CPSC 240: Computer Organization and Assembly Language
Assignment 02, Fall Semester 2024

CWID:__884614918________ Name:_Sean DeFrank_____________

Quiz Questions:

From the textbook "X86-64 Assembly Language Programming with Ubuntu," study quiz questions 8, 9, 10, and 11 on page 120. Students do not need to submit answers to the quiz questions as they are found in Appendix D of the textbook.

Programming:

1. Download the "CPSC-240 Assignment02.docx" document.
2. Design a 16-bit addition program "addition.asm", and use assembly language to realize the function of the following C++ instructions. NOTE: variable sizes and program functions should be equivalent to C/C++ instructions.

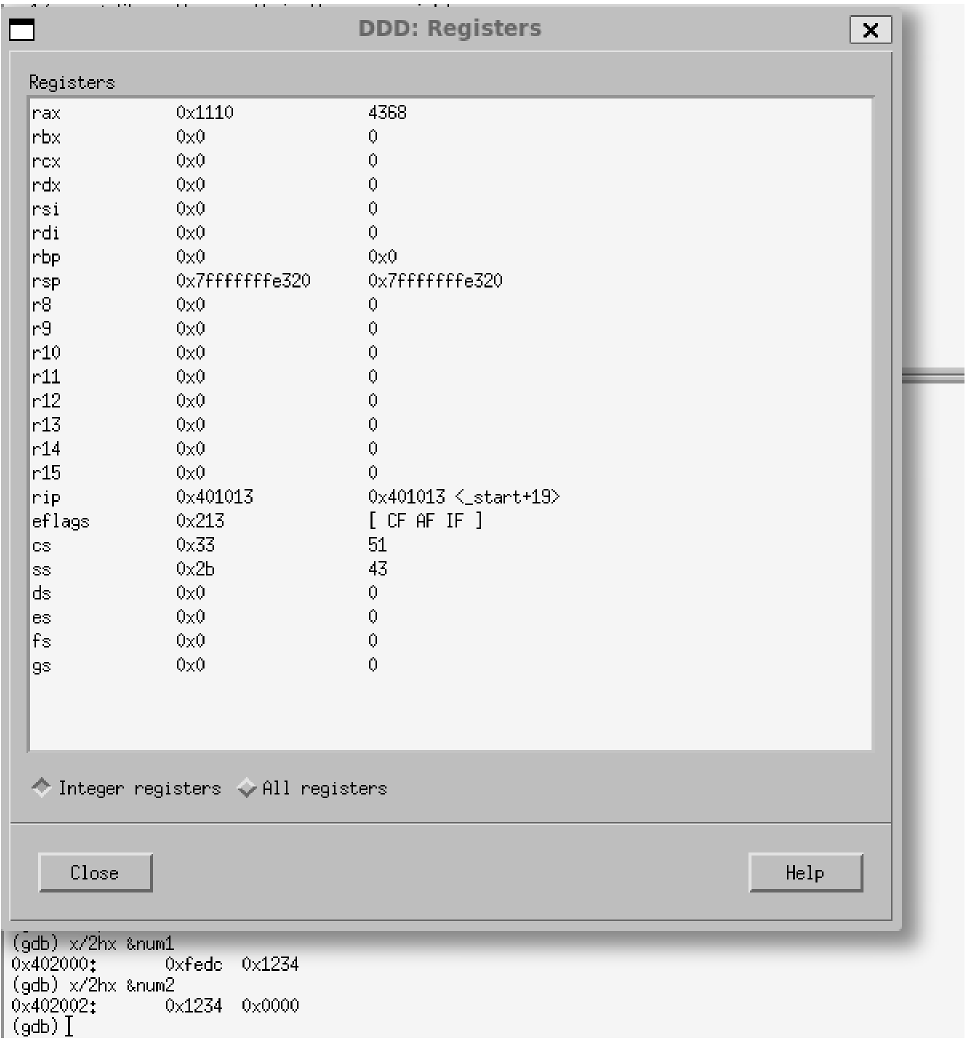
```
unsigned short num1 = 0xFEDC;          // use dw to declare 16-bit variable
unsigned short num2 = 0x1234;          // use dw to declare 16-bit variable
unsigned int sum = 0;                  // use dd to declare 32-bit variable
sum = int(num1 + num2);
```

3. Assemble the "addition.asm" file and link the "addition.o" file to get the "addition" executable file.
4. Run the "addition" file with the GDB debugger to display the simulation results of num1 and num2, as well as the simulation results of sum.
5. Insert source code (addition.asm) and simulation results (GDB debugger window) of the memory (num1, num2, and sum) in the document. Use calculator or hand calculation to verify the simulation results.
6. Design a 16-bit subtraction program "subtraction.asm", and use assembly language to realize the function of the following C++ instructions. NOTE: variable sizes and program functions should be equivalent to C/C++ instructions.

```
signed short num1 = 0x1234;            // use dw to declare 16-bit variable
signed short num2 = 0xFEDC;            // use dw to declare 16-bit variable
signed int dif = 0;                    // use dd to declare 32-bit variable
dif = int(num1 - num2);
```

7. Assemble the "subtraction.asm" file and link the "subtraction.o" file to get the "subtraction" executable file.
8. Run the "subtraction" file with the GDB debugger to display the simulation results of num1 and num2, as well as the simulation results of diff.
9. Insert source code (subtraction.asm) and simulation results (GDB debugger window) of the memory (num1, num2, and dif) in the document. Use calculator or hand calculation to verify the simulation results.
10. Save the file in pdf format and submit the pdf file to Canvas before the deadline.

```
[section .data
        num1 dw 0xFEDC      ;16-bit variable
        num2 dw 0x1234      ;16-bit variable
        sum    dd 0              ;32-bit variable for sum

section .text
        global _start

_start:

        mov ax, [num1]
        add ax, [num2]


        movzx eax, ax

        ; Storing sum
        mov [sum], eax

        ; Exit
        mov eax, 60
        xor edi, edi
        syscall]
```

[

```
rax            0x1110              4368
rbx            0x0                 0
rcx            0x0                 0
rdx            0x0                 0
rsi            0x0                 0
rdi            0x0                 0
rbp            0x0                 0x0
rsp            0x7fffffffe320      0x7fffffffe320
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0x0                 0
r13            0x0                 0
r14            0x0                 0
r15            0x0                 0
rip            0x401013            0x401013 <_start+19>
eflags         0x213               [ CF AF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
```

```
(gdb) x/2hx &num1
0x402000:       0xfedc  0x1234
(gdb) x/2hx &num2
0x402002:       0x1234  0x0000
(gdb) 
```

]

num1 = 0xFEDC, which is **65244**

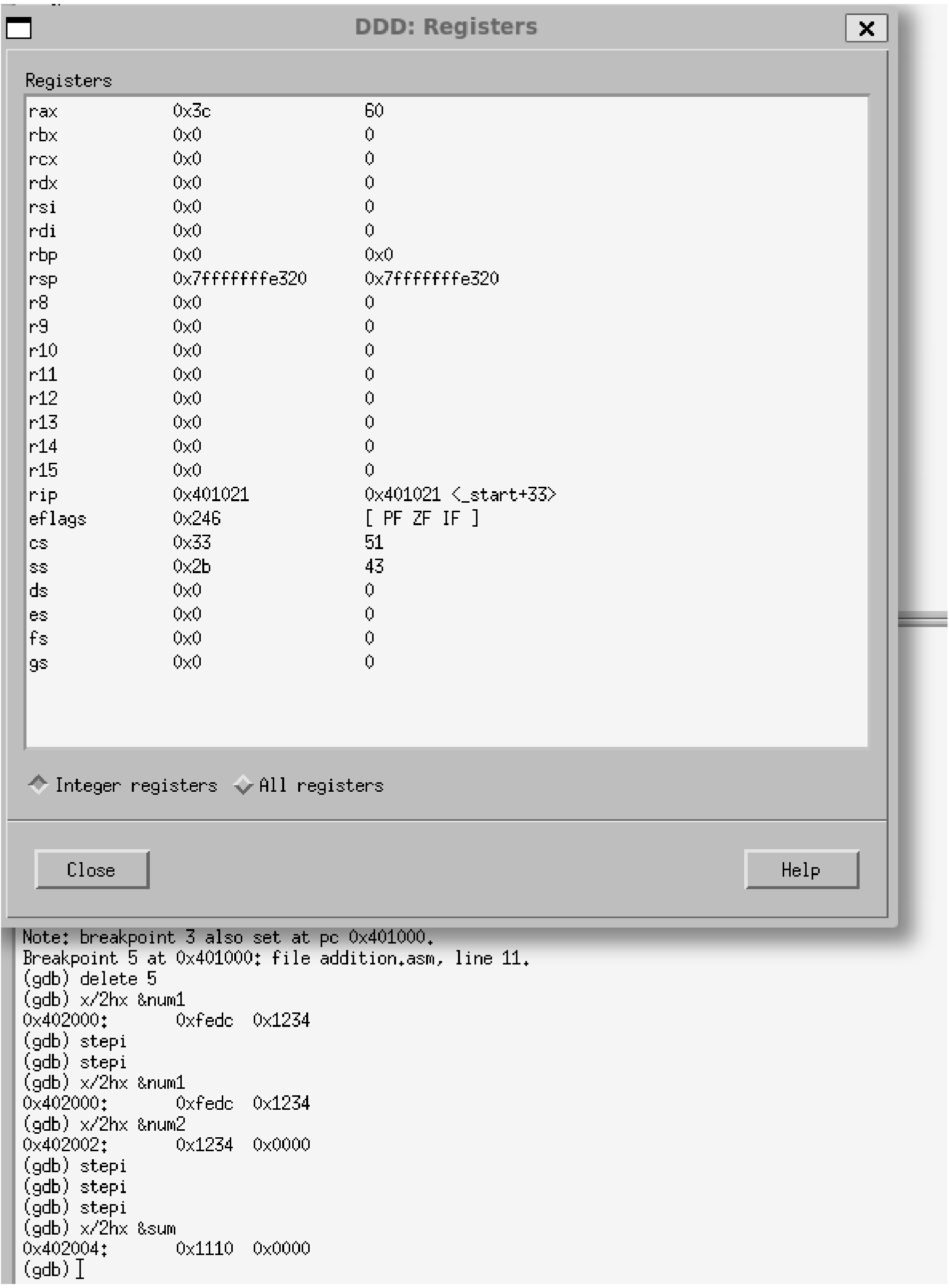
num2 = 0x1234, which is **4660**

Both in decimal.

65244 + 4660 = 69904.

So the sum is 69904 in decimal or 0x111B8 in hexadecimal.

I feel like the register wasn't showing this properly, but I know my code is working, I'm a little confused. If I did something incorrectly, please tell me in the submission comments of the assignment, I'd love to know.

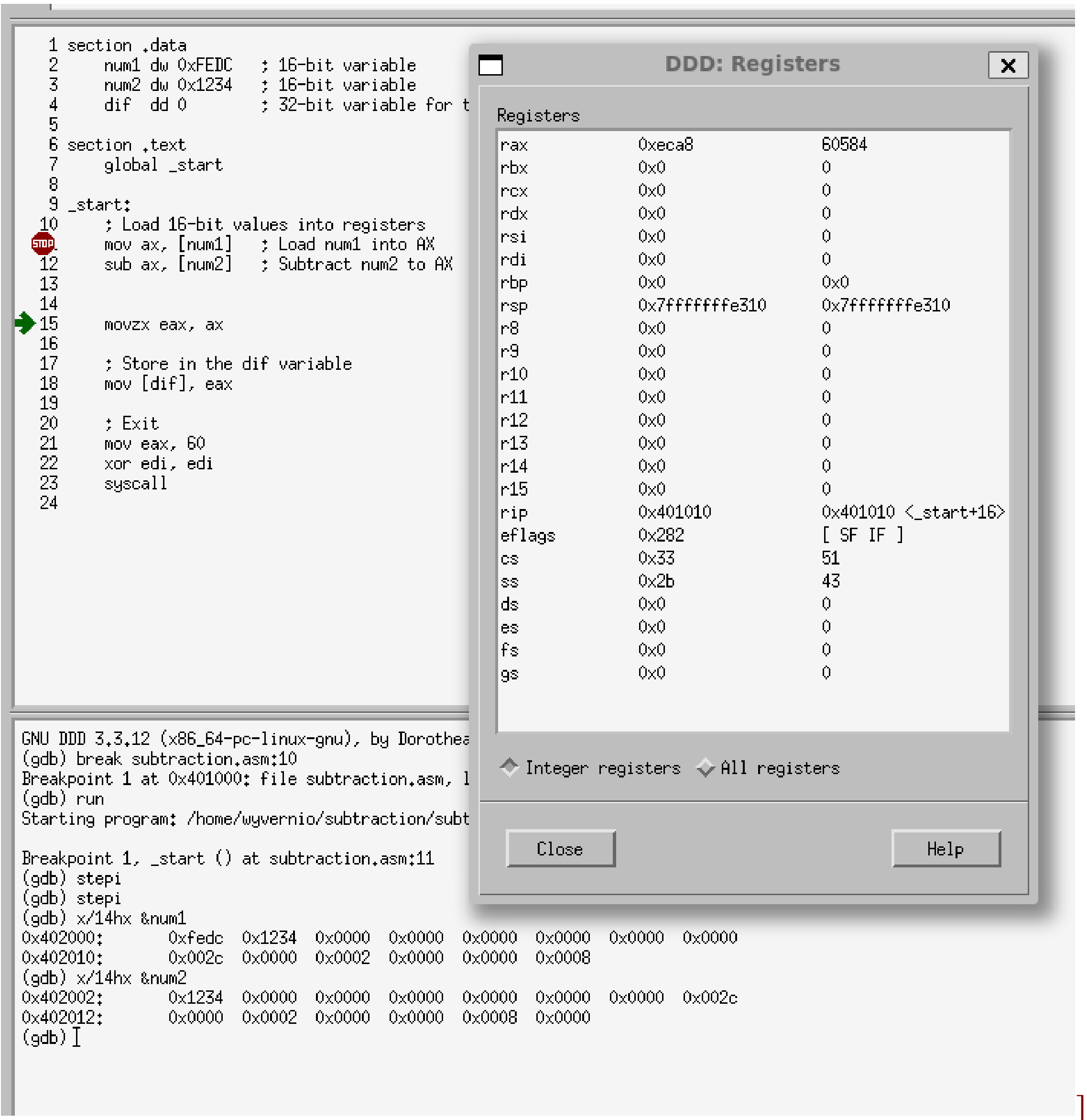
[


```
Registers
rax            0x3c             60
rbx            0x0              0
rcx            0x0              0
rdx            0x0              0
rsi            0x0              0
rdi            0x0              0
rbp            0x0              0x0
rsp            0x7fffffffe320   0x7fffffffe320
r8             0x0              0
r9             0x0              0
r10            0x0              0
r11            0x0              0
r12            0x0              0
r13            0x0              0
r14            0x0              0
r15            0x0              0
rip            0x401021         0x401021 <_start+33>
eflags         0x246            [ PF ZF IF ]
cs             0x33             51
ss             0x2b             43
ds             0x0              0
es             0x0              0
fs             0x0              0
gs             0x0              0

◇ Integer registers ◇ All registers
```

```
Note: breakpoint 3 also set at pc 0x401000.
Breakpoint 5 at 0x401000: file addition.asm, line 11.
(gdb) delete 5
(gdb) x/2hx &num1
0x402000:       0xfedc  0x1234
(gdb) stepi
(gdb) stepi
(gdb) x/2hx &num1
0x402000:       0xfedc  0x1234
(gdb) x/2hx &num2
0x402002:       0x1234  0x0000
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) x/2hx &sum
0x402004:       0x1110  0x0000
(gdb) I
```
]

```
[section .data
    num1 dw 0xFEDC      ; 16-bit variable
    num2 dw 0x1234      ; 16-bit variable
    dif   dd 0          ; 32-bit variable for the difference

section .text
    global _start

_start:
    ; Load 16-bit values into registers
    mov ax, [num1]      ; Load num1 into AX
    sub ax, [num2]      ; Subtract num2 to AX


    movzx eax, ax

    ; Store in the dif variable
    mov [dif], eax

    ; Exit
    mov eax, 60
    xor edi, edi
    syscall]
```

[



```
 1 section .data
 2     num1 dw 0xFEDC   ; 16-bit variable
 3     num2 dw 0x1234   ; 16-bit variable
 4     dif  dd 0        ; 32-bit variable for t
 5
 6 section .text
 7     global _start
 8
 9 _start:
10     ; Load 16-bit values into registers
1      mov ax, [num1]   ; Load num1 into AX
12     sub ax, [num2]   ; Subtract num2 to AX
13
14
15     movzx eax, ax
16
17     ; Store in the dif variable
18     mov [dif], eax
19
20     ; Exit
21     mov eax, 60
22     xor edi, edi
23     syscall
24
```

DDD: Registers

Registers

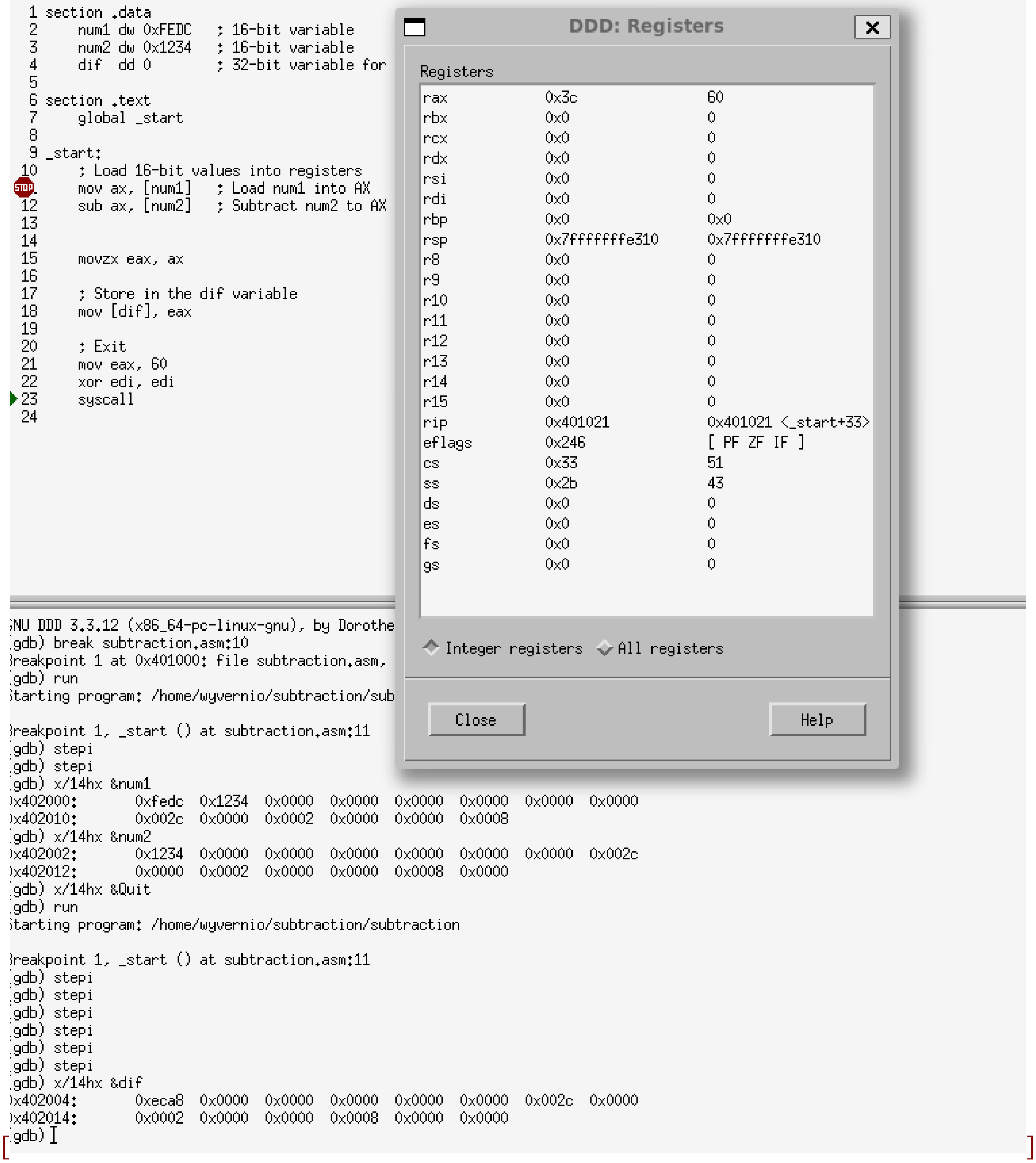
```
rax              0xeca8              60584
rbx              0x0                 0
rcx              0x0                 0
rdx              0x0                 0
rsi              0x0                 0
rdi              0x0                 0
rbp              0x0                 0x0
rsp              0x7fffffffe310      0x7fffffffe310
r8               0x0                 0
r9               0x0                 0
r10              0x0                 0
r11              0x0                 0
r12              0x0                 0
r13              0x0                 0
r14              0x0                 0
r15              0x0                 0
rip              0x401010            0x401010 <_start+16>
eflags           0x282               [ SF IF ]
cs               0x33                51
ss               0x2b                43
ds               0x0                 0
es               0x0                 0
fs               0x0                 0
gs               0x0                 0
```

◇ Integer registers ◇ All registers

Close Help

```
GNU DDD 3.3.12 (x86_64-pc-linux-gnu), by Dorothea
(gdb) break subtraction.asm:10
Breakpoint 1 at 0x401000: file subtraction.asm, l
(gdb) run
Starting program: /home/wyvernio/subtraction/subt

Breakpoint 1, _start () at subtraction.asm:11
(gdb) stepi
(gdb) stepi
(gdb) x/14hx &num1
0x402000:       0xfedc  0x1234  0x0000  0x0000  0x0000  0x0000  0x0000  0x0000
0x402010:       0x002c  0x0000  0x0002  0x0000  0x0000  0x0008
(gdb) x/14hx &num2
0x402002:       0x1234  0x0000  0x0000  0x0000  0x0000  0x0000  0x0000  0x002c
0x402012:       0x0000  0x0002  0x0000  0x0000  0x0008  0x0000
(gdb)
```

]



```
section .data
    num1 dw 0xFEDC   ; 16-bit variable
    num2 dw 0x1234   ; 16-bit variable
    dif  dd 0        ; 32-bit variable for

section .text
    global _start

_start:
    ; Load 16-bit values into registers
    mov ax, [num1]   ; Load num1 into AX
    sub ax, [num2]   ; Subtract num2 to AX


    movzx eax, ax

    ; Store in the dif variable
    mov [dif], eax

    ; Exit
    mov eax, 60
    xor edi, edi
    syscall
```

```
rax        0x3c               60
rbx        0x0                0
rcx        0x0                0
rdx        0x0                0
rsi        0x0                0
rdi        0x0                0
rbp        0x0                0x0
rsp        0x7fffffffe310     0x7fffffffe310
r8         0x0                0
r9         0x0                0
r10        0x0                0
r11        0x0                0
r12        0x0                0
r13        0x0                0
r14        0x0                0
r15        0x0                0
rip        0x401021           0x401021 <_start+33>
eflags     0x246              [ PF ZF IF ]
cs         0x33               51
ss         0x2b               43
ds         0x0                0
es         0x0                0
fs         0x0                0
gs         0x0                0
```

```
GNU DDD 3.3.12 (x86_64-pc-linux-gnu), by Dorothe
(gdb) break subtraction.asm:10
Breakpoint 1 at 0x401000: file subtraction.asm,
(gdb) run
Starting program: /home/wyvernio/subtraction/sub

Breakpoint 1, _start () at subtraction.asm:11
(gdb) stepi
(gdb) stepi
(gdb) x/14hx &num1
0x402000:       0xfedc  0x1234  0x0000  0x0000  0x0000  0x0000  0x0000  0x0000
0x402010:       0x002c  0x0000  0x0002  0x0000  0x0000  0x0008
(gdb) x/14hx &num2
0x402002:       0x1234  0x0000  0x0000  0x0000  0x0000  0x0000  0x0000  0x002c
0x402012:       0x0000  0x0002  0x0000  0x0000  0x0008  0x0000
(gdb) x/14hx &Quit
(gdb) run
Starting program: /home/wyvernio/subtraction/subtraction

Breakpoint 1, _start () at subtraction.asm:11
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) x/14hx &dif
0x402004:       0xeca8  0x0000  0x0000  0x0000  0x0000  0x0000  0x002c  0x0000
0x402014:       0x0002  0x0000  0x0000  0x0008  0x0000  0x0000
(gdb) 
```

num2 – num1 = dif is the same as saying 65244 – 4660 = 60584.

Which is exactly the same as on my register rax at the top, 60584.