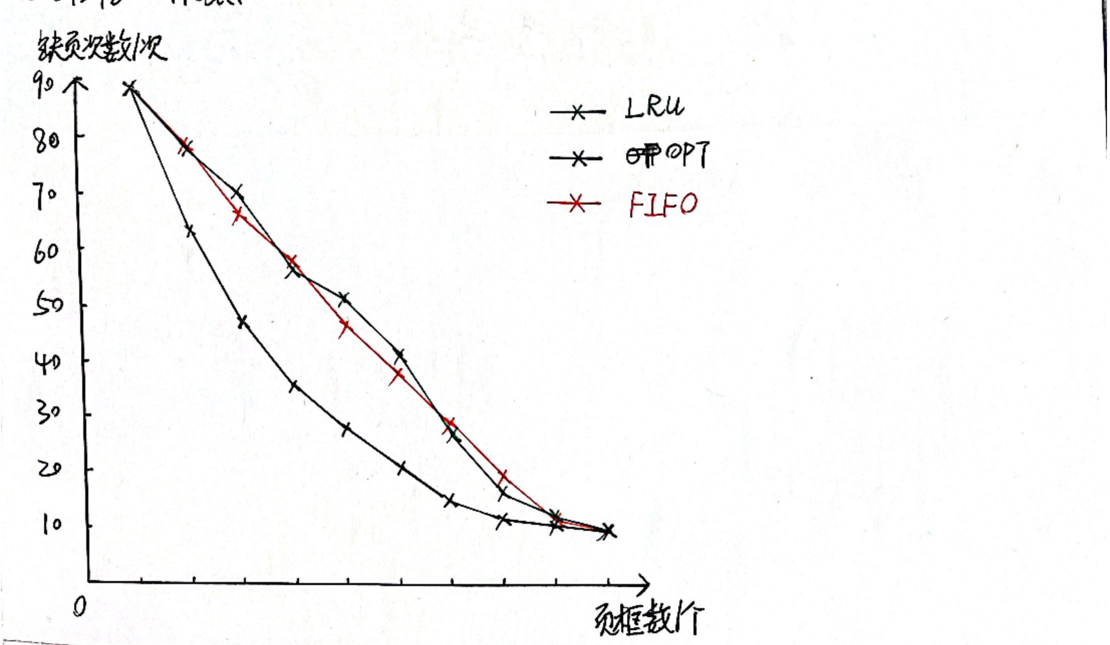
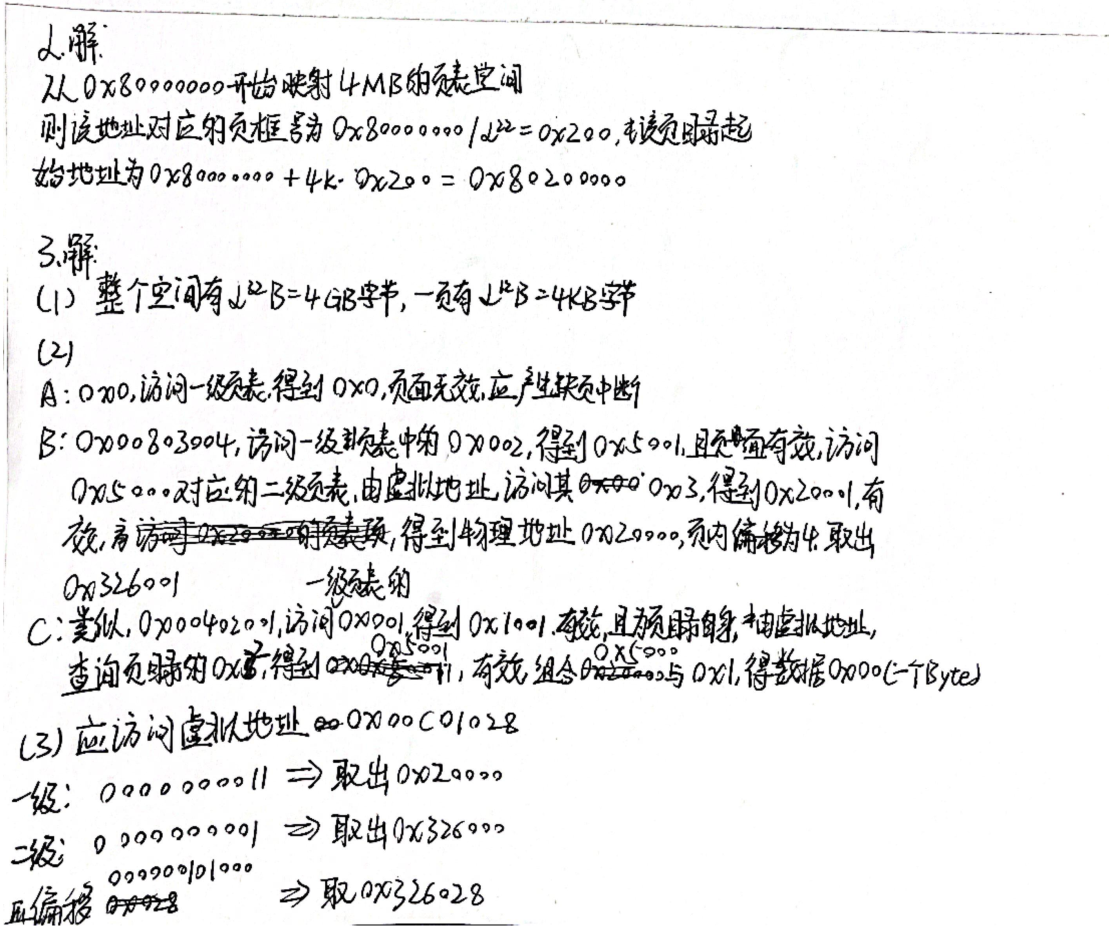
```
#include <stdio.h>

int visit[100] = {0, 9, 8, 4, 4, 3, 6, 5, 1, 5, 0, 2, 1, 1, 1, 1, 8, 8, 5,
                  3, 9, 8, 9, 9, 6, 1, 8, 4, 6, 4, 3, 7, 1, 3, 2, 9, 8, 6, 2, 9, 2, 7, 2, 7, 8, 4, 2, 3, 0, 1,
9, 4,
                  7, 1, 5, 9, 1, 7, 3, 4, 3, 7, 1, 0, 3, 5, 9, 9, 4, 9, 6, 1, 7, 5, 9, 4, 9, 7, 3, 6, 7, 7, 4,
5, 3, 5,
                  3, 1, 5, 6, 1,
                  1, 9, 6, 6, 4, 0, 9, 4, 3};

int FIFO(int frames) {
    int miss_times = 0, flag, i, j, max_cnt, max_pos;
    int tlb[2][100];
    for (i = 0; i < 100; i++) {
        tlb[0][i] = -1;
        tlb[1][i] = -1;
    }
    for (i = 0; i < 100; i++) {
        for (j = 0, flag = 0; j < frames; j++) {
            if (visit[i] == tlb[0][j]) {
                flag = 1;
                break;
            }
        }
        if (flag == 0) {
            for (j = 0, max_cnt = -1, max_pos = 0; j < frames; j++) {
                if (tlb[1][j] == -1) {
                    max_pos = j;
                    break;
                }
                if (tlb[1][j] > max_cnt) {
                    max_cnt = tlb[1][j];
                    max_pos = j;
                }
            }
            tlb[0][max_pos] = visit[i];
            tlb[1][max_pos] = 0;
            miss_times++;
        }
    }
    return miss_times;
}

int LRU(int frames) {
    int miss_times = 0, flag, i, j, max_cnt, max_pos;
    int tlb[2][100];
    for (i = 0; i < 100; i++) {
        tlb[0][i] = -1;
        tlb[1][i] = -1;
    }
    for (i = 0; i < 100; i++) {
        for (j = 0, flag = 0; j < frames; j++) {
            if (visit[i] == tlb[0][j]) {
                flag = 1;
                tlb[1][j] = 0;
                break;
            }
        }
        if (flag == 0) {
            for (j = 0, max_cnt = -1, max_pos = 0; j < frames; j++) {
                if (tlb[1][j] == -1) {
                    max_pos = j;
                    break;
                }
                if (tlb[1][j] > max_cnt) {
                    max_cnt = tlb[1][j];
                    max_pos = j;
                }
            }
            tlb[0][max_pos] = visit[i];
            tlb[1][max_pos] = 0;
            miss_times++;
        }
        for (j = 0; j < frames; j++) {
            if (tlb[1][j] != -1) {
                tlb[1][j]++;
            }
        }
    }
    return miss_times;
}

int main() {
    for (int i = 1; i <= 10; i++) {
        printf("LRU : Miss %d times when frame is %d\n", LRU(i), i);
    }
    for (int i = 1; i <= 10; i++) {
        printf("FIFO : Miss %d times when frame is %d\n", FIFO(i), i);
    }
    return 0;
}
```

OPT 算法不知为何我未能实现，故而借鉴了他人的算法，就不当做作业提交了。

20373980 林子杰





2.解:

从 0x80000000 开始映射 4MB 的页表空间

则该地址对应的页框号为 0x80000000/2^{22}=0x200, 故该页目录起始地址为 0x80000000+4k·0x200=0x80200000

3.解:

(1) 整个空间有 2^{32}B=4GB字节, 一页有 2^{12}B=4KB字节

(2)

A: 0x0, 访问一级页表, 得到 0x0, 页面无效, 应产生缺页中断

B: 0x00803004, 访问一级页表中的 0x002, 得到 0x5001, 且页面有效, 访问 0x5000 对应的二级页表, 由虚拟地址, 访问其 ~~0x00~~ 0x03, 得到 0x2001, 有效, 再~~访问 0x2000 的页表项~~, 得到物理地址 0x20000, 页内偏移为4. 取出 0x326001

一级页表的

C: 类似, 0x00402001, 访问 0x001, 得到 0x1001, 有效, 且为页目录自身, 由虚拟地址, 查询页目录的 0x2, 得到 ~~0x00~~ 0x5001, 有效, 组合 ~~0x2000~~ 0x5000 与 0x1, 得数据 0x00 (一个Byte)

(3) 应访问虚拟地址 ~~00~~ 0x00C01028

一级: 0000000011 ⇒ 取出 0x20000

二级: 0000000001 ⇒ 取出 0x326000

页偏移 000000101000 ~~0x028~~ ⇒ 取 0x326028