CPSC 240: Computer Organization and Assembly Language
Assignment 05, Fall Semester 2024

CWID:___884614918______________ Name:__Sean DeFrank___________________

Quiz Questions:

From the textbook "X86-64 Assembly Language Programming with Ubuntu," study quiz questions 5 and 6 on page 137. 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 Assignment05.docx” document.
2. Convert the following C/C++ variable declarations and arithmetic operations to x86-64 assembly language. Find an even number from the "array" array and copy that even number into the "even" array. NOTE: variable sizes and program functions should be equivalent to C/C++ instructions.
3. 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.

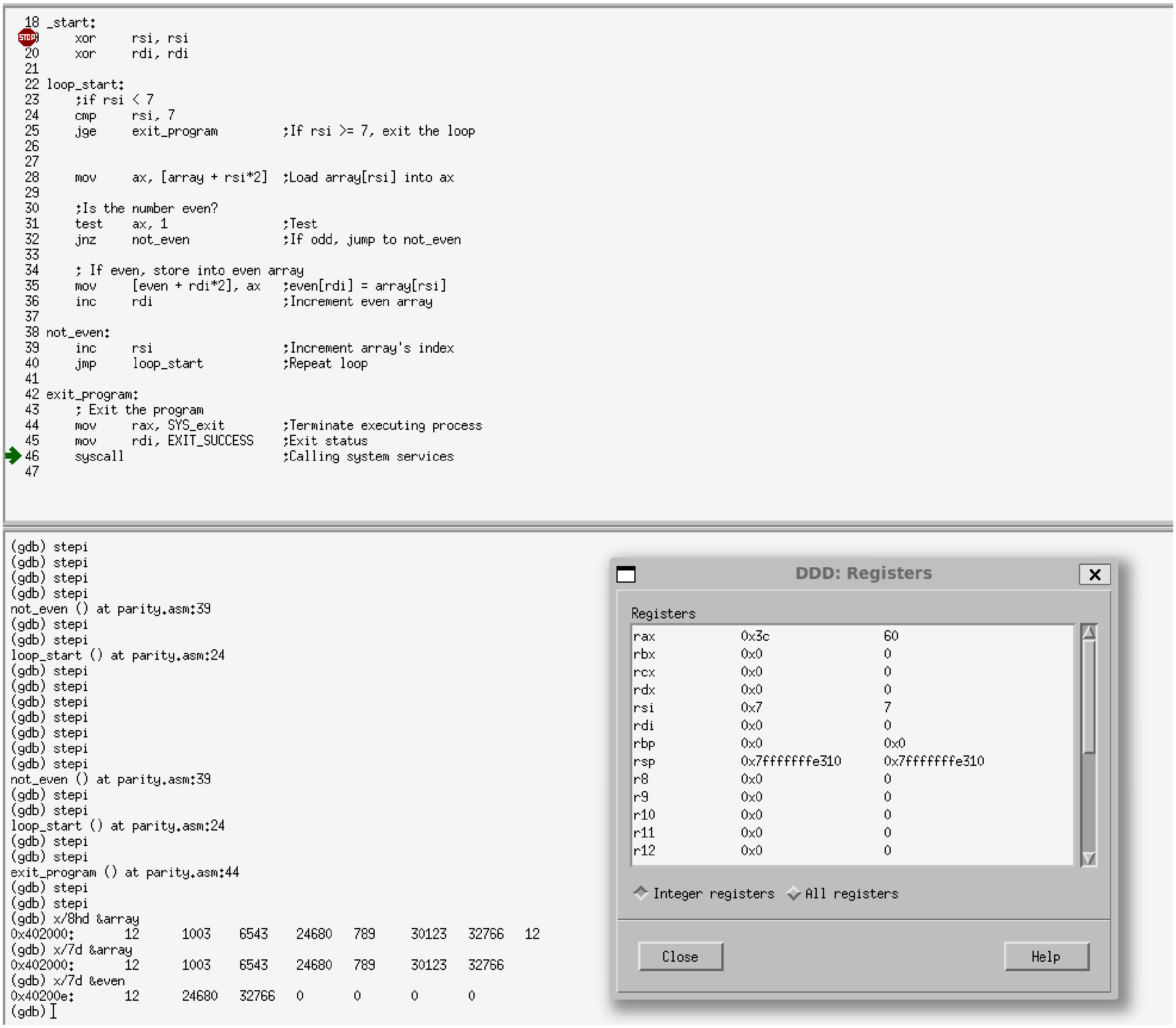
```
unsigned short array[7] = {12, 1003, 6543, 24680, 789, 30123, 32766};     // use dw for 16-bit array
unsigned short even[7];                              // use dw to declare 16-bit variable
register long rsi = 0, rdi = 0;                      // no need to declare register rsi and rdi
do {
        if(array[rsi] % 2 == 0) {
                even[rdi] = array[rsi];
                rdi++;
        }
        rsi++;
} while(rsi < 7);
```

4. Assemble the "doWhile.asm" file and link the "parity.o" file to get the "parity" executable file.
5. Run the "parity" file with the DDD/GDB debugger to display the simulation results of array and even.
6. Insert source code (parity.asm) and simulation results (GDB window) of the memory array (array and even) in the document. Use hand calculation to verify simulation results.
7. Save the file in pdf or docx format and submit the pdf or docx file to Canvas before the deadline.

```
[;parity.asm
;unsigned short array[7] = {12, 1003, 6543, 24680, 789, 30123, 32766};
;unsigned short even[7];
```

```
section .data
        SYS_exit                equu        60
        EXIT_SUCCESS        equ         0

        array       dw 12, 1003, 6543, 24680, 789, 30123, 32766
        even        times 7 dw 0

section .bss



section .text
        global _start
_start:
        xor         rsi, rsi
        xor         rdi, rdi

loop_start:
        ;if rsi < 7
        cmp         rsi, 7
        jge         exit_program                ;If rsi >= 7, exit the loop


        mov         ax, [array + rsi*2]    ;Load array[rsi] into ax

        ;Is the number even?
        test        ax, 1                           ;Test
        jnz         not_even                        ;If odd, jump to not_even

        ; If even, store into even array
        mov         [even + rdi*2], ax      ;even[rdi] = array[rsi]
        inc         rdi                             ;Increment even array

not_even:
        inc         rsi                             ;Increment array's index
        jmp         loop_start                      ;Repeat loop

exit_program:
        ; Exit the program
        mov         rax, SYS_exit                ;Terminate executing process
        mov         rdi, EXIT_SUCCESS       ;Exit status
        syscall                 ;Calling system services
```

[



```
_start:
    xor     rsi, rsi
    xor     rdi, rdi

loop_start:
    ;if rsi < 7
    cmp     rsi, 7
    jge     exit_program         ;If rsi >= 7, exit the loop


    mov     ax, [array + rsi*2]  ;Load array[rsi] into ax

    ;Is the number even?
    test    ax, 1                ;Test
    jnz     not_even             ;If odd, jump to not_even

    ; If even, store into even array
    mov     [even + rdi*2], ax   ;even[rdi] = array[rsi]
    inc     rdi                  ;Increment even array

not_even:
    inc     rsi                  ;Increment array's index
    jmp     loop_start           ;Repeat loop

exit_program:
    ; Exit the program
    mov     rax, SYS_exit        ;Terminate executing process
    mov     rdi, EXIT_SUCCESS    ;Exit status
    syscall                      ;Calling system services

```

```
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
not_even () at parity.asm:39
(gdb) stepi
(gdb) stepi
loop_start () at parity.asm:24
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
not_even () at parity.asm:39
(gdb) stepi
(gdb) stepi
loop_start () at parity.asm:24
(gdb) stepi
(gdb) stepi
exit_program () at parity.asm:44
(gdb) stepi
(gdb) stepi
(gdb) x/8hd &array
0x402000:       12      1003    6543    24680   789     30123   32766   12
(gdb) x/7d &array
0x402000:       12      1003    6543    24680   789     30123   32766
(gdb) x/7d &even
0x40200e:       12      24680   32766   0       0       0       0
(gdb) 
```

DDD: Registers

Registers

```
rax            0x3c                60
rbx            0x0                 0
rcx            0x0                 0
rdx            0x0                 0
rsi            0x7                 7
rdi            0x0                 0
rbp            0x0                 0x0
rsp            0x7fffffffe310      0x7fffffffe310
r8             0x0                 0
r9             0x0                 0
r10            0x0                 0
r11            0x0                 0
r12            0x0                 0
```

]

[Insert the simulation result verification here:

Our task is to take our array, and use a Do-While loop to iterate through the array and find all of it’s even numbers to place them into a different array called “even” while at the same time ignoring the odd numbers of the array, ensuring that only even numbers make it into the even array.
The array given holds the numbers: [12, 1003, 6543, 24680, 789, 30123, 32766].

On the first loop, we'll add 12 to our even array, as that is an even number.
On the second and third, we won't take 1003 and 6543, since they're odd.
On the fourth, we'll take 24680.
On the fifth and sixth, we won't take 789 and 30123, both odd.
On the seventh final loop, we'll take 32766, which is even.
Therefore, our final even array will be: [12, 24680, 32766, 0, 0, 0, 0].

The 0's are there because the array is empty, since it's still a 7 space array but we only put our 3 even numbers into it.

Our image above matches both our "array" and "even" arrays as stated, showing that the program fulfilled its intended purpose correctly.