1998年浙江大学计算机专业课（乙）（含操作系统和计算机组成）考研试题

注意：答案必须写在答题纸上，否则无效

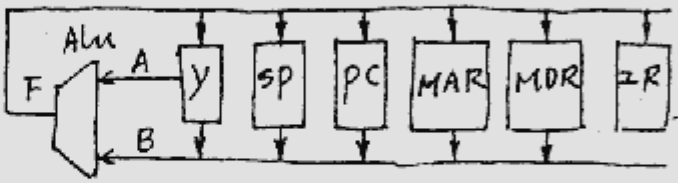
计算机组成部份：

一、某计算机的cpu内部为双总线结构，所有数据传送都通过ALU，ALU具有下列功能，cpu见下图结构。

F=A　　F=B

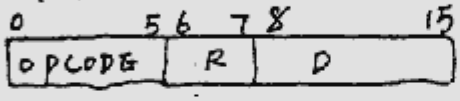
F=A+1　　F=B+1

F=A-1　　F=B-1



写出转子指令(JSR)的取指和执行的微操作序列。JSR指令占两个字，第一个字是操作码，第二个为子程序的起始地址。返回地址保存在存贮器堆栈中，堆栈指针SP始终指向栈顶。图中Y为暂存器、PC为程序计数器，MAR与MDR分别为存贮器地址与数据寄存器、IR为指令寄存器。（10分）

2、已知某机字长16位、主存按字编址，其双操作数指令格式为：



其中OP为操作码，R为通用寄存器地址、试问下列寻址方式下能访问的最大主存区为多少机器字？

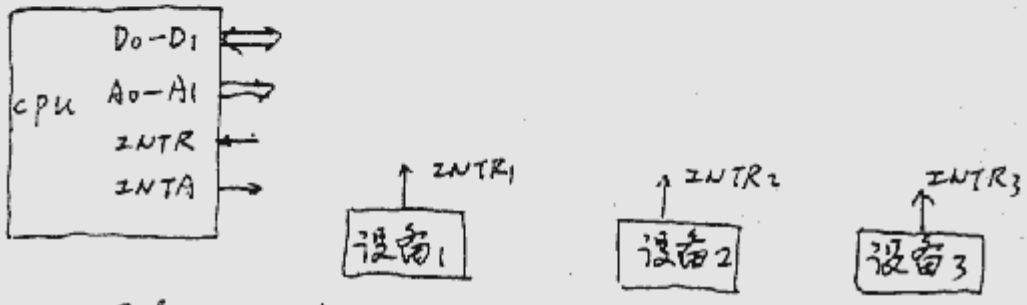
1) D为立即数：_____；2) D为主存直接地址：_____

3) D为主存间接地址_____；4)D为变址的形式地址、变址寄

存器为R1（字长16位）：_____ （5分）

3. 一个做加减法时判断溢出的简单方法是看高位的进位输入与其进位输出是否一致，证明这个判断方法是正确的。（10分）（用图表、布尔表达式、逻辑电路图两者表达）

4. 设某计算机系统配有三个中断源，而CPU只有一个中断请求输入信号（线）INTR，一个中断响应输出信号（线）INTA，如下图所示。



1）用图表示、完成你的处理方案。（10分）

2）为3使接口能以中断方式与CPU完成数据交换，试举例（输入或输出接口）设计接口的功能电路。（5分）

5. 某计算机的虚拟存储系统有40位虚拟地址，32位实际地址；页面数1M（2^{20}）个。假设有效位、保护位、修改位和使用位共用去四位（valid, protection, dirty, used）所有页面都在使用。A. 计算页表大小_____ B. 页面大小_____ C. 画出该虚拟存储系统的虚—实地址转换逻辑（包括虚地址、实地址、页表、页表寄存器）及相互关系。（10分）

操作系统部分

试题1至试题4为选择题，分别从供选择的答案中选出一个唯一正确的，填入“____”。

试题1（3分）：下列选择中，_____不是操作系统关心的主要问题。

A．管理计算机裸机。　　B．设计、提供用户程序与计算机硬件系统的界面

C．管理计算机系统资源　　D．高级程序设计语言的编译器

试题2（5分）：采用_____不会产生内部碎片。

A．分页式存储管理　　B．分段式存储管理

C．固定分区式存储管理　　D．段页式存储管理

试题3（5分）：下列几种关于进程的叙述，_____最不符合操作系统对进程的理解。

A．进程是在多程序并行环境中的完整的程序　　B．进程可以由程序、数据和进程控制块描述

C．线程(THREAD)是一种特殊的进程

D．进程是程序在一个数据集合上运行的过程，它是系统进行资源分配和调度的一个独立单位

试题4（5分）：关于临界区问题的一个算法（假设只有进程 P_0 和 P_1 可能会进入该临界区）如下（i为0或1），该算法_____。

A．不能保证进程互斥进入临界区，且会出现“饥饿”（Starvation）

B．不能保证进程互斥进入临界区，但不会出现“饥饿”

C．保证进程互斥进入临界区，但会出现“饥饿”

D．保证进程互斥进入临界区，不会出现“饥饿”

```
repeat
    retry: if (turn ≠ -1) turn := i;
           if (turn = i) go to retry;
           turn := -1;
           Critical Section（临界区）
           turn := 0;
           remainder Section（其它区域）
until false;
```

试题5（10分）：磁盘系统调度中，采用SCAN（“扫描”）调度算法为任务队列67，65，124，14，122，37，183，98服务。试计算服务结束时，磁头总共移动了几个磁道。假设磁头总在第0道至第199道之间移动；开始服务时，磁头刚从60移到67。

试题6（10分）：某操作系统（假设为“LINUX”）设定一个定时器，每间隔10ms产生一次“时间到”中断，该中断的服务程序采用下述算法：

```
struct task_struct * p = current;   /* 指针 p 指向当前运行进程的进程控制块 PCB */
                                    /* ticks 保存最近两次中断服务期间的中断次数，通常是 1 */
p->counter = p->counter - ticks;    /* counter 记录尚可运行的时间段 */
if (p->counter < 0) { /* 如果 PCB 的 counter 变量小于 0 */
    p->counter = 0;
    need_resched = 1;) /* 全程量 need_resched 置位表示应进行新一轮调度 */
```

" LINUX " 若发现 need_resched 置位，则调用如下的 schedule()，进行新一轮调度：

```
void schedule(void)
{     need_resched = 0;
      prev = current; /* 指针 prev 指向当前运行进程的 PCB */
      cli(); /* move an exhausted process to be last.. */
      if (!prev->counter && prev->policy == SCHED_RR) {
            prev->counter = prev->priority;   /* priority 表示进程的优先权 */
            move_last_runqueue(prev);}   /* 将当前进程撤到就绪进程队列末尾 */
      p = init_task.next_run; /* 指针 p 指向就绪进程队列中第一个 PCB */
      sti();      c = -1000;      next = idle_task;
      while (p != &init_task) {   /* 遍历就绪进程队列中所有 PCB */
            int weight = p->priority;   /* priority 表示进程的优先权 */
            if (weight > c)      c = weight,   next = p;
            p = p->next_run;      /* 下一个进程 */
      }
      switch_to(prev,next);}      /* CPU 从 prev 进程切换到 next 进程 */
```

试运用操作系统进程调度有关原理，分析 " LINUX " 的进程调度策略。

试题7（12分）：临界区(Critical Section)问题要求设计若干进程间的协调策略，以保证进程互斥访问共享数据。例如，如下所示之面包房算法(Bakery Algorithm)就是关于N个进程的临界区问题的著名算法。试论述该算法采用choose数组的必要性，并举例说明。

```
repeat
      choosing[i] := true;
      number[i] := MAX(number[0], number[1], ... ... , number[N-1])+1 ;
      choosing[i] := false;
      for j = 0 to N-1 do
      begin
        while choosing[j] do no-operation ;
        while number[j] ≠ 0 AND (number[j], j ) < (number[i], i) do no-operation ;
      end ;
            Critical   Section （临界区）
      number[i] := 0 ;
            remainder   Section （其它区域）
until   false ;
```