CSCI -112
Introduction to computer Systems
LAB

Instructor: Santanu Banerjee

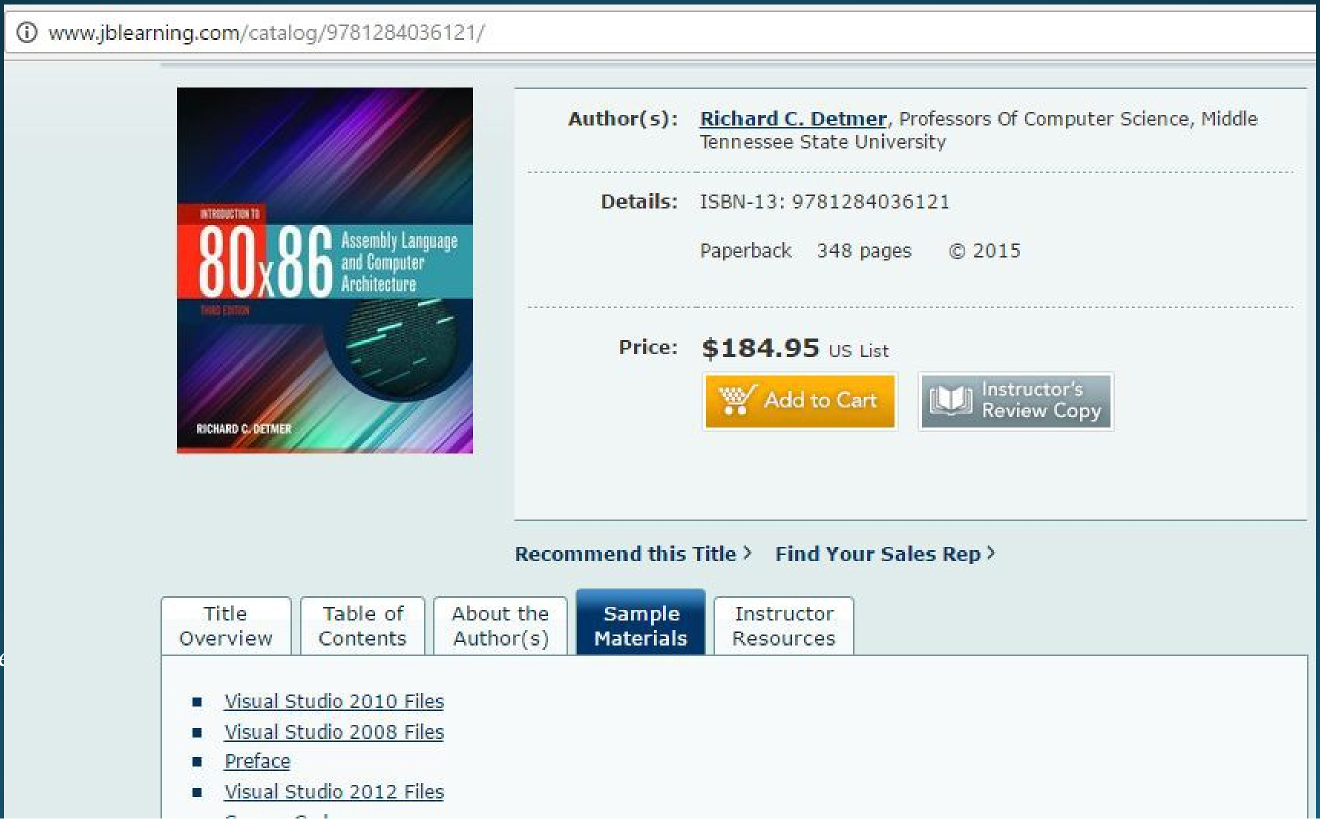
64-bit Examples

Courtesy: UMBC and JBLearning

5 bonus assignment points for completion and submission.

Framework Packages

- Download from the publisher's website.



www.jblearning.com/catalog/9781284036121/

Author(s): Richard C. Detmer, Professors Of Computer Science, Middle Tennessee State University

Details: ISBN-13: 9781284036121

Paperback 348 pages © 2015

Price: $184.95 US List

- Visual Studio 2010 Files
- Visual Studio 2008 Files
- Preface
- Visual Studio 2012 Files

These

Framework Packages

Name	Use
console32	• Produces 32-bit programs, even in a 64-bit operating system • No I/O provided • Debugger used to see register and memory contents
console64	• Produces 64-bit programs—64-bit operating system required • No I/O provided • Debugger used to see register and memory contents
windows32	• Produces 32-bit programs, even in a 64-bit operating system • Simple I/O using macros defined in the package • Debugger available to see register and memory contents
windows64	• Produces 64-bit programs—64-bit operating system required • Simple I/O using macros defined in the package • Debugger available to see register and memory contents

Use the **windows32** framework package for this exercise

These packages are used with Visual Studio for programs in this text

Exercise

1. Read Section 3.7 (Chapter 3) from the text book.
2. Read all the slides in this document.
3. Build and run the Console64 app just like the console32 app after applying the changes for 64 bit mode.

64-bit Differences

- “Direct” memory addressing is actually RIP relative: the 32-bit offset stored in the instruction is added to RIP to get the operand address.
- Extra code is required in *windows64* programs

```
sub rsp,120  ; reserve stack space for
 MainProc
          ...
add rsp, 120 ; restore stack
```

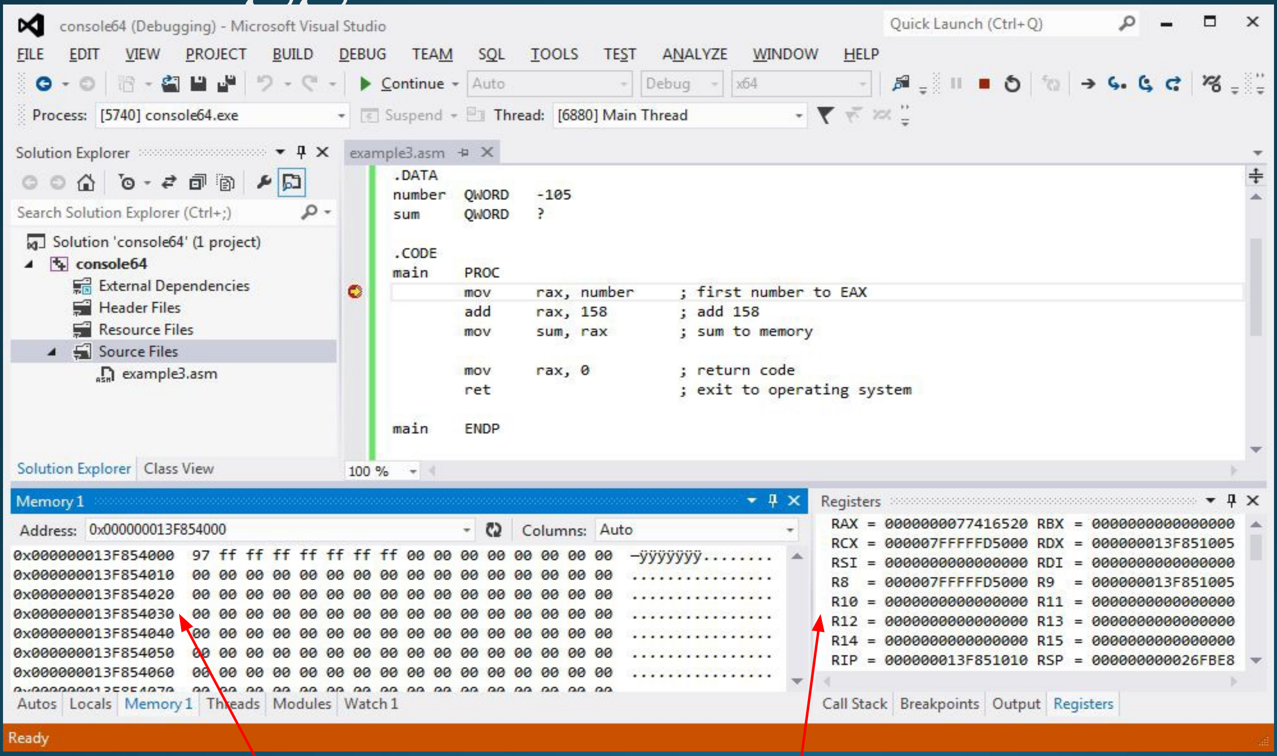
console64 Example

- Use the solution under console64 folder
- Similar to *console32*, but fewer directives
  - Not required: .586, .MODEL FLAT, .STACK
- Data segment uses QWORD instead of DWORD
- Code segment uses RAX instead of EAX
- Exits by storing 0 in RAX followed by ret

console64 Example

```
; Example assembly language program
.DATA
number  QWORD   -105
sum     QWORD   ?
.CODE
main    PROC
        mov     rax, number
        add     rax, <your student ID>
        mov     sum, rax
        mov     rax, 0
        ret
main    ENDP
END
```

Debugger


console64 (Debugging) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP
Continue
Auto
Debug
x64
Process: [5740] console64.exe
Suspend
Thread: [6880] Main Thread
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'console64' (1 project)
console64
External Dependencies
Header Files
Resource Files
Source Files
example3.asm
Solution Explorer
Class View
example3.asm
100 %
Memory 1
Columns: Auto
Registers
Autos Locals Memory 1 Threads Modules Watch 1
Call Stack Breakpoints Output Registers
Ready

```
.DATA
number  QWORD   -105
sum     QWORD   ?

.CODE
main    PROC
        mov     rax, number     ; first number to EAX
        add     rax, 158        ; add 158
        mov     sum, rax        ; sum to memory

        mov     rax, 0          ; return code
        ret                     ; exit to operating system

main    ENDP
```

Address: 0x000000013F854000

```
0x000000013F854000  97 ff ff ff ff ff ff ff 00 00 00 00 00 00 00 00  —ÿÿÿÿÿÿÿ........
0x000000013F854010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x000000013F854020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x000000013F854030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x000000013F854040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x000000013F854050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0x000000013F854060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
RAX = 0000000077416520 RBX = 0000000000000000
RCX = 000007FFFFFD5000 RDX = 000000013F851005
RSI = 0000000000000000 RDI = 0000000000000000
R8  = 000007FFFFFD5000 R9  = 000000013F851005
R10 = 0000000000000000 R11 = 0000000000000000
R12 = 0000000000000000 R13 = 0000000000000000
R14 = 0000000000000000 R15 = 0000000000000000
RIP = 000000013F851010 RSP = 000000000026FBE8
```

64-bit addresses

64-bit registers

To earn points

- Create and add code to the 64-bit project as instructed.
- Debug through each line of code and examine the register and memory contents as code executes.
- **Submit** a screenshot of the visual studio debug session showing
  - (i)your code (ii) memory (iii) registers
  - (Like the picture in the previous slide)
- Play and experiment with the code.