

电子科技大学

计算机专业类课程

实验报告

课程名称：操作系统

学 院：计算机科学与工程

专 业：计算机科学与技术

学生姓名：韩会彬

学 号：2013060105004

指导教师：薛瑞尼

日 期： 2016 年 06 月 05 日

电子科技大学

实验报告

实验三

一、实验名称：页表逻辑地址到物理地址转换

二、实验学时：4

三、实验内容和目的：

编程实现页表结构中逻辑地址到物理地址的转换。

四、实验原理：

用户空间划分：

用户程序按逻辑页划分成大小相等的部分，称为页（虚页）

从 0 开始编制页号，页内地址相对于 0 编址。

逻辑空间划分由系统自动完成的，对用户透明。

一般页大小为 2 的整数次幂，因此，地址的高位部分为页号，低位部分为页内地址。

内存空间划分：

按页的大小划分为大小相等的区域，称为内存块（物理页面，页框、实页）

从 0 开始编号

内存分配：

以页为单位进行分配

五、实验步骤和结果：

实验代码：

```
#include<stdio.h>
#include<math.h>
#include<string.h>

int twoToten(char bin[33]);

int main() {

    int R = 0, P = 0, S = 0;
```

```

char hex[17] = {};          //十六进制物理地址
char binx[65] = {};         //二进制逻辑地址
char binf[65] = {};         //二进制物理地址
char bin10[33] = {};        //二进制页内偏移
char hexa[17] = { };        //十六进制逻辑地址
for (int hi = 0; hi < 16; hi++){
    hex[hi] = '0';
    hexa[hi] = '0';
}
for (int bi = 0; bi < 64; bi++){
    binx[bi] = '0';
    binf[bi] = '0';
}

printf("enter address:");
gets_s(hexa);
printf("enter size of note 2^R B,R=:");
scanf_s("%d", &R);
printf("enter size of page 2^P KB,P=");
scanf_s("%d", &P);
P = P + 10;
printf("enter steps of page,S=:");
scanf_s("%d", &S);

/*ye biao*/
int mypage[10001] = {};
int mp = 0;
while (mp < 9999){

    for (int fm = 0; fm < 128; fm++){
        mypage[mp] = fm;
        mp = mp + 1;
        //printf("%d\t%d\n", mp, fm);
        if (mp == 10000){

            break;
        }
    }
}

/*16 to 2*/
int length = 0;

```

```

length = strlen(hexa);

for (int i = 0; i < length; i++){
    char ch = hexa[length - 1 - i];
    switch (ch)
    {
        case '0':
            binx[63 - i * 4] = '0';
            binx[63 - 1 - i * 4] = '0';
            binx[63 - 2 - i * 4] = '0';
            binx[63 - 3 - i * 4] = '0';
            break;
        case '1':
            binx[63 - i * 4] = '1';
            binx[63 - 1 - i * 4] = '0';
            binx[63 - 2 - i * 4] = '0';
            binx[63 - 3 - i * 4] = '0';
            break;
        case '2':
            binx[63 - i * 4] = '0';
            binx[63 - 1 - i * 4] = '1';
            binx[63 - 2 - i * 4] = '0';
            binx[63 - 3 - i * 4] = '0';
            break;
        case '3':
            binx[63 - i * 4] = '1';
            binx[63 - 1 - i * 4] = '1';
            binx[63 - 2 - i * 4] = '0';
            binx[63 - 3 - i * 4] = '0';
            break;
        case '4':
            binx[63 - i * 4] = '0';
            binx[63 - 1 - i * 4] = '0';
            binx[63 - 2 - i * 4] = '1';
            binx[63 - 3 - i * 4] = '0';
            break;
        case '5':
            binx[63 - i * 4] = '1';
            binx[63 - 1 - i * 4] = '0';
            binx[63 - 2 - i * 4] = '1';
            binx[63 - 3 - i * 4] = '0';
            break;
        case '6':

```

```

        binx[63 - i * 4] = '0';
        binx[63 - 1 - i * 4] = '1';
        binx[63 - 2 - i * 4] = '1';
        binx[63 - 3 - i * 4] = '0';
        break;
case '7':
        binx[63 - i * 4] = '1';
        binx[63 - 1 - i * 4] = '1';
        binx[63 - 2 - i * 4] = '1';
        binx[63 - 3 - i * 4] = '0';
        break;
case '8':
        binx[63 - i * 4] = '0';
        binx[63 - 1 - i * 4] = '0';
        binx[63 - 2 - i * 4] = '0';
        binx[63 - 3 - i * 4] = '1';
        break;
case '9':
        binx[63 - i * 4] = '1';
        binx[63 - 1 - i * 4] = '0';
        binx[63 - 2 - i * 4] = '0';
        binx[63 - 3 - i * 4] = '1';
        break;
case 'a':
case 'A':
        binx[63 - i * 4] = '0';
        binx[63 - 1 - i * 4] = '1';
        binx[63 - 2 - i * 4] = '0';
        binx[63 - 3 - i * 4] = '1';
        break;
case 'b':
case 'B':
        binx[63 - i * 4] = '1';
        binx[63 - 1 - i * 4] = '1';
        binx[63 - 2 - i * 4] = '0';
        binx[63 - 3 - i * 4] = '1';
        break;
case 'c':
case 'C':
        binx[63 - i * 4] = '0';
        binx[63 - 1 - i * 4] = '0';
        binx[63 - 2 - i * 4] = '1';
        binx[63 - 3 - i * 4] = '1';
        break;

```

```

        case 'd':
        case 'D':
            binx[63 - i * 4] = '1';
            binx[63 - 1 - i * 4] = '0';
            binx[63 - 2 - i * 4] = '1';
            binx[63 - 3 - i * 4] = '1';
            break;
        case 'e':
        case 'E':
            binx[63 - i * 4] = '0';
            binx[63 - 1 - i * 4] = '1';
            binx[63 - 2 - i * 4] = '1';
            binx[63 - 3 - i * 4] = '1';
            break;
        case 'f':
        case 'F':
            binx[63 - i * 4] = '1';
            binx[63 - 1 - i * 4] = '1';
            binx[63 - 2 - i * 4] = '1';
            binx[63 - 3 - i * 4] = '1';
            break;

        default:
            break;
    }
}

/*qu diao qian mian de 0*/

int p = 0;
int b = 64;

int k = 0;
while (p == 0) {

    //printf("p=%d,k=%d\n", p, k);
    if (binx[k] == '0' && binx[k + 1] == '0' && binx[k + 2] == '0' && binx[k + 3] ==
'0') {

        k = k + 4;
        b = b - 4;
    }
    else {

```

```

        p = 1;
    }
    if (k == 64) p = 1;
}

/*yebiao fenji*/
int num = 0, judge=1;
num = (b - P) / S-1;
while (judge != 0){          //计算每级页表位数
    num = num + 1;
    judge = b - (num*S + P);
}

int page = 0;
int table = 0;
for (int k = 0; k < num; k++){
    bin10[k] = binx[63 - P - num*S + num - k];
}
table = twoToten(bin10);          //首页偏移地址
page = mypage[table];            //次级页号
for (int ss = 1; ss < S; ss++){

    for (int k = 0; k < num; k++){
        bin10[k] = binx[63 - P - num*(S-ss-1) - k];
    }
    table = twoToten(bin10);
    page = mypage[(P / R)*ss+table];    //获取次级页号
}
for (int m = 0; m < P; m++){
    binf[63 - m] = binx[63 - m];
}
int ttt = 1, la=0, y=0;
while (ttt != 0){
    y = ttt % 2;
    if (y == 1)
        binf[63 - P - la] = '1';
    else if (y == 0)
        binf[63 - P - la] = '0';
    ttt = ttt / 2;
}

```

```

int q = 0;
int kk = 0;
b = 64;
while (q == 0){

    if (binf[kk] == '0' && binf[kk + 1] == '0' && binf[kk + 2] == '0' && binf[kk + 3] ==
'0'){

        kk = kk + 4;
        b = b - 4;
    }
    else{

        q = 1;
    }
    if (kk == 64) q = 1;
}
/*2 to 16*/
int he = 15;
for (int j = 0; j < b;){

    if (binf[63 - j] == '0' && binf[62 - j] == '0' && binf[61 - j] == '0' && binf[60 -
j] == '0'){
        hex[he] = '0';
    }
    else if (binf[63 - j] == '1' && binf[62 - j] == '0' && binf[61 - j] ==
'0' && binf[60 - j] == '0'){
        hex[he] = '1';
    }
    else if (binf[63 - j] == '0' && binf[62 - j] == '1' && binf[61 - j] ==
'0' && binf[60 - j] == '0'){
        hex[he] = '2';
    }
    else if (binf[63 - j] == '1' && binf[62 - j] == '1' && binf[61 - j] ==
'0' && binf[60 - j] == '0'){
        hex[he] = '3';
    }
    else if (binf[63 - j] == '0' && binf[62 - j] == '0' && binf[61 - j] ==
'1' && binf[60 - j] == '0'){
        hex[he] = '4';
    }
    else if (binf[63 - j] == '1' && binf[62 - j] == '0' && binf[61 - j] ==
'1' && binf[60 - j] == '0'){
        hex[he] = '5';
    }
}

```



```

        else if (binf[63 - j] == '0' && binf[62 - j] == '1' && binf[61 - j] ==
'1' && binf[60 - j] == '0') {
            hex[he] = '6';
        }
        else if (binf[63 - j] == '1' && binf[62 - j] == '1' && binf[61 - j] ==
'1' && binf[60 - j] == '0') {
            hex[he] = '7';
        }
        else if (binf[63 - j] == '0' && binf[62 - j] == '0' && binf[61 - j] ==
'0' && binf[60 - j] == '1') {
            hex[he] = '8';
        }
        else if (binf[63 - j] == '1' && binf[62 - j] == '0' && binf[61 - j] ==
'0' && binf[60 - j] == '1') {
            hex[he] = '9';
        }
        else if (binf[63 - j] == '0' && binf[62 - j] == '1' && binf[61 - j] ==
'0' && binf[60 - j] == '1') {
            hex[he] = 'A';
        }
        else if (binf[63 - j] == '1' && binf[62 - j] == '1' && binf[61 - j] ==
'0' && binf[60 - j] == '1') {
            hex[he] = 'B';
        }
        else if (binf[63 - j] == '0' && binf[62 - j] == '0' && binf[61 - j] ==
'1' && binf[60 - j] == '1') {
            hex[he] = 'C';
        }
        else if (binf[63 - j] == '1' && binf[62 - j] == '0' && binf[61 - j] ==
'1' && binf[60 - j] == '1') {
            hex[he] = 'D';
        }
        else if (binf[63 - j] == '0' && binf[62 - j] == '1' && binf[61 - j] ==
'1' && binf[60 - j] == '1') {
            hex[he] = 'E';
        }
        else if (binf[63 - j] == '1' && binf[62 - j] == '1' && binf[61 - j] ==
'1' && binf[60 - j] == '1') {
            hex[he] = 'F';
        }
        he = he - 1;
        j = j + 4;
    }

```

```

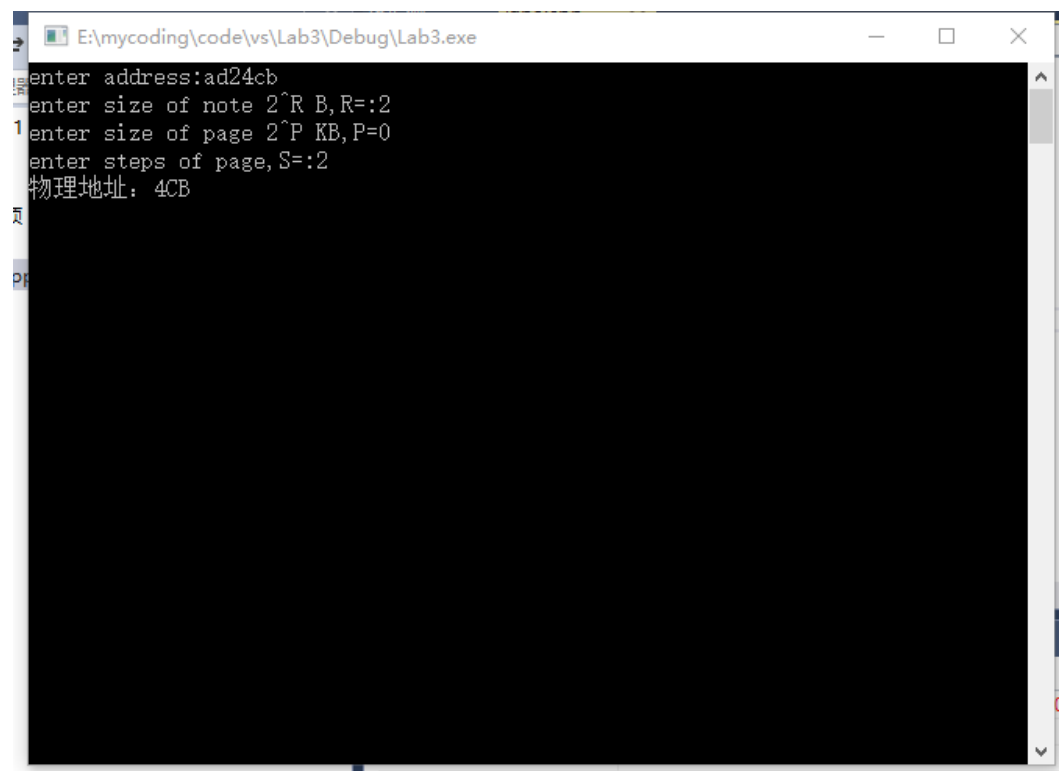
//dayin wuli dizhi
int pr = 0;
while (hex[pr] == '0') {
    pr = pr + 1;
}
printf("物理地址: ");
for (pr ; pr < 16; pr++) {
    printf("%c", hex[pr]);
}
printf("over");
return 0;
}

int twoToten( char bin[33]) {
    int page = 0;
    int n = 0;
    int k = 1;
    for (int i = 0; i < 32; i++) {
        if (bin[i] == '1') {
            n = i;
            for (int j = 0; j < n; j++) {
                k = k * 2;
            }
            page = page + k;
        }
    }

    return page;
}

```

实验结果:



```
E:\mycoding\code\vs\Lab3\Debug\Lab3.exe
enter address:ad24cb
enter size of note 2^R B,R=:2
1 enter size of page 2^P KB,P=0
enter steps of page,S=:2
物理地址: 4CB
```

六、实验结论及心得:

通过本次实验我再次熟悉了页表的结构和原理,进一步了解了页表的作用,提升了自己的编程能力。