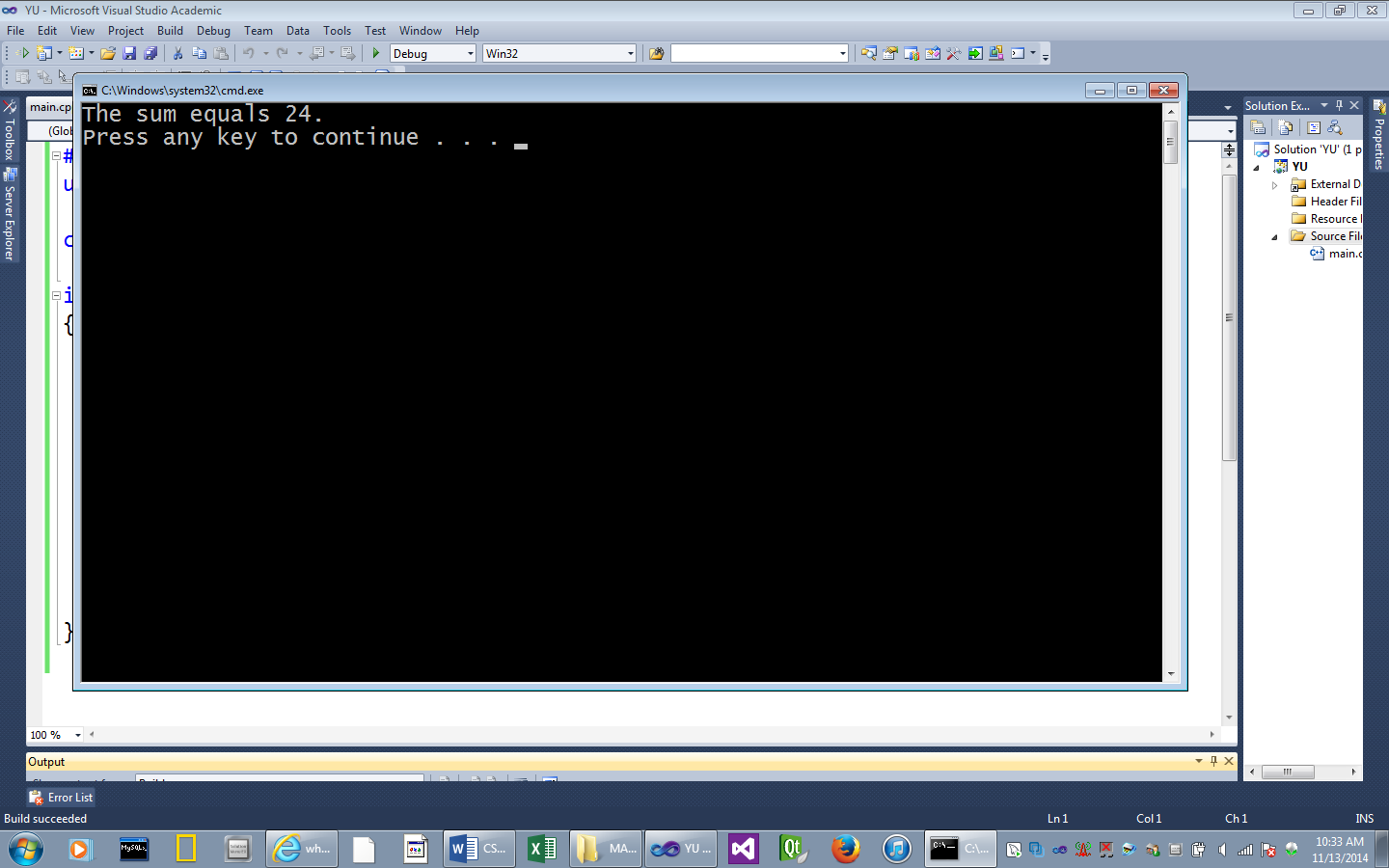
**MASM - Exercises**  - **(Ch. 4)**

**Class Exercise #1 - Convert the following C++ program in MASM**

#include <iostream>

using namespace std;

const int SIZE = 5;

int main()

{

int numbers[SIZE] = {1,3,5,7,8};

int sum = 0;

for (int i = 0; i < SIZE; i++)

{

sum = sum + numbers[i];

}

cout << "The sum equals " << sum << ".\n";

return 0;

}

Hint: Start by moving the address of numbers ( index [0] ) to the ESI

Then initialize the ECX with the size of the array.

Then use the LOOP instruction to access array elements.

With each iteration, add a array value to the sum.

Note: An array value can be added to the eax by dereferencing the ESI

Ex: add eax,[esi]

Then increment the ESI (by the number of bytes of the data type) to go to the

next array element.

**; ---- Exercise 4-1 – Solution -----------------------**

Include Irvine32.inc

ARRAY\_SIZE = 5

.data

numbers DWORD 1,3,5,7,8

sum DWORD 0

displaySum BYTE "The sum equals ",0

.code

main PROC

mov ecx, ARRAY\_SIZE ; (OR) mov ecx, LENGTHOF numbers

mov esi, OFFSET numbers

mov eax, sum

L1: add eax,[esi] ; dereferences the esi (a value in numbers)

add esi, TYPE numbers

loop L1

mov edx, OFFSET displaySum

call WriteString

call WriteInt

call Crlf

exit

main ENDP

END main

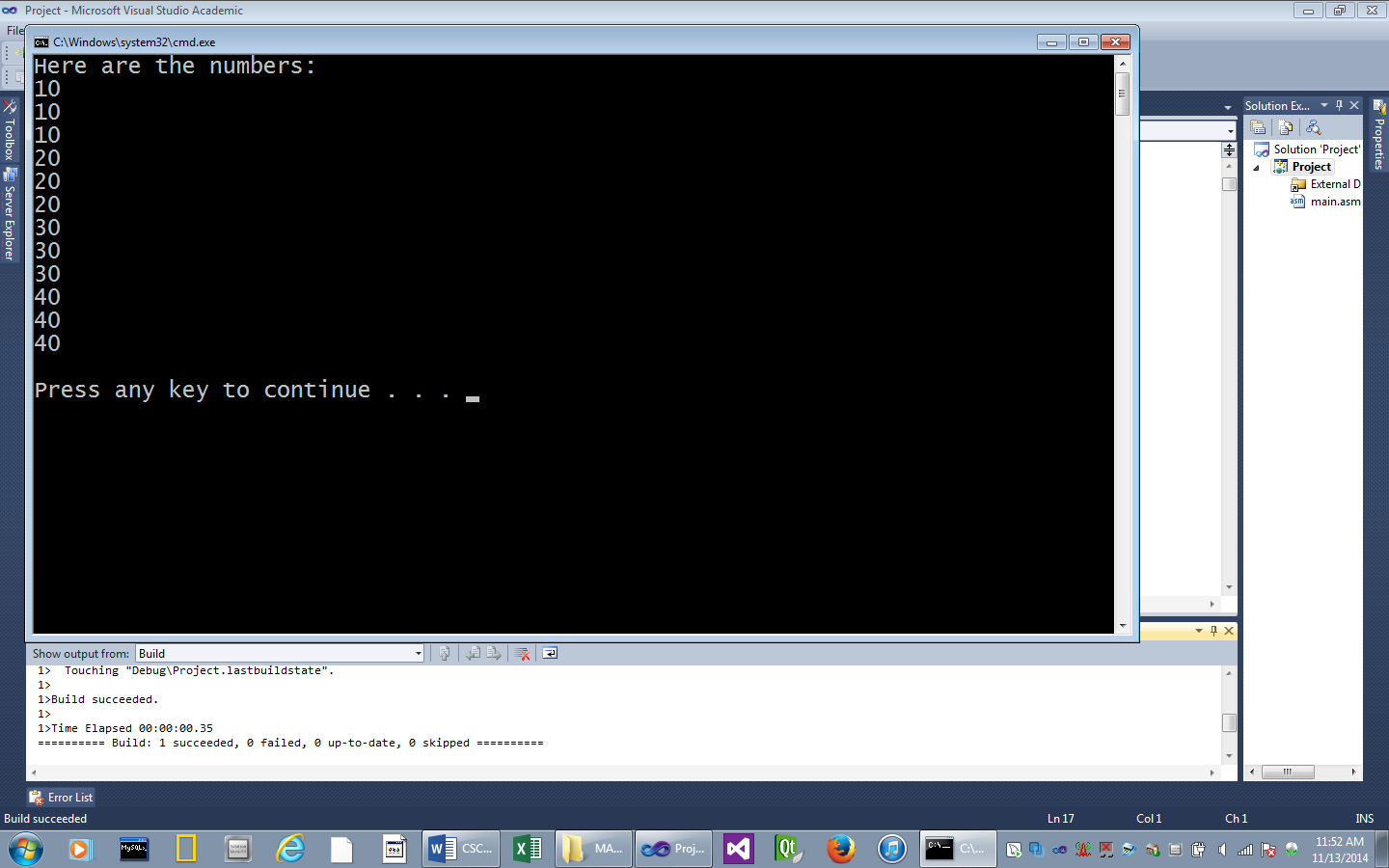
**Class Exercise #2 - Convert the following C++ program in MASM**

Convert this C++ program to MASM.

|  |
| --- |
| 10 |
| 10 |
| 10 |
| 20 |
| 20 |
| 20 |
| 30 |

Declare an array and assign these 12 numbers:

three 10’s, followed by three 20’s, then three 30’s and three 40’s



Then display the array values. etc.

#include <iostream>

using namespace std;

const int SIZE = 12;

int main()

{

int numbers[SIZE] = {10,10,10,20,20,20,30,30,30,40,40,40};

cout << "Here are the numbers:\n";

for (int i = 0; i < SIZE; i++)

{

cout << numbers[i] << endl;

}

return 0;

}

NOTE: Again, to access array elements, start by moving the address of numbers to the ESI,

and the ECX with the size of the array.

* Dereference the ESI

**; ---- Exercise 4-2 – Solution -----------------------**

Include Irvine32.inc

.data

numbers WORD 3 DUP(10), 3 DUP(20), 3 DUP(30), 3 DUP(40)

showNumbers BYTE "Here are the numbers: ",0dh, 0ah,0

.code

main PROC

mov edx, OFFSET showNumbers

call WriteString

mov ecx, LENGTHOF numbers

mov esi, OFFSET numbers

L1: mov eax, [esi] ; dereference the esi

call WriteInt

call Crlf

add esi,TYPE numbers

loop L1

call Crlf

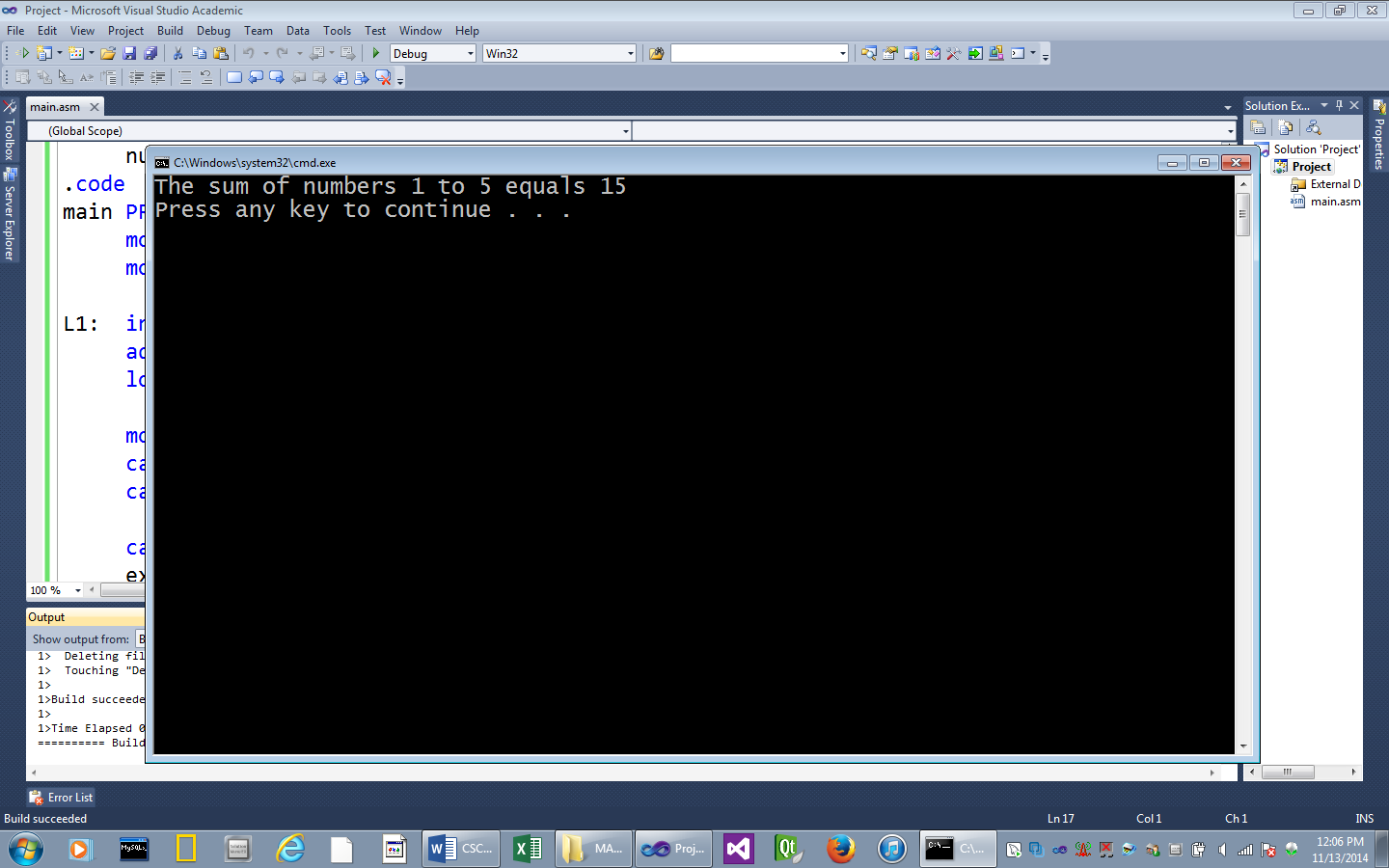
exit

main ENDP

END main

**MASM Class Exercise #3** - **Using a count-controlled loop**

1.) Write a program that uses a loop to add all numbers from 1 to 5.



2.) Here is the program in C++:

const int COUNT = 5;

int main()

{

int number = 1;

int sum = 0;

for (int i = 0; i < COUNT; i++)

{

sum = sum + number;

number++;

}

cout << "The sum equals " << sum << ".\n";

**; ---- Exercise 4-3 – Solution -----------------------**

Include Irvine32.inc

COUNT = 5 ; Declare a constant

.data

displaySum BYTE "The sum of numbers 1 to 5 equals ",0

number DWORD 0

.code

main PROC

mov ecx, COUNT ; Start the ECX with 5

mov eax, number

L1: inc number

add eax, number

loop L1 ; The ECX decrements with each loop (COUNT--;)

mov edx, OFFSET displaySum

call WriteString

call WriteInt

call Crlf

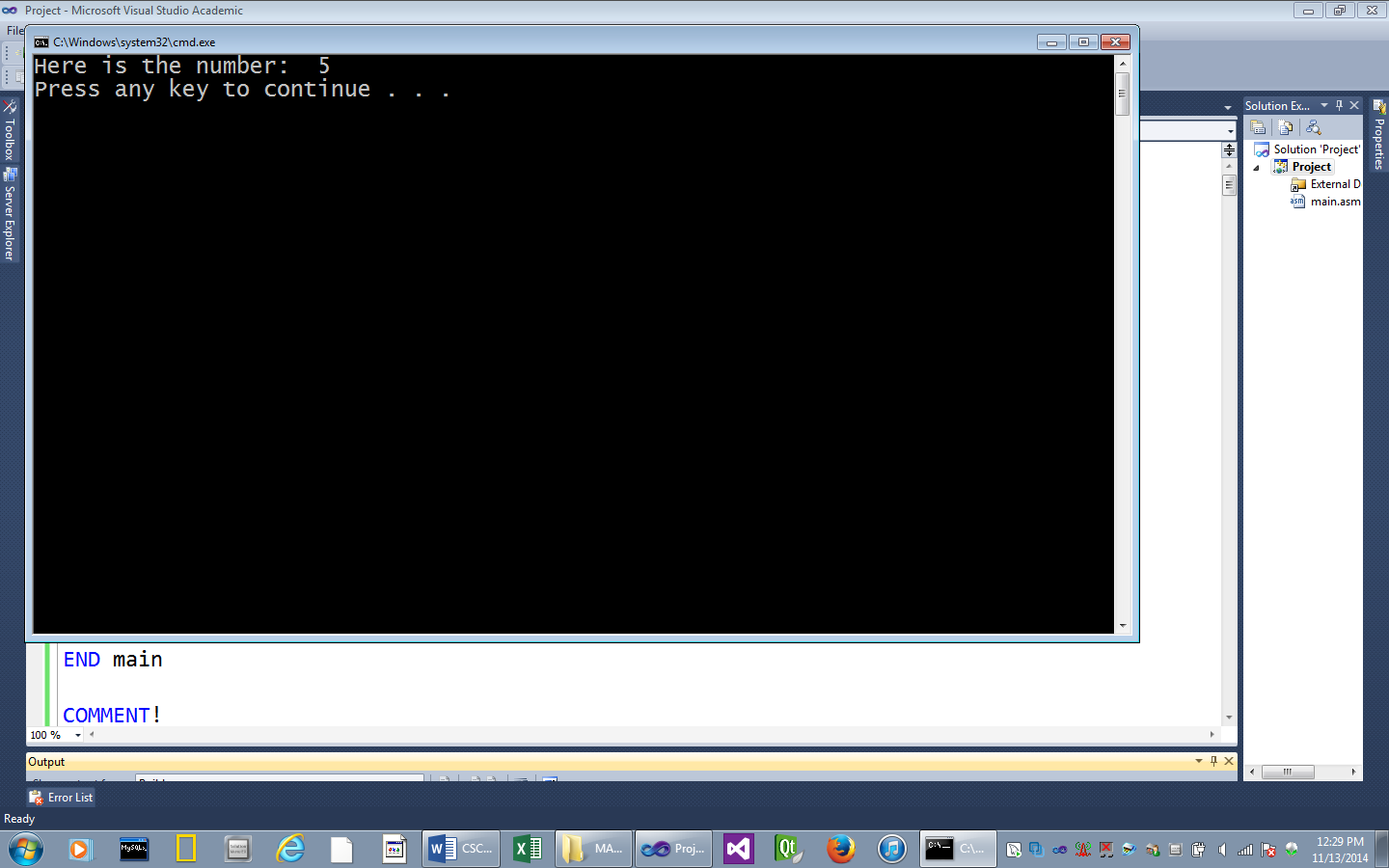
exit

main ENDP

END main

**MASM Class Exercise #4** - **Output a one byte integer**

1.) Declare a 1-byte unsigned integer variable named: number

2.) Initialize it a value of 5.

3.) Display the value.

**; ---- Exercise 4-4 – Solution -----------------------**

Include Irvine32.inc

.data

number BYTE 5

displayNum BYTE "Here is the number: ",0

.code

main PROC

mov edx, OFFSET displayNum

call WriteString

movzx eax, number

call WriteInt

call Crlf

exit

main ENDP

END main

**MASM Class Exercise #5** - **Output a one byte signed integer - Change Ex. #4**

1.) Change the variable *number* to a 1-byte **signed** **integer**

2.) Display the number, as in Ex. #4

**; ---- Exercise 4-5 – Solution -----------------------**

Include Irvine32.inc

.data

number SBYTE -5

displayNum BYTE "Here is the number: ",0

.code

main PROC

mov edx, OFFSET displayNum

call WriteString

**movsx eax, number ; Change from movzx**

call WriteInt

call Crlf

exit

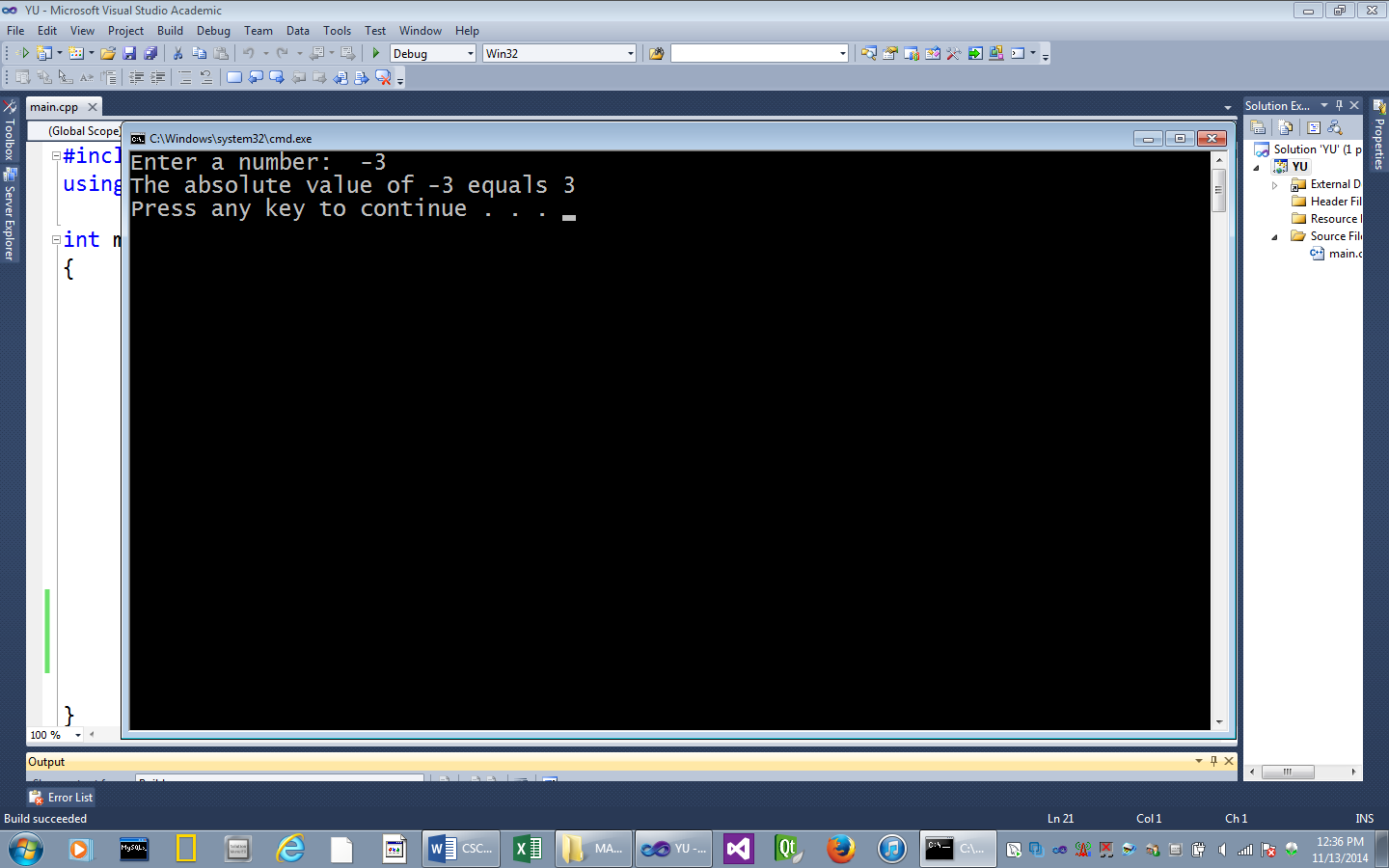
main ENDP

END main

**MASM Class Exercise #6** - **Convert the following C++ program in MASM**

This program reads an input number and then outputs the absolute value of the number.

#include <iostream>

using namespace std;

int main()

{

int number;

int result;

cout << "Enter a number: ";

cin >> number;

if (number < 0)

result = -1 \* number;

else

result = number;

cout << "The absolute value of " << number << " equals " << result;

cout << endl;

return 0;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. After the user input is read, use one of the *jmp* instructions to create the if control structure.

**; ---- Exercise 4-6 – Solution -----------------------**

Include Irvine32.inc

.data

getNum BYTE "Enter a number: ",0

showResult BYTE "The absolute value equals ",0

.code

main PROC

mov edx, OFFSET getNum

call WriteString

call ReadInt

cmp eax,0

jge Done

neg eax

Done:

mov edx, OFFSET showResult

call WriteString

call WriteDec

call Crlf

exit

main ENDP

END main

**MASM Class Exercise #7** - **Convert the following C++ program in MASM**

**NOTE: Use the .IF directive**

int main()  
{

int max = 0;

int numbers[10] = {3,5,1,9,12,77,31,2,71,4};

for (int i = 0; i < 10; i++)

{

if (numbers[i] > max)

{

max = numbers[i];

}

}

cout << "The highest value is " << max

return 0;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1.) Again use the ESI to access array elements.

2.) Use the ECX as a loop counter

3.) Create an if control structure using one of the jmp instructions.

4.) Use the LOOP instruction to continue to access array elements

**; ---- Exercise 4-7 – Solution – using .IF -----------------**

INCLUDE Irvine32.inc

.data

numbers dword 3,5,1,9,12,77,31,2,71,4

prompt byte "The biggest nubmer in the array is ",0

.code

main PROC

mov esi, offset numbers

mov ecx, lengthof numbers

mov eax, 0

mov ebx, 0

L1: mov eax, [esi]

.IF eax > ebx

mov ebx, eax

.ENDIF

add esi, type numbers

loop L1

mov eax, ebx

mov edx, offset prompt

call writestring

call writeint

call crlf

exit

main ENDP

END main

**; ---- Exercise 4-7 – Solution – Without .IF -----------------**

Include Irvine32.inc

.data

max SDWORD 0

numbers SDWORD 3,5,1,9,12,77,31,2,71,4

showResult BYTE "The highest value is ",0dh,0ah,0

.code

main PROC

mov eax, max

mov esi, OFFSET numbers

mov ecx, LENGTHOF numbers

L1:

cmp eax, [esi] ; Compare the value in the array to max

jge Greater

Loop L1

Greater:

mov eax [esi] ; max = [esi]

add esi, TYPE numbers ; Go to the next array element

LOOP L1

mov edx, OFFSET showResult

call WriteString

call WriteInt

call Crlf

exit

main ENDP

END main

**MASM Class Exercise #8** - **Convert the following C++ program in MASM**

**NOTE: Use .REPEAT – .UNTIL directives**

int main()  
{

int count = 0;

cout << "Squares of all numbers from 0 to 10:\n";

do {

cout << count << " " << count \* count << endl;

count++;

} while (count <= 10);

return 0;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**; ---- Exercise 8----** .REPEATE - .UNTIL **– Solution --------------------**

Include Irvine32.inc

.data

count SDWORD 0

msg BYTE "Squares of all numbers from 0 to 10: ",0dh,0ah,0

space BYTE " ", 0

.code

main PROC

mov edx, OFFSET msg

call WriteString ; Output a message

mov edx, OFFSET space

.REPEAT

mov eax, count

call WriteInt ; Output the number

call WriteString ; Output a space

imul eax, count

call WriteInt ; Output the square of the number

call Crlf ; Go to next line

inc count

.UNTIL count == 11

call Crlf

exit

main ENDP

END main

**MASM Class Exercise #9** - **Convert the following C++ program in MASM**

**NOTE: Use .WHILE - .ENDW directives**

int main()  
{

int count = 0;

cout << "Squares of all numbers from 0 to 10:\n";

while (count <= 10)

{

cout << count << " " << count \* count << endl;

count++;

}

return 0;

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**; ---- Exercise 9----** .WHILE - .ENDW **– Solution -----------------------**

Include Irvine32.inc

.data

count SDWORD 0

msg BYTE "Squares of all numbers from 0 to 10: ",0dh,0ah,0

space BYTE " ", 0

.code

main PROC

mov edx, OFFSET msg

call WriteString ; Output a message

mov edx, OFFSET space

.WHILE count <= 10

mov eax, count

call WriteInt ; Output the number

call WriteString ; Output a space

imul eax, count

call WriteInt ; Output the square of the number

call Crlf ; Go to next line

inc count

.ENDW

call Crlf

exit

main ENDP

END main