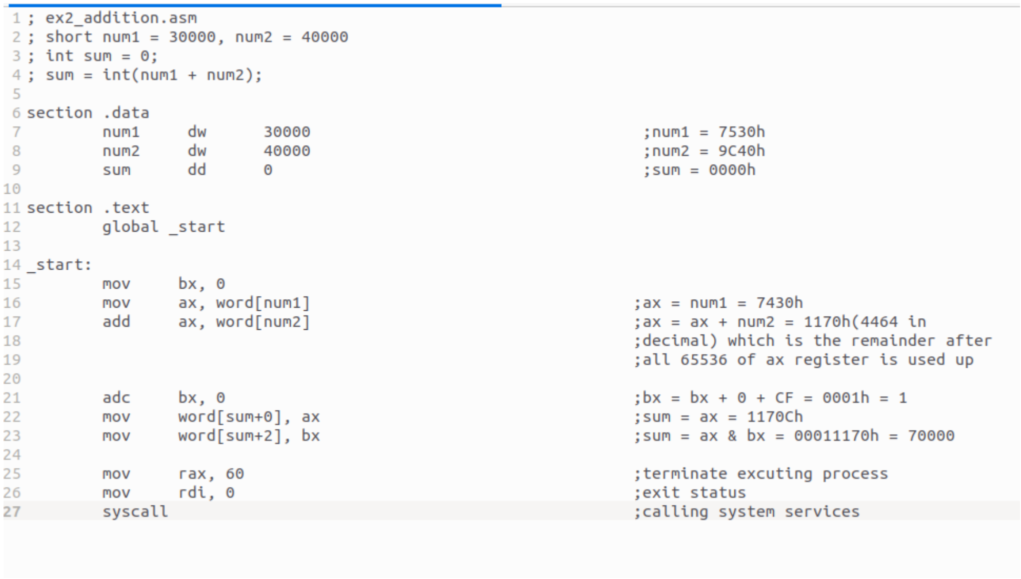
CPSC 240: Computer Organization and Assembly Language
Assignment 02, Spring Semester 2023

CWID: 886880327 Name: Andrew Saldana

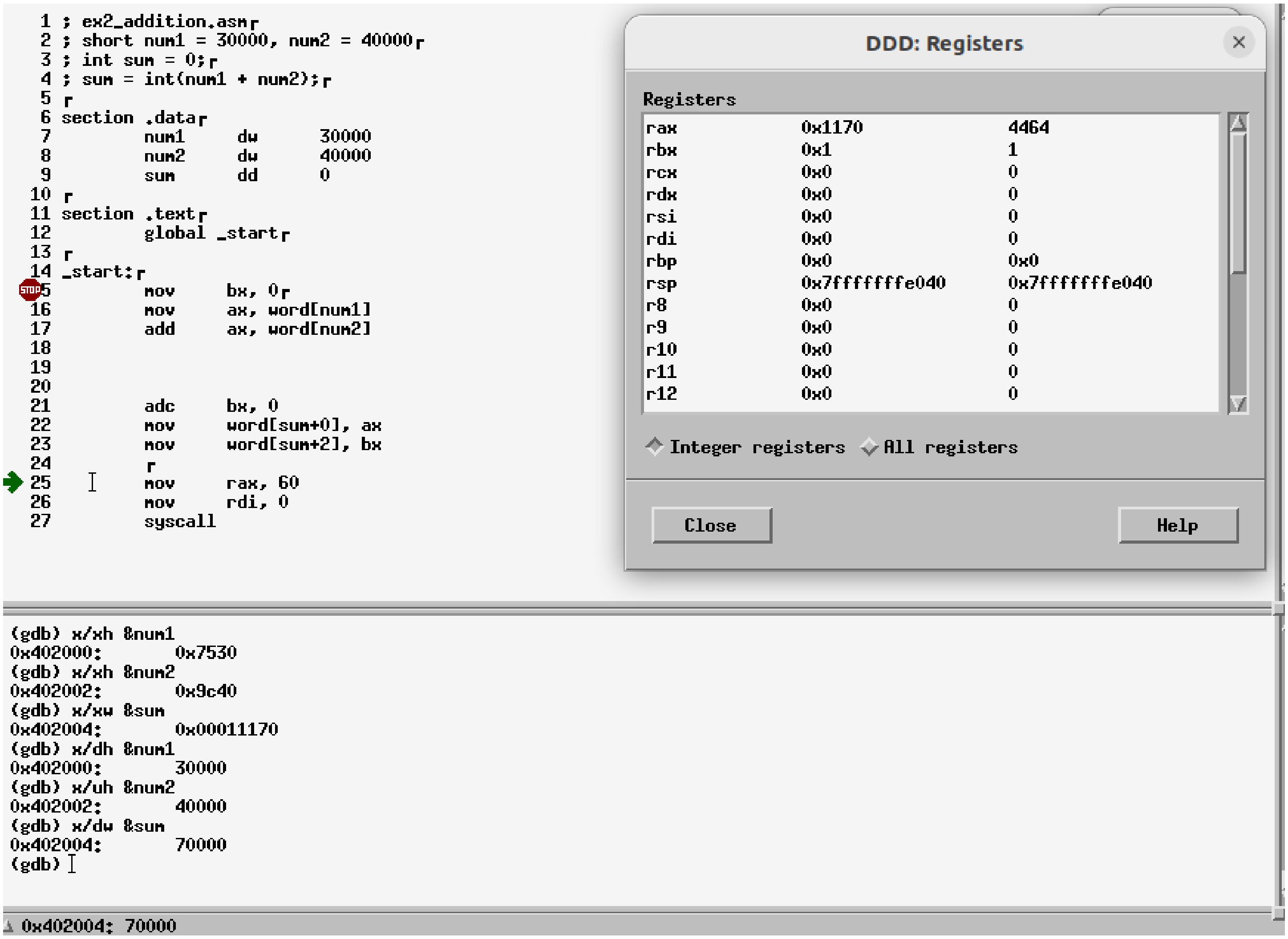
1. Download the “CPSC-240 Assignment02.docx” document.
2. Design the "addition.asm" program, and use assembly language to realize the function of the following C++ instructions.
   short num1 = 30000;
   short num2 = 40000;
   int sum = 0;
   sum = int(num1 + num2);
3. Assemble the "addition.asm" file and link the "addition.o" file to get the "addition" executable file.
4. Run the "addition" file with the DDD 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 (DDD debugger window) of the memory (num1, num2, and sum) in the document. Write an analysis to verify simulation results.
6. Design the "subtraction.asm" program, and use assembly language to realize the function of the following C++ instructions.
   short num1 = 30000;
   short num2 = 40000;
   int dif = 0;
   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 DDD 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 (DDD debugger window) of the memory (num1, num2, and dif) in the document. Write an analysis to verify simulation results.
10. Save the file in pdf format and submit the pdf file to Canvas before 23:59 pm on 02/15/2023.

[Insert addition.asm here]



```
; ex2_addition.asm
; short num1 = 30000, num2 = 40000
; int sum = 0;
; sum = int(num1 + num2);

section .data
        num1     dw      30000                                ;num1 = 7530h
        num2     dw      40000                                ;num2 = 9C40h
        sum      dd      0                                    ;sum = 0000h

section .text
        global _start

_start:
        mov     bx, 0
        mov     ax, word[num1]                               ;ax = num1 = 7430h
        add     ax, word[num2]                               ;ax = ax + num2 = 1170h(4464 in
                                                             ;decimal) which is the remainder after
                                                             ;all 65536 of ax register is used up

        adc     bx, 0                                        ;bx = bx + 0 + CF = 0001h = 1
        mov     word[sum+0], ax                              ;sum = ax = 1170Ch
        mov     word[sum+2], bx                              ;sum = ax & bx = 00011170h = 70000

        mov     rax, 60                                      ;terminate excuting process
        mov     rdi, 0                                       ;exit status
        syscall                                              ;calling system services
```

[Insert addition simulation result here]



```
; ex2_addition.asmr
; short num1 = 30000, num2 = 40000r
; int sum = 0;r
; sum = int(num1 + num2);r
r
section .datar
        num1     dw      30000
        num2     dw      40000
        sum      dd      0
r
section .textr
        global _startr
r
_start:r
        mov     bx, 0r
        mov     ax, word[num1]
        add     ax, word[num2]



        adc     bx, 0
        mov     word[sum+0], ax
        mov     word[sum+2], bx
        r
        mov     rax, 60
        mov     rdi, 0
        syscall
```

DDD: Registers

Registers

```
rax            0x1170              4464
rbx            0x1                 1
rcx            0x0                 0
rdx            0x0                 0
rsi            0x0                 0
rdi            0x0                 0
rbp            0x0                 0x0
rsp            0x7fffffffe040      0x7fffffffe040
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0x0                 0
```

```
(gdb) x/xh &num1
0x402000:       0x7530
(gdb) x/xh &num2
0x402002:       0x9c40
(gdb) x/xw &sum
0x402004:       0x00011170
(gdb) x/dh &num1
0x402000:       30000
(gdb) x/uh &num2
0x402002:       40000
(gdb) x/dw &sum
0x402004:       70000
(gdb)
```

0x402004: 70000

[Insert addition simulation result analysis here]

30000 to hex

```
30000/16 = 1875  R 0
1875/16  = 117   R 3      Hex: 7530h
117/16   = 7     R 5      binary: 0111 0101 0011 0000
7/16     = 0     R 7
```

40000 in hex

```
40000/16 = 2500  R 0
2500/16  = 156   R 4      Hex: 9C40h
156/16   = 9     R 12     binary:
9/16     = 0     R 9      1001 1100 0100 0000
```

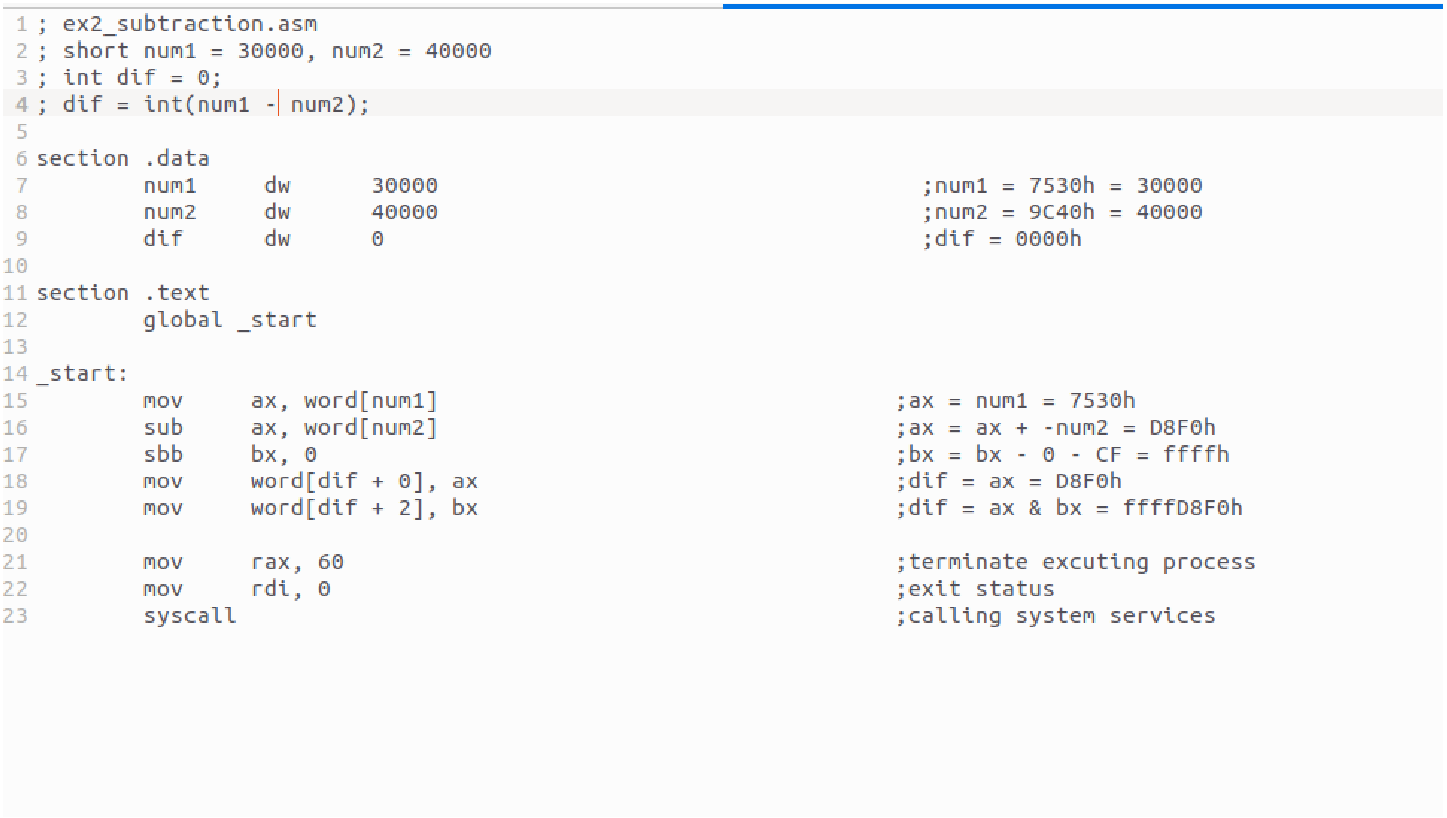
30000 + 40000

```
    1 11   1
    0111 0101 0011 0000  ← 30000
    1001 1100 0100 0000  ← 40000
   ---------------------
 1  0001 0001 0111 0000  ← 70000  signed
```

hex = 11170h

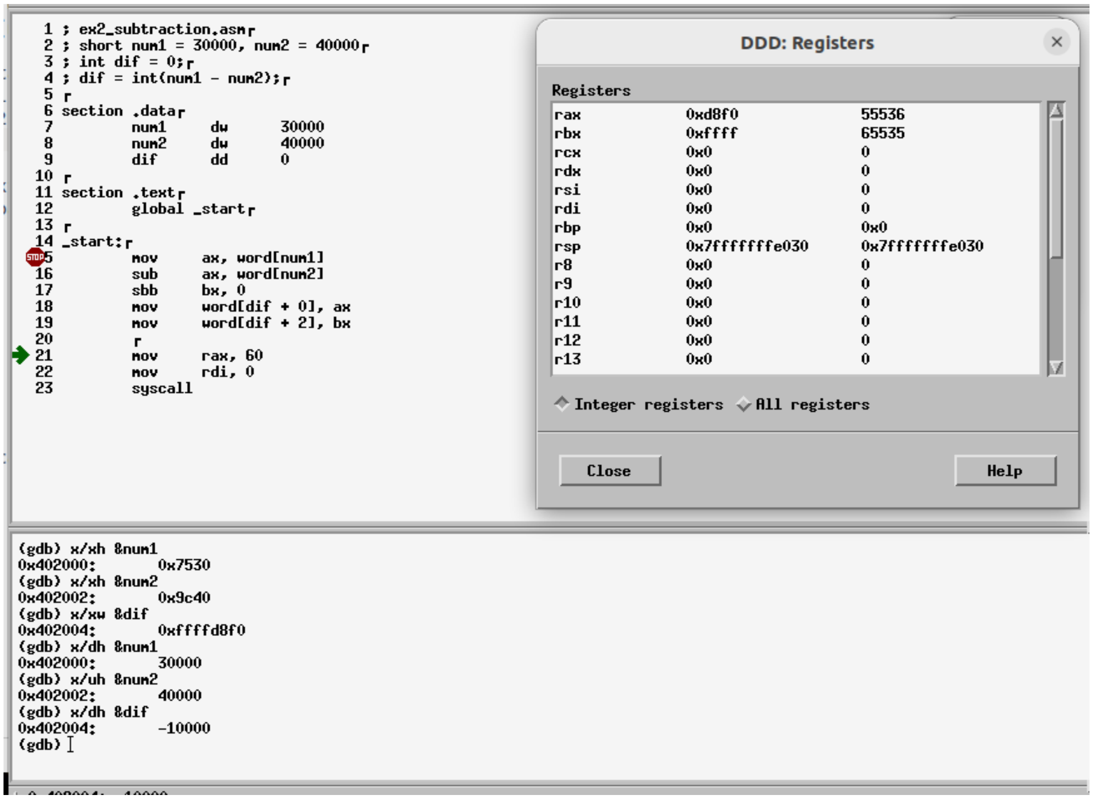
70000	hex	binary
	11170h	0001 0001 0001 0111 0000

[Insert subtraction.asm here]



```
; ex2_subtraction.asm
; short num1 = 30000, num2 = 40000
; int dif = 0;
; dif = int(num1 - num2);

section .data
        num1      dw      30000                                     ;num1 = 7530h = 30000
        num2      dw      40000                                     ;num2 = 9C40h = 40000
        dif       dw      0                                         ;dif = 0000h

section .text
        global _start

_start:
        mov     ax, word[num1]                                    ;ax = num1 = 7530h
        sub     ax, word[num2]                                    ;ax = ax + -num2 = D8F0h
        sbb     bx, 0                                             ;bx = bx - 0 - CF = ffffh
        mov     word[dif + 0], ax                                 ;dif = ax = D8F0h
        mov     word[dif + 2], bx                                 ;dif = ax & bx = ffffD8F0h

        mov     rax, 60                                           ;terminate excuting process
        mov     rdi, 0                                            ;exit status
        syscall                                                   ;calling system services
```

[Insert subtraction simulation result here]



```
; ex2_subtraction.asm
; short num1 = 30000, num2 = 40000
; int dif = 0;
; dif = int(num1 - num2);

section .data
         num1     dw      30000
         num2     dw      40000
         dif      dd      0

section .text
         global _start

_start:
         mov      ax, word[num1]
         sub      ax, word[num2]
         sbb      bx, 0
         mov      word[dif + 0], ax
         mov      word[dif + 2], bx

         mov      rax, 60
         mov      rdi, 0
         syscall
```

DDD: Registers

Registers

```
rax            0xd8f0            55536
rbx            0xffff            65535
rcx            0x0               0
rdx            0x0               0
rsi            0x0               0
rdi            0x0               0
rbp            0x0               0x0
rsp            0x7fffffffe030    0x7fffffffe030
r8             0x0               0
r9             0x0               0
r10            0x0               0
r11            0x0               0
r12            0x0               0
r13            0x0               0
```

```
(gdb) x/xh &num1
0x402000:       0x7530
(gdb) x/xh &num2
0x402002:       0x9c40
(gdb) x/xw &dif
0x402004:       0xffffd8f0
(gdb) x/dh &num1
0x402000:       30000
(gdb) x/uh &num2
0x402002:       40000
(gdb) x/dh &dif
0x402004:       -10000
(gdb) 
```

[Insert subtraction simulation result analysis here]

30000 to hex

```
30000/16 =  1875   R 0
1875/16  =  117    R 3      Hex: 7530h
117/16   =  7      R 5      binary: 0111 0101 0011 0000
7/16     =  0      R 7
```

40000 in hex

```
40000/16 =  2500   R 0
2500/16  =  156    R 4      Hex: 9C40h
156/16   =  9      R 12     binary:
9/16     =  0      R 9      1001 1100 0100 0000
```

30000 + (-40000)

```
                                 40000
  1001 1100 0100 0000
                        ← 1's compliment
  0110 0011 1011 1111
+ 0000 0000 0000 0001
  0110 0011 1100 0000   ← 2's compliment = -40000
+ 0111 0101 0011 0000   ← 30000
  1101 1000 1111 0000   ← -10000 in signed
    hex= D8F0h
```

	hex	binary
-10000	D8F0h	1101 1000 1111 0000