Exercise 1: core map

+------------------+ <- 0xFFFFFFFF (4GB)

| 32-bit |

| memory mapped |

| devices |

| |

/\/\/\/\/\/\/\/\/\/\

/\/\/\/\/\/\/\/\/\/\

| |

| Unused |

| |

+------------------+ <- depends on amount of RAM

| |

| |

| Extended Memory |

| |

| |

+------------------+ <- 0x00100000 (1MB)

| BIOS ROM |

+------------------+ <- 0x000F0000 (960KB)

| 16-bit devices, |

| expansion ROMs |

+------------------+ <- 0x000C0000 (768KB)

| VGA Display |

+------------------+ <- 0x000A0000 (640KB)

| |

| Low Memory |

| |

+------------------+ <- 0x00000000

base mem 640k, bootloader在7c00以上活动，boot栈在7c00以下，函数调用的时候还是会用到栈的

total 128M

VA's [0, 400000) to PA's [0, 400000)

之所以要将[0, 4MB)的虚拟地址空间也映射到物理地址[0, 4MB)是因为在开启Paging之后，entry.S中还有少许指令需要执行，当其执行完之后，就会进入C语言编写的代码来对内核进行初始化

VA's [f0000000, f0400000) to PA's [0, 400000)

Sections:(可重定位)

Idx Name Size VMA LMA File off Algn

0 .text 0000178e f0100000 00100000 00001000 2\*\*2

CONTENTS, ALLOC, LOAD, READONLY, CODE

1 .rodata 00000704 f01017a0 001017a0 000027a0 2\*\*5

CONTENTS, ALLOC, LOAD, READONLY, DATA

2 .stab 000044d1 f0101ea4 00101ea4 00002ea4 2\*\*2

CONTENTS, ALLOC, LOAD, READONLY, DATA

3 .stabstr 00008acd f0106375 00106375 00007375 2\*\*0

CONTENTS, ALLOC, LOAD, READONLY, DATA

4 .data 0000a300 f010f000 0010f000 00010000 2\*\*12

CONTENTS, ALLOC, LOAD, DATA

5 .bss 00000648 f0119300 00119300 0001a300 2\*\*5

CONTENTS, ALLOC, LOAD, DATA

kernel实际加载在 Extended Memory

0xf0110000~0xf0118000 kernel的栈，在.data段内

VGA b8000开始，通过虚拟地址访问+ f0000000

boot把kernel的elf放在0x10000

VA’s [0xef400000-0xef800000) to 一级页表（kern\_pgdir – KERNBASE）

jos的core map是连续分配在某一块区域的，xv6的是放在每个物理页的某一处

Exercise 2:

kernel 的2个段描述符号data 和code的dpl为0, 在长跳转的时候会指定段选择符

gdt:

SEG\_NULL # null seg

SEG(STA\_X|STA\_R, 0x0, 0xffffffff) # code seg

SEG(STA\_W, 0x0, 0xffffffff) # data seg

.set PROT\_MODE\_CSEG, 0x8

ljmp $PROT\_MODE\_CSEG, $protcseg

尽管MMU负责address translation，但是kernel还是知道规则的，因此，若kernel想访问某一个指定的物理地址，就可以换算成虚拟地址

6. kern\_pgdir 4KB

7.pages 256KB 还在前4M内

core map的链表初始化时，跳过了page 0 和VGA～pages结束（4K地址对齐），但是主动page\_free还是会回收这些区域的