ICS 第六章作业

3. (1) 16KB / 1K × 8位 = 16片 DRAM

(2) 48KB / 16KB = 3个

(3) 主存地址16位，低10位表示芯片内地址，其中前5位为行地址，后5位为列地址

4. (1) (32K×8位) / (8K×4位) = 4×2 = 8个

(2) RAM区大小为 16MB - 32kB = $(16\times 2^{10}-32)$ KB = 511 × 32 KB

故需要 (511 × 32 KB × 8位) / (8KB × 4位) = 511 × 4 × 2 = 4088个.

5. (1) 每面有 51mm × 3.92 TPM = 200 道，每道信息量为 3.14 × (355.6 − 2×51) × 90 BPM = 71664 bit，故磁盘容量为 20 × 200 × 71664 bit ≈ 273 Mbit = 34.1 MB

道密度和位密度都扩大100倍，容量扩大10000倍，容量为 273 Mbit × 10^4 ≈ 3336 GB.

(2) 平均寻道时间为 ((200−1) × 0.2 + 0) / 2 = 19.9 ms，转一圈时间为 60 × 1000 / 2400 RPM = 25 ms，平均旋转时间为 (25ms + 0) / 2 = 12.5ms，故平均存取时间 = 19.4 + 12.5 ms = 32.4 ms，平均数均传输速率为 71664 bit / 25 ms = 2.87 Mbit/s.

(3) 平均寻道时间为 ((20000−1) × 0.002 + 0) / 2 = 20ms，转一圈时间变为 25ms / 3 = 8.33ms

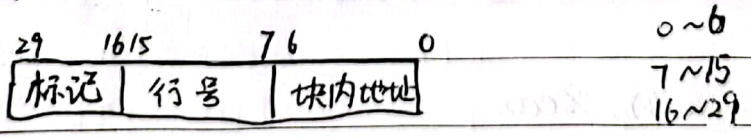
故平均存取时间 = 20ms + 8.33/2 ms = 24.2 ms；平均数均传输速率为 300 × 2.87 Mbit/s = 861 Mbit/s.

8. (1) cache共 64KB / 128B = 512行，直接映射方式下行号占9位；

主存块 128B 按字节编址，块内地址为7位；

主存地址空间为1GB，故主存地址位数为30位 ($1GB = 2^{30}B$)；

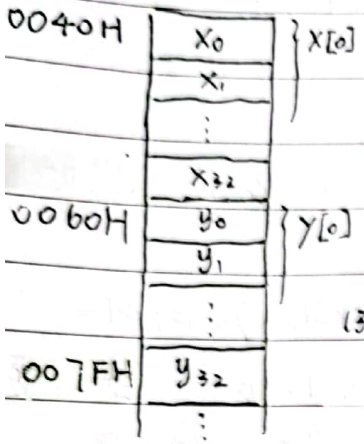
故主存地址中标记为 30 − 9 − 7 = 14位，地址划分为：标记(高14位)，行索引(中间9位)，块内地址(低7位)



(2) 采用直接映射，回写方式，不用考虑控制位与修改位，故每个cache行包括1位有效位，14位标记位和128B数据，共有 512 × (1 + 14 + 128 × 8) bit = 533968位.

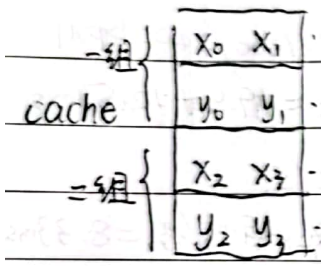
12. (1) 时间局部性无法判断，因为数组中每个元素只被访问一次；
空间局部性较好，因为是顺序访问数组中每个元素；
无法判断命中率高低；

(2)



cache 有 32 B / 16 B = 2 行，每个主存块(cache行)中可放4个元素，其中x中的元素在第四、五两个主存块中，y的元素在六、七两个主存块中，因此 x[0]~x[3] 和 y[0]~y[3] 映射到cache第一行中，x[4]~x[7] 和 y[4]~y[7] 映射到cache第二行中，因此 x[i] 与 y[i] 总是冲突，cache命中率为0.

(3) 改为2-路组相联，同时块大小为8B，故cache有 32B/8B=4行，分为两组，一个主存块放两个元素，因此x中的元素放在第8~11块中，y中的元素放在12~15块中。且因为每组有两行，x_{ai} 和 y_i 可以读取到且不冲突，命中的元素为 x_1 和 y_1，x_3 和 y_3，x_5 和 y_5，x_7 和 y_7，命中率为50%.



(4) 若改为x中12个元素，则 x[i] 与 y[i][i] 不会发生冲突，每块中四个元素可以命中3个，命中率为75%

23. (1) addl (%edx,%ecx,4),%eax，"基址+比例变址+偏移量".

(2) 取指令操作得到线性地址为 0x0 + 0x8049c08 = 0x8049c08；
取数操作得到线性地址为 (0x804d000 + 50×4) + 0x0 = 0x804d0c8；

(3)
```
    movl    $0,%ecx
.Loop
    cmpl    %ebx,%ecx
    jge     .EXIT
    addl    (%edx,%ecx,4),%eax
    incl    %ecx
    jmp     .Loop
.EXIT
```

(4) PE=1(保护模式)，PG=1(启用分页)；

(5) 不会发生缺页异常，因为I不在页面起始处，但是取a[0]时可能发生缺页，因为a[0]的地址为0x804d000恰一个页面起始处(页大小为4KB，页起始地址后12位全0)，故在执行I的过程中访问操作数时发生缺页异常，此时页故障线性地址为0x804d000，保存在CR2控制寄存器中.

(6) 虚页号为高20位 0x80049 (0000 1000 0000 0100 1001)；

虚页号中高10位(0000 1000 00)为页目录索引，低10位 (00 0100 1001)为页表索引；

第一次执行指令I时，P=1，R/W=0，U/S=1，A=1，D=0；

(7) 不会，因为指令I不在一个页面起始处，当执行I之前的指令发生了TLB缺失，会把所在页的页表项装入TLB，但在取操作数a[0]时可能发生，因为a[0]所在页第一次被访问时对应页表项可能不在TLB中.

高18位为TLB标记，低两位为TLB组索引；

指令I线性地址为0x8049c08，MMU将TLB标记(02012H)与第1组所有标记相比，找到一个相等且有效的页表项在TLB中，从而命中，取出主存地址为028B0C08H

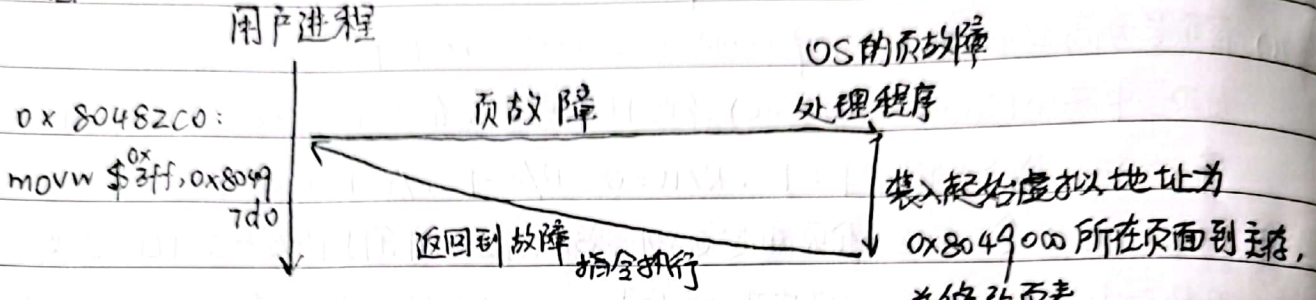
(8) cache共有 8KB/32B = $2^{13}/2^{5}$ = 256行，组数为 256/2 = $128 = 2^{7}$，主存地址划分为：高20位标记，中间7位为组索引，低5位为块内偏移地址。指令I的线性地址0x8049c08，其中低12位为页内偏移量，组索引为1100000，块内地址为01000。因为指令I不在一个主存块的起始位置故不会发生cache缺失。指令I所在主存块应映射到cache的第96组(1100000)中.

(9) 占页面个数 8000B/4KB ≈ 2个，页虚号为0000 1000 0000 0100 1101 和0000 1000 0000 0100 0110，a[1200]在后一个页中.

ICS第七章作业

4.(1). 第一行指令的线性地址为0x8048c0（在Linux初始化时所有段基址设为0，故逻辑地址就是线性地址），页大小为4KB故该指令不是页面起始地址，会将七条指令同时装入内存，故不会发生缺页异常；

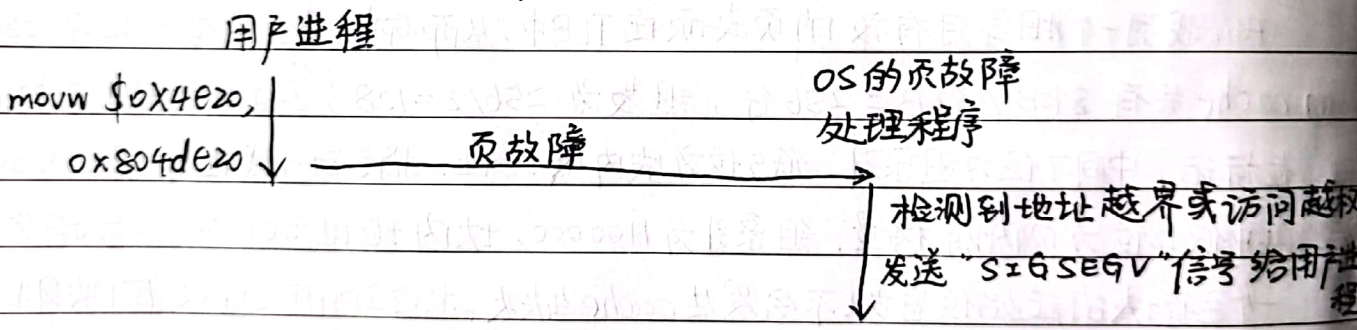
(2) 执行第第一条指令过程中，会在数据访问时发生缺页，但是可恢复的故障，处理过程如图：



执行第二行指令过程中，会在数据访问时发生缺页，同上，可恢复。

执行第六行指令过程中不会发生缺页。

执行第七页行指令会发生页故障，且无法恢复。过程如图：



(3) 会发生"整除0"故障（k默认初值为0），且不能恢复。

5. (1) 用户态，此时执行的用户程序的功能；下个周期变为内核态；

(2) 属于，通过系统门激活异常程处理程序，中断类型号为128；P=1，DPL=3，TYPE=1111B，段描述符中的内容如下：

段	基地址	限界	G	S	TYPE	DPL	D	P
用户代码段	0x0000 0000	0xFFFFF	1	1	10	3	1	1
用户数据段	0x0000 0000	0xFFFFF	1	1	2	3	1	1
内核代码段	0x0000 0000	0xFFFFF	1	1	10	0	1	1 ✓
内核数据段	0x0000 0000	0xFFFFF	1	1	2	0	1	1

取出的是内核代码段，

(3) ①确定中断类型号128(0x80)，从IDTR指向的IDT中取出第128个表项，其中P=1，DPL=3，TYPE=1111B，段选择符0x60，指向GDT中内核代码段描述符。

②根据段描述符，得到DPL=0，基地址为0，将CPL与DPL比较，在Linux内核代码段中DPL总为0，因此不会有CPL<DPL情况。

执行第5行指令时处于用户态，需切换至内核态：

1)读TR寄存器，访问TSS段；

2)将TSS段中保存的段选择符和栈指针分别装入寄存器SS和ESP中；

③将第5行指令后一条指令的逻辑地址写入CS:EIP，在当前内核栈中保存EFLAGS、CS等寄存器内容；

④将IDT的段选择符(0x08)装入CS，将IDT的偏移地址装入EIP。

这样，下一个时钟执行系统调用处理程序system_call的第一条指令，在内核完成系统调用服务后，执行最后一条指令iret，以回到原指令下一条指令。