



Course Name: Computer Architecture Lab

Course Number and Section: 14:332:333:03

Experiment: 5

Lab Instructor: Ke Xia

Date Performed: 12/1/2021

Date Submitted: 12/1/2021

Submitted by: Alec Bakholdin 185002378

Course Name: Computer Architecture Lab
Course Number and Section: 14:332:333:03

! Important: Please include this page in your report if the submission is a paper submission. For electronic submission (email or Sakai) please omit this page.

-----For Lab Instructor Use ONLY-----

GRADE: _____

COMMENTS:

Electrical and Computer Engineering Department
School of Engineering
Rutgers University, Piscataway, NJ 08854
ECE Lab Report Structure

1. Purpose / Introduction / Overview – describe the problem and provide background information
2. Approach / Method – the approach took, how problems were solved
3. Results – present your data and analysis, experimental results, etc.
4. Conclusion / Summary – what was done and how it was done

1.

```
j main

recursive:
    # base case
    addi t0, zero, 2
    bge a0, t0, recursive_start
    jr ra
recursive_start:

    # store variables to stack
    addi sp, sp, -16
    sw s0, 0(sp)
    sw s1, 4(sp)
    sw ra, 8(sp)
    mv s0, a0

    # f(x - 1)
    addi a0, s0, -1
    jal recursive
    mv s1, a0

    # f(x - 2)
    addi a0, s0, -2
    jal recursive
    add a0, a0, s1

    # restore variables from stack
    lw s0, 0(sp)
    lw s1, 4(sp)
    lw ra, 8(sp)
    addi sp, sp, 16
    jr ra

iterative:

    mv s0, a0
    addi s1, zero, 2 # i = 2, to skip the <= branch
    addi s2, zero, 1 # first
    addi s3, zero, 1 # second
    addi s4, zero, 1 # next
loop:
    bge s1, s0, iterative_return
    addi t0, zero, 1
```

```

        add s4, s2, s3 # next = first + second
        mv s2, s3
        mv s3, s4

        addi s1, s1, 1 # i++
        j loop

iterative_return:
    mv a0, s4
    jr ra

print_integer:
    mv t0, a1

    mv a1, a0
    addi a0, zero, 1
    ecall
    addi a0, zero, 11
    addi a1, zero, '\n'
    ecall

    mv a0, a1
    mv a1, t0

    jr ra

main:
    addi a0, zero, 12
    jal recursive
    jal print_integer

    addi a0, zero, 12
    jal iterative
    jal print_integer

```

x = 12 →

144
144

x = 8 →

21
21

x = 14 →

377
377

2.

```
j main

output: # args: (int *array, int length)
    mv t0, a0          # t0 = array (int*)
    mv t1, a1          # t1 = length (int)
    addi t2, zero, 0    # int i = 0
output_loop:
    bge t2, t1, output_endloop # while i < length
    addi a0, zero, 1    # set ecall to print_integer
    lw a1, 0(t0)        # print array[i]
    ecall

    addi a0, zero, 11   # set ecall to print_char
    addi a1, zero, ' ' # print '\n'
    ecall

    addi t0, t0, 4      # array++
    addi t2, t2, 1      # i++
    j output_loop

output_endloop:
    addi a0, zero, 11
    addi a1, zero, '\n'
    ecall
    jr ra

main:
    la a1, array
    lw a2, array_len

    lw s0, odd_negatives
    addi s1, zero, 0
    lw s2, even_negatives
    addi s3, zero, 0
    lw s4, zeros
    addi s5, zero, 0

    addi t0, zero, 0 # j = 0
loop:
    slli t1, t0, 2 # t1 = j*4
    add t1, a1, t1 # t1 = array + 4j
    lw t1, 0(t1) # t1 = array[t1]
```

```

    blt t1, zero, ltzero
    beq t1, zero, zero
    j continue

zero:
    slli t2, s5, 2 # zero_counter * 4
    add t2, s4, t2 # t2 = zeros + zero_counter*4
    sw zero, 0(t2) # zeros[zero_counter] = 0
    addi s5, s5, 1 # zero_counter++
    j continue

ltzero:
    andi t2, t1, 1 # determine if even
    beq t2, zero, even
    # odds here
    slli t2, s1, 2 # odd_counter * 4
    add t2, s0, t2 # t2 = odds + odd_counter*4
    sw t1, 0(t2) # odds[odd_counter] = array[j]
    addi s1, s1, 1 # odd_counter++
    j continue
even:
    slli t2, s3, 2 # even_counter * 4
    add t2, s2, t2 # t2 = evens + even_counter*4
    sw t1, 0(t2) # evens[even_counter] = array[j]
    addi s3, s3, 1 # even_counter++

continue:
    addi t0, t0, 1 # j++
    bge t0, a2, endloop # while j < array_len
    j loop
endloop:

lw a0, odd_negatives
mv a1, s1
jal output

lw a0, even_negatives
mv a1, s3
jal output


lw a0, zeros
mv a1, s5
jal output

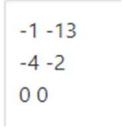
```

.data

```
odd_negatives: .word 0x40000004
even_negatives: .word 0x20000002
zeros: .word 0x50000000
array: .word -8 -6 -4 0 22 -1
array_len: .word 6
```

Output samples:

{-8 -6 -4 0 22 -1} → 

{-4 -2 0 12 13 0 -1 -13} → 

3.