CSE3666 — Lab 3

Mike Medved

February 7th, 2022

1 Prompt

In this lab, we write a program with RISC-V assembly language that performs the following tasks:

1. Read a string ‘str‘ from the console.
2. Remove the spaces (ASCII value 32) in ‘str‘ and save the result in string ‘res‘.
3. Print *res*.

Both strings are ASCII strings that end with a null character (NUL).

Skeleton code is provided in ‘lab3.s‘. Step 1 is already done. Study the code. Pseudocode for Step 2 is provided below. Step 3 is done by a system call.

Constraints in your code: Use only argument registers (like ‘a0‘ and ‘a1‘) and temporary registers (like ‘t0‘ and ‘t1‘). This will help you in the next lab. There is no need to use pseudo instruction ‘la‘. Addresses are already in a_0 and a_1.

Here are some example results:

```
a b c
abc

RISC-V: The Free and Open RISC Instruction Set Architecture
RISC-V:TheFreeandOpenRISCInstructionSetArchitecture
```

Pseudocode

```
i = 0
j = 0
do
    c = str[i]
    if c != 32
       res[j] = c
       j += 1
    i += 1
while c != 0
```

2 Deliverables

```
# CSE3666 Lab 3

.data
.align 2

# allocating space for both strings
str: .space  128
res: .space  128

.globl  main
.text

main:
        # read a string into str
        # use pseudoinstruction la to load address into register
        la      a0, str
        li      a1, 100
        li      a7, 8
        ecall

        # a0 is the address of str, a1 is the address of res
        la      a1, res

        # initialize i, j to 0
        addi    t0, x0, 0
        addi    t1, x0, 0

        # initialize t5 to 32 (SPACE)
        addi  t5, x0, 32
        j       loop

loop:
        # compute address of str[i]:
        # given base address of str[0] stored in a0,
        # we can add i + a0 (since chars are 1-byte)
        # to get the resultant address of str[i]
        add     t2, t0, a0

        # load char from target address
        # str[0] + i, into register t3.
        lb      t3, 0(t2)

        addi    t0, t0, 1     # increment loop counter i
        bne     t3, t5, write # if str[i] != 32 (SPACE), enter routine to save char
        beq     t3, x0, exit  # if str[i] == 0  (NUL), enter exit routine
        beq     x0, x0, loop  # continue the loop if this point is reached

write:
        # compute address of res[j]:
        # given base address of res[0] stored in a1,
        # we can add j + t1 (since chars are 1-byte)
        # to get resultant ddress of str[j]
        add     t4, t1, a1

        # load char from target address
        # res[0] + j, into register t3
```

```
        sb      t3, 0(t4)

        addi    t1, t1, 1     # increment loop counter j
        bne     t3, x0, loop  # if char != 0 (NUL), re-enter loop
        beq     t3, x0, exit  # if char == 0 (NUL), exit program

exit:
        # print resulting string stored in a1
        add     a0, x0, a1
        li      a7, 4
        ecall
```

3 Run Examples

```
a b c
abc

abcd e
abcdef

ab c d e f gh i j
abcdefghij

abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz

This string has escape sequences \n and special characters **
Thisstringhasescapesequences\nandspecialcharacters**

RISC-V: The Free and Open RISC Instruction Set Architecture
RISC-V:TheFreeandOpenRISCInstructionSetArchitecture

This string has lots of spaces and other funny characters!!!
Thisstringhaslotsofspacesandotherfunnycharacters!!!
```

4 Limitations

This algorithm works for all inputted strings up to the specified 100-character maximum limit, including those with non-alphanumeric special characters, ASCII control sequences, and escape characters.