Computer Science Department
California State University, Fullerton

CPSC 240 Computer Organization and Assembly Language
Quiz 02
3:00 pm to 4:15 pm
Thursday, October 19, 2023

Student Name: _____

Last 4 digits of ID: _____

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 to assemble the source code.
- Use "ld" linker to link the object code
- Use "ddd" 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 02

1. Download the "CPSC-240-09 Quiz 02.docx" document.
2. Convert the following C/C++ variable declarations and arithmetic operations to x86-64 assembly language. Use the "yasm" assembler to assemble the program, the "ld" linker to link the object code, and the "ddd" debugger to simulate the executable code.
   NOTE: variable sizes and program functions should be equivalent to C/C++ instructions.

```
signed char num[10] = {-12, 23, 34, 45, -56, 67, 78, -89, 90, 125}; //8-bit numeric array
signed char pOdd[10];                                   //8-bit non-initial array
register long rsi = 0                                   //64-bit register
register long rdi = 0                                   //64-bit register
register long rcx = 10                                  //64-bit register
do {
    if(num[rsi] > 0 && num[rsi]%2 != 0) {               //if positive odd number
        pOdd[rdi] = num[rsi];                           //copy to pOdd[rdi]
        rdi++;                                          //rdi = rdi + 1
    }
    rsi++;                                              //rsi = rsi + 1
    rcx--;                                              //rcx = rcx - 1
} while(rcx != 0);                                      //if rcx == 0 then finish
```

3. After assembling and linking, run the DDD debugger to display the simulation result values of num and pOdd in the memory.
4. Insert source code and the simulation results (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 4:15 pm on 10/19/2023.

[Attach your assembly source code here:]

```
;quiz02_9.asm
;signed char num[10] = {-12, 23, 34, 45, -56, 67, -78, 89, 90, 125};
                                                //8-bit numeric array
;signed char pOdd[10];                          //8-bit non-initial array
;register long rsi = 0                          //64-bit register
;register long rdi = 0                          //64-bit register
;register long rcx = 10                         //64-bit register
;do {
;     if(num[rsi] > 0 && num[rsi]%2 != 0) {     //if positive odd number
;             pOdd[rdi] = num[rsi];             //copy to pOdd[rdi]
;             rdi++;                            //rdi = rdi + 1
```

```
;     }
;     rsi++;                                    //rsi = rsi + 1
;     rcx--;                                    //rcx = rcx - 1
;} while(rcx != 0);                             //if rcx == 0 then finish

section .data
      num      db       -12, 23, 34, 45, -56, 67, 78, -89, 90, 125

section .bss
      pOdd     resb     10

section .text
      global _start
_start:
      mov      rsi, 0
      mov      rdi, 0
      mov      rcx, 10
doloop:
      cmp      byte[num+rsi], 0
      jl       end_if
      mov      al, byte[num+rsi]
      cbw
      mov      bl, 2
      idiv     bl
      cmp      ah, 0
      je       end_if
      mov      al, byte[num+rsi]
      mov      byte[pOdd+rdi], al
      inc      rdi
end_if:
      inc      rsi
      dec      rcx
      cmp      rcx,0
      jne      doloop

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

[Attach GDB window with all memory data here:]

```
(gdb) x/10db &num
0x402000:       -12     23      34      45      -56     67      78      -89
0x402008:       90      125
(gdb) x/10db &pOdd
0x40200c:       23      45      67      125     0       0       0       0
0x402014:       0       0
(gdb)
```