CPSC 240: Computer Organization and Assembly Language
Assignment 05, Fall Semester 2023

CWID:_885857847_____ Name:Kush Patel_____

1. Download the "CPSC-240 Assignment05.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.

```
unsigned short array[7] = {12, 1003, 6543, 24680, 789, 30123, 32766};
unsigned short even[7];
register long rsi = 0, rdi = 0;
do {
        if(array[rsi] % 2 == 0) {
                even[rdi] = array[rsi];
                rdi++;
        }
        rsi++;
} while(rsi < 7);
```

3. Assemble the "doWhile.asm" file and link the "parity.o" file to get the "parity" executable file.
4. Run the "parity" file with the DDD debugger to display the simulation results of array and even.
5. 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.
6. Save the file in pdf or docx format and submit the pdf or docx file to Canvas before 23:59 pm on 10/12/2023.

[Insert the source code of parity.asm here]

```
;unsigned short array[7] = {12, 1003, 6543, 24680, 789, 30123, 32766};
;unsigned short even[7];
;register long rsi = 0, rdi = 0;
;do {
;if(array[rsi] % 2 == 0) {
;even[rdi] = array[rsi];
;rdi++;
;}
;rsi++;
;} while(rsi < 7)

section .data
        array   dw      12, 1003, 6543, 24680, 789, 30123, 32766
        even    dw      0, 0, 0, 0, 0, 0, 0

section .bss
        max     resw    1

section .text
        global _start
_start:
        mov     rsi, 0                                          ;rsi = 0
doloop:
        mov     ax, word[array+(rsi*2)]
        mov     dx, 0
        mov     bx, 2
        div     bx                      ;ax = array[rsi]
        cmp     dx, 0                                           ;compare ax and max
        jne     not_even
        mov     ax, word[array+(rsi*2)]                                         ;if(ax>max) {
        mov     word[even+(rdi*2)], ax
        inc     rdi                                     ;    max = ax = array[rsi]
not_even:                                                       ;{
        inc     rsi                                             ;rsi = rsi + 1
        cmp     rsi, 7                                          ;compare rsi and 7
        jb      doloop                                          ;if(rsi<7) goto doloop

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

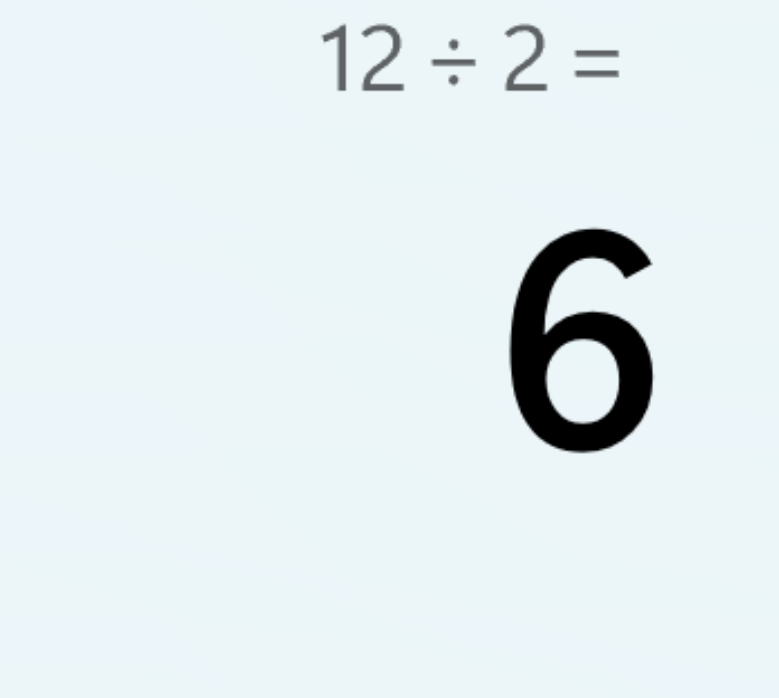
```
;unsigned short array[7] = {12, 1003, 6543, 24680, 789, 30123, 32766};
;unsigned short even[7];
;register long rsi = 0, rdi = 0;
;do {
;if(array[rsi] % 2 == 0) {
;even[rdi] = array[rsi];
;rdi++;
;}
;rsi++;
;} while(rsi < 7)

section .data
        array   dw      12, 1003, 6543, 24680, 789, 30123, 32766
        even    dw      0, 0, 0, 0, 0, 0, 0

section .bss
        max     resw    1

section .text
        global _start
_start:
        mov     rsi, 0
        mov     rdi, 0                                  ;rsi = 0
doloop:
        mov     ax, word[array+(rsi*2)]
        mov     dx, 0
        mov     bx, 2
        div     bx                      ;ax = array[rsi]
        cmp     dx, 0                                   ;compare ax and max
        jne     not_even
        mov     ax, word[array+(rsi*2)]                                 ;if(ax>max) {
        mov     word[even+(rdi*2)], ax
        inc     rdi                                 ;   max = ax = array[rsi]
not_even:                                               ;{
        inc     rsi                                     ;rsi = rsi + 1
        cmp     rsi, 7                                  ;compare rsi and 7
        jb      doloop                                  ;if(rsi<7) goto doloop

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

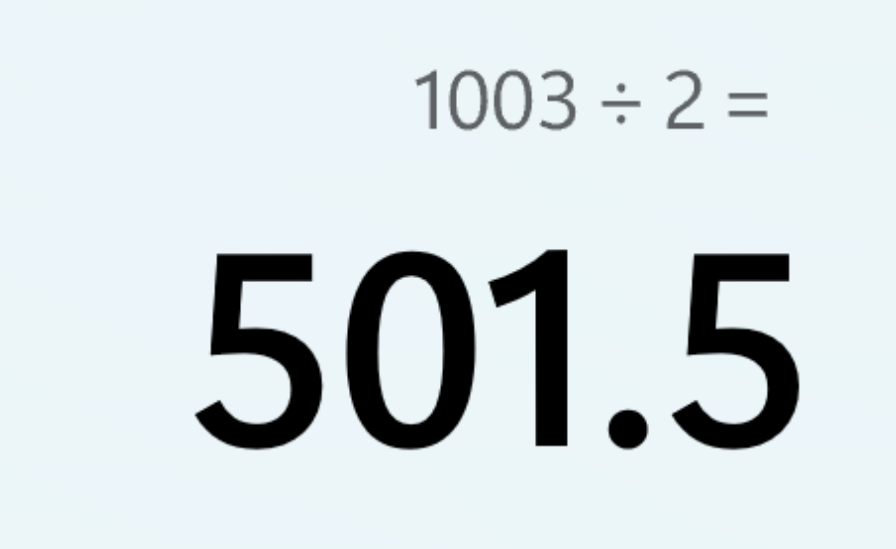
[Insert parity simulation result (GDB window with array and even) here]

```
(gdb) x/7uh &array
0x402000:       12      1003    6543    24680   789     30123   32766
(gdb) x/7uh &even
0x40200e:       12      24680   32766   0       0       0       0
(gdb)
```

```
0x40200e: 12 24680 32766 0 0 0 0
```

[Insert verification of hand calculation here]





$$6543 \div 2 =$$

$$3{,}271.5$$

$$24680 \div 2 =$$

$$12{,}340$$

$$789 \div 2 =$$

$$394.5$$

30123 ÷ 2 =

15,061.5

32766 ÷ 2 =

16,383

32766 ÷ 2 =

16,383

30123 ÷ 2 =

15,061.5

789 ÷ 2 =

394.5

24680 ÷ 2 =

12,340

6543 ÷ 2 =

Input two numbers

Dividend 12

Divisor 2

Result

Quotient 6

Remainder 0

12 / 2 = 6 R 0

Check the result:

Input two numbers

Dividend 1003

Divisor 2

Result

Quotient 501

Remainder 1

1,003 / 2 = 501 R 1

Check the result:

Input two numbers

Dividend 6543

Divisor 2

Result

Quotient 3,271

Remainder 1

6,543 / 2 = 3,271 R 1

Check the result:

Input two numbers

Dividend 24680

Divisor 2

Result

Quotient 12,340

Remainder 0

24,680 / 2 = 12,340 R 0

Check the result:

Input two numbers

Dividend 789

Divisor 2

Result

Quotient 394

Remainder 1

789 / 2 = 394 R 1

Check the result:

Input two numbers

Dividend 30123

Divisor 2

Result

Quotient 15,061

Remainder 1

30,123 / 2 = 15,061 R 1

Input two numbers

Dividend 32766

Divisor 2

Result

Quotient 16,383

Remainder 0

32,766 / 2 = 16,383 R 0