EECE416 :Microcomputer Fundamentals and Design

32-bit Example with Visual Studio

Dr. Charles Kim

Department of Electrical and Computer Engineering

Howard University

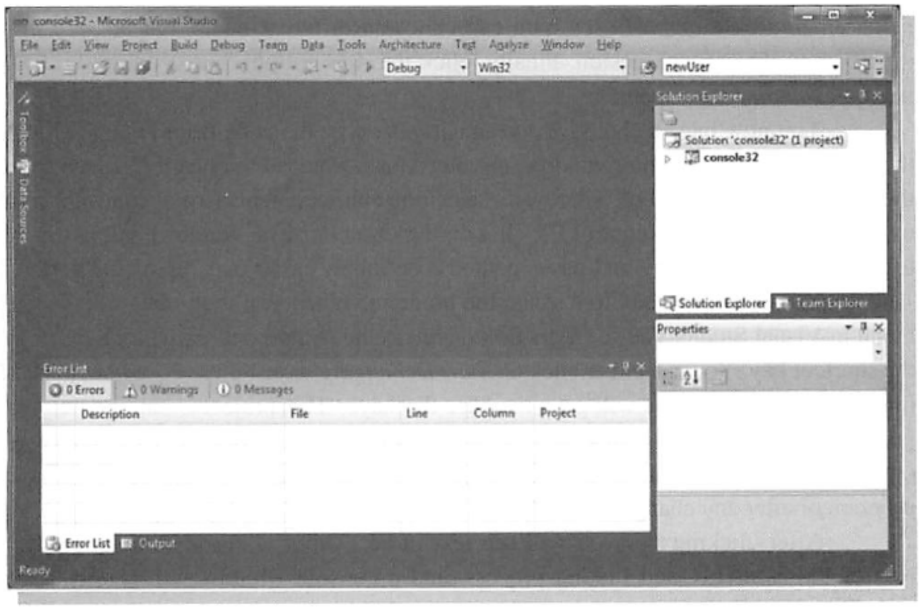
www.MWFTR.com

Assembly Language Statements

- Comments: semicolon (;) begins a comment which extends to the end of the line
- Instructions and Directives and Macros
  - Instructions
    - Code partParallel Learning
  - Directives
    - Tells Assembler to take some actions
    - .586 --- "Use 32-bit operands"
    - . MODEL FLAT --- "Flat memory model"
    - .STACK 4096 --- "Reserve 4096 Bytes for the system stack"
    - .DATA --- "data items are defined in a data segment"
    - .CODE --- "next statements are executable insturctions"
    - main PROC --- "Beginning of a procedure"
    - main ENDP --- "End of a procedure"
  - Macros
    - "Shorthand" for a sequence of statements – instructions and directives and other macros

32-bit Example with Visual Studio

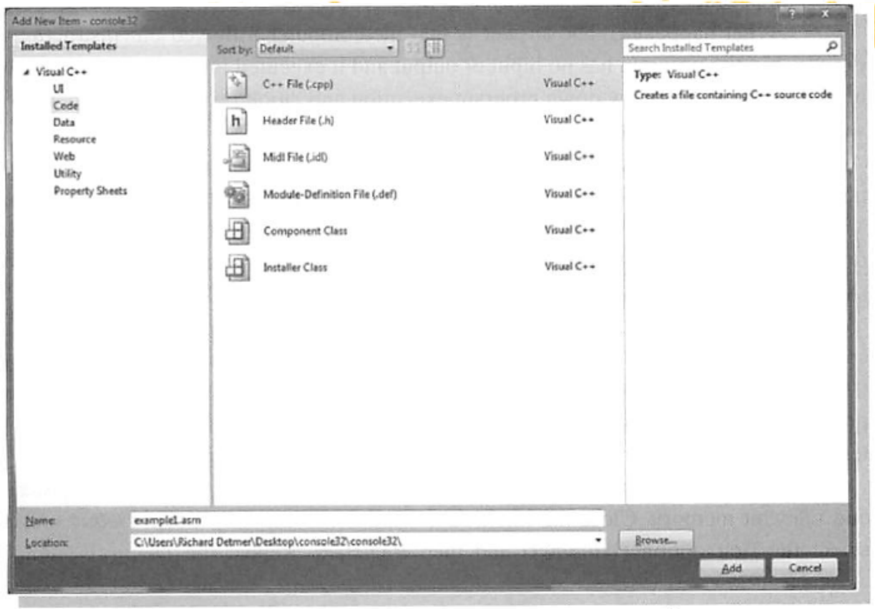
- ⌘ Microsoft Visual Studio 2010
- ⌘ Console32 project folder --- this folder's name can be changed
- ⌘ Double-click the console32.sln (Note: Never change the sub-folder's name) to start Visual Studio
- ⌘ A screen below must show



- ⌘ In the "Solution Explorer" Window (right or left), click the + symbol
- ⌘ Right-Click "Source Files"
- ⌘ Click "Add" → New Item

New ASM Code

- Adding a new assembly language file



C++ File (.cpp)	Visual C++
Header File (.h)	Visual C++
Midl File (.idl)	Visual C++
Module-Definition File (.def)	Visual C++
Component Class	Visual C++
Installer Class	Visual C++

Type: Visual C++

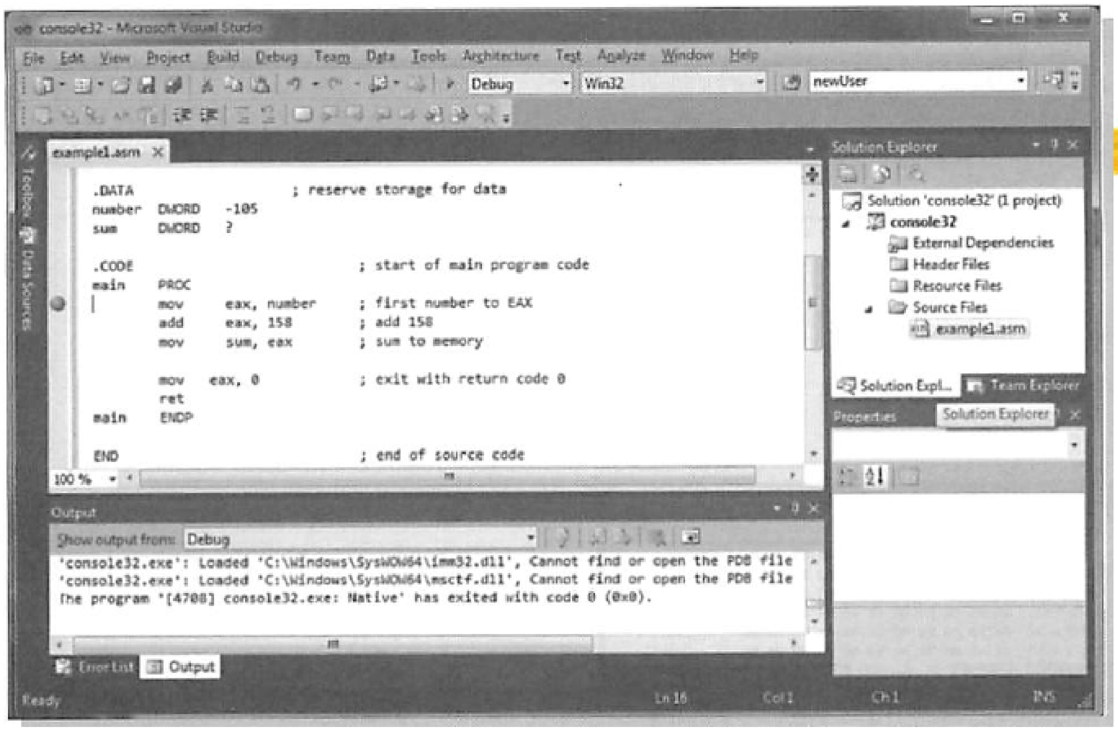
Creates a file containing C++ source code

Name: example1.asm

Location: C:\Users\Richard Detmer\Desktop\console32\console32\

- Select “Code” under “Categories”
- Type name of the file (exampl1.asm for example) in the “Name” box
- Click “Add”
- Coding: Manual Typing or Open an example code in a note-pad and select, copy, and paste to the Console32 space

Execution of the Code



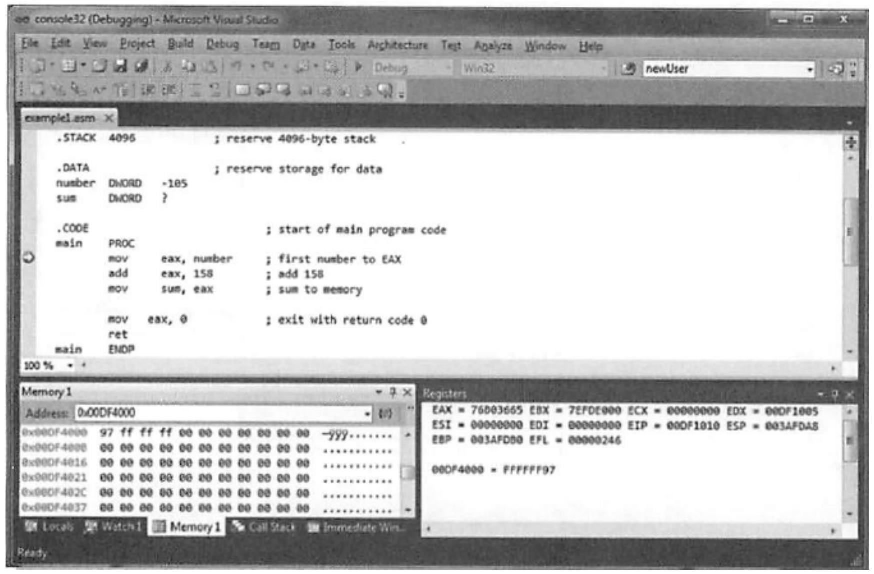
```
.DATA                          ; reserve storage for data
number  DWORD   -105
sum     DWORD   ?

.CODE                                    ; start of main program code
main    PROC
        mov     eax, number              ; first number to EAX
        add     eax, 158                 ; add 158
        mov     sum, eax                 ; sum to memory

        mov   eax, 0                     ; exit with return code 0
        ret
main    ENDP

END                                      ; end of source code
```

```
'console32.exe': Loaded 'C:\Windows\SysWOW64\imm32.dll', Cannot find or open the PDB file
'console32.exe': Loaded 'C:\Windows\SysWOW64\msctf.dll', Cannot find or open the PDB file
The program '[4708] console32.exe: Native' has exited with code 0 (0x0).
```

- Drop down “Debug” > “Start Debug” (or shortcut key F5)
- “Yes” to assemble, link, and initiate execution
- Break Point
  - Click next to an instruction → A red dot appears for a break point, a place at which the execution will halt
  - Break point is removed by clocking the red dot

Register and Memory Contents

- Drop Down “Debug” > Select “Windows” > Select “Registers”
- Drop Down “Debug” > “Windows”>”Memory” > “Memory 1”
- 2 tabbed windows appear at the bottom of the screen
- Drag “Regisiters” tab to the right-hand and drop it to the right side
- Select “Memory 1” window, and type “**`&number`**” in the “Address” box.


console32 (Debugging) - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Debug
Win32
newUser
example1.asm
100 %
Memory 1
Address: 0x00DF4000
Registers
Locals
Watch 1
Memory 1
Call Stack
Immediate Win...
Ready

```
    .STACK  4096            ; reserve 4096-byte stack

    .DATA                   ; reserve storage for data
    number  DWORD   -105
    sum     DWORD   ?

    .CODE                           ; start of main program code
    main    PROC
            mov     eax, number     ; first number to EAX
            add     eax, 158        ; add 158
            mov     sum, eax        ; sum to memory

            mov   eax, 0            ; exit with return code 0
            ret
    main    ENDP
```

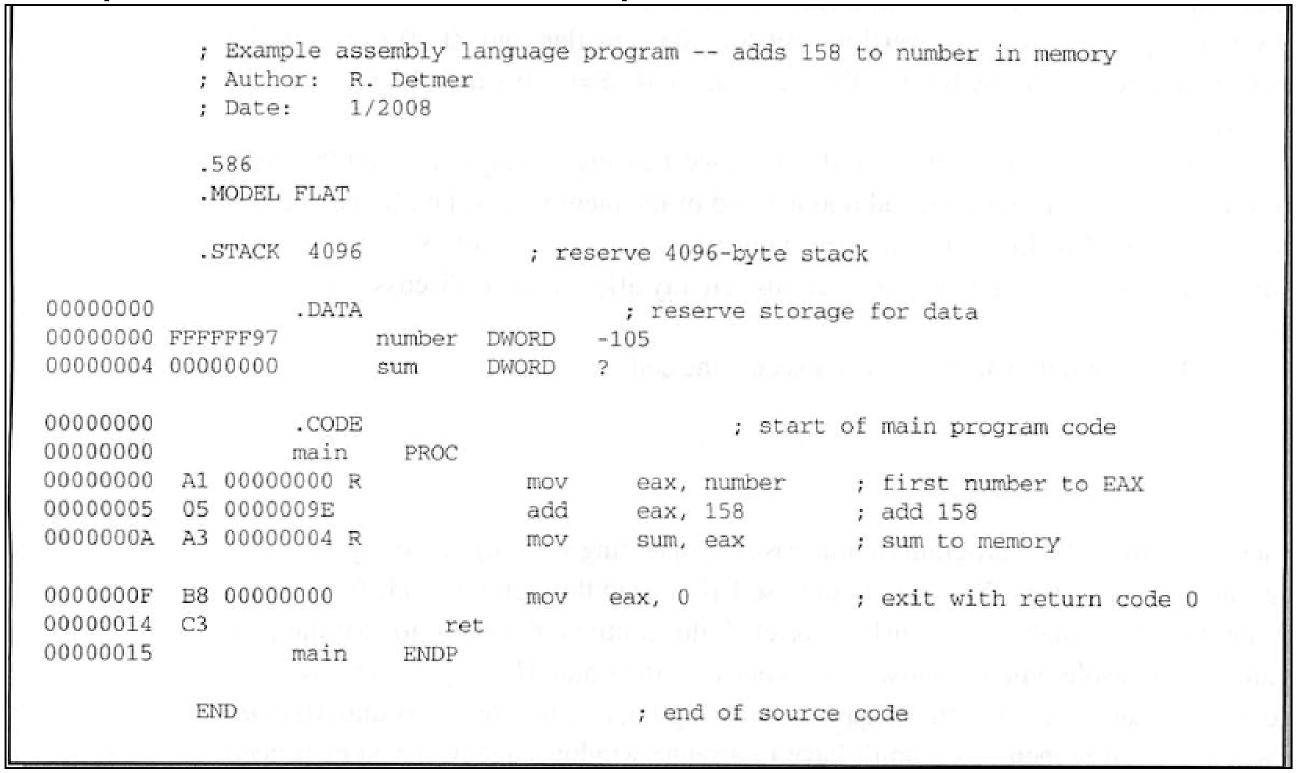
```
0x00DF4000  97 ff ff ff 00 00 00 00 00 00 00  —ÿÿÿ.......
0x00DF400B  00 00 00 00 00 00 00 00 00 00 00  ...........
0x00DF4016  00 00 00 00 00 00 00 00 00 00 00  ...........
0x00DF4021  00 00 00 00 00 00 00 00 00 00 00  ...........
0x00DF402C  00 00 00 00 00 00 00 00 00 00 00  ...........
0x00DF4037  00 00 00 00 00 00 00 00 00 00 00  ...........
```

```
EAX = 76003665 EBX = 7EFDE000 ECX = 00000000 EDX = 00DF1005
ESI = 00000000 EDI = 00000000 EIP = 00DF1010 ESP = 003AFDA8
EBP = 003AFDB0 EFL = 00000246

00DF4000 = FFFFFF97
```

- Step-Over Debug
  - Execution of 1 instruction at a time
  - Good for checking register contents and memory (and Flags)
  - Drop Down “Debug” > Step Over (or F10 shortcut key)

Listing Files

- Listing file is to be generated when a code is assembled
- Source and object code
- Location of assembly error
- “`Example1.lst`” for “Example1.asm”



```
                    ; Example assembly language program -- adds 158 to number in memory
                    ; Author:  R. Detmer
                    ; Date:    1/2008

                    .586
                    .MODEL FLAT

                    .STACK  4096            ; reserve 4096-byte stack

00000000                .DATA                   ; reserve storage for data
00000000 FFFFFF97           number  DWORD   -105
00000004 00000000           sum     DWORD   ?

00000000                .CODE                               ; start of main program code
00000000                main    PROC
00000000  A1 00000000 R             mov     eax, number     ; first number to EAX
00000005  05 0000009E               add     eax, 158        ; add 158
0000000A  A3 00000004 R             mov     sum, eax        ; sum to memory

0000000F  B8 00000000               mov   eax, 0            ; exit with return code 0
00000014  C3                    ret
00000015                main    ENDP

                    END                         ; end of source code
```