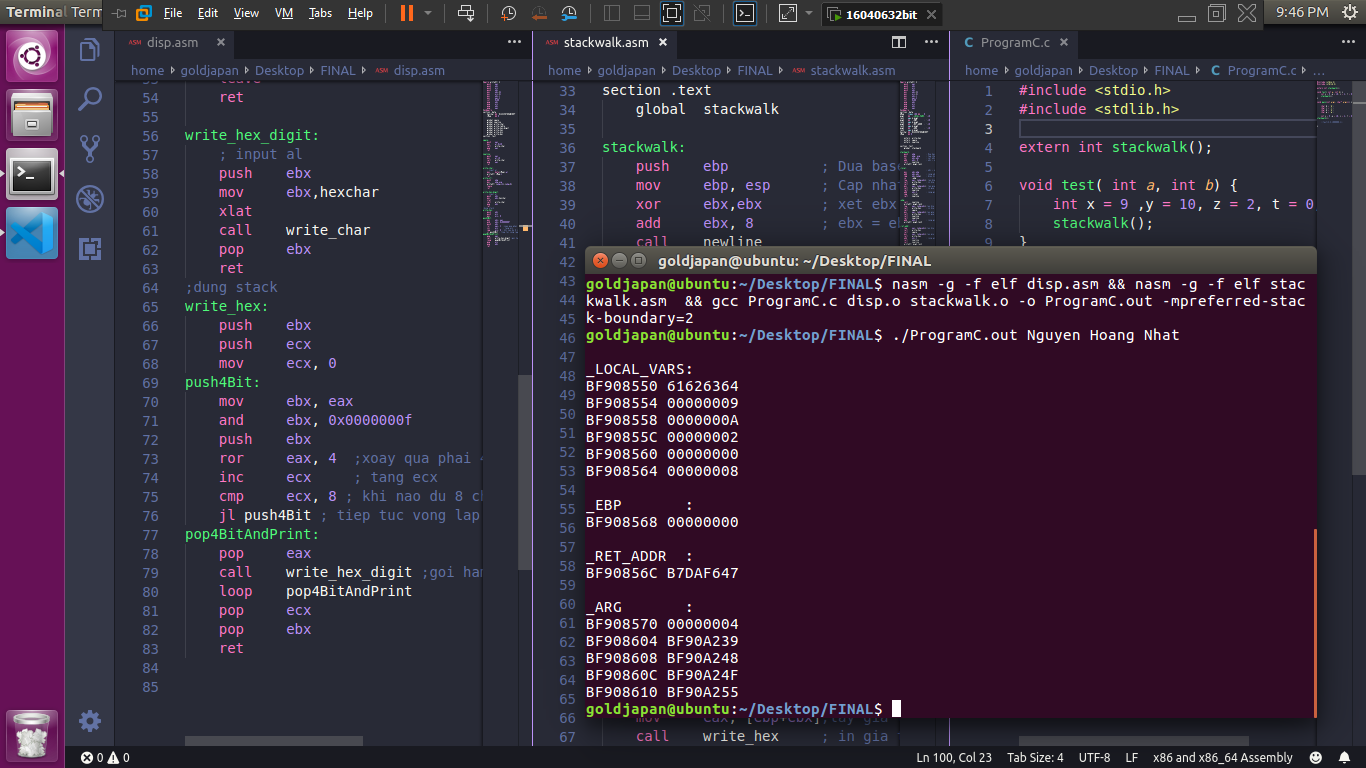


Hình 2.2.6. Xác định số argument truyền vào và vị trí tất cả argument truyền vào

1. Biên dịch chương trình

Ghép các Program label lại với nhau hoàn thành stackwalk.asm, link ProgramC.c, stackwalk.asm và disp.asm và biên dịch chương trình.

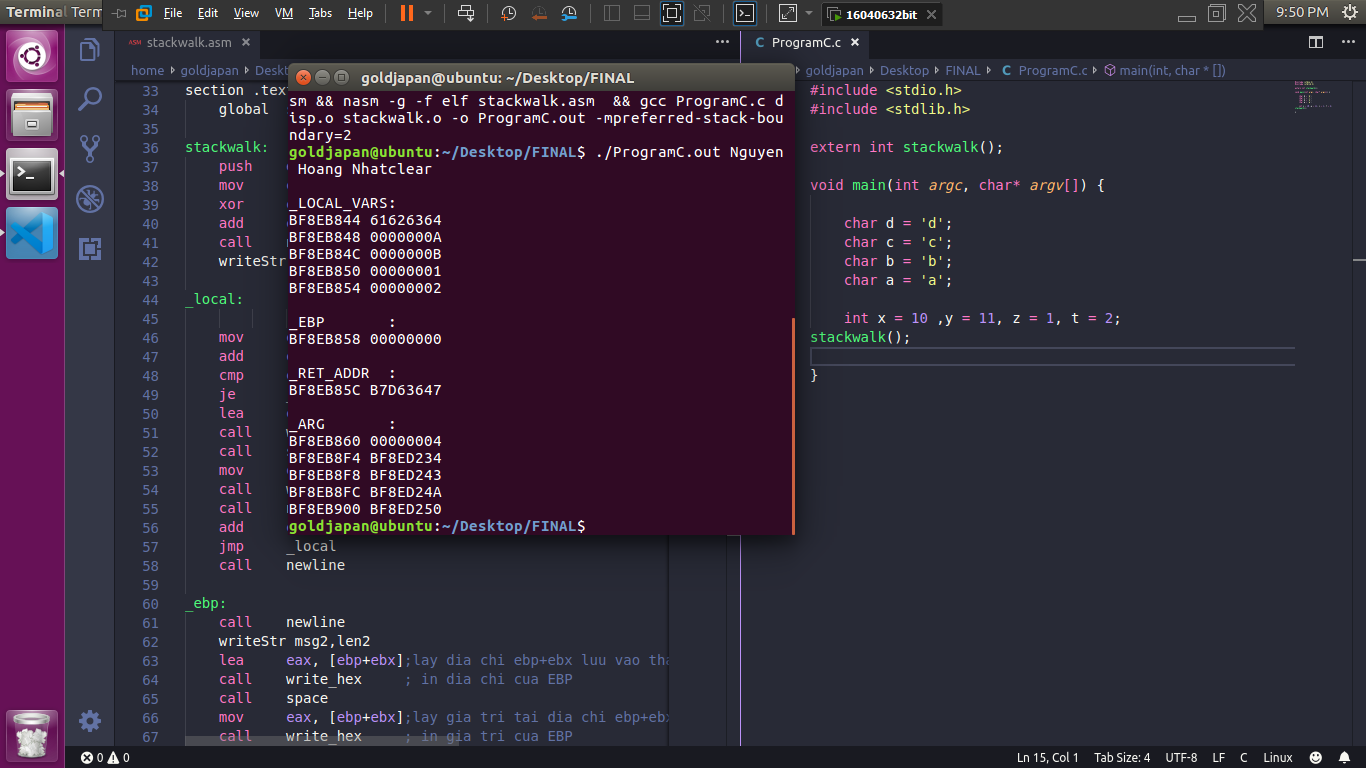
Lệnh: nasm -g -f elf disp.asm && nasm -g -f elf stackwalk.asm && gcc ProgramC.c disp.o stackwalk.o -o ProgramC.out -mpreferred-stack-boundary=2 [3]



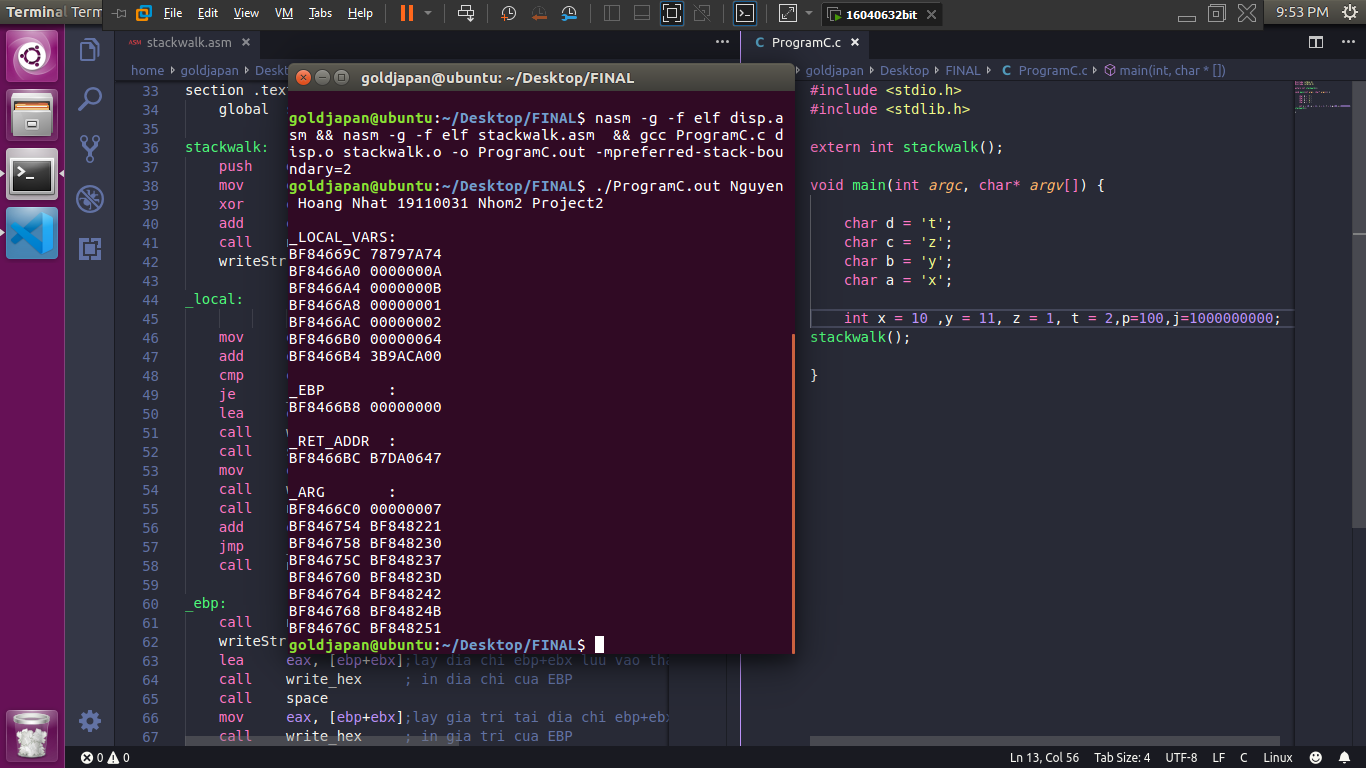
Hình 3.1 Hoàn thành đề tài

# PHẦN 3. KIỂM THỬ CHƯƠNG TRÌNH

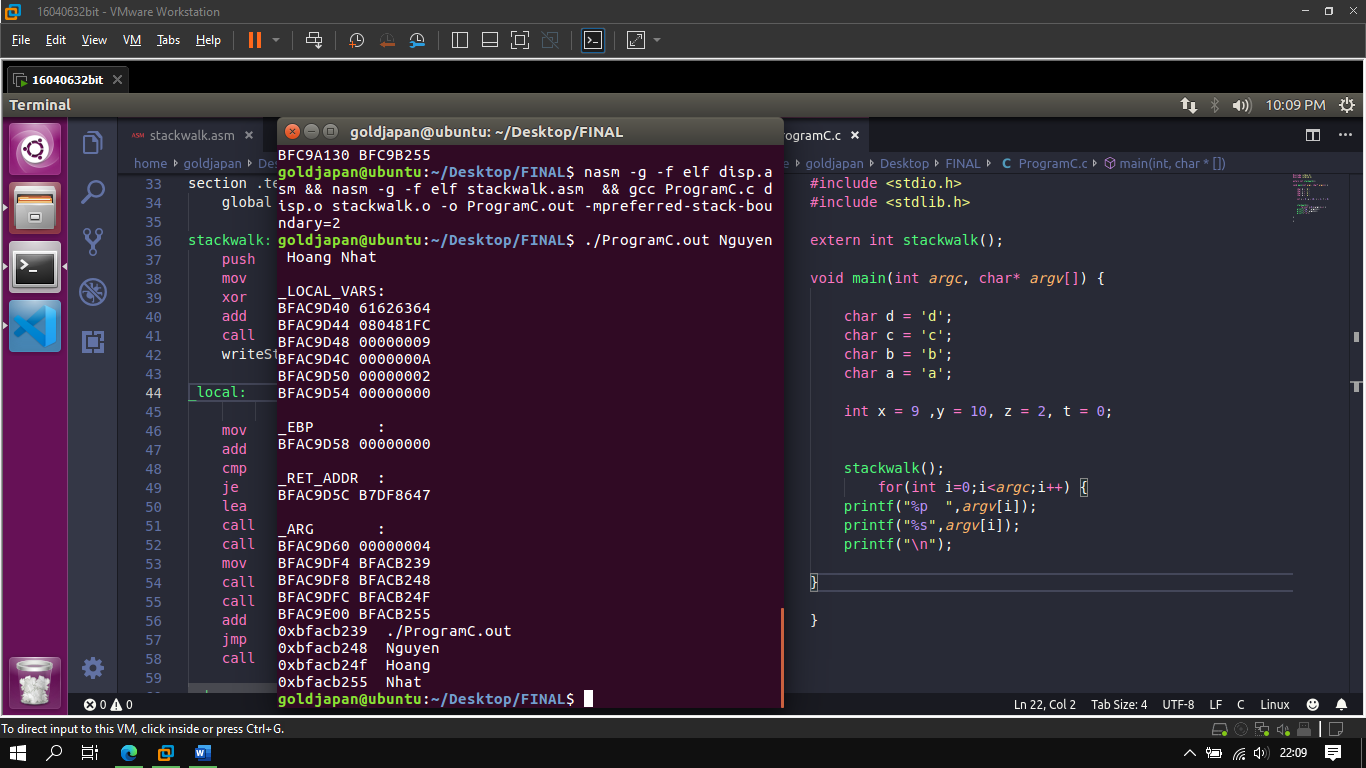
* Các trường hợp khai báo biến local vars khác nhau



* Thay đổi số lượng tham số khác nhau, số lượng biến cục bộ khai báo



* Thử thêm vòng lặp for để kiểm tra giá trị tại địa chỉ của các argument



# TÀI LIỆU THAM KHẢO

1. Nguyễn Đăng Quang, Chapter08.pdf. Retrieved from: <https://app.box.com/s/peio772s4y4li7vye898czj6ufupvz0x>
2. gcc -mpreferred-stack-boundary option. Retrieved from: https://stackoverflow.com/questions/10251203/gcc-mpreferred-stack-boundary-option
3. [Arguments to main in C - Stack Overflow](https://stackoverflow.com/questions/4176326/arguments-to-main-in-c). Retrieved from: https://stackoverflow.com/questions/4176326/arguments-to-main-in-c