605.204 - Computer Organization
Module 5: Assignment

Nick Hinke

October 2, 2022

Brief Introduction

This assignment involves the utilization of arithmetic, logical, and shift operators in armv7l. All of my resulting code can be found at `https://github.com/nhinke/computer-organization-repo/tree/master/assignments/module05` and can be cloned (along with pre-built binaries in a *bin/* folder) and viewed using the following commands:

```
git clone https://github.com/nhinke/computer-organization-repo.git
cd computer-organization-repo/assignments/module05/
```

The pre-built binaries can then be run using the following commands:

```
cd bin/
./CelsToFahr
./FahrToCels
./PrintNegInt
./FeetToInches
./InchesToFeet
./ShiftMultiply
./SwapVars
```

Note that each of the pre-built binaries will print out an example input-output sequence to the active terminal.

Problem 1

*1. Implement two programs, one to convert temperature from degrees Celsius to Fahrenheit, and one to do from Fahrenheit to Celsius. The formulas for this are: F = (C * 9/5) + 32 and C = (F - 32) * 5/9. (Hint: make sure your order of operations is correct, and multiplication is always done before division).*

Celsius to Fahrenheit

Program:

```
# Nick Hinke
# 10/02/2022
# 605.204 Computer Organization
# Module 5 Assignment - Problem 1
#
# Convert Celsius to Fahrenheit
#
.text
.global main
main:

  # push stack
  SUB sp, sp, #4
  STR lr, [sp, #0]

  # print prompt
  LDR r0, =prompt1
  BL printf

  # read input in degrees Celsius
  LDR r0, =format1
  LDR r1, =degCels
  BL scanf

  # multiply degrees Celsius by 9
  LDR r0, =degCels
  LDR r0, [r0, #0]
  MOV r1, #9
  MUL r0, r0, r1

  # divide by 5
  MOV r1, #5
  BL __aeabi_idiv

  # add 32
  MOV r1, #32
  ADD r0, r0, r1

  # print resulting degrees Fahrenheit
  MOV r2, r0
  LDR r1, =degCels
  LDR r1, [r1, #0]
  LDR r0, =output1
  BL printf

  # pop stack and return
  LDR lr, [sp, #0]
  ADD sp, sp, #4
  MOV pc, lr

.data
  prompt1: .asciz "Enter degrees Celsius: "
  output1: .asciz "Celsius: %d --> Fahrenheit: %d\n"
  format1: .asciz "%d"
  degCels: .word  0
# end main
```

Figure 1: Screenshot of program

Example:

```
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module05 $ ./bin/CelsToFahr
Enter degrees Celsius: 100
Celsius: 100 ⟶ Fahrenheit: 212
```

Figure 2: Example input-output sequence

Fahrenheit to Celsius

Program:

```
1  # Nick Hinke
2  # 10/02/2022
3  # 605.204 Computer Organization
4  # Module 5 Assignment - Problem 1
5  #
6  # Convert Fahrenheit to Celsius
7  #
8  .text
9  .global main
10 main:
11
12   # push stack
13   SUB sp, sp, #4
14   STR lr, [sp, #0]
15
16   # print prompt
17   LDR r0, =prompt1
18   BL printf
19
20   # read input in degrees Fahrenheit
21   LDR r0, =format1
22   LDR r1, =degFahr
23   BL scanf
24
25   # subtract 32 from degrees Fahrenehit
26   LDR r0, =degFahr
27   LDR r0, [r0, #0]
28   SUB r0, r0, #32
29
30   # multiply by 5
31   MOV r1, #5
32   MUL r0, r0, r1
33
```

```
34   # divide by 9
35   MOV r1, #9
36   BL __aeabi_idiv
37
38   # print resulting degrees Celsius
39   MOV r2, r0
40   LDR r1, =degFahr
41   LDR r1, [r1, #0]
42   LDR r0, =output1
43   BL printf
44
45   # pop stack and return
46   LDR lr, [sp, #0]
47   ADD sp, sp, #4
48   MOV pc, lr
49
50 .data
51   prompt1: .asciz "Enter degrees Fahrenheit: "
52   output1: .asciz "Fahrenheit: %d --> Celsius: %d\n"
53   format1: .asciz "%d"
54   degFahr: .word  0
55 # end main
```

Figure 3: Screenshot of program

Example:

```
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module05 $ ./bin/FahrToCels
Enter degrees Fahrenheit: 212
Fahrenheit: 212 ⟶ Celsius: 100
```

Figure 4: Example input-output sequence

Problem 2

2. Write a program that reads an integer number and writes out the negative value of the number. To do this, you must implement a 2's complement operation on the value, which is calculating the one's complement, and adding 1. To get a 1's complement, use the MVN operation.

Program:

```
1   # Nick Hinke
2   # 10/02/2022
3   # 605.204 Computer Organization
4   # Module 5 Assignment - Problem 2
5   #
6   # Read in integer, negate it, and print
7   #
8   .text
9   .global main
10  main:
11
12    # push stack
13    SUB sp, sp, #4
14    STR lr, [sp, #0]
15
16    # print input prompt
17    LDR r0, =prompt1
18    BL printf
19
20    # read input integer
21    LDR r0, =format1
22    LDR r1, =inputInt
23    BL scanf
24
25    # load integer into r4
26    LDR r4, =inputInt
27    LDR r4, [r4, #0]
28
29    # perform two's complement on integer
30    MVN r5, r4
31    ADD r5, r5, #1
32
```

```
33    # print results
34    MOV r1, r4
35    MOV r2, r5
36    LDR r0, =output1
37    BL printf
38
39    # pop stack and return
40    LDR lr, [sp, #0]
41    ADD sp, sp, #4
42    MOV pc, lr
43
44  .data
45    output1: .asciz "\nOriginal Integer: %d\nNegated Version: %d\n"
46    prompt1: .asciz "Enter an integer: "
47    format1: .asciz "%d"
48    inputInt: .word 0
49  # end main
```

Figure 5: Screenshot of program

Example:

```
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module05 $ ./bin/PrintNegInt
Enter an integer: 73

Original Integer: 73
Negated Version: -73
```

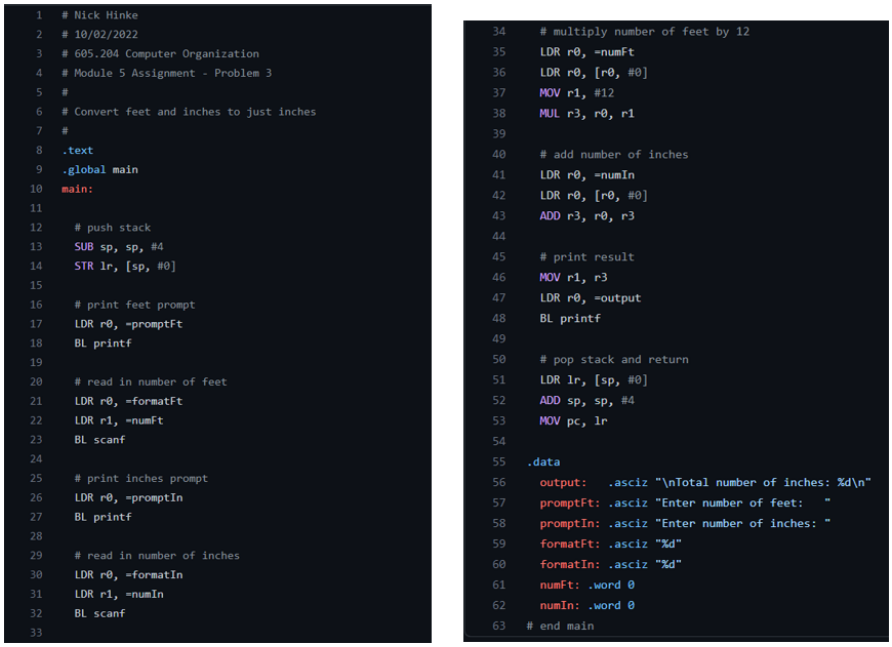
Figure 6: Example input-output sequence

Problem 3

3. Read two input numbers that represent feet and inches and convert this answer to be just inches. Then write a second program to do the reverse, taking one input in inches and convert it to feet and inches (this requires a remainder operation, which is not available on the Pi, so you'll have to write your own).

Feet and Inches to Inches

Program:



```
# Nick Hinke
# 10/02/2022
# 605.204 Computer Organization
# Module 5 Assignment - Problem 3
#
# Convert feet and inches to just inches
#
.text
.global main
main:

  # push stack
  SUB sp, sp, #4
  STR lr, [sp, #0]

  # print feet prompt
  LDR r0, =promptFt
  BL printf

  # read in number of feet
  LDR r0, =formatFt
  LDR r1, =numFt
  BL scanf

  # print inches prompt
  LDR r0, =promptIn
  BL printf

  # read in number of inches
  LDR r0, =formatIn
  LDR r1, =numIn
  BL scanf

  # multiply number of feet by 12
  LDR r0, =numFt
  LDR r0, [r0, #0]
  MOV r1, #12
  MUL r3, r0, r1

  # add number of inches
  LDR r0, =numIn
  LDR r0, [r0, #0]
  ADD r3, r0, r3

  # print result
  MOV r1, r3
  LDR r0, =output
  BL printf

  # pop stack and return
  LDR lr, [sp, #0]
  ADD sp, sp, #4
  MOV pc, lr

.data
  output:   .asciz "\nTotal number of inches: %d\n"
  promptFt: .asciz "Enter number of feet:   "
  promptIn: .asciz "Enter number of inches: "
  formatFt: .asciz "%d"
  formatIn: .asciz "%d"
  numFt: .word 0
  numIn: .word 0
# end main
```

Figure 7: Screenshot of program

Example:

```
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module05 $ ./bin/FeetToInches
Enter number of feet:   3
Enter number of inches: 9

Total number of inches: 45
```

Figure 8: Example input-output sequence

Inches to Feet and Inches

Program:

```
# Nick Hinke
# 10/02/2022
# 605.204 Computer Organization
# Module 5 Assignment - Problem 3
#
# Convert inches to feet and inches
#
.text
.global main
main:

  # push stack
  SUB sp, sp, #4
  STR lr, [sp, #0]

  # print total inches prompt
  LDR r0, =promptIn
  BL printf

  # read in number of inches
  LDR r0, =formatIn
  LDR r1, =numIn
  BL scanf

  # divide number of inches by 12 to get number of feet
  LDR r0, =numIn
  LDR r0, [r0, #0]
  MOV r1, #12
  BL __aeabi_idiv
  MOV r3, r0

  # multiply number of feet by 12
  MOV r0, r3
  MOV r1, #12
  MUL r4, r0, r1

  # subtract 12*feet from total inches to get remainder
  LDR r0, =numIn
  LDR r0, [r0, #0]
  SUB r5, r0, r4

  # print result
  MOV r1, r3
  MOV r2, r5
  LDR r0, =output
  BL printf

  # pop stack and return
  LDR lr, [sp, #0]
  ADD sp, sp, #4
  MOV pc, lr

.data
  output:   .asciz "Feet: %d Inches: %d\n"
  promptIn: .asciz "Enter total number of inches: "
  formatIn: .asciz "%d"
  numIn: .word 0
# end main
```

Figure 9: Screenshot of program

Example:

```
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module05 $ ./bin/InchesToFeet
Enter total number of inches: 45
Feet: 3 Inches: 9
```

Figure 10: Example input-output sequence

Problem 4

4. Read an input number and using left logical shifts and add instructions multiply that number by 10 and print out the result.

Program:

```
# Nick Hinke
# 10/02/2022
# 605.204 Computer Organization
# Module 5 Assignment - Problem 4
#
# Multiply input integer by 10 using left bit shift
#
.text
.global main
main:

  # push stack
  SUB sp, sp, #4
  STR lr, [sp, #0]

  # print prompt
  LDR r0, =prompt
  BL printf

  # read in input integer
  LDR r0, =format
  LDR r1, =inputInt
  BL scanf

  # store integer in r3
  LDR r3, =inputInt
  LDR r3, [r3, #0]

  # store 8*integer in r4
  LSL r4, r3, #3

  # store 2*integer in r5
  LSL r5, r3, #1

  # add results to get 10*integer in r6
  ADD r6, r4, r5

  # print results
  MOV r1, r3
  MOV r2, r6
  LDR r0, =output
  BL printf

  # pop stack and return
  LDR lr, [sp, #0]
  ADD sp, sp, #4
  MOV pc, lr

.data
  output: .asciz "%d * 10 = %d\n"
  prompt: .asciz "Enter an integer: "
  format: .asciz "%d"
  inputInt: .word 0
# end main
```

Figure 11: Screenshot of program

Example:

```
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module05 $ ./bin/ShiftMultiply
Enter an integer: -16
-16 * 10 = -160
```

Figure 12: Example input-output sequence

Problem 5 (Extra Credit)

5. In a normal swap a temporary variable is needed, eg "r2 = r0; r0 = r1; r1 = r2;" swaps r0 and r1. A swap can be implemented without the temporary r2 register using xor operations (the EOR instruction). Write a program to swap two registers using EOR instructions.

Program:

```
# Nick Hinke
# 10/02/2022
# 605.204 Computer Organization
# Module 5 Assignment - Problem 5 (EC)
#
# Swap two variables using exclusive or
#
.text
.global main
main:

  # push stack
  SUB sp, sp, #4
  STR lr, [sp, #0]

  # print first variable prompt (X)
  LDR r0, =promptX
  BL printf

  # read in first variable (X)
  LDR r0, =formatX
  LDR r1, =intX
  BL scanf

  # print second variable prompt (Y)
  LDR r0, =promptY
  BL printf

  # read in second variable (Y)
  LDR r0, =formatY
  LDR r1, =intY
  BL scanf

  # put variables in registers r3 and r4
  LDR r3, =intX
  LDR r4, =intY
  LDR r3, [r3, #0]
  LDR r4, [r4, #0]

  # swap registers r3 (X) and r4 (Y)
  EOR r3, r3, r4
  EOR r4, r3, r4
  EOR r3, r3, r4

  # print X results
  LDR r1, =intX
  LDR r1, [r1, #0]
  MOV r2, r3
  LDR r0, =outputX
  BL printf

  # print Y results
  LDR r1, =intY
  LDR r1, [r1, #0]
  MOV r2, r4
  LDR r0, =outputY
  BL printf

  # pop stack and return
  LDR lr, [sp, #0]
  ADD sp, sp, #4
  MOV pc, lr

.data
  outputX: .asciz "\nOriginal X: %d\tNew X: %d"
  outputY: .asciz "\nOriginal Y: %d\tNew Y: %d\n"
  promptX: .asciz "Enter an integer for X: "
  promptY: .asciz "Enter an integer for Y: "
  formatX: .asciz "%d"
  formatY: .asciz "%d"
  intX: .word 0
  intY: .word 0
# end main
```

Figure 13: Screenshot of program

Example:

```
rpi@rpi1:~/Documents/JHU/Computer-Organization/computer-organization-repo/assignments/module05 $ ./bin/SwapVars
Enter an integer for X: -72
Enter an integer for Y: 19

Original X: -72 New X: 19
Original Y: 19  New Y: -72
```

Figure 14: Example input-output sequence