Lab 7 Efficient Load and Store in Assembly

10/29/21

11/5/21

Jeremiah Webb

**Introduction:**

Being able to understand how loading and storing works in assembly is essential to efficient and safe programs. This lab helps specifically understanding how arguments in C are loaded and stored in assembly code, and how to effectively convert C code to assembly code.

Text

Description automatically generated**Screenshots:**

Text

Description automatically generated

Text

Description automatically generated

**Code Snippet:**

**//Code for Task 11**

task11 PROC

MOV r2, #0

task11\_loop

CMP r2, r1 ; test = r2 - r1

BGE task11\_end ; if test >= 0, then branch to task11\_end

MOV r3, r2, LSL #5 ; r3 <- r2 \* 32

ADD r3, #2 ; r3 <- r3 + 2

STRB r3, [r0, r2] ; r3 -> mem[r0 + r2] or r3 -> mem[r0 + i]

ADD r2, #1 ; r2 <- r2 + 1

B task11\_loop ; branch to task11\_loop

task11\_end

BX lr

ENDP

**//Code for Task 13**

task13 PROC

PUSH {r4-r5, lr}

; r0 = gPtrArray11a

; r1 = gPtrArray13a

; r2 = gVar1

SUB r2, #1

MOV r3, #0

task13\_loop

CMP r3, r2

BGE task13\_end

LDRB r4, [r0]

LDRB r5, [r0, #1]

ADD r4, r5

STRH r4, [r1, r3, LSL #1]

ADD r3, #1

ADD r0, #1

B task13\_loop

task13\_end

POP {r4-r5, pc}

ENDP

**//Code for Task 15**

task15 PROC

PUSH {r4-r5, lr}

; r0 = gPtrArray13a

; r1 = gPtrArray15a

; r2 = gVar1

SUB r2, #1

MOV r3, #0; i = 0

task15\_loop

CMP r3, r2

BGE task15\_end

LDRH r4, [r0], #2; load gPtr13a to temp

LDRH r5, [r0]; load gPtr11a + 1

ADD r5, r4, r5, LSL # 4; = temp + 16 \* (\*gPtr13a)

STRH r5, [r1, r3, LSL #2]

ADD r3, #1; increment i

B task15\_loop

task15\_end

POP {r4-r5, pc}

ENDP

END

**Discussion:**

The major thing I had to deal with for the program to work was reanalyzing how the arguments in C were transferred into assembly. First grasping how those functions use the arguments in C and then being able to use the LDR and push and pop instructions assisted with the overall construction of the tasks. One major insight was how to use LDRH and how that works with differently sized variables.

**Result:**

I learned specifically how registers are loaded with data and how the registers can be used to simulate C functions. Furthermore, with understanding how these registers are used, I can create more efficient code than were I to use regular C. Even doing basic functions will be more efficient in assembly. Passing back and forth data from C and assembly can now be easily done as shown by this lab.