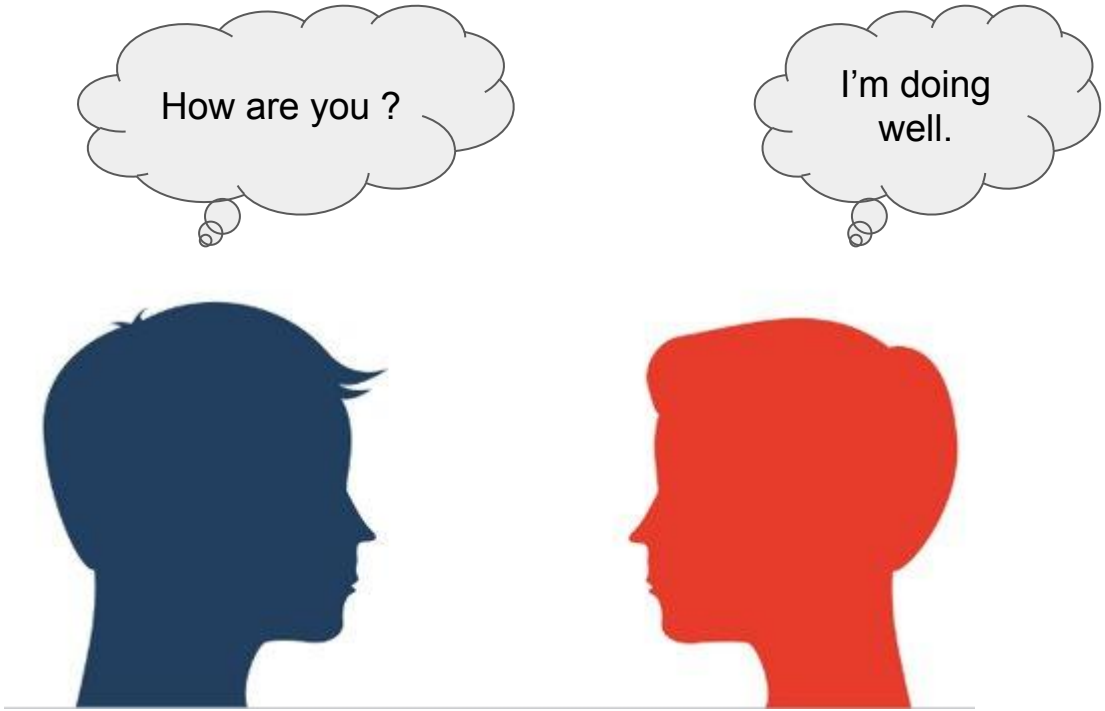
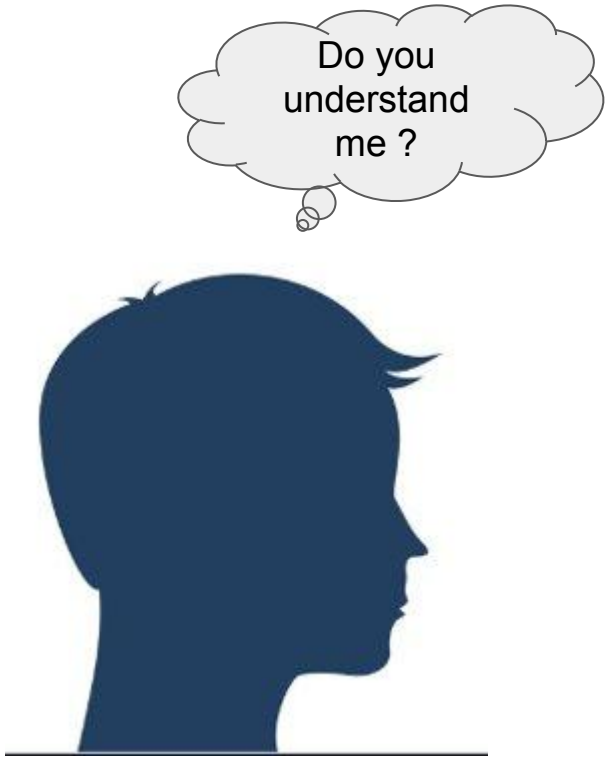
Bài 1: Compiler - Macro

Phan Hoàng Trung

Compiler

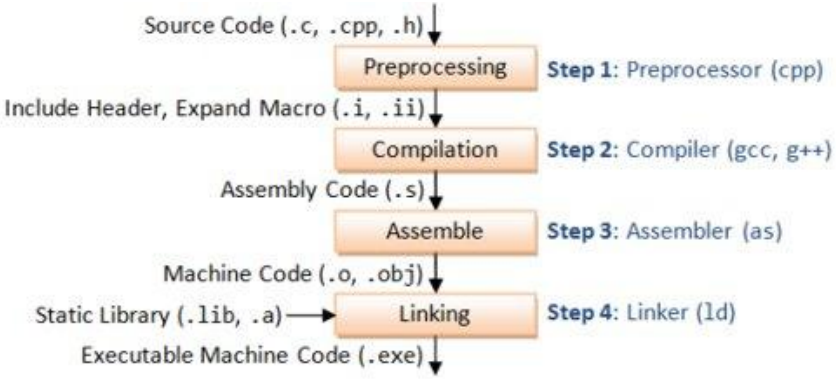


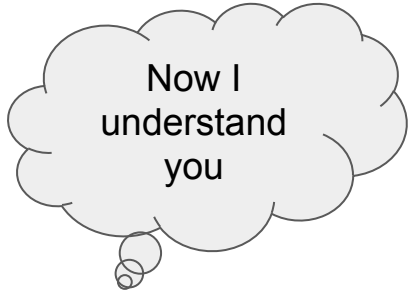
Compiler


Do you
understand
me ?

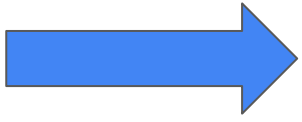

0110101010100101
0101010111101010
1011011001111110

Compiler

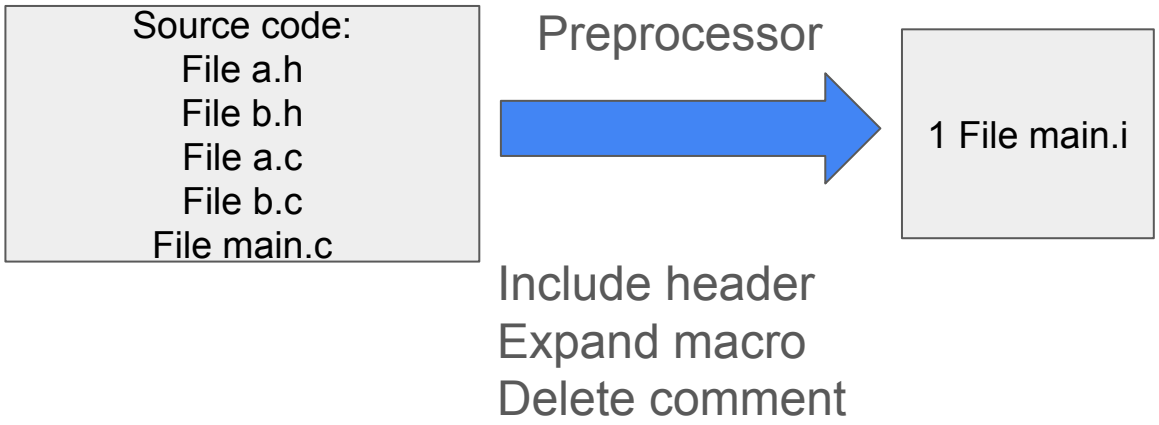

Source Code (.c, .cpp, .h)
Preprocessing
Step 1: Preprocessor (cpp)
Include Header, Expand Macro (.i, .ii)
Compilation
Step 2: Compiler (gcc, g++)
Assembly Code (.s)
Assemble
Step 3: Assembler (as)
Machine Code (.o, .obj)
Static Library (.lib, .a)
Linking
Step 4: Linker (ld)
Executable Machine Code (.exe)


Now I understand you


Machine Code
10011101000110100000
01100011010001110110
10000010111101101110
11110110001011011000
10000010011100011011
10010011000111000000



Compiler


Source code:
File a.h
File b.h
File a.c
File b.c
File main.c
Preprocessor
1 File main.i
Include header
Expand macro
Delete comment

Macro

là từ dùng để chỉ những thông tin được xử lý ở quá trình tiền xử lý (Preprocessor). Chia làm 3 nhóm chính:

- #include
- #define, #undef
- #if, #elif, #else, #ifdef, #ifndef

Chỉ thị tiền xử lý #include

Chỉ thị #include dùng để chèn nội dung của một file vào mã nguồn chương trình.

- Tái sử dụng mã nguồn
- Phân chia chương trình thành các phần nhỏ, giúp quản lý mã nguồn hiệu quả

Chỉ thị tiền xử lý #define

- Macro là một khái niệm dùng để định nghĩa một tập hợp các hướng dẫn tiền xử lý
- Dùng để thay thế một chuỗi mã nguồn bằng một chuỗi khác trước khi chương trình biên dịch.
- Giúp giảm lặp lại mã, dễ bảo trì chương trình.
- Macro được định nghĩa bằng cách sử dụng chỉ thị tiền xử lý #define

Chỉ thị tiền xử lý #define

```c
#include <stdio.h>

// Định nghĩa hằng số Pi sử dụng #define
#define PI 3.14
int main() {
    // Sử dụng hằng số Pi trong chương trình
    double radius = 5.0;
    double area = PI * radius * radius;

    printf("Radius: %.2f\n", radius);
    printf("Area of the circle: %.2f\n", area);

    return 0;
}
```

Chỉ thị tiền xử lý #define

```c
#include <stdio.h>

// Macro để tính bình phương của một số
#define SQUARE(x) ((x) * (x))

int main() {


    // Sử dụng macro để tính bình phương của num
    int result = SQUARE(5);

    printf("Result is: %d\n", result);

    return 0;
}
```

Chỉ thị tiền xử lý #define

```
#include <stdio.h>

#define DISPLAY_SUM(a,b) \
printf("This is macro to sum 2 number\n");   \
printf("Result is: %d", a+b);

int main() {

    DISPLAY_SUM(5,6)

    return 0;
}
```

Chỉ thị tiền xử lý #define

```c
#include <stdio.h>

// Định nghĩa macro để tìm số lớn hơn giữa hai số
#define MAX(x, y) ((x) > (y) ? (x) : (y))

int main() {
    int a = 10, b = 20;

    // Sử dụng macro để tìm số lớn hơn giữa a và b
    int maxNumber = MAX(a, b);

    printf("The bigger number between %d and %d is:
%d\n", a, b, maxNumber);

    return 0;
}
```

```
The bigger number between 10 and 20 is: 20
```

Chỉ thị tiền xử lý #undef

- Chỉ thị #undef dùng để hủy định nghĩa của một macro đã được định nghĩa trước đó bằng #define

Chỉ thị tiền xử lý #undef

```c
#include <stdio.h>

// Định nghĩa SENSOR_DATA
#define SENSOR_DATA 42

int main() {
    printf("Value of MY_MACRO: %d\n", MY_MACRO);

    // Hủy định nghĩa SENSOR_DATA
    #undef SENSOR_DATA
    // Định nghĩa SENSOR_DATA
    #define SENSOR_DATA 50

    printf("Value of MY_MACRO: %d\n", MY_MACRO);

    return 0;
}
```

Chỉ thị tiền xử lý #if, #elif, #else

- #if sử dụng để bắt đầu một điều kiện tiền xử lý.
- Nếu điều kiện trong #if là đúng, các dòng mã nguồn sau #if sẽ được biên dịch
- Nếu sai, các dòng mã nguồn sẽ bị bỏ qua đến khi gặp #endif
- #elif dùng để thêm một điều kiện mới khi điều kiện trước đó trong #if hoặc #elif là sai
- #else dùng khi không có điều kiện nào ở trên đúng.

Chỉ thị tiền xử lý #if, #elif, #else

```c
#include <stdio.h>

typedef enum
{
    GPIOA,
    GPIOB,
    GPIOC
} Ports;

typedef enum
{
    PIN1,
    PIN2,
    PIN3,
    PIN4,
    PIN5,
    PIN6,
    PIN7,
} Pins;

typedef enum
{
```

Chỉ thị tiền xử lý #ifdef, #ifndef

- #ifdef dùng để kiểm tra một macro đã được định nghĩa hay chưa, nếu macro đã được định nghĩa thì mã nguồn sau #ifdef sẽ được biên dịch.
- #ifndef dùng để kiểm tra một macro đã được định nghĩa hay chưa, nếu macro chưa được định nghĩa thì mã nguồn sau #ifndef sẽ được biên dịch

Chỉ thị tiền xử lý #ifdef, #ifndef

abc.txt

```
#ifndef __ABC_H
#define __ABC_H

int a = 10;

#endif
```

```
#include <stdio.h>

#include "abc.txt"
#include "abc.txt"
#include "abc.txt"


int main()
{
    printf("Hello \n");

    return 0;
}
```

Một số toán tử trong macro

```c
#include <stdio.h>

#define STRINGIZE(x) #x
#define DATA 40

int main() {

    // Sử dụng toán tử #
    printf("The value is: %s\n",
STRINGIZE(DATA));

    return 0;
}
```

```c
#include <stdio.h>

#define STRINGIZE_RESULT(x) STRINGIZE(x)
#define STRINGIZE(x) #x
#define DATA 40

int main() {

    // Sử dụng toán tử #
    printf("The value is: %s\n",
STRINGIZE_RESULT(DATA));

    return 0;
}
```

Một số toán tử trong macro

```c
#include <stdio.h>

#define DECLARE_VARIABLE(prefix, number) int prefix##number

int main() {
    // Sử dụng macro để khai báo các biến động
    DECLARE_VARIABLE(var, 1); // int var1;
    DECLARE_VARIABLE(var, 2); // int var2;


    // Gán giá trị cho các biến
    var1 = 10;
    var2 = 20;

    // In ra giá trị của các biến
    printf("var1: %d\n", var1);
    printf("var2: %d\n", var2);

    return 0;
```

Một số toán tử trong macro

Variadic macro

- Là một dạng macro cho phép nhận một số lượng biến tham số có thể thay đổi.
- Giúp định nghĩa các macro có thể xử lý một lượng biến đầu vào khác nhau

```
#define MACRO_NAME(...) // Body
macro
```

Một số toán tử trong macro

```c
#include <stdio.h>

void feature1() { printf("Feature 1 selected\n"); }
void feature2() { printf("Feature 2 selected\n"); }
void feature3() { printf("Feature 3 selected\n"); }
void feature4() { printf("Feature 4 selected\n"); }


int main()
{
    printf("1. Option 1\n");
    printf("2. Option 2\n");
    printf("3. Option 3\n");
    printf("4. Option 4\n");
    printf("5. Exit\n");
```

```c
    // Giả sử option được nhập từ người dùng
```

Một số toán tử trong macro

```
#include <stdio.h>
#define PRINT_MENU_ITEM(number, item) printf("%d. %s\n", number, item)
#define PRINT_MENU(...) \
    do { \
        const char* items[] = {__VA_ARGS__}; \
        int n = sizeof(items) / sizeof(items[0]); \
        for (int i = 0; i < n; i++) { \
            PRINT_MENU_ITEM(i + 1, items[i]); \
        } \
    } while (0)
```

Một số toán tử trong macro

```
#define CASE_OPTION(number, function) case number: function(); break;
#define HANDLE_OPTION(option, ...) \
    switch (option) { \
        __VA_ARGS__ \
        default: printf("Invalid option!\n"); \
    }
```

```
void feature1() { printf("Feature 1 selected\n"); }
void feature2() { printf("Feature 2 selected\n"); }
void feature3() { printf("Feature 3 selected\n"); }
void feature4() { printf("Feature 4 selected\n"); }
```

Một số toán tử trong macro

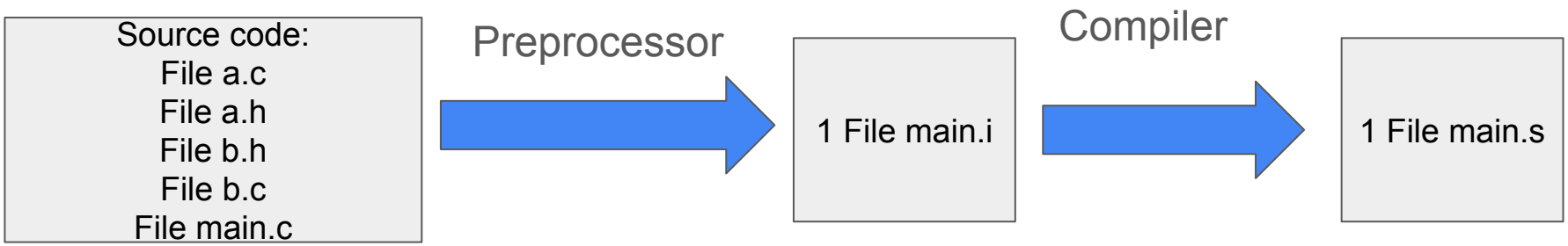
```c
int main()
{
    PRINT_MENU("Option 1", "Option 2", "Option 3","Option4", "Exit");

    // Giả sử option được nhập từ người dùng
    int option ;
    scanf("%d", &option);

    HANDLE_OPTION(option,
        CASE_OPTION(1, feature1)
        CASE_OPTION(2, feature2)
        CASE_OPTION(3, feature3)
        CASE_OPTION(4, feature4)


    )
    return 0;
}
```

Compiler


Source code:
File a.c
File a.h
File b.h
File b.c
File main.c
Preprocessor
1 File main.i
Compiler
1 File main.s

Compiler

Cú pháp:

```
Label:
     Opcode    Operand;  Comment
     Opcode    Operand;  Comment
     Opcode    Operand;  Comment
```

Compiler

- **label** nằm ở cột đầu tiên dùng để xác định vị trí trong bộ nhớ của tập lệnh hiện tại, bắt buộc phải chọn tên duy nhất cho mỗi label.
- **opcode** là mã máy chỉ cho bộ xử lý lệnh nào cần phải thực hiện.
- **operand** là toán hạng xác định vị trí của dữ liệu để thực hiện lệnh. Với tập lệnh Thumb thì có 0,1,2,3, hoặc 4 operand (toán hạng) cách nhau bằng dấu phẩy.
- **comment** là phần chú thích, nó thường được bỏ qua khi biên dịch code, nhưng nó sẽ mô tả giúp cho bạn có thể hiểu được cách phần mềm hoạt động.

Compiler

```
MOV AX, 42 ; Di chuyển giá trị 42 vào thanh ghi AX
```

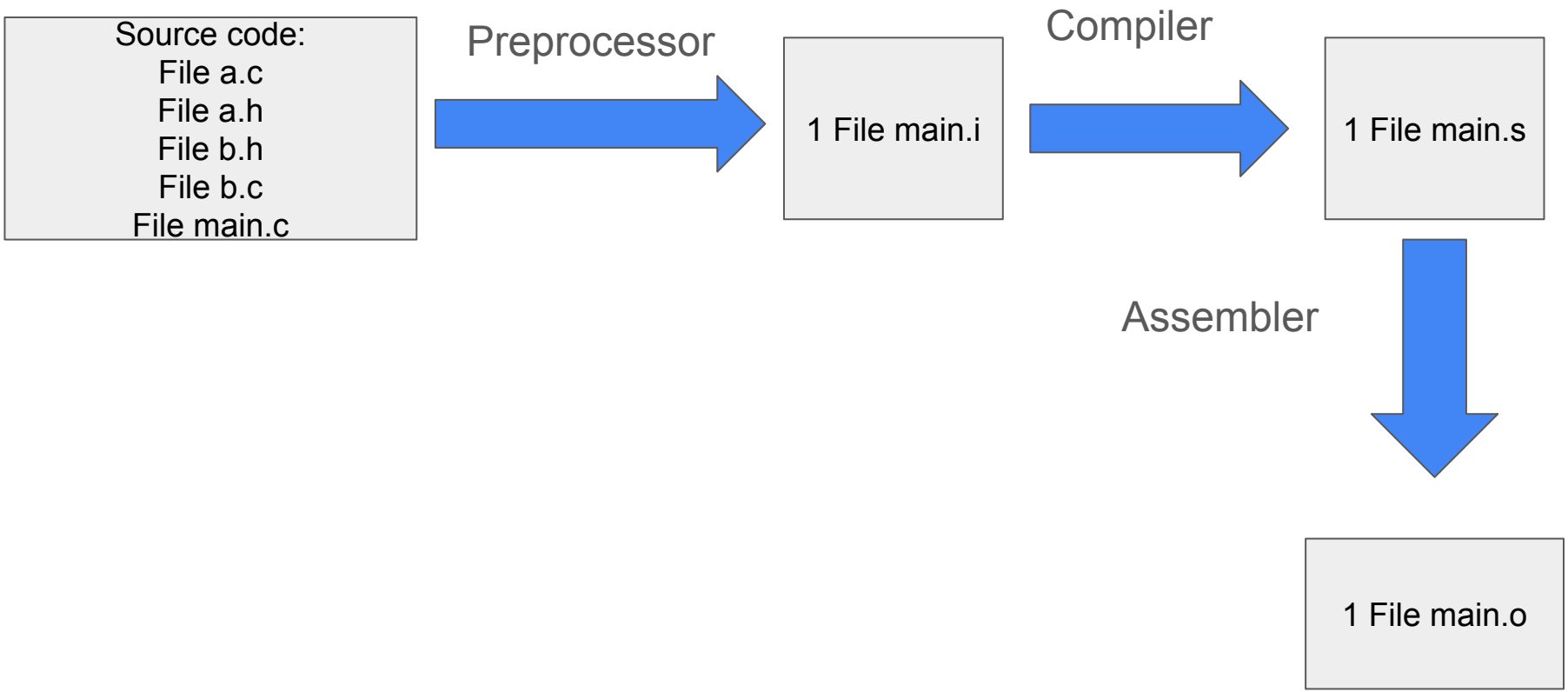
```
ADD AX, 10 ; Cộng giá trị 10 vào thanh ghi AX
```

```
MUL AX,AX,BX ; Nhân giá trị trong thanh ghi AX với
             giá trị trong thanh ghi BX
```

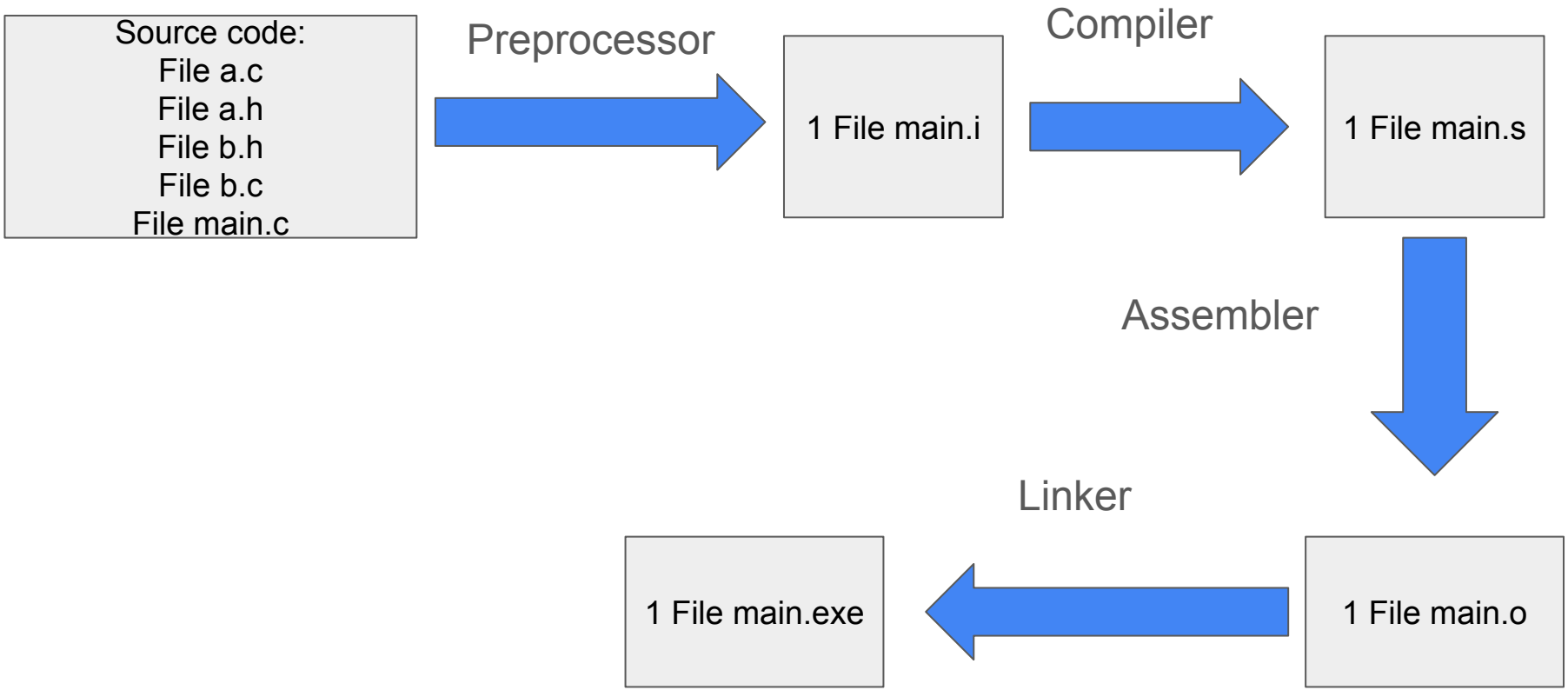
```
PUSH AX ; Đẩy giá trị từ thanh ghi AX vào ngăn xếp
PUSH 42 ; Đẩy giá trị 42 vào ngăn xếp
```

```
POP BX ; Lấy giá trị từ đỉnh của ngăn xếp và lưu vào
thanh ghi BX
```

Compiler


Source code:
File a.c
File a.h
File b.h
File b.c
File main.c
Preprocessor
1 File main.i
Compiler
1 File main.s
Assembler
1 File main.o

Compiler


Source code:
File a.c
File a.h
File b.h
File b.c
File main.c
Preprocessor
1 File main.i
Compiler
1 File main.s
Assembler
1 File main.o
Linker
1 File main.exe

Compiler

Bản chất của Run Code trong Visual Studio