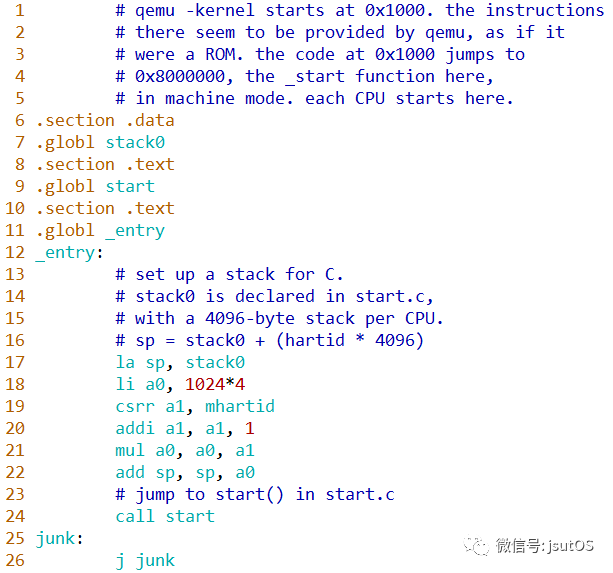
**系统启动实验指导书**

xv6系统运行在qemu虚拟机中，使用qemu启动xv6时，qemu将xv6内核加载到内存的0x8000000开始的存储空间中，然后，在RISC-V机器模式下，从 \_entry开始执行指令（kernel/entry.S:12）。xv6启动时RISC-V的页设备是禁用的，虚拟内存直接映射到物理内存。

一、entry.S

// kernel/entry.S源程序清单



\_entry开始的指令设置了一个栈（17-21），栈空间大小为4KB=(1024\*4)，mhartid是运行当前程序的CPU核的ID，其值范围是0～MaxCore-1，因为xv6运行在多核的RISC-V处理器上，系统需要为每个核设置一个堆栈。栈区是从高地址开始向低地址扩展，以16B对齐。17-21行用类语言描述如下：

sp = &stack0 # 将stack0的地址存入sp中

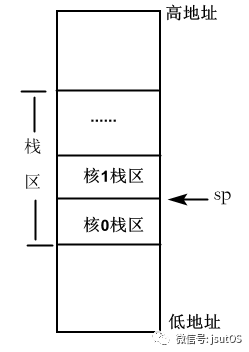
a0 = 1024\*4 # 4KB

a1 = CPU\_ID + 1

a0 = a0 \* a1

sp = sp + a0 ； #栈顶

此时堆栈设置完毕，如下图所示核0的堆栈区。sp是堆栈指针寄存器。

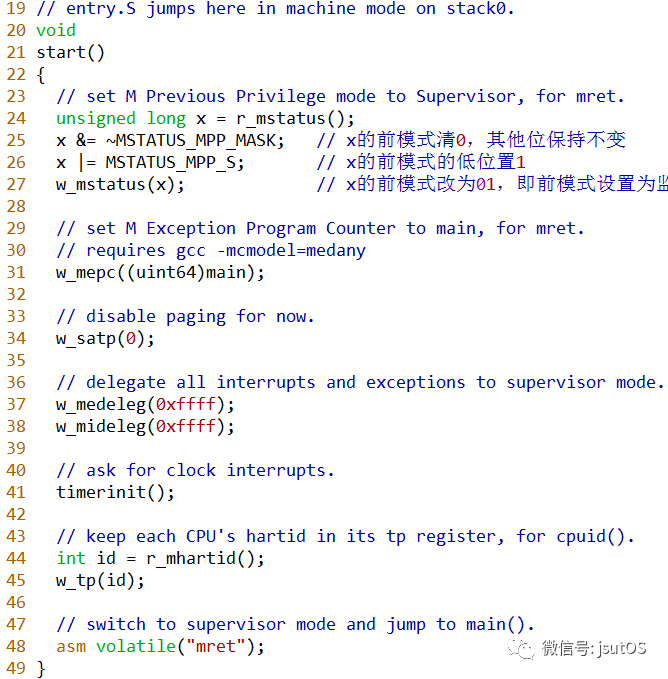


堆栈设置好后，程序调用start函数，start是一个C程序函数。一般调用start后程序不会回来，即25-26行不会执行到，若从调用start返回，说明操作系统有问题，系统转入死循环。

二、start.c

RISC-V的架构有机器模式（Machine Mode，M模式）、监督模式(Supervisor Mode，S模式）和用户模式（User Mode，U模式），并且规定机器模式是必须具备的模式，其他模式均是可选而非必选的模式。entry.S和start( )函数都是在机器模式下执行。start对环境进行必要的配置后通过mret 指令切换到监督模式，并执行main函数。

kernel/start.c程序段



start 24行：读mstatus寄存器的值保存在x中，25-26行将对应的MPP位设置为01，即前模式为监督模式。27行将修改后的x写回mstatus寄存器，当执行mret指令时，会从机器模式变换为监督模式。

31行：将main的地址看作机器模式下发生异常时指令的地址并保存在寄存器mepc，当执行mret指令时，程序从发生异常的指令处恢复执行。

34行：将0写入页表寄存器satp来禁用虚拟地址转换。

37-38行：将所有中断和异常委托给监督模式。

41行：对时钟芯片编程以产生计时器中断。

44行：读取核的ID。

45行：将ID保存到tp寄存器中。

48行：执行mret指令，系统由机器模式改变为监督模式，并从main()开始运行。

三、main.c

kernel/main.c程序段



17行：判断是否是第0个核，如果是执行18-37行，否则执行39-45行。

18行：初始化终端。

19行：初始化输出互斥锁

20-22行：显示提示信息：“xv6 kernel is booting\n”

23行：初始化物理内存页

24行：创建内核页表

25行：将h/w页表寄存器切换到内核的页表，并启用分页。

26行：初始化进程表

27行：设置trap向量

28行：安装内核向量

29行：设置中断控制器

30行：对S模式的hart设置uart启用位，及优先级阈值为0。注：hart指硬件线程。

31行：缓冲区初始化

32行：inode缓冲区初始化

33行：文件表初始化

34行：初始化虚拟磁盘。

35行：创建第一个用户进程。第一个进程执行一个小程序initcode.S（user/initcode.S:1），该程序通过调用exec系统调用重新进入内核。

36行：同步

37行：started=1

以上是核0执行的代码，对于其他核执行39-45行。

39-40行：等待核0初始化完成。

41行：同步

42行：输出核ID信息。

43-45行：初始化多个设备和子系统。

第一个进程执行一个小程序，initcode.S user/initcode.S:1，通过执行 exec 系统调用再次进入内核。exec 用一个新程序（在本例中，/init\*）替换了当前进程的内存和寄存器。一旦内核完成 \*exec ，它会返回用户空间并运行 /init。如果需要，init [user/init.c:11][init-c]会创建一个新控制台设备文件然后打开它为文件描述符0，1，2。然后它开始循环，开启一个控制台shell，在shell退出前处理孤儿和僵尸进程，并重复。系统就起来了。

**试题**

* **XV6系统如何启动？**

1. 首先，在bootasm.S中，系统必须初始化CPU的运行状态。具体地说，需要将x86 CPU从启动时默认的Intel 8088 16位实模式切换到80386之后的32位保护模式；然后设置初始的GDT，将虚拟地址直接按值映射到物理地址；最后，调用bootmain.c中的bootmain()函数。
2. bootmain()函数的主要任务是将内核的ELF文件从硬盘中加载进内存，并将控制权转交给内核程序。具体地说，此函数首先将ELF文件的前4096个字节从磁盘里加载进来，然后根据ELF文件头里记录的文件大小和不同的程序头信息，将完整的ELF文件加载到内存中。然后根据ELF文件里记录的入口点，将控制权转交给XV6系统。
3. entry.S的主要任务是设置页表，让分页硬件能够正常运行，然后跳转到main.c的main()函数处，开始整个操作系统的运行。
4. main()函数首先初始化了与内存管理、进程管理、中断控制、文件管理相关的各种模块，然后启动第一个叫做initcode的用户进程。至此，整个XV6系统启动完毕。