Microprocessor Programming

Dr. Tin Ni Ni Kyaw

Ph. D (Kumamoto University, Japan)

Associate Professor

Department of Computer Engineering and Information Technology

Yangon Technological University

Yangon, Myanmar

Microprocessor Programming

Lecture 2

Assembly Language: Input and Output

Contents

- The First Program
- Output
  - Single String Output
  - Multiple Strings Output
  - Integer Output
  - String + Integer Output
  - Integer + String + Integer Output
- User Input
  - Integer Input
  - Character Input
- Complete Program
- Summary
- Practical Works
- Assignments

The First Program

- First, we will start from a simple C program called “Hello World” program.

```
#include <stdio.h>
int main()
{
        printf("Hello World!\n");
        return 0;
}
```

The First Program (Cont.)

- Assembly code for Hello World program is as follows:

```
.386
.model flat, c
.stack 100h
printf PROTO arg1:Ptr Byte
.data
msg1 byte "Hello World!",0Ah,0
.code
main proc
INVOKE printf, ADDR msg1
ret
main endp
end
```

The First Program (Cont.)

- In assembly language, printf is used to display the output message.
- The PROTO directive indicates a prototype for the function printf.
- The arg1:Ptr Byte is a parameter and it specifies that the argument of the printf will be a pointer to a string of bytes.
- The actual message to be output is in the .data section as msg1 byte "Hello World".
- The string to be displayed must be written in double quotes.

The First Program (Cont.)

- The 0Ah is a hexadecimal code for a new line, such as \n in C.
- The 0 is the code to terminate a string used with output.
- The INVOKE directive is used to call the printf function.
- The address of the string to be output which is msg1 is given by the argument ADDR msg1 in the INVOKE.

Single String Output

- To output more than one argument, the original C program could be rewritten as follows:

```
#include <stdio.h>
int main()
{
        printf("%s\n","Hello World!");
        return 0;
}
```

- The %s indicates that there is a string in the first argument following the current formatting argument.

Single String Output (Cont.)

- Assembly code is as follows:

```
.386
.model flat, c
.stack 100h
printf PROTO arg1:Ptr Byte, printlist:VARARG
.data
msg1fmt byte "%s",0Ah,0
msg1 byte "Hello World!",0Ah,0
.code
main proc
INVOKE printf, ADDR msg1fmt, ADDR msg1
ret
main endp
end
```

Single String Output (Cont.)

- An additional argument printlist: VARARG in the PROTO statement indicates that there will be a variable number of arguments in the output.
- The msg1fmt indicates the data type that will be output.
- Lastly, the first ADDR in the INVOKE directive references the string format (msg1fmt) and the latter references the string to be output (msg1).

Multiple Strings Output

```
.386
.model flat, c
.stack 100h
printf PROTO arg1:Ptr Byte, printlist:VARARG
.data
msg1fmt byte "%s",0Ah,0
msg1 byte "Hello!",0Ah,0
msg2fmt byte "%s",0Ah,0
msg2 byte "Welcome to Microprocessor Programming Class",0Ah,0
msg3fmt byte "%s",0Ah,0
msg3 byte "This program is how to display multiple strings",0Ah,0
msg4fmt byte "%s",0Ah,0
msg4 byte "Thank you",0Ah,0
.code
main proc
INVOKE printf, ADDR msg1fmt, ADDR msg1
INVOKE printf, ADDR msg2fmt, ADDR msg2
INVOKE printf, ADDR msg3fmt, ADDR msg3
INVOKE printf, ADDR msg4fmt, ADDR msg4
ret
main endp
end
```

Integer Output

- In the following C program, the integer 5 is output.

```
#include <stdio.h>
int main()
{
        int number;
        number = 5;
        printf("%d\n",number);
        return 0;
}
```

- Here, %d indicates that there is an integer in the first argument following the current formatting argument.

Integer Output (Cont.)

- The corresponding MASM code is as follows:

```
.386
.model flat, c
.stack 100h
printf PROTO arg1:Ptr Byte, printlist:VARARG
.data
msg1fmt byte "%d",0Ah,0
number sdword ?
.code
main proc
mov number,5
INVOKE printf, ADDR msg1fmt, number
ret
main endp
end
```

Integer Output (Cont.)

- In the MASM code, the %d stands for the integer variable named number and it is declared as a signed double word in the data section.
- In the code segment, 5 is assigned to number and the variable number has been added as an argument to the INVOKE directive.
- As msg1fmt and msg1 are pointers to the strings, they need ADDR but number is a simple integer variable and so, ADDR is not needed.

String + Integer Output

- To display the output of the combination of a string and an integer, the C program is as follow.

```
#include <stdio.h>
int main()
{
	int number;
	number = 5;
	printf("%s%d\n","The number is: ",number);
	return 0;
}
```

String + Integer Output (Cont.)

- The corresponding MASM code is as follows:

```
.386
.model flat, c
.stack 100h
printf PROTO arg1:Ptr Byte, printlist:VARARG
.data
msg1fmt byte "%s%d",0Ah,0
msg1 byte "The number is: ",0
number sdword ?
.code
main proc
mov number,5
INVOKE printf, ADDR msg1fmt, ADDR msg1, number
ret
main endp
end
```

Integer + String + Integer Output

- To display the output of the combination of string and integers, the C program is as follow.

```
#include <stdio.h>
int main()
{
	int num1 = 5, num2 = 7;
	printf("\n%d%s%d\n\n",num1," is not equal to ",num2);
	return 0;
}
```

Integer + String + Integer Output (Cont.)

- The corresponding MASM code is as follows:

```
.386
.model flat, c
.stack 100h
printf PROTO arg1:Ptr Byte, printlist:VARARG
.data
msg1fmt byte 0Ah,"%d%s%d",0Ah,0Ah,0
msg1 byte " is not equal to ",0
num1 sdword 5
num2 sdword 7
.code
main proc
INVOKE printf, ADDR msg1fmt, num1, ADDR msg1, num2
ret
main endp
end
```

User Input

- scanf is used for user input.
- In C, the ampersand (&) passes the address of user input number to scanf for reading the value from the keyboard.
- For the case of assembly, ADDR is needed to precede the variable number in the invoking of scanf.
- Because, ADDR serves the same function as the ampersand (&) in C.

Integer Input

- To prompt the user for the type of input needed as shown in the C program below.

```
#include <stdio.h>
int main()
{
	int number;
	printf("\n%s","Enter an integer: ");
	scanf("%d",&number);
	printf("\n%s%d\n\n","The integer is: ",number);
	return 0;
}
```

Integer Input (Cont.)

- The corresponding assembly code is given as follows:

```
.386
.model flat, c
.stack 100h
printf PROTO arg1:Ptr Byte, printlist:VARARG
scanf PROTO arg2:Ptr Byte, inputlist:VARARG
.data
in1fmt byte "%d",0
msg0fmt byte 0Ah,"%s",0
msg1fmt byte 0Ah,"%s%d",0Ah,0Ah,0
msg0 byte "Enter an integer: ",0
msg1 byte "The integer is: ",0
number sdword ?
.code
main proc
INVOKE printf, ADDR msg0fmt, ADDR msg0
INVOKE scanf, ADDR in1fmt, ADDR number
INVOKE printf, ADDR msg1fmt, ADDR msg1, number
ret
main endp
end
```

Note: ADDR is not necessary for integer output.

Character Input

- The assembly code for character input is as follows:

```
.386
.model flat, c
.stack 100h
printf PROTO arg1:Ptr Byte, printlist:VARARG
scanf PROTO arg2:Ptr Byte, inputlist:VARARG
.data
in1fmt byte "%s",0
msg0fmt byte 0Ah,"%s",0
msg1fmt byte 0Ah,"%s%s",0Ah,0Ah,0
msg0 byte "Enter a character: ",0
msg1 byte "The character you entered is: ",0
character byte ?
.code
main proc
INVOKE printf, ADDR msg0fmt, ADDR msg0
INVOKE scanf, ADDR in1fmt, ADDR character
INVOKE printf, ADDR msg1fmt, ADDR msg1, ADDR character
ret
main endp
end
```

Note: ADDR is necessary for character/string output.

Complete Program: Using Input, Data Transfer, and Output

- The following program prompts for and inputs an integer into num1, copies it to num2, and then outputs the contents of num2.

```
#include <stdio.h>
int main(){
	int num1, num2;
	printf("\n%s","Enter an integer for num1: ");
	scanf("%d",&num1);
	num2=num1;
	printf("\n%s%d\n\n","The integer in num2 is: ",num2);
	return 0;
}
```

Complete Program: Using Input, Data Transfer, and Output (Cont.)

- This program is implemented in assembly language as follows:

```
            .386
            .model flat, c
            .stack 100h
printf      PROTO arg1:Ptr Byte, printlist:VARARG
scanf       PROTO arg2:Ptr Byte, inputlist:VARARG
            .data
in1fmt      byte "%d",0
msg0fmt     byte 0Ah,"%s",0
msg1fmt     byte 0Ah,"%s%d",0Ah,0Ah,0
msg0        byte "Enter an integer for num1: ",0
msg1        byte "The integer in num2 is: ",0
num1        sdword ?        ; first number
num2        sdword ?        ; second number
            .code
main        proc
            INVOKE printf, ADDR msg0fmt, ADDR msg0
            INVOKE scanf, ADDR in1fmt, ADDR num1
            mov eax,num1    ; load eax with the content of num1
            mov num2,eax    ; store the contents of eax in num2
            INVOKE printf, ADDR msg1fmt, ADDR msg1, num2
            ret
main        endp
            end
```

Summary

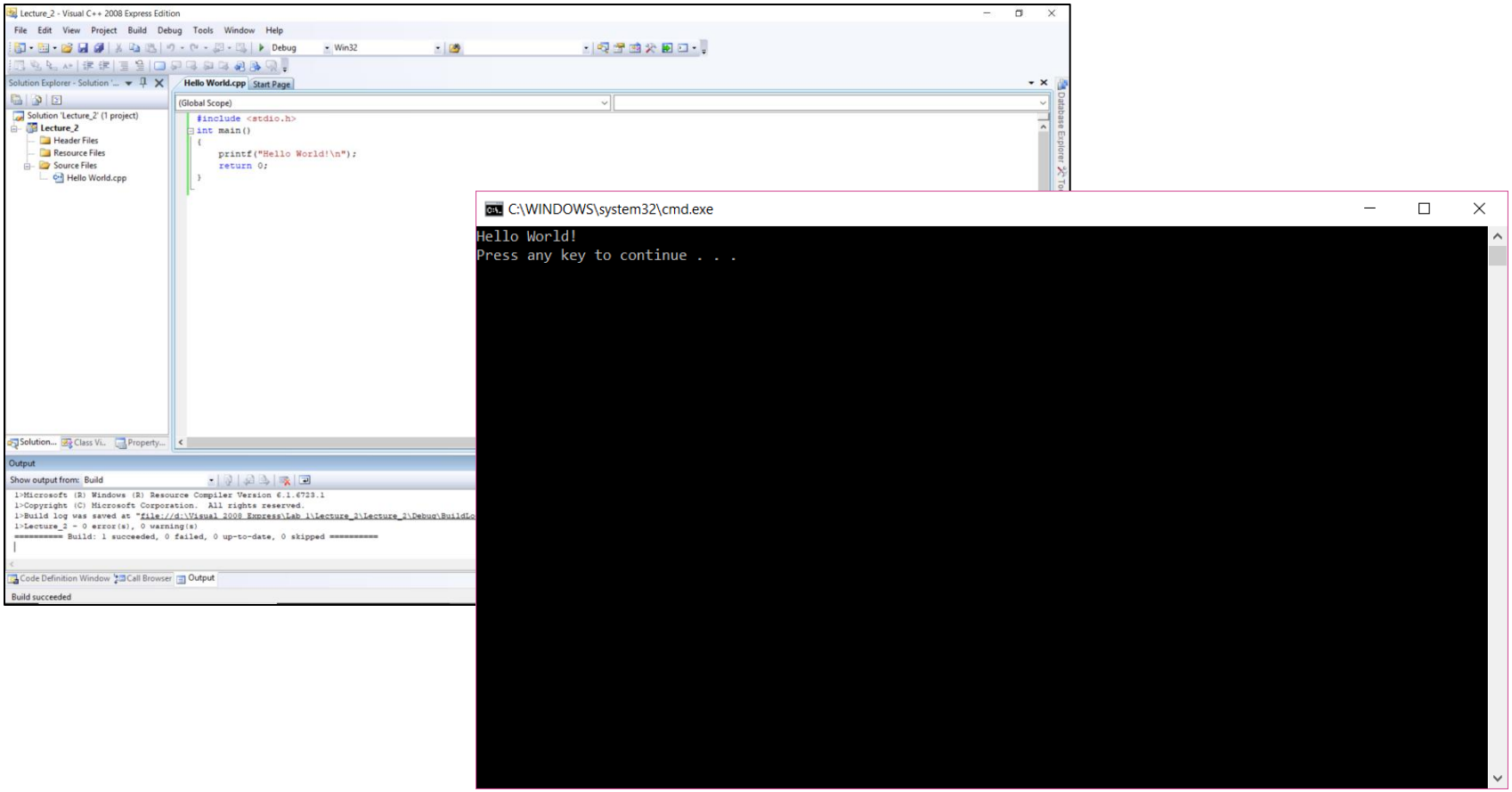
- Use the PROTO and INVOKE directives to implement the C functions scanf and printf.
- The 0Ah in assembly language is like a \n in C.
- Be sure to terminate a string for output in assembly language with a 0.
- For input, an ADDR is included for all variables.
- For output, an ADDR is included for only strings.

Microprocessor Programming

Practical Works:
Implementing Input and Output

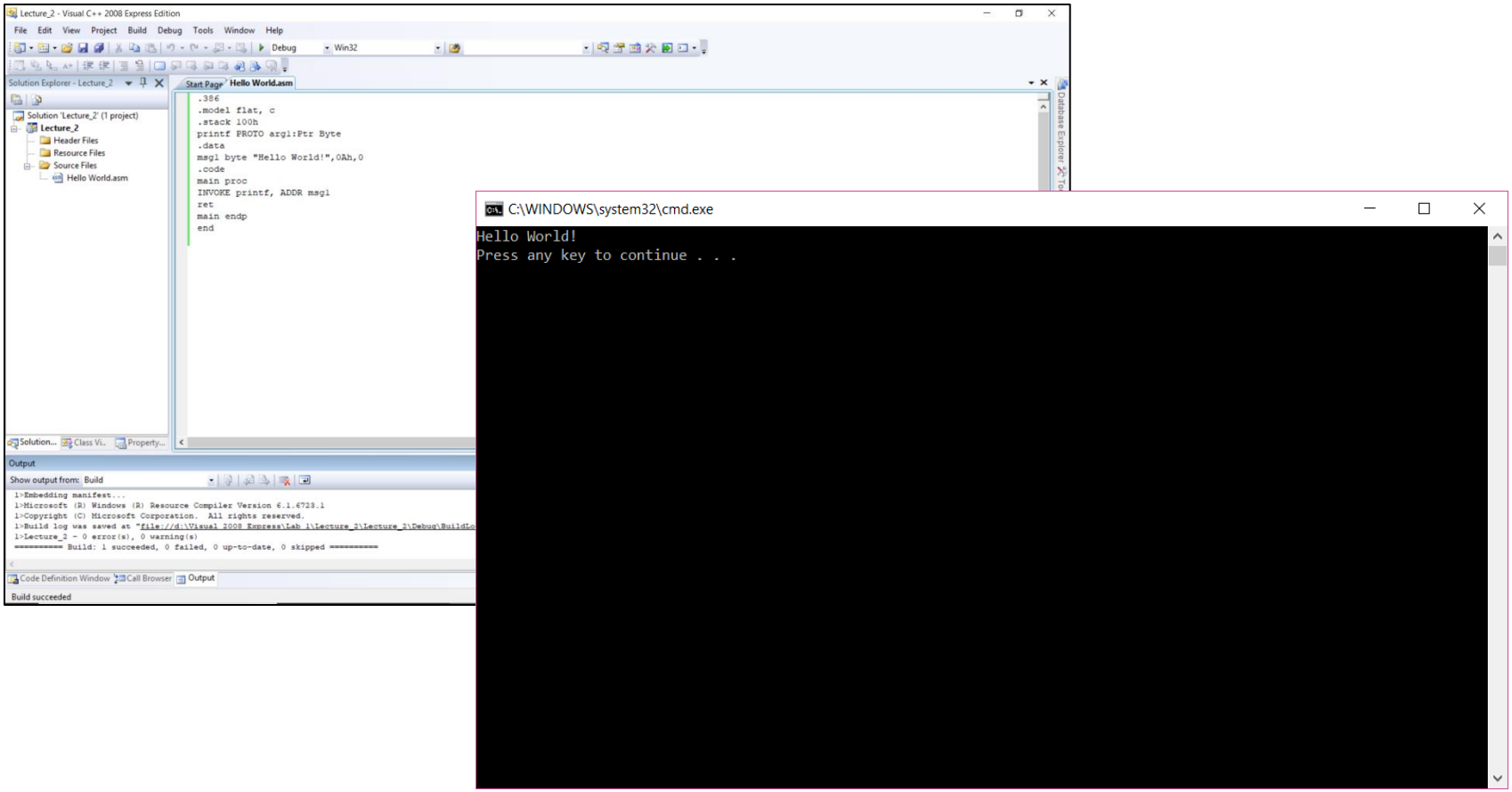
The First Program

- This Implementation is the Hello World program in C.



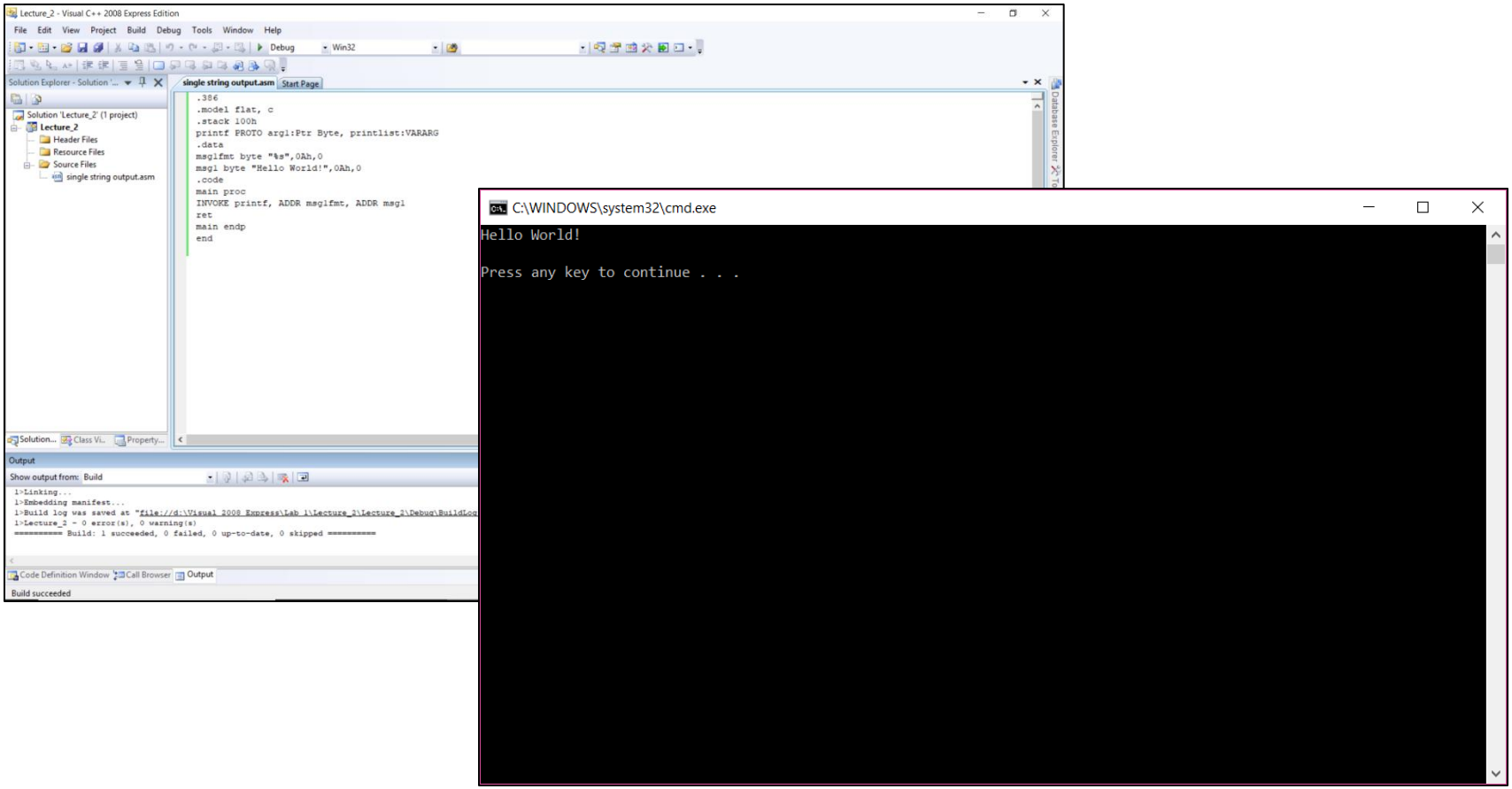
The First Program (Cont.)

- This Implementation is the Hello World program in MASM.



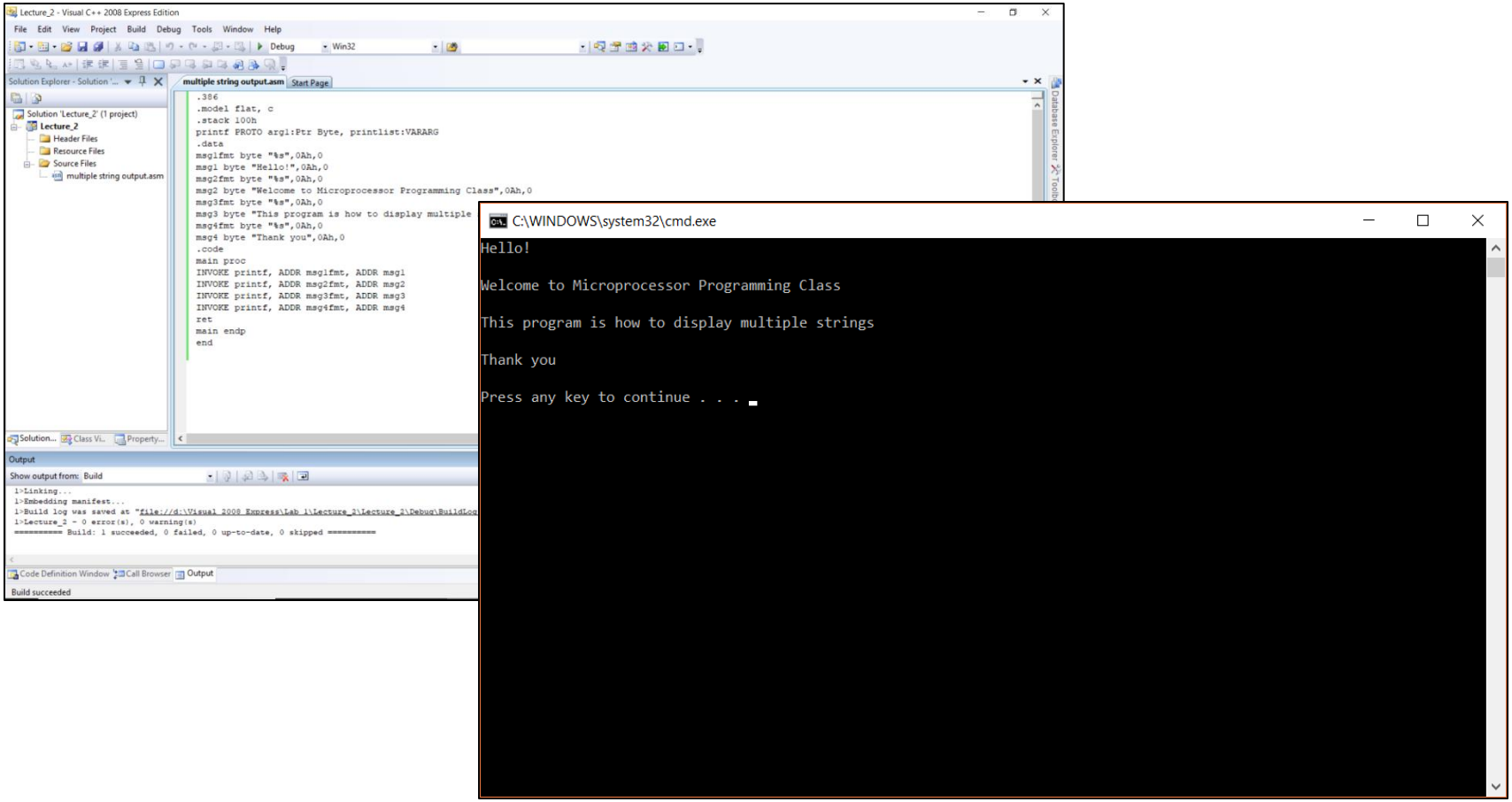
Single String Output

- This is the implementation of single string output in MASM.



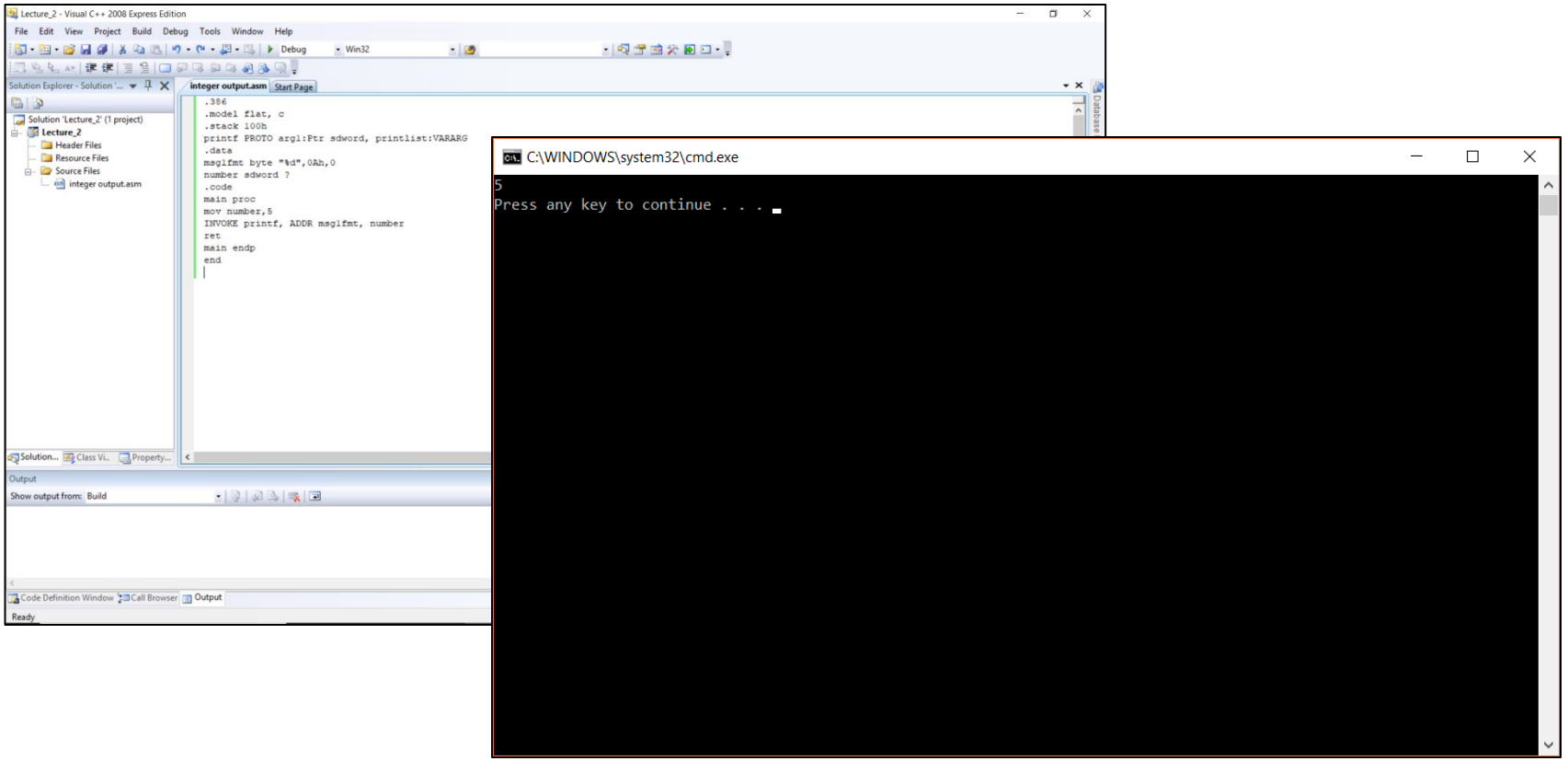
Multiple Strings Output

- This is the implementation of multiple strings output in MASM.



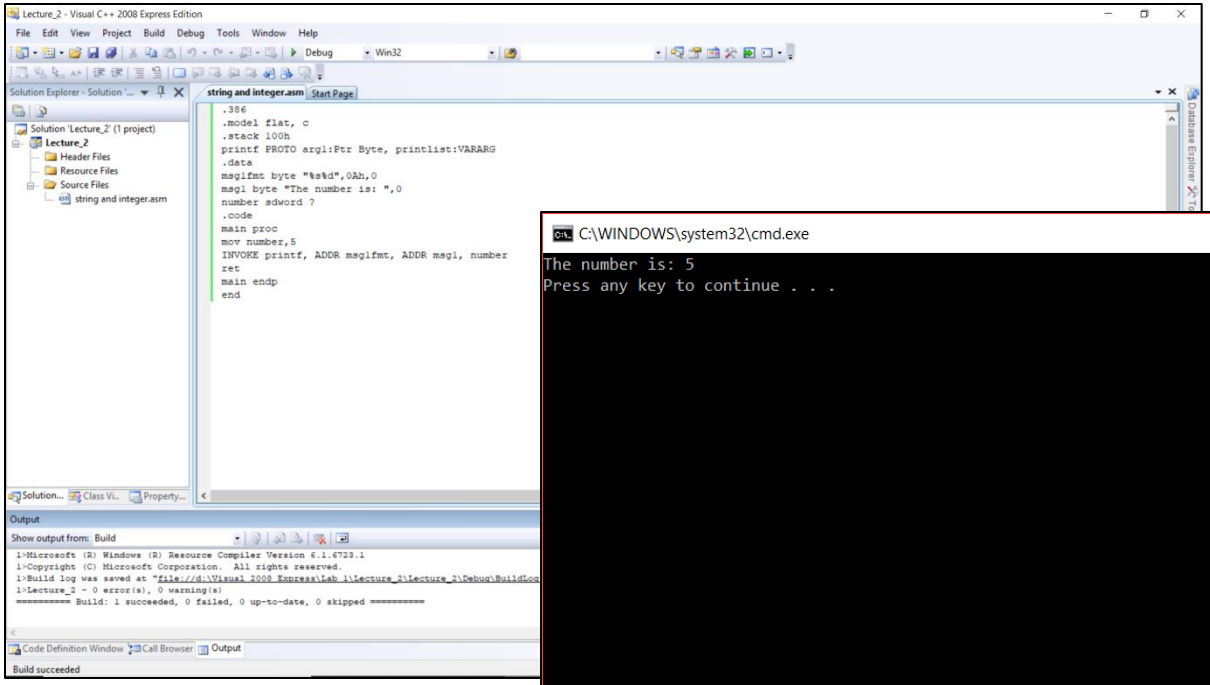
Integer Output

- This is the implementation of integer output in MASM.



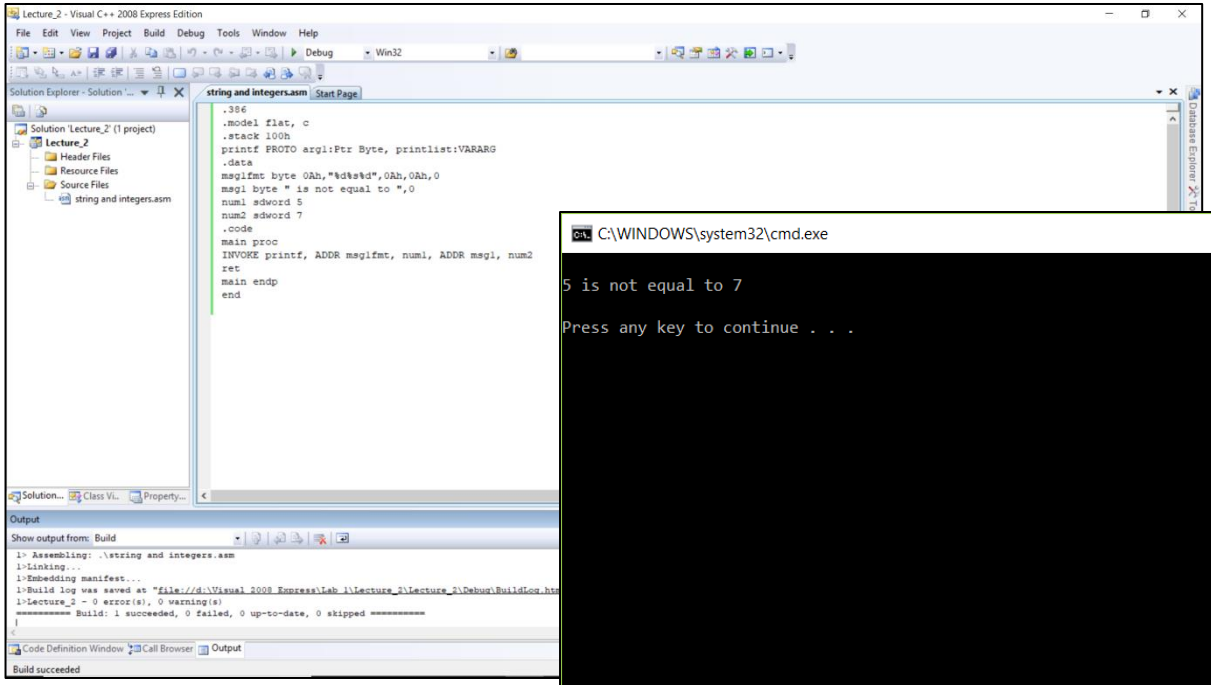
String + Integer Output

- This is the implementation of one string and one integer output in MASM.



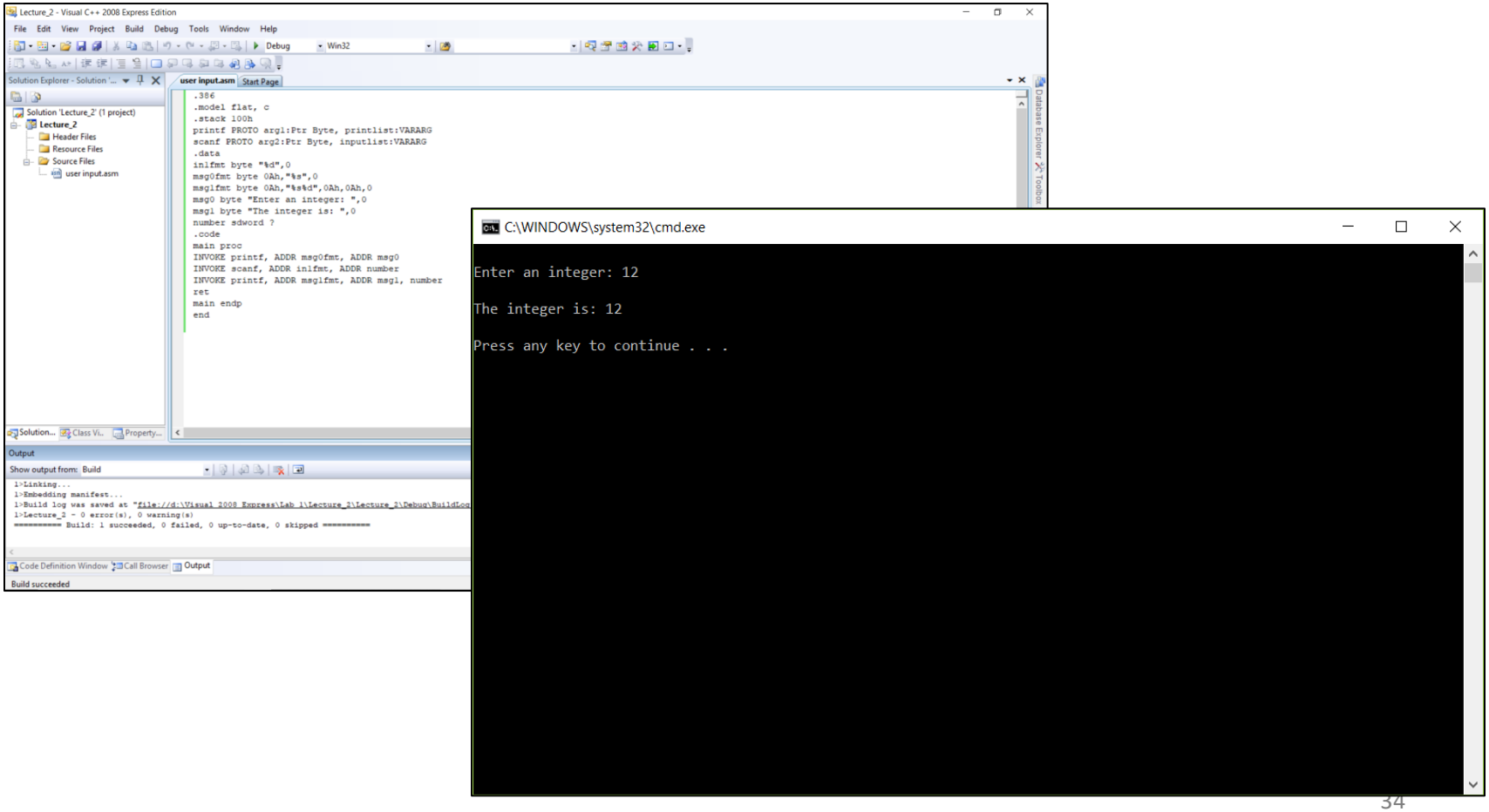
Integer + String + Integer Output

- This is the implementation of one string and two integers output in MASM.



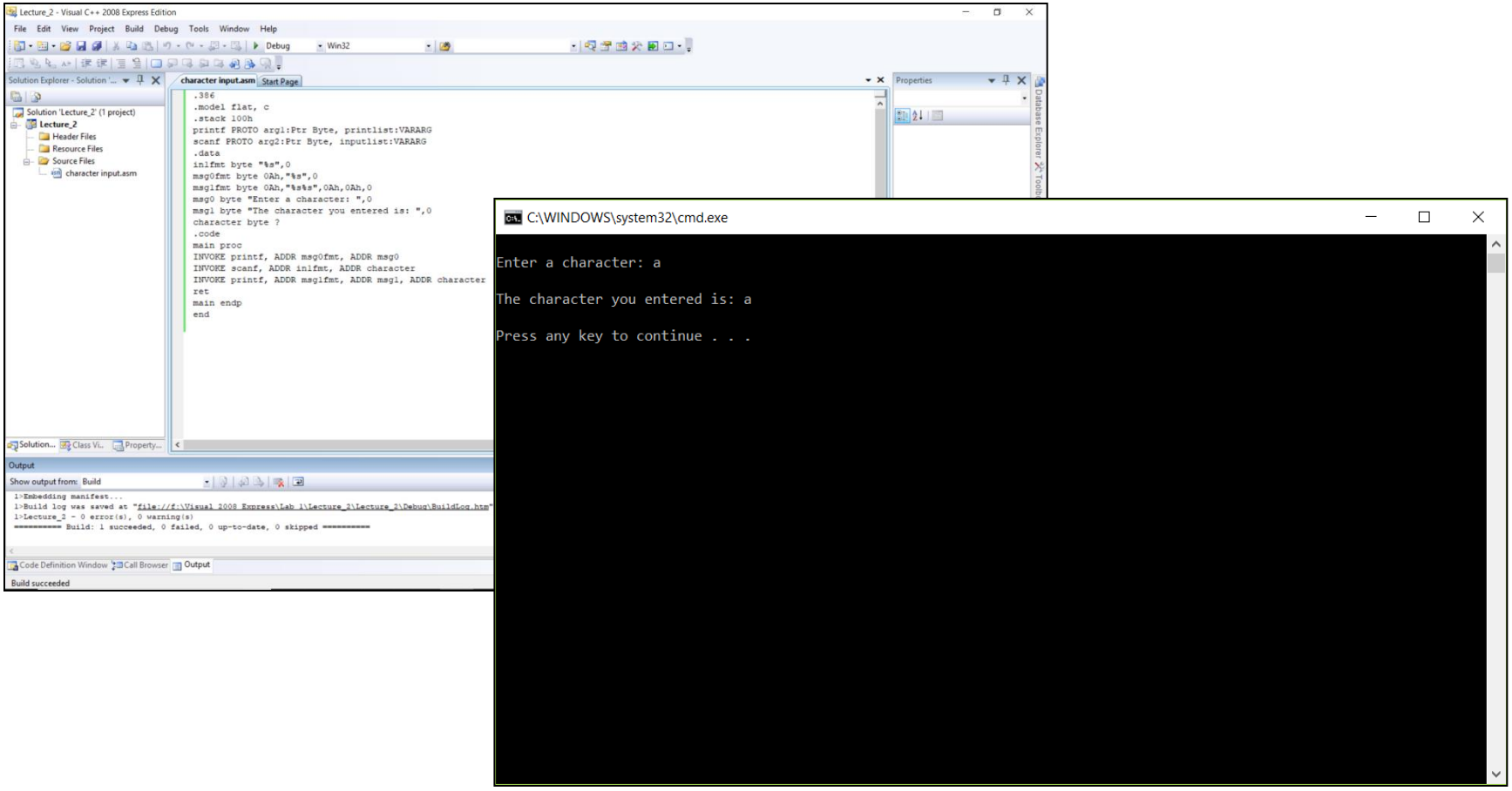
Integer Input

- This is the implementation of getting user's integer input in MASM.



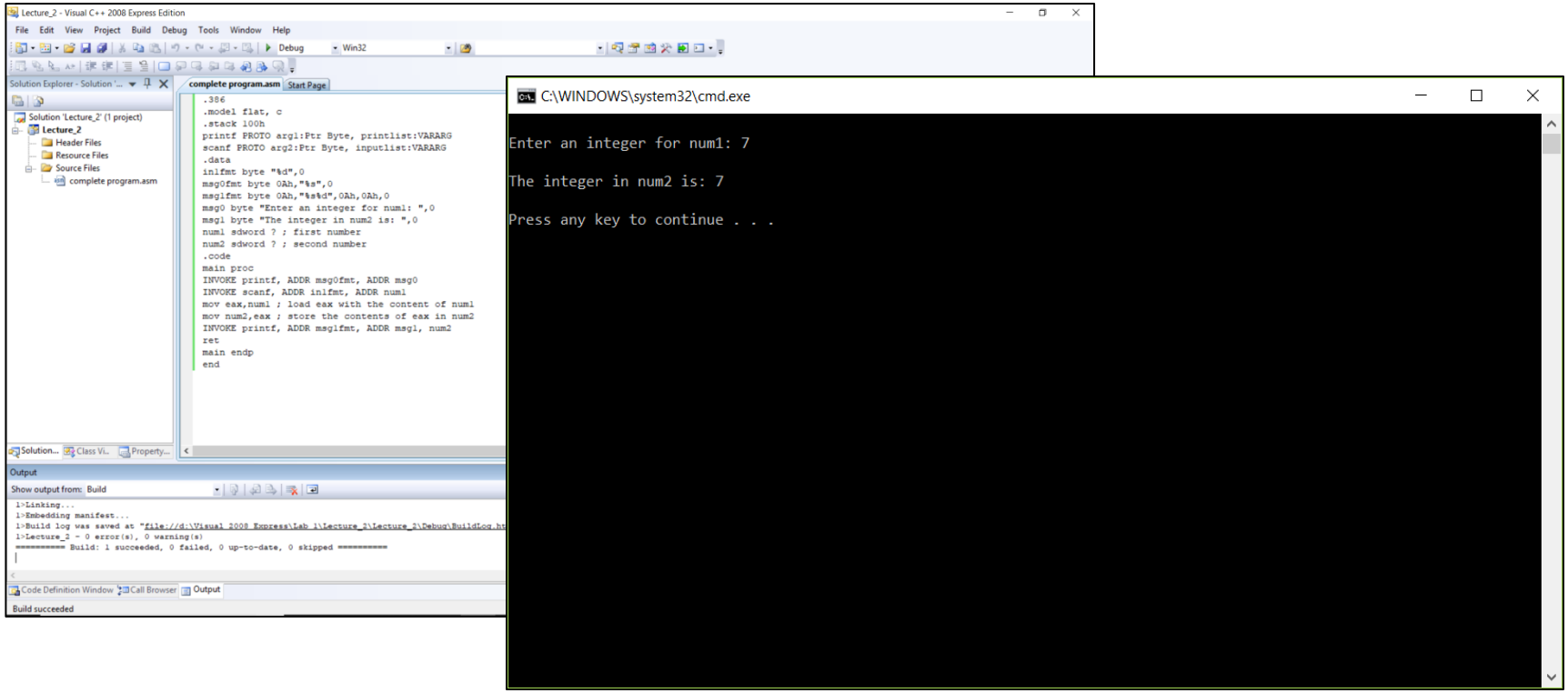
Character Input

- This is the implementation of getting user's character input in MASM.



Complete Program: Using Input, Data Transfer, and Output

- This is the implementation of how to write a complete assembly program including getting user input, data movement and displaying the output.



Microprocessor Programming

Practical Assignments (Instructions)

Assignment 1

- Given the following MASM program, what will be output to the screen?

```
            .386
            .model flat, c
            .stack 100h
printf      PROTO arg1:Ptr Byte, printlist:VARARG
            .data
msg1fmt     byte "%s%d",0
msg2fmt     byte "%s%d",0Ah,0Ah,0Ah,0
msg3fmt     byte "%s%d",0Ah,0
msg1        byte "x = ",0
msg2        byte " y = ",0
msg3        byte "z=",0
num1        sdword 1
num2        sdword 2
num3        sdword 3
            .code
main        proc
            INVOKE printf, ADDR msg1fmt, ADDR msg1, num1
            INVOKE printf, ADDR msg2fmt, ADDR msg2, num2
            INVOKE printf, ADDR msg3fmt, ADDR msg3, num3
            ret
main        endp
            end
```

Assignment 2

- Given the following MASM program, what will be output to the screen?

```
            .386
            .model flat, c
            .stack 100h
printf      PROTO arg1:Ptr Byte, printlist:VARARG
            .data
msg1fmt     byte 0Ah,"%s%d%s%d%s",0Ah,0
msg2fmt     byte 0Ah,"%s%d",0Ah,0Ah,0
msg11       byte "The first number is ",0
msg12       byte ", but the second number is ",0
msg13       byte ",",0
msg2        byte "while the third number is ",0
num1        sdword 5
num2        sdword 7
num3        sdword 11
            .code
main        proc
            INVOKE printf, ADDR msg1fmt, ADDR msg11, num1,
                   ADDR msg12, num2, ADDR msg13
            INVOKE printf, ADDR msg2fmt, ADDR msg2, num3
            ret
main        endp
            end
```

Assignment 3

- Implement the following C program in MASM. Be sure to use proper spacing on all output.

```
#include <stdio.h>
int main(){
    int x, y, z;
    x = 1;
    y = 2;
    z = 3;
    printf(%\n%d%s%d%s%d\n\n", x, " + ", y, " = ", z);
    return 0;
}
```

Assignment 4

- Implement the following C program in MASM. Be sure to use proper spacing on all output.

```
#include <stdio.h>
int main(){
    int num1, num2;
    printf("\n%s","Enter a value for num1: ");
    scanf("%d",&num1);
    printf("\n%s","Enter a value for num2: ");
    scanf("%d",&num2);
    printf("\n%s\n\n","num1    num2");
    printf("%s%d%s%d\n\n"," ",num1,"       ",num2);
    return 0;
}
```

Assignment 5

- Given the following input and output, write both the C and assembly code necessary to make it look exactly as below. Pay careful attention to spacing and the blank lines:

Input and Output

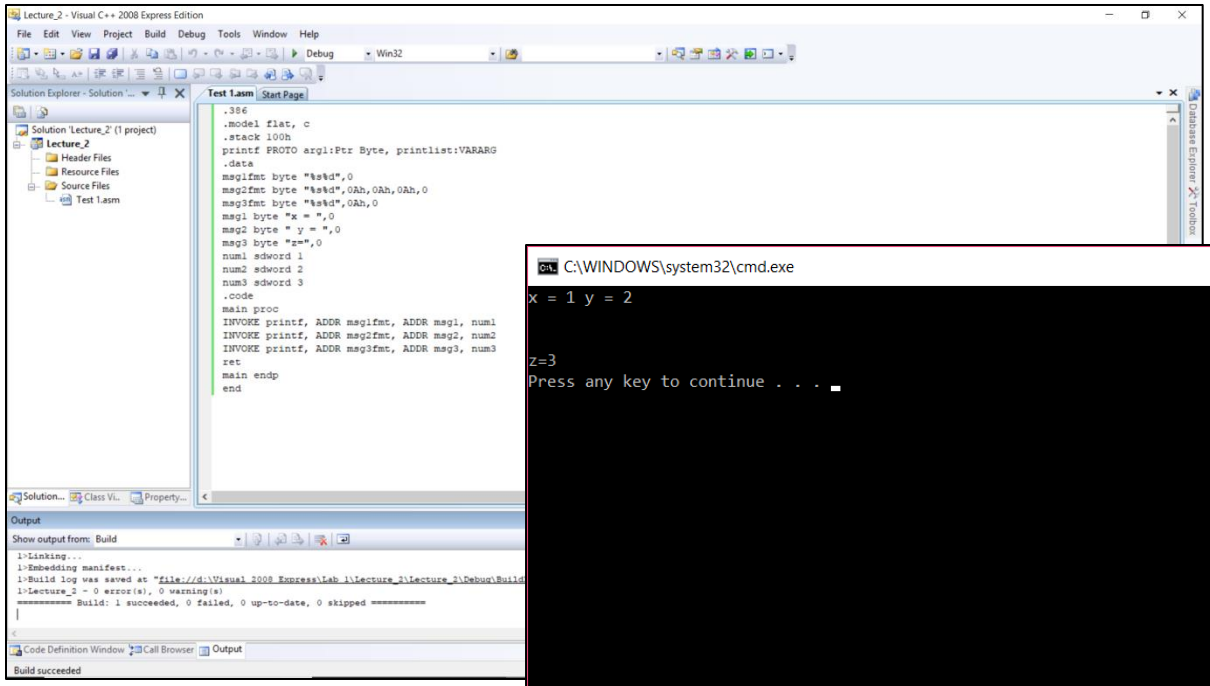
```
Enter a number: 1
Enter a larger number: 3
Enter an even larger number: 5

1 < 3 < 5

5 > 3 > 1
```

Microprocessor Programming

Practical Assignments (Report)

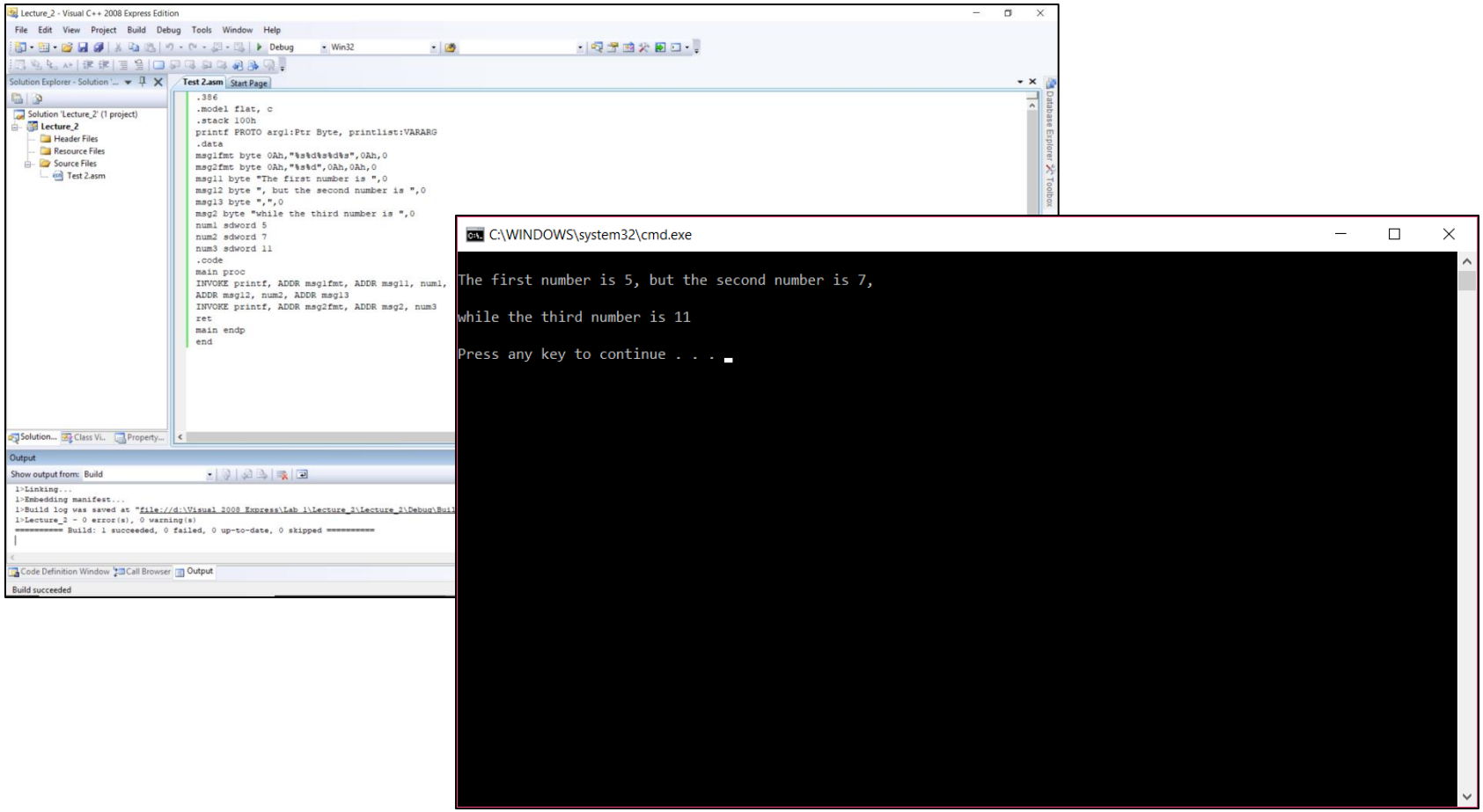
Assignment 1

- The implementation of assembly program and its output are as follows.



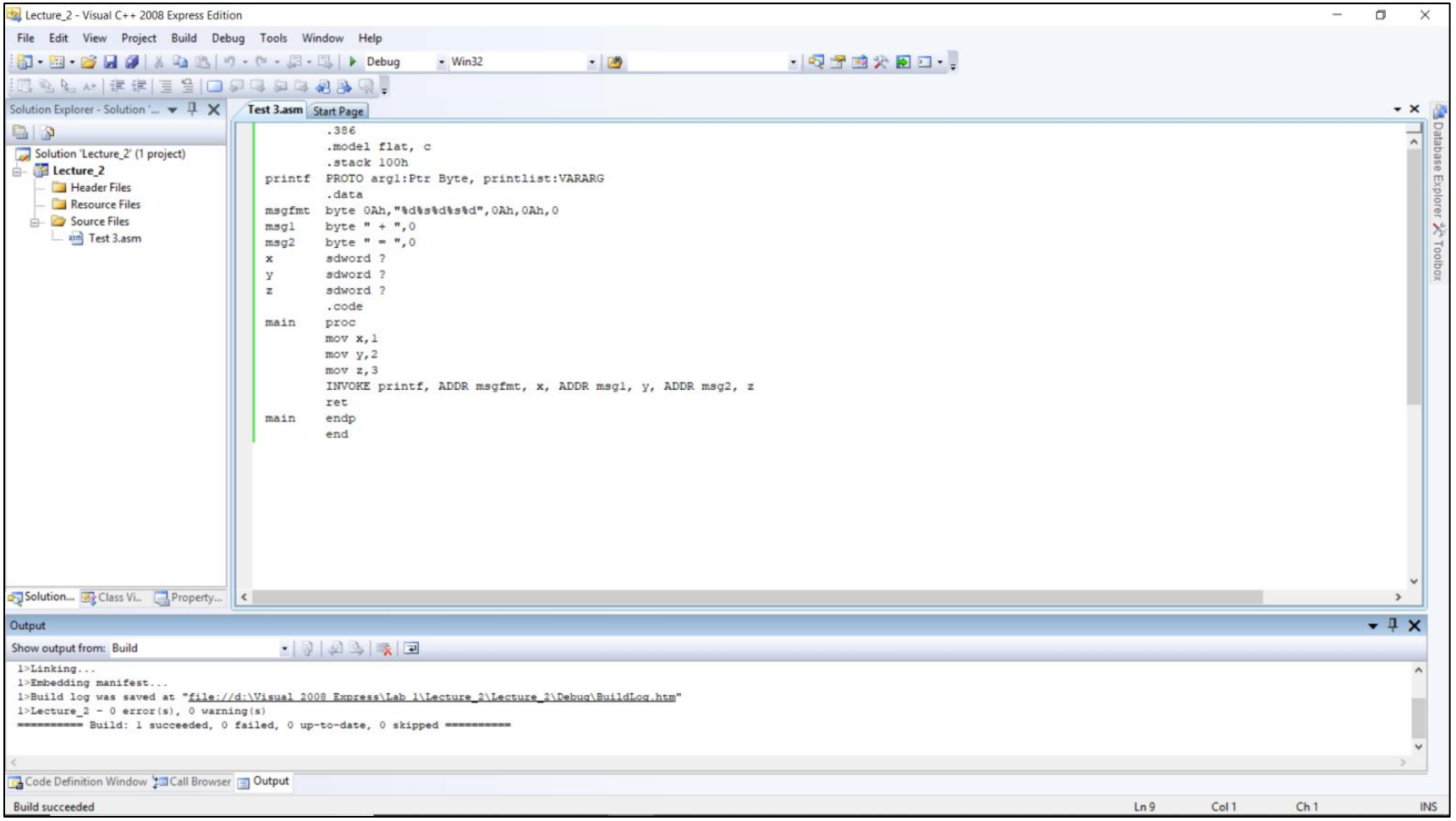
Assignment 2

- The implementation of assembly program and its output are shown in the following figures.



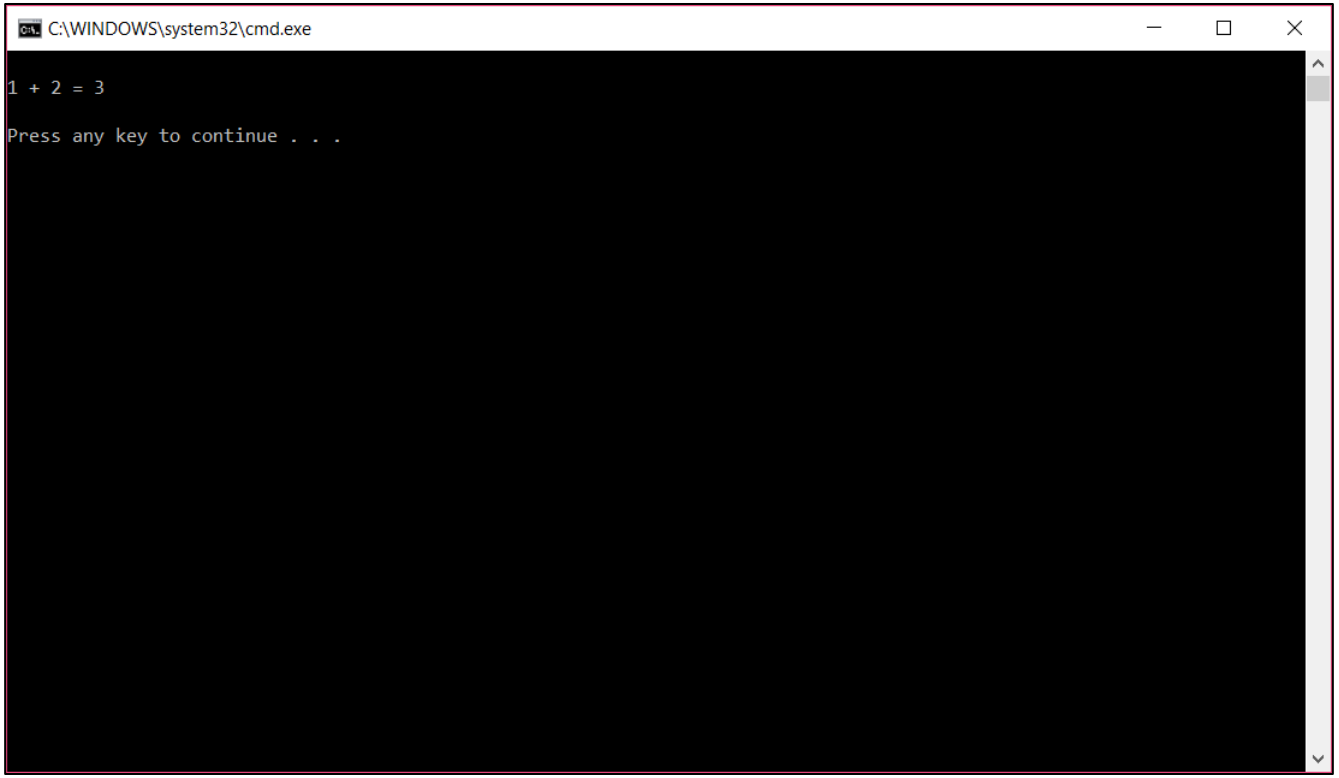
Assignment 3

- The implementation of assembly program is as follow.



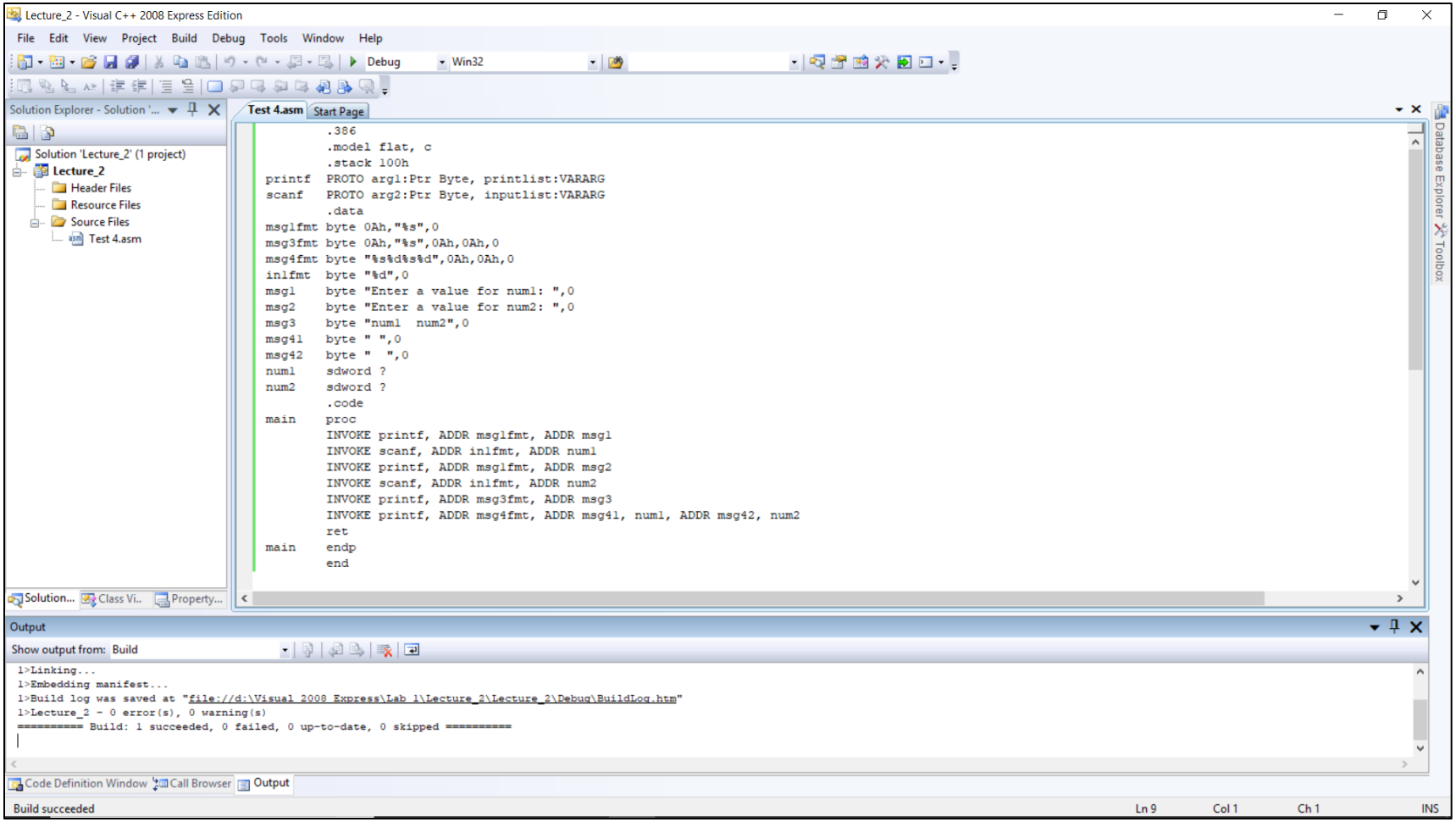
Assignment 3 (Cont.)

- The output for the above assembly program is as follow.



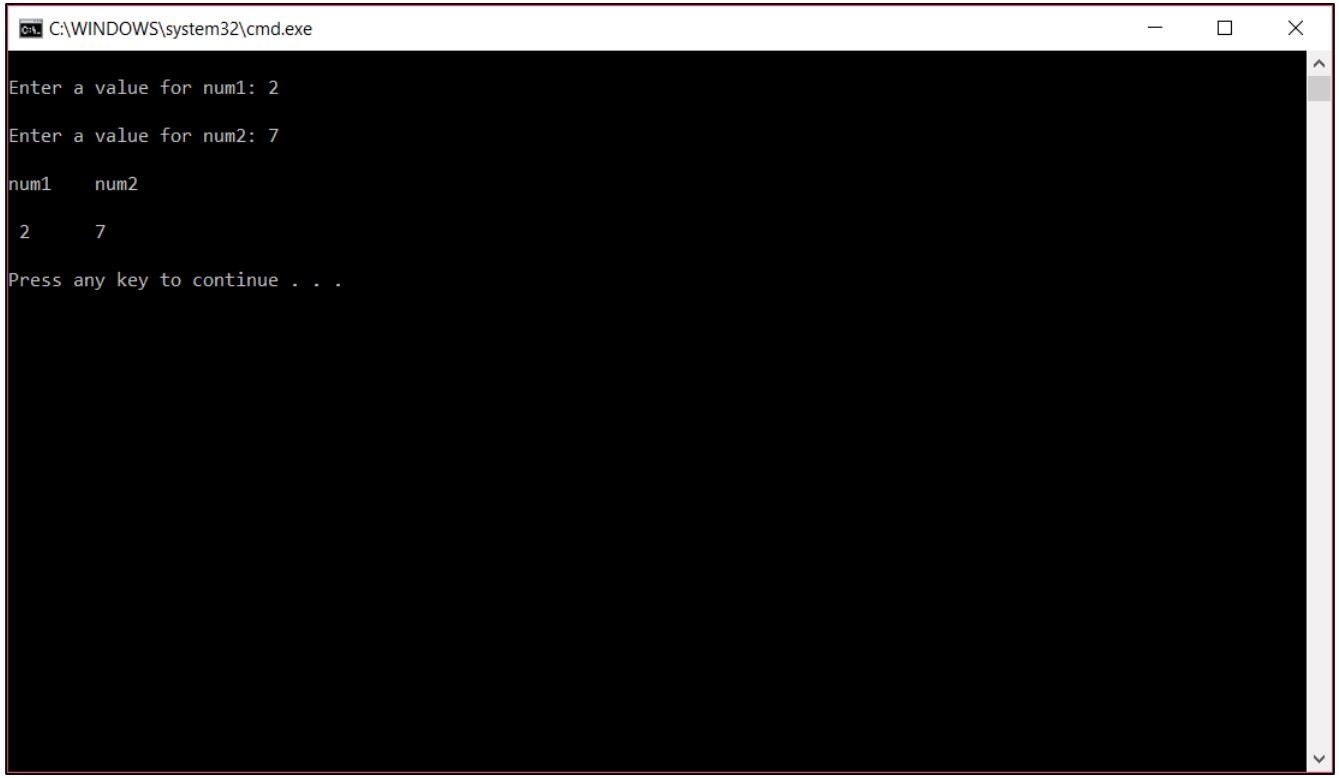
Assignment 4

- The implementation of assembly program is as follow.



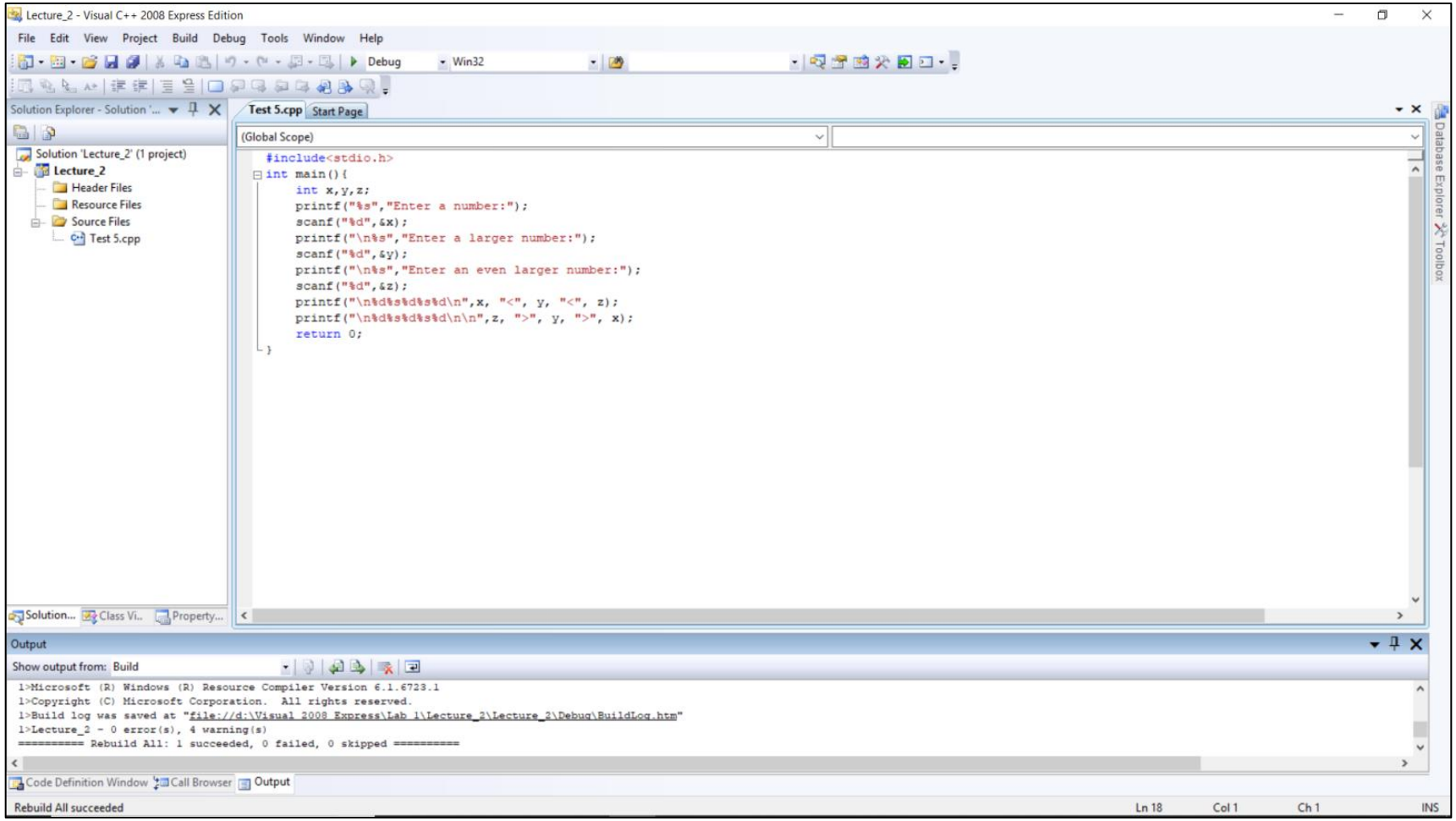
Assignment 4 (Cont.)

- The output for the above assembly program is as follow.



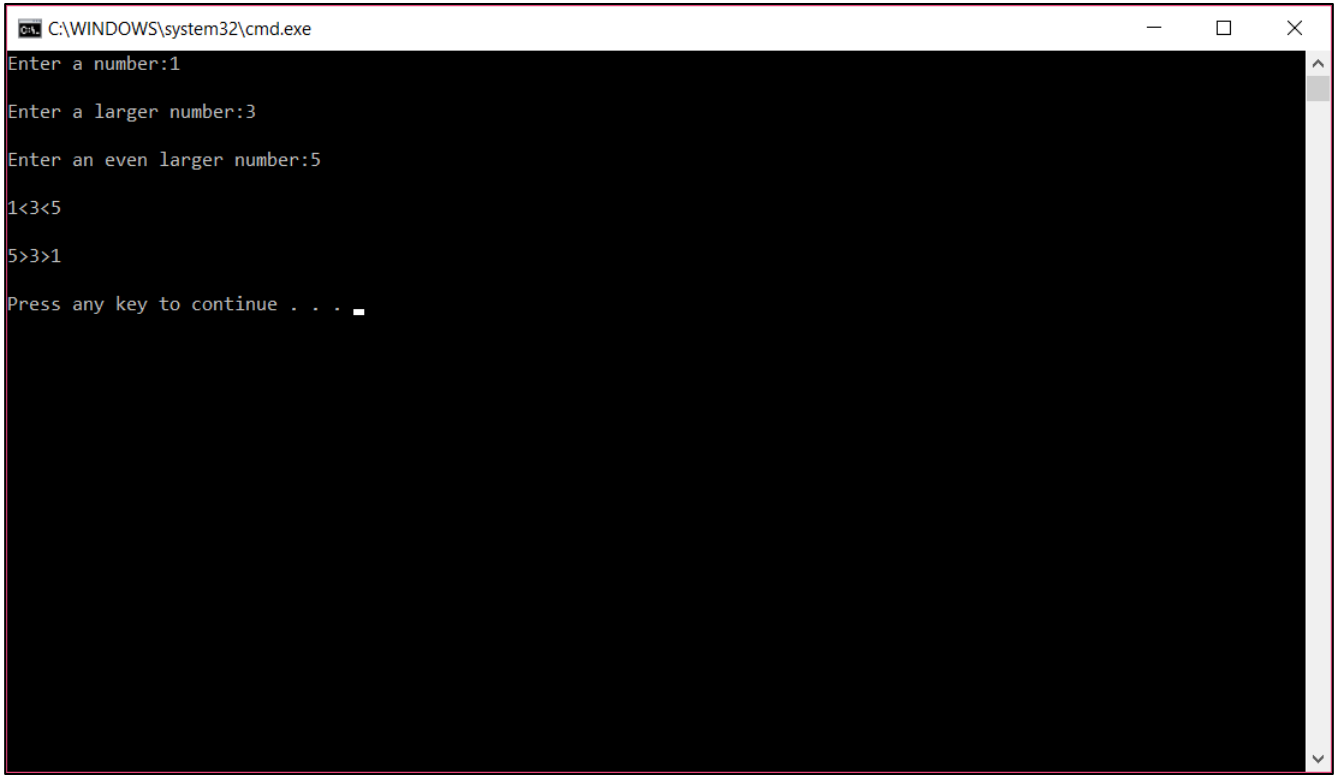
Assignment 5

- The implementation of C program is as follow.



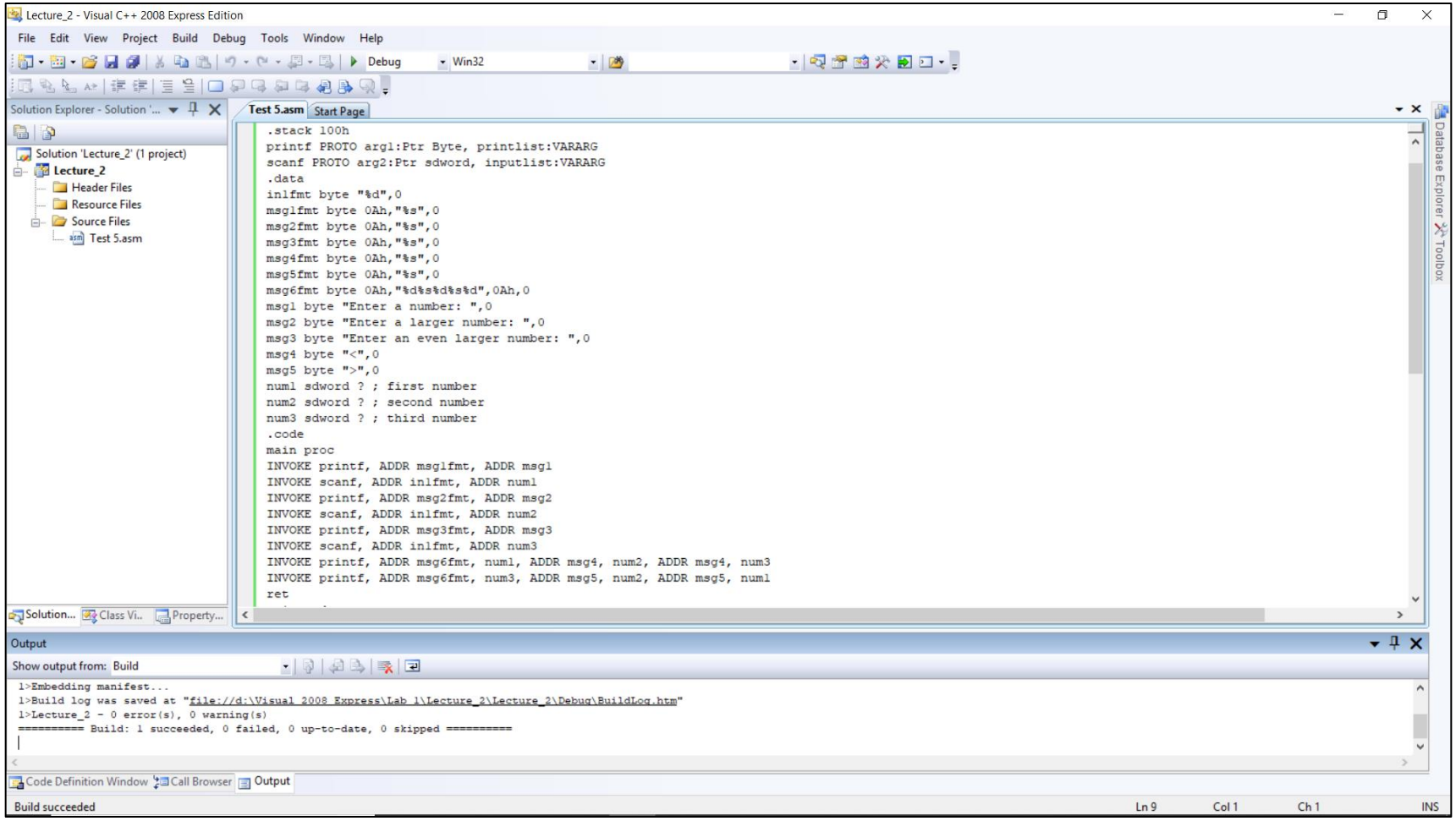
Assignment 5 (Cont.)

- The output for the above C program is as follow.



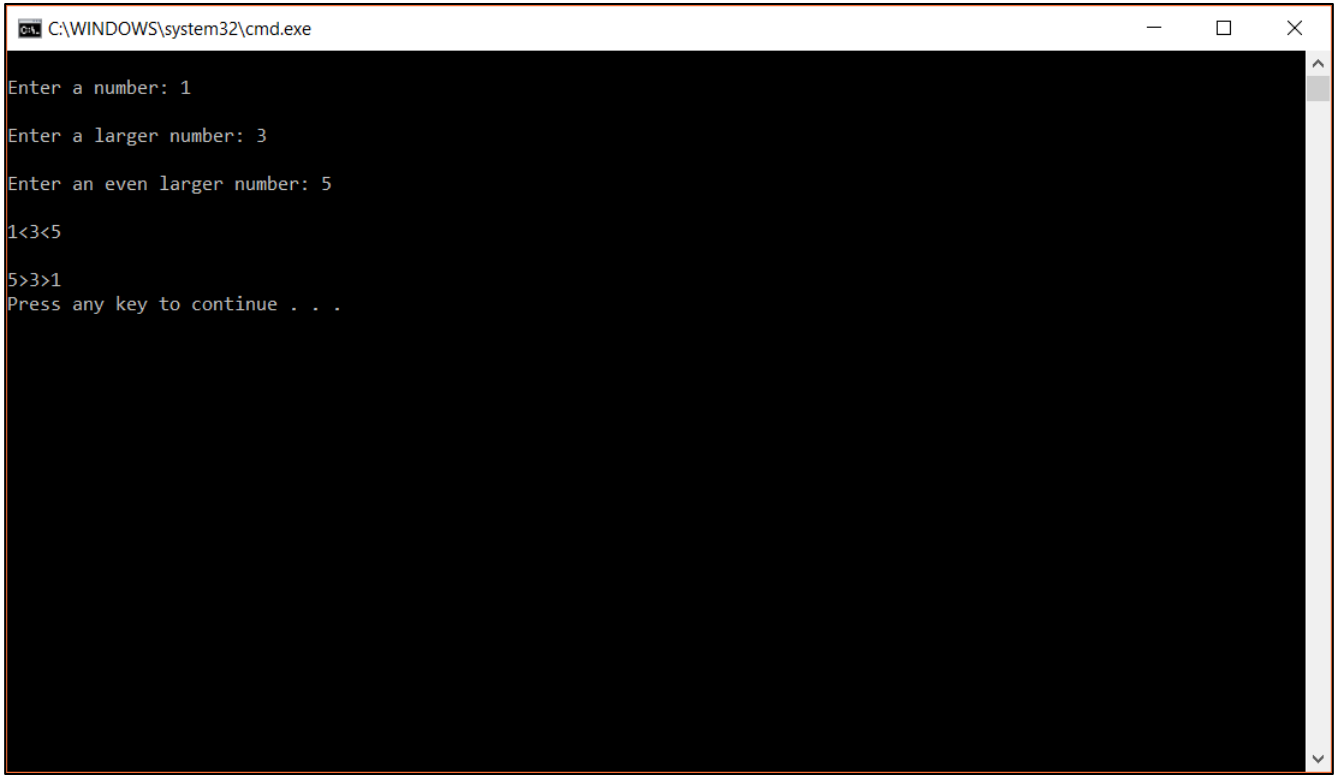
Assignment 5 (Cont.)

- The implementation of assembly program is as follow.



Assignment 5 (Cont.)

- The output for the above assembly program is as follow.



Next Lecture

- Introduction to Arithmetic Instructions
- Addition
- Subtraction
- Multiplication
- Division

Thank You