Lab 4 - EECS 211
Basics of Memory Protection
Winter 2023

Problem 1

Right now, the LC3 VM does not differentiate between user and kernel model. In this problem, you need to change the LC3 VM to add a Supervisory Mode to the processor.

- Add a memory mapped register at location 0xFE04. This register will be called MR_SM.
- Whenever MR_SM has a value of 1, then the CPU is in the supervisory (kernel mode). Whenever MR_SM has a value of 0, then the CPU is in the user mode.
- If the software tries to write any value on this register (for example by loading a value to the memory location 0xFE04) while the CPU is in the user mode, then the VM should refuse to execute such instruction, dump the memory into a file, and move to the next assembly instruction.
- Whenever a Trap instruction is executed, the MR_SM register will be set automatically to 1 by the CPU.
- Whenever the CPU is in the kernel mode, then it should allow the software to write a value of 0 in the MR_SM.

Deliverable

Append the OS from Assignment 1 to set the MR_SM to 0 just before it switches from one process to another.

Write the code of 5 simple (dummy) processes that will be located in 0x4000, 0x5000, 0x6000, 0x7000, and 0x8000, respectively. These processes should perform the following instructions:

1. Load a certain value in the register R1. For example, the first process should load the value of "1" into the register R1, the second process should load the value of "2" into the register R1, and so on.
2. Store the value inside the register R1 inside the memory location 0xFE04.
3. Store the value inside the register R1 inside the memory location 0x0200 (which is inside the OS memory).
4. Halt.

You will upload one zip file that contains (1) all your .asm files (OS and dummy processes), (2) your VM, and (3) 17 memory dump files called "memory_dump_p2_1", "memory_dump_p2_2", "memory_dump_p2_3", ..., etc, that corresponds to the memory dumps issued during the switches between these processes.

Problem 2

In this problem, you need to change the virtual LC3 processor to add a simple Memory Protection Unit (MPU). An MPU is a trimmed down version of the Memory Management Unit (MMU) which supports the memory protection functionality without providing address translations. The MPU allows the privileged instructions (i.e., those executed whenever MR_SM has a value of 1) to define memory regions and assign memory access permission and memory attributes to each of them. This means that privileged instructions can access any location in the memory. On the other hand, unprivileged load/store instructions will be protected by the MPU. In this problem, you need to do the following:

- Add a memory mapped register at location 0xFE06. This register will be called MR_MPU.
- Whenever MR_MPU has a value of 0, then the software running on the CPU can access any memory location.
- Whenever MR_MPU has a value of 1, then the software running on the CPU can access only memory locations between 0x3000 - 0x3FFF.
- Whenever MR_MPU has a value of 2, then the software running on the CPU can access only memory locations between 0x4000 - 0x4FFF.
- Whenever MR_MPU has a value of 3, then the software running on the CPU can access only memory locations between 0x5000 - 0x5FFF.
- Whenever MR_MPU has a value of 4, then the software running on the CPU can access only memory locations between 0x6000 - 0x6FFF.
- Whenever MR_MPU has a value of 5, then the software running on the CPU can access only memory locations between 0x7000 - 0x7FFF.
- Whenever MR_MPU has a value of 6, then the software running on the CPU can access only memory locations between 0x8000 - 0x8FFF.

Deliverable

Append the OS to set the MR_MPU with the right value before it switches to the process. Run the same 5 processes from the previous problem using your updated OS.

You will upload one zip file that contains (1) all your .asm files (OS and dummy processes), (2) your VM, and (3) 12 memory dump files called "memory_dump_p2_1", "memory_dump_p2_2",

"memory_dump_p2_3", ..., etc, that corresponds to the memory dumps issued during the switches between these processes.