void addStackNode(PageTable &pageTable, StackList &stackList, int pageNumber)

{

printf("OS命令CPU从外存读缺页......启动I/O硬件......\n");

if (stackList.length > 0)

{

StackNode newStackNode = new (stackNode);

StackNode stackNode = stackList.stackNode;

if (stackNode)

{

newStackNode->pageNumber = pageNumber;

newStackNode->before = stackNode->before;

newStackNode->next = stackNode;

stackNode->before = newStackNode;

stackList.stackNode = newStackNode;

}

else

{

newStackNode->pageNumber = pageNumber;

newStackNode->before = newStackNode;

newStackNode->next = NULL;

stackList.stackNode = newStackNode;

}

stackList.length--;

pageTable.pageTableItemArray[pageNumber]->P = 1;

pageTable.pageTableItemArray[pageNumber]->blockNumber = rand() % (1024 \* 8) + 1;

}

else

{

printf("内存已满！\n");

}

}

// 栈中已有结点页更新使用程度

void updateStackNode(PageTable &pageTable, StackList &stackList, int pageNumber)

{

StackNode stackNode = stackList.stackNode;

while (stackNode != NULL)

{

if (stackNode->pageNumber == pageNumber && stackList.stackNode != stackNode)

{

stackNode->before->next = stackNode->next;

if (stackNode->next != NULL)

{

stackNode->next->before = stackNode->before;

}

stackNode->before = stackList.stackNode->before;

stackNode->next = stackList.stackNode;

stackList.stackNode->before = stackNode;

stackList.stackNode = stackNode;

return;

}

stackNode = stackNode->next;

}

}

void replaceStackNode(PageTable &pageTable, StackList &stackList, int pageNumber)

{

StackNode tailStackNode = stackList.stackNode->before;

tailStackNode->before->next = NULL;

stackList.stackNode->before = tailStackNode->before;

// 换出淘汰页，释放内存

delete (tailStackNode);

stackList.length++;

pageTable.pageTableItemArray[pageNumber]->P = 0;

// 换入当前页

addStackNode(pageTable, stackList, pageNumber);

}

// 最近最久未使用算法（LRU）

void LRU(PageTable &pageTable, QuickPageTable &quickPageTable, StackList &stackList, int pageNumber)

{

// 判断该页是否在内存中

if (pageTable.pageTableItemArray[pageNumber]->P == 1)

{

// 该页在内存中

printf("当前页已在内存中！更新栈表......\n");

updateStackNode(pageTable, stackList, pageNumber);

}

else if (pageTable.pageTableItemArray[pageNumber]->P == 0)

{

// 该页不在内存中

printf("当前页不在在内存中！保留CPU现场......从外存中找到缺页......\n");

if (stackList.length > 0)

{

// 内存未满

printf("内存未满！直接为当前页分配内存......\n");

addStackNode(pageTable, stackList, pageNumber);

}

else

{

// 内存已满

printf("内存已满！置换页中......\n");

replaceStackNode(pageTable, stackList, pageNumber);

}

}

else

{

printf("系统错误！\n");

}

}