Exercise C:

copycat:

# Variable Register

# c $t0

# \*src1 $t1

# \*src2 $t2

# Students: Replace this comment with appropriate code.

L1:

lbu $t1, ($a1) # $t1 = \*src1

beq $t1, $zero, endL1 # if(\*src == \0) goto endL1

sb $t1, ($a0) # \*dest = \*src1

addi $a0, $a0, 1 # dest++

addi $a1, $a1, 1 # src1++

j L1

endL1:

L2:

lbu $t2, ($a2) # $t2 = \*src2

lb $t0, ($a2) # c = \*src

sb $t0, ($a0) # \*dest = c

addi $a0, $a0, 1 # dest++

addi $a2, $a2, 1 # src2++

bne $t2, $zero, L2

jr $ra

# void lab4reverse(const char \*str)

#

.text

.globl lab4reverse

lab4reverse:

# Students: Replace this comment with appropriate code.

# Variables Register

# $t0 back

# $t2 front

# $t5 c

add $t0, $zero, $zero # back = 0

L3:

add $t1, $a0, $t0 # $t1 = str + back

lbu $t4, ($t1) # $t4 = \*str[back]

beq $t4, $zero, endL3 # if(str[back] == \0) goto endL3

addi $t0, $t0, 1 # back++

j L3

endL3:

addi $t0, $t0, -1 # back--

add $t2, $zero, $zero # front = 0

L4:

slt $t3, $t2, $t0 # t3 = front < back

beq $t3, $zero, endL4 # if( front >= back) goto endL4

add $t1, $a0, $t0 # $t1 = str + back

lbu $t5, ($t1) # c = str[back]

add $t6, $a0, $t2 # $t6 = str + front

lbu $t7, ($t6) # $t7 = str[front]

sb $t7, ($t1) # str[back] = str[front]

sb $t5, ($t6) # str[front] = c

addi $t0, $t0, -1

addi $t2, $t2, 1

j L4

endL4:

jr $ra

Exercise E:

# bin\_and\_hex.asm

# ENCM 369 Winter 2019 Lab 4 Exercise E Partial Solution

#

# BEGINNING of start-up & clean-up code. Do NOT edit this code.

.data

exit\_msg\_1:

.asciiz "\*\*\*About to exit. main returned "

exit\_msg\_2:

.asciiz ".\*\*\*\n"

main\_rv:

.word 0

.text

# adjust $sp, then call main

addi $t0, $zero, -32 # $t0 = 0xffffffe0

and $sp, $sp, $t0 # round $sp down to multiple of 32

jal main

nop

# when main is done, print its return value, then halt the program

sw $v0, main\_rv

la $a0, exit\_msg\_1

addi $v0, $zero, 4

syscall

nop

lw $a0, main\_rv

addi $v0, $zero, 1

syscall

nop

la $a0, exit\_msg\_2

addi $v0, $zero, 4

syscall

nop

addi $v0, $zero, 10

syscall

nop

# END of start-up & clean-up code.

# int main(void)

#

.text

.globl main

main:

addi $sp, $sp, -32

sw $ra, 0($sp)

li $a0, 0x76543210

jal test

li $a0, 0x89abcdef

jal test

li $a0, 0

jal test

li $a0, -1

jal test

add $v0, $zero, $zero # r.v. = 0

lw $ra, 0($sp)

addi $sp, $sp, 32

jr $ra

# void test(int test\_value)

#

# arg / var memory location

# test\_value 44($sp)

# char str[40] 40 bytes starting at 0($sp)

#

.data

STR1: .asciiz "\n\n"

.text

.globl test

test:

addi $sp, $sp, -64

sw $a0, 44($sp)

sw $ra, 40($sp)

addi $a0, $sp, 0 # $a0 = &str[0]

lw $a1, 44($sp) # $a1 = test\_value

jal write\_in\_hex

addi $a0, $sp, 0 # $a0 = &str[0]

addi $v0, $zero, 4 # $v0 = code to print a string

syscall

addi $a0, $zero, '\n' # $a0 = '\n'

addi $v0, $zero, 11 # $v0 = code to print a char

syscall

addi $a0, $sp, 0 # $a0 = &str[0]

lw $a1, 44($sp) # $a1 = test\_value

jal write\_in\_binary

addi $a0, $sp, 0 # $a0 = &str[0]

addi $v0, $zero, 4 # $v0 = code to print a string

syscall

la $a0, STR1 # $a0 = STR1

addi $v0, $zero, 4 # $v0 = code to print a string

syscall

lw $ra, 40($sp)

addi $sp, $sp, 64

jr $ra

# void write\_in\_hex(char \*str, unsigned int word)

#

# arg / var register

# str $a0

# word $a1

# digit\_list $t9

#

.data

hex\_digits:

.asciiz "0123456789abcdef"

.text

.globl write\_in\_hex

write\_in\_hex:

ori $t0, $zero, '0'

sb $t0, 0($a0) # str[0] = '0'

ori $t0, $zero, 'x'

sb $t0, 1($a0) # str[1] = 'x'

ori $t0, $zero, '\_'

sb $t0, 6($a0) # str[6] = '\_'

sb $zero, 11($a0) # str[11] = '\0'

la $t9, hex\_digits # digit\_list = hex\_digits

srl $t1, $a1, 28 # $t1 = word >> 28

andi $t2, $t1, 0xf # $t2 = $t1 & 0xf

add $t3, $t9, $t2 # $t3 = &digit\_list[$t2]

lb $t4, ($t3) # $t4 = digit\_list[$t2]

sb $t4, 2($a0) # str[2] = $t4

srl $t1, $a1, 24 # $t1 = word >> 24

andi $t2, $t1, 0xf # $t2 = $t1 & 0xf

add $t3, $t9, $t2 # $t3 = &digit\_list[$t2]

lb $t4, ($t3) # $t4 = digit\_list[$t2]

sb $t4, 3($a0) # str[3] = $t4

srl $t1, $a1, 20 # $t1 = word >> 20

andi $t2, $t1, 0xf # $t2 = $t1 & 0xf

add $t3, $t9, $t2 # $t3 = &digit\_list[$t2]

lb $t4, ($t3) # $t4 = digit\_list[$t2]

sb $t4, 4($a0) # str[4] = $t4

srl $t1, $a1, 16 # $t1 = word >> 16

andi $t2, $t1, 0xf # $t2 = $t1 & 0xf

add $t3, $t9, $t2 # $t3 = &digit\_list[$t2]

lb $t4, ($t3) # $t4 = digit\_list[$t2]

sb $t4, 5($a0) # str[5] = $t4

srl $t1, $a1, 12 # $t1 = word >> 12

andi $t2, $t1, 0xf # $t2 = $t1 & 0xf

add $t3, $t9, $t2 # $t3 = &digit\_list[$t2]

lb $t4, ($t3) # $t4 = digit\_list[$t2]

sb $t4, 7($a0) # str[7] = $t4

srl $t1, $a1, 8 # $t1 = word >> 8

andi $t2, $t1, 0xf # $t2 = $t1 & 0xf

add $t3, $t9, $t2 # $t3 = &digit\_list[$t2]

lb $t4, ($t3) # $t4 = digit\_list[$t2]

sb $t4, 8($a0) # str[8] = $t4

srl $t1, $a1, 4 # $t1 = word >> 4

andi $t2, $t1, 0xf # $t2 = $t1 & 0xf

add $t3, $t9, $t2 # $t3 = &digit\_list[$t2]

lb $t4, ($t3) # $t4 = digit\_list[$t2]

sb $t4, 9($a0) # str[9] = $t4

andi $t2, $a1, 0xf # $t2 = word & 0xf

add $t3, $t9, $t2 # $t3 = &digit\_list[$t2]

lb $t4, ($t3) # $t4 = digit\_list[$t2]

sb $t4, 10($a0) # str[10] = $t4

jr $ra

# write\_in\_binary(char \*str, unsigned int word)

#

# Students have to replace the code for this procedure

# with code that implements the given C code.

.text

.globl write\_in\_binary

write\_in\_binary:

# Time-saving hint: This is a leaf procedure!

# Leave str and word in $a0 and $a1, and

# use t-registers for local variables.

# Get rid of the next 3 lines before writing a solution.

# Variable Register

# digit0 $t0

# digit1 $t1

# under $t2

# index $t4

# mask $t5

# bn $t7

addi $t0, $zero, '0' # digit0 = '0'

addi $t1, $zero, '1' # digit1 = '1'

addi $t2, $zero, '\_' # under = '\_'

add $t7, $zero, $zero # bn = 0

addi $t3, $a0, 39 # $t3 = str + 39

sb $zero, ($t3) # str[39] = \0

addi $t4, $zero, 38 # index = 38

addi $t5, $zero, 1 # mask = 1

L1:

and $t3, $a1, $t5 # $t3 = word & mask

bne $t3, $zero, endIF1

add $t6, $a0, $t4 # $t6 = str + index

sb $t0, ($t6) # str[index] = digit0

j endElse1

endIF1:

add $t6, $a0, $t4 # $t6 = str + index

sb $t1, ($t6) # str[index] = digit1

endElse1:

addi $t4, $t4, -1 # index--

addi $t7, $t7, 1 # bn++

sll $t5, $t5, 1 # mask = mask << 1

addi $t6, $zero, 32

beq $t7, $t6, endL1

and $t3, $t7, 3 # $t3 = bn & 3

bne $t3, $zero, endIF2

add $t6, $a0, $t4 # $t6 = str + index

sb $t2, ($t6) # str[index] = under

add $t4, $t4, -1 # index--

endIF2:

j L1

endL1:

jr $ra