**7.2**

.text

.globl main

main: addi $s1, $zero, 0 # initialize sum

addi $s0, $zero, 0 # initialize i

addi $t1, $zero, 10 # initialize temp 10

for: bge $s0, $t1, e\_for

add $s1, $s1, $s0

addi $s0, $s0, 1

j for

e\_for: li $v0, 10 # Setup Exit Syscall

syscall # Execute syscall

**7.3**

.data

prompt: .asciiz "Enter number for evaluation: "

.text

.globl main

main: la $a0, prompt # load address of prompt1 for syscall

li $v0, 4 # specify Print String service

syscall # print the prompt string

li $v0, 5 # specify Read Integer service

syscall # Read the number. After this instruction, the number read is in $v0.

move $a0, $v0 # move the read int to the argument

jal is\_odd # Call is\_odd function

move $s0, $v0

li $v0, 1 # Load print int for syscall

add $a0, $zero, $s0 # Load argument for print int

syscall # Execute print int

li $v0, 11 # Setup print char syscall

addi $a0, $zero, 10 # Load newline ascii char

syscall # execute syscall

j main

is\_odd: andi $v0, $a0, 1 # initialize sum

end: jr $ra

**7.4**

.text

.globl main

main: addi $s1, $zero, 0 # initialize sum

addi $s0, $zero, 0 # initialize i

addi $s5, $zero, 10 # initialize temp 10

for: bge $s0, $s5, e\_for # branch if i >= 10

move $a0, $s0 # move i into argument

jal is\_odd # Call is\_odd function

move $s2, $v0 # load response from is\_odd into $s1

addi $s3, $zero, 1 # initialize $s2 to 1

blt $s2, $s3, repeat # If remainder is less than 1, it's even, so go to repeat

add $s1, $s1, $s0 # sum += i

repeat: addi $s0, $s0, 1 # add 1 to i

j for

e\_for: li $v0, 10 # Setup Exit Syscall

syscall # Execute syscall

is\_odd: andi $v0, $a0, 1 # initialize sum

end: jr $ra

**7.5**

.text

.globl main

space\_count: addi $t0, $zero, 0 # Initialize count to t0 and 0

addi $t1, $zero, 0 # Initialize i to t1 and 0

li $t4, 0x20

while: add $t2, $t1, $a0 # set t2 to new address, since a char is only 1 byte

lbu $t3, 0($t2) # Set $t3 to new char

beq $t3, $zero, end # jump to end if string hits null terminated value

bne $t3, $t4, increment # If the char is not a space, increment i and go back to beginning of while otherwise:

addi $t0, $t0, 1 # increent counter

increment: addi $t1, $t1, 1

j while

end: move $v0, $t0 # set return value to count

jr $ra # Return count

**7.6**

.data

test1: .asciiz "four"

test12: .asciiz "four score and 7 years ago, our fathers brought forth on this continent a new nation"

test2: .asciiz "conceived in liberty and dedicated to the proposition that all men are created equal"

test3: .asciiz "three words here"

test4: .asciiz "five words in this string"

test5: .asciiz "there are ten words in this test string sample case "

.text

.globl main

main: la $a0, test5 # load test into argument

jal space\_count

add $s0, $v0, $zero

li $v0, 1 # Load print int for syscall

add $a0, $zero, $s0 # Load argument for print int

syscall # Execute print int

li $v0, 11 # Setup print char syscall

addi $a0, $zero, 10 # Load newline ascii char

syscall # execute syscall

li $v0, 10 # Setup Exit Syscall

syscall # Execute syscall

space\_count: addi $t0, $zero, 0 # Initialize count to t0 and 0

addi $t1, $zero, 0 # Initialize i to t1 and 0

li $t4, 0x20

while: add $t2, $t1, $a0 # set t2 to new address, since a char is only 1 byte

lbu $t3, 0($t2) # Set $t3 to new char

beq $t3, $zero, end # jump to end if string hits null terminated value

bne $t3, $t4, increment # If the char is not a space, increment i and go back to beginning of while otherwise:

addi $t0, $t0, 1 # increent counter

increment: addi $t1, $t1, 1

j while

end: move $v0, $t0 # set return value to count

jr $ra # Return count

**7.7**

void running\_avg\_four(int \*my\_array, int length){

for (int i = length - 1; i < 0; i--){

int total = my\_array[i] + my\_array[i-1] + my\_array[i-2] + my\_array[i-3];

int avg = (total >> 2);

my\_array[i] = avg;

}

}

**7.8**

.data

arr: .word 0, 0, 0, 4, 2, 8, 3, 1, 9, 10, 11, 13, 14, 15, 18, 27, 4, 9, 1, 19

.text

.globl main

main: la $s3, arr

# print original array

move $a0, $s3

addi $a1, $zero, 20

jal print\_arr

# alter array to running average

move $a0, $s3

addi $a1, $zero, 20

jal running\_avg\_4

# print running\_avg\_array

move $a0, $s3

addi $a1, $zero, 20

jal print\_arr

end\_main: li $v0, 10 # Setup Exit Syscall

syscall # Execute syscall

running\_avg\_4: move $t0, $a0 # set t0 to pointer to array

move $t1, $a1 # set t1 to length of array

add $t2, $zero, $t1 # set t2 to i, init with 3

addi $t2, $t2, -1 # subtract 1 from i

for: blt $t2, $zero, end # if i is equal to length of array, end

addi $t4, $zero, 0 # set t4 to total, init with 0

sll $t3, $t2, 2 # t3 = i \* 4

add $t5, $t3, $t0 # t5 = i + address

lw $t6, 0($t5) # t6 = value of t5

add $t4, $t4, $t6 # Add this to total

lw $t6, -4($t5) # t6 = value of t5 - 4

add $t4, $t4, $t6 # Add this to total

lw $t6, -8($t5) # t6 = value of t5 - 8

add $t4, $t4, $t6 # Add this to total

lw $t6, -12($t5) # t6 = value of t5 - 12

add $t4, $t4, $t6 # Add this to total

avg: srl $t7, $t4, 2 # set s1 (avg) to $t4 (total) / 4

add $t3, $t3, $t0

sw $t7, 0($t3) # set this address to the averaged value

addi $t2, $t2, -1 #decrement i

j for

end: jr $ra # Return count

# Print Function

print\_arr: addi $s0, $zero, 0 # initialise s0 to 0

addi $s1, $a0, 0

for\_main: bge $s0, $a1, end\_print # if s0 is greater than or equal to the length of the array, end

sll $s7, $s0, 2

add $s5, $s1, $s7

lw $s6, 0($s5)

li $v0, 1 # Setup print int syscall

add $a0, $zero, $s6 # Load ascii char into a0

syscall # execute syscall

li $v0, 11 # Setup print char syscall

li $a0, 0x20 # Load ascii char into a0

syscall # execute syscall

addi $s0, $s0, 1 #increment s0 by 1

j for\_main

end\_print: li $v0, 11 # Setup print char syscall

addi $a0, $zero, 10 # Load newline ascii char

syscall # execute syscall

jr $ra