Computer Science Department
California State University, Fullerton

CPSC 240 Computer Organization and Assembly Language
Quiz 01
12:10 PM to 01:25 PM
Thursday, March 2, 2023

Student Name: Andrew Saldana

Last 4 digits of ID: 0327

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 assembler for the program design.
- Copy and paste your assembly source code and DDD debugger window to the end of the word file and save it in pdf or docx format.
- Submit you pdf or docx file to Canvas before the deadline.
  NOTE: Email submissions will not be graded.
- Any content submitted after the due date will be regarded as a make-up quiz.

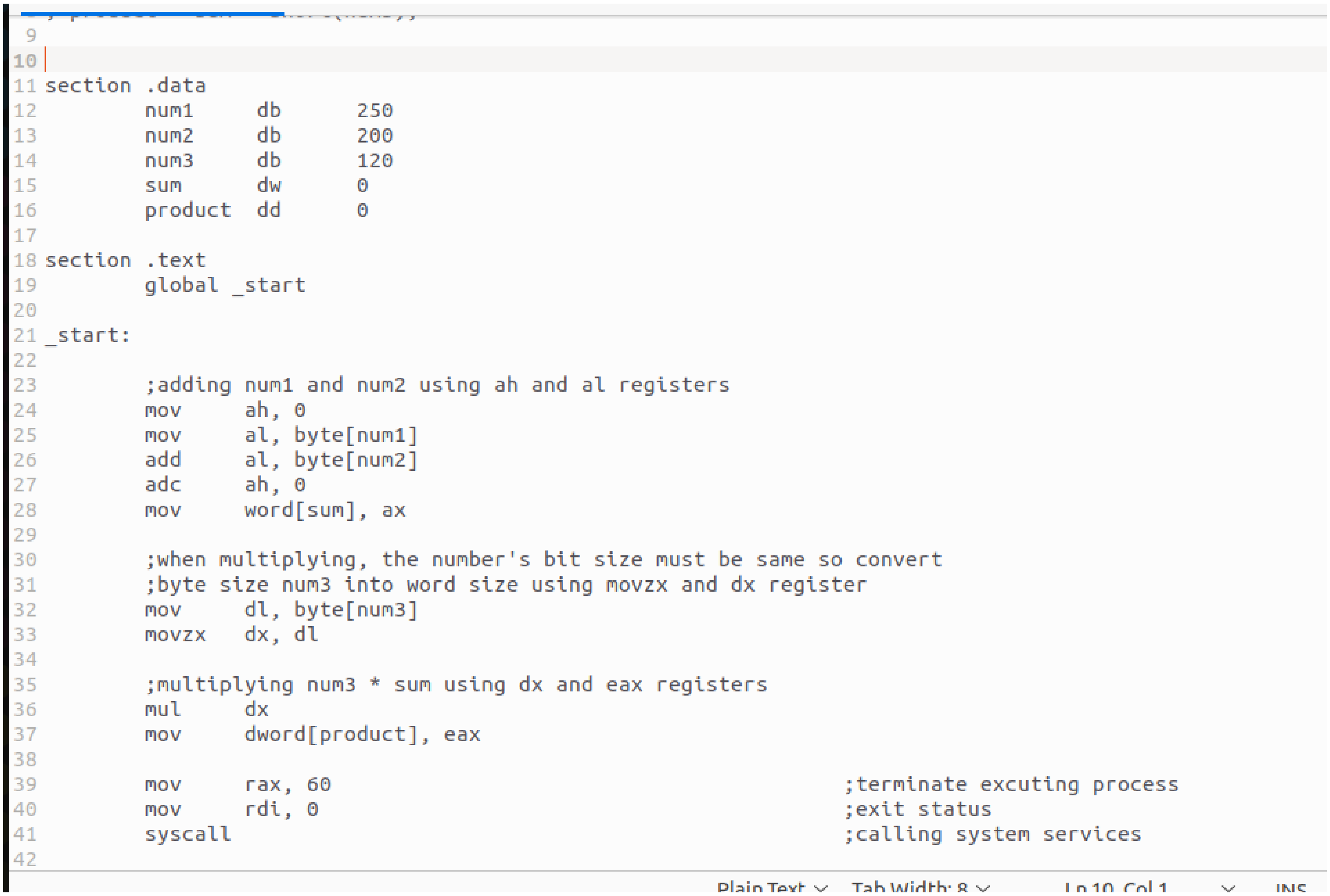
Quiz 01

1. Download the “CPSC-240-01 Quiz 01.docx” document.
2. Use x86-64 assembly language to implement the following C/C++ arithmetic operations.

```
unsigned char num1 = 250;            //data type: 8 bits
unsigned char num2 = 200;            //data type: 8 bits
unsigned char num3 = 120;            //data type: 8 bits
unsigned short sum = 0               //data type: 16 bits
unsigned int product = 0;            //data type: 32 bits

sum = num1 + num2;
product = sum * short(num3);
```

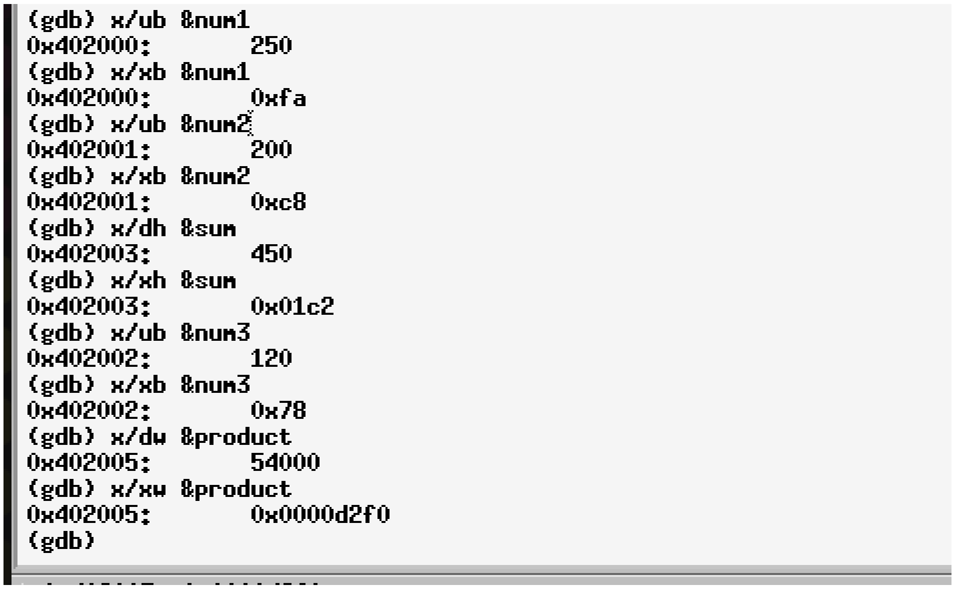
3. After assembling and linking, run the DDD debugger to display the simulation results of the register window before terminate program and the memories of num1, num2, num3, sum, and product.
4. Insert source code and the simulation results (Register window and GDB window) to the bottom of the document.
5. Save the file in pdf or docx format and submit the pdf or docx file to Canvas before the deadline.
6. Deadline is 1:25 pm on 03/02/2023.

[Attach your assembly source code here:]



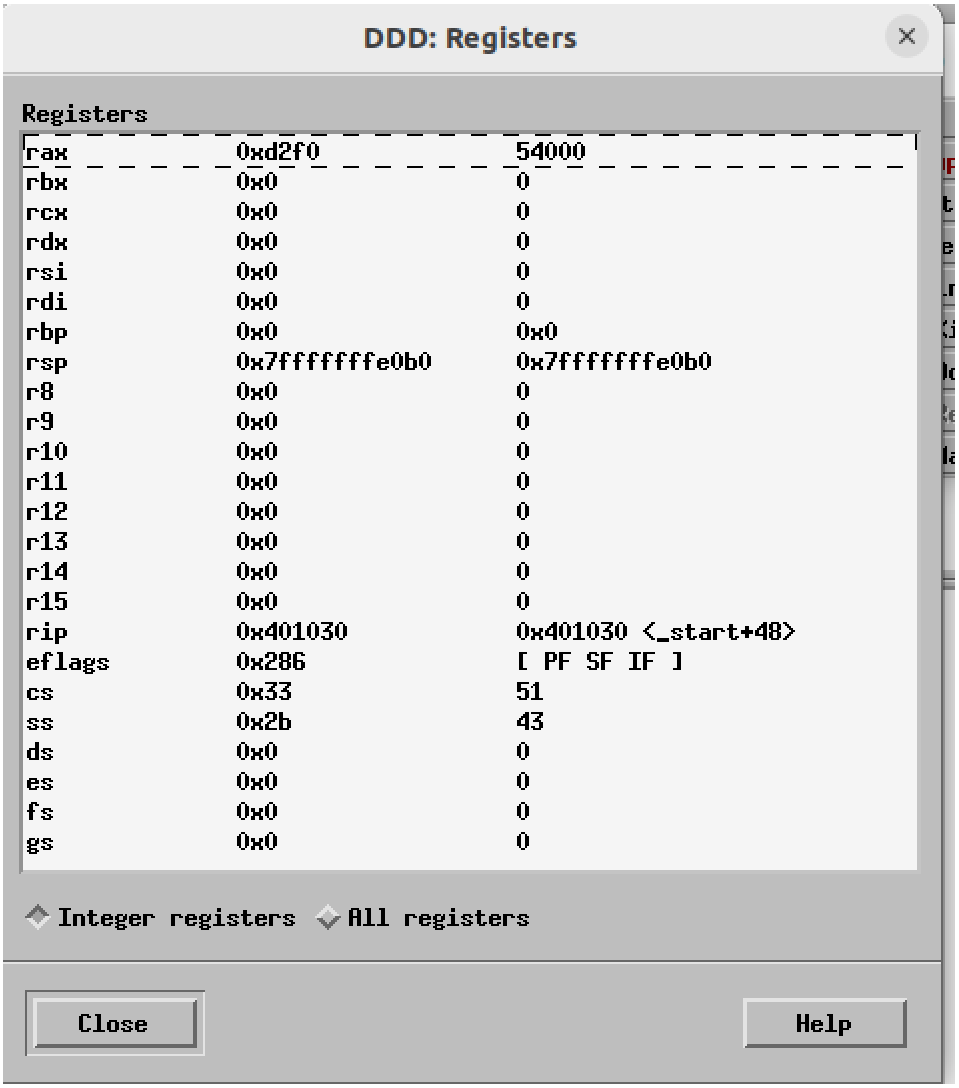
```
section .data
        num1    db      250
        num2    db      200
        num3    db      120
        sum     dw      0
        product dd      0

section .text
        global _start

_start:

        ;adding num1 and num2 using ah and al registers
        mov     ah, 0
        mov     al, byte[num1]
        add     al, byte[num2]
        adc     ah, 0
        mov     word[sum], ax

        ;when multiplying, the number's bit size must be same so convert
        ;byte size num3 into word size using movzx and dx register
        mov     dl, byte[num3]
        movzx   dx, dl

        ;multiplying num3 * sum using dx and eax registers
        mul     dx
        mov     dword[product], eax

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

```

[Attach Register window with relative register here:]



```
(gdb) x/ub &num1
0x402000:       250
(gdb) x/xb &num1
0x402000:       0xfa
(gdb) x/ub &num2
0x402001:       200
(gdb) x/xb &num2
0x402001:       0xc8
(gdb) x/dh &sum
0x402003:       450
(gdb) x/xh &sum
0x402003:       0x01c2
(gdb) x/ub &num3
0x402002:       120
(gdb) x/xb &num3
0x402002:       0x78
(gdb) x/dw &product
0x402005:       54000
(gdb) x/xw &product
0x402005:       0x0000d2f0
(gdb)
```

[Attach GDB window with all memory data here:]



```
rax            0xd2f0              54000
rbx            0x0                 0
rcx            0x0                 0
rdx            0x0                 0
rsi            0x0                 0
rdi            0x0                 0
rbp            0x0                 0x0
rsp            0x7fffffffe0b0      0x7fffffffe0b0
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            0x401030            0x401030 <_start+48>
eflags         0x286               [ PF SF IF ]
cs             0x33                51
ss             0x2b                43
ds             0x0                 0
es             0x0                 0
fs             0x0                 0
gs             0x0                 0
```