Ramzi El-abdallah

Lab 1

Part 1

4) Stack Pointer (R13): 0x20018000
Link Register (R14): 0x08000281
Program Counter (R15): 0x080001D0

5) Instruction: SUB sp, sp, #0x28
Address: 0x080001D0

It is the same as the address in register R15 (Program counter). This represents the address of the current instruction being run.

6) R13 (SP) and R15 (PC) values changed. Program counter was incremented to the next instruction and stack pointer goes to the next address on the stack.

7) 0x080001D6 E9CD2308 STRD r2, r3, [sp, #0x20]
0x080001DA E9CD0106 STRD r0, r1, [sp, #0x18]

8) Stack Pointer (R13): 0x20017FD8
Link Register (R14): 0x08000281
Program Counter (R15): 0x080001E2

9) Stack Pointer (R13): 0x20017FD8
Link Register (R14): 0x080001E7
Program Counter (R15): 0x080001E6

Link Register changed because the program branched to a subroutine and that was the address of the routine. Program counter address is the address of the next instruction. Program counter does agree with Disassembly Window.

10) R0: 0x20017FFD Value of scr pointer
R1: 0x20017FE9 Value of dst pointer

12) a: “Hello World!”

13) b: Holds the copied string that will be capitalized

14) R2 holds the value of the character

15) R0: 0x20017FFD
R1: 0x20017FE9
R2: 0x00000000

R14: 0x080001E7
R15: 0x080001B8

16) R15: 0x080001E6

17) When a subroutine is finished it loads the address from LR to PC and continues where it left off in main.

Part 2

```
__asm void my_lowercase(char *str)
{
cap_loop
        LDRB r1, [r0]     ; Load byte into r1 from memory pointed to by r0 (str pointer)
        CMP r1, #'A'-1    ; compare it with the character before 'a'
        BLS cap_skip      ; If byte is lower or same, then skip this byte
        CMP r1, #'Z'      ; Compare it with the 'z' character
        BHI cap_skip      ; If it is higher, then skip this byte
        ADDS r1,#32       ; Else subtract out difference to capitalize it
        STRB r1, [r0]     ; Store the capitalized byte back in memory
cap_skip
        ADDS r0, r0, #1   ; Increment str pointer
        CMP r1, #0        ; Was the byte 0?
        BNE cap_loop      ; If not, repeat the loop
        BX lr             ; Else return from subroutine
}
```