第五章

5.1

多道程序设计：管理单处理器系统中的多个进程，让其交替执行，不是真正意义并行。

多处理技术：管理多处理器中的多个进程，可以重叠执行。

5.3

a

P1	P2
x=x-1; x=x+1; [x=10]	
	x=x-1; [x=9]
if(x!=10) [x=9]	
	x=x+1 [x=10]
printf("x is %d", x) ［x is 10]	

b

P1	P2
x=x-1 [x=9]; LD R0, x [R0=9]; INCR R0 [R0=10]	
	LD R0 x [R0=9]; DEC R0 [R0=8]; STO R0, x [x=8];
STO R0, x [x=8];	
if(x!=10) [x=8] ; printf("x is %d",x) [x is 10]	

5.4

> *参考而已*

$tally \in [2, 50N]$

如何获得最小值？假设一个进程X为整个竞争的“失败者”，在最后写回自增的结果Y覆盖在它之前执行的进程结果。那么进程X在最后一次自增之前肯定会完成

- 先自增49次
- 接着从寄存器中读取值 Z，而有 $Z = Y - 1$，接着自增一次

Z非负，Y必然是一个大于等于1的值，所以自增结果大于等于2，当仅当Z取0的时候有最小值

例如假设以下两个进程情况

- 进程1 载入$tally$，自增1，接着失去处理器未写回内存
- 进程2 载入$tally$，此时内存中$tally$还是0，完成49次自增，接着写回内存$tally = 49$，失去处理器控制权
- 进程1重新获得处理器，将第一步中未写回的值写到内存，此时$tally = 1$
- 进程2重新获得处理器，载入$tally = 1$ 自增到$tally = 2$, 失去处理器未写回内存
- 进程1重新获得处理器，载入$tally = 1$, 运行剩余的49次自增，$tally = 50$，写回内存
- 进程2重新获得处理器，将2写回，存储这个值作为最终结果

考虑上界：刚好全部错开执行，最大自增数为100，则上界为100

5.5 忙等待（一直占用CPU）的效率一定比阻塞等待的效率低吗？请辨析并解释

忙等待是一个进程一直占用CPU的资源，阻塞等待则是等待资源进程的切入切出（简要解释两个概念）在忙等待时，可能会消耗大量的CPU资源在无用指令上，直到满足条件；在进行阻塞等待时，系统将从就绪队列中依次选取进程执行，几乎没有浪费CPU资源（简要资源对比，可选取其他方面不一定只是CPU）如果忙等待一个很快就可以满足的条件，需要的时间小于上下文切换的开销，显然此时忙等待的效率较高（选取一个方面比较）

5.6 程序思考题

反例如下（考虑CPU进程切换）：

步骤	语句	进程	备注
0	-	-	turn=0, blocked[0] = false
1	blocked[1]=true	1	turn=0, blocked[0] = false
2	while(turn != 1) while(blocked[0])	1	turn=0, blocked[0] = false
3	blocked[0] = true	0	turn=0, blocked[1] = true
4	while(turn != 0)	0	turn=0, blocked[1] = true
5	**Critical Section**	**0**	0进程进入临界区
6	turn = 1	1	turn=1, blocked[0] = true
7	while(turn != 1)	1	turn=1, blocked[0] = true
8	Critical Section	**1**	1进程进入临界区

该题的改题标准：是否讲清楚

关键点：

- 进程切换点：哪个语句执行前/后发生了进程切换

5.22

操作	意义
p1	empty-1

p2	缓冲区+1
p3	full+1
c1	full-1
c2	缓冲区-1
c3	empty+1

5.22x

```
设置信号量rdr(初值1)，wtr(初值1)，m(初值1)，x(初值1)，y(初值1)
DataType The_Data;
int reader_counter=0, writer_counter=0;
Main(){
  Create_thread(reader,1);...;  Create_thread(reader,N); /*创建n个读者线程；*/
  Create_thread(writer,1);...;  Create_thread(writer, M); /*创建m个写者线程；*/
}
reader(i) {
   p(rdr);v(rdr);
   p(x);
   reader_counter++;
   if (reader_counter==1) { p(wtr);p(m);}
   v(x);
   reading(The_Data);
   p(x);
   reader_counter--;
   if (reader_counter==0) { v(wtr);v(m);}
   v(x);
}

writer(i) {
   p(wtr);v(wtr);
   p(y);
   writer_counter++;
   if (writer_counter==1)  p(rdr);
   v(y);
   p(m);
   writing(The_Data);
   v(m);
   p(y);
   writer_counter--;
   if (writer_counter==0)  v(rdr);
   v(y);
}
```