清华大学本科生考试试题专用纸	
考试课程：操作系统（A卷）	时间：2019年04月8日上午09:50~11:25

系别：_____	班级：_____	学号：_____	姓名：_____

答卷注意事项:	1. 答题前，请先在试题纸和答卷本上写明A卷或B卷、系别、班级、学号和姓名。
	2. 在答卷本上答题时, 要写明题号, 不必抄题。
	3. 答题时, 要书写清楚和整洁。
	4. 请注意回答所有试题。本试卷有17个小题，共14页。
	5. 考试完毕, 必须将试题纸和答卷本一起交回。

一、(20分)判断题

1. [ ] 二次机会（时钟）页面替换算法需要读取和修改页表项中访问位才能有效实现。
2. [ ] 运行在用户态的进程只能通过系统调用才能直接获得操作系统的服务。
3. [ ] 在内核态中执行的内核线程不需要拥有自己独占的页表。
4. [ ] 如没有页机制的虚存管理，则无法实现采用Copy on Write机制的fork系统调用。
5. [ ] 如果父进程比子进程先结束，那么子进程结束后将成为资源无法被回收的僵尸进程。
6. [ ] 在x86-32中，在内核态通过对EFLAGS的设置可直接屏蔽时钟中断。
7. [ ] 在x86-32中，只有正确设定了段描述符表的相关配置，才能实现中断和异常处理。
8. [ ] 基于动态链接库的应用程序需要编译器能生成地址无关代码并需要OS加载库和程序。
9. [ ] 在ucore的bootloader代码实现中，在主引导记录 (master boot record) 格式下的512 字节的引导扇区 (boot sector) 中，除去最后两个字节 0x55AA, 还剩下 510 字节都能用来存放引导代码。
10. [ ] 在`lab1`中实现打印函数调用堆栈信息时， `print_debuginfo()` 的参数不仅可以是`eip-1`，还可以是`eip` 或`eip-2` 。##

二、问答题

11. (9分)

在`lab1`中实现打印函数调用堆栈信息时， `read_ebp` 为内联函数，而`read_eip` 为非内联函数。请结合`lab1`的实现回答，若`read_ebp` 为非内联函数，或`read_eip` 为内联函数， `print_stackframe()`函数的实现是否还能完成对应的功能？为什么？

附：lab1参考答案中与题干干相关的部分关键代码：

```
static inline uint32_t read_ebp(void) __attribute__((always_inline));

  // ...

static inline uint32_t read_ebp(void) {
  uint32_t ebp;
  asm volatile ("movl %%ebp, %0" : "=r" (ebp));
  return ebp;
}

// ...

static __noinline uint32_t read_eip(void) {
  uint32_t eip;
  asm volatile("movl 4(%%ebp), %0" : "=r" (eip));
  return eip;
}

// ...

void print_stackframe(void) {
  uint32_t ebp = read_ebp(), eip = read_eip();
  int i, j;
  for (i = 0; ebp != 0 && i < STACKFRAME_DEPTH; i ++) {
  cprintf("ebp:0x%08x eip:0x%08x args:", ebp, eip);
  uint32_t *args = (uint32_t *)ebp + 2;
  for (j = 0; j < 4; j ++) {
      cprintf("0x%08x ", args[j]);
    }
    cprintf("\n");
    print_debuginfo(eip - 1);
    eip = ((uint32_t *)ebp)[1];
    ebp = ((uint32_t *)ebp)[0];
  }
}
```

12. (15分)

假设采用连续内存分配的最佳匹配算法（ best fit ），有 1024KB 内存可供分配。使用`free list`组织未分配的内存，`free list`项是`(STARTADDR, LENGTH)`, 单位均为 KB, 按`STARTADDR`排序。 针对下面内存操作序列，回答问题。

- `alloc(SIZE)`函数将分配一块区域, 如果分配成功其返回一个元组`(STARTADDR, LENGTH)`, 失败的话其返回失败.
- `free(STARTADDR, LENGTH)`将一块分配的区域释放. 题中`free`的参数一定是之前某个`alloc`的返回值.

操作	返回值	free list
Z = alloc 1025KB	失败	[(0, 1024)]
A = alloc 256KB	(0, 256)	[(256, 768)]
B = alloc 128KB	(256, 128)	[(384, 640)]
C = alloc 256KB	(384, 256)	[(640, 384)]
D = alloc 128KB	(640, 128)	[(768, 256)]
free B	/	[(256, 128), (768, 256)]
free D	/	[(256, 128), (640, 384)]
E = alloc 512KB	____	____
F = alloc 257KB	____	____
G = alloc 64KB	____	____
H = alloc 64KB	____	____
free F	____	____
I = alloc 300KB	失败	同上

1. 模仿示例, 将上表填空
2. 分配 I 为什么失败了? 当时总共剩余的空闲内存有多少? 这种现象称为什么? 试列出解决这种现象的两种方法。

13. (12分)

有一台假想的计算机，页大小（page size）为32 Bytes，支持32KB的虚拟地址空间（virtual address space）,有4KB的物理内存空间（physical memory）。采用二级页表，每个页目录项（page directory entries，PDEs）和页表项（page-table entries， PTEs）的大小都为1 Byte，每个页目录表和页表的大小都为32 Bytes。页目录基址寄存器（page directory base register，PDBR）保存了页目录表的物理页号。4KB的物理内存布局和具体值请见本题最后部分。

格式说明：

```
PTE格式（8 bit，左高位，右低位） ：
  VALID | PFN6 ... PFN0
PDE格式（8 bit，左高位，右低位） ：
  VALID | PT6 ... PT0
```

其中

```
VALID==1表示，表示映射存在；VALID==0表示，表示映射不存在。
PFN6...PFN0:  表示页帧号
PT6...PT0:  表示页表的物理基址>>5
```

请给出下列虚地址对应的pde index (页目录项的索引)，pde contents（页目录项值），pte index (页表项的索引)，pte contents (页表项值)。请回答下列虚地址是否有合法对应的物理内存。如果没有，请写出原因；如果有，给出物理地址所在内存中的值（一个字节）。

注意：下面只是回答的格式形式，具体的数值没有意义。

```
Virtual Address 7268:
  ---->pde index:0x1c  pde contents:(valid 1, pfn 0x5e)
    --    --> pte index:0x13  pte contents:(valid 1, pfn 0x65)
      --       -->Translates to Physical Address 0xca8 --> Value: 16  //
有，值为16
```

Virtual Address 0a90 Virtual Address 7913 Virtual Address 7215 Virtual Address 2cff

PDBR: 29 (decimal) [This means the page directory is held in this page]

```
page   0: 7f 7f 7f 7f 7f 7f 7f fb 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
f8 7f 7f 7f
page   1: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page   2: 0d 02 1e 17 0f 15 06 07 10 17 14 00 13 1c 17 1c 1e 06 09 10 1e 0b 1d 1a 10 13 1d 18
09 1a 06 0b
page   3: 15 16 17 04 08 16 14 0a 1e 09 1b 04 0e 1e 15 17 1b 13 10 19 10 1d 17 00 08 1b 03 00
12 15 1e 1d
page   4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page   5: 1d 0f 08 0f 0f 00 1b 12 16 19 18 00 15 13 14 11 1b 02 0b 18 0c 1c 1c 11 16 01 08 10
00 14 10 04
page   6: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page   7: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f e1 7f a4 7f 7f 7f 7f 7f 7f 7f 7f e0 7f 7f
7f 7f 7f 8f
page   8: 0f 11 15 03 1a 06 18 0c 0f 19 1c 10 01 09 16 19 1b 11 03 08 0d 04 0a 09 16 0e 11 09
16 11 02 1c
```

```
page   9: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  10: 1a 1c 0f 08 13 0c 05 03 0c 04 08 1e 1c 1b 0f 17 05 13 1e 18 1e 13 09 00 08 19 10 13
18 09 04 1b
page  11: 0d 05 1e 0d 1b 1e 1a 0b 05 0d 07 16 17 1d 0b 17 16 15 05 19 0b 0e 17 1e 1a 1e 1e 19
0a 09 1b 04
page  12: 7f 7f ac 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
7f 7f 7f 7f
page  13: 12 17 07 14 15 0d 18 0d 04 02 04 16 0f 04 1d 00 12 0e 1b 12 1a 06 10 18 0b 16 0c 0a
08 15 1e 01
page  14: 15 09 0c 16 0a 0f 0f 0f 02 0c 0b 0d 08 0d 16 02 02 10 11 18 01 00 12 06 0f 09 02 11
09 0f 1a 0d
page  15: 01 03 0b 03 17 0d 0f 06 11 00 03 04 09 19 08 06 0e 00 0b 13 11 1a 0e 10 04 10 1b 12
01 0c 17 00
page  16: 02 07 1e 06 13 0e 0e 0f 05 19 02 07 00 08 0c 1b 0b 03 08 1e 01 13 0b 14 0a 15 0f 14
1b 12 04 01
page  17: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  18: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f a0 7f 7f 7f 7f 7f 7f
7f 7f 7f 7f
page  19: 7f 7f 7f 7f 7f 7f 7f 7f f7 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
7f 8b 7f 7f
page  20: 19 1a 0f 18 0b 17 11 07 01 0c 0b 12 19 09 16 04 03 1b 1c 08 07 17 08 14 05 17 1c 05
03 09 15 06
page  21: 7f 7f 7f 7f 7f b0 7f 7f 7f 7f 7f 7f 7f 7f 7f e6 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 88 7f
bf 7f 7f 7f
page  22: 05 14 1e 06 17 09 00 17 0a 05 0d 07 0c 0d 0c 12 08 02 02 03 10 0b 19 0f 15 1e 01 00
04 0b 11 17
page  23: 09 16 03 16 06 0c 06 04 04 1e 03 1c 0e 1b 05 12 15 09 13 1c 0c 00 12 18 0e 1a 0b 0d
0d 02 02 13
page  24: 0b 0f 1c 0c 0b 02 14 0e 14 19 01 09 14 0b 16 06 0c 0e 0a 18 10 1d 0e 1b 16 05 12 1b
00 0a 06 0f
page  25: 7f 7f 7f 7f 7f c9 7f 7f 7f 7f 7f c4 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f c5
7f 7f 7f 7f
page  26: 1c 04 03 07 0b 0f 14 07 14 13 0d 1e 0d 05 00 0e 0b 15 15 06 05 04 09 09 17 0e 02 1b
08 1e 17 12
page  27: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f e5 7f
7f 7f 7f 7f
page  28: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 8d
7f 7f 7f 7f
page  29: 7f a3 9b ea a8 87 95 cc 9c 99 ad 80 8c df 93 fd f4 c1 eb ae c3 fc d5 b5 7f a7 b3 92
b2 b8 da f6
page  30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  31: 02 18 1a 0e 17 18 1a 1d 14 12 01 06 1e 10 1d 06 19 18 16 09 1e 0a 1e 1d 02 02 0c 08
0c 05 17 02
page  32: 10 11 15 00 04 04 09 19 1a 16 11 00 08 0f 16 0b 0e 09 10 0d 0b 1c 12 14 1a 16 08 17
13 10 15 19
```

```
page  33: 0a 0f 1a 07 0c 08 0e 01 13 1a 1c 0b 05 16 17 05 01 14 0b 10 11 1b 03 08 01 15 02 1a
12 0f 08 1a
page  34: 08 07 03 08 12 1d 0b 0f 05 0c 06 0b 04 1c 19 05 01 0d 0a 1b 06 1b 1b 10 1b 19 04 1b
1d 0d 19 01
page  35: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 85 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
7f 7f 7f 7f
page  36: 05 01 06 02 0b 17 17 05 08 09 01 1c 16 0d 07 0c 00 03 0e 06 0b 17 18 17 05 1e 1c 09
11 02 0d 10
page  37: 0a 10 03 16 0a 04 07 0c 0d 18 03 06 09 05 17 03 01 11 16 19 11 0a 1b 16 0e 11 1d 1e
1c 04 0a 0c
page  38: 1b 01 14 12 08 16 09 02 0f 01 11 06 0f 1d 15 11 18 0d 11 1d 1e 00 13 17 1a 10 03 0d
08 0f 03 03
page  39: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f f5
7f 7f 7f 7f
page  40: 7f 7f bd 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f f1 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
7f 7f 7f 7f
page  41: 04 13 1a 11 17 0d 15 06 15 04 05 03 0a 00 15 1c 19 01 0f 04 11 00 1d 0e 05 0c 09 0f
0c 1d 09 02
page  42: 05 00 09 0e 10 13 01 1d 04 05 03 0b 04 16 0a 03 0b 19 11 02 12 0f 10 17 04 15 17 1c
0b 06 15 1e
page  43: 0b 06 07 08 05 15 1c 06 0a 0e 10 12 1c 19 01 0a 17 18 0a 1b 02 0a 0e 06 14 00 1d 17
06 1a 11 19
page  44: 0c 01 0c 02 18 15 1d 16 0a 13 04 0d 03 1d 14 07 10 0c 1b 14 15 13 04 11 1a 0b 00 09
00 11 0e 03
page  45: 7f 7f 7f 7f 7f 7f a5 7f 7f 7f 7f 7f 7f 7f 7f cd 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
7f 7f 7f 7f
page  46: e4 7f 7f 7f c6 7f 7f 83 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 82 7f 7f 7f 7f 7f 7f 7f 7f 7f
7f 7f 7f 7f
page  47: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  48: 0e 12 1c 0b 1b 1d 13 16 1a 0c 11 0f 03 19 02 0d 0e 15 03 13 00 12 16 1d 00 18 1e 08
1c 19 09 0e
page  49: 08 02 0d 0f 0a 16 09 1e 1c 11 03 11 13 14 0d 16 1e 15 01 13 13 10 1d 01 00 16 03 04
19 0b 1b 18
page  50: 7f 7f 7f 7f 7f 7f 7f 7f 7f b4 d7 ab 7f 7f f3 7f 96 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
7f 7f 7f 7f
page  51: 7f 7f 94 7f 7f 7f 7f db 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
7f 7f 97 7f
page  52: 00 0d 1b 14 07 03 1b 02 13 12 05 0d 1a 1e 02 00 08 01 1c 02 0b 1a 0e 07 01 07 09 1a
0d 11 08 0c
page  53: 7f 7f 7f 7f 7f 7f c8 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
7f 7f 7f 7f
page  54: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  55: 17 0a 0b 07 1d 14 12 05 1e 1b 04 07 1b 01 0f 0b 16 0e 0d 07 0b 00 02 13 0f 04 11 11
07 0b 0b 07
page  56: 7f 7f 7f 7f fe 7f 7f e3 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f b1 8e 7f 7f 7f 7f 7f
7f 7f 7f 7f
```

```
page  57: 02 06 05 10 0f 0c 1b 01 19 0d 05 08 07 1e 11 09 08 1c 08 1a 0a 00 00 1c 1d 1a 10 00
10 07 0e 1a
page  58: 1a 10 15 16 15 18 1e 15 11 0a 1e 06 09 0a 0a 00 17 0c 05 0c 18 1e 12 11 02 19 1c 1b
13 0b 0a 18
page  59: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  60: 0b 1d 1a 0c 04 1b 18 1c 0c 07 10 10 16 04 1d 10 08 12 02 19 10 06 01 10 18 07 1d 12
0f 1e 0a 15
page  61: 16 10 04 07 16 01 00 16 08 11 06 07 1d 1d 1b 0e 08 16 0d 18 0d 13 0e 13 0c 0f 03 10
1e 03 03 04
page  62: 06 1d 11 16 1b 08 0b 1b 04 17 03 15 15 1d 1a 0f 06 04 10 0f 02 1b 0f 15 06 07 00 0a
08 0d 0b 19
page  63: 1e 05 01 0a 1c 01 06 05 1a 0a 0d 10 1d 13 16 0a 05 1c 10 12 0b 0c 12 14 05 01 09 1e
06 0c 01 0a
page  64: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  65: 7f 7f 7f 7f 7f 7f 7f 7f 7f d8 7f 7f 7f 7f 7f 7f 7f d3 7f 7f 7f bc d1 7f 7f 7f 7f 7f
7f 7f 7f a1
page  66: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  67: 7f d9 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f a6 7f 7f 7f e2 7f 7f 7f 7f 7f 7f 7f 7f
7f 7f 7f 7f
page  68: 09 18 1b 08 03 14 12 0c 05 1e 03 1a 07 07 0c 04 17 05 08 0a 18 01 16 08 14 1e 05 00
09 16 07 0a
page  69: 02 1b 1a 04 0a 10 17 0e 1c 05 11 02 06 04 08 0c 15 00 1c 05 14 04 00 0e 0e 1d 04 1b
0d 1e 02 0f
page  70: 1b 1e 06 17 1d 12 0f 0b 0f 12 12 03 1e 08 10 06 13 02 0a 0b 09 0f 04 0f 18 13 1e 1d
0e 0e 17 1b
page  71: 06 04 04 0c 1c 1e 1e 0b 00 18 16 1c 00 1c 1a 1b 00 1c 03 10 09 12 0e 1c 00 17 11 0a
1a 08 18 0e
page  72: 0f 1e 05 0e 12 17 02 03 08 14 06 01 0c 04 15 1d 17 11 0c 00 0d 10 09 0c 1d 0b 08 0d
1c 01 0e 1e
page  73: 03 1a 0a 05 15 04 14 00 1a 05 05 10 14 0d 0a 0e 18 10 05 19 04 11 06 08 00 11 02 1e
17 1c 0f 1a
page  74: 14 01 03 08 16 04 1d 1b 11 1b 13 04 0c 00 0d 1e 0c 14 0b 14 0a 1c 0c 08 0d 02 0a 16
0c 0d 07 19
page  75: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  76: 7f 7f 7f 7f ba 7f 7f 7f 7f 7f 7f 7f 7f 7f c7 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
7f 7f 7f e8
page  77: 00 17 0e 16 19 1c 00 08 0a 10 09 0d 0f 17 02 0b 07 0d 0b 07 1d 00 01 16 06 1c 13 19
1d 17 1d 05
page  78: 11 01 15 0e 1e 07 03 02 11 13 0f 1b 1e 04 10 0d 02 06 11 0d 08 01 15 1b 04 1a 0c 1c
1d 1d 00 17
page  79: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
```

```
page  81: 08 19 1a 16 07 0e 14 10 09 06 0b 19 17 10 00 0d 13 06 0d 10 05 05 08 1d 1d 01 14 1b
0c 13 0c 1d
page  82: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  83: 13 09 05 03 03 05 0f 11 15 0d 17 00 0a 03 10 11 12 0d 0b 0a 03 05 0f 11 03 1a 00 03
0d 0a 10 15
page  84: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  85: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f f9 7f 7f 7f 7f 7f
98 7f 7f 7f
page  86: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  87: 1c 1a 1e 13 13 0e 06 18 12 05 07 06 16 12 06 15 11 08 0d 1e 04 07 12 1a 1c 0c 17 03
0b 1a 14 0d
page  88: 0f 0c 0d 0e 18 04 03 1c 0f 15 1c 16 11 1d 13 0d 05 04 18 1b 0b 10 15 12 08 1c 11 16
15 0d 15 05
page  89: 1a 15 03 13 01 0a 0a 1c 06 0b 11 06 0e 09 16 18 12 1a 0f 17 02 0c 0a 1b 0f 0b 14 1d
19 18 14 18
page  90: 7f 7f 7f 7f 7f 7f 7f 7f be 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f ec 7f 7f 7f 90 7f 7f
7f 7f 7f 7f
page  91: 02 19 16 04 18 10 0e 0a 0d 0a 0b 08 15 02 17 08 19 01 19 05 0e 15 0b 0b 09 0c 09 0c
07 00 19 0c
page  92: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  93: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  94: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page  95: 7f a2 7f 7f 7f 7f ed 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
7f 7f 7f 7f
page  96: 1b 0a 09 04 0d 17 1c 0d 19 05 1c 15 07 0f 16 19 0a 16 1c 05 03 07 16 1d 16 1b 0e 11
0f 0c 15 00
page  97: 1b 12 13 0c 1d 11 1b 1c 15 1e 0f 0b 1a 10 09 15 0e 03 19 0f 1e 10 1d 08 06 10 17 19
14 03 0c 01
page  98: 04 1a 0c 1a 1c 15 0e 09 0c 1b 0b 0d 1d 1b 01 1b 13 19 04 1e 1e 06 10 0e 17 13 19 01
13 0a 06 1d
page  99: 13 17 0f 14 0f 15 05 09 09 11 1b 0e 10 1b 04 08 14 18 10 07 19 0f 00 04 19 09 03 08
18 03 0c 19
page 100: 0e 14 0c 03 02 05 0d 08 11 14 1c 04 1d 00 1b 1e 0b 18 0f 11 1a 1d 09 06 1c 12 1b 06
1a 04 04 08
page 101: 13 07 1b 06 03 01 16 06 09 1e 16 16 11 13 00 19 0a 1a 0c 1c 00 1e 02 0f 0e 1e 16 1e
1c 0d 04 04
page 102: 15 09 04 03 0f 1e 00 11 15 07 04 1c 18 0a 1d 0e 1a 15 0e 0b 02 0b 16 0a 09 0a 02 1a
14 13 1a 12
page 103: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
page 104: 19 0b 18 18 0c 14 18 1e 1c 18 04 12 15 02 0e 1a 1c 1c 02 12 1a 14 09 01 1c 1a 14 0b
19 19 16 12
```

```
page 105: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 106: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f fa 7f 7f 9f 7f 7f 7f 7f 7f 7f ca 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page 107: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f b7 7f 7f 7f 7f
page 108: 09 09 13 1a 15 1d 16 19 12 0b 0d 0f 10 19 08 02 16 08 0d 03 19 08 09 18 0f 08 1b 0c 0b 1d 18 17
page 109: 1e 03 07 0b 16 10 0c 1c 0c 06 00 1e 1e 1e 12 10 1d 10 0a 14 07 0f 05 07 0b 15 19 02 0f 1a 03 07
page 110: 1d 06 15 09 15 19 14 0f 14 15 05 16 03 1a 0a 02 00 12 02 0c 01 15 1a 00 13 14 11 1d 09 1b 10 06
page 111: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 112: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
page 113: 01 07 0b 0b 11 09 14 18 18 16 12 08 1a 08 1a 0d 13 0f 14 0a 11 16 0b 12 0e 0e 0b 10 15 05 04 19
page 114: 1d 0c 02 1b 1b 18 19 00 0b 14 1a 0e 09 10 17 05 09 07 05 05 00 0b 07 18 1a 06 11 0d 14 0d 1d 13
page 115: 0a 0c 01 00 11 1b 13 04 14 04 0e 17 0c 09 09 09 00 16 0c 05 09 01 0e 12 13 0d 1e 04 06 11 0c 09
page 116: 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page 117: 1e 16 0d 18 0e 1b 16 07 12 1e 0e 0b 1a 10 02 11 06 08 07 11 11 04 04 11 14 02 01 03 12 01 08 0a
page 118: 7f a9 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f ee 7f 7f 7f 7f 7f 7f 9a 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page 119: 12 0f 0a 0e 0a 18 11 11 04 0d 19 1c 07 0f 0d 1b 13 11 0c 10 19 04 01 1e 1c 08 1e 13 1e 07 1d 07
page 120: 0a 15 13 04 15 02 05 11 03 1d 0d 13 1d 03 1e 01 12 14 1e 0b 1d 14 0e 05 07 0b 0a 0b 19 0b 13 01
page 121: 0a 12 14 07 15 06 00 01 0f 15 06 0c 15 08 10 12 0c 06 15 15 05 19 14 1a 18 0f 16 09 14 02 0a 0f
page 122: 14 10 1e 0b 1d 05 06 14 05 1c 0c 05 1d 17 13 12 13 07 0b 16 1a 11 12 11 07 07 04 1a 14 17 1b 13
page 123: 19 06 1b 01 1d 17 1e 11 05 1e 0d 02 12 0f 0e 04 03 12 0e 03 0d 16 16 0e 0c 0b 06 06 0d 03 16 15
page 124: 7f 7f 7f 7f 7f 7f 8a 7f 7f b9 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f f2 7f 7f 7f 7f 7f 7f 7f
page 125: ce 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
page 126: 1e 1c 16 13 0d 0e 0c 05 01 0b 1d 02 0a 11 18 1b 12 0f 1b 08 11 0d 0c 0b 0f 1b 1c 02 0d 07 1d 11
page 127: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

14. (4分)

页表自映射机制是指：将一个页目录项指向的物理地址，设置为页目录自己的物理地址。这样我们就将整个页目录和页表都映射到了虚拟地址空间。

假设在x86-32下，我们将页目录的最后一项（下标为0x3ff=1023）设置为自映射。

1. 页目录的起始虚拟地址是?
2. 用自映射机制实现类似ucore lab2中`get_pte`函数：（假设页表存在）

```
pte_t *get_pte(uintptr_t la) {
    return (pte_t*)(/* ???? (只允许使用la) */);
}
```

15. (15分)

1. 试说明描述全局和局部置换算法的不同，并分别各举出一种属于全局置换和局部置换的算法。
2. 假定在一个虚拟存储系统中某进程分配了4个物理页面，当进程按**c, a, d, b, e, b, a, b, c, d**的序列进行页面访问时，当分别使用时钟置换算法与LRU算法时会出现多少次缺页？要求说明过程。
3. 什么是Belady现象？请判断OPT、LRU、FIFO、Clock和LFU等各页面置换算法是否存在Belady现象?

16. (10分)

下面是ucore中进程切换的相关代码。

文件`/kern/process/proc.h`

```
// Saved registers for kernel context switches.
// Don't need to save all the %fs etc. segment registers,
// because they are constant across kernel contexts.
// Save all the regular registers so we don't need to care
// which are caller save, but not the return register %eax.
// (Not saving %eax just simplifies the switching code.)
// The layout of context must match code in switch.S.
struct context {
    uint32_t eip;
    uint32_t esp;
    uint32_t ebx;
    uint32_t ecx;
    uint32_t edx;
    uint32_t esi;
    uint32_t edi;
    uint32_t ebp;
};

#define PROC_NAME_LEN               50
#define MAX_PROCESS                 4096
```

```
#define MAX_PID                     (MAX_PROCESS * 2)

extern list_entry_t proc_list;

struct inode;

struct proc_struct {
    enum proc_state state;                      // Process state
    int pid;                                    // Process ID
    int runs;                                   // the running times of Proces
    uintptr_t kstack;                           // Process kernel stack
    volatile bool need_resched;                 // bool value: need to be
rescheduled to release CPU?
    struct proc_struct *parent;                 // the parent process
    struct mm_struct *mm;                       // Process's memory management
field
    struct context context;                     // Switch here to run process
    struct trapframe *tf;                       // Trap frame for current
interrupt
    uintptr_t cr3;                              // CR3 register: the base addr
of Page Directroy Table(PDT)
    uint32_t flags;                             // Process flag
    char name[PROC_NAME_LEN + 1];               // Process name
    list_entry_t list_link;                     // Process link list
    list_entry_t hash_link;                     // Process hash list
    int exit_code;                              // exit code (be sent to
parent proc)
    uint32_t wait_state;                        // waiting state
    struct proc_struct *cptr, *yptr, *optr;     // relations between processes
    struct run_queue *rq;                       // running queue contains
Process
    list_entry_t run_link;                      // the entry linked in run
queue
    int time_slice;                             // time slice for occupying
the CPU
    skew_heap_entry_t lab6_run_pool;            // FOR LAB6 ONLY: the entry in
the run pool
    uint32_t lab6_stride;                       // FOR LAB6 ONLY: the current
stride of the process
    uint32_t lab6_priority;                     // FOR LAB6 ONLY: the priority
of process, set by lab6_set_priority(uint32_t)
    struct files_struct *filesp;                // the file related info(pwd,
files_count, files_array, fs_semaphore) of process
};
```

文件`/kern/trap/trap.h`

```
struct trapframe {
    struct pushregs tf_regs;
    uint16_t tf_gs;
    uint16_t tf_padding0;
    uint16_t tf_fs;
    uint16_t tf_padding1;
    uint16_t tf_es;
    uint16_t tf_padding2;
    uint16_t tf_ds;
    uint16_t tf_padding3;
    uint32_t tf_trapno;
    /* below here defined by x86 hardware */
    uint32_t tf_err;
    uintptr_t tf_eip;
    uint16_t tf_cs;
    uint16_t tf_padding4;
    uint32_t tf_eflags;
    /* below here only when crossing rings, such as from user to kernel */
    uintptr_t tf_esp;
    uint16_t tf_ss;
    uint16_t tf_padding5;
} __attribute__((packed));
```

文件`/kern/process/proc.c`

```
// proc_run - make process "proc" running on cpu
// NOTE: before call switch_to, should load  base addr of "proc"'s new PDT
void
proc_run(struct proc_struct *proc) {
    if (proc != current) {
        bool intr_flag;
        struct proc_struct *prev = current, *next = proc;
        local_intr_save(intr_flag);
        {
            current = proc;
            load_esp0(next->kstack + KSTACKSIZE);
            lcr3(next->cr3);
            switch_to(&(prev->context), &(next->context));
        }
        local_intr_restore(intr_flag);
    }
}
```

文件`/kern/process/switch.S`

```
.text
.globl switch_to
switch_to:                      # switch_to(from, to)

    # save from's registers
    movl 4(%esp), %eax
    popl 0(%eax)
    movl %esp, 4(%eax)
    movl %ebx, 8(%eax)
    movl %ecx, 12(%eax)
    movl %edx, 16(%eax)
    movl %esi, 20(%eax)
    movl %edi, 24(%eax)
    movl %ebp, 28(%eax)

    # restore to's registers
    movl 4(%esp), %eax
    movl 28(%eax), %ebp
    movl 24(%eax), %edi
    movl 20(%eax), %esi
    movl 16(%eax), %edx
    movl 12(%eax), %ecx
    movl 8(%eax), %ebx
    movl 4(%eax), %esp

    pushl 0(%eax)

    ret
```

请分析函数`proc_run()`和`switch_to()`的代码，然后回答下面问题。

1. "切换进程地址空间"是哪一条语句完成的；
2. "恢复下一个进程的EIP"是哪一条语句完成的?
3. "切换进程内核栈"是哪一条语句完成的?
4. 指令`movl 24(%eax), %edi`的功能是什么？其中的24表示什么含义？eax寄存器中是什么数据?

17. (15分)

考虑下列程序

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main(){
    int val = 2;

    printf("%d", 0);
    fflush(stdout);

    if (fork() == 0) {
        val++;
        printf("%d", val);
        fflush(stdout);
    } else {
        val--;
        printf("%d", val);
        fflush(stdout);
        wait(NULL);
    }

    val++;
    printf("%d", val);
    fflush(stdout);
    exit(0);
}
```

上述代码没有检查返回值，我们假设所有函数均正确返回。 对于下列字符串，确定字符串是否可能是程序的一种输出，圈出Y/N。

```
  01432   Y N
  01342   Y N
  03142   Y N
  01234   Y N
  03412   Y N
```