**Lab Session 11**



**High Level Language Interface**



**Objectives**

* General Conventions, Model Directive
* Implementing Inline Assembly Code

**.Model Directive**

.MODEL directive determines

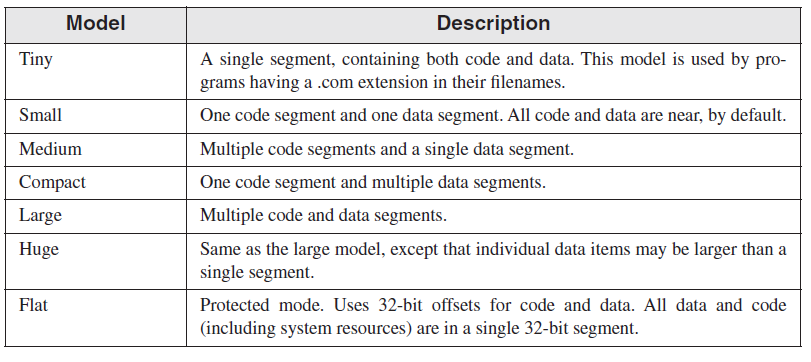
* memory model type
* procedure naming scheme
* parameter passing convention



Required parameter that determines the size of code and data pointers.

These are particularly important when assembly language is called by programs written in other programming languages.

**Memory Model**



**Memory Options**

Language specifier ->determines calling and naming conventions for proceduresand public symbols

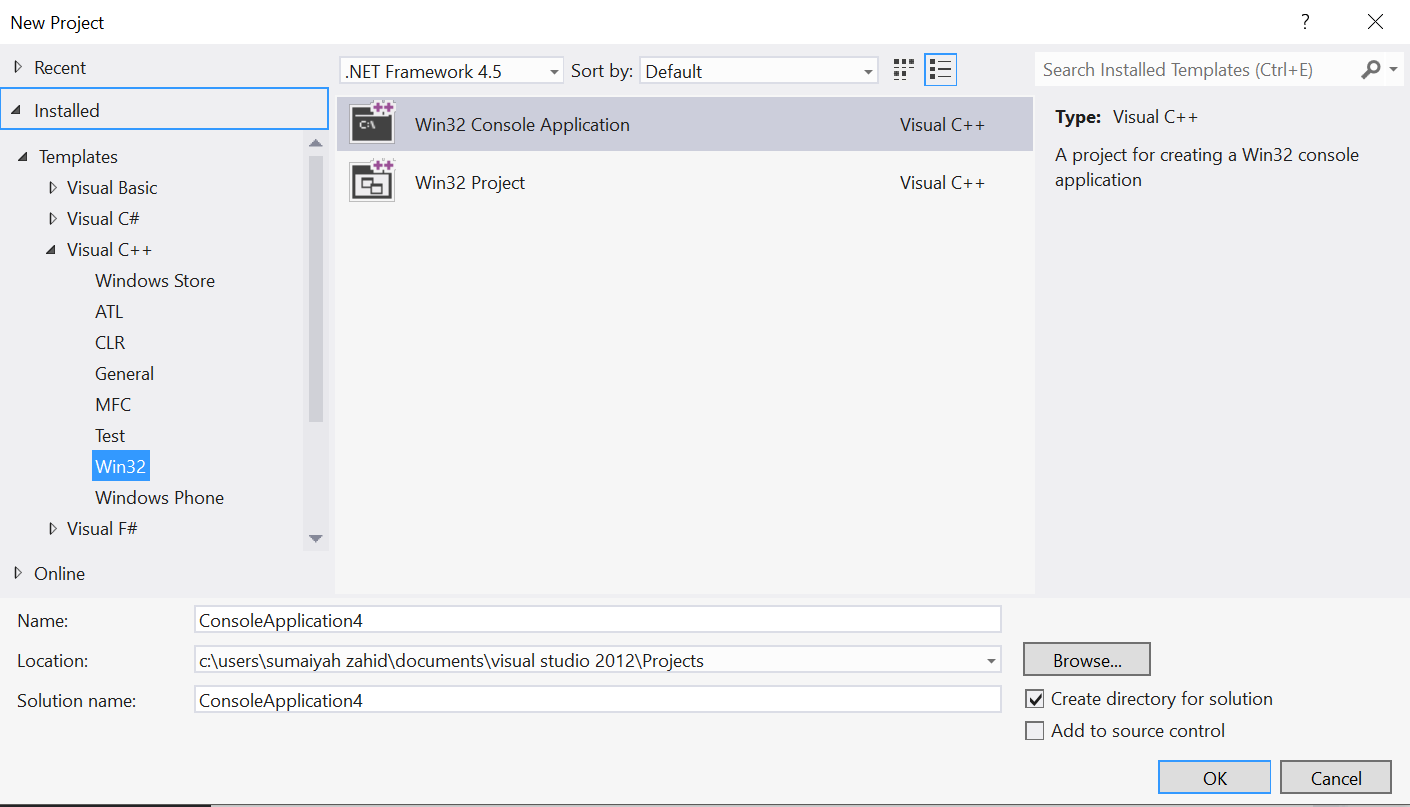
Stack distance -> can be NEARSTACK (the default) or FARSTACK

We mostly uses *.model flat, STDCALL*

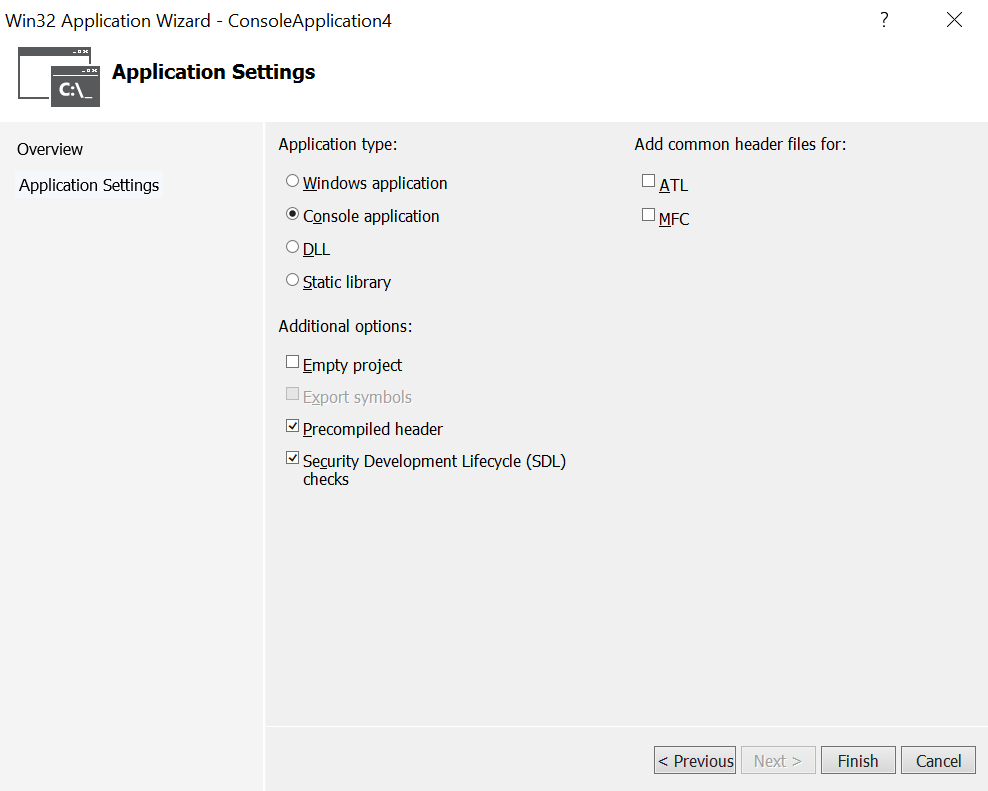
**STDCALL** is the language specifier used when calling MS-Windows functions.

**Steps to follow**

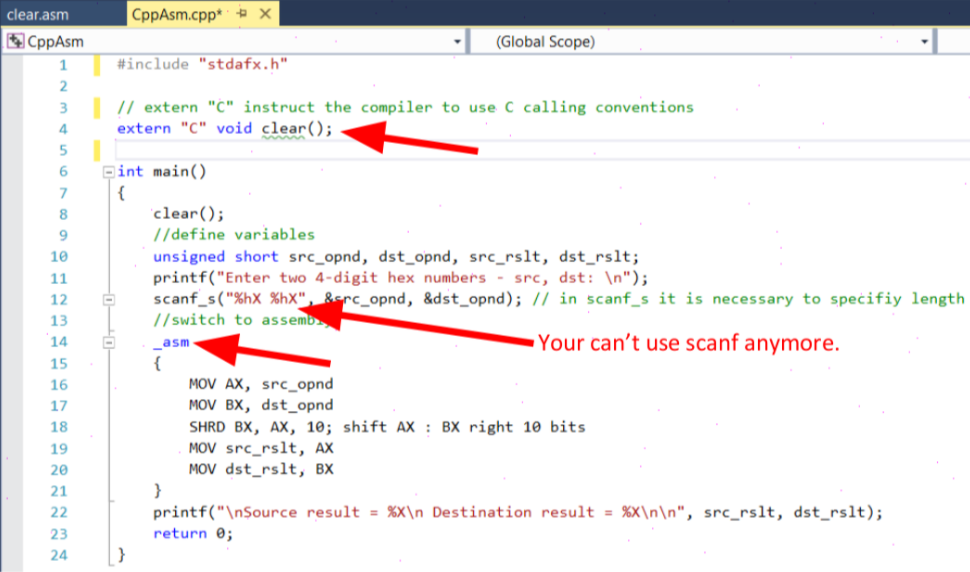
1. Select **New Project > Visual C++ > Win32 >Win32 Console Application**



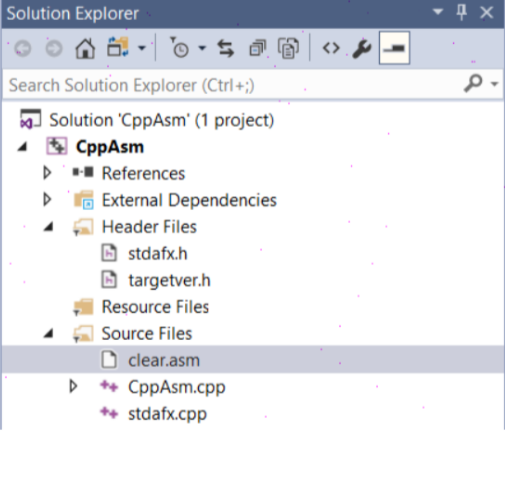
1. Make sure to check Pre Compiled Header



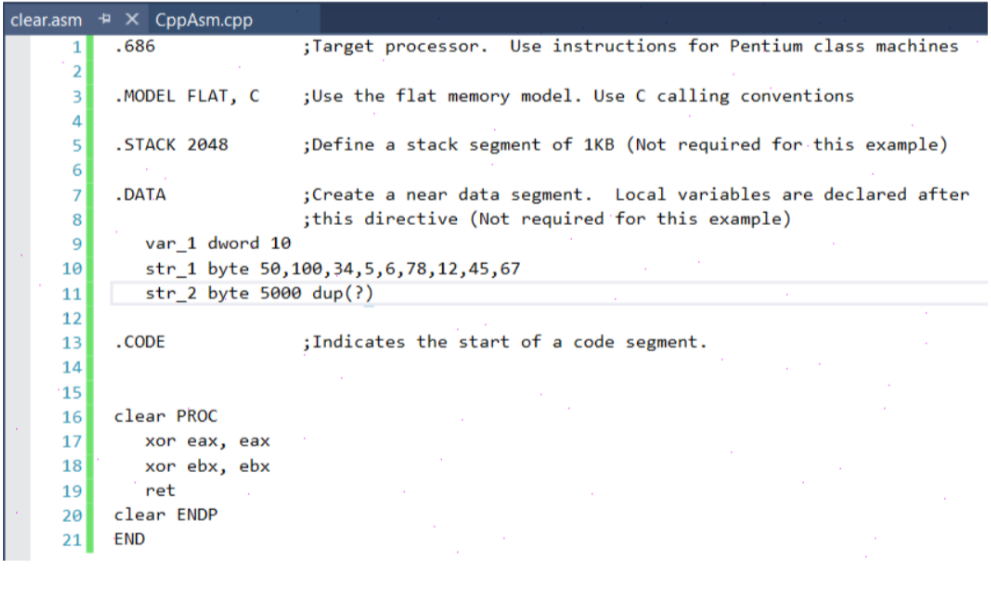
1. Write your C++ code in source file



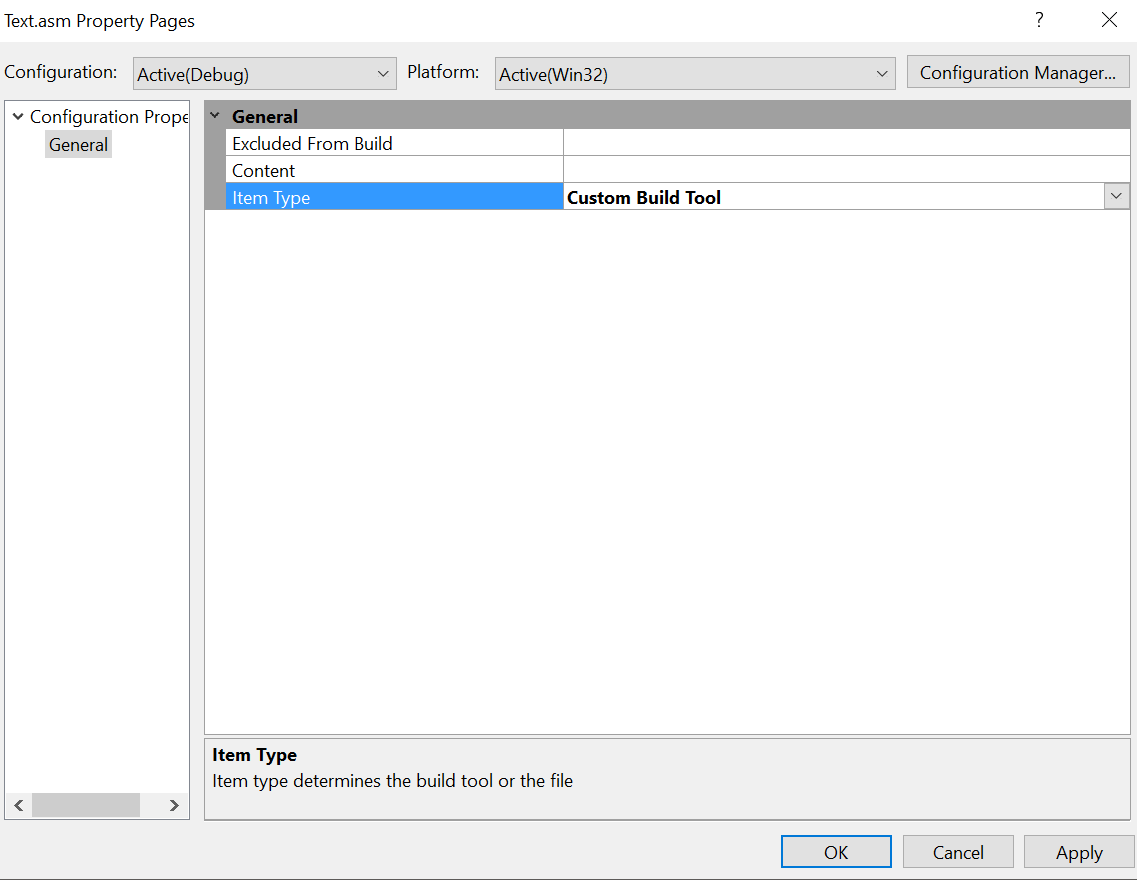
1. Add .asm file to the project



1. Write Assembly code in your .asm file



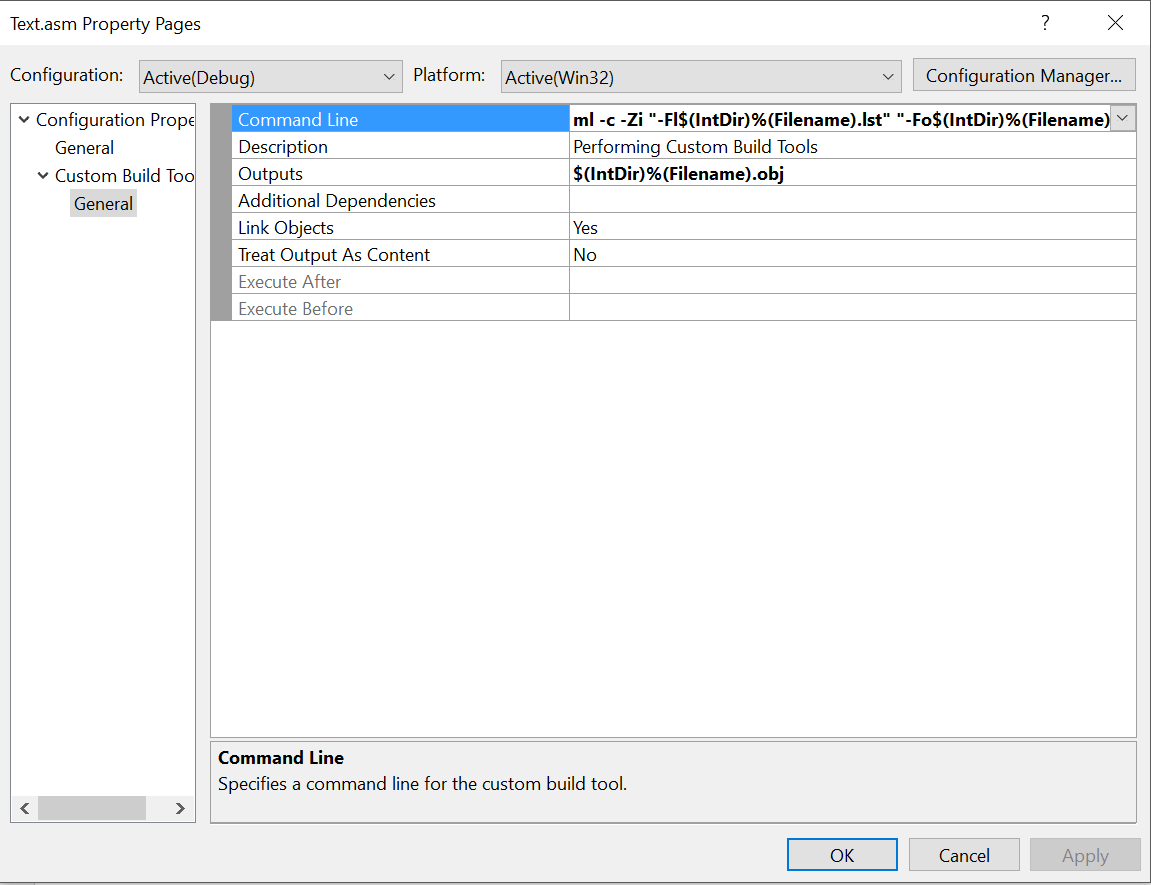
1. Right click on .asm file and select Properties. Select **Configuration Properties > General > Item Type > Custom Build Tool**and Apply the changes.



1. Select **Configuration Properties > Custom Build Tool > General**

**Command Line:**ml -c -Zi "-Fl$(IntDir)%(Filename).lst" "-Fo$(IntDir)%(Filename).obj" "%(FullPath)"

**Outputs:**$(IntDir)%(Filename).obj



**Example:**

**C++ Code:**

#include"stdafx.h"

// extern "C" instruct the compiler to use C calling conventions

extern"C"void clear();

int main()

{

clear();

//define variables

unsignedshort src\_opnd, dst\_opnd, src\_rslt, dst\_rslt;

printf("Enter two 4-digit hex numbers - src, dst: \n");

scanf\_s("%hX %hX", &src\_opnd, &dst\_opnd); // in scanf\_s it is necessary to specifiy length

//switch to assembly

\_asm

{

MOV AX, src\_opnd

MOV BX, dst\_opnd

SHRD BX, AX, 4; shift AX : BX right 10 bits

MOV src\_rslt, AX

MOV dst\_rslt, BX

}

printf("\nSource result = %X\n Destination result = %X\n\n", src\_rslt, dst\_rslt);

return 0;

}

**Assembly Code:**

.686 ;Target processor. Use instructions for Pentium class machines

.MODEL FLAT, C ;Use the flat memory model. Use C calling conventions

.STACK 2048 ;Define a stack segment of 1KB (Not required for this example)

.DATA ;Create a near data segment. Local variables are declared after

;this directive (Not required for this example)

var\_1 dword 10

str\_1 byte 50,100,34,5,6,78,12,45,67

str\_2 byte 5000 dup(?)

.CODE ;Indicates the start of a code segment.

clear PROC

xor eax, eax

xor ebx, ebx

ret

clear ENDP

END

**Example2 :**

.686 ;Target processor. Use instructions for Pentium class machines

.MODEL FLAT, C ;Use the flat memory model. Use C calling conventions

;INCLUDE Irvine32.inc

.STACK 2048 ;Define a stack segment of 1KB (Not required for this example)

askForInteger PROTO C

showInt PROTO C, value:SDWORD, outWidth:DWORD

OUT\_WIDTH = 8

ENDING\_POWER = 10

.data

intValDWORD ?

.CODE ;Indicates the start of a code segment.

clear PROC

xoreax, eax

xorebx, ebx

INVOKE askForInteger ; call C++ function

movintVal,eax ; save the integer

movecx,ENDING\_POWER ; loop counter

L1: push ecx ; save loop counter

shl intVal,1 ; multiply by 2

INVOKE showInt,intVal,OUT\_WIDTH

pop ecx ; restore loop counter

loop L1

ret

clear ENDP

END

**Cpp**

#include"stdafx.h"

#include<iostream>

#include<iomanip>

usingnamespacestd;

// extern "C" instruct the compiler to use C calling conventions

extern"C" {

voidclear();

intaskForInteger();

voidshowInt(int value, int width);

}

intmain()

{

clear( );

return 0;

}

intaskForInteger()

{

int n;

cout<<"Enter an integer between 1 and 90,000:";

cin>> n;

return n;

}

// Display a signed integer with a specified width.

voidshowInt( intvalue, intwidth )

{

cout<<setw(width) <<value;

}

**ACTIVITIES:**

1. Take two number as input and perform AND operation using \_asm directive.
2. Take two number as input and perform OR operation using \_asm directive
3. Generate first 10 number of Fibonacci series through asm directive calling in C/C++.
4. Write a program finds smallest of the three integers
5. Write a program finds largest of the two integers
6. Write a program to sort an array of 10 integers using c to Assembly call and vice versa.
7. Write a program to find factorial of a given number.