

# CPSC 240: Computer Organization and Assembly Language

## Assignment 04, Fall Semester 2024

CWID: 884614918 Name: Sean  
DeFrank

### Programming:

1. Download the "CPSC-240 Assignment04.docx" document.
2. Design "multiple.asm" program to implement an if-else structure in assembly language, 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 num = 65535;           // use dw to declare 16-bit variable
unsigned short mul_3 = 0, other = 0;   // use dw to declare 16-bit variable
if(num % 3 == 0 && num % 5 != 0) {
    mul_3++;
} else {
    other++;
}
```
3. Assemble the "multiple.asm" file and link the "multiple.o" file to get the "multiple" executable file.
4. Run the "multiple" file with the GDB debugger to display the memory of **num**, as well as the simulation results of **mul\_3** and **other**.
5. Insert source code (multiple.asm) and simulation results (GDB window) of the memory (num, mul\_3, and other) in the document. Write an analysis to verify simulation results.
6. Save the file in pdf format and submit the pdf file to Canvas before the deadline.

```
[; multiple.asm
; unsigned short num = 65535;
; unsigned short mul_3 = 0, other = 0;
; if(num % 3 == 0 && num % 5 != 0) {
;     mul_3++;
; } else {
;     other++;
; }
```

```
section .data
    SYS_exit    equ    60
    EXIT_SUCCESS equ    0
    num         dw      65535
    mul_3       dw      0          ;mul_3 = 0
    other       dw      0          ;other = 0
```

section .text

global \_start

\_start:

mov ax, [num]

;num % 3

mov cx, 3

xor dx, dx

div cx

cmp dx, 0 ;check num % 3 == 0

jne else\_block ;If num % 3 != 0, jump to else\_block

;num % 5

mov ax, [num] ;Load 'num' again into AX (since div modifies AX)

mov cx, 5 ;Set divisor as 5

xor dx, dx ;Clear DX for division

div cx

cmp dx, 0 ;check num % 5 == 0

je else\_block ;If num % 5 == 0, jump to else\_block

if\_block:

;If num % 3 == 0 and num % 5 != 0, mul\_3++

mov ax, [mul\_3] ;Load mul\_3 into AX

inc ax ;Increment AX (mul\_3++)

mov [mul\_3], ax ;Store the result

jmp end\_if ;Jump to end of program

else\_block:

;Else, other++

mov ax, [other] ;Load other into AX

inc ax ;(other++)

mov [other], ax ;Store the result back in other

end\_if:

; Exit

mov rax, SYS\_exit

mov rdi, EXIT\_SUCCESS

```

;terminate]

```

[In this spot, I'm going to insert screenshots of the changed in my register to show the values changing correctly according to which step of the program we're currently at. Then finish off with the value of num.

The screenshot displays the GNU Debugger (GDB) interface. The main window shows the assembly code for a program named 'multiple.asm'. The code is as follows:

```

3 ; unsigned short mul_3 = 0, other = 0;
4 ; if(num % 3 == 0 && num % 5 != 0) {
5 ;     mul_3++;
6 ; } else {
7 ;     other++;
8 ; }
9
10 section .data
11     SYS_exit    equ    60
12     EXIT_SUCCESS equ    0
13     num          dw    65535
14     mul_3        dw    0           ;mul_3 = 0
15     other        dw    0           ;other = 0
16
17 section .text
18     global _start
19
20 _start:
21
22     mov     ax, [num]
23
24     ;num % 3
25     mov     cx, 3
26     xor     dx, dx
27     div     cx
28     cmp     dx, 0           ;check num % 3 == 0
29     jne     else_block     ;If num % 3 != 0, jump to else_block
30
31     ;num % 5
32     mov     ax, [num]       ;Load 'num' again into AX (since div modifies AX)
33     mov     cx, 5           ;Set divisor as 5
34     xor     dx, dx         ;Clear DX for division
35     div     cx

```

The GDB console shows the following commands and output:

```

GNU GDB 3.3.12 (x86_64-pc-linux-gnu), by Dorothea LReading symbols from multiple...
(gdb) break multiple.asm:20
Breakpoint 1 at 0x401000: file multiple.asm, line 22.
(gdb) run
Starting program: /home/wyvernio/multiple/multiple

Breakpoint 1, _start () at multiple.asm:22
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi

```

The Register window on the right shows the state of the registers:

Register	Value	Comment
rax	0xffff	65535
rbx	0x0	0
rcx	0x3	3
rdx	0x0	0
rsi	0x0	0
rdi	0x0	0
rbp	0x0	0x0
rsp	0x7fffffff320	0x7fffffff320
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	0x40100f	0x40100f <_start+15>
eflags	0x246	[ PF ZF IF ]
cs	0x33	51
ss	0x2b	43
ds	0x0	0
es	0x0	0
fs	0x0	0
gs	0x0	0

The bottom of the window shows the status bar with 'Integer registers' and 'All registers' buttons, and 'Close' and 'Help' buttons.

```

19
20 _start:
21
22 mov     ax, [num]
23
24 ;num % 3
25 mov     cx, 3
26 xor     dx, dx
27 div     cx
28 cmp     dx, 0           ;check num % 3 == 0
29 jne     else_block      ;If num % 3 != 0, jump to else_block
30
31 ;num % 5
32 mov     ax, [num]       ;Load 'num' again into AX (since div modifies AX)
33 mov     cx, 5           ;Set divisor as 5
34 xor     dx, dx          ;Clear DX for division
35 div     cx
36 cmp     dx, 0           ;check num % 5 == 0
37 je      else_block      ;If num % 5 == 0, jump to else_block
38
39 if_block:
40 ;If num % 3 == 0 and num % 5 != 0, mul_3++
41 mov     ax, [mul_3]     ;Load mul_3 into AX
42 inc     ax              ;Increment AX (mul_3++)
43 mov     [mul_3], ax     ;Store the result
44 jmp     end_if          ;Jump to end of program
45
46 else_block:
47 ;Else, other++
48 mov     ax, [other]     ;Load other into AX
49 inc     ax              ;(other++)
50 mov     [other], ax     ;Store the result back in other
51

```

GNU DDD 3.3.12 (x86\_64-pc-linux-gnu), by Dorothea L

Reading symbols from multiple...

(gdb) break multiple.asm:20

Breakpoint 1 at 0x401000: file multiple.asm, line 22.

(gdb) run

Starting program: /home/wyvernio/multiple/multiple

Breakpoint 1, \_start () at multiple.asm:22

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

(gdb) stepi

DDD: Registers

Registers

rax	0x5555	21845
rbx	0x0	0
rcx	0x3	3
rdx	0x0	0
rsi	0x0	0
rdi	0x0	0
rbp	0x0	0x0
rsp	0x7fffffff320	0x7fffffff320
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	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

Close

Help

[illegible]

```

19
20 _start:
21 ■ mov ax, [num]
22
23 ;num % 3
24 mov cx, 3
25 xor dx, dx
26 div cx
27 cmp dx, 0 ;check num % 3 == 0
28 jne else_block ;If num % 3 != 0, jump to else_block
29
30 ;num % 5
31 mov ax, [num] ;Load 'num' again into AX (since div mod
32 mov cx, 5 ;Set divisor as 5
33 xor dx, dx ;Clear DX for division
34 div cx
35 cmp dx, 0 ;check num % 5 == 0
36 ➔ je else_block ;If num % 5 == 0, jump to else_block
37
38 if_block:
39 ;If num % 3 == 0 and num % 5 != 0, mul_3++
40 mov ax, [mul_3] ;Load mul_3 into AX
41 inc ax ;Increment AX (mul_3++)
42 mov [mul_3], ax ;Store the result
43 jmp end_if ;Jump to end of program
44
45 else_block:
46 ;Else, other++
47 mov ax, [other] ;Load other into AX
48 inc ax ;(other++)
49 mov [other], ax ;Store the result back in other
50
51

```

Starting program: /home/wyvernio/multiple/multiple

Breakpoint 1, \_start () at multiple.asm:22

```

(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi

```

DDD: Registers

Registers

rax	0x3333	13107
rbx	0x0	0
rcx	0x5	5
rdx	0x0	0
rsi	0x0	0
rdi	0x0	0
rbp	0x0	0x0
rsp	0x7fffffff320	0x7fffffff320
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	0x40102a	0x40102a <_start+42>
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

Close

Help

65535 / 3 is = 21845, as evidenced by this calculator.

Calculator

Programmer

HEX

5555

DEC

21,845

OCT

52 525

BIN

0101 0101 0101 0101

Memory

65535 ÷ 3 =

There's n

21,845

Our other answer is also correct, as well as the remainder being 0 for both  $\text{num} \% 3$  and  $\text{num} \% 5$ , as 65535 is divisible by both 3 and 5 respectively, meaning it cannot have a remainder.

And below this section is the section with the result of mul\_3 and other, where you can find the final register proof.

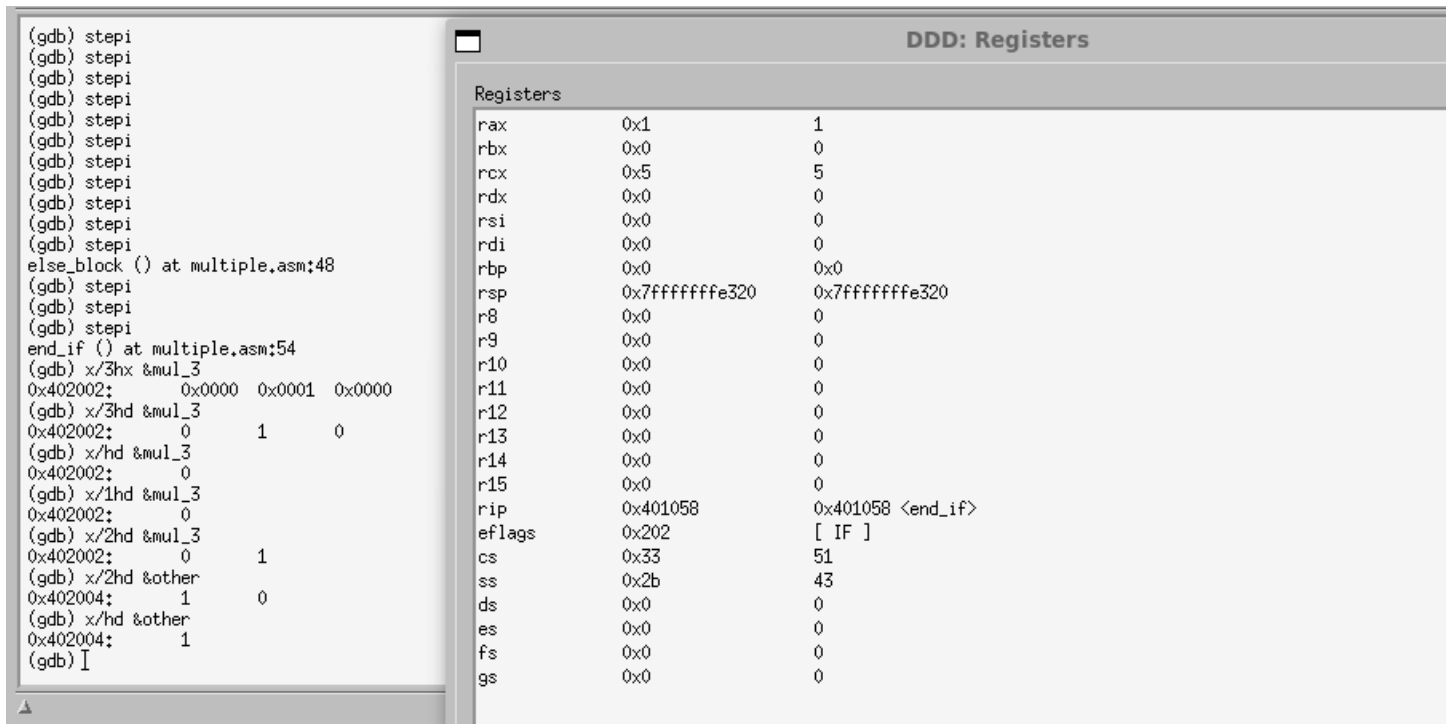
Here is num in both hexadecimal and decimal. ☺

```
25  mov    cx, 3
26  xor    dx, dx
27  div    cx
28  cmp    dx, 0           ;check num % 3 == 0
29  jne    else_block      ;If num % 3 != 0, jump to else_block
30
31  ;num % 5
32  mov    ax, [num]       ;Load 'num' again into AX (since div modifies AX)
33  mov    cx, 5           ;Set divisor as 5
34  xor    dx, dx          ;Clear DX for division
35  div    cx
36  cmp    dx, 0           ;check num % 5 == 0
37  je     else_block      ;If num % 5 == 0, jump to else_block
38
39 if_block:
40  ;If num % 3 == 0 and num % 5 != 0, mul_3++
41  mov    ax, [mul_3]     ;Load mul_3 into AX
42  inc    ax              ;Increment AX (mul_3++)
43  mov    [mul_3], ax      ;Store the result
44  jmp    end_if          ;Jump to end of program
45
46 else_block:
47  ;Else, other++
48  mov    ax, [other]     ;Load other into AX
49  inc    ax              ;Increment AX (other++)
50  mov    [other], ax     ;Store the result back in other
51
52 end_if:
53  ; Exit
54  mov    rax, SYS_exit
55  mov    rdi, EXIT_SUCCESS
56  syscall                ;terminate
57
```

```
else_block () at multiple.asm:48
(gdb) run
Starting program: /home/wyvernio/multiple/multiple

Breakpoint 1, _start () at multiple.asm:22
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
else_block () at multiple.asm:48
(gdb) stepi
(gdb) stepi
(gdb) stepi
end_if () at multiple.asm:54
(gdb) x/hd &num
0x402000: -1
(gdb) x/3hx &num
0x402000: 0xffff 0x0000 0x0001
(gdb) x/3hd &num
0x402000: -1 0 1
```

[



```
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
else_block () at multiple.asm:48
(gdb) stepi
(gdb) stepi
(gdb) stepi
end_if () at multiple.asm:54
(gdb) x/3hx &mul_3
0x402002: 0x0000 0x0001 0x0000
(gdb) x/3hd &mul_3
0x402002: 0 1 0
(gdb) x/hd &mul_3
0x402002: 0
(gdb) x/1hd &mul_3
0x402002: 0
(gdb) x/2hd &mul_3
0x402002: 0 1
(gdb) x/2hd &other
0x402004: 1 0
(gdb) x/hd &other
0x402004: 1
(gdb) ]
```

Registers		
rax	0x1	1
rbx	0x0	0
rcx	0x5	5
rdx	0x0	0
rsi	0x0	0
rdi	0x0	0
rbp	0x0	0x0
rsp	0x7fffffff320	0x7fffffff320
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	0x401058	0x401058 <end_if>
eflags	0x202	[ IF ]
cs	0x33	51
ss	0x2b	43
ds	0x0	0
es	0x0	0
fs	0x0	0
gs	0x0	0

The reason why mul\_3 is 0 is because mul\_3's conditions were not entirely fulfilled, num % 3 has to = 0 and num % 5 had to != 0, and since this was not true, mul\_3 returns as false, or 0.

This is further evidenced by “other” being equal to 1, as this means that the else section under num % 5 was executed, meaning that that the remainder of num being divided by 5 was 0.

Therefore, num % 5 = 0 was the reason why both mul\_3 is 0 and other is 1. ]