**question1**

The macro `MPBOOTPHYS(s)` is to translate the link address to the physical address.

The code in `mpentry.S` is above KERNBASE for which it can't be addressed directly in the physical address. And we need physical address rather than virtual address.We have copied the memory above `mpentry_start` to `MPENTRY_PADDR`, so we need `MPBOOTPHYS` to get the physical address of the dataa and instructions.

**question2**

There are still mutical processes changing from user mode and pushing there information to kernel stack just before the lock. We don't protect the progress that "push information to kernel stack", so we need separate kernel stacks to cancel the conflict.

**question3**

The variable e is stored above KERNBASE, for which its memory is mapped to the same location in all environments. All environments share the same kernel mappings.

**question4**

The process itself doesn't have a construct to save its registers, so the old environment's registers must be saved.

Firstly, `_alltraps` in `trapentry.S` push the context to the stack. Then, in `trap()` in `kern/trap.c`, `curenv->env_tf = *tf;`, copy trap frame (which is currently on the stack) into 'curenv->env_tf', so that running the environment will restart at the trap point.