Course ​​​: Computer Organization – ENCM 369

Lab # ​​​: Lab 4

Student Name ​: Nimna Wijedasa

Lab Section ​​: B04

EX C

# string-funcs.asm

# ENCM 369 Winter 2023 Lab 4 Exercise C

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

.data

exit\_msg\_1:

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

exit\_msg\_2:

.asciz ".\*\*\*\n"

main\_rv:

.word 0

.text

# adjust sp, then call main

andi sp, sp, -32 # round sp down to multiple of 32

jal main

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

sw a0, main\_rv, t0

la a0, exit\_msg\_1

li a7, 4

ecall

lw a0, main\_rv

li a7, 1

ecall

la a0, exit\_msg\_2

li a7, 4

ecall

lw a0, main\_rv

addi a7, zero, 93 # call for program exit with exit status that is in a0

ecall

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

# void copycat(char \*dest, const char \*src1, const char \*src2)

#

.text

.globl copycat

copycat:

L1:

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

beq t1, zero, L2 #if (\*src1 == '\0') goto L2

lbu t0,(a1)

sb t0, (a0) # \*dest = \*src1

addi a1, a1, 1 # \*src++

addi a0, a0, 1 # \*dest++

j L1

L2:

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

sb t2,(a0)

addi a2, a2, 1 # \*src2++

addi a0, a0, 1 # \*dest++

bne t2,zero,L2 # if (c == '\0') goto

jr ra

# void lab4reverse(const char \*str)

#

.text

.globl lab4reverse

lab4reverse:

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

K1:

add t1,a0,t0 # str[back]

lbu t5,(t1)

beq t5,zero,K2 # if (str[back] == '\0') goto L2

addi t0,t0,1 # back++

j K1

K2:

li t1,0 # t1 = front = 0

addi t0,t0,-1 # back--

K3:

ble t0,t1,K4 # if (back <= front) goto L4

add t2,a0,t0 # t2 = c = str[back]

lbu t6,(t2)

add t3,a0,t1 # t3 = str[front]

lbu t4,(t3) # t4 = letter at str[front]

sb t4,(t2) # str[back] = str[front]

sb t6,(t3) # str[front] = c

addi t0,t0,-1 # back--

addi t1,t1,1 # front++

j K3

K4:

jr ra

# void print\_in\_quotes(const char \*str)

#

.text

.globl print\_in\_quotes

print\_in\_quotes:

add t0, a0, zero # copy str to t0

addi a0, zero, '"'

li a7, 11

ecall

mv a0, t0

li a7, 4

ecall

li a0, '"'

li a7, 11

ecall

li a0, '\n'

li a7, 11

ecall

jr ra

# Global arrays of char for use in testing copycat and lab4reverse.

.data

.align 5

# char array1[32] = { '\0', '\*', ..., '\*' };

array1: .byte 0, '\*', '\*', '\*', '\*', '\*', '\*', '\*'

.byte '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*'

.byte '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*'

.byte '\*', '\*', '\*', '\*', '\*', '\*', '\*', '\*'

# char array2[] = "X";

array2: .asciz "X"

# char array3[] = "YZ";

array3: .asciz "YZ"

# char array4[] = "123456";

array4: .asciz "123456"

# char array5[] = "789abcdef";

array5: .asciz "789abcdef"

# int main(void)

#

# string constants used by main

.data

sc0: .asciz ""

sc1: .asciz "good"

sc2: .asciz "bye"

sc3: .asciz "After 1st call to copycat, array1 has "

sc4: .asciz "After 2nd call to copycat, array1 has "

sc5: .asciz "After 3rd call to copycat, array1 has "

sc6: .asciz "After 4th call to copycat, array1 has "

sc7: .asciz "After use of lab4reverse, array2 has "

sc8: .asciz "After use of lab4reverse, array3 has "

sc9: .asciz "After use of lab4reverse, array4 has "

sc10: .asciz "After use of lab4reverse, array5 has "

.text

.globl main

main:

# Prologue only needs to save ra

addi sp, sp, -32

sw ra, 0(sp)

# Body

# Start tests of copycat.

la a0, array1 # a0 = array1

la a1, sc0 # a1 = sc0

la a2, sc0 # a2 = sc0

jal copycat

la a0, sc3

li a7, 4

ecall

la a0, array1 # a0 = array1

jal print\_in\_quotes

la a0, array1 # a0 = array1

la a1, sc1 # a1 = sc1

la a2, sc0 # a2 = sc0

jal copycat

la a0, sc4

li a7, 4

ecall

la a0, array1 # a0 = array1

jal print\_in\_quotes

la a0, array1 # a0 = array1

la a1, sc0 # a1 = sc0

la a2, sc2 # a2 = sc2

jal copycat

la a0, sc5

li a7, 4

ecall

la a0, array1 # a0 = array1

jal print\_in\_quotes

la a0, array1 # a0 = array1

la a1, sc1 # a1 = sc1

la a2, sc2 # a2 = sc2

jal copycat

la a0, sc6

li a7, 4

ecall

la a0, array1 # a0 = array1

jal print\_in\_quotes

# End tests of lab4cat; start tests of lab4reverse.

la a0, array2 # a0 = array2

jal lab4reverse

la a0, sc7

li a7, 4

ecall

la a0, array2 # a0 = array2

jal print\_in\_quotes

la a0, array3 # a0 = array3

jal lab4reverse

la a0, sc8

li a7, 4

ecall

la a0, array3 # a0 = array3

jal print\_in\_quotes

la a0, array4 # a0 = array4

jal lab4reverse

la a0, sc9

li a7, 4

ecall

la a0, array4 # a0 = array4

jal print\_in\_quotes

la a0, array5 # a0 = array5

jal lab4reverse

la a0, sc10

li a7, 4

ecall

la a0, array5 # a0 = array5

jal print\_in\_quotes

# End tests of lab4reverse.

mv a0, zero # r.v. from main = 0

# Epilogue

lw ra, 0(sp)

addi sp, sp, 32

jr ra

EX E

# bin\_and\_hex.asm

# ENCM 369 Winter 2023 Lab 4 Exercise E Partial Solution

#

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

.data

exit\_msg\_1:

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

exit\_msg\_2:

.asciz ".\*\*\*\n"

main\_rv:

.word 0

.text

# adjust sp, then call main

andi sp, sp, -32 # round sp down to multiple of 32

jal main

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

sw a0, main\_rv, t0

la a0, exit\_msg\_1

li a7, 4

ecall

lw a0, main\_rv

li a7, 1

ecall

la a0, exit\_msg\_2

li a7, 4

ecall

lw a0, main\_rv

addi a7, zero, 93 # call for program exit with exit status that is in a0

ecall

# 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

mv a0, 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: .asciz "\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]

li a7, 4 # a7 = code to print a string

ecall

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

li a7, 11 # a7 = code to print a char

ecall

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

lw a1, 44(sp) # a7 = test\_value

jal write\_in\_binary

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

li a7, 4 # a7 = code to print a string

ecall

la a0, STR1 # a0 = STR1

addi a7, zero, 4 # a7 = code to print a string

ecall

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:

.asciz "0123456789abcdef"

.text

.globl write\_in\_hex

write\_in\_hex:

li t0, '0'

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

li t0, 'x'

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

li t0, '\_'

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

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

la t6, hex\_digits # digit\_list = hex\_digits

srli t1, a1, 28 # t1 = word >> 28

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

add t3, t6, t2 # t3 = &digit\_list[t2]

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

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

srli t1, a1, 24 # t1 = word >> 24

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

add t3, t6, t2 # t3 = &digit\_list[t2]

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

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

srli t1, a1, 20 # t1 = word >> 20

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

add t3, t6, t2 # t3 = &digit\_list[t2]

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

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

srli t1, a1, 16 # t1 = word >> 16

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

add t3, t6, t2 # t3 = &digit\_list[t2]

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

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

srli t1, a1, 12 # t1 = word >> 12

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

add t3, t6, t2 # t3 = &digit\_list[t2]

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

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

srli t1, a1, 8 # t1 = word >> 8

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

add t3, t6, t2 # t3 = &digit\_list[t2]

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

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

srli t1, a1, 4 # t1 = word >> 4

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

add t3, t6, t2 # t3 = &digit\_list[t2]

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

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

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

add t3, t6, t2 # t3 = &digit\_list[t2]

lbu 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.

li t0,0 # t0 = 0 = bn

li t1,'0' # t1 = digit0 = '0'

li t2,'1' # t2 = digit1 = '1'

li t3,'\_' # t3 = \_ = under

addi t4,t0,38 # t4 = index = 38

sb zero,39(a0) # last element = '/0'

li t5,1 # mask = 1

L1:

and t6,a1,t5 # word & mask

beq t6,zero,L2 # if ((word & mask) == 0) goto L2

add t6,a0,t4 # str[index]

sb t2,(t6) # str[index] = digit1

j L3

L2:

add t6,a0,t4 # str[index] = digit1

sb t1,(t6)

L3:

addi t4,t4,-1 # index--

addi t0,t0,1 # bn++

slli t5,t5,1 # mask = mask << 1

li t6,32

beq t0,t6,L6 # if (bn == 32) goto L6

andi t6,t0,3

bne t6,zero,L4 # if ((bn & 3) != 0) goto L4

add t6,a0,t4 # str[index]

sb t3,(t6) # str[index] = under

addi t4,t4,-1 # index--

L4:

j L1

L6:

jr ra