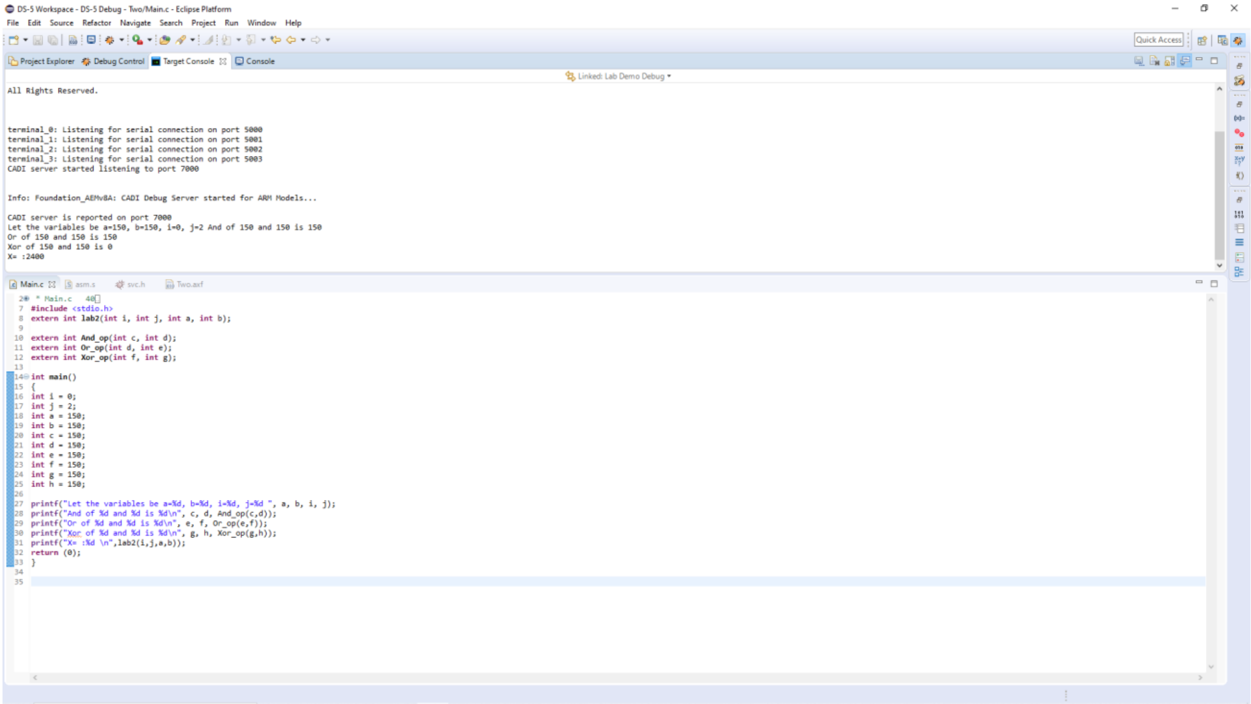
Implemented C and Assembly program along with screenshots (building and executing in debug window displaying the output along with project explorer tab)


DS-5 Workspace - DS-5 Debug - Two/Main.c - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer
Debug Control
Target Console
Console
Linked: Lab Demo Debug
Main.c
asm.s
svc.h
Two.axf

```
All Rights Reserved.



terminal_0: Listening for serial connection on port 5000
terminal_1: Listening for serial connection on port 5001
terminal_2: Listening for serial connection on port 5002
terminal_3: Listening for serial connection on port 5003
CADI server started listening to port 7000


Info: Foundation_AEMv8A: CADI Debug Server started for ARM Models...

CADI server is reported on port 7000
Let the variables be a=150, b=150, i=0, j=2 And of 150 and 150 is 150
Or of 150 and 150 is 150
Xor of 150 and 150 is 0
X= :2400
```

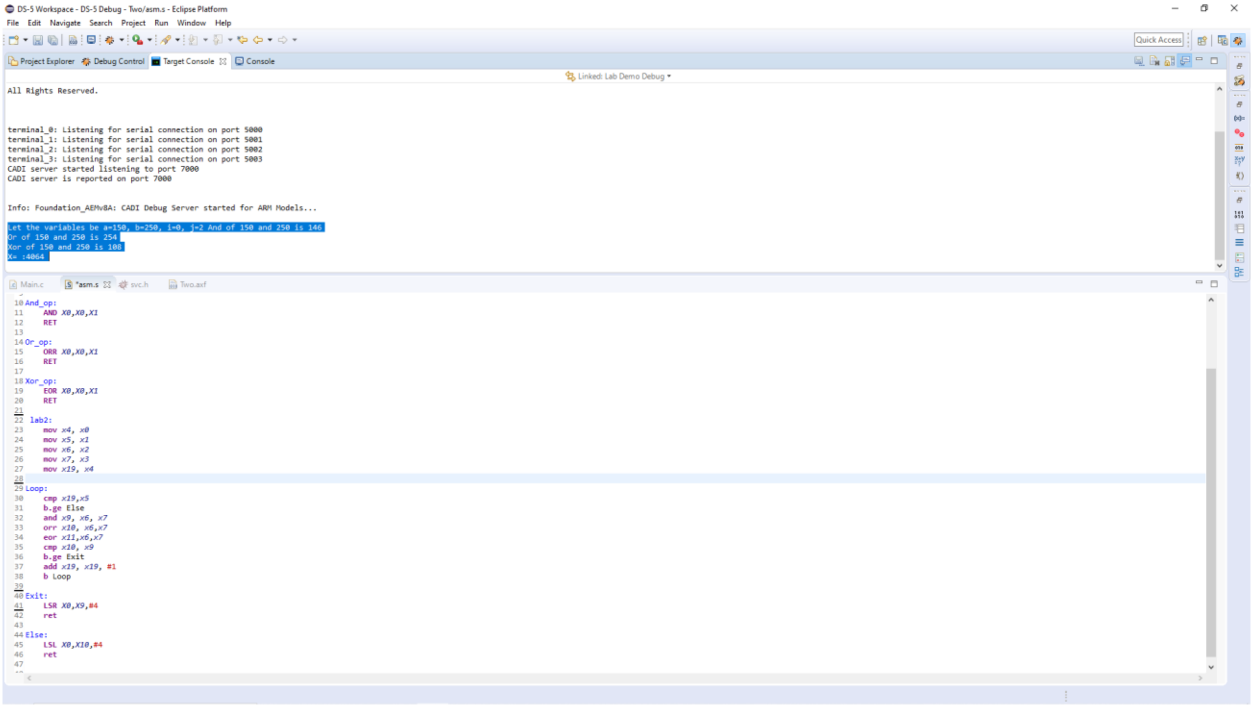
```c
 * Main.c  
#include <stdio.h>
extern int lab2(int i, int j, int a, int b);

extern int And_op(int c, int d);
extern int Or_op(int d, int e);
extern int Xor_op(int f, int g);

int main()
{
int i = 0;
int j = 2;
int a = 150;
int b = 150;
int c = 150;
int d = 150;
int e = 150;
int f = 150;
int g = 150;
int h = 150;

printf("Let the variables be a=%d, b=%d, i=%d, j=%d ", a, b, i, j);
printf("And of %d and %d is %d\n", c, d, And_op(c,d));
printf("Or of %d and %d is %d\n", e, f, Or_op(e,f));
printf("Xor of %d and %d is %d\n", g, h, Xor_op(g,h));
printf("X= :%d \n",lab2(i,j,a,b));
return (0);
}

```



```
All Rights Reserved.



terminal_0: Listening for serial connection on port 5000
terminal_1: Listening for serial connection on port 5001
terminal_2: Listening for serial connection on port 5002
terminal_3: Listening for serial connection on port 5003
CADI server started listening to port 7000
CADI server is reported on port 7000


Info: Foundation_AEMv8A: CADI Debug Server started for ARM Models...

Let the variables be a=150, b=250, i=0, j=2 And of 150 and 250 is 146
Or of 150 and 250 is 254
Xor of 150 and 250 is 108
X= :4064
```

```
And_op:
    AND X0,X0,X1
    RET

Or_op:
    ORR X0,X0,X1
    RET

Xor_op:
    EOR X0,X0,X1
    RET

 lab2:
    mov x4, x0
    mov x5, x1
    mov x6, x2
    mov x7, x3
    mov x19, x4

Loop:
    cmp x19,x5
    b.ge Else
    and x9, x6, x7
    orr x10, x6,x7
    eor x11,x6,x7
    cmp x10, x9
    b.ge Exit
    add x19, x19, #1
    b Loop

Exit:
    LSR X0,X9,#4
    ret

Else:
    LSL X0,X10,#4
    ret

```

Brief description of the following with respect to the c code and assembly implemented:

A. External functions used in C code

The code was used to establish the variables i, j, a, and b as well as used to print out of the results for the program. C-H are just a repeat of the values for a and b since I wanted to display the value of the AND, OR, and XOR values. Initially, I attempted to use just the a and b variables, however the value of operations was completely wrong until I gave the operations their own variables.

B. All the loop functions in the assembly code

The loop functions begin by comparing the i to j. If i is not greater than or equal to j, it will do the AND, OR, and XOR functions. Once they are complete, the loop will compare AND and OR to see if they have the same value. In this case, if they do not have the same value than it will add one to i. The loop will continue until i and j have the same value or, if AND and OR have the same value.

C. How does the procedure call and branches work in this program?

The branches work in two different ways for the program, the exit of the loop and the else statement. The first branch will exit the loop when AND and OR have the same value and shift the AND value 4 to the right. If they do not have the same values than the program will continue the loop until i is greater than j. When that happens, the second branch will shift the OR value 4 to the left.

In brief mention the different variables and registers that are used by the program by making use of the debug control window and the .axf file. Also provide the start address and end address of each function (init, start, main, and the address of all the implemented functions).

The program starts by doing the AND, OR, AND XOR functions with the variables c-h. Initially, I only had A-B to each function, however this made the program given incorrect values for the operation, so I separated them and put the functions above the main code to prevent the functions from interfering with the results. Variables i,j,a, and b were each moved to a new register, so that the c code can print the initial values. Registers X6 and X7 were variables a and b while registers X19 and X5 were the i and j variables. Registers X9, X10, and X11 were the function values of a and b. X0 gave the final value of each function in the program.

/*Main.c*/

```c
#include <stdio.h>
extern int lab2(int i, int j, int a, int b);

extern int And_op(int c, int d);
extern int Or_op(int d, int e);
extern int Xor_op(int f, int g);

int main()
{
int i = 0;
int j = 2;
int a = 150;
int b = 250;
int c = 150;
int d = 250;
int e = 150;
int f = 250;
int g = 150;
int h = 250;

printf("Let the variables be a=%d, b=%d, i=%d, j=%d  ", a, b, i, j);
printf("And of %d and %d is %d\n", c, d, And_op(c,d));
printf("Or of %d and %d is %d\n", e, f, Or_op(e,f));
printf("Xor of %d and %d is %d\n", g, h, Xor_op(g,h));
printf("X= :%d \n",lab2(i,j,a,b));
return (0);
}
```

/*asm.S*/

```
.global lab2, And_op, Or_op, Xor_op

And_op:
        AND X0,X0,X1
        RET

Or_op:
        ORR X0,X0,X1
        RET

Xor_op:
        EOR X0,X0,X1
        RET

 lab2:
        mov x4, x0
        mov x5, x1
        mov x6, x2
        mov x7, x3
        mov x19, x4

Loop:
        cmp x19,x5
        b.ge Else
        and x9, x6, x7
        orr x10, x6,x7
        eor x11,x6,x7
        cmp x10, x9
        b.ge Exit
        add x19, x19, #1
        b Loop

Exit:
        LSR X0,X9,#4
        ret

Else:
        LSL X0,X10,#4
        ret
```