Q5.用C语言描述包含TLB的页式存储管理过程

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#define len sizeof(struct page)

struct pagetable

{

int pagenum;

int blocknum;

char state;

}num[1024];

struct page

{

int pagenum;

int blocknum;

int data[1024];

struct page \*next;

}\*head=NULL,\*tail=NULL;

int TLB[2]={0,0};

void

print(int a)

{

int i;

printf("\n\*\*\*\*\*pagenum blocknum state\n");

for(i=0;i<=a;i++)

{

num[i].pagenum=i;

printf("%8d%8d%8c",num[i].pagenum,num[i].blocknum,num[i].state);

printf("\n");

}

printf("\n");

}

int main( void )

{

int i,j,a,b,c,d,e;

char s;

char ch;

struct page \*p;

/\* #####Initialize pagetable!!!#####\*/

printf("\*\*\*\*\*Please input the size of process:");

scanf("%d",&a);

b=a/1024;

for(i=0;i<=b;i++)

{

num[i].pagenum=i;

num[i].blocknum=-1;

num[i].state='F';

}

print(b);

/\*#####Start to request the page!#####\*/

do

{

ch=getchar(); //getch();

j=0;

printf("\*\*\*\*\*please input the adress:");

scanf("%d",&c);

if(c>a)

printf("\n\*\*\*\*\*The adress is slop over!\n");

else

{

d=c/1024;

e=c%1024;

s=num[d].state;

if(s=='T')

{

print(b);

printf("The target in this page is coded as %d\n",e);

}

else

{

for(i=0;i<=b;i++)

{

// num[i].pagenum=i;

num[i].blocknum=-1;

num[i].state='F';

}

num[d].blocknum=0;

num[d].state='T';

TLB[0] = d;

if(d + 1 <= b)

{

num[d+1].blocknum=1;

num[d+1].state='T';

TLB[1] = d + 1;

}

print(b);

printf("The target in this page is coded as %d\n",e);

}

}

}while(c>=0);

return 0;

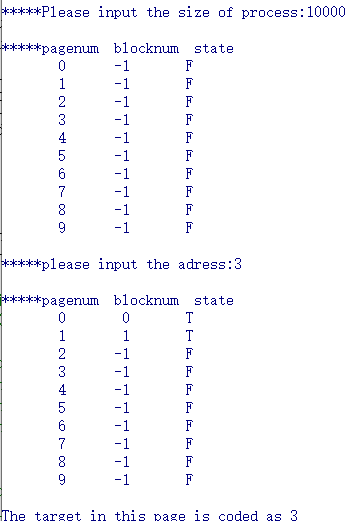
}

该c语言代码实现了页大小为1KB，总页数1024页的页表，并且TLB的大小为2页。

运行该程序时，首先输入进程所需内存空间的大小，随后会打印分配的各页面的情况。

之后可以输入访问的地址：1）如果访问的地址已经在TLB中，则TLB不做修改，并打印该地址在页面上的编码。2）如果访问的地址不在TLB中，则将TLB的状态置为空，所有页表的加载状态置为F，随后将要访问地址所在的页表的状态修改为T，加载到TLB中（如果有下一页的话，也将下一页加载到TLB中）。

截图提供了测试用例：



初始化空间为10000

访问虚拟地址3

访问虚拟地址2222