Cacheable Overlay Manager RISC-V

Ronen Haen and Ofer Shinaar

In the early days of embedded computing there was a technique to load code on Real-Tim at the moment it was needed for execution. This technique was named Overlay, and it was threaded with the toolchain (compiler, liker, and more, etc.) providing easy application-interface for the SW developers.

NASA used this technique in their early shuttles' flight-control system. Flight computers had very limited memory, so they used SW overlays only a small amount of code necessary for a particular phase of the flight (e.g. , ascent, on-orbit, or entry activates) is loaded in computer memory at any one time. At quiescent points in the mission the memory contents are "swapped out" for program applications that are needed for the next phase of the mission. (https://www.nap.edu/read/5018/chapter/4)

This technique slowly disappeared due to the fact that memories became bigger in capacity (and small in size), cheaper, and MMU emerged, replacing the need for overlay software solutions.

Today, IoT devices (Internet of things) are quite strict with memory size and power, alongside requirements for simple HW implementation which does not contain MMU or high-end operation system to manage it (linux/windows).
Those needs, alongside RISC-V code density challenges, rises a need to revive the overlay concept to fit to RISC-V ISA, and use its toolchain to support it.

We would like to present a "catchable" firmware manager to be threaded with the Real-Time code and to the toolchain.
Cacheable Overlay Manager RISC-V (ComRV), a technique which fits small devices (as IoT's), and does not need any HW support.

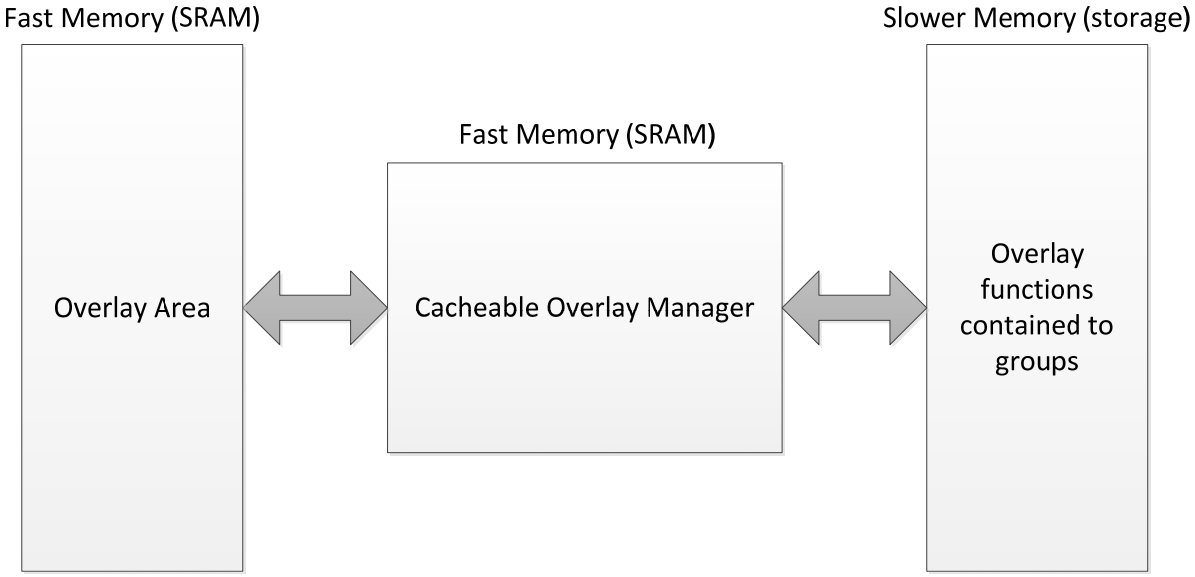

Figure 1 - Basic Overlay concept

A run-time engine is running on the fast memory deciding which function to load or not from storage device to fast memory heap as displayed on ***Figure 1 - Basic Overlay concept:***

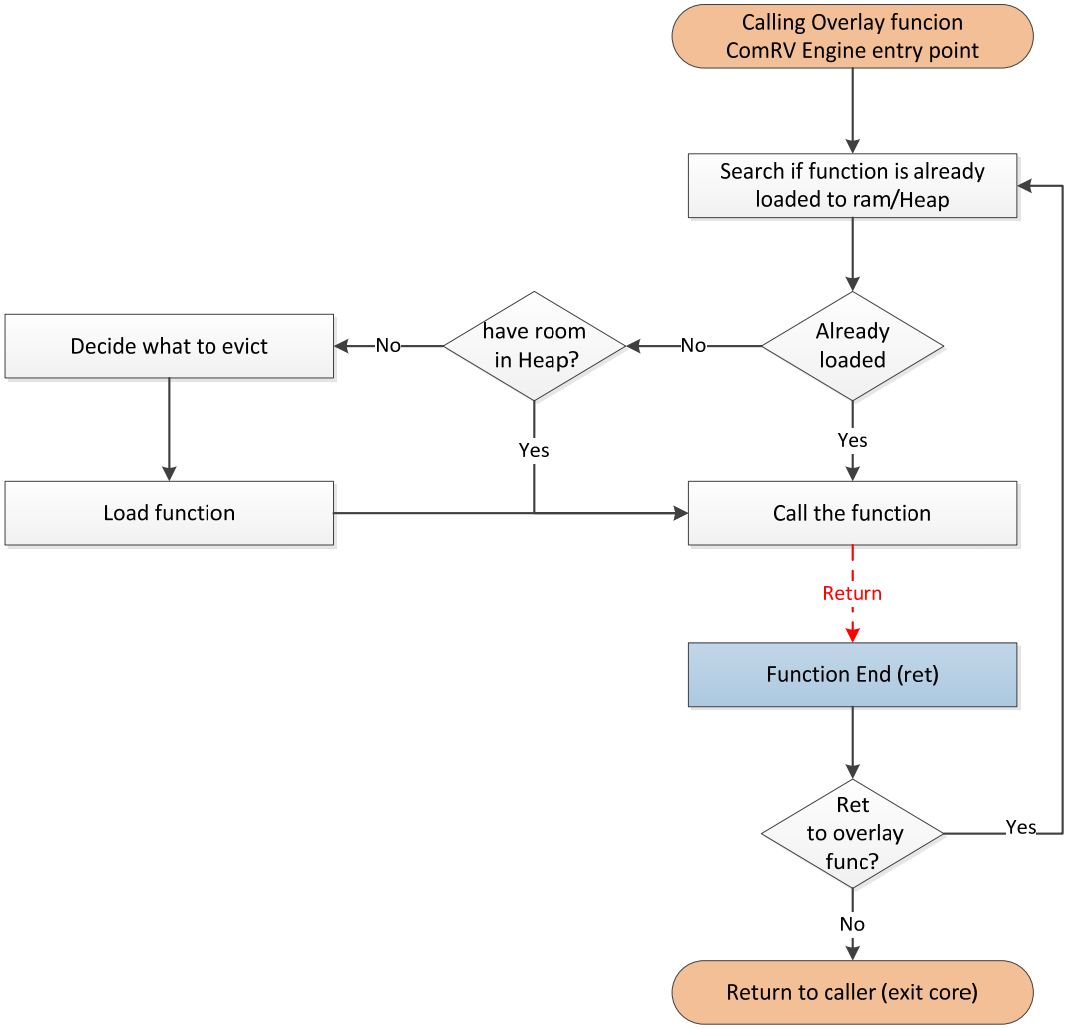

Figure 2 - basic ComRV Flow

The ComRV engine acts as cache mechanism, as shown in ***Figure 2 - basic ComRV Flow.*** A call for overlay function will be followed by calling ComRV engine to decide if the function is loaded to heap or not. If not loaded, the ComRV will need to decide what is preferred/candidate to evict using same concepts as an HW cache.

We plan to promote ComRV FW engine, and the toolchain related changes to support it, to the open-source community.

