

上海大学 2003 年攻读硕士学位研究生 入学考试试题

招生专业: 计算机系统结构

考试科目: 计算机组成原理与数据结构

计算机软件与理论

计算机应用技术

第一部分: 计算机组成原理

一、填空题: (30 分)

1. 若数 X 在计算机中用补码表示为 1.0000000 , 该数 X 的真值为 1。
2. 任何进位计数制都包含两个基本要素, 即 基数 和 位权, 八进制数中第 N 位上的权为 8^{N-1} 。
3. 运算器一次能运算处理器的数据的二进制位数称为 字长。
4. 根据操作数所在位置, 指出其寻址方式, 操作数在寄存器中, 为 寄存器 寻址方式; 操作数地址在寄存器中称为 寄存器寻址 寻址方式, 操作数在指令中称为 指令 寻址方式, 操作数地址 (主存) 在指令中为 指令寄存器 寻址方式, 操作数的地址为某一寄存器的内容与位移量之和则可以是 基址寻址、变址、基址变址 寻址方式。
5. 控制器的主要功能是 指令取指、指令译码、指令执行 及 I/O 控制和对异常情况以及某些请求进行处理。
6. 根据目前常用的存储介质可以将存储器分为 半导体存储器、磁存储器 和光存储器三种。
7. 主存储器和 CPU 之间增加高速缓冲存储器的目的是 提高存取速度。
8. 已知二进制数 $N_0N_1N_2N_3$, 若奇偶检验值用 P 表示, 则奇校验值为 $\overline{P_{奇}}$, 偶校验值为 $P_{偶}$ 。奇偶校验只能检测 单比特错误, 无法检测 双比特错误。
9. 根据总线控制部件所处位置不同, 总线的控制方式可分为如下两种: 总线控制逻辑基本集中在一起的, 称为 集中式 总线控制; 总线控制逻辑分散在总线上各部件的, 称为 分布式 总线控制。

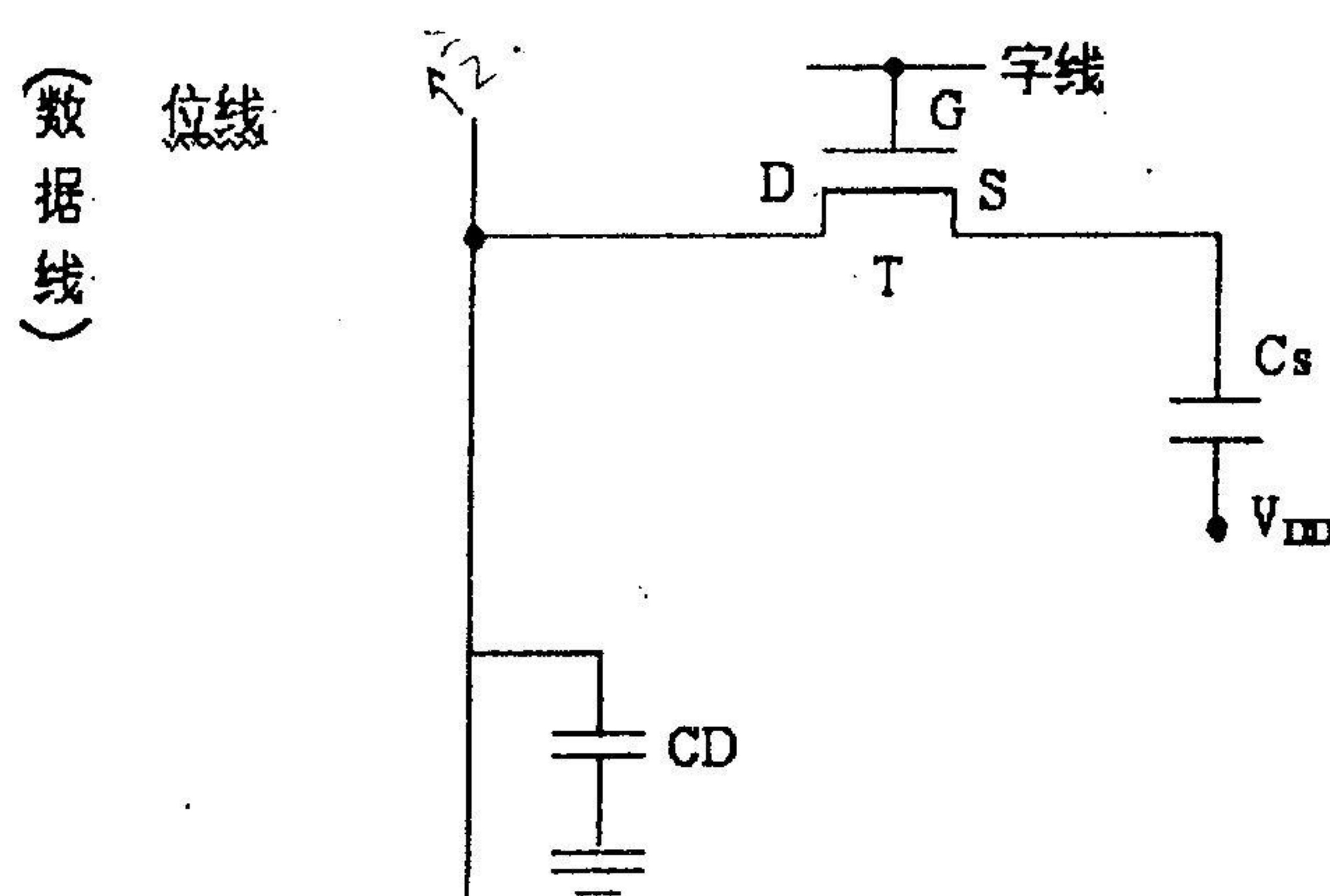
第2页(共7页)

为分争式总线控制。按总线上两种部件通讯时采用的同步方式不同,总线的控制方法又可分为同步控制与异步控制,前者是同步通讯,后者是异步通讯。

10. CPU在总线时刻采样中断请求信号(在开中断情况下),而在数据总线时刻去采样DMA的总线请求信号。

二、综合题:(45分)

1. (8分) 按图分析单管 DRAM 记忆单元的工作原理。



2. (8分) 设写入代码为 100110101, 试画出 NRZ、MFM (改进调频制) 的写电流波形和读出信号波形。
3. (8分) 若被校验数 $x=1100B$, 选择的生成的多项式为 x^3+x+1 , 试计算 CRC 码。
4. 某计算机有四级中断, 其响应优先级从高到低依次为 1, 2, 3, 4。假定在运行用户程序时同时出现 3 级和 4 级中断请求, 在处理 4 级中断时, 又出现了 1 级中断请求。在处理 1 级中断时又出现 2 级中断请求。
- (1) (3分) 试问以上各个中断处理完成的顺序。
- (2) (5分) 并画出处理该多重中断的示意图。
- (3) (3分) 现要求中断处理次序改为 3, 1, 2, 4, 根据下示格式, 写出各中断源的屏蔽字。

中断源	1, 2, 3, 4 屏蔽字
1	
2	
3	
4	

5. (10 分) 某 CPU 的主频为 8MHz, 若已知每个机器周期平均包含 4 个主振周期, 该机的平均指令执行速度为 0.8MIPS, 试求该机的平均指令周期每个指令周期含有几个机器周期? 若改用主频周期为 $0.4\mu\text{s}$ CPU 芯片, 则计算机的平均指令执行速度为多少 MIPS? 若要得到平均每秒 40 万次的指令执行速度, 则应采用主频为多少的 CPU 芯片?

第二部分: 数据结构

一. 填写下列各题 (6 分)

- 深度为 h 的完全二叉树至少有 h 个结点, 至多有 $2^h - 1$ 个结点。
- 已知广义表 $LS = (a, (b, c, d), e)$, 运用 head 和 tail 函数取出 LS 中单元素 b 的运算是: $\text{head}(\text{head}(\text{tail}(LS)))$ 。
- 下列程序段的复杂度为 \sqrt{n} 。

```

i=0; s=0;
while (s<n)
{
    i++;
    s=s+i
}

```

二. 解答下列各题 (14 分)

- (8 分) 设有下列一段正文: CAST CATS SAT AT A TASA, 其中所用的字符集

第4页(共7页)

$D = \{C, A, S, T\}$ 。请用哈夫曼算法设计一套 D 中字符的二进制编码, 使上述正文的二进制内部表示最短, 并且采用这种编码表示上文时, 字符间不必用分隔符就能识别。

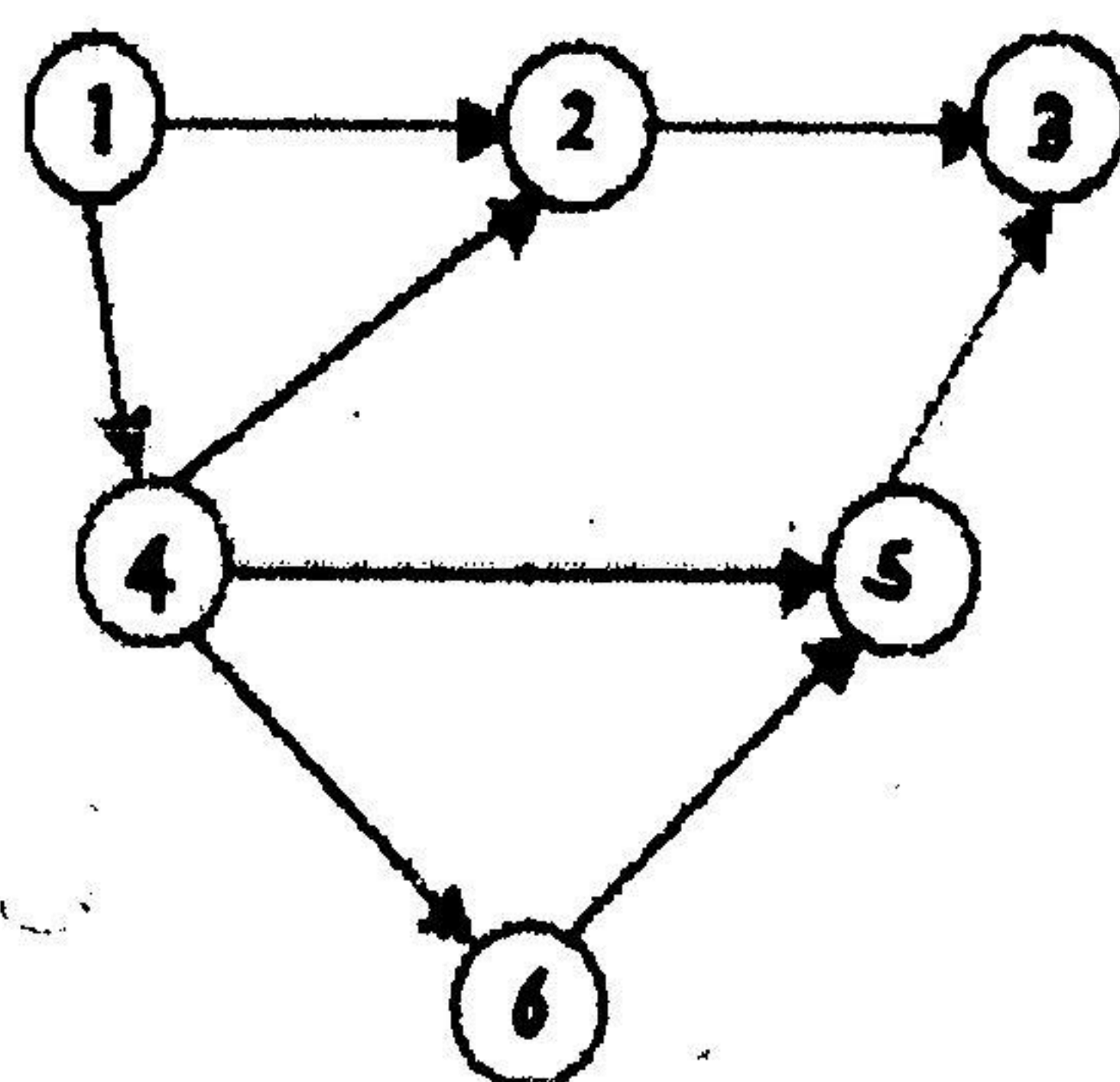
要求:

- 1) 画出哈夫曼树;
- 2) 计算该哈夫曼树的加权路径长度 WPL;
- 3) 给出 D 中字符的编码表;
- 4) 将上述正文译成二进制编码。

2. (6分) 已知一个有向图如下:

(1) 请给出其邻接表表示;

(2) 请给出其拓扑有序序列。



三. 算法填空, 完善下列程序, (32分)

1. (8分) 下面是一个求两个集合 A 和 B 之交 $C=A \cap B$ 的程序。若 C 是 A 和 B 之交, 则当且仅当 e 是 A 中的一个元素, 同时也是 B 中的一个元素时, e 才是 C 中的一个元素。集合用有序链表实现, 空链表表示空集, 非空有序链表表示非空集合。初始时, A 、 B 集合中的元素按非递减次序在链表中排列。集合 C 初始时空, 操作完成后, A 、 B 保持不变, C 中元素按非递减次序在链表中排列。下面的函数 $\text{intersection}(A, B)$ 实现集合运算 $A \cap B$, 并返回表示结果集合的链表 C 的链表首结点的地址。在执行 $A \cap B$ 运算之前, 用于表示结果集合的链表首先增加一个附加的表头结点, 以便新结点的添加, 当 $A \cap B$ 运算执行完毕, 再删除并释放表示结果集合的链表的表头结点。

程序

```

typedef struct node { char element;
                      struct node *link;
} NODE;

NODE *intersection(NODE *A, NODE *B)
{ NODE *C, *p;

```


A/B
B

```
C=(NODE *)malloc(sizeof(NODE));
```

A/B
B

```
p=C;
```

A/B
B

```
while ( A < B )
```

A/B
B

```
{ if (A->element==B->element)
```

A/B
B

```
{p->link=(NODE *)malloc(sizeof(NODE));
```

A/B
B

```
p = p->link (B)
```

A/B
B

```
p->element=A->element;
```

A/B
B

```
A=A->link;
```

A/B
B

```
B = B->link (C)
```

A/B
B

```
}
```

A/B
B

```
else
```

A/B
B

```
if (A->element < B->element)
```

A/B
B

```
A=A->link;
```

A/B
B

```
else
```

A/B
B

```
B= B->link;
```

A/B
B

```
(D)
```

A/B
B

```
p=C;
```

A/B
B

```
C=C->link;
```

A/B
B

```
free(p);
```

A/B
B

```
return(C);
```

A/B
B

```
}
```

2. (24 分) 在下面所给的程序中, find_path(t, a)在一般二叉树 t 中, 查找值为 a 的结点, 若该结点在树中, 则输出从根结点到值为 a 的结点之路径上的所有结点值; 若该结点不在树中, 则输出不在树中的信息。查找过程借助于非递归的树遍历来实现。程序中使用一个链接栈, 用于存放查找路径上所有结点的地址, 而链接栈栈顶结点的 top 字段取值为 'L' 或 'R', 若取值为 'L', 则正在对其地址存放于栈顶的结点之左子树进行查找; 若取值为 'R', 则正在对其右子树进行查找。在执行查找后, 删除并释放链接栈的所有结点。reverse_link_list(top)把链表 top 倒转过来。print_link_list(top)输出链表中所指向的树中各相关结点之值。delete_link_list(top)逐个删除并释放 top 中所有结点。

所谓倒转链表, 就是链表中原来的第一个结点, 第二个结点, ……., 最后第二个结点, 最后一个结点, 在倒转之后依次成为链表中的最后一个结点, 最后第二个结点, ……., 第二个结点, 第一个结点。

第6页(共7页)

程序

```
typedef struct node{ char data;
                    struct node *lchild, *rchild;
                    } NODE;
```

```
typedef struct s_node {NODE *addr;
                      char tag;
                      struct s_node *link;
                      } S_NODE;
```

```
S_NODE *reverse_link_list(S_NODE *top)
```

```
{ S_NODE *p=NULL, *q=top, *r;
```

```
  while (q!=NULL)
```

```
  { r=q->link;
```

```
    q->link=p
```

```
    p=q;
```

```
    q = r (A)
```

```
  }
```

```
  return(p);
```

```
}
```

```
void print_link_list(S_NODE *top)
```

```
{ while ( top != NULL ) (B) top != null
```

```
  { print("%c", top->addr->data);
```

```
    top=top->link;
```

```
  }
```

```
}
```

```
void delete_link_list(S_NODE *top)
```

```
{ S_NODE *s;
```

```
  while (top!=NULL)
```

```
  { s=top;
```

```
    top = top->link (C);
```

```
    free(s);
```

```
  }
```

```
}
```

```
void find_path(NODE *t, char a)
```

```
{ S_NODE *top=NULL, *s;
```

```
  while ( top != NULL ) (D) top != null
```

```
  { while (t!=NULL)
```

```
    { if (t->data==a)
```

```
      { print("The path is");
```

```
        top= reverse_link_list(top);
```

```
        print_link_list(top);
```

```
        printf("%c\n", a)
```



```

        delete_link_list(top);
        return;
    }
    s=(S_NODE *)malloc(sizeof(S_NODE));
    s->addr=t;
    s->tag='L';
    s->link=top;
    top=s;
    top = top->link; ;
}
if ( t->data (F) ? a )
{
    t=top->addr;
    if (top->tag=='L')
    {
        top->tag='R';
        top = top->link; ;
    }
    else
    {
        s=top;
        top=top->link;
        free(s);
        return (H) return ;
    }
}
}
printf("%c is not in tree. \n",a);
}

```

四. 算法设计 (23 分)

1. (13 分) 为了便于求二叉树中结点的前趋和后继, 引入线索二叉树。用 Pascal 语言或 C 语言完成如下工作:

- (1) 写出线索链表的数据类型定义
- (2) 写出在后序线索树中求给定结点 p 的后继结点的算法。

2. (10 分) 以有序顺序表表示静态查找表时, 进行查找的方法除折半查找之外, 还有斐波那契查找。斐波那契查找是根据斐波那契数列的特点对表进行分割的。假设表中元素个数 n 比某个斐波那契数 F_u 小 1, 即 $n=F_u-1$ 。请用 Pascal 语言或 C 语言写出斐波那契查找算法。