实验思考题

3.1

- 之所以按照逆序插入，是因为按照一般人的习惯，每次alloc出来的东西应该是有顺序的。按照逆序插入的话，将从envs[0]开始alloc

3.2

- 首先是必要性，这样生成的envid具有两两不同的性质，且envid可以反映他在envs中的位置然后是充分性，这样生成的envid在寻找下标的时候，也就是调用ENVX的时候仅需做一些简单位运算，效率高

3.3

- 之所以可以复制一部分pgdir，是因为那一部分（UTOP以上）对于任何进程来说都是相同的。总的来说，UTOP是用户写的边界，ULIM是用户读的边界。UTOP一下为用户自由使用，UTOP与ULIM之间的一段则用来给进程进入内核态查询使用。因为UVPT需要索引到用户pgdir的地址。保存之后可以很快的查找到在2G2G布局下，从进程的角度来说，前一部分完全经由MMU达成，后一部分只有当成为内核态进程之后，才能用一个固定的方法访问得到。而物理内存就在那里，永远也不会动。怎么访问，要看是走流程还是直接开花。

3.4

- 比如是加载模式的可选择，比如对齐方式，比如pgdir的选定具体的场景比如我现在为了效率，制定两个进程使用同一个pgdir或者同一个pgdir的一部分（size）来实现“内存共享”，这时便可以把这个参数利用起来。相当于是加载模式的参数化，类似的应用比如像sort函数中的自定义cmp

3.5

- 先ROUNDDOWN然后剩下的补齐，补零即可。具体分析请详见难点分析

3.6

- 是虚拟空间。
- 原因：在gxemul断点的时候，所有的指令位置均为0x8000....，因此是虚拟地址。理论上没有任何函数可以直接访问物理地址，就算不走MMU也要高位除0进行映射一样。这种统一是一种标准，是ELF格式的标准。就像ASCII码那样。如果没有标准，那实际的应用兼容性将会更差

3.7

- 应为curenv的cp0，因为那是上一次异常发生时的现场

3.8

- TIMESTACK是0x82000000，是时钟中断来临后保存现场的栈。在中断保护中，有li sp,0x82000000。TIME就是内部的时钟中断，STACK就是栈，这个栈本质上与kernel_sp

没有特别大的不同。只有在时钟中断来临进入exc_handler的时候，才会保存现场到TIMESTACK，其他的异常或中断都保存到kernel_sp

3.9

- 首尾定义函数，暂且不提。最后两句返回+延迟槽，暂且不提前两句相当于时钟初始化，设置定时器的频率然后将CP0的状态进行调整，以达到可以接受下一次中断的效果宏里还有一些保存和恢复临时使用的寄存器的操作

3.10

- 初始化接受中断许可数为pri，每接受一次中断，许可数（cnt）减一，为0时切换进程，同时完成新进程的初始化

实验难点

- 在本次实验我个人觉得比较难是在理解代码的时候，然后在理解各个宏定义的作用。在本次实验我花了4个小时写代码，1周+理解代码，因为有时候就想不到要怎么做，然后在第八步骤的时候花了比较多时间。
- 然后也有做还原系统因为我在做实验的时候不小心把实验环境的配置改了。