## TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN ĐẠI HỌC QUỐC GIA TP.HCM



**BÁO CÁO THỰC HÀNH HỌC PHẦN**

**THIẾT KẾ SOC**

**TUẦN 1  
  
  
Giảng viên: HUỲNH HỮU THUẬN**

**Sinh viên: Nguyễn Thành Trung MSSV: 20200381**

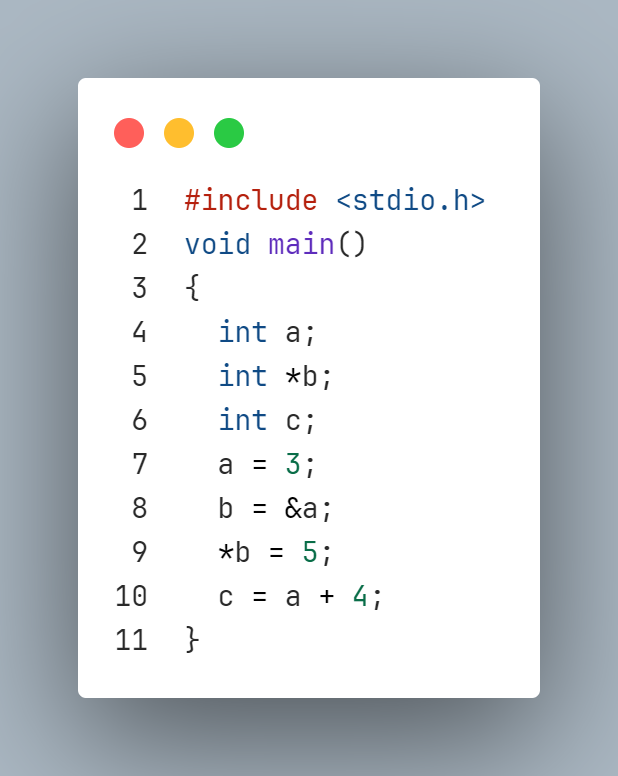
1. **CÂU HỎI CHUẨN BỊ Ở NHÀ**
2. Cho biết hệ thống SoC là gì?

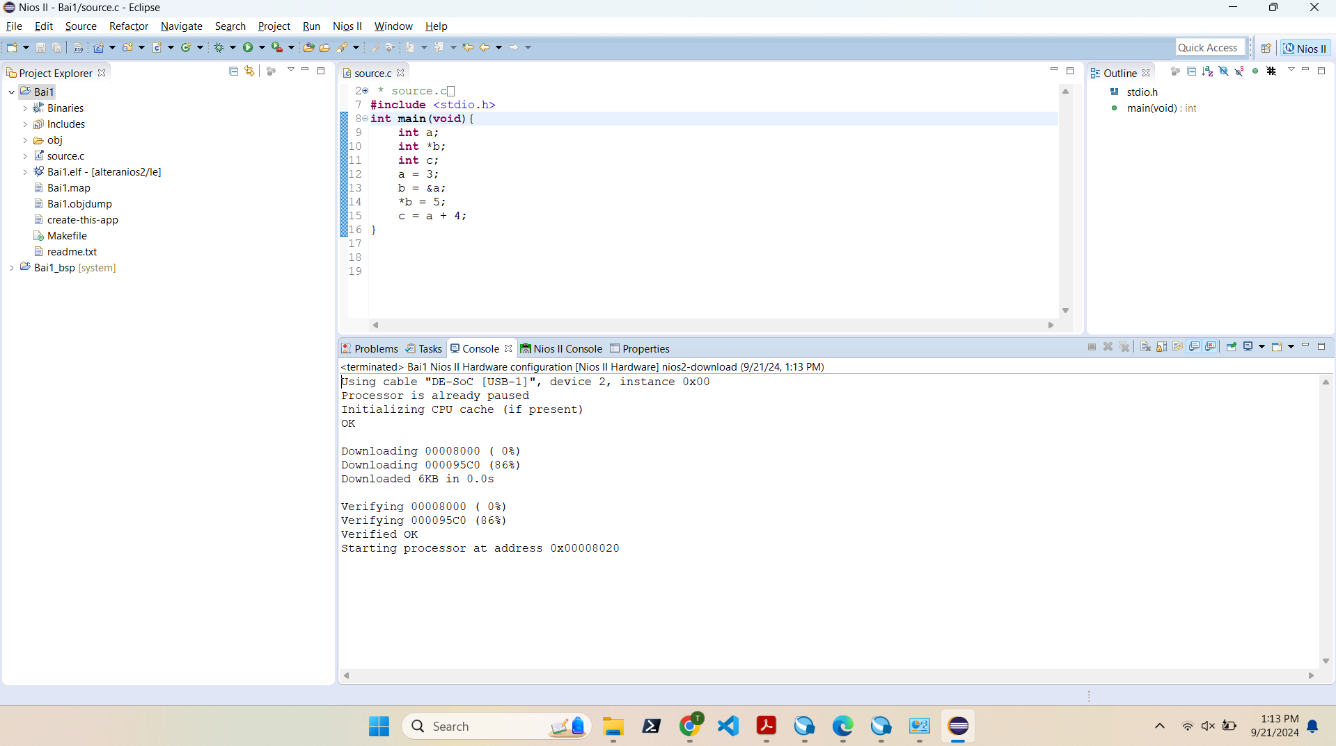
Hệ thống SoC còn được gọi là hệ thống trên vi mạch (còn được gọi là hệ thống trên chip) được viết tắt là SoC. Nó là một vi mạch IC được tích hợp các thành phần của một máy tính hoặc các hệ thống điện tử khác. Hệ thống SoC có thể bao gồm các khối chức năng kĩ thuật số (digital), tương tự (analog), tín hiệu kết hợp (mixed-signal) và cả khối tần số audio (RF). SoC xuất hiện trong điện thoại đi động- một thiết bị điện tử tốn ít năng lượng. Ứng dụng điển hình của các hệ thống SoC là các hệ thống nhúng.

1. Dựa vào tài liệu tham khảo [2], cho biết chức năng của mô đun Jtag-uart.
2. **Giao diện Avalon**: JTAG UART core cung cấp giao diện Avalon, giúp ẩn đi sự phức tạp của giao tiếp JTAG đối với lập trình viên phần mềm. Các bộ vi xử lý như Nios II hoặc Nios V có thể đọc/ghi vào các thanh ghi dữ liệu và điều khiển qua giao diện này.
3. **FIFO hai chiều**: JTAG UART core có các FIFO (First In, First Out) hai chiều giúp tăng băng thông khi truyền dữ liệu qua JTAG. Các FIFO này có thể được tùy chỉnh về độ sâu để tận dụng tài nguyên bộ nhớ có sẵn trong FPGA.
4. **Tích hợp ngắt (Interrupt)**: Core này có một đầu ra ngắt có thể kích hoạt khi dữ liệu sẵn sàng đọc hoặc khi FIFO ghi sẵn sàng nhận dữ liệu, giúp xử lý dữ liệu hiệu quả.
5. **Kết nối JTAG tích hợp**: Core sử dụng mạch điều khiển JTAG tích hợp trong FPGA để kết nối với các mạch logic nội bộ, cho phép nhiều nút logic có thể giao tiếp qua cùng một giao diện JTAG.
6. **Hỗ trợ phần mềm**: Intel cung cấp driver và thư viện phần mềm hỗ trợ cho các bộ vi xử lý Nios II và Nios V, cho phép lập trình viên dễ dàng truy cập JTAG UART thông qua các tiện ích như nios2-terminal hoặc juart-terminal.
7. **Cấu hình bảng mã ký tự (Code Page)**: Đảm bảo rằng bảng mã ký tự của máy tính host và hệ thống trên FPGA tương thích nhau để truyền dữ liệu ký tự chính xác.
8. **BÁO CÁO THỰC HÀNH**
9. Tìm hiểu cách Debug và cho biết ý nghĩa của và phím tắt của các biểu tượng sau trong cửa sổ debug của NIOS II:

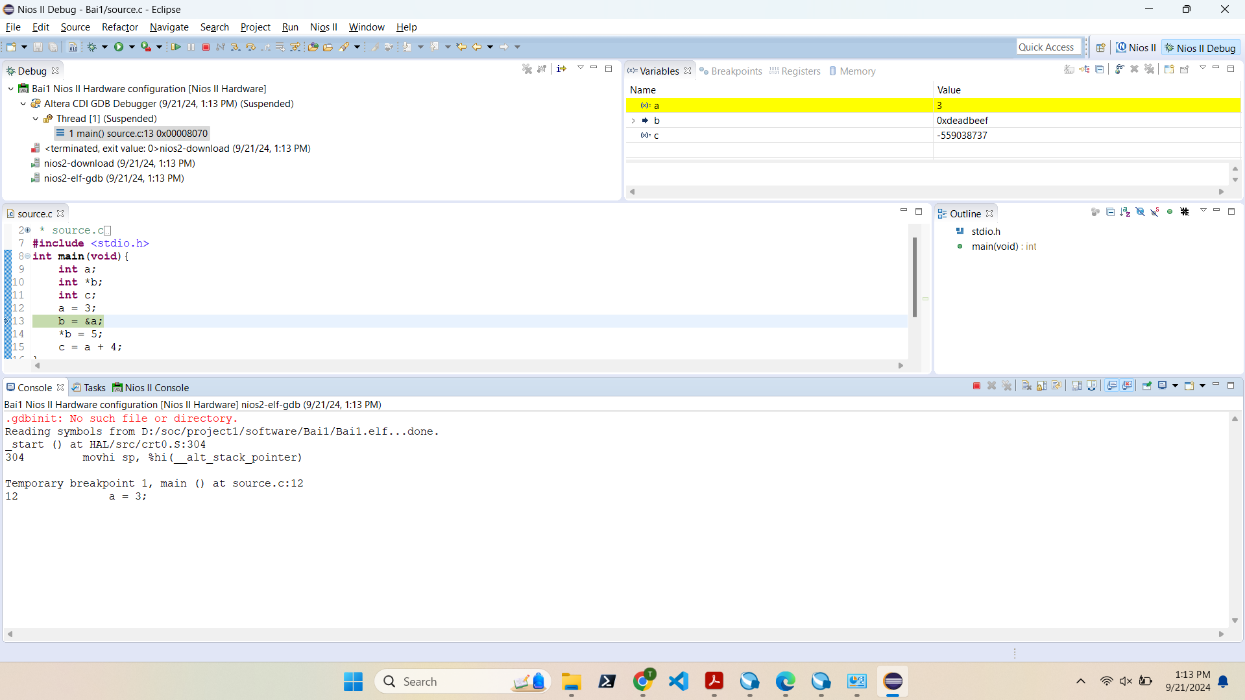
Ý nghĩa của quá phía tắt theo thứ tự từ trái sang phải là:

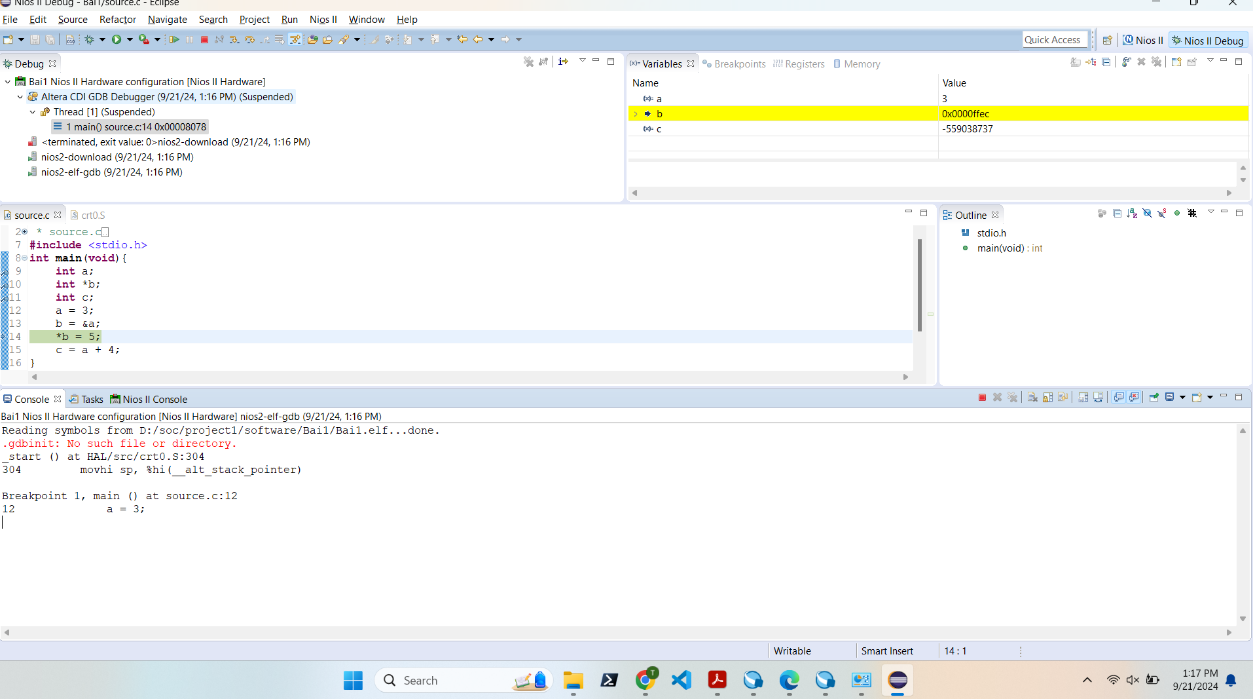
1. **Debug:** Chạy chương trình trong chế độ gỡ lỗi.
2. **Run:** Chạy chương trình trên phần cứng hoặc trình giả lập Nios II.
3. **Run Last Tool:** chạy lại chương trình gần nhất.
4. **Resume:** tiếp tục quá trình thực thi từ điểm dừng tại thời điểm hiện tại.
5. **Pause:** tạm dừng quá trình thực thi chương trình tại thời điểm hiện tại.
6. **Stop:** Dừng quá trình thực thi hiện tại.
7. **Disconnect:** Ngắt kết nối với mục tiêu.
8. **Step Into:** Đi vào từng bước trong mã nguồn để theo dõi quá trình thực thi chi tiết.
9. **Step Over:** Tiến hành thực thi dòng lệnh mà không đi sâu vào các hàm hoặc thủ tục con.
10. **Step Return:** Chạy đến khi hoàn thành hàm hiện tại rồi dừng.
11. **Drop to Frame:** quay lại điểm bắt đầu của phương thức hoặc khung ngăn xếp (stack frame) hiện tại.
12. **Step Filters:**
13. Tiến hành debug đoạn code sau:



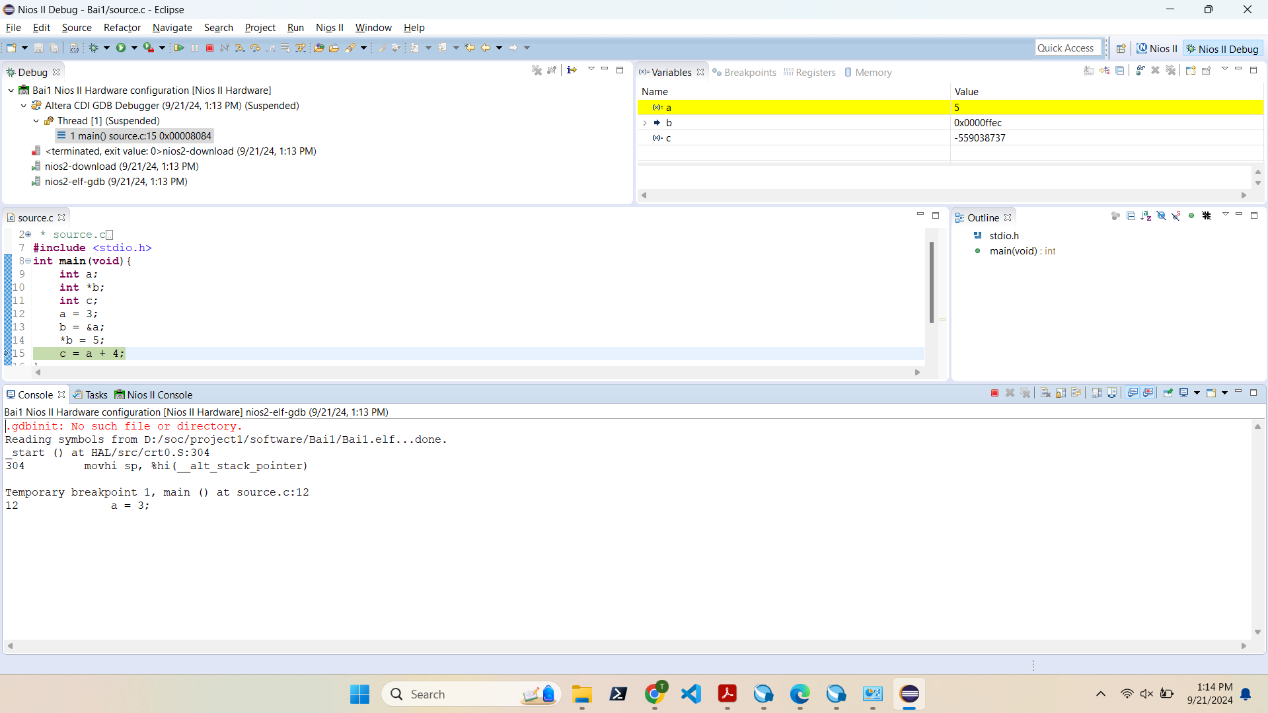
Nạp code thành công

Chuyển qua debug:

Gán A = 3 và gán con trỏ b = địa chỉ của biến a

Truy cập vào con trỏ b và gán giá trị là 5

Gán giá trị của a + 4 vào biến c



Kết thúc chương trình kết qua c = 9 vì khi b lấy địa chỉ của a và gán lại giá trị bằng 5 nên a lúc này bằng 5 + 4 => kết quả bằng 9.

