

Mark Lucernas
CISC 211
Prof. Saied Moezzi
December 9, 2020

cisc-211-13@raspberrypi

Unit 8 Assignment

Problem 1

Overview

```
1 0 -----
2 0 Mark Lucernas
3 0 Unit 8 Assignment
4 0 -----
5 0 -----
6 0 Constants
7 0 -----
8 .equ EXIT, 1          @ Exit code for system call
9
10 0 Data Section
11 0 -----
12 .data
13 0 -----
14 0 Initialize Data
15 0 -----
16 .balign 4
17 string1: .asciz "Mark"
18 string2: .asciz "Lucernas"
19 message: .asciz "%s, %s, %s\n"
20
21 0 Uninitialized Data
22 0 -----
23 .bss
24 finalString: .skip 255 @ Reserve memory for concatenated result string
25
26 0 Code Section
27 0 -----
28 .text
29 -global main
30 -extern printf
31
32 main: push {ip, lr} @ push return address (lr) + dummy register (ip)
33
34 ldr r1, addrString1 @ r1 <- string1
35 ldr r2, addrString2 @ r2 <- string2
36 ldr r3, addrFinalString @ r3 <- finalString
37 bl concat @ Call concat()
38
39 ldr r3, =message @ Load message (output format)
40 bl printf
41
42 exit: pop {ip, lr} @ Pop the values from stack into ip and pc register
43 mov r0, #0 @ r0 <- 0 (Exit code)
44 mov r7, #EXIT @ Request to exit program
45 svc #0 @ Perform the system call
46
47 0 Functions
48 0 -----
49 0 Concatenate string from r1 and r2 and store it in r3
50 0 -----
```

```
1 /**
2  * Creates Concat class that has a concat function that concatenates two char
3  * array parameters and store it in another char array.
4  *
5  * @author Mark Lucernas
6  * Created on 12/07/2020.
7  */
8 public class LucernasAssignment8 {
9
10     public static void main(String[] args) {
11         char[] string1 = { 'M', 'a', 'r', 'k' };
12         char[] string2 = { 'L', 'u', 'c', 'e', 'r', 'n', 'a', 's' };
13         char[] stringConcat = new char[string1.length + string2.length];
14
15         concat(string1, string2, stringConcat);
16
17         for (int i = 0; i < string1.length; i++) {
18             System.out.print(string1[i]);
19         }
20         System.out.print(", ");
21
22         for (int i = 0; i < string2.length; i++) {
23             System.out.print(string2[i]);
24         }
25         System.out.print(", ");
26
27         for (int i = 0; i < stringConcat.length; i++) {
28             System.out.print(stringConcat[i]);
29         }
30     }
31
32     static void concat(char[] string1, char[] string2, char[] stringConcat) {
33         // Add all string1 values into stringConcat
34         for (int i = 0; i < string1.length; i++) {
35             stringConcat[i] = string1[i];
36         }
37         // Add all string2 values into stringConcat
38         for (int i = 0; i < string2.length; i++) {
39             stringConcat[i + string1.length] = string2[i];
40         }
41     }
42 }
```

```
cisc-211-13@raspberrypi:~/unit-8_assignment $ ./run_programs.sh
[ COMPILER ] Compiling 'Lucernas_assignment_8.a' and 'LucernasAssignment8.java'...
a5 -o Lucernas_assignment_8.o Lucernas_assignment_8.a
gcc -o Lucernas_assignment_8 Lucernas_assignment_8.o
javac LucernasAssignment8.java
[ RUN ] Running 'Lucernas_assignment_8'...
Mark, Lucernas, MarkLucernas
[ RUN ] Running 'LucernasAssignment8.class'...
Mark, Lucernas, MarkLucernas
[ CLEAN ] Cleaning up...
rm -f *.o *.class Lucernas_assignment_8
Done
cisc-211-13@raspberrypi:~/unit-8_assignment $
```

```
--Register group: general
r0 0x0 77 r1 0x21028 135208
r2 0x2102d 135213 r3 0x21043 135235
r4 0x1 1 r5 0x0 0
r6 0x10318 66328 r7 0x0 0
r8 0x0 0 r9 0x0 0
r10 0x76ffff00 1996484608 r11 0x0 0
r12 0x76ffff00 2130702720 sp 0x76ffff00 0x76ffff00
lr 0x1045 66044 pc 0x10474 66011
```

```
0x10474 <loop1> ldrb r0, [r3, #4]
0x10478 <loop1+4> strb r0, [r3, #4]
0x1047c <loop1+8> cmp r0, #0
0x10480 <loop1+12> addne r4, r4, #1
0x10484 <loop1+16> bne 0x10474 <loop1>
0x10488 <loop1+20> mov r5, #0
0x1048c <loop2> ldrb r0, [r2, #5]
0x10490 <loop2+4> strb r0, [r3, #4]
```

```
Native process 31751 Int: loop1 177 PC: 0x10474
0x00010478 in loop1 ()
0x0001047c in loop1 ()
0x00010480 in loop1 ()
0x00010484 in loop1 ()
0x00010474 in loop1 ()
(gdb)
```

High-level Language: Java Screenshot

```
1 /**
2  * Creates Concat class that has a concat function that concatenates two char
3  * array parameters and store it in another char array.
4  *
5  * @author Mark Lucernas
6  * Created on 12/07/2020.
7  */
8 public class LucernasAssignment8 {
9
10     public static void main(String[] args) {
11         char[] string1 = { 'M', 'a', 'r', 'k' };
12         char[] string2 = { 'L', 'u', 'c', 'e', 'r', 'n', 'a', 's' };
13         char[] stringConcat = new char[string1.length + string2.length];
14
15         concat(string1, string2, stringConcat);
16
17         for (int i = 0; i < string1.length; i++) {
18             System.out.print(string1[i]);
19         }
20         System.out.print(", ");
21
22         for (int i = 0; i < string2.length; i++) {
23             System.out.print(string2[i]);
24         }
25         System.out.print(", ");
26
27         for (int i = 0; i < stringConcat.length; i++) {
28             System.out.print(stringConcat[i]);
29         }
30     }
31
32     static void concat(char[] string1, char[] string2, char[] stringConcat) {
33         // Add all string1 values into stringConcat
34         for (int i = 0; i < string1.length; i++) {
35             stringConcat[i] = string1[i];
36         }
37         // Add all string2 values into stringConcat
38         for (int i = 0; i < string2.length; i++) {
39             stringConcat[i + string1.length] = string2[i];
40         }
41     }
42 }
```

High-level Language: Java Source Code

```
/**
 * Creates Concat class that has a concat function that concatenates two char
 * array parameters and store it in another char array.
 *
 * @author Mark Lucernas
 * Created on 12/07/2020.
```

```

*/
public class LucernasAssignment8 {

    public static void main(String[] args) {
        char[] string1 = { 'M', 'a', 'r', 'k' };
        char[] string2 = { 'L', 'u', 'c', 'e', 'r', 'n', 'a', 's' };
        char[] stringConcat = new char[string1.length + string2.length];

        concat(string1, string2, stringConcat);

        for (int i = 0; i < string1.length; i++) {
            System.out.print(string1[i]);
        }
        System.out.print(", ");

        for (int i = 0; i < string2.length; i++) {
            System.out.print(string2[i]);
        }
        System.out.print(", ");

        for (int i = 0; i < stringConcat.length; i++) {
            System.out.print(stringConcat[i]);
        }
    }

    static void concat(char[] string1, char[] string2, char[] stringConcat) {
        // Add all string1 values into stringConcat
        for (int i = 0; i < string1.length; i++) {
            stringConcat[i] = string1[i];
        }
        // Add all string2 values into stringConcat
        for (int i = 0; i < string2.length; i++) {
            stringConcat[i + string1.length] = string2[i];
        }
    }
}

```

Assembly Screenshot

```
1 @ -----
2 @ Mark Lucernas
3 @ Unit 8 Assignment
4 @ -----
5
6 @ Constants
7 @ -----
8 .equ EXIT, 1 @ Exit code for system call
9
10 @ Data Section
11 @ -----
12 .data
13
14 @ Initialize Data
15 @ -----
16 .balign 4
17 string1: .asciz "Mark"
18 string2: .asciz "Lucernas"
19 message: .asciz "%s, %s, %s\n"
20
21 @ Uninitialized Data
22 @ -----
23 .bss
24 finalString: .skip 255 @ Reserve memory for concated result string
25
26 @ Code Section
27 @ -----
28 .text
29 .global main
30 .extern printf
31
32 main: push {ip, lr} @ push return address (lr) + dummy register (ip)
33
34 ldr r1, addrString1 @ r1 <- string1
35 ldr r2, addrString2 @ r2 <- string2
36 ldr r3, addrFinalString @ r3 <- finalString
37 bl concat @ Call concat()
38
39 ldr r0, =message @ Load message (output format)
40 bl printf
41
42 exit: pop {ip, lr} @ Pop the values from stack into ip and pc register
43 mov r0, #0 @ r0 <- 0 (Exit code)
44 mov r7, #EXIT @ Request to exit program
45 svc 0 @ Perform the system call
46
47 @ Functions
48 @ -----
49
50 @ Concatenate string from r1 and r2 and store it in r3
51
52 concat: push {r1, r2, r3, r4, r5, lr} @ Save registers
53 mov r4, #0 @ r4 <- 0 (i = 0)
54 loop1: ldrb r0, [r1, r4] @ Load next byte (char) of string1 into r0
55 strb r0, [r3, r4] @ Store next byte (char) of string1 into finalString
56 cmp r0, #0 @ Compare if r0 is empty (no more to load from string1)
57 addne r4, #1 @ If not empty: r4 <- r4 + 1
58 bne loop1 @ Repeat loop if r0 not empty
59
60 mov r5, #0 @ r5 <- 0 (j = 0)
61
62 loop2: ldrb r0, [r2, r5] @ Load next byte (char) of string2 into r0
63 strb r0, [r3, r4] @ Store next byte (char) of string2 into finalString
64 cmp r0, #0 @ Compare if r0 is empty (no more to load from string1)
65 addne r4, #1 @ If not empty: r4 <- r4 + 1
66 addne r5, #1 @ If not empty: Go to next string1 byte address
67 bne loop2 @ Repeat loop if r0 not empty
68 pop {r1, r2, r3, r4, r5, lr} @ Restore registers
69 bx lr @ Return function
70
71 addrString1: .int string1 @ Get address of string1
72 addrString2: .int string2 @ Get address of string2
73 addrFinalString: .int finalString @ Get address of finalString
```

Assembly Source Code

```
@ -----
@ Mark Lucernas
@ Unit 8 Assignment
@ -----

@ Constants
@ -----
.equ EXIT, 1                @ Exit code for system call

@ Data Section
@ -----
.data

@ Initialize Data
@ -----
.balign 4
string1: .asciz "Mark"
string2: .asciz "Lucernas"
message: .asciz "%s, %s, %s\n"

@ Uninitialized Data
@ -----
.bss
result: .skip 255           @ Reserve memory for concated result string

@ Code Section
@ -----
.text
.global main
.extern printf

main:  push {ip, lr}        @ push return address (lr) + dummy register (ip)

      ldr r1, addrString1   @ r1 <- string1 address
      ldr r2, addrString2   @ r2 <- string2 address
      ldr r3, addrResult    @ r3 <- result address
      bl concat             @ Call concat()

      ldr r0, =message      @ Load message (output format)
      bl printf             @ Print message

exit:  pop {ip, lr}         @ Pop the values from stack into ip and pc register
      mov r0, #0            @ r0 <- 0 (Exit code)
      mov r7, #EXIT         @ Request to exit program
      svc 0                 @ Perform the system call
```

@ Functions

@ -----

@ Concatenate string from r1 and r2 and store it in r3

concat: push {r1, r2, r3, r4, r5, lr} @ Save registers

mov r4, #0 @ r4 <- 0 (i = 0)

loop1: ldrb r0, [r1, r4] @ Load next byte (char) of string1 into r0
strb r0, [r3, r4] @ Store next byte (char) of string1 into finalString
cmp r0, #0 @ Compare if r0 is empty (no more to load from string1)
addne r4, #1 @ If not empty: r4 <- r4 + 1
bne loop1 @ Repeat loop if r0 not empty

mov r5, #0 @ r5 <- 0 (j = 0)

loop2: ldrb r0, [r2, r5] @ Load next byte (char) of string2 into r0
strb r0, [r3, r4] @ Store next byte (char) of string2 into finalString
cmp r0, #0 @ Compare if r0 is empty (no more to load from string1)
addne r4, #1 @ If not empty: r4 <- r4 + 1
addne r5, #1 @ If not empty: Go to next string1 byte address
bne loop2 @ Repeat loop if r0 not empty
pop {r1, r2, r3, r4, r5, lr} @ Restore registers
bx lr @ Return function

addrString1: .int string1 @ Get address of string1

addrString2: .int string2 @ Get address of string2

addrResult: .int result @ Get address of result

GDB Debugger of Assembly Code

```
+---Register group: general-----+
|r0      0x4d    77              r1      0x21028  135208
|r2      0x2102d 135213          r3      0x21043  135235
|r4      0x1     1              r5      0x0     0
|r6      0x10318 66328          r7      0x0     0
|r8      0x0     0              r9      0x0     0
|r10     0x76fff000 1996484608   r11     0x0     0
|r12     0x7efff180 2130702720   sp      0x7efff0e0 0x7efff0e0
|lr      0x10454 66644          pc      0x10474 0x10474 <loop1>

> 0x10474 <loop1>      ldrb  r0, [r1, r4]
0x10478 <loop1+4>      strb  r0, [r3, r4]
0x1047c <loop1+8>      cmp   r0, #0
0x10480 <loop1+12>     addne r4, r4, #1
0x10484 <loop1+16>     bne   0x10474 <loop1>
0x10488 <loop1+20>     mov   r5, #0
0x1048c <loop2>        ldrb  r0, [r2, r5]
0x10490 <loop2+4>      strb  r0, [r3, r4]

native process 31751 In: loop1                                L??  PC: 0x10474
0x00010478 in loop1 ()
0x0001047c in loop1 ()
0x00010480 in loop1 ()
0x00010484 in loop1 ()
0x00010474 in loop1 ()
(gdb) |
```

Compile and Run

```
cisc-211-13@raspberrypi:~/unit-8_assignment $ ./run_programs.sh

[ COMPILE ] Compiling 'lucernas_assignment_8.s' and 'LucernasAssignment8.java'...
as -o lucernas_assignment_8.o lucernas_assignment_8.s
gcc -o lucernas_assignment_8 lucernas_assignment_8.o
javac LucernasAssignment8.java

[ RUN ] Running 'lucernas_assignment_8'...
Mark, Lucernas, MarkLucernas

[ RUN ] Running 'LucernasAssginment8.class'...
Mark, Lucernas, MarkLucernas

[ CLEAN ] Cleaning up...
rm *.o *.class lucernas_assignment_8

Done

cisc-211-13@raspberrypi:~/unit-8_assignment $ |
```

Problem 2

Exercise 7.2

- a) **STR** and **B**. because these instructions write to the register file when they shouldn't.