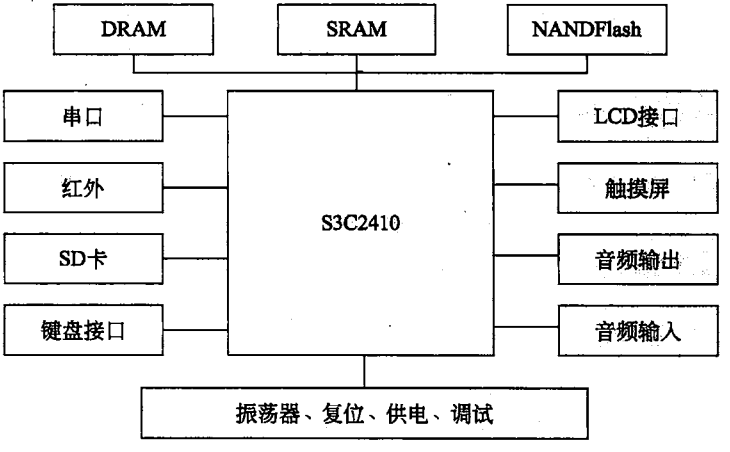
第 14 章　嵌入式系统设计师下午试题分析与解答

试题一（共 15 分）

下面是关于 PDA 设计方案的叙述，仔细阅读并分析，回答问题 1 至问题 3，将答案填入答题纸的对应栏内。

【说明】

个人数字助理（Personal Digital Assistant，PDA）是典型的嵌入式系统，具有计算、电话、网络和个人信息管理等多项功能。某单位欲开发一款 PDA 产品，选择 S3C2410 作为 CPU，存储器采用 SRAM、DRAM 和 NAND Flash 三种内置存储器，显示器采用 LCD，下图为 PDA 的硬件示意图。软件采用嵌入式 Linux 操作系统。



PDA 的硬件配置图

【问题 1】

不同类型的存储器，其特性也不同，请完成下表中的空白处内容，在“易失性”栏中填写“是”或“否”，在“相对读写速度”栏中填写“快”、“中”或“慢”。

存储器的设备特征

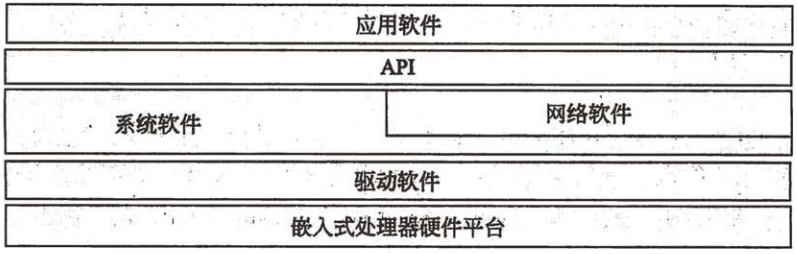
存储器种类	易　失　性	相对读写速度
SRAM		
DRAM		
NAND Flash		

【问题 2】

该 PDA 产品的软件如下所示：

（1）记事本　　（6）游戏软件
（2）电源管理　　（7）GUI 软件
（3）TCP/IP 协议栈　　（8）GPS 导航定位软件
（4）文件系统　　（9）处理触摸屏的软件
（5）LCD 驱动程序　　（10）Word 文字处理软件

下图是 PDA 软件的层次关系示意图，共分为 4 类软件。

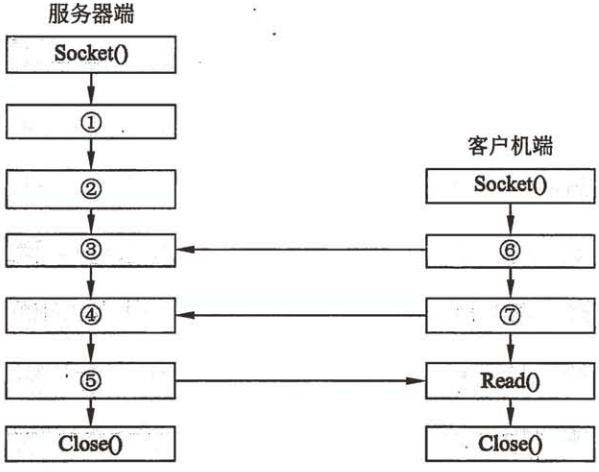


PDA 软件层次关系示意图

请说明上述 10 个软件所属的软件类别（将软件的编号填入答题纸相应的位置）。
（注意：每个选项只能属于一类软件，有重复者按选错对待）

【问题 3】

该 PDA 产品的操作系统采用嵌入式 Linux，网络协议采用 TCP/IP，下图是未完成的面向连接的 socket 通信流程图，请从下列子程序（参数和返回值略）中选择恰当者填入下图所示流程图的相应编号处。



面向连接的 socket 通信流程图

（1）Accept()　　（2）Bind()　　（3）Connect()
（4）Listen()　　（5）Read()　　（6）Write()

试题一分析

本题以 PDA 设计为背景，考查嵌入式系统设计的基本概念和软件设计能力。

【问题 1】

存储器是构成嵌入式系统硬件的重要组成部分，易失性和读写速度是存储器重要的性能指标。嵌入式系统中使用的存储器主要包括随机存储器、只读存储器和混合型存储器等。它们还可再进行细分，如 SRAM、DRAM、掩模 ROM、PROM、EPROM、EEPROM、Flash 和 NVRAM 等。

从易失性上讲，SRAM 和 DRAM 是易失的；从相对读写速度上讲，静态 RAM 即 SRAM 最快，下来是 DRAM，NAND Flash 最慢。

【问题 2】

PDA 的软件按其软件结构图所示依次划分为应用层的应用软件、操作系统层的系统软件和网络部分软件、硬件隔离层的驱动软件。

记事本、游戏软件、GPS 导航定位软件和 Word 文字处理软件在应用层，应属于应用软件；电源管理、文件系统和 GUI 软件属于操作系统层的系统软件；TCP/IP 协议栈属于网络部分软件；LCD 驱动程序、处理触摸屏的软件属于硬件隔离层的驱动软件。

【问题 3】

Socket（套接字）是进程间的通信机制，既适用于同一台计算机上的进程间通信，也使用于网络环境的进程间通信。网络通信有两种主要模式，一种为面向连接的通信，另一种为无连接通信。

在面向连接的 socket 通信模式中，通信双方要先通过一定的步骤在互相之间建立起一种虚拟的连接，或者说虚拟的线路，然后再通过虚拟的连接线路进行通信。在通信的过程中，所有报文传递都保持着原来的次序，报文在网络中传输是可靠的。

面向连接的 socket 通信流程图是一个客户端/服务器模型，服务器端程序的功能是监听其端口，如果发现有客户端的请求到来，就产生一个子进程与客户端进行通信。服务器端首先调用 Socket()创建一个 socket，然后调用 Bind()与本地地址/端口号绑定，成功之后就通过调用在相应的 socket 上监听。当 Accept()捕捉到一个连接服务请求时，就生成新的 socket，并通过这个新的 socket 与客户端通信，然后关闭该 socket。

客户端程序首先创建一个 socket，通过调用 Connect 函数与服务器建立连接，连接成功后与服务器通信，接收服务器发过来的数据，最后关闭 socket，结束程序。

参考答案

【问题 1】

存储器种类	易失性（是/否）	相对速度(快/中/慢)
SRAM	是	快
DRAM	是	中
NAND Flash	否	慢

【问题2】

属于应用软件的是：（1）、（6）、（8）、（10）；

属于系统软件的是：（2）、（4）、（7）；

属于网络部分软件的是：（3）；

属于驱动软件的是：（5）、（9）。

【问题3】

①：（2）

②：（4）

③：（1）

④：（5）

⑤：（6）

⑥：（3）

⑦：（6）

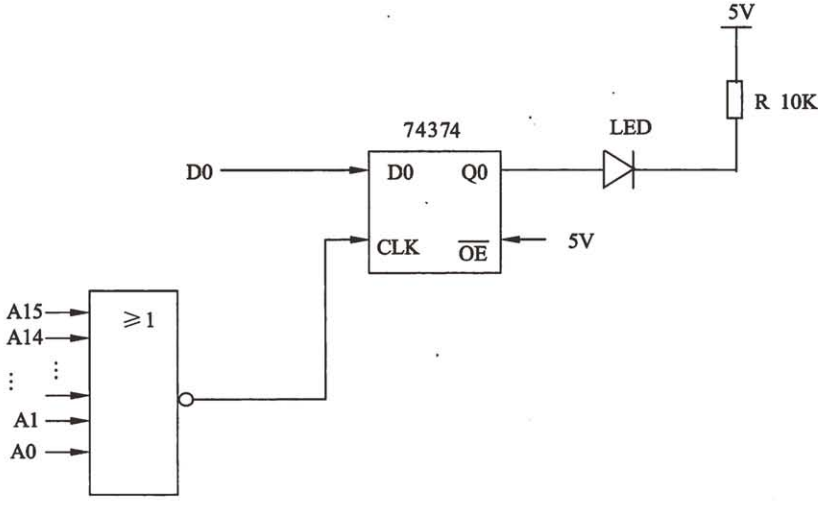
试题二（共15分）

阅读以下关于LED接口电路的叙述，回答问题1至问题2，将答案填入答题纸的对应栏内。

【说明】

某计算机系统采用内存和接口统一编址方式。内存可寻址空间为1MB，内存地址用A0～A19传送，读写信号分别为/MEMR和/MEMW；接口可寻址空间为64KB，接口地址用A0～A15传送，读写信号分别为/IOR和/IOW。

在该计算机系统上设计的LED接口电路如下图所示，分配的接口地址为0000H。图中的74374为锁存器，其真值表见下表。



LED接口电路图

74374 锁存器的真值表

INPUTS			OUTPUTS
$\overline{OE}$	CLK	D	Q
L	↑	H	H
L	↑	L	L
L	Hor L	X	Q
H	X	X	Z

Z 表示高阻。

【问题 1】

上图所示的 LED 接口电路中有设计错误，请找出其中至少 4 处错误（从编号为①～⑧的备选答案中选择）。

① 74374 的 $\overline{OE}$ 接 5V

② A16～A19 没参加接口地址译码

③ LED 的限流电阻 R 的阻值太小

④ 译码器为或非门

⑤ LED 阴极接电源

⑥ /MEMW 没参加接口地址译码

⑦ /IOW 没参加接口地址译码

⑧ LED 的限流电阻 R 的阻值太大

【问题 2】

请针对问题 1 找出的 LED 接口电路设计中的错误，简要分析其故障原因。

试题二分析

本题考查 LED 接口电路应用方面的知识。

【问题 1】

① 图中设计“74374 的 $\overline{OE}$ 接 5V”的做法是错误的。因为 74374 是具有三态输出的 D 触发器，其三态输出的允许信号 $\overline{OE}$ 应该接地，如果 $\overline{OE}$ 接 5V，使得 74374 输出高阻。

② 图中“A16～A19 没参加接口地址译码”的描述是正确的。因为根据题意，要求的是接口寻址，接口地址用 A0～A15 传送，故 A16～A19 不参加接口地址译码。

③ 从图中可以看出，LED 的限流电阻 R 的阻值太大，应该减小。

④ 图中设计的译码器采用的是或非门，这种设计是不正确的。因为根据题意分配的接口地址为 0000H，所以译码器采用或门才能符合设计要求。

⑤ 图中设计“LED 阴极接电源”的做法是错误的。LED 阴极应接地，并将电阻减小；或者将 LED 阴极接到 Q0 端，并将电阻减小。

⑥ /MEMR 和/MEMW 是内存读写信号，读写接口时不需要用到。

⑦ 图中设计的“/IOW”没参加接口地址译码的做法是错误的。

⑧ 图中设计的“LED的限流电阻R的阻值太大”的做法是错误的。

【问题2】

根据问题1的分析结果可知，设计中74374的/OE端应接地，并且应采用或非门做译码器，译码输入应是A0～A15加/IOW信号。对于“LED阴极接电源”的设计错误可以采用两种办法纠正：将LED阴极接地，电阻减小；或者将LED阴极接到Q0端，并将电阻减小。

参考答案

【问题1】

① 74374的$\overline{OE}$接5V

④ 译码器为或非门

⑤ LED阴极接电源

⑦ /IOW没参加接口地址译码

⑧ LED的限流电阻R的阻值太大

【问题2】

1. 74374的$\overline{OE}$接5V，使得74374输出高阻，应该接地；

2. 译码器中或非门设计错误，使得74374的锁存时序错误，应为或门；

3. LED阴极接电源，接反了，不会发光，应该是阳极接电源；

4. /IOW没参加接口地址译码，使得接口地址和内存地址冲突；

5. LED的限流电阻R的阻值太大，LED不会发光，R的阻值应约为330Ω～1KΩ。

试题三（共15分）

下面是关于嵌入式软件测试方面的叙述，回答问题1和问题2，将解答填入答题纸的对应栏内。

【说明】

甲公司是一个专业的软件测评中心，承担了某机载软件测试任务。王工是该测试任务的负责人。用户指出，被测件是控制飞机飞行的关键软件，其安全性要求很高，必须按有关规定开展测评工作。

【问题1】

王工与被测方讨论被测件的测试计划时，在测试环境方面产生了分歧。王工认为：由于当前被测件的实验平台要用于系统联试，没有时间提供给测评工作，测评工作可在仿真环境下进行，没有必要非得在目标机环境下测试；而被测方认为：软件测评工作仅仅用仿真环境是不够的，不能真实反映软件特性，可根据需要安排在实验平台上进行。

请对双方的意见进行分析，回答①～④问题：

王工和被测方的意见__①__。

① A. 都对　　B. 王工正确而被测方不完全正确

C. 都不完全正确　　D. 被测方正确而王工不完全正确

你对有关测试环境的建议如下（请将建议填入答题纸的对应栏内）：

1. ＿＿＿＿＿＿＿＿＿＿②＿＿＿＿＿＿＿＿＿＿；

2. ＿＿＿＿＿＿＿＿＿＿③＿＿＿＿＿＿＿＿＿＿；

3. ＿＿＿＿＿＿＿＿＿＿④＿＿＿＿＿＿＿＿＿＿。

【问题 2】

仔细阅读以下有关修正的条件判定覆盖（MC/DC）和条件判定覆盖（C/DC）的叙述，回答①～④问题，并将其填入答题纸的对应栏内。

由于被测件是关键级软件，按有关规定，被测件的测试必须达到 MC/DC。MC/DC 要求测试集满足＿①＿条件；C/DC 要求测试集满足＿②＿条件。下图所示的例子中给出了两个判定条件的例子，则满足 MC/DC 要求的测试集是{＿③＿}，满足 C/DC 要求的测试集是{＿④＿}。

```
IF A OR B THEN
    ⋮
ELSE
    ⋮
ENDIF
```

两个判定条件的例子

试题三分析

本题考查嵌入式软件测试方面的基本知识和实际分析问题，解决问题的能力。

【问题 1】

在软件测试，尤其是嵌入式软件测试中，用仿真环境虽然可以完成部分测试工作，如单元测试，但是这些测试工作只能认为是编写测试用例的工作，为了确保软件与硬件的配套性，必须将测试用例在实际目标机上运行，因此，A、B、D 三种说法都不正确，C 是正确的选项。由于嵌入式系统的特殊性，本题主要考查考生对嵌入式软件测试工作的要求理解程度。

在了解嵌入式系统特性的基础上，应能给出以下建议：

第一条“要在目标机环境下进行软件测试”，这是嵌入式系统必须遵循原则；

第二条“采用在目标机环境下和在仿真环境下相结合的方法”，这是一种并行工作的有效方式；

第三条“妥善安排目标机环境的使用，使甲方的开发与乙方的测试两不误”，类似于第二条，只是提醒项目开发方在必要的时刻应安排目标机环境给测试方使用。

【问题2】

根据宇航系统的软件安全性考虑要求，宇航系统将根据设备对飞行器安全性影响危害程度将软件分为5级（A～E），其中A级软件是关键软件，在软件测试中必须达到100%修正的条件判定覆盖（MC/DC）和条件判定覆盖（C/DC）。这里主要考查考生是否理解这两种测试要求的具体含义。

MC/DC要求测试集条件是：首先应满足C/DC的测试条件；判定中每个条件的取值都必须独立地影响判定的结果，即在其他条件取值不变的前提下，仅改变这个条件的值，而使判定结果改变。因此，对所列的A、B两个条件变量，{ FF，TF，FT }集合就满足了MC/DC要求。

C/DC要求测试集条件是：判定中每个条件的真值与假值都至少取一次；同时判定的真值与假值也要求至少取一次。因此，对所列的A、B两个条件变量，{ TT，FF }集合就满足了C/DC要求。

参考答案

【问题1】

① C

② 要在目标机环境下进行软件测试

③ 采用在目标机环境下和在仿真环境下相结合的方法

④ 妥善安排目标机环境的使用，使甲方的开发与乙方的测试两不误

【问题2】

① 首先应满足C/DC的测试条件；判定中每个条件的取值都必须独立地影响判定的结果，即在其他条件取值不变的前提下，仅改变这个条件的值，而使判定结果改变。

② 判定中每个条件的真值与假值都至少取一次；同时判定的真值与假值也要求至少取一次。

③ FF，TF，FT

④ TT，FF

试题四（共15分）

阅读以下关于80X86处理器方面的叙述，请回答问题1至问题3，将解答填入答题纸的对应栏内。

【说明】

80X86计算机中的寻址方式包括程序寻址和数据寻址两类。数据寻址方式是指获取指令所需的操作数或操作数地址的方式；程序寻址方式是指程序中出现转移和调用时的程序定位方式。

部分数据寻址方式见下表，其中，为每种数据寻址方式分配一个编号。

数据寻址方式及编号

编　号	寻 址 方 式
1	直接寻址
2	寄存器间接寻址
3	基址寻址
4	变址寻址
5	带比例因子的变址寻址
6	基址变址寻址

程序寻址方式见下表，其中，为每种程序寻址方式分配一个编号。

程序寻址方式及编号

编　号	寻 址 方 式
1	段内直接寻址方式
2	段内间接寻址方式
3	段间直接寻址方式
4	段间间接寻址方式

【问题 1】

按照“数据寻址方式及编号”表所列出的数据寻址方式，说明下表中各汇编指令指定的操作数或操作数地址属于哪类数据寻址方式，将答案填写在答题纸的对应栏中（直接填写编号即可）。

汇编指令采用的数据寻址方式

指　令	寻 址 方 式
MOV　ECX,　[EAX+24]	
IMUL　EBX,　TABLE [ESI*4], 7	
INC　WORD　PTR [500]	
ADD　EAX,　TABLE[ESI]	
MOV　EAX,　[ESI][EBX]	
MOV　[ECX],　EDX	

【问题 2】

按照上表给出的程序寻址方式，说明下表中各汇编指令中指定的地址属于哪类程序寻址方式，将答案填写在答题纸的对应栏中（直接填写编号即可）。

汇编指令属于的程序寻址方式

指　令	寻 址 方 式
JMP　BX	
CALL　2600H: 3800H	
JMP　WORD　PTR[BP+TABLE]	
CALL　DWORD　PTR[DI]	
JMP　1000H	

【问题3】

以下汇编程序用于求寄存器AX中符号数的绝对值。请将下面汇编程序的空(1)～(4)补充完整，并将解答填入答题纸的对应栏中。

```
CMP   AX, (1)
JL    (2)
JMP   (3)
YESNEG:  NEG   AX
NONEG:   MOV  RESULT, (4)
```

将上述汇编程序改进如下，请将改进后的汇编程序的空（5）～（8）补充完整，并将解答填入答题纸的对应栏中。

```
CMP   AX, (5)
JGE   (6)
NEG   (7)
NONEG:   MOV   RESULT, (8)
```

试题四分析

本题考查汇编语言的基础知识。

【问题1】

数据寻址方式有以下几种：

（1）直接寻址。这种寻址方式的位移量就是操作数的有效地址，位移量直接包含在指令中，它与操作数一起存放在代码段区域。例如INC WORD PTR [500]，该指令的有效地址为500，它的线性地址=数据段基地址+500。

（2）寄存器间接寻址。这种寻址方式是由寄存器给出有效地址的指针，即有效地址是基址或变址寄存器中的内容。例如MOV [ECX],EDX，操作数的逻辑地址=数据段基地址+ECX中的内容。

（3）基址寻址。基址寄存器的内容，加上指令格式中的位移量而形成操作数的有效地址。例如MOV ECX,[EAX+24]，操作数的逻辑地址=数据段基地址+由EAX中内容加位移量24组成操作数的有效地址。

（4）变址寻址。与基址寻址方式相似，其有效地址的形成是变址寄存器的内容加上指令格式中的位移量。例如ADD EAX,TABLE[ESI]，操作数的逻辑地址=数据段基地址+ESI中的内容加TABLE变量的地址组成操作数的有效地址。

（5）带比例因子的变址寻址。是变址寻址方式的另一种寻址方式，指操作数的有效地址等于变址寄存器内容乘以比例因子再加上指令格式中的位移量。例如 IMULEBX, TABLE[ESI*4],7，操作数的逻辑地址=数据段基地址+ESI中的内容乘以4再加上TABLE

变量的地址形成的有效地址。

（6）基址变址寻址。操作数的有效地址等于基址寄存器的内容加变址寄存器的内容。例如 MOV EAX,[ESI][EBX]，操作数的逻辑地址=数据段基地址+EBX 中的内容加 ESI 中的内容形成的操作数有效地址。

【问题 2】

程序寻址方式有以下几种：

（1）段内直接寻址。是指把指令本身提供的位移量加到指令指针寄存器中去形成目标有效地址的寻址方式。例如 JMP 1000H，调用地址在指令中给出。

（2）段内间接寻址。程序转移的地址存放在寄存器或存储单元中，由于此寻址方式仅修改 IP 的内容，因此这种寻址方式只能在段内进行程序转移。例如 JMP BX，转移地址由 BX 给出；又如 JMP WORD PTR [BP+TABLE]，转移地址由 BP+TABLE 所指的存储单元给出。

（3）段间直接寻址。这种寻址方式是在指令中直接给出 16 位的段基值和 16 位的偏移地址来更新当前 CS 和 IP 的内容。例如 CALL 2500H:3600H，调用的段地址和偏移地址都在指令中给出。

（4）段间间接寻址。这种寻址方式是由指令中给出的存储器数据寻址方式，包括存放转移地址偏移量和段地址。其低位字地址单元存放的是偏移地址，高位字地址单元中存放的是转移段基值。这样既更新了 IP 的内容，又更新了 CS 的内容。例如 CALL DWORD PTR[DI]，调用地址在 DI、DI+1、DI+2、DI+3 所指的内容单元中，前两个字节为偏移量，后两个字节为段地址。

【问题 3】

本题中的汇编程序用于求寄存器 AX 中符号数的绝对值，算法一的分支条件是 AX<0，当条件满足时，即 AX 为负数，需要求补；当条件不满足时，即 AX 为正数，不需要求补，转向保存结果。

算法二的分支条件是 AX>=0，只需要判断条件不满足时，即 AX 为负数，进行求补运行，然后保存结构。

参考答案

【问题 1】

指　令	寻 址 方 式
MOV　ECX, [EAX+24]	3
IMUL　EBX, TABLE [ESI*4],7	5
INC　WORD　PTR [500]	1
ADD　EAX, TABLE [ESI]	4

续表

指　令	寻址方式
MOV EAX, [ESI][EBX]	6
MOV [ECX], EDX	2

【问题2】

指　令	寻址方式
JMP BX	2
CALL 2600H:3800H	3
JMP WORD PTR [BP+TABLE]	2
CALL DWORD PTR [DI]	4
JMP 1000H	1

【问题3】

（1）0　（2）YESNEG　（3）NONEG　（4）AX

（5）0　（6）NONEG　（7）AX　（8）AX

试题五（共15分）

阅读以下关于利用信号量机制解决进程同步与互斥方面的应用实例，回答问题1至问题3，将答案填入答题纸的对应栏内。

【说明】

在多道程序系统中，进程是并发执行的。这些进程间存在着不同的相互制约关系，主要表现为同步和互斥两个方面。信号量机制是解决进程间同步与互斥的有效方法。下面是信号量应用实例。

下图所示代码是在μC/OS-II操作系统上运行的一个应用的主函数。该函数创建了任务Task1和Task2，其中Task1实现从键盘读入一个字符的功能，Task2将该字符输出到屏幕，它们使用信号量和一个公共变量buffer来传递该字符。

主函数、Task1和Task2中所调用的函数原型说明如下：

a. 创建一个信号量：OS_EVENT *OSSemCreate(INT16U value);

b. 创建一个任务：INT8U OSTaskCreate(void(* task)(void *pd), void *pdata,OS_STK *ptos,INT8U prio);

c. 开始执行多任务：void OSStart(void);

d. 从键盘读入一个字符：char scanc();

e. 输出一个字符至屏幕：void printc(char ch);

f. 发出一个信号量：INT8U OSSemPost(OS_EVENT *pevent);

g. 等待一个信号量：void OSSemPend(OS_EVENT *pevent,INT16U timeout,INT8U *err)

```
INT8U buffer;  /*全局变量*/
OS_EVENT *emptySem;
OS_EVENT *fullSem;
void APP_vMain (void)
 {
      OSInit();
      emptySem = OSSemCreate(1);
      fullSem = OSSemCreate(0);
      OSTaskCreate(Task1,(void *)0,&TaskStartStk[TASK_STK_SIZE - 1], 0);
      OSTaskCreate(Task2,(void *)0,&TaskStartStk[TASK_STK_SIZE - 1], 0);
      OSStart();                          /* Start multitasking*/
 }
```

应用的主函数

【问题 1】

请简述什么是临界资源？什么是临界区？访问临界资源应遵循哪些原则？

【问题 2】

设 S 为信号量，P、V 操作的形式化定义如下图（a）和（b）所示，请完成该形式化定义，将应填入__(n)__处的内容写在答题纸的对应栏中。

```
P(S)
{
   __(1)__;
   IF( __(2)__ ) {
       阻塞该进程;
       将该进程插入信号量 S 的等待队列;
    }
 }
```

（a）P 操作的形式化定义

```
V(S)
{
   __(3)__;
   IF( __(4)__ ) {
       从信号量 S 的等待队列中取出队首进程;
       将其插入就绪队列;
    }
 }
```

（b）V 操作的形式化定义

【问题3】

请根据本题要求完善任务Task1和任务Task2的程序代码，填补图中的空缺，将答案填写在答题纸的对应栏中。

```
void   Task1 (void *pdata)
{
  INT8U readc;
  INT8U err;
     INT8U ret;
      while( 1 ){
           readc = scanc();
        ___(1)___;
        ___(2)___;
        ___(3)___;
      }
  }
```

```
void   Task2 (void *pdata)
{
  INT8U writec;
  INT8U err;
     INT8U ret;
  while( 1 ){
           ___(4)___;
       writec = buffer;
        ___(5)___;
        ___(6)___;
      }
  }
```

试题五分析

本题考查进程同步与互斥的基本概念和应用。

【问题1】

在多道程序系统中，进程是并发执行的，这些进程之间存在着不同的相互制约关系。进程之间的这种制约关系来源于并发进程的合作以及对资源的共享。

进程在运行过程中，一般会与其他进程共享资源，而有些资源的使用具有排他性。系统中的多个进程可以共享系统的各种资源，然而其中许多资源一次只能为一个进程所使用，通常把一次仅允许一个进程使用的资源称为临界资源。许多物理设备都属于临界

资源，如打印机、绘图机等。除物理设备外，还有许多变量、数据等都可由若干进程所共享，它们也属于临界资源。

进程中访问临界资源的那段代码称为临界区，也称为临界段。

访问临界资源应遵循如下原则：

（1）空闲让进（或有空即进）。当没有进程处于临界区时，可以允许一个请求进出临界区的进程立即进入自己的临界区；

（2）忙则等待（或无空则等）。当已有进程进入其临界区时，其他试图进入临界区的进程必须等待；

（3）有限等待。对要求访问临界资源的进程，应保证能在有限时间内进入自己的临界区；

（4）让权等待。当进程不能进入自己的临界区时，应释放处理机。

【问题 2】

信号量是荷兰著名的计算机科学家 Dijkstra 于 1965 年提出的一个同步机制，其基本思想是在多个相互合作的进程之间使用简单的信号来同步。

在操作系统中，信号量是表示资源的实体，除信号量的初值外，信号量的值仅能由 P 操作（又称 Wait 操作）和 V 操作（又称 Signal 操作）改变。

设 S 为一个信号量，P（S）执行时主要完成的功能为：先执行 S = S–1，若 S≥0，则进程继续运行；若 S<0，则阻塞该进程，并将它插入该信号量的等待队列中。

V（S）执行时主要完成的功能为：先执行 S=S+1，若 S>0，则进程继续执行；若 S≤0，则从该信号量等待队列中移出第一个进程，使其变为就绪状态并插入就绪队列，然后再返回原进程继续执行。

P、V 操作的形式化描述如下：

```
P(S)
{
      S --;
      IF( S<0 )  {
          阻塞该进程;
          将该进程插入信号量 S 的等待队列;
       }
}


V(S)
{
      S++;
      IF( S <= 0 ) {
```

```
            从信号量 S 的等待队列中取出队首进程;
            将其插入就绪队列;
        }
    }
```

【问题 3】

本问题是信号量应用实例。

μC/OS-II 操作系统提供了操作信号量的若干系统调用，任务 Task1 为了实现从键盘读入一个字符并写到 buffer 的功能，就必须在读完字符后调用 OSSemPend()和 OSSemPost()对写 buffer 缓冲的动作加锁和解锁。

Task2 为了完成将该字符输出到屏幕，也必须在读 buffer 缓冲的动作时加锁和解锁。这样就避免任务 Task1 和 Task2 同时操作 buffer 缓冲的资源冲突。

因此，任务 Task1 的代码如下：

```
void    Task1 (void *pdata)
{
    INT8U readc;
    INT8U err;
   INT8U ret;
        while( 1 ){
            readc = scanc();
OSSemPend(emptySem,0,&err);
Buffer = readc;
ret = OSSemPost(fullSem);
        }
    }
```

任务 Task2 的代码如下：

```
void    Task2 (void *pdata)
{
    INT8U writec;
    INT8U err;
   INT8U ret;
    while( 1 ){
        OSSemPend(fullSem,0,&err);
        writec = buffer;
        ret = OSSemPost(emptySem);
        Printc(writec);
        }
    }
```

参考答案

【问题 1】

临界资源：一次只能给一个进程访问的资源。

临界区：进程中访问临界资源的那段代码。

访问临界资源时应遵循如下原则：

（1）空闲让进（或有空即进）。

（2）忙则等待（或无空则等）。

（3）有限等待。

（4）让权等待。

【问题 2】

（1）S--

（2）S<0

（3）S++

（4）S<=0

【问题 3】

（1）OSSemPend(emptySem,0,&err)

（2）Buffer = readc

（3）ret = OSSemPost(fullSem)

（4）OSSemPend(fullSem,0,&err)

（5）ret = OSSemPost(emptySem)

（6）Printc(writec)