Title: Modular Development of a Certified Concurrent Operating System Kernel and Hypervisor

Description: Formal verification is a key to secure and reliable software, and Operating System (OS) kernels and hypervisors form the backbone of every safety-critical software system in the world. Therefore, it is highly desirable to formally verify the correctness of these programs. In this sense, our team introduced CertiKOS (Certified Kit Operating System), a hypervisor architecture that leverages formal certification to ensure correctness and counter information leakage in cloud computing, as well as mCertiKOS, the verified version of CertiKOS. However, mCertiKOS does not have several features that we have already implemented in CertiKOS, such as parallel (multi-core) hardware support, device drivers, and an input / output memory management unit (IOMMU). So, the goal of my research is extending mCertiKOS to support those features.

Among what I have mentioned, I will focus on adding concurrency to mCertiKOS. To achieve the goal, my work can be divided as two parts, which are the works on mCertiKOS implementation as well as verification tools that we are currently using. As we have discussed before, mCertiKOS assumes a runtime environment consisting of a single processor. Therefore, it would be highly desirable to extend the current kernel and hypervisor to support multi-core hardware platform which is prevalent today. In addition to that, extending mCertiKOS is strongly related to extending the software verification tools that we are currently using. Most lines of C code in mCertiKOS are written in ClightX which is a variant of the CompCert Clight language. In addition, we are currently using CompCertX to compile the program written in ClightX, which is also a variant of CompCert. However, CompCertX cannot compile parallel programs, so we cannot directly use CompCertX as a verification tool for the kernel which supports multi-core hardware. This implies us that we need a more powerful tool to verify a concurrent OS kernel and hypervisor or have to find a way that can make CompCertX to use for parallel program verification.