Computer Science Department
California State University, Fullerton

CPSC 240-11/12 Computer Organization and Assembly Language
Quiz 01
12:00 noon to 1:15 pm
Thursday, October 3, 2024

Student Name: Sean DeFrank

Last 4 digits of ID: 884614918

Note:

- University regulations on academic honesty will be strictly enforced.
- You have 75 minutes to complete this Quiz.
- Open books, slides and sample programs.
- Turn off or turn vibration your cell phone.
- Use "yasm/nasm" assembler to assemble the source code.
- Use "ld" linker to link the object code
- Use "ddd/gdb" debugger to simulate the executable code.
- Each student can only submit solution once, and secondary submissions will not be graded. If you have submitting problems, please inform your instructor before you leave the classroom.
- Any content submitted after the due date will be regarded as a make-up quiz.

Quiz 01

1. Download the "CPSC-240-11 Quiz 01.docx" document from Canvas.
2. Convert the following C/C++ variable declarations and arithmetic operations to x86-64 assembly language. Use the "yasm/nasm" assembler to assemble the program, the "ld" linker to link the object code, and the "ddd/gdb" debugger to simulate the executable code. NOTE: variable sizes and program functions should be equivalent to C/C++ instructions.

```
signed short num1 = 20000;          //16-bit signed variable
signed short num2 = 30000;          //16-bit signed variable
signed short num3 = -20000;         //16-bit signed variable
signed int sum = 0                  //32-bit signed variable
signed long product = 0;            //64-bit signed variable
sum = int(num1 + num2);
product = int(num3) * sum;
```

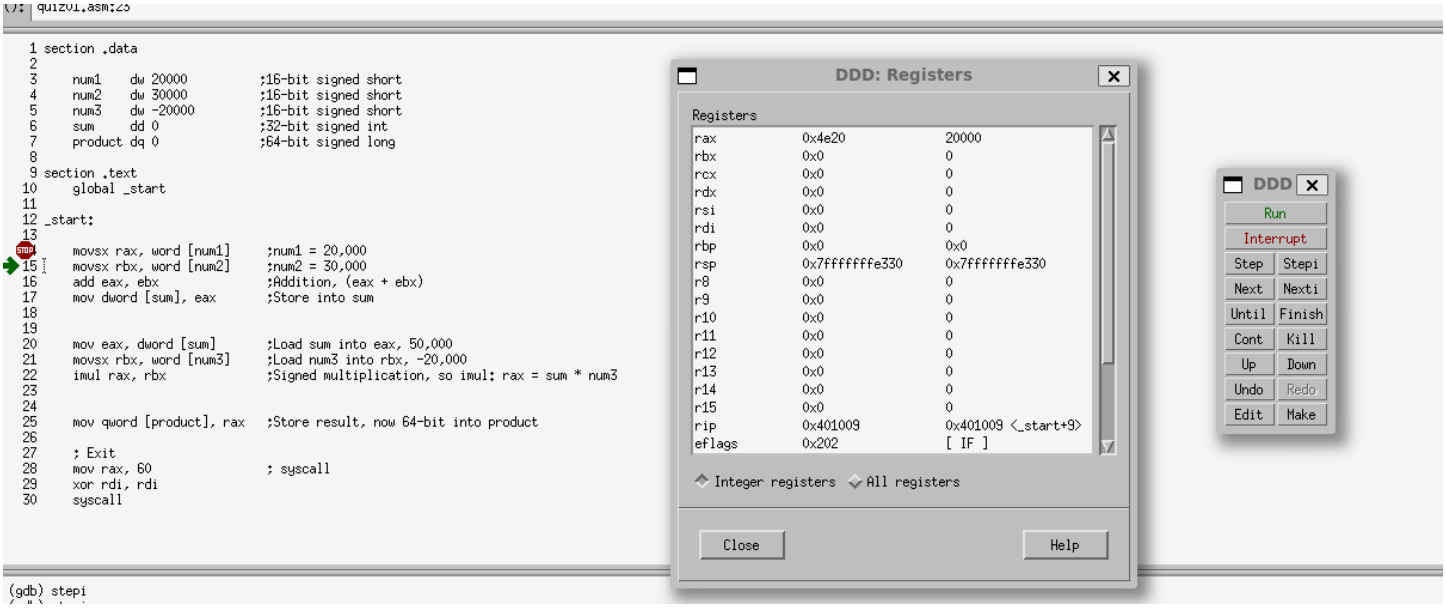
3. After assembling and linking, run the DDD/GDB debugger to display the simulation results of the decimal values of num1, num2, num3, sum, and product in GDB panel before terminate program.
4. Insert source code and the simulation results (GDB panel) to the bottom of the document.
5. Save the file in pdf or docx format and submit the pdf or docx file to Canvas before the deadline.
6. Deadline is 1:15 pm on 10/03/2024.

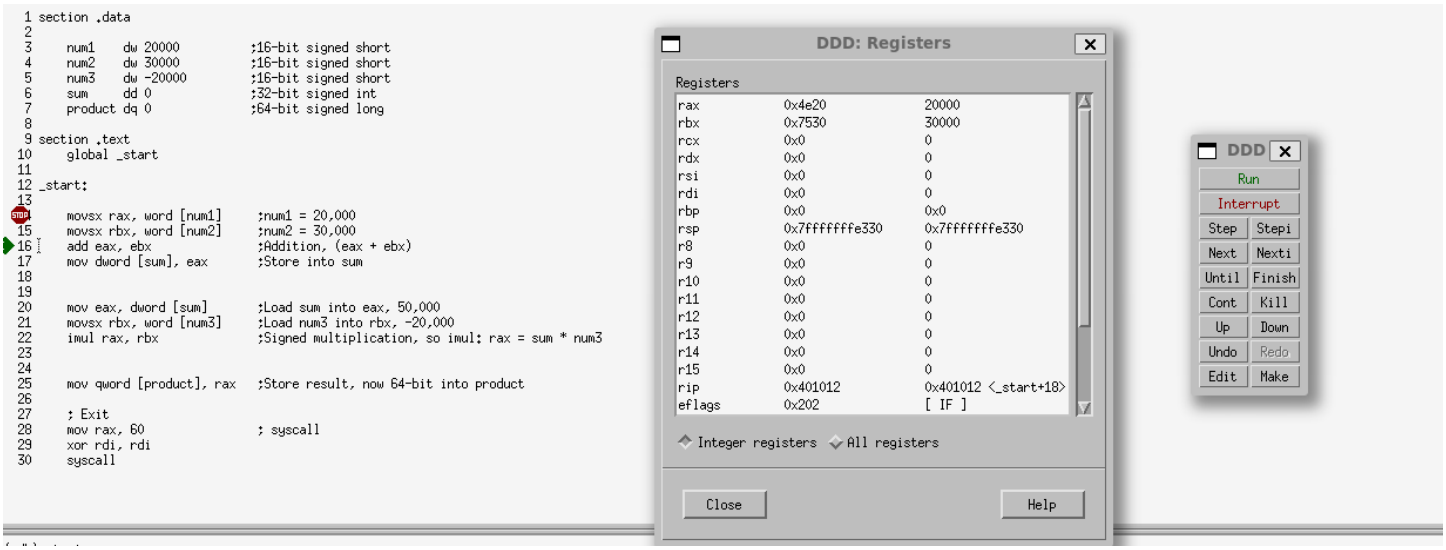
[Copy and paste your assembly source code here:

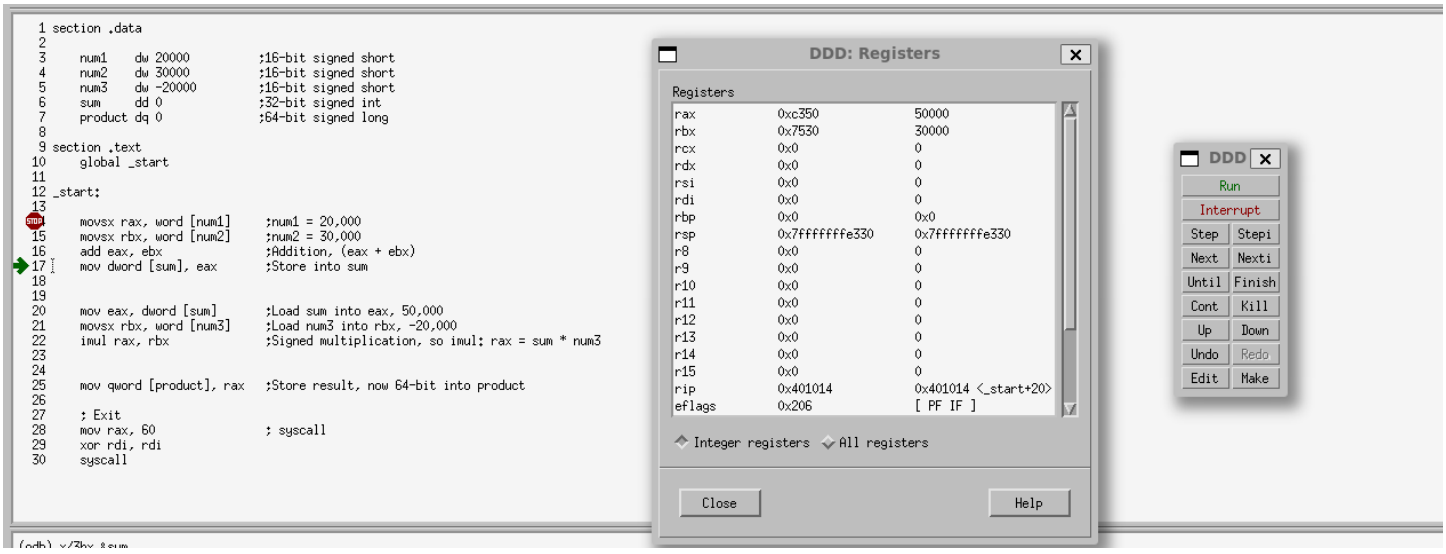
```
section .data

    num1      dw 20000          ;16-bit signed short
    num2      dw 30000          ;16-bit signed short
    num3      dw -20000         ;16-bit signed short
    sum       dd 0              ;32-bit signed int
    product dq 0                ;64-bit signed long

section .text
    global _start
```

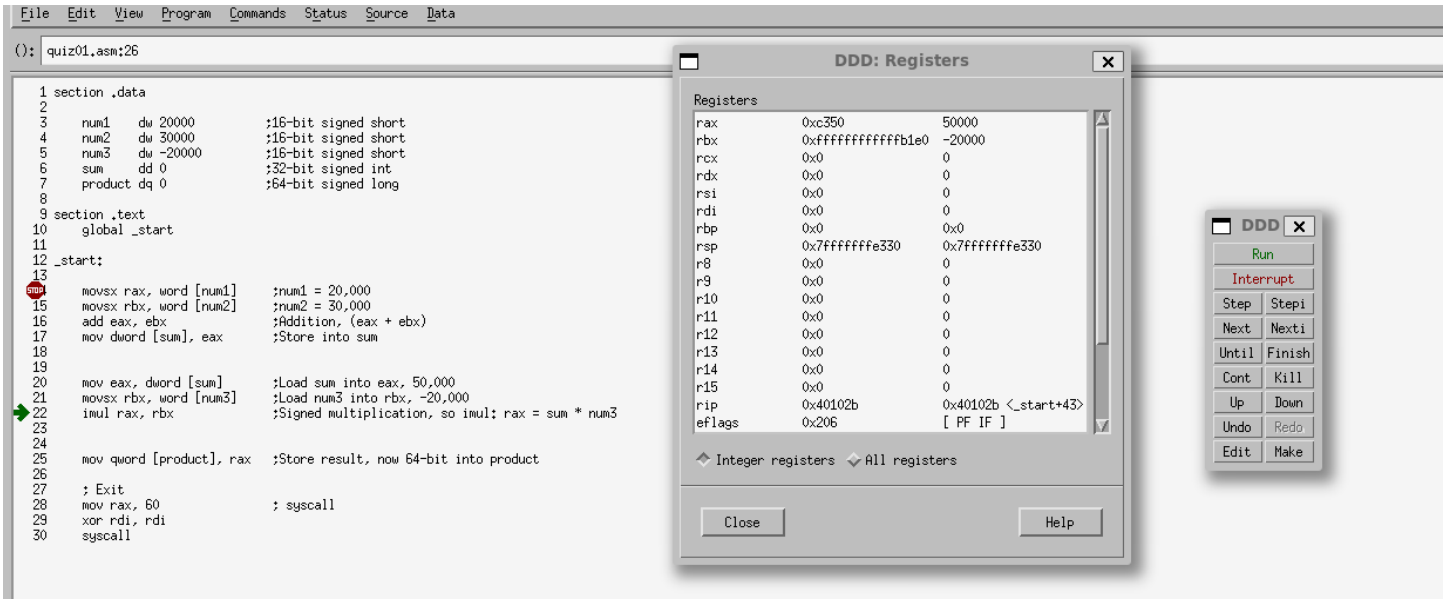
```
_start:

    movsx rax, word [num1]        ;num1 = 20,000
    movsx rbx, word [num2]        ;num2 = 30,000
    add eax, ebx                  ;Addition, (eax + ebx)
    mov dword [sum], eax          ;Store into sum


    mov eax, dword [sum]          ;Load sum into eax, 50,000
    movsx rbx, word [num3]        ;Load num3 into rbx, -20,000
    imul rax, rbx                 ;Signed multiplication, so imul: rax = sum * num3


    mov qword [product], rax      ;Store result, now 64-bit into product

    ; Exit
    mov rax, 60                    ; syscall
    xor rdi, rdi
    syscall]
```

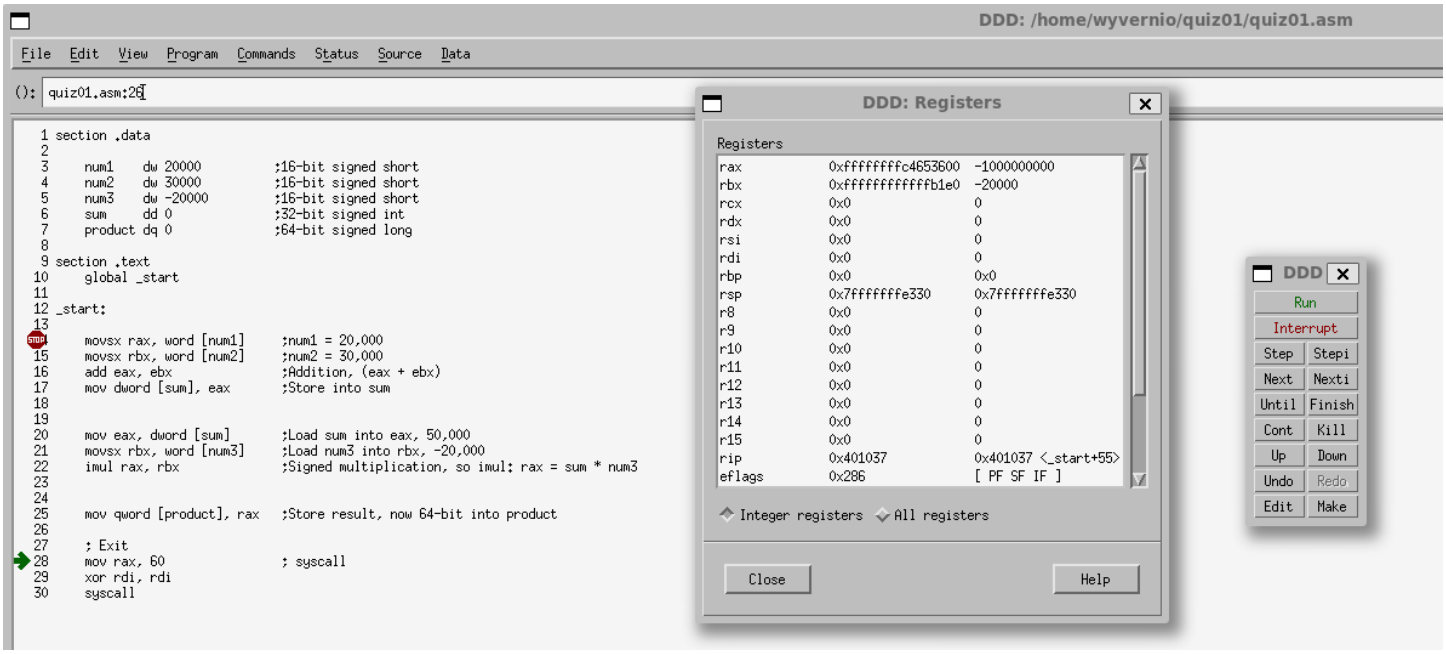
[Attach GDB panel with all variable value here:


quiz01.asm:23
1 section .data
2
3 num1 dw 20000 ;16-bit signed short
4 num2 dw 30000 ;16-bit signed short
5 num3 dw -20000 ;16-bit signed short
6 sum dd 0 ;32-bit signed int
7 product dq 0 ;64-bit signed long
8
9 section .text
10 global _start
11
12 _start:
13
14 movsx rax, word [num1] ;num1 = 20,000
15 movsx rbx, word [num2] ;num2 = 30,000
16 add eax, ebx ;Addition, (eax + ebx)
17 mov dword [sum], eax ;Store into sum
18
19
20 mov eax, dword [sum] ;Load sum into eax, 50,000
21 movsx rbx, word [num3] ;Load num3 into rbx, -20,000
22 imul rax, rbx ;Signed multiplication, so imul: rax = sum * num3
23
24
25 mov qword [product], rax ;Store result, now 64-bit into product
26
27 ; Exit
28 mov rax, 60 ; syscall
29 xor rdi, rdi
30 syscall
DDD: Registers
Registers
rax 0x4e20 20000
rbx 0x0 0
rcx 0x0 0
rdx 0x0 0
rsi 0x0 0
rdi 0x0 0
rbp 0x0 0x0
rsp 0x7fffffffe330 0x7fffffffe330
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 0x401009 0x401009 <_start+9>
eflags 0x202 [IF]
Integer registers
All registers
Close
Help
DDD
Run
Interrupt
Step
Stepi
Next
Nexti
Until
Finish
Cont
Kill
Up
Down
Undo
Redo
Edit
Make
(gdb) stepi


1 section .data
2
3 num1 dw 20000 ;16-bit signed short
4 num2 dw 30000 ;16-bit signed short
5 num3 dw -20000 ;16-bit signed short
6 sum dd 0 ;32-bit signed int
7 product dq 0 ;64-bit signed long
8
9 section .text
10 global _start
11
12 _start:
13
14 movsx rax, word [num1] ;num1 = 20,000
15 movsx rbx, word [num2] ;num2 = 30,000
16 add eax, ebx ;Addition, (eax + ebx)
17 mov dword [sum], eax ;Store into sum
18
19
20 mov eax, dword [sum] ;Load sum into eax, 50,000
21 movsx rbx, word [num3] ;Load num3 into rbx, -20,000
22 imul rax, rbx ;Signed multiplication, so imul: rax = sum * num3
23
24
25 mov qword [product], rax ;Store result, now 64-bit into product
26
27 ; Exit
28 mov rax, 60 ; syscall
29 xor rdi, rdi
30 syscall
DDD: Registers
Registers
rax 0x4e20 20000
rbx 0x7530 30000
rcx 0x0 0
rdx 0x0 0
rsi 0x0 0
rdi 0x0 0
rbp 0x0 0x0
rsp 0x7fffffffe330 0x7fffffffe330
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 0x401012 0x401012 <_start+18>
eflags 0x202 [IF]
Integer registers
All registers
Close
Help
DDD
Run
Interrupt
Step
Stepi
Next
Nexti
Until
Finish
Cont
Kill
Up
Down
Undo
Redo
Edit
Make


1 section .data
2
3 num1 dw 20000 ;16-bit signed short
4 num2 dw 30000 ;16-bit signed short
5 num3 dw -20000 ;16-bit signed short
6 sum dd 0 ;32-bit signed int
7 product dq 0 ;64-bit signed long
8
9 section .text
10 global _start
11
12 _start:
13
14 movsx rax, word [num1] ;num1 = 20,000
15 movsx rbx, word [num2] ;num2 = 30,000
16 add eax, ebx ;Addition, (eax + ebx)
17 mov dword [sum], eax ;Store into sum
18
19
20 mov eax, dword [sum] ;Load sum into eax, 50,000
21 movsx rbx, word [num3] ;Load num3 into rbx, -20,000
22 imul rax, rbx ;Signed multiplication, so imul: rax = sum * num3
23
24
25 mov qword [product], rax ;Store result, now 64-bit into product
26
27 ; Exit
28 mov rax, 60 ; syscall
29 xor rdi, rdi
30 syscall
DDD: Registers
Registers
rax 0xc350 50000
rbx 0x7530 30000
rcx 0x0 0
rdx 0x0 0
rsi 0x0 0
rdi 0x0 0
rbp 0x0 0x0
rsp 0x7fffffffe330 0x7fffffffe330
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 0x401014 0x401014 <_start+20>
eflags 0x206 [PF IF]
Integer registers
All registers
Close
Help
DDD
Run
Interrupt
Step
Stepi
Next
Nexti
Until
Finish
Cont
Kill
Up
Down
Undo
Redo
Edit
Make





In order for the program to succeed, we had to add together 20k and 30k to make 50k, then multiply it by -20k to get -1 billion. This required the use of signed variables and imul instead of mul. Sorry to make you look through all the registers, I felt like, since it was a quiz, I should be as clear as possible and take a screenshot of my register every step that it changed, proven by the breakpoint on the left. Below is also the calculator result of the final answer, figured it was better to be safe than sorry. Have a good day!


Calculator
Programmer
Memory
50000 × -20000 =
There's nothing saved in memory.
-1,000,000,000
HEX FFFF FFFF C465 3600
DEC -1,000,000,000
OCT 1 777 777 777 770 431 233 000
BIN 1111 1111 1111 1111 1111 1111 1111 1111 1100 0100 0110 0101 0011 0110 0000 0000
QWORD
MS

]