Faculty Member:____________________ **Dated: ________________**

Semester:_____________ **Section: ________________**

School of Electrical Engineering and Computer Science

EE-222 Microprocessor Systems

Lab2-B: Saving Values in Registers and Debug them using Win Debugger

Name	Reg. No.	Viva / Quiz / Lab Performance 5 Marks	Analysis of data in Lab Report 5 Marks	Modern Tool usage 5 Marks	Ethics and Safety 5 Marks	Individual and Team Work 5 Marks	Total 25 Marks

EXPERIMENT 02

SAVING VALUES IN REGISTERS AND DEBUG THEM USING WIN DEBUGGER

OBJECTIVES:

1. Getting introduced to assembly language
2. Learning some basic commands
3. Introduction to the syntax of assembly language programming
4. Learning the use of debugger

EQUIPMENT:

SOFTWARE:

- Microsoft assembler (MASM)

DISCUSSION:

Before starting coding in assembly we should get familiarized with some basic coding parameters, assembly language syntax and some basics of microprocessors.

Introduction to Registers:

There are four type of registers in microprocessors:

1. AX
2. BX
3. CX
4. DX

AX, BX are mainly used for arithmetic operations and saving address. CX is used for saving the values for counts which is used in executing loop instructions and DX is mainly used for I/O operations.

PROGRAM STRUCTURE:

MEMORY MODELS:

The size of code and data a program can have is determined by specifying memory model using the .MODEL directive. The models used are SMALL, LARGE, and HUGE but the appropriate one is .SMALL. The model directive should come before any segment definition.

DATA SEGMENT:

The data segment is used for all the variables definitions. We use **.DATA** directive followed by variable and constant declaration.

STACK SEGMENT:

The purpose of stack segment is to set aside a block of memory to store the stack. The declaration syntax is:

.stack   size

If we write:

.stack 100h

100 bytes would be reserved for stack. If size omitted 1KB is the default size.

CODE SEGMENT:

Code segment contains all the programming instructions. The declaration syntax is:

```
.model small

.stack 100h

.data

            ; Data definitions go here

.code

.startup

            ; Instructions go here

.end
```

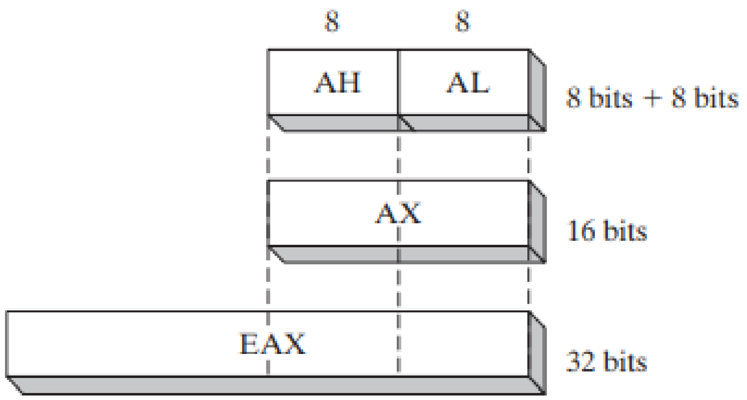
The last line here should be END directive followed by the exit.

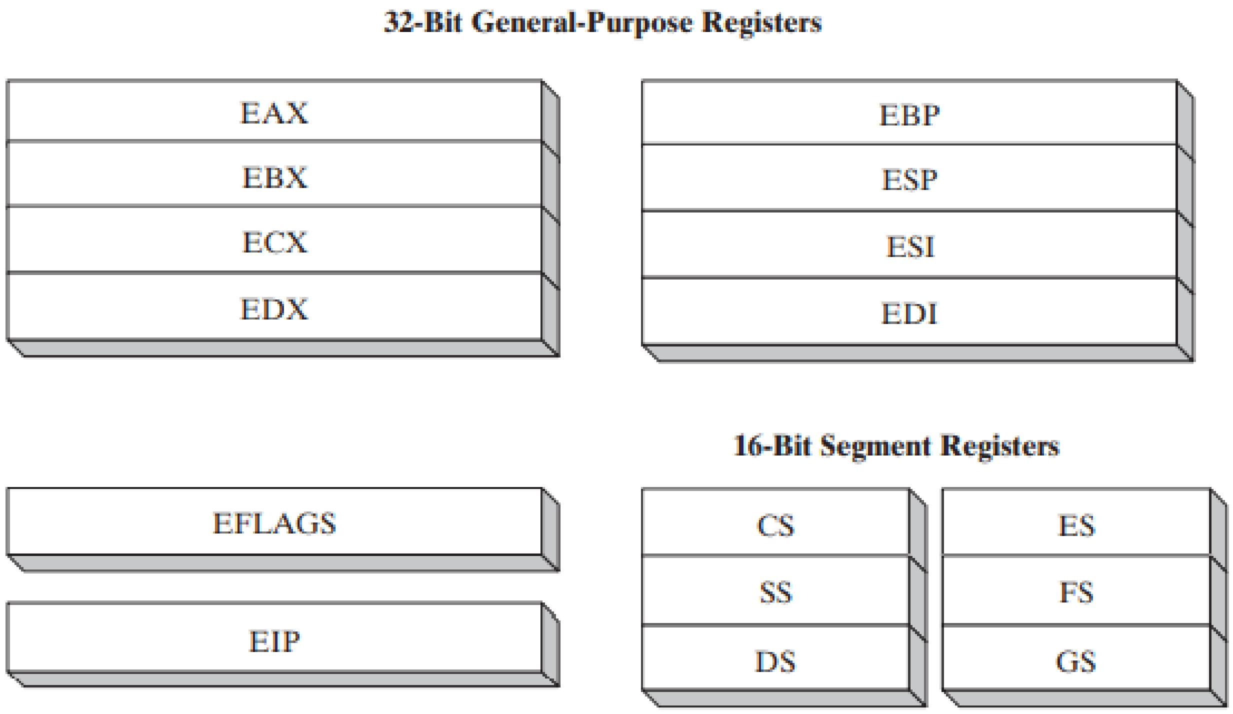
REAL MODE:

In real-address mode, only 1 MByte of memory can be addressed, from hexadecimal 00000 to FFFFF. The processor can run only one program at a time, but it can momentarily interrupt that program to process requests (called interrupts) from peripherals. Application programs are permitted to access any memory location, including addresses that are linked directly to system hardware. The MS-DOS operating system runs in real-address mode, and Windows 95 and 98 can be booted into this mode.

PROTECTED MODE:

In protected mode, the processor can run multiple programs at the same time. It assigns each process (running program) a total of 4 Giga Byte of memory. Each program can be assigned its own reserved memory area, and programs are prevented from accidentally accessing each other's code and data. MS-Windows and Linux run in protected mode.





Operand	Description
reg8	8-bit general-purpose register: AH, AL, BH, BL, CH, CL, DH, DL
reg16	16-bit general-purpose register: AX, BX, CX, DX, SI, DI, SP, BP
reg32	32-bit general-purpose register: EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP
reg	Any general-purpose register
sreg	16-bit segment register: CS, DS, SS, ES, FS, GS
imm	8-, 16-, or 32-bit immediate value
imm8	8-bit immediate byte value
imm16	16-bit immediate word value
imm32	32-bit immediate doubleword value
reg/mem8	8-bit operand, which can be an 8-bit general register or memory byte
reg/mem16	16-bit operand, which can be a 16-bit general register or memory word
reg/mem32	32-bit operand, which can be a 32-bit general register or memory doubleword
mem	An 8-, 16-, or 32-bit memory operand

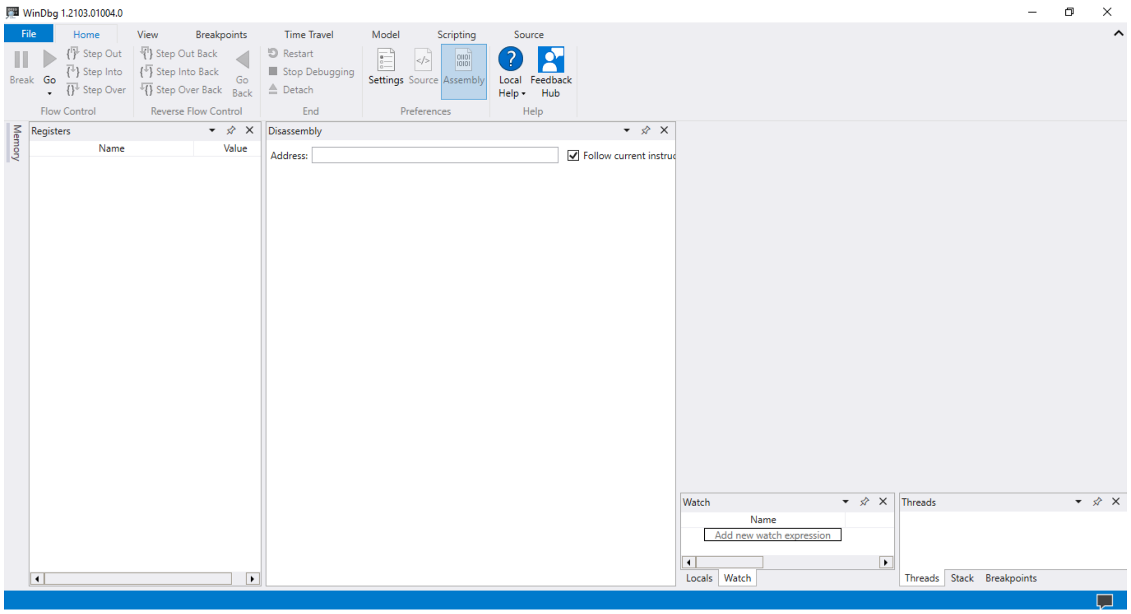
32-Bit	16-Bit	8-Bit (High)	8-Bit (Low)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

32-Bit	16-Bit
ESI	SI
EDI	DI
EBP	BP
ESP	SP

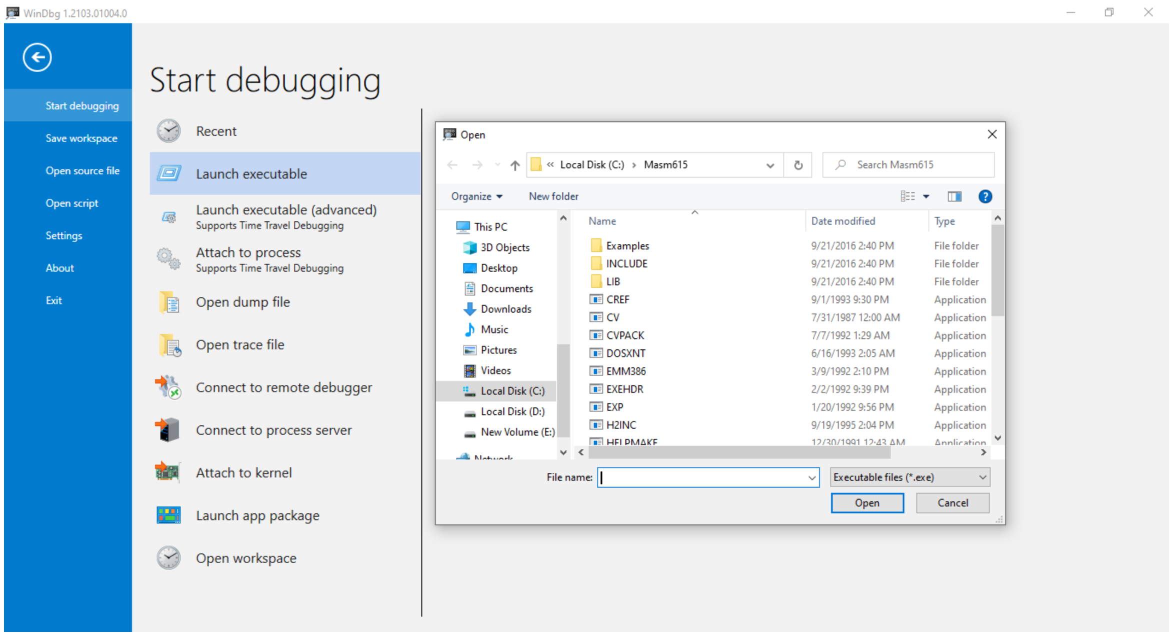
GETTING STARTED WITH WIN DEBUG V1.2:

As in case of MASM we are dealing with 32-bit protected mode using Kip Irvine library, therefore for 32-bit we have to use a windows debugger called Win Debug version 1.2. You can download it by clicking the following link.
https://www.microsoft.com/en-us/p/windbg-preview/9pgjgd53tn86?activetab=pivot:overviewtab

1. After installing Win debug application from Microsoft store when opened it will look like.

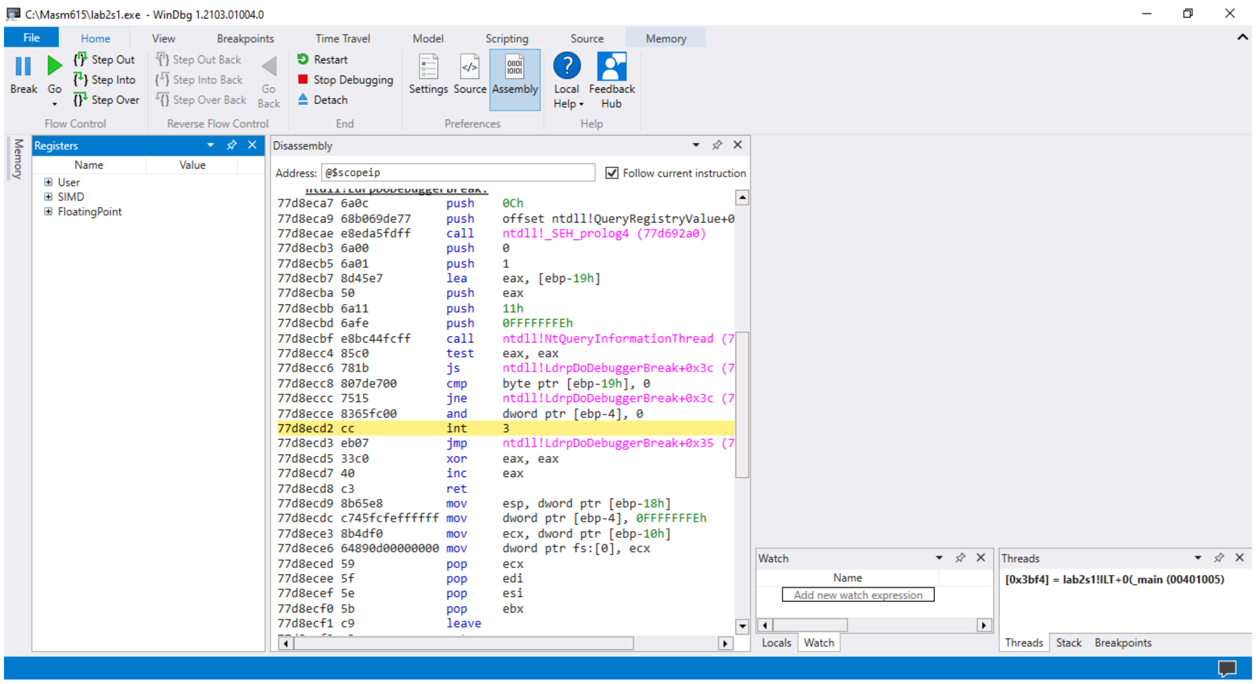


2. Click file, click launch executable and then input the executable file from the MASM directory



WinDbg 1.2103.01004.0
Start debugging
Save workspace
Open source file
Open script
Settings
About
Exit
Start debugging
Recent
Launch executable
Launch executable (advanced)
Supports Time Travel Debugging
Attach to process
Supports Time Travel Debugging
Open dump file
Open trace file
Connect to remote debugger
Connect to process server
Attach to kernel
Launch app package
Open workspace
Open
Local Disk (C:) › Masm615
Search Masm615
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
New Volume (E:)
Network
Name
Date modified
Type
Examples 9/21/2016 2:40 PM File folder
INCLUDE 9/21/2016 2:40 PM File folder
LIB 9/21/2016 2:40 PM File folder
CREF 9/1/1993 9:30 PM Application
CV 7/31/1987 12:00 AM Application
CVPACK 7/7/1992 1:29 AM Application
DOSXNT 6/16/1993 2:05 AM Application
EMM386 3/9/1992 2:10 PM Application
EXEHDR 2/2/1992 9:39 PM Application
EXP 1/20/1992 9:56 PM Application
H2INC 9/19/1995 2:04 PM Application
HELPMAKE 12/30/1991 12:43 AM Application
File name:
Executable files (*.exe)
Open
Cancel

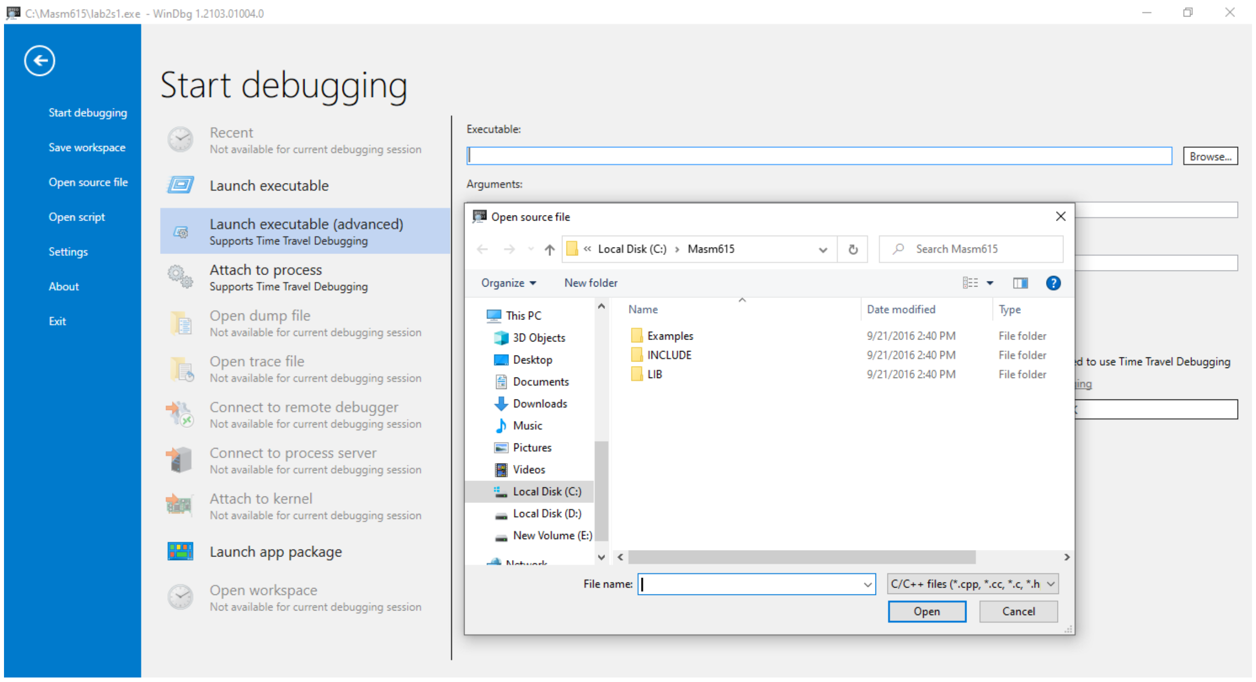
3. After launching executable file following windows will appear in Win debug.



```
77d8eca7 6a0c            push    0Ch
77d8eca9 68b069de77      push    offset ntdll!QueryRegistryValue+0
77d8ecae e8eda5fdff      call    ntdll!_SEH_prolog4 (77d692a0)
77d8ecb3 6a00            push    0
77d8ecb5 6a01            push    1
77d8ecb7 8d45e7          lea     eax, [ebp-19h]
77d8ecba 50              push    eax
77d8ecbb 6a11            push    11h
77d8ecbd 6afe            push    0FFFFFFFEh
77d8ecbf e8bc44fcff      call    ntdll!NtQueryInformationThread (7
77d8ecc4 85c0            test    eax, eax
77d8ecc6 781b            js      ntdll!LdrpDoDebuggerBreak+0x3c (7
77d8ecc8 807de700        cmp     byte ptr [ebp-19h], 0
77d8eccc 7515            jne     ntdll!LdrpDoDebuggerBreak+0x3c (7
77d8ecce 8365fc00        and     dword ptr [ebp-4], 0
77d8ecd2 cc              int     3
77d8ecd3 eb07            jmp     ntdll!LdrpDoDebuggerBreak+0x35 (7
77d8ecd5 33c0            xor     eax, eax
77d8ecd7 40              inc     eax
77d8ecd8 c3              ret
77d8ecd9 8b65e8          mov     esp, dword ptr [ebp-18h]
77d8ecdc c745fcfeffffff  mov     dword ptr [ebp-4], 0FFFFFFFEh
77d8ece3 8b4df0          mov     ecx, dword ptr [ebp-10h]
77d8ece6 64890d00000000  mov     dword ptr fs:[0], ecx
77d8eced 59              pop     ecx
77d8ecee 5f              pop     edi
77d8ecef 5e              pop     esi
77d8ecf0 5b              pop     ebx
77d8ecf1 c9              leave
```

[0x3bf4] = lab2s1!ILT+0(_main (00401005)

4. Now open source file of yours with .asm extension that you had saved in MASM directory.



Start debugging

Recent
Not available for current debugging session

Launch executable

Launch executable (advanced)
Supports Time Travel Debugging

Attach to process
Supports Time Travel Debugging

Open dump file
Not available for current debugging session

Open trace file
Not available for current debugging session

Connect to remote debugger
Not available for current debugging session

Connect to process server
Not available for current debugging session

Attach to kernel
Not available for current debugging session

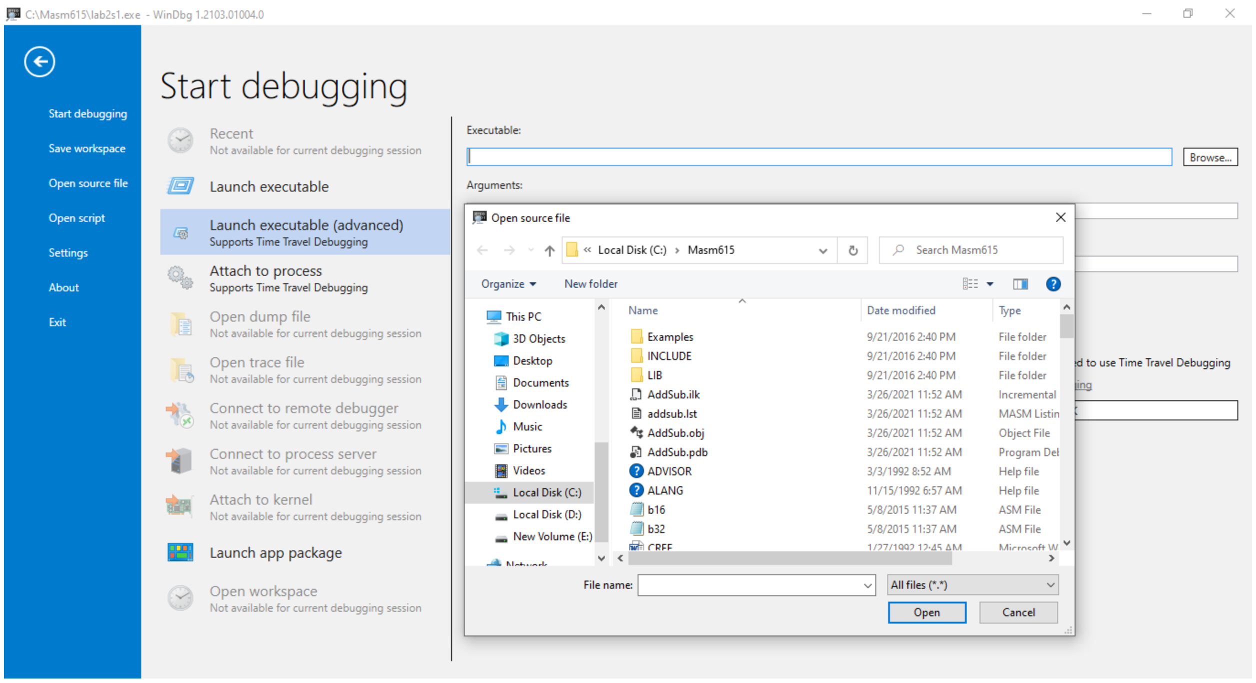
Launch app package

Open workspace
Not available for current debugging session

Name	Date modified	Type
Examples	9/21/2016 2:40 PM	File folder
INCLUDE	9/21/2016 2:40 PM	File folder
LIB	9/21/2016 2:40 PM	File folder

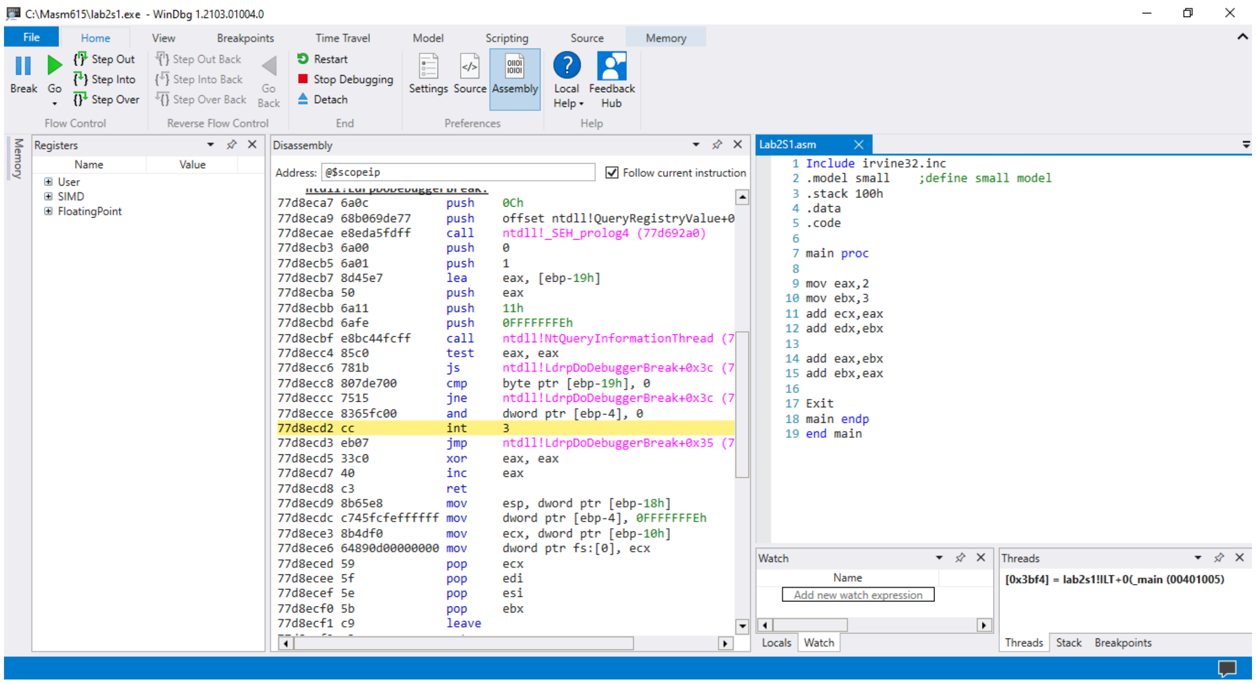
d to use Time Travel Debugging

5. Select all files instead of C/C++ and select your .asm file from the directory.


C:\Masm615\lab2s1.exe - WinDbg 1.2103.01004.0
Start debugging
Start debugging
Save workspace
Open source file
Open script
Settings
About
Exit
Recent
Not available for current debugging session
Launch executable
Launch executable (advanced)
Supports Time Travel Debugging
Attach to process
Supports Time Travel Debugging
Open dump file
Not available for current debugging session
Open trace file
Not available for current debugging session
Connect to remote debugger
Not available for current debugging session
Connect to process server
Not available for current debugging session
Attach to kernel
Not available for current debugging session
Launch app package
Open workspace
Not available for current debugging session
Executable:
Browse...
Arguments:
d to use Time Travel Debugging
Open source file
« Local Disk (C:) › Masm615
Search Masm615
Organize
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
New Volume (E:)
Network
File name:
All files (*.*)
Open
Cancel

Name	Date modified	Type
Examples	9/21/2016 2:40 PM	File folder
INCLUDE	9/21/2016 2:40 PM	File folder
LIB	9/21/2016 2:40 PM	File folder
AddSub.ilk	3/26/2021 11:52 AM	Incremental
addsub.lst	3/26/2021 11:52 AM	MASM Listin
AddSub.obj	3/26/2021 11:52 AM	Object File
AddSub.pdb	3/26/2021 11:52 AM	Program Del
ADVISOR	3/3/1992 8:52 AM	Help file
ALANG	11/15/1992 6:57 AM	Help file
b16	5/8/2015 11:37 AM	ASM File
b32	5/8/2015 11:37 AM	ASM File
CREF	1/27/1992 12:45 AM	Microsoft W

6. After opening the source file, a new window will appear in which your code is shown.

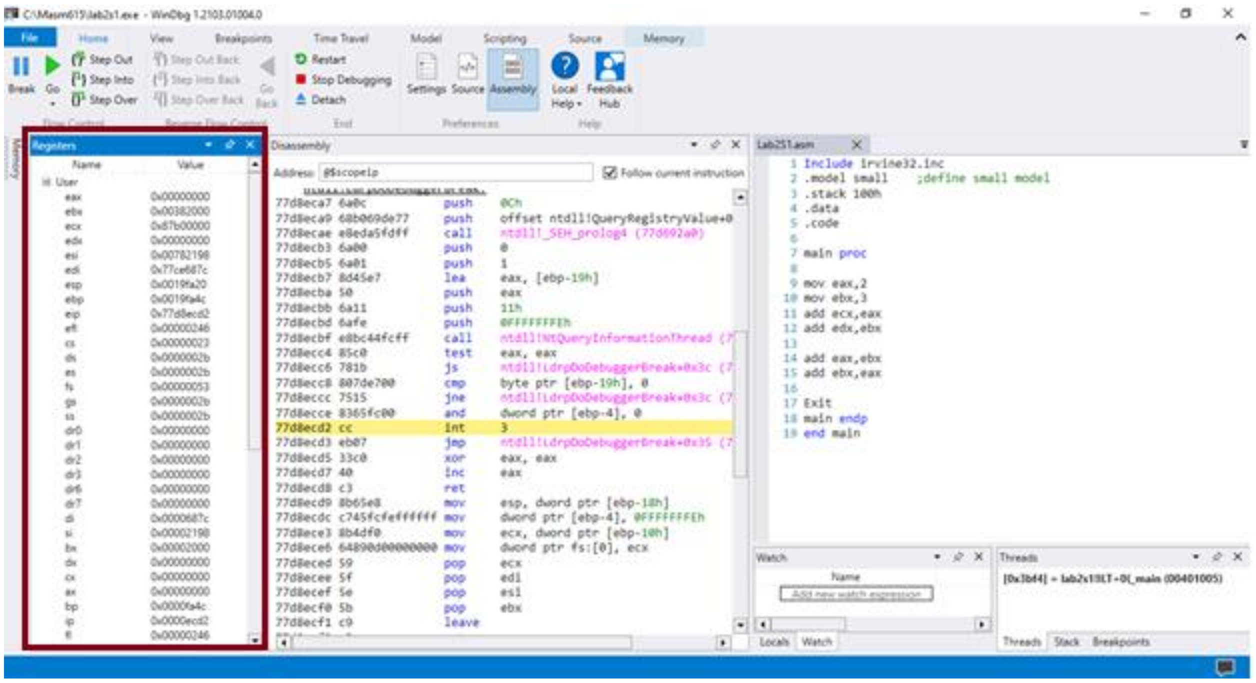


```
77d8eca7 6a0c            push    0Ch
77d8eca9 68b069de77      push    offset ntdll!QueryRegistryValue+0
77d8ecae e8eda5fdff      call    ntdll!_SEH_prolog4 (77d692a0)
77d8ecb3 6a00            push    0
77d8ecb5 6a01            push    1
77d8ecb7 8d45e7          lea     eax, [ebp-19h]
77d8ecba 50              push    eax
77d8ecbb 6a11            push    11h
77d8ecbd 6afe            push    0FFFFFFFEh
77d8ecbf e8bc44fcff      call    ntdll!NtQueryInformationThread (7
77d8ecc4 85c0            test    eax, eax
77d8ecc6 781b            js      ntdll!LdrpDoDebuggerBreak+0x3c (7
77d8ecc8 807de700        cmp     byte ptr [ebp-19h], 0
77d8eccc 7515            jne     ntdll!LdrpDoDebuggerBreak+0x3c (7
77d8ecce 8365fc00        and     dword ptr [ebp-4], 0
77d8ecd2 cc              int     3
77d8ecd3 eb07            jmp     ntdll!LdrpDoDebuggerBreak+0x35 (7
77d8ecd5 33c0            xor     eax, eax
77d8ecd7 40              inc     eax
77d8ecd8 c3              ret
77d8ecd9 8b65e8          mov     esp, dword ptr [ebp-18h]
77d8ecdc c745fcfeffffff  mov     dword ptr [ebp-4], 0FFFFFFFEh
77d8ece3 8b4df0          mov     ecx, dword ptr [ebp-10h]
77d8ece6 64890d00000000  mov     dword ptr fs:[0], ecx
77d8eced 59              pop     ecx
77d8ecee 5f              pop     edi
77d8ecef 5e              pop     esi
77d8ecf0 5b              pop     ebx
77d8ecf1 c9              leave
```

```
Include irvine32.inc
.model small     ;define small model
.stack 100h
.data
.code

main proc

mov eax,2
mov ebx,3
add ecx,eax
add edx,ebx

add eax,ebx
add ebx,eax

Exit
main endp
end main
```

[0x3bf4] = lab2s1!ILT+0(_main (00401005)

7. Now expand the user tab inside register window by clicking on plus with it.



C:\Masm615\lab2s1.exe - WinDbg 1.2103.01004.0

Registers

Name	Value
User	
eax	0x00000000
ebx	0x00382000
ecx	0xd7b00000
edx	0x00000000
esi	0x00782198
edi	0x77ce687c
esp	0x0019fa20
ebp	0x0019fa4c
eip	0x77d8ecd2
efl	0x00000246
cs	0x00000023
ds	0x0000002b
es	0x0000002b
fs	0x00000053
gs	0x0000002b
ss	0x0000002b
dr0	0x00000000
dr1	0x00000000
dr2	0x00000000
dr3	0x00000000
dr6	0x00000000
dr7	0x00000000
di	0x0000687c
si	0x00002198
bx	0x00002000
dx	0x00000000
cx	0x00000000
ax	0x00000000
bp	0x0000fa4c
ip	0x0000ecd2
fl	0x00000246

Disassembly — Address: @$scopeip — Follow current instruction

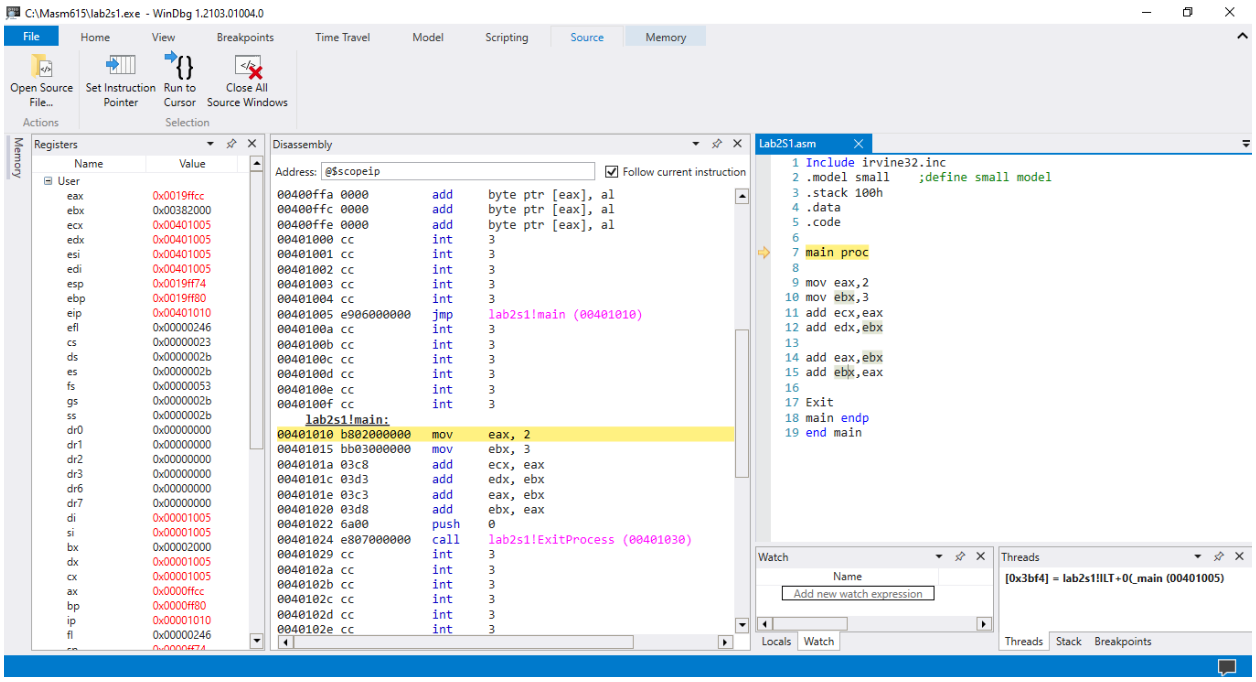
```
77d8eca7 6a0c            push    0Ch
77d8eca9 68b069de77      push    offset ntdll!QueryRegistryValue+0
77d8ecae e8eda5fdff      call    ntdll!_SEH_prolog4 (77d692a0)
77d8ecb3 6a00            push    0
77d8ecb5 6a01            push    1
77d8ecb7 8d45e7          lea     eax, [ebp-19h]
77d8ecba 50              push    eax
77d8ecbb 6a11            push    11h
77d8ecbd 6afe            push    0FFFFFFFEh
77d8ecbf e8bc44fcff      call    ntdll!NtQueryInformationThread (7
77d8ecc4 85c0            test    eax, eax
77d8ecc6 781b            js      ntdll!LdrpDoDebuggerBreak+0x3c (7
77d8ecc8 807de700        cmp     byte ptr [ebp-19h], 0
77d8eccc 7515            jne     ntdll!LdrpDoDebuggerBreak+0x3c (7
77d8ecce 8365fc00        and     dword ptr [ebp-4], 0
77d8ecd2 cc              int     3
77d8ecd3 eb07            jmp     ntdll!LdrpDoDebuggerBreak+0x35 (7
77d8ecd5 33c0            xor     eax, eax
77d8ecd7 40              inc     eax
77d8ecd8 c3              ret
77d8ecd9 8b65e8          mov     esp, dword ptr [ebp-18h]
77d8ecdc c745fcfeffffff  mov     dword ptr [ebp-4], 0FFFFFFFEh
77d8ece3 8b4df0          mov     ecx, dword ptr [ebp-10h]
77d8ece6 64890d00000000  mov     dword ptr fs:[0], ecx
77d8eced 59              pop     ecx
77d8ecee 5f              pop     edi
77d8ecef 5e              pop     esi
77d8ecf0 5b              pop     ebx
77d8ecf1 c9              leave
```

Lab2S1.asm

```
Include irvine32.inc
.model small    ;define small model
.stack 100h
.data
.code

main proc

mov eax,2
mov ebx,3
add ecx,eax
add edx,ebx

add eax,ebx
add ebx,eax

Exit
main endp
end main
```

Threads: [0x3b44] - lab2s1!ILT+0(_main (00401005)

8. Now click on source, left click inside your script file and then on top left side left click on “Run to Cursor”. After you do that a yellow arrow will appear on the corresponding line.



C:\Masm615\lab2s1.exe - WinDbg 1.2103.01004.0

Registers

Name	Value
User	
eax	0x0019ffcc
ebx	0x00382000
ecx	0x00401005
edx	0x00401005
esi	0x00401005
edi	0x00401005
esp	0x0019ff74
ebp	0x0019ff80
eip	0x00401010
efl	0x00000246
cs	0x00000023
ds	0x0000002b
es	0x0000002b
fs	0x00000053
gs	0x0000002b
ss	0x0000002b
dr0	0x00000000
dr1	0x00000000
dr2	0x00000000
dr3	0x00000000
dr6	0x00000000
dr7	0x00000000
di	0x00001005
si	0x00001005
bx	0x00002000
dx	0x00001005
cx	0x00001005
ax	0x0000ffcc
bp	0x0000ff80
ip	0x00001010
fl	0x00000246

Disassembly — Address: @$scopeip — Follow current instruction

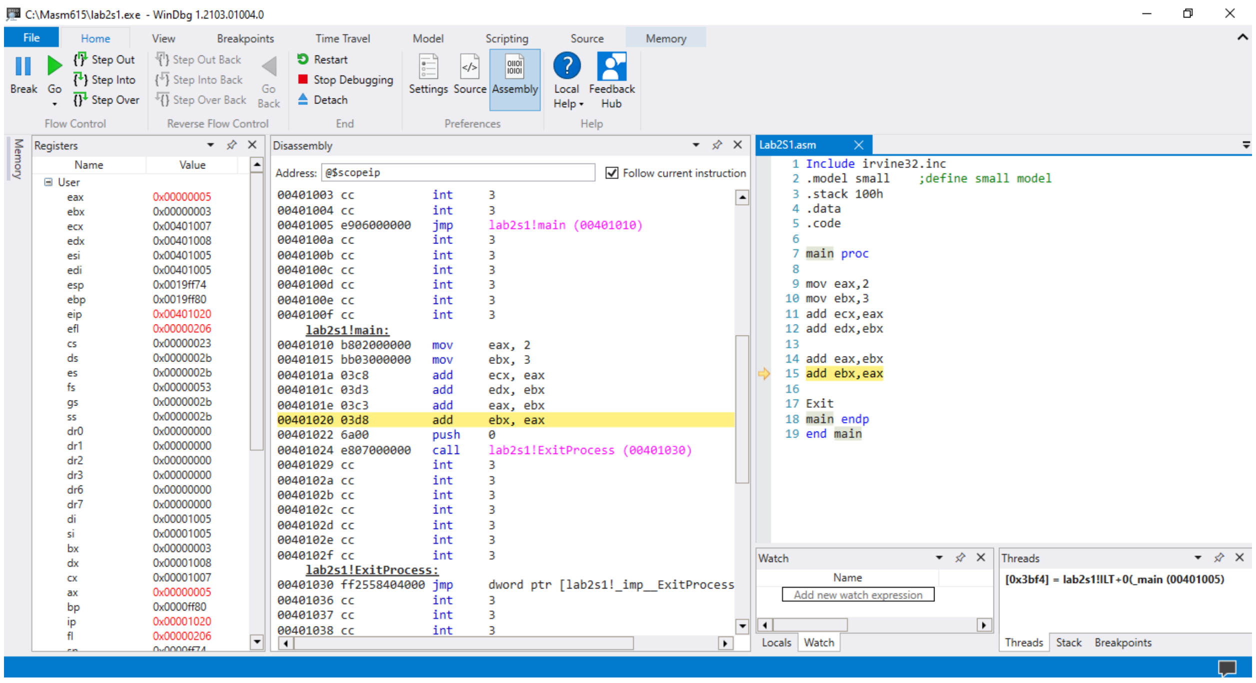
```
00400ffa 0000        add     byte ptr [eax], al
00400ffc 0000        add     byte ptr [eax], al
00400ffe 0000        add     byte ptr [eax], al
00401000 cc          int     3
00401001 cc          int     3
00401002 cc          int     3
00401003 cc          int     3
00401004 cc          int     3
00401005 e906000000  jmp     lab2s1!main (00401010)
0040100a cc          int     3
0040100b cc          int     3
0040100c cc          int     3
0040100d cc          int     3
0040100e cc          int     3
0040100f cc          int     3
    lab2s1!main:
00401010 b802000000  mov     eax, 2
00401015 bb03000000  mov     ebx, 3
0040101a 03c8        add     ecx, eax
0040101c 03d3        add     edx, ebx
0040101e 03c3        add     eax, ebx
00401020 03d8        add     ebx, eax
00401022 6a00        push    0
00401024 e807000000  call    lab2s1!ExitProcess (00401030)
00401029 cc          int     3
0040102a cc          int     3
0040102b cc          int     3
0040102c cc          int     3
0040102d cc          int     3
0040102e cc          int     3
```

Lab2S1.asm

```
Include irvine32.inc
.model small    ;define small model
.stack 100h
.data
.code

main proc

mov eax,2
mov ebx,3
add ecx,eax
add edx,ebx

add eax,ebx
add ebx,eax

Exit
main endp
end main
```

Threads: [0x3bf4] - lab2s1!ILT+0(_main (00401005)

9. Now press F10 button to do step wise execution of instructions of the code.



Registers

Name	Value
User	
eax	0x00000005
ebx	0x00000003
ecx	0x00401007
edx	0x00401008
esi	0x00401005
edi	0x00401005
esp	0x0019ff74
ebp	0x0019ff80
eip	0x00401020
efl	0x00000206
cs	0x00000023
ds	0x0000002b
es	0x0000002b
fs	0x00000053
gs	0x0000002b
ss	0x0000002b
dr0	0x00000000
dr1	0x00000000
dr2	0x00000000
dr3	0x00000000
dr6	0x00000000
dr7	0x00000000
di	0x00001005
si	0x00001005
bx	0x00000003
dx	0x00001008
cx	0x00001007
ax	0x00000005
bp	0x0000ff80
ip	0x00001020
fl	0x00000206

Address: @$scopeip ✓ Follow current instruction

```
00401003 cc          int     3
00401004 cc          int     3
00401005 e906000000  jmp     lab2s1!main (00401010)
0040100a cc          int     3
0040100b cc          int     3
0040100c cc          int     3
0040100d cc          int     3
0040100e cc          int     3
0040100f cc          int     3
    lab2s1!main:
00401010 b802000000  mov     eax, 2
00401015 bb03000000  mov     ebx, 3
0040101a 03c8        add     ecx, eax
0040101c 03d3        add     edx, ebx
0040101e 03c3        add     eax, ebx
00401020 03d8        add     ebx, eax
00401022 6a00        push    0
00401024 e807000000  call    lab2s1!ExitProcess (00401030)
00401029 cc          int     3
0040102a cc          int     3
0040102b cc          int     3
0040102c cc          int     3
0040102d cc          int     3
0040102e cc          int     3
0040102f cc          int     3
    lab2s1!ExitProcess:
00401030 ff2558404000 jmp    dword ptr [lab2s1!_imp__ExitProcess
00401036 cc          int     3
00401037 cc          int     3
00401038 cc          int     3
```

```
Include irvine32.inc
.model small    ;define small model
.stack 100h
.data
.code

main proc

mov eax,2
mov ebx,3
add ecx,eax
add edx,ebx

add eax,ebx
add ebx,eax

Exit
main endp
end main
```

[0x3bf4] = lab2s1!ILT+0(_main (00401005)

SAMPLE CODE:

PROGRAM DESCRIPTION:

Start with moving values in registers using Mov instruction and then using Add instruction to add the register values.

```
Include irvine32.inc
.model small    ;define small model
.stack 100h
.data
.code

main proc

mov eax,2
mov ebx,3

call dumpregs

add ecx,eax
add edx,ebx

call dumpregs
```

```
add eax,ebx
add ebx,eax

call dumpregs

Exit      ;termination of program
main endp
end main
```

1. Note down the contents of registers EAX, EBX, ECX and EDX as displayed by the program

2. Check the contents of registers against the instructions.

3. Do the contents of register ECX and EDX match the expected result?

4. If not, what step needs to be taken?

EXERCISES:

1. Write an assembly code to display the data of registers before the termination command than edit your code so that the data will be displayed on new line.
2. Write an assembly code for example code discussed above but now display all the values of destination register after every instruction i.e Mov, Add etc
3. Write an assembly code to user input any small alphabet than print its capital version on the screen (Hint: use ASCII table)