

CPSC 240: Computer Organization and Assembly Language

Assignment 04, Spring Semester 2023

CWID: [886880327](#) Name: [Andrew Saldana](#)

1. Download the "CPSC-240 Assignment04.docx" document.
2. Design the "parity.asm" program, and use assembly language to realize the function of the following C++ instructions.
unsigned short num1 = 325;
unsigned short even = 0, odd = 0;
if(num1 % 2 == 0) {
 even++;
} else {
 odd++;
}
}
3. Assemble the "parity.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 memory of **num1**, as well as the simulation results of **even** and **odd**.
5. Insert source code (parity.asm) and simulation results (GDB window) of the memory (num1, even, and odd) 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 23:59 pm on 03/05/2023.

[Insert parity.asm source code here]

```
1 ; unsigned short num1 = 325;
2 ; unsigned short even = 0, odd = 0;
3 ; if(num1 % 2 == 0) {
4 ;     even++;
5 ; } else {
6 ;     odd++;
7 ; }
8
9 section .data
0     num1    dw 325                ;num1 = 0145h = 325
1     even    dw 0                  ;even = 0000h = 0
2     odd     dw 0                  ;odd = 0000h = 0
3
4 section .text
5     global _start
6
7 _start:
8     mov ax, word[num1]            ;ax = num1 = 0145h = 325
9     mov cx, 2                     ;cx = 2 = 0000h
0     div cx                        ;ax = ax/cx = 325/2 = 162 = 00A2h
1                                 ;dx = ax%cx = 1 = 0001h
2     cmp dx, 0                     ;compare dx(1) with 0
3     jne is_odd                    ;if dx != 0, jump to is_odd label
4
5 is_even:
6     inc word[even]                ;even++
7     jmp end                       ;jump to end of program to terminate
8 is_odd:
9     inc word[odd]                 ;odd++
0     jmp end                       ;jump to end of program to terminate
1
2 end:
3     mov rax, 60
4     mov rdi, 0
5     syscall
6
7
```

Plain Text ▾ Tab Width: 8 ▾ Line Col 14 ▾ INC

[Insert parity simulation result here]

```

4 ;     even++;
5 ; } else {
6 ;     odd++;
7 ; }
8
9 section .data
10     num1    dw 325           ;num1 = 0145h = 325
11     even    dw 0            ;even = 0000h = 0
12     odd     dw 0            ;odd = 0000h = 0
13
14 section .text
15     global _start
16
17 _start:
18     mov ax, word[num1]      ;ax = num1 = 0145h = 325
19     mov cx, 2               ;cx = 2 = 0000h
20     div cx                  ;ax = ax/cx = 325/2 = 162 = 00A2h
21                             ;dx = ax%cx = 1 = 0001h
22     cmp dx, 0               ;compare dx(1) with 0
23     jne is_odd              ;if dx != 0, jump to is_odd label
24
25 is_even:
26     inc word[even]          ;even++
27     jmp end                 ;jump to end of program to terminate
28 is_odd:
29     inc word[odd]           ;odd++
30     jmp end                 ;jump to end of program to terminate
31
32 end:
33     mov rax, 60
34     mov rdi, 0
35     syscall

```

DDD: Registers

Registers		
rax	0xa2	162
rbx	0x0	0
rcx	0x2	2
rdx	0x1	1
rsi	0x0	0
rdi	0x0	0
rbp	0x0	0x0
rsp	0x7fffffff060	0x7ff
r8	0x0	0
r9	0x0	0
r10	0x0	0
r11	0x0	0
r12	0x0	0

Integer registers
All registers

Close
Help

```

(gdb) x/dh &num1
0x402000: 325
(gdb) x/xh &num1
0x402000: 0x0145
(gdb) x/dh &even
0x402002: 0
(gdb) x/xh &even
0x402002: 0x0000
(gdb) x/dh &odd
0x402004: 1
(gdb) x/xh &odd
0x402004: 0x0001

```

[Insert parity simulation result analysis here]

325 in hex

$$\begin{array}{rcl} 325/16 & = & 20 \text{ R } 5 \\ 20/16 & = & 1 \text{ R } 4 \\ 1/16 & = & 0 \text{ R } 1 \end{array} \left. \vphantom{\begin{array}{rcl} 325/16 \\ 20/16 \\ 1/16 \end{array}} \right\} \begin{array}{l} \text{hex (16 bits)} \\ 0145\text{h} \end{array}$$

$$325/2 = 162 \text{ R } 1$$

162 in hex

$$\begin{array}{rcl} 162/16 & = & 10 \text{ R } 2 \\ 10/16 & = & 0 \text{ R } 10 \end{array} \left. \vphantom{\begin{array}{rcl} 162/16 \\ 10/16 \end{array}} \right\} \begin{array}{l} \text{hex in 16 bits} \\ 00A2\text{h} \end{array}$$

		362/2		Compare dx(i) and 0 jne is-odd		
num	divisor	ax	dx	jump?	even	odd odd++
325	2	162	1	yes, jump to is-odd:	0	1

odd = 1
even = 0