Christopher Martin, Jennifer Wolfe, Geoffrey Corey
April 26, 2013
CS 472 Lab 2

Part 1: Problem Set 5

(1) **Write a simple program to perform Z = A + B + C - (D x E)**

```
        AREA        LabTwo, CODE, READONLY
        ENTRY

        MOV         r0, #5
        MOV         r1, #2
        MOV         r2, #2
        MOV         r3, #2
        MOV         r4, #3
        ADD         r0, r0, r1
        ADD         r0, r0, r2
        MUL         r5, r3, r4
        SUB         r5, r0, r5

        END
```

(2) **Assume A, B, C, D, and E are 16-bit values in memory.**

```
        AREA        LabTwo, CODE, READONLY
        ENTRY
        LDR         r0, A
        LDR         r1, B
        LDR         r2, C
        LDR         r3, D
        LDR         r4, E
        ADD         r0, r0, r1
        ADD         r0, r0, r2
        MUL         r5, r3, r4
        SUB         r5, r0, r5

A        DCD        4
B        DCD        12
C        DCD        -2
D        DCD        2
E        DCD        3

        END
```

(3) **Write a program that has deliberate syntax errors, and then debug it.**

The syntax errors we did was not put whitespace before the opcode in part 1 and then we put whitespace before the directive when it needs to be next to the margin. We were getting an A1163E error.

```
        AREA        LabTwo, CODE, READONLY
        ENTRY

        LDR         r0, A
        LDR         r1, B
```

```
            LDR         r2, C
            LDR         r3, D
            LDR         r4, E
            ADD         r0, r0, r1
            ADD         r0, r0, r2
            MUL         r5, r3, r4
            SUB         r5, r0, r5

A           DCD         4
B           DCD         12
C           DCD         -2
D           DCD         2
E           DCD         3

            END
```

Part 2: Examination of compiler output

Palindrome Checking

C function

```c
/* C */
#include <string.h>
#include <stdio.h>

int main(void){
        char str[4] = "mom",rev[4];
        int i,j, k;

        for(i = strlen(str)-1 , j = 0; i >= 0; i--, j++){
                rev[j] = str[i];
        }
       rev[j] = '\0';


        if(strcmp(rev,str)){
                k = 0;
        }else{
                k = 1;
        }

        return 0;
}
```

O0

```
;## -O0 ##
main:
        push        {r7, lr}
        sub         sp, sp, #24
        add         r7, sp, #0
        movw         r3, #:lower16:__stack_chk_guard
        movt         r3, #:upper16:__stack_chk_guard
```

```
        ldr     r3, [r3, #0]
        str     r3, [r7, #20]
        movw    r3, #28525
        movt    r3, 109
        str     r3, [r7, #12]
        add     r3, r7, #12
        mov     r0, r3
        bl      strlen
        mov     r3, r0
        add     r3, r3, #-1
        str     r3, [r7, #0]
        mov     r3, #0
        str     r3, [r7, #4]
        b       .L2
.L3:
        add     r2, r7, #12
        ldr     r3, [r7, #0]
        adds    r3, r2, r3
        ldrb    r2, [r3, #0]    @ zero_extendqisi2
        add     r1, r7, #16
        ldr     r3, [r7, #4]
        adds    r3, r1, r3
        strb    r2, [r3, #0]
        ldr     r3, [r7, #0]
        add     r3, r3, #-1
        str     r3, [r7, #0]
        ldr     r3, [r7, #4]
        add     r3, r3, #1
        str     r3, [r7, #4]
.L2:
        ldr     r3, [r7, #0]
        mvn     r3, r3
        lsr     r3, r3, #31
        uxtb    r3, r3
        cmp     r3, #0
        bne     .L3
        add     r2, r7, #16
        ldr     r3, [r7, #4]
        adds    r3, r2, r3
        mov     r2, #0
        strb    r2, [r3, #0]
        add     r2, r7, #16
        add     r3, r7, #12
        mov     r0, r2
        mov     r1, r3
        bl      strcmp
        mov     r3, r0
        cmp     r3, #0
        beq     .L4
        mov     r3, #0
        str     r3, [r7, #8]
```

```
        b       .L5
.L4:
        mov     r3, #1
        str     r3, [r7, #8]
.L5:
        mov     r3, #0
        mov     r0, r3
        movw    r3, #:lower16:__stack_chk_guard
        movt    r3, #:upper16:__stack_chk_guard
        ldr     r2, [r7, #20]
        ldr     r3, [r3, #0]
        cmp     r2, r3
        beq     .L6
        bl      __stack_chk_fail
.L6:
        add     r7, r7, #24
        mov     sp, r7
        pop     {r7, pc}
```

O1

```
;## -O1 ##
main:
        push    {lr}
        sub     sp, sp, #12
        movw    r3, #:lower16:__stack_chk_guard
        movt    r3, #:upper16:__stack_chk_guard
        ldr     r3, [r3, #0]
        str     r3, [sp, #4]
        movw    r3, #28525
        movt    r3, 109
        add     r0, sp, #8
        str     r3, [r0, #-8]!
        mov     r0, sp
        bl      strlen
        subs    r0, r0, #1
        bmi     .L2
.L5:
        subs    r0, r0, #1
        bpl     .L5
.L2:
        mov     r0, #0
        movw    r3, #:lower16:__stack_chk_guard
        movt    r3, #:upper16:__stack_chk_guard
        ldr     r2, [sp, #4]
        ldr     r3, [r3, #0]
        cmp     r2, r3
        beq     .L4
        bl      __stack_chk_fail
.L4:
        add     sp, sp, #12
        pop     {pc}
```

O2

```
;## -O2 ##
main:
        movs        r0, #0
        bx        lr
```

O3

```
;## -O3 ##
main:
        movs        r0, #0
        bx        lr
```

Os

```
;## -Os ##
main:
        movs        r0, #0
        bx        lr
```

Discussion

-O0 In the -O0 assembly version of a palindrome checker, there is significantly more code used here than in the version written for the homework assignment. One of the most notable differences is the amount of keywords used for values. The other difference is that the assembly written for the homework assignment is much easier to follow. The other difference is that the C version is written assuming a working operating system in place, instead of bare bones access to the CPU itself, as well as library interaction.

-O1 In the -O1 assembly version, there was a significant drop in the length of code being executed. Following the code is still much harder to do because there are still non-documented offsets, variables, and jump locations not directly documented in the segment of code. But there is much more code reuse in this version. there is also the same difference in the C version vs. this one in that the C version assumes there are libraries already available, the the homework version assumes no libraries available.

-O2 In this optimization level, there seems to be an error not reported through the web interface, as there are only 2 operations performed. This type of unexpected behaviour can occur when utilizing optimization levels. However, the amount of failure in this situation signals that an error has occurred somewhere else.

-O3 Again, this optimization level only shows 2 operations when translated to assembly; just as in the -O2 case. This amount of errors could mean that in the ARM version of the string.h libraries could behave incorrectly when any optimization level above 1 is used.

-O2 This optimization level is the most surprising with the results. This optimization level would logically seem like it should work, however the results showed -Os was broken as well. With this amount of broken compiled code, it would seem as if the way the palindrome checking was written is not the best possible algorithm possible and could be quite fragile.