Description of the Program:

The purpose of this program is to accept two integers from the user (x, y), and find the sum of all even integers between [x, y], i.e. x = 3, y = 10: **4 + 6 + 8 + 10** = **28**

Algorithm Analysis:

The algorithm I used is very straight forward, I first accept the user inputs, and check them to see if x > y, by this program's definitions x must be < y. An error message will be displayed, if x > y. Then I check to see if x and y are odd or even using **andi** and **beq** commands to jump to locations and 'fix' them if they are odd to make them even. I add one to x if it is odd since it is the bottom bound, and subtract one from y if it is odd, since it is the top bound. I then go into a very simple for loop that has an exit condition of x >= y, if true it jumps to the end of the program, and prints out the results. If is false it iterates the loop one more time, adding two to x, and adding x to a sum variable.

Outputs:

```
Please enter first int: 3
Please enter first int: 10
Sum from [x, y]: 28
End of program
```

Proof: [3, 10] = 4 + 6 + 8 + 10 = 28

```
Please enter first int: -8
Please enter first int: -3
Sum from [x, y]: -18
End of program
```

Proof: [-8, -3] = -8 + (-6) + (-4) = -18

```
Please enter first int: -5
Please enter first int: 9
Sum from [x, y]: 14
End of program
```

Proof: [-5, 9] = -4 + (-2) + 2 + 4 + 6 + 8 = 14

Source Code on Pg. 2

```
1: #Travis Ritter, Section: 01
2: .data
3: Prompt1: .asciiz "Please enter first int: "
4: Prompt2: .asciiz "Please enter second int: "
5: Prompt3: .asciiz "Sum from [x, y]: "
6: End: .asciiz "\nEnd of program"
7: Error: .asciiz "First number MUST be less than second"
8:
9: .text
10: main: #loads prompts, accepts user input, and checks if x > y
11:     li $v0, 4 ##load string print service
12:     la $a0, Prompt1 ##load string into a0
13:     syscall ## print
14:     li $v0, 5 ##load int reading service
15:     syscall ##read an int
16:     move $s1, $v0 ##store the input (x)
17:
18:     li $v0, 4 ##load string print service
19:     la $a0, Prompt1 ##load string into a0
20:     syscall ## print
21:     li $v0, 5 ##load int reading service
22:     syscall ##read an int
23:     move $s2, $v0 ##store the input (y)
24:
25:     sgt $t0, $s1, $s2 #x must be less than y, so if it is greater than, that
26:               #is an error
27:     beq $t0, 1, error #if this equals 1, then x > y, which is not allowed
28:
29:     andi $t0, $s1, 1 #checks the rightmost bit against 1
30:     beq $t0, 1, fixX #if it does equal 1, that means that the last bit is 1, meaning
31:              #it is odd then it jumps and adds one, to make it even
32:
33: checkY: #check to see if Y is odd, if it is jump to make it even
34:     andi $t0, $s2, 1 #checks the rightmost bit against 1
35:     beq $t0, 1, fixY #if it does equal 1, that means that the last bit is 1, meaning
36:              #it is odd then it jumps and subtracts one, to make it even
37:
38: loop: #for loop that adds the even integers from [x, y]
39:     add $s3, $s3, $s1 #add s1 (x) to the total
40:     addi $s1, $s1, 2 #increment the first number by 2, since we are adding even number
s
41:     bgt $s1, $s2, end #check to see if the first number is
42:     j loop #if it doesn't branch, then jump back to the top and loop again
43:
44: end: #prints out results and exits program
45:     li $v0, 4 #load string print service
46:     la $a0, Prompt3 #load message into a0
47:     syscall #print message
48:     move $a0, $s3 #move the sum total to a0
49:     li $v0, 1 #load int print service
50:     syscall #print int
```

```
51:     li $v0, 4 #load string print service
52:     la $a0, End #load ending message into a0
53:     syscall #print end message
54:     li $v0, 10 #load system exit
55:     syscall #exit program
56:
57: error: #prints out error when x > y, and exits program
58:     li $v0, 4 #load string print service
59:     la $a0, Error #load error message
60:     syscall #print error message
61:     li $v0, 10 #load system exit
62:     syscall #exit
63:
64: fixX: #used to correct x when it is odd, and make it even
65:     addi $s1, $s1, 1 #add 1 to the x to make it even
66:     j checkY #jump to thenext check
67: fixY: #used to correct y when it is odd, and make it even
68:     addi $s2, $s2, -1 #subtract 1 to y to make it even
69:     j loop #jump to the start of the loop
```

C Source Code:

```c
#include <stdio.h>

int main(void) {
  printf("Enter: ");
  int input1;
  scanf("%d", &input1);

  printf("Enter 2: ");
  int input2;
  scanf("%d", &input2);

  int sum = 0;
  if(input1 < input2) {
    for(int i = input1; i <= input2; i++) {
      if(i % 2 == 0) {
        sum += i;
      }
    }
  } else {
    printf("First input must be less than second");
  }

  printf("%d", sum);

  return 0;
}
```