Lab 2 - EECS 211
Interrupts/Traps and Service Routines
Winter 2023

Problem 1

Currently, the LC3 VM implements the "Trap" and "Interrupt" handling differently than how the hardware actually works. However, since the OS context switching depends mainly on the interrupts and traps, it is important to re-implement this part in the LC3 VM to match what happens in an actual hardware.

A trap occurs whenever the assembly instruction "TRAP (op code = 0xF)" is used. The assembly instruction also contains a number called the trap vector (trapvect8). When the CPU executes the trap, it performs the following two steps:

- The value in the program counter (PC) register will be copied to R7
- The PC will point to the address saved in Memory[trapvect8].

Let's consider the example below:

Listing 1: Assembly Code # 1

```
        .ORIG   0x0025
        .FILL TRAP_HALT

        .ORIG 0x0100
TRAP_HALT
            ....               ; put your Trap Service Routine code here



            .ORIG 0x3000
            ....               ; the software code starts from here
            TRAP 0x25          ; this is exactly equivalent to .HALT
```

Recall that the HALT trap has a trapvect8 equal to 0x25. In the example above, the value of 0x0100 is saved inside the memory location 0x25. Therefore, executing the HALT trap assembly instruction will force the PC will point to the address 0x0100.

In this problem, make the following changes to the LC3 VM:

1. Change the LC3 VM such to implement the right sequence of actions for the traps. Recall this is the trap vector for the LC3 VM:
   - TRAP_GETC = 0x20, /* get character from keyboard, not echoed onto the terminal */
   - TRAP_OUT = 0x21, /* output a character */
   - TRAP_PUTS = 0x22, /* output a word string */
   - TRAP_IN = 0x23, /* get character from keyboard, echoed onto the terminal */
   - TRAP_PUTSP = 0x24, /* output a byte string */
   - TRAP_HALT = 0x25 /* halt the program */
2. If the software uses the TRAP instruction with any other value for trapvect8 (other than the ones above), your VM should dump the memory/registers to a file (using the same template used in the previous assignment).
3. Now you can write code for each trap service routines (which is considered the first part of your OS). The code for the trap service routines will be located at 0x0100. For now, will have very simple trap service routines that simply uses TRAP 0xFF to dump the memory followed by an infinite loop.

To test your code, use the simple code below:

Listing 2: Assembly Code # 2

```
            .ORIG 0x3000
            TRAP 0x25           ; this is exactly equivalent to .HALT
```

Deliverable

You need to upload one file named “memory_dump_p1_1” that contains the memory/registers for the assembly code #2 above.

Problem 2

In this problem, we are going to edit the HALT trap to run a process. We will assume the code of this process will be located in 0x4000,

This simple (dummy) process should perform the following instructions:

1. Load a value ”1” in the register R1.

2. Dump the memory using TRAP xFF instruction.

The main body of the OS is located at 0x3000 (where the CPU starts executing). For this simple OS, it just needs to run the process by calling the HALT trap instruction. You should use the register R3 to store the jump address value of your dummy process in the HALT trap service routine.

You should also force a dump of the memory at the very beginning of the HALT trap service routine (by using TRAP 0xFF instruction).

Deliverable

You will upload one zip file that contains (1) all your .asm files (OS and dummy process), (2) your VM, and (3) 2 memory dump files called "memory_dump_p2_1" and "memory_dump_p2_2".