Lab 4: Multi-core and Locks

18307130024 Jimmy Tan

1 Multi-processing Systems

This section is based on ARMv8-A Programmer Guide.

1.1 Getting the information about the running core

Software may need to know which core its code is executing upon. `ARMv8` provides the *Multi-Processor Affinity Register* (`MPIDR_EL1`) to identify the core thus satisfying the demand.

To configure a virtual machine, `EL2` and `EL3` can set `MPIDR_EL1` to different values at run-time. `MPIDR_EL3` is based on the physical core and its value cannot be modified.

1.2 Symmetric multi-processing

In symmetric multi-processing (`SMP`), each core has the same view of memory and of shared hardware. There is a layer of abstraction from the softwares' perspective because the operating system hides the complexity of scheduling the tasks within or between the cores.

There are several trade-offs on whether to spread the tasks on more or less cores:

- Energy. On one hand, scheduling tasks on fewer cores can provide more idle resources. On the other hand, to achieve the same performance of a single core, spreading tasks on multiple cores requires lower frequencies, thus saving the power.
- Interrupt handling. Interrupts can be handled by multiple cores, which can reduce interrupt latency and the time spent on context switching. Similarly, only allowing interrupts to be handled on a few certain cores can reduce complexity.

1.3 Timers