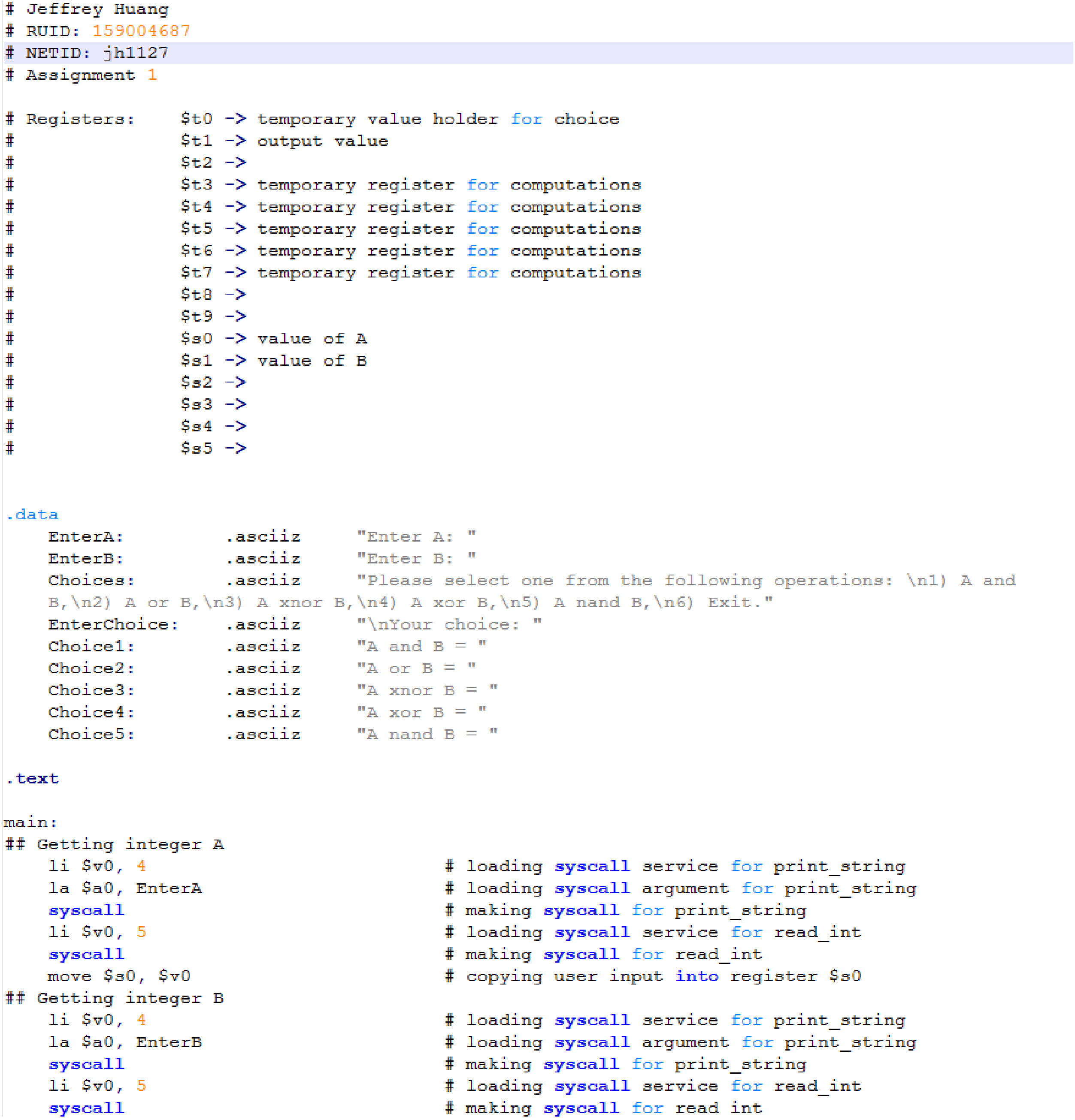
2016

Computer Architecture Lab

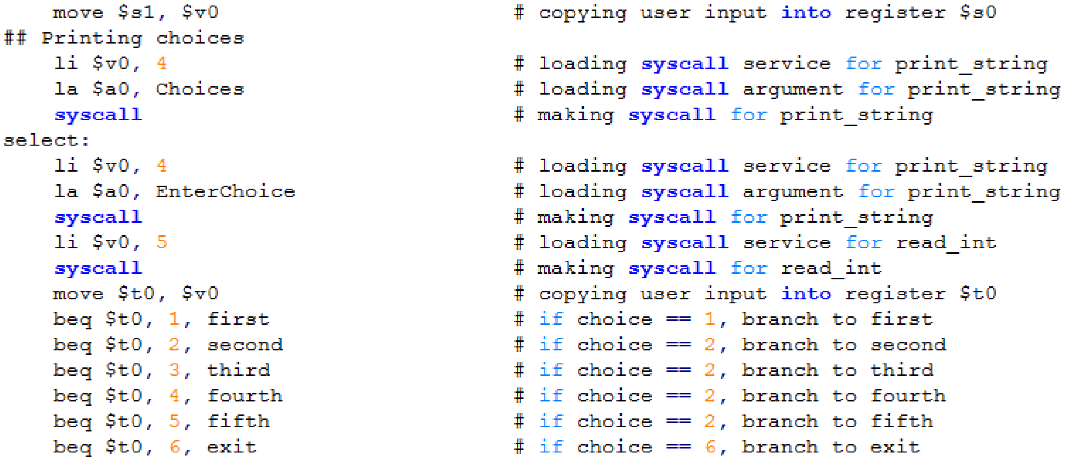
LAB 3 : ARITHMETIC OPERATIONS (BASIC AND FLOAT) AND COMBINATORIAL LOGIC

JEFFREY HUANG | RUID: 159-00-4687

Code Used for Assignment 1



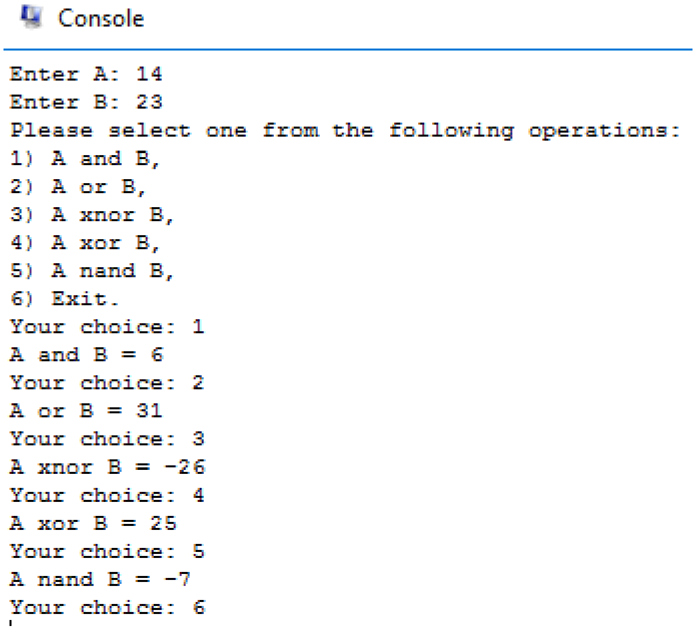
```
# Jeffrey Huang
# RUID: 159004687
# NETID: jh1127
# Assignment 1

# Registers:    $t0 -> temporary value holder for choice
#               $t1 -> output value
#               $t2 ->
#               $t3 -> temporary register for computations
#               $t4 -> temporary register for computations
#               $t5 -> temporary register for computations
#               $t6 -> temporary register for computations
#               $t7 -> temporary register for computations
#               $t8 ->
#               $t9 ->
#               $s0 -> value of A
#               $s1 -> value of B
#               $s2 ->
#               $s3 ->
#               $s4 ->
#               $s5 ->


.data
    EnterA:         .asciiz     "Enter A: "
    EnterB:         .asciiz     "Enter B: "
    Choices:        .asciiz     "Please select one from the following operations: \n1) A and
    B,\n2) A or B,\n3) A xnor B,\n4) A xor B,\n5) A nand B,\n6) Exit."
    EnterChoice:    .asciiz     "\nYour choice: "
    Choice1:        .asciiz     "A and B = "
    Choice2:        .asciiz     "A or B = "
    Choice3:        .asciiz     "A xnor B = "
    Choice4:        .asciiz     "A xor B = "
    Choice5:        .asciiz     "A nand B = "

.text

main:
## Getting integer A
    li $v0, 4                               # loading syscall service for print_string
    la $a0, EnterA                          # loading syscall argument for print_string
    syscall                                 # making syscall for print_string
    li $v0, 5                               # loading syscall service for read_int
    syscall                                 # making syscall for read_int
    move $s0, $v0                           # copying user input into register $s0
## Getting integer B
    li $v0, 4                               # loading syscall service for print_string
    la $a0, EnterB                          # loading syscall argument for print_string
    syscall                                 # making syscall for print_string
    li $v0, 5                               # loading syscall service for read_int
    syscall                                 # making syscall for read int
```



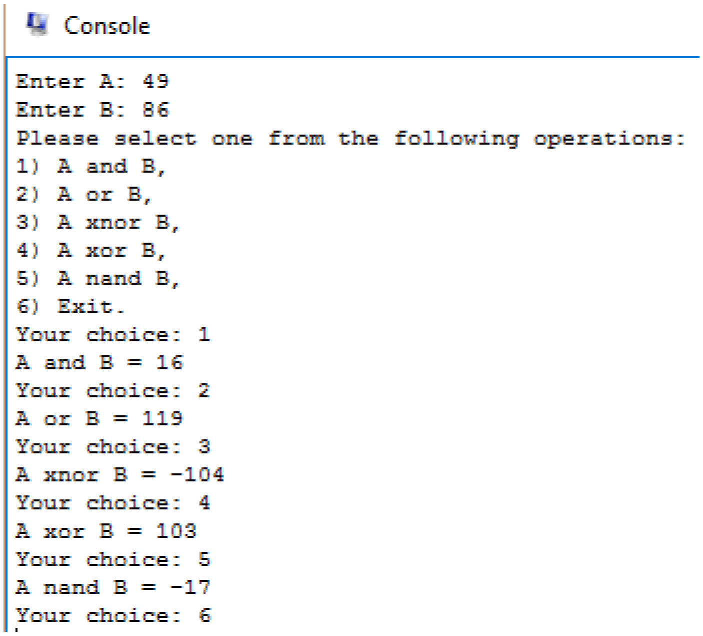
```
    move $s1, $v0                           # copying user input into register $s0
## Printing choices
    li $v0, 4                               # loading syscall service for print_string
    la $a0, Choices                         # loading syscall argument for print_string
    syscall                                 # making syscall for print_string
select:
    li $v0, 4                               # loading syscall service for print_string
    la $a0, EnterChoice                     # loading syscall argument for print_string
    syscall                                 # making syscall for print_string
    li $v0, 5                               # loading syscall service for read_int
    syscall                                 # making syscall for read_int
    move $t0, $v0                           # copying user input into register $t0
    beq $t0, 1, first                       # if choice == 1, branch to first
    beq $t0, 2, second                      # if choice == 2, branch to second
    beq $t0, 3, third                       # if choice == 2, branch to third
    beq $t0, 4, fourth                      # if choice == 2, branch to fourth
    beq $t0, 5, fifth                       # if choice == 2, branch to fifth
    beq $t0, 6, exit                        # if choice == 6, branch to exit
```

```
first:
    #nor $t3, $s0, $s0                  # A nor A
    #nor $t4, $s1, $s1                  # B nor B
    #nor $t1, $t3, $t4                  # (A nor A) nor (B nor B)
    and $t1, $s0, $s1                   # A and B
    li $v0, 4                           # loading syscall service for print_string
    la $a0, Choice1                     # loading syscall argument for print_string
    syscall                             # making syscall for print_string
    li $v0, 1                           # loading syscall service for print_integer
    move $a0, $t1                           # loading syscall argument for print_integer
    syscall                             # making syscall for print_integer
    j select                            # jump to select

second:
    #nor $t3, $s0, $s1                  # A nor B
    #nor $t1, $t3, $t3                  # (A nor B) nor (A nor B)
    or $t1, $s0, $s1                    # A or B
    li $v0, 4                           # loading syscall service for print_string
    la $a0, Choice2                     # loading syscall argument for print_string
    syscall                             # making syscall for print_string
    li $v0, 1                           # loading syscall service for print_integer
    move $a0, $t1                       # loading syscall argument for print_integer
    syscall                             # making syscall for print_integer
    j select                            # jump to select

third:
    nor $t3, $s0, $s1                   # A nor B
    nor $t4, $s0, $t3                   # A nor (A nor B)
    nor $t5, $s1, $t3                   # B nor (A nor B)
    nor $t1, $t5, $t4                   # (A nor (A nor B)) nor (B nor (A nor B))
    li $v0, 4                           # loading syscall service for print_string
    la $a0, Choice3                     # loading syscall argument for print_string
    syscall                             # making syscall for print_string
    li $v0, 1                           # loading syscall service for print_integer
    move $a0, $t1                       # loading syscall argument for print_integer
    syscall                             # making syscall for print_integer
    j select                            # jump to select

fourth:
    nor $t3, $s0, $s0                   # A nor A -> $t3
    nor $t4, $s1, $s1                   # B nor B -> $t4
    nor $t5, $s0, $s1                   # A nor B -> $t5
    nor $t6, $t3, $t4                   # (A nor A) nor (B nor B)
    nor $t1, $t6, $t5                   # ((A nor A) nor (B nor B)) nor (A nor B)
    li $v0, 4                           # loading syscall service for print_string
    la $a0, Choice4                     # loading syscall argument for print_string
    syscall                             # making syscall for print_string
    li $v0, 1                           # loading syscall service for print_integer
    move $a0, $t1                       # loading syscall argument for print_integer
    syscall                             # making syscall for print_integer
fifth:
    nor $t3, $s0, $s0                   # A nor A -> $t3
    nor $t4, $s1, $s1                   # B nor B -> $t4
    nor $t5, $t3, $t4                   # (A nor A) nor (B nor B)
    nor $t1, $t5, $t5                   # ((A nor A) nor (B nor B)) nor ((A nor A) nor (B nor B))
    li $v0, 4                           # loading syscall service for print_string
    la $a0, Choice5                     # loading syscall argument for print_string
    syscall                             # making syscall for print_string
    li $v0, 1                           # loading syscall service for print_integer
    move $a0, $t1                           # loading syscall argument for print_integer
    syscall                             # making syscall for print_integer
    j select                            # jump to select

exit:
    li $v0, 10                          # loading syscall service for end_program
    syscall                             # making syscall for end_program
```

Outputs for Assignment 1



```
Enter A: 14
Enter B: 23
Please select one from the following operations:
1) A and B,
2) A or B,
3) A xnor B,
4) A xor B,
5) A nand B,
6) Exit.
Your choice: 1
A and B = 6
Your choice: 2
A or B = 31
Your choice: 3
A xnor B = -26
Your choice: 4
A xor B = 25
Your choice: 5
A nand B = -7
Your choice: 6
```

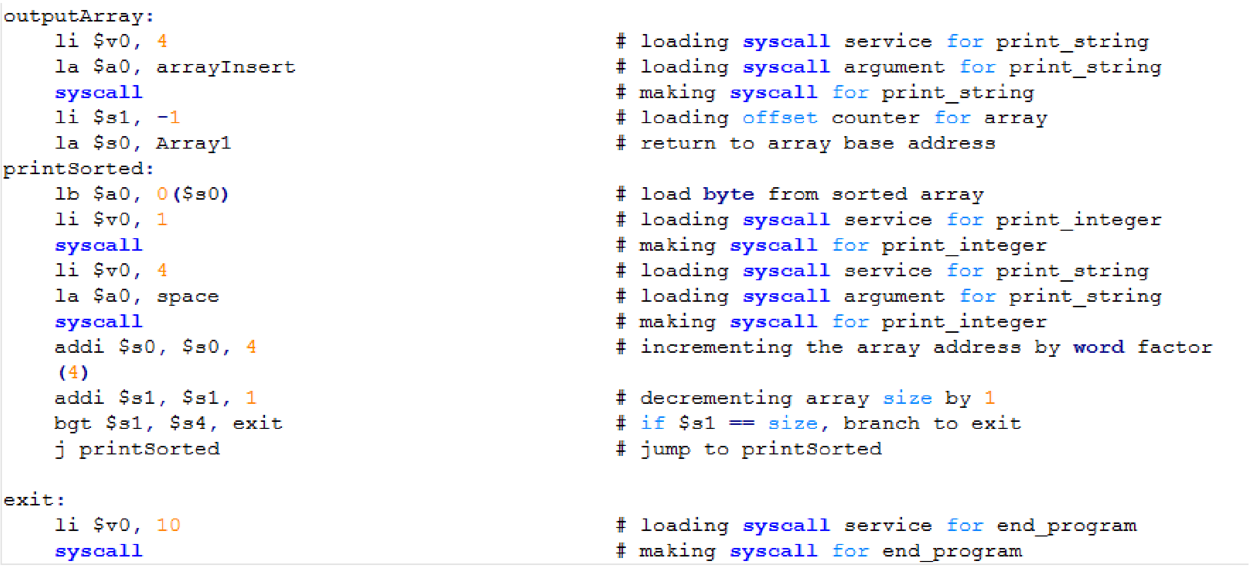


```
Enter A: 49
Enter B: 86
Please select one from the following operations:
1) A and B,
2) A or B,
3) A xnor B,
4) A xor B,
5) A nand B,
6) Exit.
Your choice: 1
A and B = 16
Your choice: 2
A or B = 119
Your choice: 3
A xnor B = -104
Your choice: 4
A xor B = 103
Your choice: 5
A nand B = -17
Your choice: 6
```

Code Used for Assignment 2

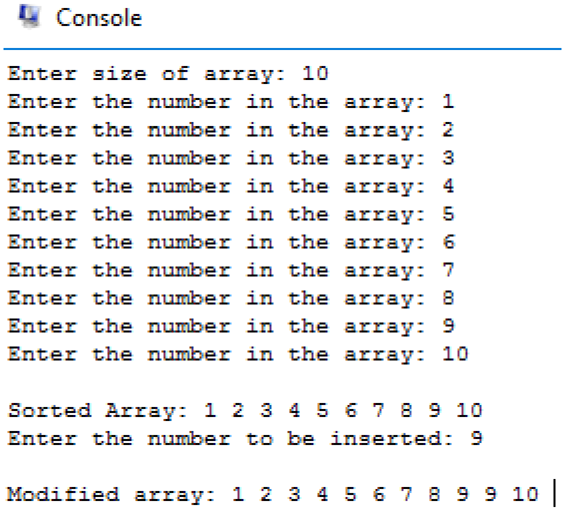
```
# Jeffrey Huang
# RUID: 159004687
# NETID: jh1127
# Assignment 2

# Registers:    $t0 -> array index counter for Array1
#               $t1 -> temporary array base address (Array1)
#               $t2 ->
#               $t3 ->
#               $t4 ->
#               $t5 ->
#               $t6 -> value LO
#               $t7 ->
#               $t8 ->
#               $t9 ->
#               $s0 -> Array1 base location
#               $s1 -> array size (abitrary value)
#               $s2 ->
#               $s3 -> number to be inserted
#               $s4 -> copy of array size
#               $s5 ->

.data
    Array1:         .space      400
    arraySize:      .asciiz     "Enter size of array: "
    inputArr:       .asciiz     "Enter the number in the array: "
    inputNum:       .asciiz     "\nEnter the number to be inserted: "
    arraySort:      .asciiz     "\nSorted Array: "
    arrayInsert:    .asciiz     "\nModified array: "
    space:          .asciiz     " "

.text

main:
    la $s0, Array1                              # loading array base address to $s0
    move $t1, $s0                               # moving array base address to $t1
    li $v0, 4                                   # loading syscall service for print_string
    la $a0, arraySize                           # loading syscall argument for print_string
    syscall                                     # making syscall for print_string
    li $v0, 5                                   # loading syscall service for read_int
    syscall                                     # making syscall for read_int
    move $s1, $v0                               # moving array size to register
    addi $s1, $v0, -1                           # size-- for index adjustment
    move $s4, $s1                               # copy of array size
## Getting user input
input:
    bgt $t0, $s1, endInput                      # if counter == size, branch to endInput
    li $v0, 4                                   # loading syscall service for print_string
    la $a0, inputArr                            # loading syscall argument for print_string
    syscall                                     # making syscall for print_string
    li $v0, 5                                   # loading syscall service for read_int
    syscall                                      # making syscall for read_int
    sw $v0, ($t1)                                # storing input into array
    addi $t0, $t0, 1                             # incrementing the array index counter
    addi $t1, $t1, 4                             # incrementing the address by 4 for word
    j input                                      # jump to input
endInput:
    la $s0, Array1                               # loading base address from Array1
    li $t0, 0                                    # changing the index counter to 0

## Bubblesort algorithm
outerLoop:
    beq $t0, $s1, printArraySort                 # if counter == size, branch to intialize
    addi $t0, $t0, 1                             # array index ++
    li $t1, 0                                    # changing base address to 0
    move $t3, $s0                                # moving the base array address to $t3

innerLoop:
    beq $t1, $s1, outerLoop                      # if $t1 == $s1, branch to outerLoop
    lw $t4, ($t3)                                # loading first element to $t4
    lw $t5, 4($t3)                               # loading second element to $t5
    addi $t1, $t1, 1                             # incrementing array counter by 1
    blt $t5, $t4, swap                           # if $t5 < $t4, branch to swap
    addi $t3, $t3, 4                             # increment the counter by 4 for word
    j innerLoop                                  # jump to innerLoop

swap:
    sw $t5, ($t3)                                # storing the second element as first element
    sw $t4, 4($t3)                               # storing the first element as second element
    addi $t3, $t3, 4                             # increment the counter by 4 for word
    j innerLoop                                  # jump to innerLoop

printArraySort:
    li $v0, 4                                    # loading syscall service for print_string
    la $a0, arraySort                            # loading syscall argument for print_string
    syscall                                      # making syscall for print_string
printArray:
    blt $s1, $0, getInput                        # if $s1 < 0, branch to getInput
    lb $a0, 0($s0)                               # load byte from sorted array
    li $v0, 1                                    # loading syscall service for print_integer
    syscall                                      # making syscall for print_integer
    li $v0, 4                                    # loading syscall service for print_string
    la $a0, space                                # loading syscall argument for print_string
    syscall                                      # making syscall for print_integer
    addi $s0, $s0, 4                             # incrementing the array address by word factor
    (4)
    addi $s1, $s1, -1                            # decrementing array size by 1
    j printArray                                 # jump to printArray

getInput:
    move $s1, $s4                                # copying original array size
    li $v0, 4                                    # loading syscall service for print_string
    la $a0, inputNum                             # loading syscall argument for print_string
    syscall                                      # making syscall for print_string
    li $v0, 5                                    # loading syscall service for read_int
    syscall                                      # making syscall for read_int
    move $s3, $v0                                # moving number to be inserted to $s3
    j binSearchStart                             # jump to binSearchStart
```

```
## Binary Search Algorithm
binSearchStart:
    la $t1, Array1                              # loading Array1 into $s0
    li $t4, 0                                   # loading beginning location
    move $t5, $s4                               # size--
    li $s0, 2                                   # temporary division by 2

binSearch:
    add $t6, $t5, $t4                           # ( HI + LO )
    div $t6, $s0                                # ( HI + LO ) / 2
    move $s1, $t6                               # moving LO to temporary register
    sll $t6, $t6, 2                             # shifting two into mid
    add $t1, $t1, $t6                           # array[mid]
    lw $t7, 0($t1)                              # temp = array[mid]
    bgt $s3, $t7, greaterThan                   # if input > array[mid], branch to greaterThan
    blt $s3, $t7, lessThan                      # if input < array[mid], branch to lessThan
    sub $t1, $t1, $t6                           # else input == array[mid]
    move $t5, $s1                               # saving input value to location
    j shiftArray                                # jump to shiftArray

greaterThan:
    bge $t4, $t5, insertAfter                   # if beginning >= size, then branch to
    insertAfter
    addi $t4, $s1, 1                            # else LO = mid + 1
    sub $t1, $t1, $t6                           # returning to array base address
    j binSearch                                 # jump to binSearch

lessThan:
    bge $t4, $t5, insertBefore                  # if beginning >= size, then branch to
    insertBefore
    addi $t5, $s1, -1                           # else HI = mid - 1
    sub $t1, $t1, $t6                           # returning to array base address
    j binSearch                                 # jump to binSearch

insertAfter:
    sub $t1, $t1, $t6                           # returning to array base address
    addi $t5, $t5, 1                            # size++
    j shiftArray                                # jump to shiftArray

insertBefore:
    move $t5, $t4                               # starting position is insert location
    sub $t1, $t1, $t6                           # returning to array base address
    move $s1, $t6                               # insert at current position

shiftArray:
    sll $t6, $t5, 2                             # finding the address with offset
    add $t1, $t1, $t6                           # returning array to base address
    lw $t4, 0($t1)                              # storing previous value temporarily
    sw $s3, 0($t1)                              # inserting the new value
    move $s3, $t4                               # replacing the inserted value with previous
    value
    addi $t1, $t1, 4                            # incrementing the address by 4
shift:
    lw $t4, 0($t1)                              # storing previous value temporarily
    sw $s3, 0($t1)                              # inserting the new value
    move $s3, $t4                               # replacing the inserted value with previous
    value
    addi $t5, $t5, 1                            # incrementing the array counter
    bgt $t5, $t0, outputArray                   # if counter >= size, branch to
    addi $t1, $t1, 4                            # incrementing the address by 4
    j shift                                     # jump to shift
```

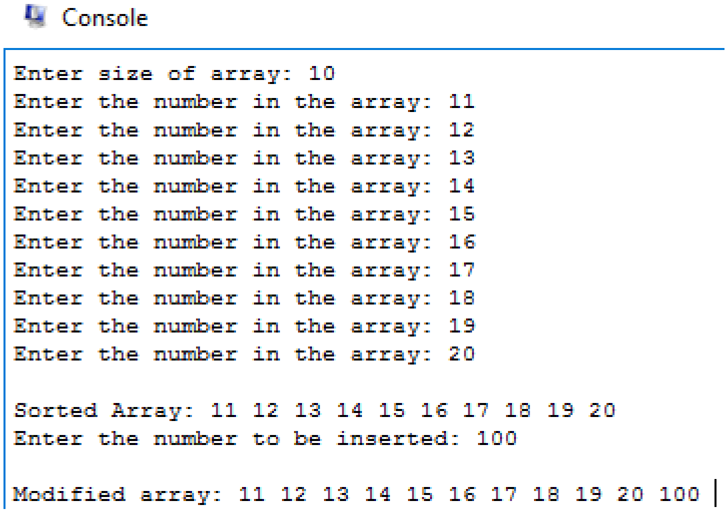


```
outputArray:
    li $v0, 4                                       # loading syscall service for print_string
    la $a0, arrayInsert                             # loading syscall argument for print_string
    syscall                                         # making syscall for print_string
    li $s1, -1                                      # loading offset counter for array
    la $s0, Array1                                  # return to array base address
printSorted:
    lb $a0, 0($s0)                                  # load byte from sorted array
    li $v0, 1                                       # loading syscall service for print_integer
    syscall                                         # making syscall for print_integer
    li $v0, 4                                       # loading syscall service for print_string
    la $a0, space                                   # loading syscall argument for print_string
    syscall                                         # making syscall for print_integer
    addi $s0, $s0, 4                                # incrementing the array address by word factor
    (4)
    addi $s1, $s1, 1                                # decrementing array size by 1
    bgt $s1, $s4, exit                              # if $s1 == size, branch to exit
    j printSorted                                   # jump to printSorted

exit:
    li $v0, 10                                      # loading syscall service for end_program
    syscall                                         # making syscall for end_program
```

Outputs for Assignment 2



```
Enter size of array: 10
Enter the number in the array: 1
Enter the number in the array: 2
Enter the number in the array: 3
Enter the number in the array: 4
Enter the number in the array: 5
Enter the number in the array: 6
Enter the number in the array: 7
Enter the number in the array: 8
Enter the number in the array: 9
Enter the number in the array: 10

Sorted Array: 1 2 3 4 5 6 7 8 9 10
Enter the number to be inserted: 9

Modified array: 1 2 3 4 5 6 7 8 9 9 10
```



```
Enter size of array: 10
Enter the number in the array: 11
Enter the number in the array: 12
Enter the number in the array: 13
Enter the number in the array: 14
Enter the number in the array: 15
Enter the number in the array: 16
Enter the number in the array: 17
Enter the number in the array: 18
Enter the number in the array: 19
Enter the number in the array: 20

Sorted Array: 11 12 13 14 15 16 17 18 19 20
Enter the number to be inserted: 100

Modified array: 11 12 13 14 15 16 17 18 19 20 100
```

Explanation of Assignment 3 (Part A)

i	2^-i	bit value	decimal value	
1	0.5	1	0.5	=1*2^-1
2	0.25	0	0	=0*2^-2
3	0.125	1	0.125	=1*2^-3
4	0.0625	1	0.0625	=1*2^-4
5	0.03125	1	0.03125	=1*2^-5
6	0.015625	1	0.015625	=1*2^-6
7	0.007813	0	0	=0*2^-7
8	0.003906	0	0	=0*2^-8
9	0.001953	0	0	=1*2^-9
10	0.000977	1	0.000976563	=1*2^-10
11	0.000488	1	0.000488281	=1*2^-11
12	0.000244	1	0.000244141	=1*2^-12
13	0.000122	0	0	=0*2^-13
14	6.1E-05	0	0	=0*2^-14
15	3.05E-05	0	0	=0*2^-15
16	1.53E-05	1	1.52588E-05	=1*2^-16
17	7.63E-06	0	0	=0*2^-17
18	3.81E-06	0	0	=0*2^-18
19	1.91E-06	1	1.90735E-06	=1*2^-19
20	9.54E-07	1	9.53674E-07	=1*2^-20
21	4.77E-07	0	0	=0*2^-21
22	2.38E-07	1	2.38419E-07	=1*2^-22
23	1.19E-07	1	1.19209E-07	=1*2^-23
		APPROXIMATE VALUE =	0.736102462	

Explanation of Assignment 3 (Part B)

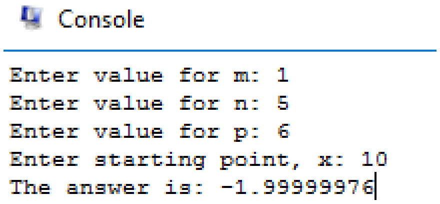
sign	exponent (8 bits)	fraction (23 bits)
1	100 0000 0	110 0110 0110 0110 0110 0110

Code Used for Assignment 4

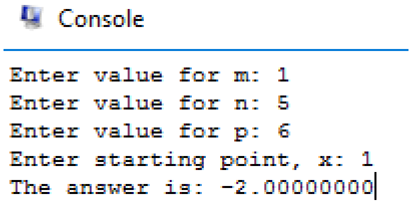
```
# Jeffrey Huang
# RUID: 159004687
# NETID: jh1127
# Assignment 4

# Registers:    $f0  -> user input value
#               $f1  -> value for m
#               $f2  -> value for n
#               $f3  -> value for p
#               $f4  -> value for starting point = x
#               $f5  -> value for x_i
#               $f6  -> value of cons. = 0.00001
#               $f7  -> x ^ 2 -> m * x ^ 2 -> m * x ^ 2 + n * x + p -> value for x_(i+1)
#               $f8  -> n * x -> 2.0 -> x_(i+1) - x_i
#               $f9  -> m * x
#               $f10 ->
#               $f11 ->
#               $f12 -> output argument for syscall

.data
    inputM:     .asciiz     "Enter value for m: "
    inputN:     .asciiz     "Enter value for n: "
    inputP:     .asciiz     "Enter value for p: "
    inputX:     .asciiz     "Enter starting point, x: "
    output:     .asciiz     "The answer is: "

.text

main:
## Getting the value of m
    li.s $f6, 0.00001               # loading 0.00001 as constant (cons.)
    li $v0, 4                       # loading syscall service for print_string
    la $a0, inputM                  # loading syscall argument for print_string
    syscall                         # making syscall to print_string
    li $v0, 6                       # loading syscall service for read_float
    syscall                         # making syscall to read_float
    mov.s $f1, $f0                  # copying user input to $f1
## Getting the value of n
    li $v0, 4                       # loading syscall service for print_string
    la $a0, inputN                  # loading syscall argument for print_string
    syscall                         # making syscall to print_string
    li $v0, 6                       # loading syscall service for read_float
    syscall                         # making syscall to read_float
    mov.s $f2, $f0                  # copying user input to $f2
## Getting the value of p
    li $v0, 4                       # loading syscall service for print_string
    la $a0, inputP                  # loading syscall argument for print_string
    syscall                         # making syscall to print_string
    li $v0, 6                       # loading syscall service for read_float
    syscall                         # making syscall to read_float
    mov.s $f3, $f0                  # copying user input to $f3
```

```
## Getting the value of x
    li $v0, 4                       # loading syscall service for print_string
    la $a0, inputX                  # loading syscall argument for print_string
    syscall                         # making syscall to print_string
    li $v0, 6                       # loading syscall service for read_float
    syscall                         # making syscall to read_float
    mov.s $f4, $f0                  # copying user input to $f3
## Setting the value of x_i
    mov.s $f5, $f4                  # copying the value of $f4 to $f5

loop:
## Getting value of f(x)
    mul.s $f7, $f4, $f4             # x * x = x ^ 2
    mul.s $f7, $f6, $f1             # m * x * x = m * x ^ 2
    mul.s $f8, $f2, $f4             # n * x
    add.s $f9, $f7, $f8             # m * x ^ 2 + n * x
    add.s $f9, $f9, $f3             # m * x ^ 2 + n * x + p = f(x)
## Getting value of f'(x)
    li.s $f8, 2.0                   # $f8 -> 2.0
    mul.s $f10, $f1, $f4            # m * x
    mul.s $f8, $f8, $f10            # 2 * m * x
    add.s $f8, $f8, $f2             # 2 * m * x + n = f'(x)
## Getting value of x_(i + 1)
    div.s $f8, $f9, $f8             # f(x) / f'(x)
    sub.s $f7, $f5, $f8             # x_(i+1) = x_i - f(x) / f'(x)
    sub.s $f8, $f7, $f5             # x_(i+1) - x_i
    abs.s $f8, $f8                  # | x_(i+1) - x_i |
    c.lt.s $f8, $f6                 # if | x_(i+1) - x_i | < 0.00001
    bc1t printOutput                # then printOutput
    mov.s $f5, $f7                  # else x_i = x_(i+1)
    j loop                          # jump to loop

printOutput:
    li $v0, 4                       # loading syscall service for print_string
    la $a0, output                  # loading syscall argument for print_string
    syscall                         # making syscall to print_string
    li $v0, 2                       # loading syscall service for print_float
    mov.s $f12, $f5                 # loading syscall argument for print_float
    syscall                         # making syscall to print_float
    j exit                          # jump to exit
exit:
    li $v0, 10                      # loading syscall service for end_program
    syscall                         # making syscall to end_program
```

Outputs for Assignment 5

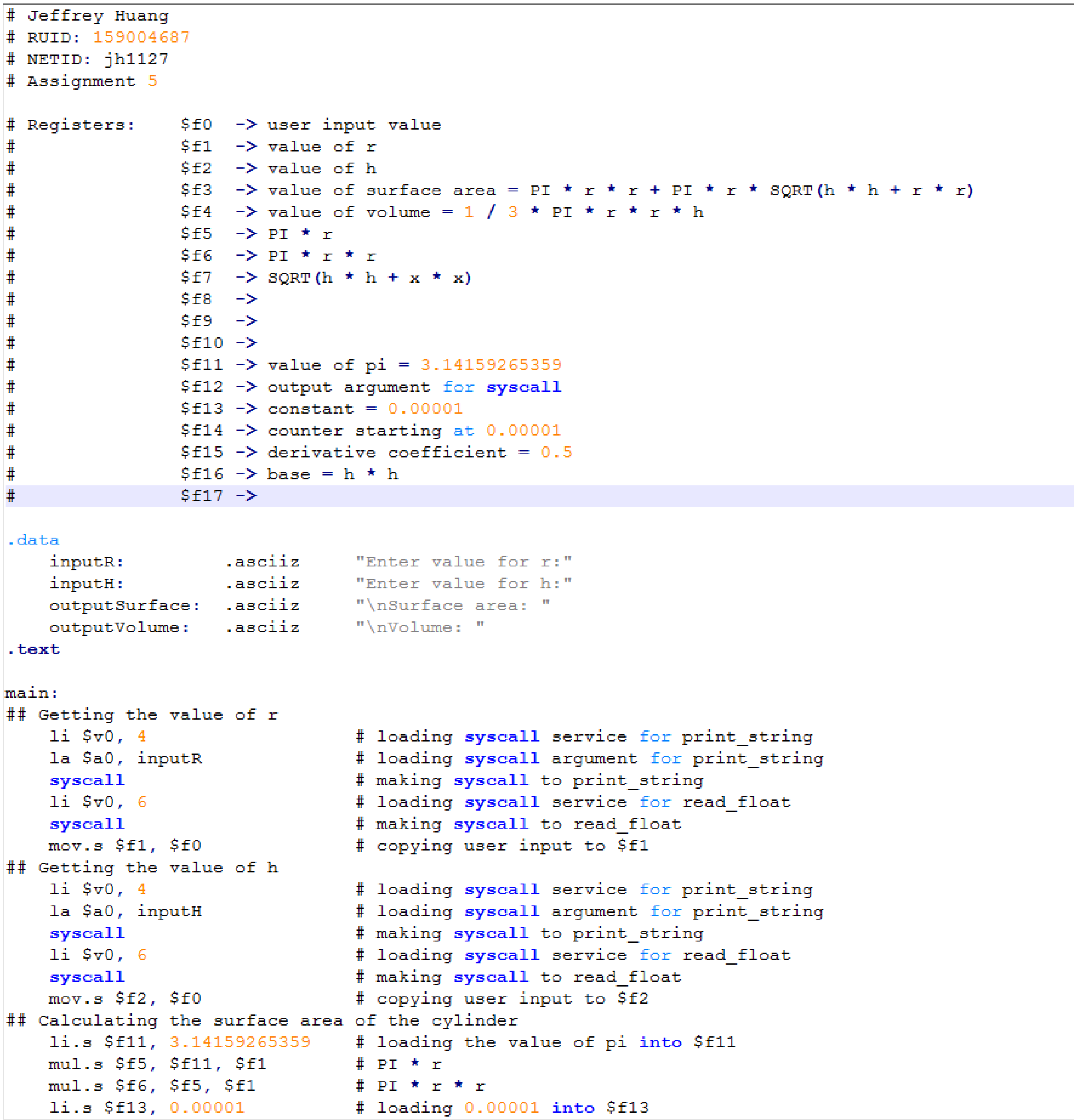


```
Enter value for m: 1
Enter value for n: 5
Enter value for p: 6
Enter starting point, x: 10
The answer is: -1.99999976
```

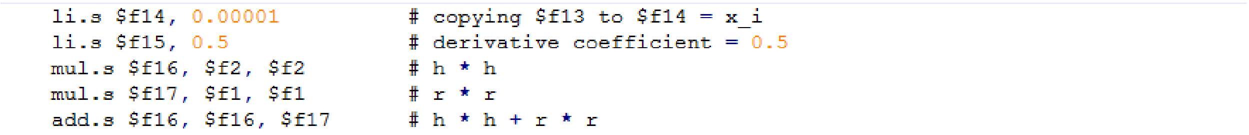


```
Enter value for m: 1
Enter value for n: 5
Enter value for p: 6
Enter starting point, x: 1
The answer is: -2.00000000
```

Code Used for Assignment 5



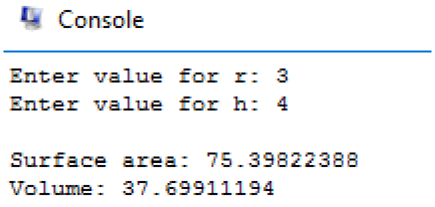
```
# Jeffrey Huang
# RUID: 159004687
# NETID: jh1127
# Assignment 5

# Registers:    $f0  -> user input value
#               $f1  -> value of r
#               $f2  -> value of h
#               $f3  -> value of surface area = PI * r * r + PI * r * SQRT(h * h + r * r)
#               $f4  -> value of volume = 1 / 3 * PI * r * r * h
#               $f5  -> PI * r
#               $f6  -> PI * r * r
#               $f7  -> SQRT(h * h + x * x)
#               $f8  ->
#               $f9  ->
#               $f10 ->
#               $f11 -> value of pi = 3.14159265359
#               $f12 -> output argument for syscall
#               $f13 -> constant = 0.00001
#               $f14 -> counter starting at 0.00001
#               $f15 -> derivative coefficient = 0.5
#               $f16 -> base = h * h
#               $f17 ->

.data
    inputR:         .asciiz     "Enter value for r:"
    inputH:         .asciiz     "Enter value for h:"
    outputSurface:  .asciiz     "\nSurface area: "
    outputVolume:   .asciiz     "\nVolume: "
.text

main:
## Getting the value of r
    li $v0, 4                   # loading syscall service for print_string
    la $a0, inputR              # loading syscall argument for print_string
    syscall                     # making syscall to print_string
    li $v0, 6                   # loading syscall service for read_float
    syscall                     # making syscall to read_float
    mov.s $f1, $f0              # copying user input to $f1
## Getting the value of h
    li $v0, 4                   # loading syscall service for print_string
    la $a0, inputH              # loading syscall argument for print_string
    syscall                     # making syscall to print_string
    li $v0, 6                   # loading syscall service for read_float
    syscall                     # making syscall to read_float
    mov.s $f2, $f0              # copying user input to $f2
## Calculating the surface area of the cylinder
    li.s $f11, 3.14159265359    # loading the value of pi into $f11
    mul.s $f5, $f11, $f1        # PI * r
    mul.s $f6, $f5, $f1         # PI * r * r
    li.s $f13, 0.00001          # loading 0.00001 into $f13
```



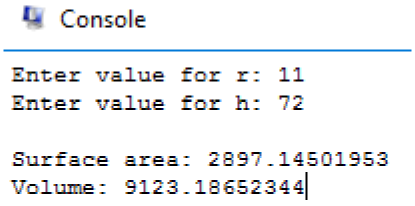
```
    li.s $f14, 0.00001          # copying $f13 to $f14 = x_i
    li.s $f15, 0.5              # derivative coefficient = 0.5
    mul.s $f16, $f2, $f2        # h * h
    mul.s $f17, $f1, $f1        # r * r
    add.s $f16, $f16, $f17      # h * h + r * r
```

```
squareRoot:
    div.s $f7, $f16, $f14       # (h * h + r * r) / x_i
    add.s $f7, $f7, $f14        # (h * h + r * r) / x_i + x_i
    mul.s $f7, $f15, $f7        # 0.5 * ((h * h + r * r) / x_i + x_i)
    sub.s $f11, $f7, $f14       # 0.5 * ((h * h + r * r) / x_i + x_i) - x_i
    abs.s $f11, $f11            # | 0.5 * ((h * h + r * r) / x_i + x_i) - x_i |
    c.lt.s $f11, $f13           # if | 0.5 * ((h * h + r * r) / x_i + x_i) - x_i | < 0.00001
    bc1t surfaceArea            # then branch to surfaceArea
    mov.s $f14, $f7             # else x_i = x_(i+1)
    j squareRoot                # jump to squareRoot
surfaceArea:
    mul.s $f3, $f7, $f5         # PI * r * SQRT(h * h + x * x)
    add.s $f3, $f6, $f3         # PI * r * r + PI * r * SQRT(h * h + x * x)
## Printing out the surface area of the cylinder
    li $v0, 4                   # loading syscall service for print_string
    la $a0, outputSurface       # loading syscall argument for print_string
    syscall                     # making syscall to print_string
    li $v0, 2                   # loading syscall service for print_float
    mov.s $f12, $f3             # loading syscall argument for print_float
    syscall                     # making syscall to print_float
## Calculating the volume of the cylinder
    li.s $f5, 1.0               # loading numerator into $f5
    li.s $f7, 3.0               # loading denominator into $f7
    div.s $f5, $f5, $f7         # constant of function (1.0 / 3.0)
    mul.s $f4, $f5, $f6         # (1.0 / 3.0) * PI * r * r
    mul.s $f4, $f4, $f2         # (1.0 / 3.0) * PI * r * r * h
## Printing the volume of the cylinder
    li $v0, 4                   # loading syscall service for print_string
    la $a0, outputVolume        # loading syscall argument for print_string
    syscall                     # making syscall to print_string
    li $v0, 2                   # loading syscall service for print_float
    mov.s $f12, $f4             # loading syscall argument for print_float
    syscall                     # making syscall to print_float
    li $v0, 10                  # loading syscall service for end_program
    syscall                     # making syscall to end_program
```

Outputs for Assignment 5



```
Enter value for r: 3
Enter value for h: 4

Surface area: 75.39822388
Volume: 37.69911194
```



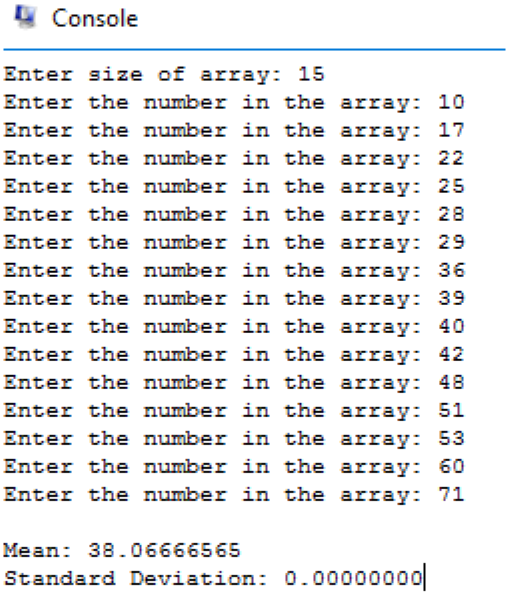
```
Enter value for r: 11
Enter value for h: 72

Surface area: 2897.14501953
Volume: 9123.18652344
```

Code Used for Assignment 6

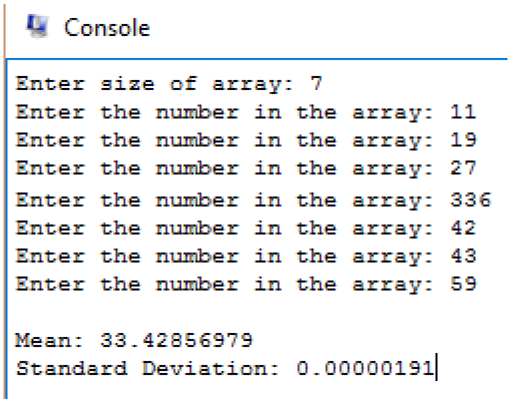
```
# Jeffrey Huang
# RUID: 159004687
# NETID: jh1127
# Assignment 2

# Registers:    $t0 -> address of array
#               $t1 -> user inputted value of array size
#               $t2 -> standard counter
#               $t3 ->
#               $t4 ->
#               $f0 -> user input values
#               $f1 -> sum of inputted values -> mean
#               $f2 -> float value of size -> temporary place holder -> temp
#               $f3 -> summation of (temp - mean) ^ 2 / (size - 1)
#               $f4 -> value of standard deviation

.data
    array:              .space      400
    arraySize:          .asciiz     "Enter size of array: "
    inputArr:           .asciiz     "Enter the number in the array: "
    outputMean:         .asciiz     "\nMean: "
    outputDeviation:    .asciiz     "\nStandard Deviation: "
.text

main:
    la $t0, array               # loading array into $t0
    li $v0, 4                   # loading syscall service for print_string
    la $a0, arraySize           # loading syscall argument for print_string
    syscall                     # making syscall to print_string
    li $v0, 5                   # loading syscall service for read_int
    syscall                     # making syscall service for read_int
    move $t1, $v0               # copying user input to arraySize
    li.s $f1, 0.0               # sum = 0;

input:
    li $v0, 4                   # loading syscall service for print_string
    la $a0, inputArr            # loading syscall argument for print_string
    syscall                     # making syscall to print_string
    li $v0, 6                   # loading syscall service for read_float
    syscall                     # making syscall to read_float
    s.s $f0, 0($t0)             # copying user input to array[i]
    add.s $f1, $f1, $f0         # sum += input
    addi $t0, $t0, 4            # address += 4
    addi $t2, $t2, 1            # i++
    beq $t2, $t1, endInput      # if i == size, branch to endInput
    j input                     # jump to input

endInput:
    sll $t2, $t1, 2             # returning the counter to the original value
    sub $t0, $t0, $t2           # returning the address of array to base value
    mtc1 $t1, $f2               # convert (int) size to (float) size
    cvt.s.w $f2, $f2            # convert float to int to register $f2
    div.s $f1, $f1, $f2         # mean = sum / size
    li $v0, 4                   # loading syscall service for print_string
    la $a0, outputMean          # loading syscall argument for print_string
    syscall                     # making syscall to print_string
    li $v0, 2                   # loading syscall service for print_float
    mov.s $f12, $f1             # loading syscall argument for print_float
    syscall                     # making syscall to print_float
    li $t2, 0                   # counter = 0
```

```
StdDeviation:
    l.s $f2, 0($t0)             # temp = array[i]
    sub.s $f3, $f2, $f1         # temp - mean
    mul.s $f3, $f3, $f3         # (temp - mean) * (temp - mean) = (temp - mean) ^ 2
    addi $t0, $t0, 4            # address += 4
    addi $t2, $t2, 1            # i++
    bge $t2, $t1, calculate     # if $t2 >= $t1, branch to calculate
    j StdDeviation              # jump to StdDeviation

calculate:
    addi $t2, $t1, -1           # size += -1
    mtc1 $t2, $f2               # convert (int) size-1 to (float) size-1
    cvt.s.w $f2, $f2            # convert float to int to register $f2
    div.s $f3, $f3, $f2         # (temp - mean) ^ 2 / (size - 1)
    li.s $f13, 0.00001          # loading 0.00001 into $f13
    li.s $f14, 0.00001          # copying $f13 to $f14 = x_i
    li.s $f15, 0.5              # derivative coefficient = 0.5

squareRoot:
    div.s $f7, $f3, $f14        # ((temp - mean) ^ 2 / (size - 1)) / x_i
    add.s $f7, $f7, $f14        # ((temp - mean) ^ 2 / (size - 1)) / x_i + x_i
    mul.s $f7, $f15, $f7        # 0.5 * (((temp - mean) ^ 2 / (size - 1)) / x_i + x_i)
    sub.s $f4, $f7, $f14        # 0.5 * (((temp - mean) ^ 2 / (size - 1)) / x_i + x_i) - x_i
    abs.s $f4, $f4              # | 0.5 * (((temp - mean) ^ 2 / (size - 1)) / x_i + x_i) - x_i |
    c.lt.s $f4, $f13            # if | 0.5 * (((temp - mean) ^ 2 / (size - 1)) / x_i + x_i) -
    x_i | < 0.00001
    bc1t printDeviation         # then branch to surfaceArea
    mov.s $f14, $f7             # else x_i = x_(i+1)
    j squareRoot                # jump to squareRoot

printDeviation:
    li $v0, 4                   # loading syscall service for print_string
    la $a0, outputDeviation     # loading syscall argument for print_string
    syscall                     # making syscall to print_string
    li $v0, 2                   # loading syscall service for print_float
    mov.s $f12, $f4             # loading syscall argument for print_float
    syscall                     # making syscall to print_float
    li $v0, 10                  # loading syscall argument for end_program
    syscall                     # making syscall to end_program
```

Outputs for Assignment 6



```
Enter size of array: 15
Enter the number in the array: 10
Enter the number in the array: 17
Enter the number in the array: 22
Enter the number in the array: 25
Enter the number in the array: 28
Enter the number in the array: 29
Enter the number in the array: 36
Enter the number in the array: 39
Enter the number in the array: 40
Enter the number in the array: 42
Enter the number in the array: 48
Enter the number in the array: 51
Enter the number in the array: 53
Enter the number in the array: 60
Enter the number in the array: 71

Mean: 38.06666565
Standard Deviation: 0.00000000
```



```
Enter size of array: 7
Enter the number in the array: 11
Enter the number in the array: 19
Enter the number in the array: 27
Enter the number in the array: 336
Enter the number in the array: 42
Enter the number in the array: 43
Enter the number in the array: 59

Mean: 33.42856979
Standard Deviation: 0.00000191
```