

Prog # 1

Demetrius Johnson

CIS 310 (Yoon)

June 2, 2021

1. Build/run the program without debugging.
2. Run the program in the debugging mode and add the register window to a debugging session (see Fig. 3.5).

Notice, I added another variable to store the result of the arithmetic operation, called **FinalResult**; also in this screenshot I show what the value of all of my registers are after all operations have been completed:

The screenshot shows the Visual Studio IDE with an assembly program open. The program is titled "Program#1-CIS-310...etriusJohnson.asm". The code includes comments about the course and author, followed by standard assembly directives like `.386`, `.model flat,stdcall`, and `.stack 4096`. It defines a `main` procedure that moves values from variables `A`, `B`, `C1`, and `D` into registers `EAX`, `EBX`, `ECX`, and `EDX` respectively. It then performs an addition `add eax, B`, adds `D` to `ECX`, and subtracts `ECX` from `EAX` to store the final result in `FinalResult`. The program ends by calling `ExitProcess`.

At the bottom, the "Registers" window is open, showing the state of the CPU registers. The `EAX` register contains `00000023`, `EBX` contains `00000014`, `ECX` contains `0000000F`, `EDX` contains `00000005`, `ESI` contains `00401005`, and `EDI` contains `00401005`. The `EIP` register contains `0040103A`. The `OV`, `UP`, `EI`, `PL`, `ZR`, `AC`, `PE`, and `CY` flags are all set to 0.

The "Call Stack" window on the right shows the current call stack with the following entries:

Name
Project.exe!main() Line 35
[External Code]
ntdll.dll![] (Frames below may be incorrect and/or missing, no symbols loaded for ntdll.dll)

The bottom status bar shows the "Registers" tab is active, along with "Autos", "Locals", "Threads", "Modules", and "Watch 1".

In this screenshot, I demonstrate what is stored in my variables and registers:

```

ogram#1-CIS-310...etriusJohnson.asm -b X
1  ;CIS-350 SUMMER 2021 WITH PROFESSOR DAVID YOON
2  ;Name: Demetrius Johnson
3  ;Date: 5-30-21
4  ;Program Description: Introduction to MASM; define variables and use registers to complete addition and subtraction operations
5
6
7  .386
8  .model flat,stdcall
9  .stack 4096
10 ExitProcess proto,dwExitCode:dword
11
12 .DATA
13
14 A  DWORD 30d
15 B  DWORD 20d
16 C1 DWORD 10d
17 D  DWORD 5d
18 FinalResult DWORD ? ; store the final result of my operation in this location (variable)
19
20 .CODE
21 main proc
22
23 mov eax, A
24 mov ebx, B
25 mov ecx, C1
26 mov edx, D
27
28 ;below, we need to perform the operation: A = (A + B) - (C1 + D)
29
30 add eax, B          ; eax = A + B
31 add ecx, D          ; ecx = C1 + D
32 sub eax, ecx        ; eax = eax - ecx == (A + B) - (C1 + D)
33 mov FinalResult, eax ; store final result in a variable
34
35 invoke ExitProcess,0 ; 0 seconds elapsed
36 main endp
37 end main

```

% No issues found Ln: 35 Ch: 1

atch 1 Search (Ctrl+E) Search Depth: 3

Name	Value	Type
wordVal	identifier "wordVal" is undefined	
ax	35	unsigned short
dx	5	unsigned short
A	30	unsigned long
FinalResult	35	unsigned long

Add item to watch

utos Locals Registers Threads Modules Watch 1

Call Stack

Name
Project.exe!main() Line 35
[External Code]
ntdll.dll![] [Frames below may be incorrect and/or missing, no symbols loaded for ntdll.dll]

Call Stack Breakpoints Exception Settings Output

3. Generate the listing file like the one in Fig. 3.8.

In the next two screenshots, you see the listing file generated from my program. I read from the Irvine book that the memory locations correspond to some starting address relative to wherever the operating system begins assigning memory; as an example, the start memory location 00000000 (32-bit address represented as 8 hex digits) is really some other starting location value in memory, and thus every value after the start is also relative to that location. They simply generate the listing file starting from a relative 0-address location so it is easier for us programmers to identify and analyze memory usage and allocation that occurs for and during the program:

```
File Edit Format View Help
Microsoft (R) Macro Assembler Version 14.28.29333.0      06/02/21 14:40:23
Program#1-CIS-310-DemetriusJohnson.asm                Page 1 - 1

;CIS-350 SUMMER 2021 WITH PROFESSOR DAVID YOON
;Name: Demetrius Johnson
;Date: 5-30-21
;Program Description: Introduction to MASM; define variables and use registers to complete addition and subtraction operations

.386
.model flat,stdcall
.stack 4096
ExitProcess proto,dwExitCode:dword

00000000      .DATA

00000000 0000001E      A      DWORD 30d
00000004 00000014      B      DWORD 20d
00000008 0000000A      C1     DWORD 10d
0000000C 00000005      D      DWORD 5d
00000010 00000000      FinalResult DWORD ?      ; store the final result of my operation in this location (variable)

00000000      .CODE
00000000      main proc

00000000 A1 00000000 R    mov eax, A
00000005 8B 1D 00000004 R  mov ebx, B
0000000B 8B 0D 00000008 R  mov ecx, C1
00000011 8B 15 0000000C R  mov edx, D

;below, we need to perform the operation: A = (A + B) - (C1 + D)

00000017 03 05 00000004 R    add eax, B      ; eax = A + B
0000001D 03 0D 0000000C R    add ecx, D      ; ecx = C1 + D
00000023 2B C1             sub eax, ecx      ; eax = eax - ecx == (A + B) - (C1 + D)
00000025 A3 00000010 R      mov FinalResult, eax      ; store final result in a variable

;invoke ExitProcess,0

0000002A 6A 00      *      push +00000000h
0000002C E8 00000000 E *      call ExitProcess
00000031             main endp

end main
^Microsoft (R) Macro Assembler Version 14.28.29333.0      06/02/21 14:40:23
Program#1-CIS-310-DemetriusJohnson.asm                Symbols 2 - 1
```

CIS 310 – YOON – SUMMER 21 – PROGRAM #1 – DEMETRIUS JOHNSON

File Edit Format View Help

Program#1-CIS-310-DemetriusJohnson.asm

Symbols 2 - 1

Segments and Groups:

N a m e	Size	Length	Align	Combine	Class
FLAT					GROUP
STACK	32 Bit	00001000	DWord	Stack	'STACK'
_DATA	32 Bit	00000014	DWord	Public	'DATA'
_TEXT	32 Bit	00000031	DWord	Public	'CODE'

Procedures, parameters, and locals:

N a m e	Type	Value	Attr
ExitProcess	P Near	00000000	FLAT Length= 00000000 External STDCALL
main	P Near	00000000	_TEXT Length= 00000031 Public STDCALL

Symbols:

N a m e	Type	Value	Attr
@CodeSize	Number	00000000h	
@DataSize	Number	00000000h	
@Interface	Number	00000003h	
@Model	Number	00000007h	
@code	Text	_TEXT	
@data	Text	FLAT	
@fardata?	Text	FLAT	
@fardata	Text	FLAT	
@stack	Text	FLAT	
A	DWord	00000000	_DATA
B	DWord	00000004	_DATA
C1	DWord	00000008	_DATA
D	DWord	0000000C	_DATA
FinalResult	DWord	00000010	_DATA

0 Warnings
0 Errors

<

Upload the screenshots from 1, 2 and 3.