Hayes Crowley

CS\_204\_Lab\_08

Prof. Papademas

**MODIFIED TEMPERATURE CONVERTER SOURCE CODE:**

# declare global so programmer can see actual addresses.

.globl welcome

.globl f\_to\_c\_prompt

.globl c\_to\_f\_prompt

.globl fahrenheitSumText

.globl celsiusSumText

.globl choice

# Data Area

.data

welcome:

.asciiz " This program converts Celsius to Fahrenheit \n\n"

choice:

.asciiz " Type 1 and hit enter to convert from f to c,\n else type 0 to convert from c to f: "

c\_to\_f\_prompt:

.asciiz " Enter an integer Celsius temperature: "

f\_to\_c\_prompt:

.asciiz " Enter an integer Fahrenheit temperature: "

fahrenheitSumText:

.asciiz " \n F = "

celsiusSumText:

.asciiz " \n C = "

coldText:

.asciiz "\nBrrrr!!!\n"

hotText:

.asciiz "\nIt's SWELTERING!\n"

zero\_as\_float:

.float 0.0

scalar\_1\_float:

.float 1.8

scalar\_2\_float:

.float 32.0

#Text Area (i.e. instructions)

.text

main:

# load float operands into coprocessor registers

lwc1 $f4, zero\_as\_float

lwc1 $f6, scalar\_1\_float

lwc1 $f8, scalar\_2\_float

# Display welcome

li $v0, 4

la $a0, welcome

syscall

# jump and link to c\_or\_f choice prompt, which saves choice of 1 or 0 in $s0 register

jal c\_or\_f

# conditional branches

beqz $s0, c\_to\_f

bnez $s0, f\_to\_c

c\_or\_f:

# prompt for choice

li $v0, 4

la $a0, choice

syscall

# get the input

li $v0, 5

syscall

# save choice of 1 or 0 to $s0 register

move $s0, $v0

# return to program counter

jr $ra

c\_to\_f:

# prompt for input

li $v0, 4

la $a0, c\_to\_f\_prompt

syscall

# read in and store float to $f0, floating point subprogram return register

li $v0, 6

syscall

# floating point arithmetic

# $f0 was our previous functions return value, $f6 is a scalar operand, etc.

# here, we are storing the results of these functions to $f12 cumulatively

# end result is ((C \* 1.8) + 32) stored to float register $f20, which is preserved across calls

mul.s $f12, $f0, $f6

add.s $f20, $f12, $f8

j display\_fahrenheit

f\_to\_c:

# display prompt

li $v0, 4

la $a0, f\_to\_c\_prompt

syscall

# take float input, goes into float return $f0

li $v0, 6

syscall

# formula is (F - 32) / 1.8

sub.s $f12, $f0, $f8

div.s $f20, $f12, $f6

j display\_celsius

display\_fahrenheit:

li $v0, 4

la $a0, fahrenheitSumText

syscall

# display float stored in $f20, fahrenheit value

li $v0, 2

add.s $f12, $f4, $f20

syscall

b exit

display\_celsius:

li $v0, 4

la $a0, celsiusSumText

syscall

# display float stored in $f20, celsius value

li $v0, 2

add.s $f12, $f4, $f20

syscall

b exit

exit:

li $v0, 10

syscall

**SCREENSHOTS OF OUTPUT:**

**Graphical user interface, text, application

Description automatically generated**

**LAB QUESTIONS**

1. The assembly statements with explanatory comments:  
   **# this loads the print\_string function from system services, equivalent to `li $v0, 10`**

ori $v0, $0, 4

**# load into argument register $a0 the string to print\_string**

la $a0, prompt

**# execute the preceding commands**

syscall

1. The assembly statements with explanatory comments:

# if (F < 60), printf ("Brrr!!\n");

**# set less than immediate, if $s0 is less than 60, set temp register $t0 to 1 else set to zero**

slti $t0, $s0, 60

**# branch to after label if $t0 is equal to zero.**

beq $t0, $0, after

**# bitwise OR to return system\_service code 4, print\_string**

ori $v0, $0, 4

**# load cold text into arg address 0**

la $a0, coldText

**# fire print\_string with cold text argument**

syscall

1. The assembly statements with explanatory comments:

**# declare global values in outside scope so they can be accessible by all functions in the .asm file**

.globl welcome

.globl prompt

.globl sumText

1. I don’t recall seeing any looping structures within the original temps.asm file.
2. C will be greater, work is shown below:

* F = -50
* C = (-50 – 32) / 1.8
* C = -82 / 1.8
* C = -45.555
* **C = -45.555 > F = -50**