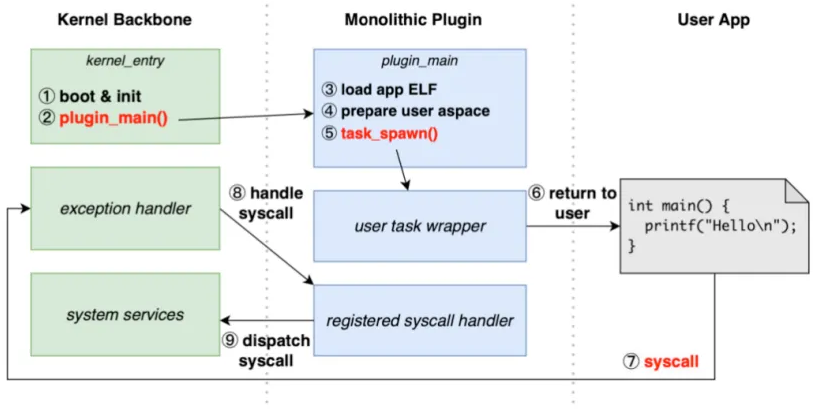
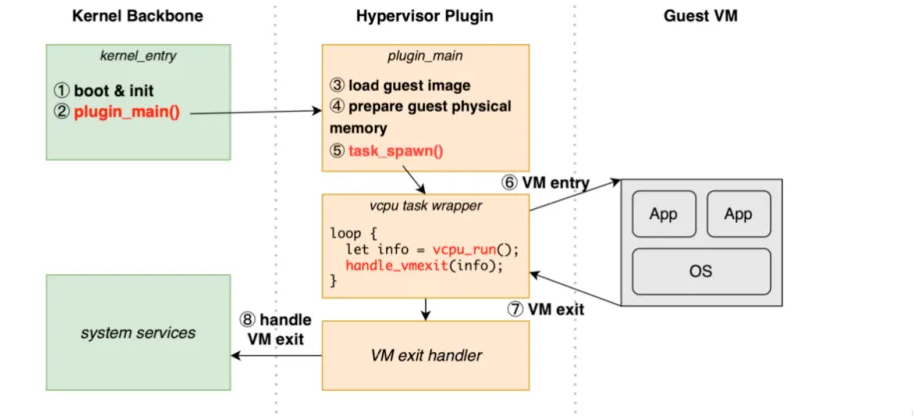
# 组件化内核的异构扩展实现

## ARCEOS 接入异构内核 —— 宏内核

### Monolithic Plugin




### Hypervisor Plugin




## Arceos 接入异构内核细节：TASK扩展

1.采用指针的方式，将一些调度信息和资源信息分离，将资源信息用指针的方式存储在Task中

```
struct AxTask {
    id: usize,
    state: AtomicU8,
    /// task extended data
    task_ext_ptr: *mut u8,
}
```

2.初始化方式

- 预先定义好扩展对象

- 调用相关宏来初始化扩展域，需要初始化Task和Task-ext
- 之后便可像正常域使用

```
// Define task extended data for monolithic kernel
struct MonolithicTaskExt {
    proc_id: usize, // process ID
    uctx: UspaceContext, // user space context
    aspace: AddrSpace, // virtual memory address space
}
axtask::def_task_ext!(MonolithicTaskExt);
// Usage in monolithic kernel plugin
fn spawn(entry) {
    let new_task = AxTask::new(entry);
    new_task.init_task_ext(MonolithicTaskExt { proc_id, uctx, aspace });
    axtask::spawn_task(task)
}
fn mmap(addr, len, prot, flags) {
    axtask::current_task().task_ext().aspace.mmap(addr, len, prot, flags)
}
```

3.Task拓展具体实现方式

- 编译期确定扩展域大小
- 在堆上申请内存
- 将扩展域指针指向该内存
- 对外提供相关的引用接口

```
// Expansion of macro def_task_ext!
static __AX_TASK_EXT_SIZE: usize = size_of::<MonolithicTaskExt>();
static __AX_TASK_EXT_ALIGN: usize = align_of::<MonolithicTaskExt>();
impl TaskExtRef<MonolithicTaskExt> for axtask::AxTask {
    fn task_ext(&self) -> &MonolithicTaskExt {
        unsafe { &*self.task_ext_ptr as *const MonolithicTaskExt }
    }
}

// Defined in axtask
pub trait TaskExtRef<T: Sized> {
    fn task_ext(&self) -> &T;
}
impl AxTask {
    fn init_task_ext<T: Sized>(&mut self, data: T) {
        self.task_ext_ptr = alloc(__AX_TASK_EXT_SIZE, __AX_TASK_EXT_ALIGN);
        self.task_ext_ptr.write(data);
    }
}
```