单选题：

1. 下列关于CISC的说法正确的是？ （5分） **B**
2. 编码长度不可变 B. 寻址方式更多

C. 采用了优化的编译技术 D. ARM专注于这种架构的指令集

1. Cortex-A8中哪个处理器（工作）模式可以进行软件仿真？ （5分）**A**
2. 未定义指令模式 B. IRQ模式 C. FIQ模式 D. 数据访问中止模式
3. 下列哪种中断常用于保护虚存和存储器？（5分）**A**
4. 指令预取中止异常 B. 快速中断异常

C. 未定义指令异常 D. 软件中断异常

1. 以下Thumb指令有几处错误？ （5分）**C**

LDR R2, [R0, #50] STRB R1, [R2, #40]

1. 0 B.1 C.2 D.3
2. 关于ARMv7的汇编语言，正确的是？ （5分）**C**
3. IF和ELSE有相应的操作码
4. 变量标号不用顶格写
5. ARM汇编程序结构中的数据段可有可无
6. Msr属于合法的指令
7. 关于UART的说法，正确的是？ （5分） **C**
8. UART是基于RS-485接口的
9. UART是全双工的工作方式
10. UART可通过控制寄存器控制波特率大小
11. UART将外部的串行数据直接转给计算机
12. 下列哪种嵌入式处理器常用于数字信号处理？（5分）**B**

A. MCU B. DSP C. MPU D. SoC

1. S5PV210中的内核属于那种构架？（5分）**C**

A. Cortex-M B. Cortex-R C. Cortex-A D. 以上均不是

1. 下列选项中数据存取速度依次递增的是？（5分）**B**
2. 高速缓存、寄存器组、外存、内存
3. 外存、内存、高速缓存、寄存器组
4. 寄存器组、高速缓存、内存、外存
5. 外存、内存、寄存器组、高速缓存
6. NAND Flash和NOR Flash哪个支持串行读写？ （5分）**A**

A. NAND Flash B. NOR Flash C.均不支持 D.均支持

1. 关于自旋锁和信号量的理解正确的是？ （5分）**B**
2. 自旋锁申请失败后，任务会进入阻塞（挂起）状态
3. 信号量不能在中断服务程序中使用
4. 自旋锁可适用于锁保持时间很长的情况
5. 自旋锁可由限定数量的任务共享，超过共享数后新任务无法获得锁
6. Linux系统中，python适合安装在哪个目录？ （5分）**C**
7. /var B. /root C. /usr D. /lib
8. μC /OS-II中，某个事件标志组（OS\_FLAG\_GRP）的等待任务链表中某个任务 （OS\_FLAG\_NODE）中关于事件等待类型的字段OSFlagNodeWaitType 的值设定为OS\_FLAG\_WAIT\_SET\_AND，且任务的OSFlagNodeFlags即信号过滤器字段的值为00101101，则事件标志组（OS\_FLAG\_GRP）对应的OSFlagFlags字段值为多少时，该任务可以结束等待？ （5分）**B**
9. 00110101 B. 10101111 C .00000000 D. 00101110
10. 关于占先式内核和非占先式内核说法错误的是？ （5分）**C**
11. 非占先式内核的性质决定了它难以决定任务的响应时间
12. μC /OS-II操作系统内核属于占先式内核
13. 占先式内核可以直接使用不可重入函数
14. 占先式内核中，最高任务一旦就绪，总能得到CPU控制权
15. 关于字符设备和块设备错误的是？ （5分）**C**
16. 字符设备一般只能顺序读写，块设备可以随机读写
17. 串口属于一种字符设备
18. 控制台属于一种块设备
19. 块设备的数据通常需要缓冲区间接传给驱动程序
20. 如果要保存文件路径到数据库，应使用SQLite的哪种数据类型？（5分）**D**
21. INTEGER B. NULL C. REAL D. TEXT
22. 关于SQLite的描述，错误的是？ （5分）**B**
23. B-Tree数据结构可用于建立索引
24. 其独占性的共享锁的粒度可以到行级别
25. SQLite可以用于存储图像数据
26. SQLite是支持ACID特性的
27. 在SQLite中，什么组件与索引有关？ （5分） **B**
28. Tokenizer B. B-Tree C . Pager D. Virtual Machine
29. 哪种场景适用于将GPIO设为输入模式？ （5分） **C**
30. 点亮LED灯 B. 驱动蜂鸣器响起

C. 读入用户按键的变化 D. 以上均不适用

20. 以下哪个不属于RTOS的特性？ （5分） **D**

1. 可以实现多任务运行 B. 实时性

C. 可进行定制 D. 须运行在有MMU的嵌入式系统上

简答：

1. 请解释以下ARMv7汇编代码的意义，必要时可画图。 （20分）

mrs r0, cpsr

bic r0, r0, #0xc0

msr cpsr\_c, r0

答案：mrs r0, cpsr表示的是将cpsr寄存器当前值传送到r0寄存器；

bic r0, r0, #0xc0表示将r0的第7和第8位清零，对应cpsr寄存器，则是将FIQ和IRQ中断屏蔽位清零，打开FIQ和IRQ两种中断。（画出cpsr寄存器部分图，强调I位和F位所在位置）

msr cpsr\_c, r0表示将r0的值传送回cpsr，其中\_c表示的是控制域，代表cpsr寄存器的前8位。

1. 阅读以下ARM汇编代码并回答问题。 （20分）

AREA DTBUF, DATA, READWRITE

count DCB #0xFFFF

AREA EXAMPLE1, CODE, READWRITE

ENTRY

CODE32 //32位指令集

START

MVN R0, #0x00FF ①

MOV R1, #0x00FF ②

ADD R2, R0, R1 ③

B START

END

1）请指出这段程序中出现的错误。（20分）  
答案：  
 第一处是count DCB #0xFFFF，因为DCB用于声明一个字节数据，而0xFFFF已经超出一个字节的范围；

第二处是AREA EXAMPLE1, CODE, READWRITE，因为代码段是只读属性，这里的READWRITE设成了可读可写属性；

第三处是//32位指令集 ，因为注释必须用分号开头；

第四处是最后的END，这个是伪指令，表示代码段结束，不能顶格写。

（2）①②③代码运行后R2的值是多少？请给出计算过程。（10分）

答案：

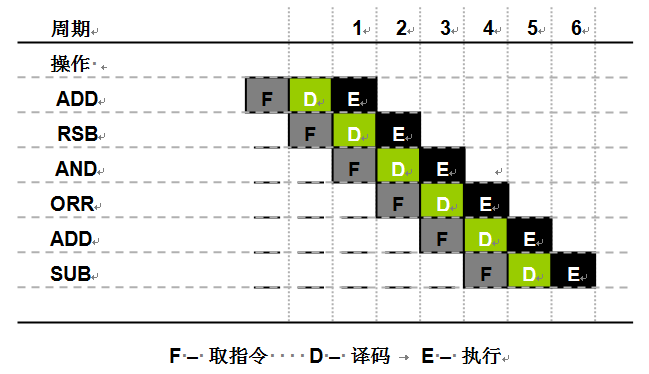
MVN R0, #0x00FF后，R0的值为 0xFFFFFF00

MOV R1, #0x00FF，R1的值为 0x00FF

ADD R2, R0, R1，R2的值为R0+R1，即 0xFFFFFF00+0x00FF，计算结果为0xFFFFFFFF。

1. 请画图，并解释ARMv7架构中，ARM状态下的三级流水线工作原理，以及PC值发生的变化。 （20分）

参考下图，可不写指令：（15分）



设正在执行的指令地址为PC，则正在译码指令地址为PC+4，正在执行指令地址为PC+8。 （5分）

4. 请说明ARM Cortex-A8中所有类型寄存器的作用。 （15分）

R0-R7寄存器为通用寄存器，可用于数据传输，算术运算和逻辑运算等，R8-R12寄存器可以表示FIQ模式下和非FIQ模式下的相应的存储器。 （3分）

R13为堆栈寄存器SP，在子程序调用时用于保存通用寄存器等。 （3分）

R14为链接寄存器LR，在子程序调用时用于保存返回地址。 （3分）

R15为程序计数器PC，表示指令地址，用于进行指令顺序执行或跳转。 （3分）

CPSR寄存器和SPSR寄存器用于保存程序当前的各种状态位。 （3分）

1. 请说明SRAM和DRAM各自的特点是什么？并举一个SRAM或DRAM适用的场景？ （15分）

SRAM存储单元电路是以双稳态电路为基础，因此状态稳定，只要不掉电，信息不会丢失。

DRAM存储单元电路是以电容为基础，即使不掉电也会因电容放电而丢失信息，需要定时刷新，因此在工作时必须配合DRAM控制器。

存储容量大的时候，可以使用DRAM；特别高速的应用，可以选择SRAM；

对功耗敏感，可以使用SRAM；芯片集成了DRAM控制器，更适合使用DRAM；

32位的嵌入式系统，更适合使用DRAM；

复杂的嵌入式系统可以选择SRAM和DRAM混用。（列举一个即可）

1. 请解释为什么Flash相对于EEPROM具有在线编程的特点？ （15分）

Flash的擦写特点：可以通过标准电压进行擦写，因此可以直接通过单片机的工作电压进行擦写，无需从开发板中拆除再擦写。

EEPROM的擦写特点：只能通过高压电的方式进行擦写，因此无法直接通过单片机工作电压及逆行擦写，必须从开发板中拆除后再擦写。

综上所述，Flash无需从开发板中拆除再擦写，具有在线编程的特点。

1. 请举出三个定时器应用有关的例子，并详细解释是如何进行应用的。 （15分）

答案：

**（有以下例子中的其中三个，并且有详细说明即可。）**

1. 嵌入式操作系统的任务调度，特别是具有时间片轮转调度功能的嵌入式操作系统必须使用定时器产生时间片。
2. 嵌入式操作系统的软件时钟需要基于硬件定时器产生定时信号。
3. 通信电路的波特率发生器。
4. 实时时钟电路。
5. 集成的片上A/D转换和D/A转换电路。
6. 具有液晶控制器的嵌入式处理器应用液晶屏的刷新。
7. 处理器监控电路，如看门狗等。
8. 集成的动态存储器控制器用于动态存储器的刷新。
9. 设当前指令集为Thumb指令集，阅读代码并回答问题： （30分）

LDRH R0, [R3, #40]

STRB R8, [R0, #0x1C]

STRH R4, [R2, #0x40]

LDRB R0, [R1, #0x2D]

（1）以上代码有几处错误？请指出具体的错误之处。（20分）

（2）ARM状态和Thumb状态如何相互切换？（10分）

（1）第一处，STRB R8, [R0, #0x1C]指令中R8有误，超出R0-R7的范围。

第二处，STRH R4, [R2, #0x40]指令中0x40有误，0x40转成十进制后是64，已经超出了半字寻址中的0-62个字节的范围。

第三处，LDRB R0, [R1, #0x2D]指令中0x2D有误，0x2D转成十进制后是45，已经超出了字节寻址中的0-31个字节的范围。

（2）Thumb到ARM状态，首先可以进行复位，因为复位后默认进入ARM状态，或执行BL或BLX指令，且目标地址最后一位设为0。

ARM转换到Thumb状态，执行BL或BLX指令，且目标地址最后一位设为1。

9.运用Linux常用命令完成以下内容（根目录出发，不考虑超级用户权限）：

在/usr/local/目录下，新建一个test\_data目录，以及一个test\_config目录。在test\_config目录下新建一个config文件，将其文件权限设置为：拥有者可读写执行，所在组可读写，其他组可读，修改后单独查看改文件权限信息。然后，将test\_config目录移动到test\_data目录中，随后删除test\_config目录（忽略提示信息），最后设定10分钟后重启。 （20分）

cd /usr/local或cd usr, cd local; mkdir test\_data; mkdir test\_config;  
cd test\_config; touch config; chmod 764 config; ls -l config;

cd ..或cd /usr/local; mv test\_config test\_data; cd test\_data;

rm -rf test\_config; shutdown -r 10

10.设μC/OS-II就绪表中OSUnMapTbl结构存储的数据为： （30分）

0, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0x00 to 0x0F\*/

4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0x10 to 0x1F \*/

5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0x20 to 0x2F \*/

4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0x30 to 0x3F \*/

6, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0x40 to 0x4F \*/

4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0x50 to 0x5F \*/

5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0x60 to 0x6F \*/

4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0x70 to 0x7F \*/

7, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0x80 to 0x8F \*/

4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0x90 to 0x9F \*/

5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0xA0 to 0xAF \*/

4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0xB0 to 0xBF \*/

6, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0xC0 to 0xCF \*/

**4**, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0xD0 to 0xDF \*/

5, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0, /\* 0xE0 to 0xEF \*/

4, 0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, 0, 1, 0 /\* 0xF0 to 0xFF \*/

目前有5个任务（ABCDE）进入就绪状态，优先级分别是：26, 28, 35, 47, 54。

1. 0xD0 to 0xDF 行（倒数第三行）下划线标记数据4如何得出？（10分）
2. 演示如何查找到5个任务中优先级最高的任务？（20分）

（1）0xD0 to 0xDF这一行中标记下划线的数据4，是因为当前位置是0xD0，转

换为二进制是 11010000，从0开始算第一次出现1的位置是第4位，因此为4。

（2）26：第3组，28：第3组，35：第4组，47：第5组，54：第6组：所以OSRdyGrp的值为：01111000（2），120（10），78（16）。

查表OSUnMapTbl[78]得到的值是3，表示第3组有任务进入就绪状态。

使用表OSRdyTbl[]，第3组的状态是：有26和28两个优先级的任务进入就绪状态，所以OSRdyTbl[0]为：00010100(2), 20(10)，查表OSUnMapTbl[20]，得到值为2。

综合得到（3<<3）＋2 ＝ 26，即当前就绪任务中，优先级最高的为26。

11.本课程的实验中，有一个实验是移植嵌入式Linux内核实验，请回答：（20分）  
（1）以gcc编译器为例，简要说明从源文件.c开始，最终生成可执行文件的全部过程。 （10分）

（2）解释为什么要使用arm-linux-gcc编译器，而不能直接用PC上Ubuntu系统自带的gcc编译器？ （10分）

（1）首先.c源文件在通过预处理后为.i格式文件，.i格式和.h头文件经过编译后成为.s格式文件，.s格式文件经过汇编后成为.o文件，最终将.o文件进行链接，得到最终的可执行文件。

（2）根据交叉编译的概念，是指程序的编译平台以及运行平台不一致，其中编译平台（宿主机）一般是个人电脑PC，而运行平台（目标机）一般是嵌入式开发板。因此，自带的gcc编译器不可用是因为它编译的程序只能在宿主机PC中运行，而arm-linux-gcc编译器编译的程序可以在目标机也就是嵌入式开发版中运行。

12.请举例解释嵌入式操作系统中的优先级反转现象，并简述解决方法。（15分）

设有任务A，B和C。优先级依次由高到低。

此时，任务B，C同时阻塞，任务C就绪，任务C先运行。

任务C执行过程中，申请了信号量。

1. B同时唤醒进入就绪状态，此时A任务先运行，B和C阻塞。

A运行时，申请信号量S失败被阻塞，此时B任务优先级更高，运行B任务。

B完成后，C继续执行并释放信号量S，随后A运行，最后是在B之后完成。

解决方法：在A申请信号量S失败被阻塞之后，C继承A的任务优先级，使得C的优先级高于任务B，先于任务B运行，等C释放信号量后，任务A解除阻塞先于任务B运行，从而解决了优先级反转的问题。

13.请说明你在进行SQLite和Python移植实验遇到的问题及解决方案？（20分）

合理即可

14.请解释为什么实验中的LED灯程序需要写成死循环形式。（10分）

CPU一旦从某个地址运行，它就会从这个地址往后依次取指运行，当运行完我们的代码，它不会停止，还会往后继续取指运行，但是后面的指令是未知的，CPU运行后不知道会是什么结果，可能正常执行，也可能出现异常，所以我们应该让CPU一直在那里死循环。

15.请说明自旋锁与信号量之间的区别，并说明其适用的场景。（20分）

（1）首先自旋锁可在不能睡眠的代码中使用，如在中断服务程序（ISR）中使用，而信号量不可以；

（2）其次自旋锁和信号量的实现机制不一样，自旋锁的申请者会不断循环检测锁是否可用，不会被阻塞，而信号量中，申请信号量失败后会被阻塞；  
（3）自旋锁只有一个持有者，信号量可以在消耗完之前有多个持有者；

（4）最后通常自旋锁被用在多处理器系统。

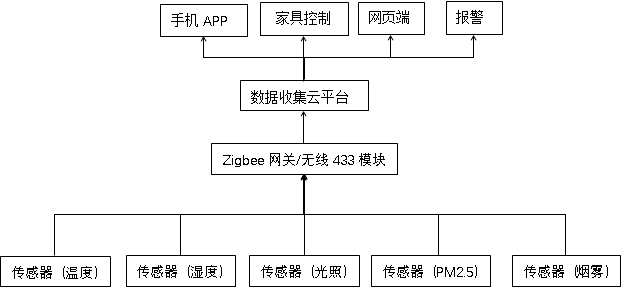
（5）自旋锁通常适合保持时间非常短的情况，它可以在任何上下文中使用，而信号量用于保持时间较长的情况，只能在进程上下文中使用。

16.现有一个智慧家居项目，要求采集温度等数据（其他数据合理的可自行补充），并使用这些数据进行智能家居控制。该系统使用S5PV210或STM32F407处理器，可选择Zigbee或无线315/433MHz技术用于传感器网络。请给出： （25分）

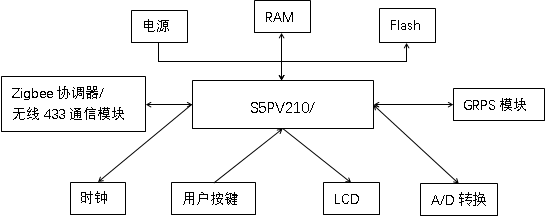
1. 系统设计的总体框架（含软件和硬件系统）； （10分）

（2）系统硬件设计方案，并说明其中两种硬件的作用（结合项目）。（15分）

（1）



（2）



例：

LCD屏幕：实时显示当前的数值，包括温度、湿度、CO2浓度、PM2.5等。

RAM：存放当前正在运行的代码和数据，例如要输出到屏幕的数值等。

Flash：可用于存放检测数据的日志文件，可用于存放嵌入式操作系统代码。