《嵌入式系统设计》教材复习题及课外练习题

第 1 章　嵌入式系统概述

教材 P11：1，2，4，5，7，9

1. 嵌入式系统的概念是什么？（教材 P11-1）
2. 嵌入式系统的特点是什么？（教材 P11-2）
3. 嵌入式操作系统的主要特点是什么？（教材 P11-7）
4. 叙述嵌入式系统的分类。（教材 P11-8）
5. 嵌入式处理器包括哪几种类型？

答：嵌入式处理器可以分为四类：

嵌入式微处理器 EMPU（Embedded Microprocessor Unit）嵌入式微处理器就是和通用计算机的微处理器对应的 CPU。在应用中，一般是将微处理器装配在专门设计的电路板上，在母板上只保留和嵌入式相关的功能即可，这样可以满足嵌入式系统体积小和功耗低的要求。

嵌入式微控制器 EMCU（Embedded Microcontroller Unit）嵌入式微控制器又称为单片机，它将 CPU、存储器（少量的 RAM、ROM 或两者都有）和其它外设接口封装在同一片集成电路里。

嵌入式数字信号处理器 EDSP（Embedded Digital Signal Processor）嵌入式 DSP 专门用来对离散时间信号进行极快的处理计算，提高编译效率和执行速度。在数字滤波、FFT、谱分析、图像处理等领域应用广泛。

嵌入式片上系统 ESoC （Embedded System on Chip）。•ESoC：在一个硅片上实现一个更为复杂的系统。

6. 如何理解嵌入式系统？

答：嵌入式系统指的是以应用为中心和以计算机技术为基础的，并且软硬件是可裁剪的，能满足应用系统对功能、可靠性、成本、体积、功耗等指标严格要求的专用计算机系统。

7. 与通用计算机相比，嵌入式系统有哪些特点？

答：与通用计算机相比，嵌入式系统特点：

1：嵌入式系统通常是面向特定应用的；

2：嵌入式系统功耗低、体积小、集成度高、成本低；

3：嵌入式系统具有较长的生命周期；

4：嵌入式系统具有固化的代码；

5：嵌入式系统开发需要专用开发工具和环境；

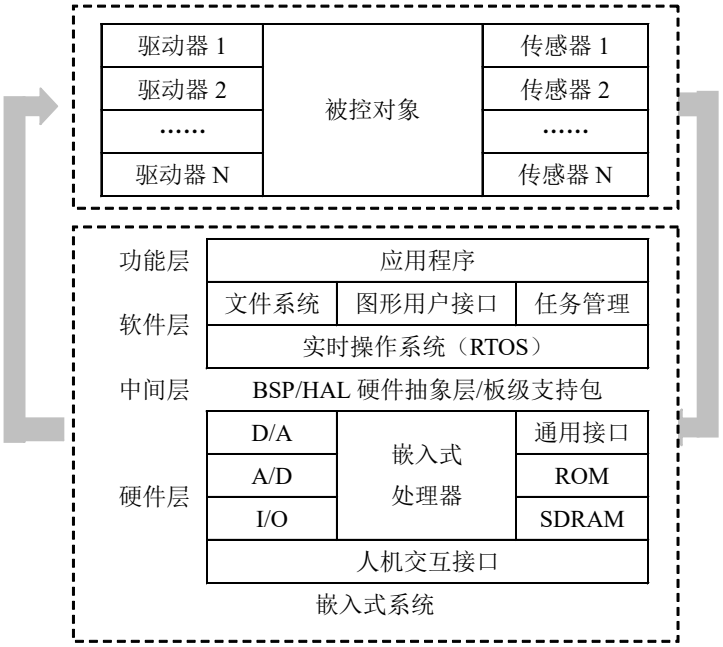
6：嵌入式系统软件需要 RTOS 开发平台；

7：嵌入式系统开发人员以应用专家为主；

8：嵌入式系统是知识集成系统。

8. 说明嵌入式系统的典型组成，分析常见嵌入式产品大体的结构组成？

答：嵌入式系统的典型组成：



这类产品从总体上来看由两部分组成：硬件、软件。

硬件：嵌入式芯片，由嵌入试 CPU、内存 RAM、ROM、寄存器、输入/输出通道组成。

软件：嵌入式操作系统、应用程序（C 语言、汇编、JAVA 语言等编成）。

然后这两部分由外壳包装起来。用户使按下键后，系统检测到后，经过输入通道收集，送到 OS，然后 OS 根据按键的命令，发出相应的命令，调用响应的程序。处理完毕之后，把结果输出。

9. 嵌入式系统的 BooTLoader 的功能是什么？

答：BootLoader 是系统加电后、操作系统内核或用户应用程序运行之前，首先必须运行的一段程序代码。通过这段程序,为最终调用操作系统内核、运行用户应用程序准备好正确的环境。（对于嵌入式系统来说，有的使用操作系统，也有的不使用操作系统，但在系统启动时都必须运行 BootLoader，为系统运行准备好软硬件环境。）

10. BSP 作为一种嵌入式软件，它的主要特点是“与硬件和操作系统都相关”。√

11. _____完全把系统软件和硬件部分隔离开来，从而大大提高了系统的可移植性。D

A、图形用户接口　B、驱动映射层 C、硬件交互层 D、硬件抽象层

12. 下面关于哈佛结构描述正确的是_____。A

A、程序存储空间与数据存储空间分离　B、存储空间与 IO 空间分离

C、程序存储空间与数据存储空间合并　D、存储空间与 IO 空间合并

13. SoC 是一种基于 IP（Intellectual Property）核嵌入式系统级芯片设计技术，它将许多功能模块集成在一个芯片上。√

14. 说明嵌入式系统的硬件组成。

答：嵌入式系统的硬件是以嵌入式处理器为中心，由存储设备、I/O 设备、通信接口设备、扩展设备接口以及电源等必要的辅助接口构成。

15. 下列不属于嵌入式操作系统的是（　　）。C

A．Windows CE　　B．VxWorks

C．windowsXP　　D．UC/OS

16. BSP 作为一种嵌入式软件，它的主要特点是（　　）。C

A．与硬件有关，与操作系统无关　　B．与硬件无关，与操作系统有关

C．与硬件和操作系统都相关　　D．与操作系统和硬件都无关

17. 嵌入式系统的主要应用领域有哪些（至少指出 5 个以上）？

答：消费电子、通信设备、家庭设备、汽车电子、工业控制、军事国防、医疗电子等。

18. 下列的（　　）描述不属于嵌入式系统的特点。B

A．专用性强　　B．软件丰富　　C．可靠性高　　D．实时性强

19. RTOS 的含义是（　　）。D

A．片上系统　B．先进精简指令集机器　C．存储器管理单元　　D．实时操作系统

20. 举例说明嵌入式系统的“嵌入性”、“专用性”、“计算机系统”的基本特征。

答：按照嵌入式系统的定义，嵌入式系统有3个基本特点，即“嵌入性”、“专用性”与“计算机”。

“嵌入性”由早期微型机时代的嵌入式计算机应用而来，专指计算机嵌入到对象体系中，实现对象体系的智能控制。当嵌入式系统变成一个独立应用产品时，可将嵌入性理解为内部嵌有微处理器或计算机。

“计算机”是对象系统智能化控制的根本保证。随着单片机向 MCU、SoC 发展，片内计算机外围电路、接口电路、控制单元日益增多，“专用计算机系统”演变成为“内含微处理器”的

现代电子系统。与传统的电子系统相比较，现代电子系统由于内含微处理器，能实现对象系统的计算机智能化控制能力。

“专用性”是指在满足对象控制要求及环境要求下的软硬件裁剪性。嵌入式系统的软、硬件配置必须依据嵌入对象的要求，设计成专用的嵌入式应用系统。

21. 下面的产品或系统，属于嵌入式系统的是（　　）。C

A．“天河一号”超级计算机　　B．戴尔 XPS13D-7508T 笔记本电脑

C．苹果 iPhone 6手机　　D．三星900X3K-K01超极本

22. 谈一谈嵌入式系统的发展趋势。

答：产品种类不断丰富；应用范围不断普及；性能不断提高；功耗不断降低，体积不断缩小；网络化、智能化程度不断提高；软件成为影响价格的主要因素。

23. 什么是初始化程序，bootloader 的作用。

答：初始化程序是系统加电后运行的第一段软件代码。

在嵌入式系统中，通常整个系统的加载启动任务就完全由 BootLoader 来完成。简单地说，BootLoader 就是在操作系统内核运行之前运行的一段小程序。通过这段小程序，可以初始化硬件设备、建立内存空间的映射图，从而将系统的软、硬件环境带到一个合适的状态，以便为调用应用程序或者操作系统内核准备好正确的环境。

24. 下列产品中不属于嵌入式系统的是：

A) 有线电视机顶盒

B) 服务器

C) 电饭煲

D) 路由器

【解析】广义上讲，凡是带有微处理器的专用软硬件系统都可称为嵌入式系统。狭义上讲，嵌入式系统强调那些使用嵌入式微处理器构成的具有自己的操作系统和特定功能、用于特定场合的独立系统。嵌入式系统以应用为中心、以计算机技术为基础、软硬件可裁剪、功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。选项中 A、C、D 均属于嵌入式系统；而服务器是网络环境中的高性能计算机，不属于嵌入式系统。所以本题选 B。

25. 下面关于嵌入式系统逻辑组成的叙述中，错误的是：

A) 嵌入式系统由硬件和软件两部分组成

B) 嵌入式系统硬件的主体是中央处理器(CPU)和存储器

C) 嵌入式系统的 CPU 比通用计算机简单得多，它们都是 8 位字长的处理器

D) 嵌入式系统通过输入/输出(I/O)接口和输入/输出设备与外部世界进行联系

【解析】嵌入式系统与通用计算机一样，也由硬件和软件两部分组成。硬件的主体是 CPU 和存储器，它们通过 I/O 接口和 I/O 设备与外部世界联系。现在嵌入式系统中使用最多的还是 8 位和 16 位 CPU，但 32 位是技术发展的主流。故本题选 C。

26. 按照软硬件技术的复杂程度嵌入式系统分为低端系统、中端系统和高端系统三大类，下面有关低端系统的叙述中错误的是：

A) 硬件大多采用 4 位或 8 位单片机

B) 由监控程序对系统进行控制，不使用操作系统

C) 家用洗衣机、吸尘器、电磁炉等属于低端嵌入式应用系统

D) 它们正在被 32 位的高端系统所取代

【解析】嵌入式系统的分类有多种。按系统的软硬件技术复杂度，嵌入式系统分为低端系统、中端系统和高端系统。其中低端系统硬件大多采用 4 位或 8 位单片机，不使用操作系统，由监控程序对系统进行控制，在工控领域和白色家电领域占主导地位。选型中 D 项表述错误，故选 D。

27. 片上系统（SOC 或 SoC）是目前广泛使用的一种嵌入式处理芯片，下面有关叙述中错误的是：

A) SoC 是电子设计自动化水平提高和大规模集成电路制造技术发展的成果

B) SoC 芯片既包含处理器又包含存储器，既有数字电路也有模拟电路，单个芯片就能实现数据的采集、转换、存储、处理和 I/O 等多种功能

C) SoC 已成为集成电路设计的发展趋势， 32 位嵌入式处理芯片大多是 SoC

D) 智能手机已经广泛采用 SoC，平板电脑大多还使用传统的 Intel 处理器

【解析】随着电子设计自动化水平的提高和 VLSI 制造技术的飞速发展，半导体加工已经从微米、亚微米进入到深亚微米的时代，单个芯片上可以集成几亿个甚至几十亿个晶体管，因而能够把计算机或其他一些电子系统的全部电路都集成在单个芯片上，这种芯片就是所谓的片上系统。SoC 芯片中既包含数字电路，也可以包含模拟电路，甚至还能包含数字/模拟混合电路和射频电路。由于 SoC 将嵌入式系统的几乎全部功能都集成在一块芯片中，单个芯片就能实现数据的采集、转换、存储、处理和 I/O 等多种功能。目前，大多数 32 位的嵌入式处理芯片均为 SoC，SoC 逐渐成为集成电路设计的主流发展趋势。D 选项中平板电脑使用的也是 SOC 技术，故本题选 D。

28. 下面与嵌入式处理器有关的叙述中，错误的是：

A) 嵌入式处理器本身就是一个嵌入式最小硬件系统

B) 嵌入式处理器只有在供电的情况下才有可能正常工作

C) 嵌入式处理器工作时需要时钟信号

D) 大多数基于 ARM 处理器核的嵌入式处理器芯片都有调试接口

【解析】嵌入式处理器本身是不能独立工作的，必须给它供电，加上时钟信号，提供复位信号等才可能工作；嵌入式最小硬件系统一般包括嵌入式处理器、时钟电路、电源电路、复位电路、存储器和调试测试接口；而大多数基于 ARM 处理器核的处理器芯片都有调试接口。故本题 A 项错误。

29. 下面关于引导加载程序（Bootloader）的叙述中，正确的是：

A) 引导加载程序是硬件发生故障后由 OS 启动执行的

B) 加载和启动操作系统是引导加载程序的一项重要任务

C) Bootloader 包含加电自检和初始化程序，不包含设备驱动程序

D) 相同体系结构的硬件平台一定使用相同的引导加载程序

【解析】嵌入式系统加电后执行的第一批最初操作称为引导或者自举（Boot），对应的程序称为引导程序或者加载程序，其英文术语是 Bootloader；引导加载程序主要完成内存加电自检、外设存在自检、内存地址映射、初始化外围设备、内存寻址定位、加载和启动操作系统。由于硬件平台的不同，每种平台的引导程序也有所差异。故本题选 B。

30. 在开发低端、中端、高端等类型的嵌入式系统时，一般都需要选择和利用合适的开发平台来进行。下面有关嵌入式系统开发平台的叙述中，不正确的是：

A) 开发平台中的软件开发工具，通常会包括：项目管理器、编辑器、编译器、连接器等

B) 开发平台中的软件开发工具往往都作为一个整体提供给开发人员使用，以提高开发工作效率。

C) 有的开发平台中还包含一些中间件和软件组件，以满足特定应用领域的各种应用开发。

D) 用开发平台所开发出的低端嵌入式系统应用软件，必须基于某一个嵌入式操作系统上运行。

【解析】嵌入式系统的开发平台包含大量开发工具，软件开发工具通常包括：项目管理器、编辑器、编译器、连接器、定位器等；这些软件开发工具往往都使用统一的用户界面并作为一个整体提供给开发人员使用，以提高开发工作效率；有的开发平台中还包含一些中间件和软件组件，以满足特定应用领域的各种应用开发；嵌入式系统的开发平台大多采用宿主机-目标机的架构，宿主机是开发用机，目前大多数运行 Windows 操作系统，

而应用软件的开发和调试都是通过宿主机开完成。故本题选 D。

31. 下面关于 JTAG 的叙述中，错误的是：

A) JTAG 是 ARM 内核独有的一种测试接口，其他种类的嵌入式处理器一般没有该接口

B) JTAG 可用于实现嵌入式系统的在线编程功能

C) 多个器件可以通过 JTAG 接口串联在一起，形成一个 JTAG 链

D) SWD 是 Cortex-M 内核提供的功能与 JTAG 类似的调试接口

【解析】JATG 可用于实现嵌入式系统的在线编程功能，其标准允许多个芯片通过 JTAG 接口串联在一起，实现对多个器件的测试；目前大多数嵌入式 CPU、DSP、FPGA 器件都支持 JTAG 标准；SWD 是 Cortex-M 内核提供的功能与 JTAG 类似的调试接口。故本题选 A。

32. 1

33. 1

34. 1

35. 1

36. 1

37. 1

38. 1

39. 1

第 2 章　ARM 微处理器概述与编程模型

教材 P36：1-8

40. 简述 ARM 和 Thumb 状态的区别及如何进行状态转换。（教材 P36-1）

答：ARM 状态：执行 ARM 指令集

Thumb 状态：执行 Thumb 指令集

使用跳转指令 BX/BLX

格式：BX/BLX　Rm　当 Rm[0]为1时，从 ARM 态跳转到 Thumb 态

41. 简述 ARM9 处理器的内部寄存器结构，并分别说明 R13、R14、R15 寄存器的作用。（教

材 P36-2）

42. ARM 体系结构支持的异常类型有哪些？说明各种异常的向量地址。（教材 P36-8）

答:复位：当发生复位异常时，处理器立即停止当前程序，进入禁止中断的管理模式，并从地址0x00000000或0xFFFF0000处开始执行。

未定义指令：ARM 处理器认为当前指令未定义时，便产生了未定义指令中断。该异常可用于协处理器软件仿真。

软件中断：当用户模式下的程序使用指令 SWI 时，处理器便产生软件中断，进入管理模式，以调用特权操作。

指令预取中止：当处理器预取指令的地址不存在，或该地址不允许当前指令访问，存储器会向处理器发出中止信号；只有当预取的指令被执行时，才会产生指令预取中止异常。

数据访问中止：若处理器数据访问指令的地址不存在，或该地址不允许当前指令访问时，就会产生数据中止异常。

外部中断请求：当处理器的外部中断请求引脚有效，而且 CPSR 中的 I 位为0时，就会产生 IRQ 异常。系统的外设可通过该异常请求中断服务。

快速中断请求：处理器的快速中断请求引脚有效，而且 CPSR 中的 F 位为0时，将产生 FIQ 异常。

异 常 类 型	工 作 模 式	特定地址（低端）	特定地址（高端）	优 先 级
复位	管理模式	0x00000000	0xFFFF0000	1
未定义指令	未定义指令中止模式	0x00000004	0xFFFF0004	6
软件中断（SWI）	管理模式	0x00000008	0xFFFF0008	6
指令预取中止	中止模式	0x0000000C	0xFFFF000C	5
数据访问中止	中止模式	0x00000010	0xFFFF0010	2
外部中断请求（IRQ）	外部中断模式	0x00000018	0xFFFF0018	4
快速中断请求（FIQ）	快速中断模式	0x0000001C	0xFFFF001C	3

43. 简述 ARM 微处理器的七种工作模式。（教材 P36-4）

答：ARM 微处理器支持7种工作模式，分别为：

1）用户模式（usr）：ARM 处理器正常的程序执行状态；

2) 快速中断模式（fiq）：用于高速数据传输或通道管理；

3) 外部中断模式（irq）：用于通用的中断处理；

4) 管理模式（svc）：操作系统使用的保护模式；

5) 数据访问终止模式（abt）：当数据或指令预取终止时进入该模式 ，用于虚拟存储及存

储保护 ；

6) 系统模式（sys）：运行具有特权的操作系统任务；

7) 未定义指令中止模式（und）：当未定义指令执行时进入该模式， 可用于支持硬件协处理器的软件仿真

44. ARM 处理器的异常有哪几种，分别进入哪种工作模式，每种异常的返回功能分别采用什么指令？

答：1）复位异常（管理模式）；无返回

2）未定义指令异常（未定义模式）；返回指令 MOVS PC,R14

3）软件中断（SWI）异常（管理模式）；返回指令 MOVS PC,R14

4）指令预取中止异常（中止模式）；返回指令 SUBS PC,R14,#4

5）数据访问中止（中止模式）；返回指令 SUBS PC,R14,#8

6）快速中断请求（FIQ）（FIQ 模式）；返回指令 SUBS PC,R14,#4

7）外部中断请求（IRQ）（IRQ 模式）。返回指令 SUBS PC,R14,#4

45. ARM 处理器结构体系中具有 T 变种处理器核可以工作在 ARM 状态和 Thumb 状态。（ √ ）

46. ARM9 处理器的当前程序状态寄存器结构如图所示，请说明各位的功能。

31	30	29	28	27	26		8	7	6	5	4	3	2	1	0
N	Z	C	V	—	—	...	—	I	F	T	M4	M3	M2	M1	M0

答：N：负标志位，运算结果的第31位值，记录标志设置的结果。

Z：零标志位，如果标志设置操作的结果为0，则置位。

C：进位标志位，记录无符号加法溢出，减法无错位，循环移位。

V：溢出标志位，记录标志设置操作的有符号溢出。

I：中断禁止标志位，置位时禁止 IRQ 中断，否则允许 IRQ 中断使能。

F：中断禁止标志位，置位时禁止 FIQ 中断，否则允许 FIQ 中断使能。

T：控制位，置位时处理器运行在 Thumb 状态下，清零时处理器运行在 ARM 状态下。

M0~M4：模式控制位，定义处理器的7中模式。

47. S3C2440 具有多少通用 I/O 口？它们具有哪些功能？

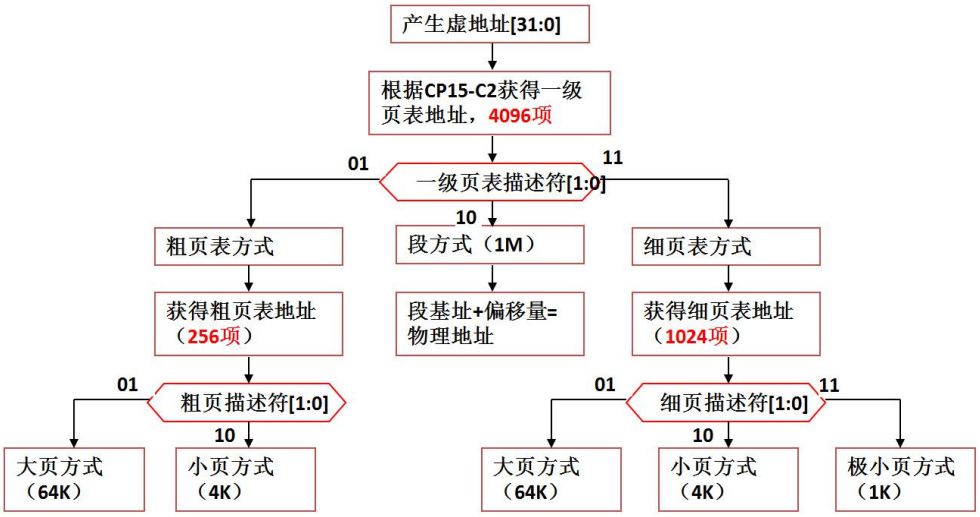
48. CPSR 是当前程序状态字，其作用是保存状态和工作模式。 √

49. ARM 处理器复位后，强制为管理模式，并进入 ARM 状态，进而执行。 √

50. S3C2440 具有 70 个中断源。 ×

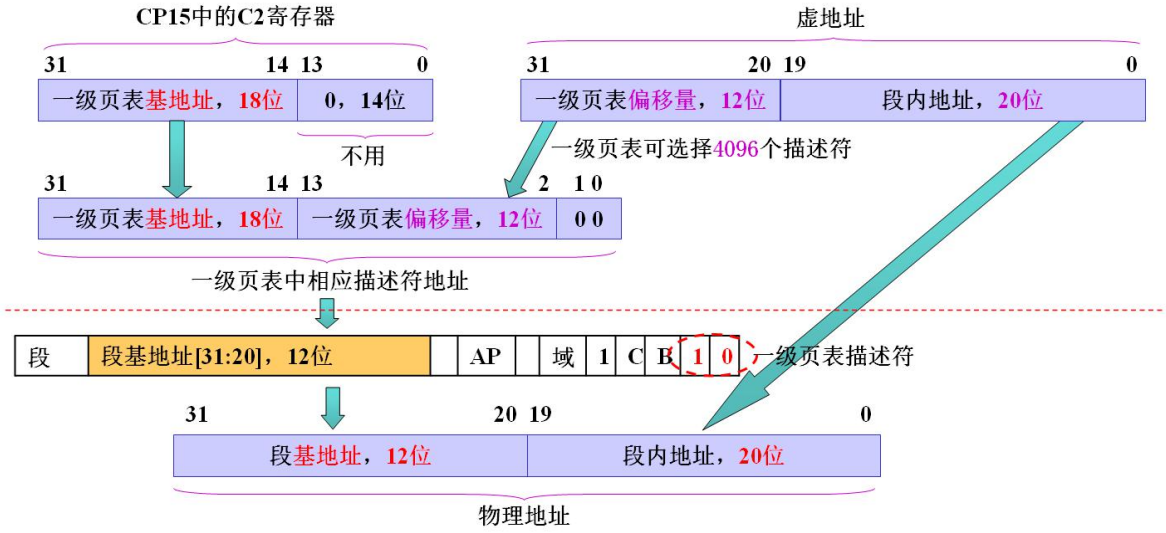
51. 画图说明 S3C2440 MMU 地址变换流程。

答：



52. 画图说明 S3C2440 MMU 段地址变换过程。

答：



53. CPU 和内存之间进行地址转换时，_____将地址从虚拟地址空间映射到物理地址空间。A

A、MMU　　B、TCB　　C、DMA　　D、CACHE

54. 以冯.诺伊曼结构为体系结构的是: _____。A

A、ARM7　　B、ARM9　C、ARM10　　D、ARM11

55. 下面哪一种工作模式不属于 ARM 特权模式_____。A

A、用户模式　　B、管理模式　C、软中断模式　　D、FIQ 模式

56. ARM7TDMI 中的 T、D、S、I 分别表示什么含义？

答：T：支持高密度16位的 Thumb 指令集

D：支持片上调试

S：ARM7TDMI 的可综合（synthesizable）版本（软核）

I：支持 EmbededICE 观察硬件

M：支持64位乘法

57. .ARM 处理器有几种工作状态，各自的特点。工作状态之间如何进行转换，异常响应时，处理器处于何种状态。

答：ARM 有两种工作状态：

①ARM 状态,此时处理器执行32位的字对齐的 ARM 指令。

②Thumb 状态，此时处理器执行16位的、半字对齐的 Thumb 指令. 在程序的执行过程中,微处理器可以随时在两种工作状态之间切换,并且不影响处理器运行模式和相应寄存器中的内容。ARM 指令集和 Thumb 指令集均有切换处理器状态的指令，并可在两种工作状态之间切换，但 ARM 微处理器在上电或复位后，应该处于 ARM 状态。

执行 BX 跳转指令，将操作数的状态位（位0）设置为1时，可以使处理器从 ARM 状态切换到 Thumb 状态。此外，当处理器处于 Thumb 状态时发生异（如 IRQ、FIQ、Undef、Abort、SWI 等），则异常处理返回时，自动切换到 Thumb 状态。

执行 BX 跳转指令，将操作数的状态位（位0）设置为0时，可以使处理器从 Thumb 状态切换到 ARM 状态。此外，当处理器进行异常处理时,把 PC 指针放入异常模式链接寄存器中,并从异常向量地址开始执行程序,也可以使处理器切换到 ARM 状态。

THUMB 指令集在功能上只是 ARM 指令集的一个子集,某些功能只能在 ARM 状态下执行,如 CPSR 和协处理器的访问.

进行异常响应时,处理器会自动进入 ARM 状态.

即使是一个单纯的 THUMB 应用系统,必须加一个汇编的交互头程序,因为系统总是自动从 ARM 开始启动。

58. ARM 处理器的寄存器 R13、R14、R15 的专用功能各是什么？寄存器 CPSR、SPSR 的功能各是什么？

答：1）寄存器 R13保存堆栈指针 SP；

2）寄存器 R14用作子程序链接寄存器，也称为 LR ，用以保存返回地址；

3）R15（PC）用作程序计数器。

4）CPSR 包含条件码标志、中断禁止位、当前处理器模式以及其它状态和控制信息。所有处理器模式下都可以访问当前的程序状态寄存器 CPSR。

5）在每种异常模式下都有一个对应的物理寄存器——程序状态保存寄存器 SPSR。当异常出现时，SPSR 用于保存　　CPSR 的状态，以便异常返回后恢复异常发生时的工作状态。

59. IRQ 中断处理程序可以执行指令 SUBS　PC , R14_irq , #4 从 IRQ 中断返回，说明指令中减 4 的原因。

答：三级流水线，产生 IRQ 后，PC 已经更新。

60. 简述 ARM 处理器对异常中断的响应过程。

答：返回地址、当前状态、修改模式。

61. 说明 ARM 寄存器中 R13，R14，R15 的特殊作用。

答：堆栈指针（SP），链接地址寄存器（LR），程序计数器（PC）

62. ARM 采用 RISC 体系结构。√

63. ARM 是指 Advanced RISC Machines 先进精简指令集机器。√

64. 哈佛体系结构为数据和程序提供了各自独立的存储器，程序计数器只指向程序存储器而不指向数据存储器。√

65. 小端模式含义是：从存储器读数据或向存储器写数据，字数据与存储器存储对应关系为低存低；高存高。√

66. 当 ARM 微处理器执行 16 位的 Thumb 指令集时，工作在 Thumb 状态。√

67. ARM7 是一种低电压、通用 32 位 RISC 微处理器单元。√

68. ARM 指令都是 32 位的字，必须以字为单位边界对齐。√

69. S3C2440 共有 37 个寄存器，各寄存器均为 32 位。√

70. ARM 处理器的工作状态包括（　　）。C

A．ARM 状态、CPU 状态　　B．CPU 状态、存储器状态

C．ARM 状态、Thumb 状态　　D．存储器状态、Thumb 状态

71. ARM 处理器复位后，执行（　　）。C

A．强制为用户模式，并进入 ARM 状态，进而执行

B．强制为用户模式，并进入 Thumb 状态，进而执行

C．强制为管理模式，并进入 ARM 状态，进而执行

D．强制为管理模式，并进入 Thumb 态，进而执行

72. ARM 处理器异常的优先级如何定义？

答：异常中断的优先级：复位（最高优先级）--> 数据异常中止--->FIQ --> IRQ---> 预取指异常中止--->SWI---->未定义指令（包括缺协处理器）。

73. 若寄存器 R1=0x01020304，分别按小端模式和大端模式存储在 0x30000 字单元中，试分别列出两种模式下内存存储内容，并标出内存地址。

74. 假设 R0 的内容为 0x1000，寄存器内容 R1=0x11223344、R2=0x55667788，按小端存储模式，所有存储器字节单元内容为 0。连续执行下述指令后，分析每条指令执行后，寄存器 R0、R1、R2 内容如何变化，存储器相关字节单元内容如何变化。

```
STMIB  R0!,{R1,R2}
LDMIA  R0!,{R1,R2}
```

答：

（1）执行 STMIB R0!,{R1,R2}后：

R0=0x1008

R1=0x11223344（不变）

R2=0x55667788（不变）

产生变化的存储器字节单元：

存储器字节单元（0x1004）=0x44

存储器字节单元（0x1005）=0x33

存储器字节单元（0x1006）=0x22

存储器字节单元（0x1007）=0x11

存储器字节单元（0x1008）=0x88

存储器字节单元（0x1009）=0x77

存储器字节单元（0x100A）=0x66

存储器字节单元（0x100B）=0x55

（2）继续执行 LDMIA R0!,{R1,R2}后：

R0=0x1010

R1=0x55667788

R2=0

存储器字节单元不变化。

75. ARM 处理器模式和 ARM 处理器状态有什么区别？说明具体的工作模式和工作状态。

答：ARM 处理器模式指用户模式、快中断模式、中断模式、管理模式、中止模式、未定

义模式和系统模式。

ARM 处理器状态指 ARM 状态和 Thumb 状态。

ARM 两种处理器状态下均有上述7种模式。

ARM 有7种处理器模式。

用户模式：正常程序运行的工作模式，不能直接从用户模式切换到其它模式

系统模式：用于支持操作系统的特权任务等，可以直接切换到其它模式

快中断模式：用于快速中断处理，支持高速数据传输及通道处理，只有在 FIQ 异常响应时，才进入此模式。

中断模式：用于通用中断处理，只有在 IRQ 异常响应时，才进入此模式。

管理模式：供操作系统使用的一种保护模式，只有在系统复位和软件中断响应时，才进入此模式。

中止模式：用于虚拟内存和/或存储器保护。

未定义模式：支持软件仿真的硬件协处理器，只有在未定义指令异常响应时，才进入此模式。

ARM 有2种工作状态：

ARM 执行 ARM 指令集，则处于 ARM 状态。

ARM 执行 Thumb 指令集，则处于 Thumb 状态。

76. ARM9 中 CPSR 的作用是什么？ARM9 条件码 GE 的含义是什么？

答：CPSR 作用：当前程序状态字，保存状态和工作模式。

GE 含义：有符号大于等于。

77. 简述 ARM 处理器的小端模式和大端模式存储器组织。

答：1)小端存储器组织是较高的有效字节存放在较高的存储器地址，较低的有效字节存放在较低的存储器地址。

2)大端存储器组织是较高的有效字节存放在较低的存储器地址，较低的有效字节存放在较高的存储器地址。

78. 简述冯•诺依曼结构与哈佛结构，并指出 ARM9 处理器属于哪种结构。

答：哈佛体系结构：独立的程序存储器和数据存储器。

冯•诺依曼结构：将数据和指令都存储在一个统一的存储器中。

ARM9处理器属于哈佛结构。

79. 当 ARM 微处理器执行 16 位的 Thumb 指令集时，工作在 Thumb 状态。√

80. ARM7 是一种低电压、通用 32 位 RISC 微处理器单元。√

81. ARM 指令都是 32 位的字，必须以字为单位边界对齐。√

82. S3C2440 共有 37 个寄存器，各寄存器均为 32 位。√

83. CPSR 寄存器中的 Z 位为 1 表示_____。运算结果为零

84. 复位异常时，处理器立即停止当前程序，进入禁止中断的管理模式，并从地址_____处开始执行。0X00000000

85. 在 ARM 处理器中，（　　）寄存器包括可控的全局中断禁止位，控制中断禁止位就可以打开或者关闭中断。A

A．CPSR　　B．SPSR　　C．PC　　D．IR

86. 说明大端模式和小端模式的区别。

答：小端模式：从存储器读数据或向存储器写数据，字数据与存储器存储对应关系为低存低；高存高；

大端模式：从存储器读数据或向存储器写数据，字数据与存储器存储对应关系为低存高；高存低。

87. CPSR 寄存器中 C 标志位的含义是（　　）。D

A．状态选择　　B．中断允许　　C．符号标志　　D．进位标志

88. 哈佛体系结构和冯·诺依曼体系结构有何不同？ARM7、ARM9 处理器各属于哪个体系结构？

答：1）冯·诺依曼机：将数据和指令都存储在存储器中的计算机。

哈佛机：为数据和程序提供了各自独立的存储器。

2）ARM7使用冯·诺依曼体系结构。

ARM9使用哈佛体系结构。

89. ARM7 微处理器有哪两种工作状态？其中哪种工作状态对寄存器的访问受到一定限制，并说明受到怎样的限制。

答：1） ARM 状态，Thumb 状态

2）Thumb 状态对寄存器的访问受到一定限制

3）Thumb 指令对 R8-R15寄存器的访问受到一定限制。

90. 说明 ARM 的 3 级流水线和 5 级流水线。

91. ARM 存储数据类型有哪几种？

92. ARM 处理器的数据存储格式有哪几种？并简要说明。

93. 解释中断处理中 ISR 与 FIQ 的区别

94. ARM9 处理器如何保证 FIQ 异常响应的快速性？

答：为 FIQ 设置了独立的 R8-R12寄存器，实现寄存器快速切换。

CPSR 和 SPSR 也是独立的。

95. 在 ARM 微处理器系列中，ARM9TDMI、ARM920T、ARM926EJ-S 中的字母“TDMIEJS”各自代表什么含义？

答：T：支持16为压缩指令集 Thumb，称为 T 变种；

D：支持片上 Debug，称为 D 变种；

M：内嵌硬件乘法器 Multiplier，支持长乘指令，称为 M 变种；

I：嵌入式 ICE，支持片上断点和调试，称为 I 变种；

E：表示支持增强型 DSP 指令（E 变种）；

J：表示支持 Java 加速器 Jazelle（J 变种）；

-S 表示是 ARM 可综合版本。

96. FIQ 中断的入口地址是（ A）。

A、0x0000001C B、0x00000008 C、0x00000018 D、0x00000014

97. 寄存器 R15 除了可以做通用寄存器外，还可以做（ A）

A. 程序计数器 B. 链接寄存器 C. 堆栈指针寄存器 D. 基址寄存器

98. 下列关于存储管理单元（MMU）说法错误的是（B）。

A. MMU 提供的一个关键服务是使各个任务作为各自独立的程序在其自己的私有存储空间中运行。

B. 在带 MMU 的操作系统控制下，运行的任务必须知道其他与之无关的任务的存储需求情况，这就简化了各个任务的设计。

C. MMU 提供了一些资源以允许使用虚拟存储器。

D. MMU 作为转换器，将程序和数据的虚拟地址（编译时的连接地址）转换成实际的物理地址，即在物理主存中的地址。

99. 下列 CPSR 寄存器标志位的作用说法错误的是（D）。

A. N：负数　　B. Z: 零　　C. C: 进位　　D. V: 借位

100. ARM 处理器有几种工作模式？并做说明每种工作模式的含义。

答：ARM 处理器有7种工作模式:

用户模式(usr)- 正常程序执行的模式

快速中断模式(fiq)- FIQ 异常响应时进入此模式

中断模式(irq)- IRQ 异常响应时进入此模式

管理员模式(svc)- 系统复位和软件中断响应时进入此模式

中止模式(abt)- 用于虚拟存储及存储保护　　系统模式(sys)- 与用户类似，但有直接切换到其它模式等特权

未定义模式(und)- 未定义指令异常响应时进入此模式

除了用户模式外，其他模式均可视为特权模式

101.对于 ARM 系列，最适合高端应用的嵌入式处理器是:

A) ARM9　　B) ARM Cortex-M　　C) ARM Cortex-A

D) ARM Cortex-R

【解析】ARM 公司在经典处理器 ARM11 以后的产品改用 Cortex 命名，并分成 A、R 和 M 三类，旨在为各种不同的市场提供服务。其中："A"系列面向尖端的基于虚拟内存的操作系统和用户应用；"R"系列针对实时系统；"M"系列对微控制器。故本题选 C。

102.以下关于 ARM 程序状态寄存器 CPSR 说法错误的是：

A) CPSR 记录了 ARM 运行过程中的标志状态

B) CPSR 决定是否切换到 Thumb 状态

C) CPSR 决定是否允许快速中断 FIQ

D) CPSR 决定堆栈深度

【解析】CPRS 为当前程序状态寄存器，记录了 ARM 运行过程中的标志状态；其中 T 为 ARM 与 Thumb 指令切换，F 为禁止快速中断 FIQ 的控制位。因此 A、B、C 项正确，本题选 D。

103.关于 ARM 处理器异常中断响应过程中，以下说法正确的是：

A) SPSR 的值保存到 CPSR 中

B) 设置当前状态寄存器 CPSR 的相应位

C) 断点地址会自动保存在 R13 中

D)　自动把异常向量地址写入 R14 中

【解析】ARM 对异常的响应过程如下：将 CPRS 的值保存到将要执行的异常中断对应的各自 SPSR 中；设置 CPSR 的相应位；将引起异常指令的下一条地址（断点地址）保存到 R14 中；给 PC 强制赋值，转入向量地址，以便执行相应的处理程序。故本题 B 项正确。

104.关于 ARM 处理器的 MMU，以下说法错误的是：

A)　MMU 是存储器管理部件　B)　MMU 控制存储器访问顺序

C)　MMU 控制存储器的访问权限　D)　MMU 通过查 TLB 表得到虚拟地址

【解析】MMU 是 Memory　Management　Unit 的缩写，中文名是内存管理单元，它是 CPU）中用来管理虚拟存储器、物理存储器的控制线路，同时也负责虚拟地址映射为物理地址，以及提供硬件机制的内存访问授权；MMU 进行虚拟地址到物理地址的转换通过查找页表来完成，每次在访问内存时先查 TLB，查不到时再到内存中去查整个页表。故 MMU 通过查 TLB 表得到的是物理地址，D 项错误。

105.下面关于 AMBA 的叙述中，错误的是：

A)　ARM 公司定义的 AMBA 其中文名为"先进微控制器总线体系结构"

B)　ARM 公司定义的 AMBA 是用于连接和管理片上系统中各功能模块的开放标准和片上互连规范

C)　至 2011 年，AMBA 已从 AMBA1.0 发展到了 AMBA4.0

D)　ARM7 和 ARM11 采用的 AMBA 的版本相同

【解析】AMBA　是 ARM 公司公布的总线协议，其中文名为"先进微控制器总线体系结构"；用于连接和管理片上系统中各功能模块的开放标准和片上互连规范；AMBA 有多个版本，至 2011 年，AMBA 已从 AMBA1.0 发展到了 AMBA4.0，性能随版本的发展而逐步提高，ARM7 采用 AMBA1，而 ARM9 采用 AMBA2。故本题选 D。

106.下面与嵌入式处理器复位相关的叙述中，错误的是：

A)　一般情况下,为保证系统可靠复位，复位信号有效电平的时间宽度必须为若干个处理器时钟周期

B)　ARM 复位后 PC 指针指向的地址是可选的

C)　嵌入式系统可使用外接典型复位芯片来保证系统可靠复位

D)　当嵌入式处理器的复位引脚标记为 nRESET 时，表示低电平复位

【解析】一般情况下,为保证系统可靠复位，复位信号有效电平的时间宽度必须为若干个处理器时钟周期；嵌入式系统可使用外接典型复位芯片来保证系统可靠复位；嵌入式处理器都有一个系统复位引脚为 nRESET 或 RESET，n 表示低电平复位，不带 n 的表示高

电平复位；ARM 复位后 PC 无条件的指向 0x00000000 处。故 B 项错误。

107.下面是关于基于 ARM 内核的嵌入式芯片中的中断控制器的叙述，其中错误的是：

A) 中断控制器是连接 AMBA 的系统总线和外围总线的桥接器

B) 一般采用向量中断或嵌套向量中断方式管理中断

C) 向量中断区分中断的优先级，并且每个中断都有各自的中断处理程序地址

D) 高优先级的中断可以进入低优先级中断的处理过程中，待高优先级中断处理完成后再继续执行低优先级中断处理

【解析】连接 AMBA 的系统总线和外围总线的是桥接器，DMA 连接在 AMBA 的系统总线上，故 A 错误；DMA 一般采用向量中断或嵌套向量中断方式管理中断，向量中断区分中断的优先级，并且每个中断都有各自的中断处理程序地址，高优先级的中断可以进入低优先级中断的处理过程中，待高优先级中断处理完成后再继续执行低优先级中断处理。故本题选 A。

108.

109.

110.

111.

112.

113.

114.1

115.1

116.1

117.

第 3 章　ARM9 指令系统

教材 P59：1，4，6-10

118.简述 ARM9 的 LDM/STM 指令中空、满、递增、递减的含义。（教材 P60-6）

119.BIC 指令的作用是什么？（教材 P60-8）

120.BX 和 BL 指令有什么不同？（教材 P60-10）

121.指令 ADDS　R0，R1，R2 中，S 的含义是“加法运算影响程序状态标志”。√

122. MOV　R0，#0x2468 这个指令有错误吗？为什么？如果有错请提出修改意见。

答：错误。因为正确条件是首尾环24个零，偶数位置。可改为 LDR　R0，=0x2468

123.在 ARM 汇编编程中，若将地址为 UARTADD 的外设端口数据读到 R0 中，如何设计指令？

答：

```
LDR     R1，=UARTADD
LDR     R0，  [R1]
```

124.指令 LDR {<cond>}<Rd>, <addressing_mode>的操作伪代码描述如下：

```
if   ConditionPassed(cond)   then
    if   adderss[1:0] = = 0b00   then
            Value = Memory[adderss,4]
    else   if   adderss[1:0] = = 0b01   then
            Value = Memory[adderss,4] Rotate_Right 8
    else   if   adderss[1:0] = = 0b10   then
            Value = Memory[adderss,4] Rotate_Right 16
    else   if   adderss[1:0] = = 0b11   then
            Value = Memory[adderss,4] Rotate_Right 24
    if   (Rd is R15)   then
        if (architecture version 5 or above)   then
            PC = value AND 0xFFFFFFFE
            T Bit = value[0]
        else
            PC = value AND 0xFFFFFFFC
  else
        Rd = value
```

回答下列问题：

1. 分析或说明指令的具体操作过程。
2. 说明 T Bit = value[0]的目的。

答：

1.操作过程：

判断指令执行的条件。

判断 adderss[1:0] 的值。

adderss[1:0] = = 0b00，直接加载内存中的值。

若 adderss[1:0] = = 0b01，先将内存中的数据循环右移8位，再加载

若 adderss[1:0] = = 0b10，先将内存中的数据循环右移16位，再加载

若 adderss[1:0] = = 0b11，先将内存中的数据循环右移24位，再加载

若加载目标为 PC，则依 ARM 和 Thumb 状态处理。

2. T Bit = value[0]的目的是设置 ARM 或 Thumb 状态

125.对大端模式，R0=0x11223344，先执行 STR R0 ,[R1] 指令，后执行 LDRB R2,[R1] 指令,R2 的值是_____。0x11

126.已知寄存器存放的数据为 r13=0x40001000，r1=0x01，r2=0x02，r3=0x03。分析执行指令 STMFD r13!，{r1-r3}后，r13 的值是多少，并说明该指令引起的堆栈存储区的变化（即堆栈存储单元地址与入栈数据的对应关系，比如，堆栈存储区字单元 0x40001000 地址存放的数据是 0x01，可以表示为[0x40001000]=0x01，也可以画图说明）。

答：r13=0x40000FF4， [0x40000FF4]=0x01，[0x40000FF8]=0x02，[0x40000FFC]=0x03

127.写出可以替换伪指令 LDR R0, =0x12 的指令_____。MOV R0，#0X12

128. LDMEA 指令,其中 EA 指的是_____。D

A、满递减堆栈　　B、满递增堆栈

C、空递减堆栈　　D、空递增堆栈

129.下列指令错误的是_____。B

A、MOV R1， #128　　B、MOV R1，0x3FF

C、LDR R1，=0x1128　　D、LDR R1，0x3F

130.已知从存储器字单元（32 位字）0x40001000 地址开始，依次存放的数据为[0x40001000]=0x01，[0x40001004]=0x02，[0x40001008]=0x03，寄存器存放的数据为 r13=0x40001000，r1=0x00000000，r2=0x00000000，r3=0x00000000。分析执行指令 LDMFD r13!，{r1-r3}后，r13、r1、r2、r3 的值分别是多少？

答：r13=0x4000100C，r1=0x01，r2=0x02，r3=0x03

131.标号 L1 处为 Thumb 指令，当前状态为 ARM 状态，写出转移到 L1 处使用的指令。

答：ADR R0,L1+1

BX R0

132. S3C2440 的处理器对内存的访问只能通过 Load/Store 指令来实现。（ √ ）

133.由于 CPU 内部寄存器的访问速度较高，根据 ATPC 标准，应尽可能使函数的参数控制在 4 个以下。（ √ ）

134.当发生取指中止异常时，将当前的 PC 存入 LR，执行取指中止异常中断服务程序；写出当从中断服务程序返回时使用的指令。

答：SUBS PC,R14,#4

135.参考 CPSR 寄存器中各标志位的含义，使处理器工作在系统模式下。请补全程序中的操作码。

```
___   R0, CPSR
AND   R0, R0,#0xFFFFFFE0
ORR   R0, R0,#0x1F
___   CPSR_fsxc, R0
```

答：MRS，MSR

136.下面代码实现什么功能

```
EOR  R1, R0, R0, ROR #16 ;
BIC  R1, R1, #0xFF0000
MOV  R0, R0, ROR #8
EOR  R0, R0, R1, LSR #8
```

答：ABCD→DCBA

137.简述 ARM 处理器的寻址方式，并回答在 ATPCS 规则中，规定数据栈采用那种类型。

答：ARM 指令系统支持以下8种寻址方式：

寄存器寻址；立即寻址；寄存器间接寻址；变址寻址；寄存器移位寻址；多寄存器寻址；相对寻址；堆栈寻址。

在 ATPCS 规则中，规定数据栈采用满递减类型。

138.ARM 处理器的寻址方式有哪些？各写一条说明。

答：1）寄存器寻址，ADD R0，R1，R2

2）立即寻址，ADD R0，R0，＃1

3）寄存器间接寻址，LDR R0，[R1]

4）变址寻址，LDR R0，[R1，＃4]

5）寄存器移位寻址，ADD　R0，R1，R2，ROR　＃5

6）多寄存器寻址，LDMIA　R0，{R1-R5}

7）堆栈寻址，STMFD　SP!，{R1-R7，LR}

8）相对寻址，BEQ　process1

139.假设 R0 的内容为 0x8000，寄存器 R1、R2 的内容分别为 0x01 与 0x10，存储器内容为 0。连续执行下述指令后，说明每条指令执行后 PC 如何变化？存储器及寄存器的内容如何变化？

STMIB R0!,{R1,R2}

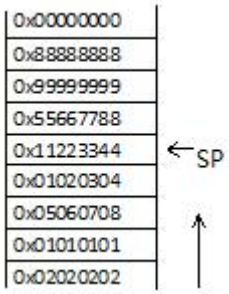
LDMIA R0!,{R1,R2}

答：执行第一条指令后，存储器中地址为0x8004保存的内容为 R1的内容，即0X01，存储器中地址为0x8008保存的内容为 R2的内容，即0X10，寄存器 R1,R2的内容不变，仍分别为0x01与0x10。R0的内容为0x8008。PC=PC+4。

执行第二条指令后，存储器内容不变，寄存器 R1保存的是存储器地址为0x8008的内容：0x10,R2保存的是存储器的地址为0x800c 的内容，即0。R0的内容为0x8010。PC=PC+4。

140.执行完 LDR R0，[R1,#0x4]!指令后，R1 的值不改变。错

141. SP 及内存关系如图所示，SP=0x100, 执行完 LDMIA SP!，{R0-R2}后 R0=?R1=?R2=?SP=?



答：SP 及内存关系如图所示，SP=0x100, 执行完 LDMIA SP!，{R0-R2}后 R0=?R1=?R2=?SP=?

R0=0X11223344

R1=0X55667788

R2=0X99999999

SP=0X10C

142.下面的指令实现什么功能？

MRS　R0，CPSR

ORR　R0，R0,0X1F

MSR　CPSR_c,R0

答：修改工作模式

143.下面程序实现什么功能？

CMP　R0，　#1

BLLT　　DOSUB1

BLGE　DOSUB2

答：根据 R0不同执行不同的子程序

144.指出下列指令中第 2 操作数的寻址方式：

（1）ADD　R4，R0，R1

（2）ADD　R5，R0，#0x66

答：（1）寄存器寻址

（2）立即寻址

145. R0 的内容为#20，R1 的内容为#10 请问下面语句执行完以后，R0 和 R1 的值是多少？

CMP　R0，#10

ADDNE　R0，R0，R1

答：R0的内容为#30，R1的内容为#10。

146.指令 ADDS　R0，R1，R2 中，S 的含义是（　　）。A

A．加法运算影响程序状态标志　　B．无符号数加法运算

C．有符号数加法运算　　D．无意义

147.标号 L1 处为 Thumb 指令，程序当前运行状态为 ARM 状态，则转移到 L1 使用（　　）。

B

A．B　L1

B．ADR　R0，L1+1

BX　　R0

C．BX　L1

D．ADR　R0，L1

BX　　R0

148.假定 R0、R1 中的内容为无符号数，写出指令实现“若 R0 的内容和 R1 的内容相等，则转去执行 XYZ”的判断。

答：CMP R0, R1

BEQ XYZ

149.指出下列指令的寻址方式：

（1）ADD　R1，R2，R3

（2）ADD　R0，R1，#0X05

（3）ADD　R0，R2，R1 LSL #3

（4）LDR　R0，[R1，#13]

（5）STMFD　R13!，{R0，R1，R2，R3，R4}

答：（1）寄存器寻址

（2）立即寻址

（3）寄存器移位寻址

（4）基址变址寻址

（5）堆栈寻址

150. R1=0X200,R0=0x2 执行 STR R0,[R1],#12 后内存 0x200 值是_____，内存 0x20C 值是_____（如不能确定值是多少填“不确定”）。0x2, 不确定

151.将 R0,R1 压入堆栈，使用指令 STMFD SP!,{R0,R1}进栈；出栈指令使用：B

A) LDMFA SP!,{R0,R1}

B) LDMFD SP!,{R0,R1}

C) LDMEA SP!,{R0,R1}

D）LDMED SP!,{R0,R1}

152.用于判断 R0 的最低位是否为 1 的指令是:A

A. TST R0,#0x01

B. TEQ R0,#0x01

C. BIC R0,R0,#0x01

D. AND R0,R0,#0x01

153.指令 LDR R2,[R0,R1]!执行后，结果如下：C

A．R0+R1地址的内容赋给 R2; R0=R0+4

B．R0地址的内容赋给 R2; R0=R0+4

C．R0+R1地址的内容赋给 R2; R0=R0+R1

D．R1地址的内容赋给 R2; R0=R0+R1

154.执行 B　LABLE 指令 将立即跳转到 LABLE 处继续执行，其中 LABLE 说法正确的是：

C

A.LABLE 是一个存储器的绝对地址。

B.LABLE 是相对于 PC 的一个偏移量，由连接器计算给出。

C.LABLE 是相对于 PC 的一个偏移量，由编译器计算给出。

D.以上都不对.

155.发生取指中止异常，执行取指中止异常中断服务程序，从中断服务程序返回使用的指令是：C

A.MOV PC,LR

B.ADDS PC,R14,#4

C.SUBS PC,R14,#4

D.SUBS PC,R14,#8

156.将常数 0x11223344 赋给寄存器 R0,使用的指令是：B

A.MOV R0,#0x11223344

B.LDR R0,=0x11223344

C.NUM EQU 0x11223344

MOV R0,NUM

D.NUM EQU 0x11223344

LDR R0,NUM

157.用汇编语言实现 128 位数的减法。

答：第一个128位数由高到低位于 R7~R4中

第二个128位数由高到低位于 R11~R8中

SUBS R0，R4，R8

SBCS R1，R5，R9

SBCS R2，R6，R10

SBC R3，R7，R11

158.分析下面程序的功能。

STMFD SP!,{R0-R6}

LDR R6,=SRC

LDMIA R6!,{R0-R5}

LDR R6,=DST

STMIA R6!,{R0-R5}

LDMFD SP!,{R0-R6}

答：将 SRC 开始的6个字传输到 DST

159.已知 R1=0x30, R5=1,R6=2,R7=0x3FC ，执行

STMIA R1!,{R7,R6,R5}

LDMDA R1!,{R5-R7}

R1,R5,R6,R7 的值分别是多少。

答：R7=随机; R6=0x3FC; R5=2;R1=0x30

160.符合什么条件的立即数是合法立即数？，0X1010 是合法立即数吗？

答：8位二进制数循环右移偶数位；不是

161.进栈指令如下: STMFD R13!,{R7，R2，R0 }，画图说明指令执行后堆栈变化。

答：注意：高寄存器存储高地址

162.说出下面程序执行什么功能？写出程序执行后 R1 的结果。

```
start
        MOV R2,#5
        MOV R1,#1
L0
        MUL R5,R1,R2
        MOV R1,R5
        SUBS R2,R2,#1
        BNE L0
```

答：5！=120

163.分析下面程序段实现什么功能。

```
ADR R0, TTCODE+1
BX R0
```

答：跳转并切换到 thumb 状态

164.指令 LDR R0,[R1,#13]中第二操作数的寻址方式是（　　）。B

A．寄存器寻址　　B．基址变址寻址

C．寄存器移位寻址　　D．立即数寻址

165.阅读程序段，并回答问题：

```
LDR　R0, =0X12345678
LDR　R1, =0X40002000
STR　R0, [R1]
```

试回答以下问题：

1）若 ARM 系统中以大端格式存储，写出字节存储单元及其对应的数据。

2）若 ARM 系统中以小端格式存储，写出字节存储单元及其对应的数据。

答：

1）大端格式：

40002000：12

40002001：34

40002002：56

40002003：78

2）小端格式：

40002000：78

40002001：56

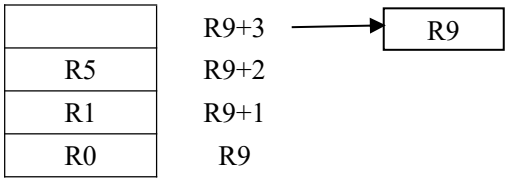
40002002：34

40002003：12

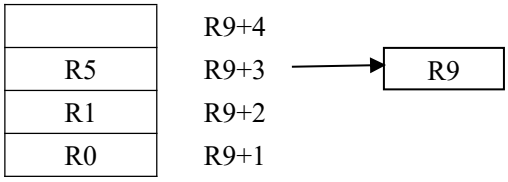
166.画图说明下列指令的区别。

```
STMIA　　R9!,　　{R0,R1,R5}
STMIB　　R9!,　　{R0,R1,R5}
STMDA　　R9!,　　{R0,R1,R5}
STMDA　　R9!,　　{R0,R1,R5}
```

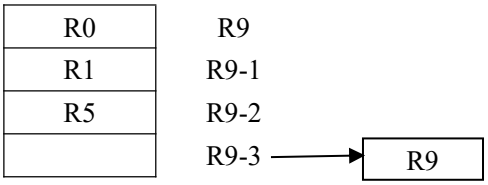
答：STMIA　　R9!,　　{R0,R1,R5}



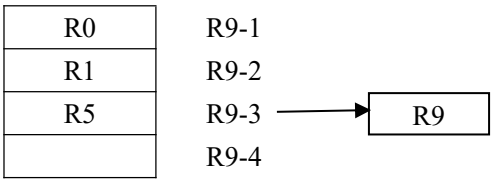
STMIB R9!, {R0,R1,R5}



STMDA R9!, {R0,R1,R5}



STMDA R9!, {R0,R1,R5}



167.在指令中加“！”和不加“！”有什么区别？如果

R1=0x100

[0x100]=0x0ff

[0x104]=0x0ee

分别执行下面每条指令，指出 R0 中存放的结果。

LDR R0, [R1, #4]

LDR R0, [R1, #4]!

LDR R0, [R1], #4

答：

加“！”和不加“！”指出是否修改基地址。

LDR R0, [R1, #4]；R0=0x0ee

LDR R0, [R1, #4]! ；R0=0x0ee

LDR R0, [R1], #4；R0=0x0ff

168.阅读下列程序，说明程序实现的功能。

```
CMP     R0,       #maxindex
LDRLO       PC, [PC,R0,LSL #2]
B           indexouttofrange
DCD     handler0
DCD     handler1
DCD     handler2
DCD     handler3
……
```

答：根据 R0中的偏移值进行程序分支

169. ARM 指令集中，大多数指令是条件执行的，这里所说的条件执行指的是什么？列举 4 个以上的条件。

答：是指满足条件指令才能执行，比如：CS、CC、EQ、NE、HI、LS、GT、LT。

170.为什么说在第二操作数中，如果使用立即数，则必须是 8 位二进制数在 32 位字中被循环右移偶数位后的值？

答：因为在第二操作数中，立即数字段被分配了12位二进制位，其中，8位是数据本身，4位是偶数位右移次数。

171.指令中 ADDS R0,R1,R2,S 的含义是（ ）。3

(1) 有符号数加法运算

(2) 无符号数加法运算

(3) 加法运算影响程序状态标志

(4) 无意义

172.下面哪个指令可以用于实现子程序调用： 3

(1) B

(2) CMP

(3) BL

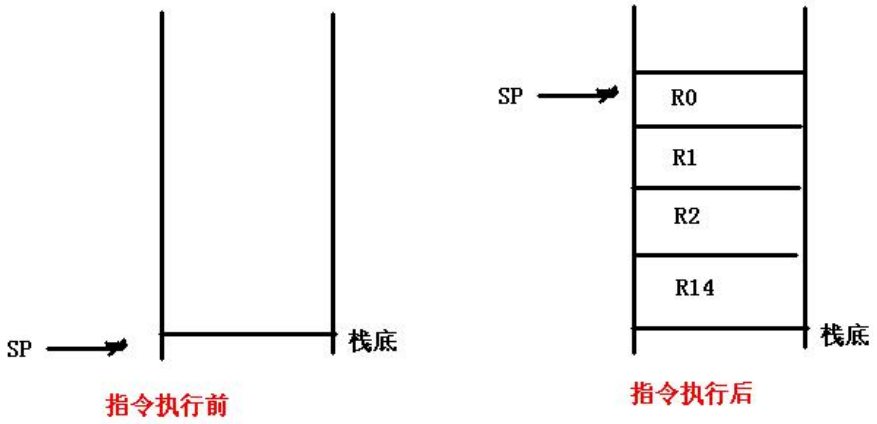
(4) LDM

173.下面哪个立即数是合法的：2

A.#0x1FF　B.#0x3FC　C.#0x1FE　D.#0x1010

174.进栈指令如下: STMFD R13!,{R14，R0-R2 }，画图说明指令执行前后堆栈变化。

答：



175.解释 ORR　R0，R0，R1，LSR #5 的含义，并指出这个指令中的 LSR #5 会不会带来额外的开销。

答：r1的值右移5位后，与 r0的内容做“或”运算，结果保存在 r0中。LSR #5不会带来额外的开销，它是本指令操作的一部分。

176.如图所示，R13=0x4010

执行 LDMDB R13！，{R0，R1，R2}后 R2，R13 的值是多少？

	0x99999999
0x401c	0x88888888
0x4018	0x77777777
0x4014	0x66666666
0x4010	0x55555555
0x400c	0x44444444
0x4008	0x33333333
0x4004	0x22222222
0x4000	0x11111111

答：R2＝0x44444444　　R13＝0x4004

177.下面哪条指令不是合法的 ARM 指令 C

A. LDR R1, [R0,#-0x12]

B. LDR R1, [R0,-R2, LSL #2]

C. LDR R1, [R0]!, #0x04

D. LDR R1, [R0, #0x04]!

178.语句 LDR R0, [R1, #4]!的执行结果为()：D

A. R0<-[R1] R1 不变

B. R0<-[R1+4] R1 不变

C. R0<-[R1],R1<-R1+4

D.R0<-[R1+4], R1<-R1+4

179.假设 R0 的内容为 0x4000，寄存器 R1，R2 内容分别为 0x10 和 0x20，存储器内容为空。执行指令 STMIB R0! ,{R1,R2}以后，存储单元 0x4000 及 0x4004 的内容变为：C

A. [0x4000]=0x10, [0x4004]=0x20

B. [0x4000]=0x10, [0x4004]=空

C. [0x4000]=空, [0x4004]=0x10

D．[0x4000]=0x20, [0x4004]=0x10

180.说明 MOV 指令与 LDR 加载指令的区别。

答：MOV：用于寄存器之间的数据传送

LDR：用于存储器单元向寄存器传送数据

181.调用子程序是用 B 还是用 BL 指令？写出一个返回子程序的指令。

答：调用子程序用 BL 指令。

一个返回子程序的指令：MOV PC, LR

182.若加法或减法指令的执行结果需要影响标志位，应该如何编写这样的指令？列举 3 个指令实例。

答：指令中加 S。例如：

ADDS r0,r1,r1

ADCS R1,R2,R3

SUBS r1,R2,R3

183.要实现多寄存器的内容保存及恢复（即寄存器的入栈和出栈），应该使用什么样的汇编指令来实现？给出实例。

答：使用多寄存器寻址 LDM、STM 指令。

例如：

STMFD R13!，{R0-R8}

……

LDMFD R13!，{ R0-R8}

184.BX 指令执行时，状态切换与否是由什么决定的？指出 BX 指令和 BL 指令的不同。

答：BX 指令执行时，状态切换与否是由 BX Rn 指令中 Rn 的内容的最低位决定的，为0表示切换到 ARM 状态，为1表示切换到 Thumb 状态。

BX 指令和 BL 指令的不同在于，

BX 指令：带切换分支转移，4G 范围

BL 指令：保存返回地址的转移，范围有限，小于4G

185. LDR 伪指令与 LDR 加载指令的功能和应用有何区别？举例说明？

答：LDR 伪指令：用于将一个立即数读取到相应的寄存器中，需要用等号（=）来连接地址值。如：LDR　R0，=0xFFF00000

LDR 加载指令：用于从内存中加载数据到寄存器中。如：LDR　R1，[R0]

186.分析下面程序的执行情况

```
MOV　R0，#0
MOV　R1，#10
MOV　R2，#1
L
ADD　R0，R0，R2
ADD　R2，R2，#1
SUBS　R1，R1，#1
BNE　L
```

答：　　完成（1+2+3+4+5+6+7+8+9+10）

187.将 R1 中的 8 位二进制数存储到由 R2+1 指示的单元，并自动更新地址的 ARM 指令是：

A)　STRB　R1,[R2,#1]!

B)　STRH　R1,[R2,#1]!

C)　STREQ　R1,[R2,#1]!

D)　STR　R1,[R2,#1]!

【解析】根据题目意思，本题要用的指令是 STRB，且采用基址加变址寻址方式，同时需要自动更新地址，则指令为 STRB　R1,[R2,#1]!，故选 A。

188.

189.

190.

191.

192.

193.

194.

195.

196.

197.

198.

199.

200.

201.

202.

203.

204.

第 4 章　嵌入式程序设计基础

教材 P84：3，6，8，9，10

205.编写一个完整的程序，程序结构是什么？（教材 P84-3）

206.LDR 指令和 LDR 伪操作有什么不同？（教材 P60-6）

答：LDR 指令是 CPU 指令，完成实质性的数据传送。例如 LDR　R0，[R1, #4]，表示把存储器的字单元 R1+4的内容传送到 R0中。

LDR 伪操作是汇编指令，完成汇编阶段的数据传送，汇编后需要用相应的 CPU 指令替换。例如 LDR　R0，=0X4000，表示把数据4000H 传送到 R0中。

207.编写完整的汇编程序：统计 20 个字数据中所有位中 1 的个数，如果为奇数则在 R0 中存放 1，如果为偶数则在 R0 中存放 0。（教材 P60-8）

208.编写完整的汇编程序：将从存储器地址 SRC 开始的 NUM 个字的数据复制到存储器地址 DST 开始的存储器中。（教材 P60-10）

209.用 ARM 汇编代码实现以下 C 程序段。

```
if ( x - y<3 )
    x=0;
else
    y=0;
```

答：

```
SUB    R0,R0,R1
CMP    R0, #3                ；比较
MOVLT    R0, #0
MOVGE    R1, #0
```

210.阅读下列 C 内嵌 ARM 汇编程序段，并回答问题：

```
#include <stdio.h>
void my_strcpy(char *src,const char *dst)
{
    int ch;
    _asm
    {
        loop
        ldrb ch,[src],#1
        strb ch,[dst],#1
        cmp ch,#0
        bne loop
    }
}
int main(void)
{
    const char *a=”Hello world”;
    char b[20];
    _asm
```

```
    {
        mov r0,a
        mov r1,b
        bl my_strcpy,{r0,r1}
    }
    printf("Original string: %s\n",a);
    printf("Copied string: %s\n",b);
    return 0;
}
```

问题：

（1）指出程序的功能。

（2）写出程序段输出的信息。

答：（1）功能：数据拷贝

（2）输出信息：

Original string: Hello world

Copied string:; Hello world

211.在下面的程序段中，汇编语句至少有 1 处不规范的地方，找出并改正，同时解释标记为①~④的语句。

```
            AREA      INT, CODE, READONLY
            ENTRY
    start   LDR       R1, =SRCSTR
            LDR       R0, =dststr
            BL        strcopy                          ；①

STOP：      B         STOP

strcopy
            .……（此处表示省略的程序段）

            Mov       PC,  LR                          ；②

            AREA      Strings, DATA, READWRITE         ；③
srcstr      DCB       "First string - source",0
dststr      Space     100
            END                                        ；④
```

答：start　LDR　　R1, =SRCSTR　　；此处 start 要顶格书写；SRCSTR 要小写

STOP：　　B　STOP　　；此处 STOP 后应无“：”

Mov　　PC,　LR　　；此处 Mov 应大小写一致

①跳转到子程序 strcopy

②子程序返回

③定义一个数据段 Strings 读写属性

④结束汇编

212.下面的内存表首地址为 0X40003100，请问数据域 a ，Y，String 的地址是多少？

```
MAP     0X40003100
a       field       4
b           field       4
X       field       8
Y       field       8
String  field       16
```

答：0X40003100；0X40003110；0X40003118

213. STM、LDM 指令中用到寄存器{R0-R3,R7,R6,R9},现用一寄存器列表名 pblock 代替用到

的寄存器，给出定义指令。

答：pblock　RLIST　{R0-R3,R7,R6,R9}

214.在 C 语言与汇编程序混合编程中，子程序调用的 ATPCS 规定了哪些基本规则。简要说明寄存器使用规则。

答：基本规则有三个方面内容，分别是寄存器的使用规则及其相应的名字，数据栈的使用规则，参数传递规则。

寄存器的使用规则:

（1）子程序通过寄存器 R0~R3来传递参数。 这时寄存器可以记作： A0~A3 ，被调用的子程序在返回前无需恢复寄存器 R0~R3的内容。

（2）在子程序中，使用 R4~R11来保存局部变量,这时寄存器 R4~R11可以记作：V1~V8。如果在子程序中使用到 V1~V8的某些寄存器，子程序进入时必须保存这些寄存器的值，在返回前必须恢复这些寄存器的值，对于子程序中没有用到的寄存器则不必执行这些操作。在 THUMB 程序中，通常只能使用寄存器 R4~R7来保存局部变量。

（3）寄存器 R12用作子程序间 scratch 寄存器，记作 ip；在子程序的连接代码段中经常会有这种使用规则。

（4）寄存器 R13用作数据栈指针，记做 SP；在子程序中寄存器 R13不能用做其他用途。寄存器 SP 在进入子程序时的值和退出子程序时的值必须相等。

（5）寄存器 R14用作连接寄存器，记作 lr ；它用于保存子程序的返回地址，如果在子程序中保存了返回地址，则 R14可用作其它的用途。

（6）寄存器 R15是程序计数器，记作 PC ；它不能用作其他用途。

（7）ATPCS 中的各寄存器在 ARM 编译器和汇编器中都是预定义的。

215.下面的程序中，语句 IMPORT　strhello 的作用是什么？执行 main 过程后的结果是什么？

C 语言代码文件 str.c：

```
char *strhello="Hello world!\n\0";
```

汇编代码文件 hello.s

```
1        AREA ||.text||, CODE, READONLY
2 main PROC
3        STMFD      sp!,{lr}
4        LDR        r0,=strtemp
```

```
5           LDR     r0,[r0]
6           BL          _printf
7           LDMFD       sp!,{pc}
8 strtemp
9       DCD         strhello
10          ENDP
11          EXPORT      main
12          IMPORT      strhello
13          IMPORT      _main
14          IMPORT      _main
15          IMPORT      _printf
16          IMPORT      ||Lib$$Request$$armlib||, WEAK
 17      END
```

答：汇编语言访问 C 变量；打印输出字符串“Hello world!”。

216.有 200 个有符号字节数据，存储在以 0x40003200 为起始地址的内存单元中,试编写汇编程序，找出其中的最大值、最小值以及存储地址,分别存储到内存单元 0x40003101（最大值）、0x40003002（最小值）、0x40003010(最大值地址)、0x40003014(最小值地址)中。

答：

(1)、程序框架

(2)、找出最大、最小值

(3)、存入指定地址单元

217.编写汇编程序将地址单元 0x40003000～0x40003188 的数据复制到存储单元 0x40003200～0x40003388 中。要求程序中必须使用 LDRB、STRB、LDR、STR、LDM、STM 指令。

答：(1)、程序框架

(2)、计算出使用 LDM、STM 传送次数

计算出使用 LDR、STR 传送次数

计算出使用 LDRB、STRB 传送次数

(3) 完成传送功能

218.编写一段 ARM 汇编程序：循环累加队列 rjarray 中的所有元素，直到碰上零值元素，结

果放在 r4 中。程序框架如下，补充代码完成上述功能。

```
AREA  total，CODE，READONLY
ENTRY
Start    MOV   r4，#0
ADR    r0，rjarray
；在此补充代码
```

答：

```
Loop   LDR    r1，[r0]，#4
ADD   r4，r4，r1
CMP   r1，#0
BNE   loop
stop          B         stop
rjarray
DCD   0x11
DCD   0x22
……
DCD   0x0
END
```

219.编写子程序实现将存储器中起始地址 S1 开始的 6 个字数据移动到 S2 处（要求使用 LDM 和 STM 语句）。

答：

```
STMFD SP!,{R0-R6,LR}
LDR R6，=S1;
LDMIA R6!，{R0-R5}
LDR R6,  =S2
STMIA R6!,{R0-R5}
LDMFD SP!,{R0-R6，PC}
```

220.用汇编语言实现以下 C 语言语句。

```
For(i=limit;i>=1;i--) {fact=fact*i}
```

答：

```
Fact        MUL R0,R1,R0
            SUBS R1,R1,#1
```

```
        BNE fact
```

221. DCB 用于分配一块字节单元并用伪指令中指定的表达式进行初始化。（　　）√

222.用 ARM 汇编语言编写完整的子程序，该程序从 NN 地址处连续读取 20 个字符，并将这 20 个字符复制到目的地址标号 MM 处。

答：

```
AREA    ||.text||，CODE，READONLY
NN      DCB     “12345678901234567890”
MM      DCB     “12345678901234567890”
MY _SUB
        STMFD   R13!，{R0-R3}
        LDR     R0，=NN
        LDR     R1，=MM
        MOV     R2，#20
LOOP
        LDRB    R3，[R0]，#1
        STRB    R3，[R1]，#1
        SUBS    R2，R2，#1
        BNE     LOOP
        LDMFD   R13!，{R0-R3}
        MOV     PC，LR
        END
```

223.C 语言程序可以嵌套加入汇编程序模块。　　（√）

224.有 20 个有符号的字数据，依次存放在内存 BUFF 开始字单元中。试用 ARM 汇编语言编写完整的程序（包括代码段、数据段），从中找出最大值、最小值，并分别放入内存字单元 MAX、MIN 中。

（1）画出程序设计流程图。

（2）编写完整源程序。

答：流程图（略）

```
        AREA Search，CODE，READONLY          ；代码段
        CODE32
NUM     EQU     20                  ；定义数据个数
        ENTRY
```

```
start
        LDR    R3，=BUFF        ；设置初始地址
        LDR    R4，=NUM         ；取数据个数
        LDR    R0，[R3]         ；R0 存放最大数
        LDR    R1，[R3]         ；R1 存放最小数
loop
        LDR    R2，[R3]，#4     ；取比较数据
        CMP    R2，R0
        MOVGT     R0，R2            ；若取出的数大于 R0 中数据，则更新 R0
        CMP    R2，R1
        CMPLTR1，R2             ；若取出的数小于 R1 中数据，则更新 R1
        SUBS   R4，#0x01        ；计数减 1
        BNE    loop             ；计数未完，继续
        LDR    R3，=MAX
        STR         R0，[R3]
        LDR    R3，=MIN
        STR         R1，[R3]
stop
        MOV    R0，#0x18        ；返回系统
        LDR    R1，0x20026
        SWI         0x123456

        AREA DefineData，DATA，READWRITE          ；数据段
BUFF     DCD      23,54,34,64,35,34,……,98,0F5,39      ；定义 20 个有符号字数据
MAX      DCD      0          ；最大值单元
MINDCD      0          ；最小值单元
        END
```

225.ADR 和 ADRL 伪指令都是将基于 PC 的地址值或基于寄存器的地址值读取到寄存器中，二者的区别是什么？

答：ADR　小范围地址读取

ADRL　中等范围地址读取，类似于 ADR，　但比 ADR 读取更大范围的地址。

226.用 ARM 汇编语言编写完整的程序，实现 1+2+3+…+N。

答：；功能：计算1+2+...+N 的值

；说明：N≥0，当 N=0时结果为0；当 N=1时结果为1。

```
N               EQU          100                ；定义 N 的值为 100
                AREA   Example5,CODE,READONLY    ；声明代码段 Example5
                ENTRY                ；标识程序入口
                CODE32                    ；声明 32 位 ARM 指令
ARM_CODE     LDR          SP,=0x40003F00；设置堆栈指针
```

```
        ADR     R0,THUMB_CODE+1
        BX      R0              ; 跳转并切换处理器状态
        LTORG                   ; 声明文字池
        CODE16                  ; 声明 16 位 Thumb 指令
THUMB_CODE LDR      R0,=N       ; 设置子程序 SUM_N 的入口参数
        BL      SUM_N       ; 调用子程序 SUM_N
        B       THUMB_CODE
; 名称：SUM_N
; 功能：计算 1+2+...+N 的值
; 入口参数：R0      N 的值
; 出口参数：R0      运算结果
; 占用资源：R0
; 说明：当 N=0 时结果为 1；当 N=1 时结果为 1。若运算溢出，结果为 0。
SUM_N
        PUSH    {R1-R7,LR}; 寄存器入栈保护
        MOVS    R2,R0       ; 将 N 的值复制到 R2，并影响条件码标志
        BEQ     SUM_END     ; 若 N 的值为 0，则返回。(此时 R0 没有被更改)
        CMP     R2,#1
        BEQ     SUM_END     ; 若 N 的值为 1，则返回。(此时 R0 没有被更改)
        MOV     R1,#1       ; 初始化计数器 R1=1
        MOV     R0,#0       ; 初始化结果寄存器 R0=0
SUM_L1      ADD     R0,R0,R1        ; R0 = R0 + R1
        BCS     SUM_ERR     ; 结果溢出，跳转到 SUM_ERR
        CMP     R1,R2       ; 将计数器的值与 N 比较
        BHS     SUM_END     ; 若计数器的值≥N，则运算结束
        ADD     R1,R1,#1
        B       SUM_L1
SUM_ERR     MOV     R0,#0
SUM_END     POP     {R1-R7,PC}; 寄存器出栈，返回
        END
```

227.汇编语言编写的函数 strcopy 用于实现将字符串 S 拷贝到字符串 d，下边用法正确的是：

C

A．C 语言直接调用函数 strcopy (d,s)即可实现将字符串 s 拷贝到字符串 d;

B．C 语言首先声明 void strcopy(char *dnstr,const char *snstr);然后调用函数 strcopy (d,s)即可实现将字符串 s 拷贝到字符串 d;

C．C 语言首先声明 extern　void strcopy(char *dnstr,const char *snstr);然后调用函数 strcopy (d,s)即可实现将字符串 s 拷贝到字符串 d;

D．C 语言首先声明 void extern strcopy(char *dnstr,const char *snstr);然后调用函数 strcopy (d,s)即可实现将字符串 s 拷贝到字符串 d;

228.已知 R0=a, R1=b，用汇编语言实现 if ((a!=0x10)&&(b!=0x30)) a=a+b

答：

```
    AREA      Exp, CODE, READONLY
a EQU   0x03
b EQU   0x04
c EQU   0x10
d EQU   0x30
     ENTRY
    CODE32
start
    LDR       r0, =a
    LDR       r1, =b
    LDR       r2, =c
    LDR       r3, =d
    CMP       r0,r2     ;a!=0x10
    BEQ       stop
    CMP       r1,r3     ;b!=0x30
    BEQ       stop
    ADD       r0,r0,r1 ;a=a+b
stop
    MOV       r0, #0x18
    LDR       r1, =0x20026
```

```
        SWI     0x123456
        END
```

红色部分　程序框架

蓝色　　寄存器赋值

黑色　　比较部分

a=a+b　运算

229.编写汇编程序计算内存 0x40003000 开始的 20 个字节单元数据之和，如果和小于 100 则将这 20 个单元复制到内存 0x40003020 开始的地址处，否则将这 20 个单元清零。

答：

```
        AREA    Exp, CODE, READONLY
ADDR1   EQU 0x40003000
ADDR2   EQU 0x40003200
CNT     EQU 20
VALUE   EQU 100
        ENTRY
        CODE32
start
        LDR     r0, =ADDR1
        LDR     r2, =CNT
        LDR     r3,=VALUE
        MOV     R4,#0
l0      LDRB    R5,[r0],#1
        ADD     r4,r4,r5
        SUBS    r2,r2,#0x01
        BNE     l0
l1      CMP     r4,r3
        BCC     l3
        LDR     r0, =ADDR1
        LDR     r2, =CNT
        MOV     R4,#0
l2      strb    r4,[r0],#1
```

```
        subs    r2,r2,#1
        bne     l2
        b stop
l3      LDR     r0, =ADDR1
        LDR     r1, =ADDR2
        LDR     r2, =CNT
l4      LDRB    r4,[r0],#1
        STRB    r4,[r1],#1
        subs    r2,r2,#1
        bne     l4
stop
        MOV     r0, #0x18
        LDR     r1, =0x20026
        SWI     0x123456
        END
```

(1)红色，程序框架

(2)绿色，累计和，

(3)粉色，20 地址单元清 0

(4)黑色　拷贝

230.说出下面函数的主要功能。

```
void    my_strcopy(const char *src,char *dst)
{ int ch
  __asm
    { loop:
      LDRB ch,[src],#1
      STRB    ch,[dst],#1
      CMP ch,#0
      BNE    loop
    }
}
```

答：字符串复制

231.已知 R0=a, R1=b，用汇编语言实现 if (a>b)a++ else b++

232.数组 a、b 分别存放在 0x4000 与 0x5000 为起始地址的区域内，将以下 C 语言程序转为完整的汇编程序：

```
for (i=1;i<8;i++)
 {
    b[i]=a[i];
    if(b[i]=0)  b[i]=-1;
}
```

答：

```
        AREA   COPY1,CODE,READONLY ; 声明代码段 COPY1
SS1     EQU    0x4000                ; 源地址
DD1     EQU    0x5000                ; 目的地址
NUM     EQU    8                     ; 字计数
        ENTRY                        ; 标识程序入口
        CODE32                       ; 声明 32 位 ARM 指令
START
LDR     R0, =SS1                ; 设置源地址
LDR     R1, =DD1                ; 设置目标地址
MOV     R2, #0                  ; 设置字计数初始值为 0
LOOP
LDR     R3, [R0], #4            ; 取一个字源数据
CMP     R3, #0                  ; 字源数据与 0 比较
MVNEQ   R3, #0                  ; 为 0，送-1
STR     R3, [R1], #4            ; 送目标字单元
ADD     R2, R2, #1              ; 字计数加 1
CMP     R2, #NUM                ; 达到预定的 NUM 吗？
BCC     LOOP                    ; 未达到，继续
HALT
B       HALT                    ; 达到，暂停
END
```

233.修改程序错误

```
AREA addreg,CODE,READONLY
ENTRY
Main        ADR r0,ThumbProg
                    BX      r0
CODE16
ThumbProg                   ADR r0,ARMProg
                                    BX      r0
CODE32
ARMProg         MOV R4,#4
                            MOV R5,#5
                            SUB    R4,R4,R5
STOP            MOV R0,#0X18
                            MOV R0,#20026
                            SWI     #0xAB
END
```

答：

```
AREA addreg,CODE,READONLY
ENTRY
Main        ADR r0,ThumbProg+1
                BX      r0
CODE16
ThumbProg                   ADR r0,ARMProg
                             BX      r0
CODE32
ARMProg         MOV R4,#4
                             MOV R5,#5
                             SUB    R4,R4,R5
STOP                MOV R0,#0X18
                            MOV R0,#20026
                            SWI     #0x123456
END
```

234.编写程序将 R1 的高 8 位数据转移到 R0 的低 8 位中，保存到地址 0x40003000 单元内

答：

```
AREA ByteCopY, CODE, READONLY
COUNT        EQU   0x44332211
A1           EQU   0x40003000
             ENTRY
start
             LDR R1,=COUNT
             MOV R0,R1,LSR#24
             LDR R1,=A1
             STR R0,[R1]
             END
```

235.编写汇编程序计算 X！的值。

答：

```
X       EQU     9           ; 定义 X 的值为 9
n       EQU     8           ; 定义 n 的值为 8
            AREA  Example4,CODE,READONLY   ; 声明代码段 Example4
            ENTRY           ; 标识程序入口
            CODE32          ; 声明 32 位 ARM 指令
START   LDR     SP,=0x40003F00    ; 设置堆栈(满递减堆栈，使用 STMFD/LMDFD 指令)
            LDR     R0,=X
            LDR     R1,=n
            BL      POW             ; 调用子程序 POW，返回值为 R0
HALT    B       HALT
; 名称：POW
; 功能：整数乘方运算。
; 入口参数：R0      底数
;           R1      指数
; 出口参数：R0      运算结果
; 占用资源：R0、R1
```

```
; 说明：本子程序不考虑溢出问题
POW
        STMFD SP!,{R1-R12,LR}       ; 寄存器入栈保护
        MOVS  R2,R1                 ; 将指数值复制到 R2，并影响条件码标志
        MOVEQ   R0,#1               ; 若指数为 0，则设置 R0=1
        BEQ     POW_END             ; 若指数为 0，则返回
        CMP     R2,#1
        BEQ     POW_END             ; 若指数为 1，则返回。(此时 R0 没有被更改)
        MOV     R1,R0               ; 设置 DO_MUL 子程序的入口参数 R0 和 R1
        SUB     R2,R2,#1            ; 计数器 R2 = 指数值减 1
POW_L1    BL    DO_MUL              ; 调用 DO_MUL 子程序，R0 = R1 * R0
        SUBS  R2,R2,#1              ; 每循环一次，计数器 R2 减 1
        BNE     POW_L1              ; 若计数器 R2 不为 0，跳转到 POW_L1
POW_END   LDMFD   SP!,{R1-R12,PC}   ; 寄存器出栈，返回
; 名称：DO_MUL
; 功能：32 位乘法运算。
; 入口参数：R0    乘数
;           R1    被乘数
; 出口参数：R0    计算结果
; 占用资源：R0、R1
; 说明：本子程序不会破坏 R1
DO_MUL    MUL     R0,R1,R0          ; R0 = R1 * R0
        MOV     PC,LR               ; 返回
                END
```

236.修改下面嵌入式汇编的错误

```
__asm
{
MOV   R0,x
ADD   y,R0,x/y
}
```

答：

```
__asm
{
mov      var,x
add y,var,x/y
}
```

237.说出下面函数的主要功能

```
void    my_strcopy(const char *src,char *dst)
{ int ch
   __asm
      { loop:
          LDRB ch,[src],#1
         STRB    ch,[dst],#1
         CMP ch,#0
         BNE    loop
      }
}
```

答：字符复制，结束条件 0

238.分析程序实现什么功能

```
num         EQU         2
MOV         R0, #0
myfunc                                  ;
            CMP         R0, #num
            BCS      DoError            ;
            ADR         R3, JumpTable      ;
            LDR         PC, [R3,R0,LSL#2]
JumpTable
            DCD         DoAdd
            DCD         DoSub
DoAdd
           ……
DoSub
```

……..

DoError

……..

答：查表或散转

239.用汇编程序实现如下功能：将 R1 的高 16 位数据与低 16 位交换，保存到地址 0x40003000 内。

答：程序结构书写规范,高低位交换,保存到 0x40003000 中

参考程序：

```
A1              EQU   0x40003000
start
        LDR R3,=A1
        MOV R0,R1,LSR#16
        AND R2,R1,#0XFF
        ORR R0 R0 R2
        STR R0,[R3]
        END
```

240.用汇编语言实现 1+2+3+…+N。

答：程序结构书写规范,循环体,求和

参考程序（核心部分程序，是全部）

```
CMP     R2,#1
        BEQ     SUM_END     ; 若 N 的值为 1，则返回。
        MOV     R1,#1       ; 初始化计数器 R1=1
        MOV     R0,#0       ; 初始化结果寄存器 R0=0
SUM_L1      ADD     R0,R0,R1        ; R0 = R0 + R1
        BCS         SUM_ERR     ; 结果溢出，跳转到 SUM_ERR
        CMP     R1,R2       ; 将计数器的值与 N 比较
        BHS     SUM_END     ; 若计数器的值≥N，则运算结束
        ADD     R1,R1,#1
        B       SUM_L1
 SUM_END
```

241.简要说明 ATPCS 关于子程序参数传递是如何规定的。

答：R0－R3依次表示4个参事，超过4个使用堆栈传递32位返回值使用 R0，64位返回值使用 R0、R1，依此类推

242. C 语言调用汇编程序时，使用哪些寄存器用来传递参数？

答：如果函数有4个参数，则将分别用 r0 、r1、r2和 r3来传递，如果参数多于4个，则多余的参数将被压入堆栈。

243.以下程序片段的主要功能是什么，程序中有何错误，如果有请改正。

```
        CODE32
A1   ADR R0,T1
        BX R0
        CODE16
T1   MOV R0,#10
        ……
END
```

答：

ARM 状态到 THRUM 状态切换

有错误，A1　　ADR R0,T1 应该改为 A1　　ADR R0,T1+1

244.编写程序将地址 1000H～1030H 数据全部搬迁到 2000H～2030H 中，并将源数据区清零。

245.阅读下列 C 内嵌 ARM 汇编程序段，并回答问题：

```
#include <stdio.h>
void my_strcpy(char *src,const char *dst)
{
int ch;
_asm
{
loop
ldrb ch,[src],#1
strb ch,[dst],#1
cmp ch,#0
bne loop
}
```

```
}
int main(void)
{
const char *a=”Hello world”;
char b[20];
_asm
{
mov r0,a
mov r1,b
bl my_strcpy,{r0,r1}
}
printf(“Original string: %s\n”,a);
printf(“Copied string: %s\n”,b);
return 0;
}
```

问题：

（1）指出程序的功能。

（2）写出程序段输出的信息。

246.已知 32 位变量 X、Y 存放在存储器的地址 0x90010、0x90014 中，要求实现 Z=X+Y，其中 Z 的值存放在 0x90018 中。

答：

```
AREA EX4_41,CODE,READONLY
        ENTRY
        CODE32
START   LDR R0,=0x90010          ;变量 X 的地址送入 R0
        LDR R1,[R0],#4           ;变量 X 的值读入 R1
        LDR R2,[R0],#4           ;变量 Y 的值读入 R2
        ADD R1,R1,R2             ;X+Y 结果存入 R1
        STR R1,[R0]              ;结果存入 z 中
```

```
        B START
        END
```

247.已知 32 位有符号数 X 存放在存储器的地址 0x90010 中，要求实现：

$$Y=\begin{cases} X & (X\geqq 0) \\ -X & (X<0) \end{cases}$$

其中 Y 的值存放在 0x90010 中。

答：

```
AREA EX4_42,CODE,READONLY
        ENTRY
        CODE32
START   LDR R1,=0x90010          ;加载变量 X 的地址->R1
        MOV R0,#0                ;0->R0
        LDR R2,[R1]              ;将 X 的值加载到 R2
        CMP R2,#0                ;X 与 0 比较,影响标志位
        SUBLT R2,R0,R2           ;X<0 执行该语句,提到-X
        STR R2,[R1]              ;保存结果
        B START
        END
```

248.已知 32 位有符号数 X 存放在存储器的地址 0x90010 中，要求实现：

$$Y=\begin{cases} 1 & (X>0) \\ 0 & (X=0) \\ -1 & (X<0) \end{cases}$$

其中 Y 的值存放在 0x90010 中。

答：

```
AREA EX4_43,CODE,READONLY
        ENTRY
        CODE32
START   LDR R1,=0x90010          ;加载变量 X 的地址->R1
```

```
        LDR R2,[R1]             ;加载 X 的值->R2
        CMP R2,#0               ;与 0 比较
        BEQ ZERO                ;为 0 则跳转到 ZERO 处理
        BGT PLUS                ;大于 0 则跳转到 PLUS 处理
        MOV R0,#-1              ;否则小于 0,将 R0 设置为-1
        B FINISH                ;跳转到结束
PLUS    MOV R0,#1               ;大于 0,将 R0 设置为 0
        B FINISH                ;跳转到结束
ZERO    MOV R0,#0               ;等于 0,将 R0 设置为 0
FINISH STR R0,[R1]              ;将结果 R0 保存
        B START
        END
```

249.多分支 ARM 汇编的程序

答：

```
  AREA EX4_44,CODE,READONLY
        ENTRY
        CODE32
START   CMP R0,#8               ;与 8 比较
        ADDLT PC,PC,R0,LSL#2    ;小于 8 则根据 R0 计算跳转地址,
                                ;并用该地址修改 PC
        B method_d              ;大于 8 程序跳转到默认分支段执行
        B method_0              ;分支表结构,其偏移量由 R0 决定
        B method_1
        B method_2
        B method_3
        B method_4
        B method_5
        B method_6
        B method_7
method_0                        ;method_0 的入口
        MOV R0,#1               ;method_0 的功能
```

```
        B end0
method_1
        MOV R0,#2
        B end0
method_2
        MOV R0,#3
        B end0
method_3
        MOV R0,#4
        B end0
method_4
        MOV R0,#5
        B end0
method_5
        MOV R0,#6
        B end0
method_6
        MOV R0,#7
        B end0
method_7
        MOV R0,#8
        B end0
method_d
        MOV R0,#0
end0    B START
        END
```

250.编制程序使 S=1+2×3+3×4+4×5+……+N(N+1)，直到 N 等于 10 为止。

答：

```
AREA EX4_45,CODE,READONLY
        ENTRY
        CODE32
```

```
START   MOV R0,#1                ;R0 用作累加,置初值 1,S
        MOV R1,#2                ;R1 用作第一个乘数,初值为 2,N
REPEAT ADD R2,R1,#1              ;R2 用作第二个乘数,N+1
        MUL R3,R2,R1             ;实现 N*(N+1)部分积存于 R3
        ADD R0,R0,R3             ;将部分积累加至 R0
        ADD R1,R1,#1             ;修改 N 的值得到下一轮乘数
        CMP R1,#10               ;循环次数比较
        BLE REPEAT               ;未完则重复
        B START
        END
```

251.编制程序，求两个数组 DATA1 和 DATA2 对应的数据之和，并把和数存入新数组 SUM 中，计算一直进行到两数之和为零时结束，并把新数组的长度存于 R0 中。

答：

```
AREA BlockData,DATA,READWRITE      ;定义数据段
DATA1   DCD 2,5,0,3,-4,5,0,10,9          ;数组 DATA1
DATA2   DCD 3,5,4,-2,0,8,3,-10,5         ;数组 DATA2
SUM     DCD 0,0,0,0,0,0,0,0,0            ;数组 SUM

        AREA Ex4_46,CODE,READONLY        ;定义代码段
        ENTRY
        CODE32
START   LDR R1,=DATA1            ;数组 DATA1 的首地址存入到 R1
        LDR R2,=DATA2            ;数组 DATA2 的首地址存入到 R2
        LDR R3,=SUM              ;数组 SUM 的首地址存入到 R3
        MOV R0,#0                ;计数器 R0 的初始值置 0
LOOP    LDR R4,[R1],#04          ;取 DATA1 数组的一个数,同时修改地址指针
        LDR R5,[R2],#04          ;取 DATA1 数组的一个数,同时修改地址指针
        ADDS R4,R4,R5            ;相加并影响标志位
        ADD R0,R0,#1             ;计数器加 1
        STR R4,[R3],#04          ;保存结果到 SUM 中,同时修改地址指针
```

```
        BNE LOOP                    ;若相加的结果不为 0 则循环
        END
```

252.在以 BUF 为首地址的字存储区中存放有 10 个无符号数 0x0FF, 0x00, 0x40, 0x10, 0x90, 0x20, 0x80, 0x30, 0x50, 0x70, 请将它们按从小到大的顺序排列在 BUF 存储区中，编写程序。

答：

```
N EQU 10
        AREA EX4_47,CODE,READONLY
        ENTRY
        CODE32
START   LDR R0,=BUF                 ;指向数组的首地址
        MOV R1,#0                   ;外循环计数器
        MOV R2,#0                   ;内循环计数器
LOOPI   ADD R3,R0,R1,LSL #2         ;外循环首地址放入 R3
        MOV R4,R3                   ;外循环首地址放入 R4
        ADD R2,R1,#1                ;内循环计数器初值
        MOV R5,R4                   ;内循环下一地址初值
        LDR R6,[R4]                 ;取内循环第一个值 R4
LOOPJ   ADD R5,R5,#4                ;内循环下一地址值
        LDR R7,[R5]                 ;取出下一地址值 R7
        CMP R6,R7                   ;比较
        BLT NEXT                    ;小则取下一个
        SWP R7,R6,[R5]              ;大则交换,最小值 R6
        MOV R6,R7
NEXT    ADD R2,R2,#1                ;内循环计数
        CMP R2,#N                   ;循环中止条件
        BLT LOOPJ                   ;小于 N 则继续内循环,实现比较一轮
        SWP R7,R6,[R3]              ;否则内循环一轮结束
        ADD R1,R1,#1                ;外循环计数
        CMP R1,#N-1                 ;处循环中止条件
```

```
        BLT LOOPI                       ;小于 N-1 继续执行外循环
        B START

        AREA BlockData,DATA,READWRITE
BUF     DCD 0x0ff,0x00,0x40,0x10,0x90,0x20,0x80,0x30,0x50,0x70
        END
```

253.编写一程序，查找存储器从 0x400000 开始的 100 个字中为 0 的数目，将其结果存到 0x400190 中。

答：MOV R0，#0x400000

```
    MOV R1，#0
    MOV R7，#100
    LP
    LDR R2，[R0]，#4
    CMP R2，#0
    BNE NEXT
    ADD R1，R1，#1
NEXT
    SUBS R7，R7，#1
    BNE LP
    STR  R1，[R0]
    B  $
```

254.改错并指出程序功能

```
    AREA   JUMP,CODE,READONLY
    NUM    EQU   2
    ENTRY
START
    MOV  R0,0（#0）
    MOV  R1,#3
    MOV  R2,#2
    BL    FUNC
```

```
FUNC
    CMP   R0,#NUM
    MOVCS PC,LR
    ADR  R3,JTABLE
    LDR  PC,[R3,R0,LSR #2]
JTABLE DCD  DOADD
    DCD  DOSUB
DOADD
    ADD R0,R1,R2
    MOV PC,LR
DOSUB
    SUB R0,R1,R2
    MOV PC,LR
    END
```

答：通过跳转表指令实现两个数 2+3 的求和。

255.用完整 ARM 汇编语言编写程序：使用 LDR 指令读取 0x40001000 上的数据，将数据加 3，若结果小于 100 则使用 STR 指令把结果写回原地址，若结果大于等于 100，则把 0 写回原地址。然后再次读取 0x40001000 上的数据，将数据加 1，判断结果是否小于 100 周而复始循环。

答：

```
COUNT EQU       0x40001000 ; 定义一个变量，地址为 0x40001000
        AREA   Example2,CODE,READONLY; 声明代码段 Example2
        ENTRY              ; 标识程序入口
        CODE32             ; 声明 32 位 ARM 指令
START  LDR      R1，=COUNT  ; R1 <= COUNT
        MOV     R0,#0      ; R0 <= 0
        STR     R0,[R1]    ; [R1] <= R0，即设置[COUNT]为 0

LOOP    LDR     R1,=COUNT
        LDR     R0,[R1]    ; R0 <= [R1]
        ADD     R0,R0,#3   ; R0 <= R0 + 1
        CMP     R0,#100    ; R0 与 100 比较，影响条件码标志
        MOVHS   R0,#0      ; 若 R0 大于等于 100，则此指令执行，R0 <= 0
        STR     R0,[R1]    ; [R1] <= R0，即保存 COUNT
        B       LOOP
        END
```

256.用 ARM 汇编语言编写完整的子程序，该程序从 0x40001000 地址处连续读取 100 个字符，并将这 100 个字符复制到目的地址标号 DIST 处。

答：参考程序：

```
AREA ||.text||,CODE,READONLY
DIST    DCB “123456789……01234567890”；100 个字符
CODE 32
ENTRY
MY _SUB
        STMFD R13!,{R0-R3}
        MOV R0,=# 0x40001000
        LDR R1,= DIST
        MOV R2,#100
LOOP
        SUBS R2,R2,#1
        LDRB R3,[R0],#1
        STRB R3,[R1],#1
        BNE   LOOP
        LDMFD R13!,{R0-R3}
        MOV PC,LR
        END
```

257.存储器从 0x400000 开始的 100 个单元中存放着 ASCII 码，编写程序，将其所有的小写字母转换成大写字母，对其它的 ASCII 码不做变换。

答：

```
    MOV    R0，#0x400000
    MOV    R1，#0
LP
    LDRB   R2，[R0,R1]
    CMP R2，#0x61
    BLO NEXT
    CMP R2，#0x7B    ;0x7A 为 z
    SUBLO  R2， R2，#0x20
    STRBLO  R2，[R0,R1]
NEXT
    ADD R1， R1，#1
    CMP R1，#100
    BNE LP
```

258.以下对伪指令的解释错误的是：

A) DCW 0x12；在内存区域分配半字的内存空间并初始化为 0x0012

B) CODE32；伪指令通知汇编器，其后的指令序列为 32 位的 ARM 指令

C) Baud EQU 2400；为定义一个 16 位常量 Baud 值为 2400

D) EXTERN SUB1;当前文件引用外部标号 SUB1

【解析】伪指令 DCW 用于分配一片连续的半字存储单元并用指定的数据初始化；CODE32 通知编译器，其后的指令序列为 32 位的 ARM 指令；EQU 是等于伪指令，用于为程序中的常量、标号等定义一个等效的字符名称；EXTERN 是外部标号引用声明伪指令，用于通知编译器要使用的标号在其他的源文件中定义，但要在当前文件中引用。故 C 项错误。

259.

260.

261.

262.

263.

264.

265.

266.

267.

268.

269.

270.

271.

272.

273.

第 5 章 嵌入式内部可编程模块

教材 P184：1-11

274.S3C2440 芯片外部可寻址的存储空间是多少？（教材 P184-1）

答：外部存储空间：1GB。存储块（Bank） ：8个

275.S3C2440 芯片有几个 GPIO 端口？分别是什么？（教材 P184-2）

答：S3C2440有130个 GPIO 引脚，分布在9个 GPIO 端口：

（1）端口 A：GPA，23个引脚，无内部上拉电阻，主要用于输出、地址总线、Bank 选择、 Nand flash 控制等。

（2）端口 B：GPB，11个引脚，主要用于输入输出、DMA 请求、定时器输出等。

（3）端口 C：GPC，16个引脚，主要用于输入输出、LCD 接口。

（4）端口 D：GPD，16个引脚，主要用于输入输出、 LCD 接口、SPI 接口。

（5）端口 E：GPE，16个引脚， 主要用于输入输出、 SPI 接口、SD 卡接口、IIC 接口、IIS 接口、AC97控制器接口。

（6）端口 F：GPF，8个引脚，主要用于输入输出、中断请求输入口。

（7）端口 G：GPG，16个引脚，主要用于输入输出、中断请求输入口、UART 接口、SPI 接口。

（8）端口 H：GPH，11个引脚，主要用于输入输出、时钟输出、UART 接口。

（9）端口 J：GPJ，13个引脚，主要用于输入输出、摄像头接口。

每个 GPIO 端口均是多功能的。

276.看门狗定时器原理是什么？（教材 P184-5）

277.触摸屏接口模式分哪几类？（教材 P184-8）

278.编程序实现流水灯。（教材 P184-9）

279.编程序实现三角波、方波（教材 P184-11）

280.FIQ、IRQ 有什么不同？向量 IRQ 和非向量 IRQ 有什么不同？

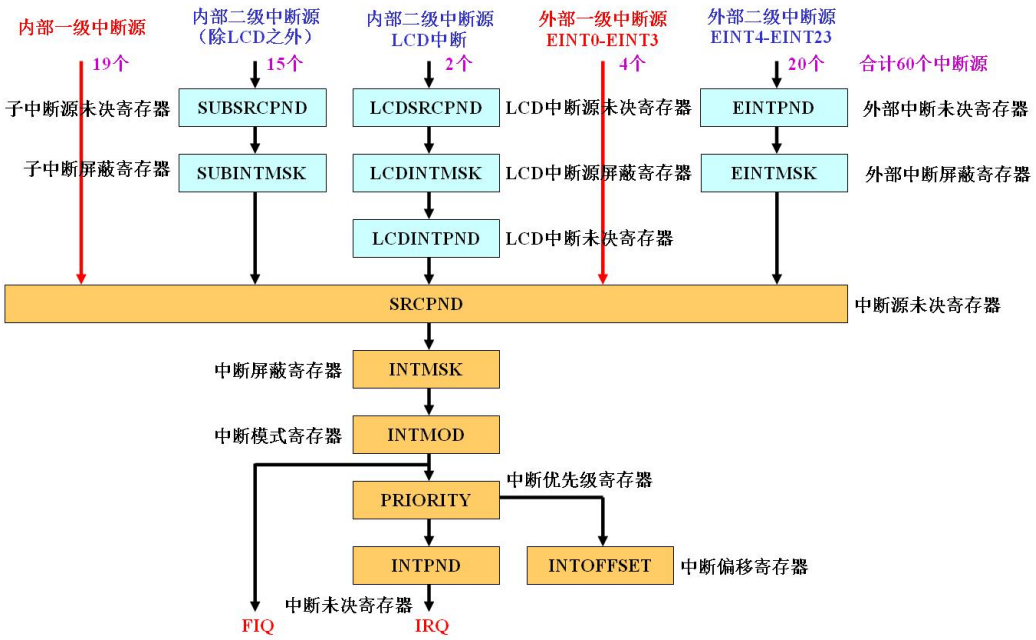
答：

FIQ 是快速中断，具有最高优先级，中断处理转入 FIQ 模式；IRQ 是普通中断，优先级低于 FIQ，中断处理转入 IRQ 模式。

向量 IRQ 支持16个向量 IRQ 中断，16个优先级，能为每个中断源设置服务程序地址；非向量 IRQ 支持一个非向量 IRQ 中断，所有中断都共用一个相同的服务程序入口地址。

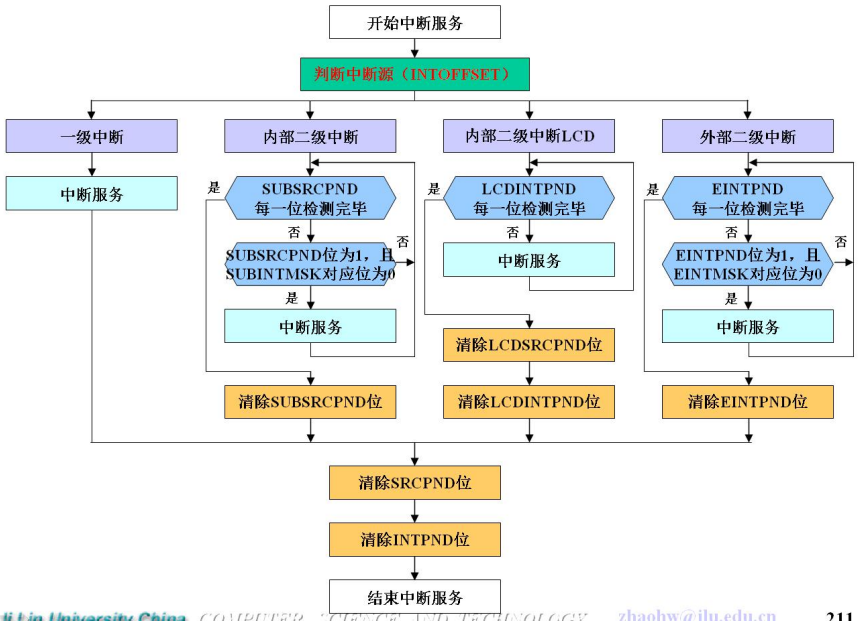
281.依据内部、外部、一级、二级中断源，描述 S3C2440 的中断机制。

答：



282.画图说明 S3C2440 中断处理流程。

答：



283.说明 S3C2440 优先级仲裁功能，并描述优先级仲裁过程。

答：SRCPND（中断源未决寄存器）中的32个中断请求通过7个仲裁器（优先级判别逻辑）的选择，最终生成一个优先级最高的中断源，设置在 INTPND（中断未决寄存器）中。

仲裁器功能：确定中断源的优先级，6个一级仲裁器、1个二级仲裁器。

优先级仲裁过程：

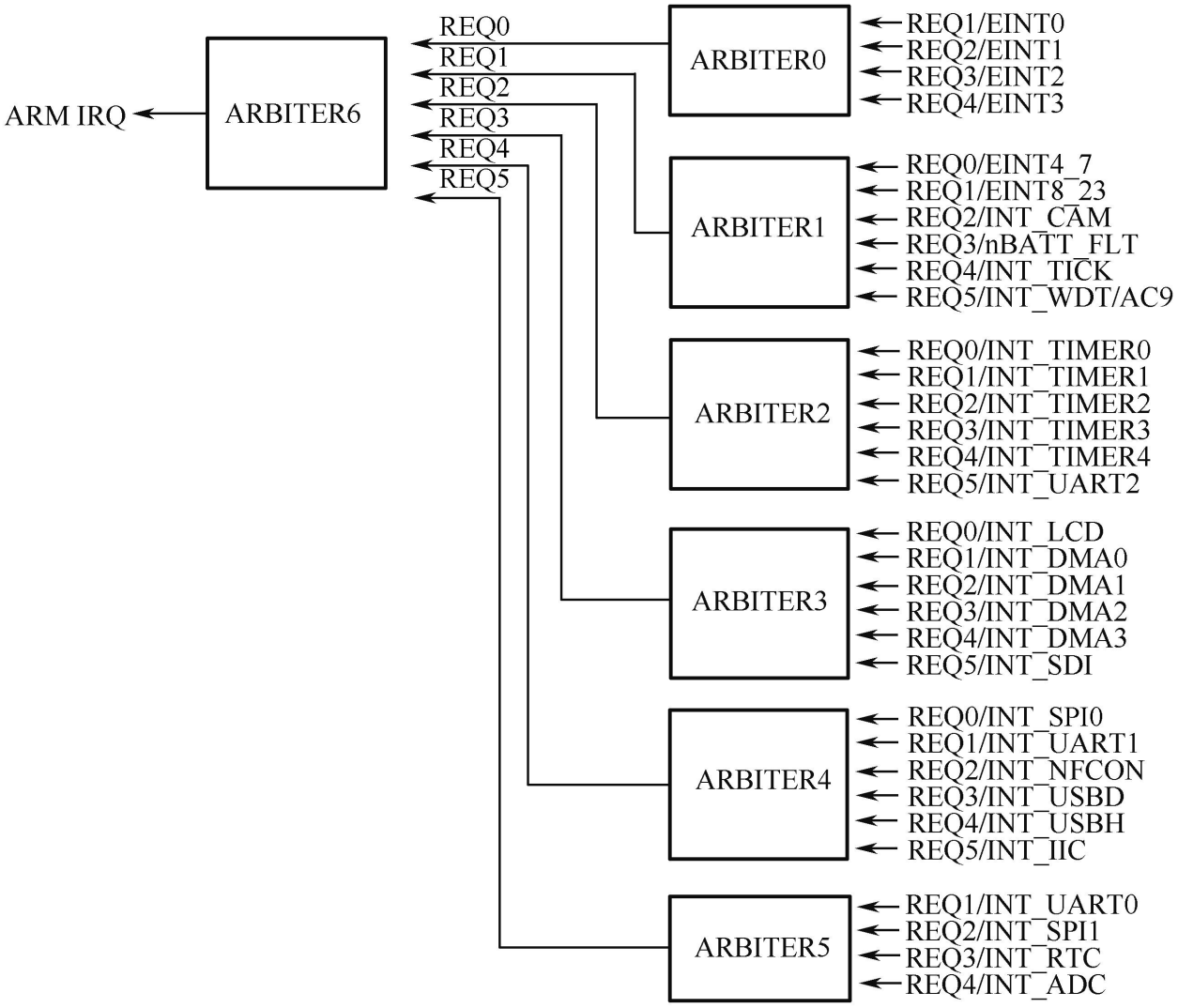
（1）先一级：每个一级仲裁器对自己的分组进行优先级判别，选择一个优先级最高的中断，送给二级仲裁器。 6个一级仲裁器形成6个中断源。

（2）后二级：二级仲裁器在一级仲裁器输出的6个中断源中选择一个优先级最高的中断源进行触发。

仲裁器控制：每个仲裁器有2个控制信号，ARB_MODE、ARB_SEL。

ARB_MODE：仲裁模式，1位。0=固定优先级，1=循环优先级。

优先级总规则（无论是固定优先级，还是循环优先级）：对于每个仲裁模块， REQ0、REQ5的优先级不变，并且 REQ0最高、REQ5最低。



284.简要说明 ARM 的异常的响应和返回的过程。

答：异常的进入：

（1）将下一条指令的地址存入相应连接寄存器 LR，以便程序在处理异常返回时能从正确的位置重新开始执行。

（2）将 CPSR 复制到相应的 SPSR 中。

（3）根据异常类型，强制设置 CPSR 的运行模式位。

（4）强制 PC 从相关的异常向量地址取下一条指令执行，从而跳转到相应的异常处理程序。也可以设置中断禁止位来阻止其他无法处理的异常嵌套。

异常的返回：

（1）将链接寄存器 LR 的值减去相应的偏移量后送到 PC 中。

（2）将 SPSR 复制回 CPSR 中。

（3）如果进入时设置了中断禁止位，那么清除该标志。

285.分析并总结 S3C2440 的中断源类型。

答：60个中断源的分类：S3C2440芯片管理60个中断源，可以按2种规则分类。

1. 按中断源相对芯片的位置分类：

（1）内部中断源：中断源来自 S3C2440芯片内部， 36个。

（2）外部中断源：中断源来自 S3C2440芯片外部， 24个。

2. 按中断源能否被直接响应（响应直接性）分类：

（1）一级中断源：中断源产生的中断请求可以被直接响应，23个。

（2）二级中断源：中断源产生的中断请求不能被直接响应， 37个。二级中断源需要通过不同的逻辑复合形成新的中断源（即复合中断源），然后才能得到处理器的响应。

32个中断请求的分类：SRCPND 保存32个中断源的中断请求状态。

（1）直接中断源：中断源具有单一性，中断请求的来源单一，一旦中断请求被响应，可以直接定位中断来源， 23个。

（2）复合中断源：中断源具有多源性，中断请求是由多个其他中断源的中断请求复合在一起的，一旦中断请求被响应，需要通过其他方法定位中断来源， 9个。

286.Flash 存储器主要有（　　）和（　　）两大类。答案为：Nor 、Nand

287.S3C2440A 内部有 3 个时钟，分别是 ARM 的内核时钟 （　　），AHB 总线时钟（　　）和 I/O 接口的时钟（　　）。答案为：FCLK、 HCLK 、PCLK （顺序不能错）

288.Nand　Flash 比 Nor　Flash 成本高，可靠性差。（×）

289.ARM 处理器采用 AMBA 总线架构，分别是先进系统总线　　（　　）、先进高性能总线（　　）和先进外围总线（　　）。答案为： ASB、 AHB、 APB （顺序不能错）

290. Flash 存储器应该接在（　　）总线上。B

A、ASB　　B、AHB　　C、APB　　D、CAN

291.某公司生产的 Nand Flash 存储器的组织结构为 2048block×32page×2048byte ，则其容量是（　）。B

A、64MB　　B、 128MB　　C、32MB　　D、 256MB

292. S3C2440A 的电源管理有 4 种模式：（　　）模式、（　　）模式、（　　）模式和（　　）模式。答案：正常；慢速；空闲；睡眠（顺序可颠倒）

293. S3C2440A 共有 （　）个 GPIO 口;可分为 （ ）组。答案：130 ， 9

294. S3C2440A 的 GPIO 口只能作为输出使用的是（　）。B

A 、GPB 口　　B、GPA 口　　C、 GPF 口　　D、GPG 口

295.设 S3C2440A 处理器的 PCLK 的频率 f pclk =40MHz，经过 1/100 预分频和 1/4 分频后，送给定时器的计数时钟周期（f TCLK）的输出周期是 （　　） μs 。C

A 、100　　B、1　　C、10　　D 、0.1

296.使用 S3C2440A 处理器的定时器 timer1 来产生频率为 1KHz、占空比为 30%的方波。已知定时器 timer1 的时钟的输出频率 f TCLK 是 1 MHz，则定时器 timer1 的计数初值寄存器 TCNTB1 的数值应设定为（　　），比较寄存器 TCMPB1 数值应设定为（　　）。100 ,30

297.假设某嵌入式系统底层通讯采用 DMA 数据传输，如图所示是未完成的 DMA 数据传输工作流程图，请从下面①~⑧中选择正确的答案，完成该图，将答案填写在答题纸的对应栏中。

备选答案：

① 字计数器计数

② DMA 发送中断

③ DMA 响应

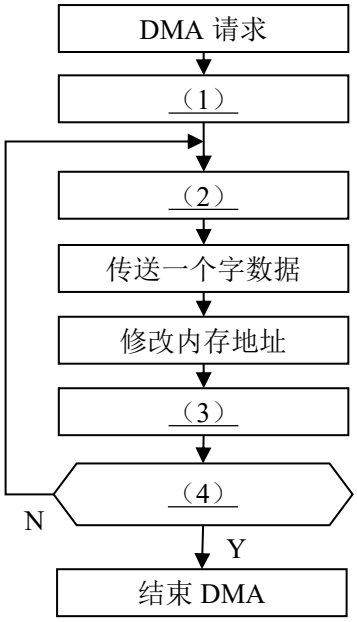
④ DMA 接收 4 个字节

⑤ 发送内存地址

⑥ 再次修改内存地址

⑦ 传送结束

⑧ 继续传送



1. 题三-1 图

答:（1）DMA 响应，或③

（2）发送内存地址，或⑤

（3）字计数器计数，或①

（4）传送结束，或⑦

298.S3C2440 的中断模式有哪两种？

299.S3C2440 的中断控制寄存器有几个，每个的作用是什么？

300.S3C2440 的中断源未决寄存器和中断未决寄存器的区别和作用有哪些？

答：中断源未决寄存器的作用：保存中断源申请中断的状态。
中断未决寄存器的作用：保存经过中断优先级判别后最高优先级的申请中断状态。

301.如何清除中断请求？

302.如何使能某中断源申请的中断？

303.如何屏蔽某中断源申请的中断？

304.外部中断 0（EXTINT0）通过 F 口的 GPF0、外部中断 11（EXTINT11）通过 G 口的 GPG3 向 CPU 申请中断，此时两个口的控制寄存器 GPFCON 和 GPGCON 如何设置？

305.S3C2440 中断有几种触发方式？如何选择中断触发方式？

306.定时器配置寄存器 TCFG0[23：16]，[15：8]，[7：0]各有什 么作用？

307.定时器配置寄存器 TCFG1[19：0]各有什么作用？

308.定时器控制寄存器 TCON[22：0]各有什么作用？

309.寄存器 TCNTBn，TCMPBn 作用有什么不同?

310.PWM 定时器软件编程步骤有几步？

311.简述脉宽调制原理及用法。

312.如何改变 PWM 输出频率？

313.如何改变 PWM 输出占空比？

314.说明 4 线电阻式触摸屏的工作原理。

答：触摸屏工作时，上下导体层相当于电阻网络。当某一层电极加上电压时，会在该网络上形成电压梯度。如有外力使得上、下两层在某一点接触，则在另一层未加电压的电极上可测得接触点处的电压，从而知道接触点处的坐标。例如，在顶层的电极（X+，X-）上加上电压，则在顶层导体层上形成电压梯度；当有外力使得上、下两层在某一点接触时，在底层（Y+，Y-）电极上就可以测得接触点处的电压；再根据该电压与电极（X+）之间的距离关系，即可知道该处的 X 坐标；然后，将电压切换到底层电极（Y+，Y-）上，并在顶层（X+，X-）电极上测量接触点处的电压，从而确定 Y 坐标。

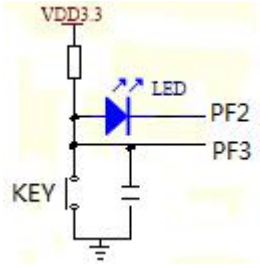
315.在嵌入式系统的存储结构中，存取速度最快的是_____。B

A 内存　　B、寄存器组　C、Flash　　D、Cache

316.电路如下图所示，GPF2、GPF3 是 S3C2440 两个引脚，编程可能涉及到寄存器 rGPFDAT、rGPFUP、rGPFCON，完成下列工作：

1.阐述电路工作原理？

2.编程实现如下功能：初始条件 LED 不亮，按键 KEY 按下奇数次 LED 点亮，按键 KEY 按下偶数次 LED 不亮。



答：1、GPF2送低电平，LED 亮。Key 按下时，GPF3为低电平。

2、（1）、初始化

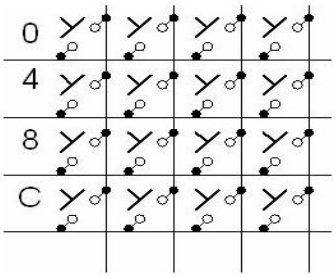
（2）、判断按键

（3）、LED 控制

（4）、其他

317.矩阵式键盘逻辑电路如图：

要求完成下列功能：



（1）、使用 ARM 处理器 S3C2440 的 GPF 口设计键盘接口电路。

（2）、编写初始化子程序。

（3）、编写按键识别子程序并说明按键识别原理

答：（1）硬件连线图

（2）配置寄存器

(3) 按键识别子程序

318. S3C2440 自带一个（ ）A/D 转换器。A

A．8 路 10 位 B．10 路 8 位 C．8 路 12 位 D．12 路 8 位

319. S3C2440 在 UART 连接 UART 时支持用（ ）信号进行自动流控制。B

A．RxD 和 TxD B．nRTS 和 nCTS C．nRTS 和 RxD D．nCTS 和 TxD

320. S3C2440 具有（ ）个中断源。D

A．24 B．32 C．48 D．60

321. IRQ 中断处理程序可以执行指令 SUBS PC , R14_irq , #4 从 IRQ 中断返回，说明指令中减 4 的原因。

答：三级流水线

322. UART 操作出现的帧错误是指（ ）。A

A．接收到的数据没有有效的停止位

B．新的数据已经覆盖了旧的数据

C．接收器检测到了意料之外的奇偶校验结果

D．RxDn 的输入被保持为 0 状态的时间超过了一个帧传输的时间

323. S3C2440 内部集成了 UART。在 UART 的操作中，会出现哪些错误状态？简要描述这些错误状态。

答：① 溢出错误：新的数据已经覆盖了旧的数据，因为旧的数据没有及时被读入。

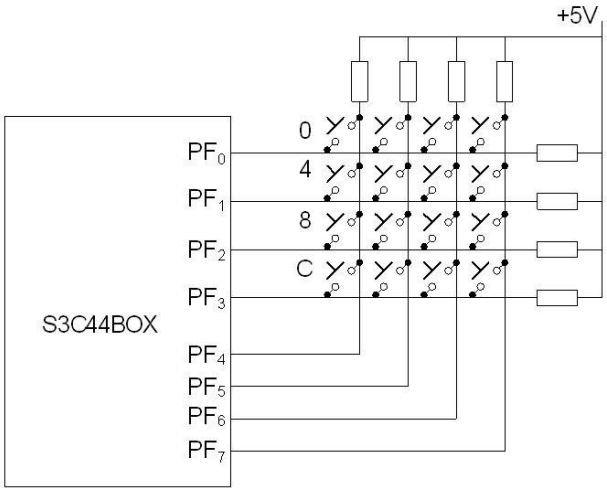
② 奇偶校验错误：接收器检测到了意料之外的奇偶校验结果。

③ 帧错误：接收到的数据没有有效的停止位。

④ 中止状况：RxDn 的输入被保持为0状态的时间超过了一个帧传输的时间。

⑤接收超时：在 FIFO 模式下，接收 FIFO 不应为空，但当接收器在3个字时间内都没有接收到任何数据时，就认为发生了接收超时状况。

324.叙述行扫描法识别键盘的工作过程。根据给出的电路图及相关寄存器内容编写行扫描法获得键值程序，C 语言实现。



PF 口的寄存器有 3 个：PF 口数据寄存器 PDATF、PF 口上拉电阻寄存器 PUPF 和 PF 口控制寄存器 PCONF。

答：键盘某一行为低，其余接高，读取列值，低电平表示键按下，依此原理可以判断按键，

控制寄存器值是否正确（包括端口说明，那些线是输入线，那些线是输出线，根据说明给出控制寄存器的值），按照行扫描原理编写程序

参考程序：

```
#include <string.h>
#include <stdio.h>
char ReadKeyVal(void)
{
unsigned char    i,j,H_val,L_val;
char keyval= -1;
rPCONF = 0x55;
rPUPF=0x00;
rPDATF=0xf0;
if((L_val=(rPDATF&0xf0))!=0xf0)
{//有键按下
H_val=0xfe;//行值，从第 0 行开始判断
for(i=0;i<4;i++)
```

```
{
rPDATF=H_val;//行电平输出
for(j=0;j<100;j++);//软件延时
if((L_val=(rPDATF&0xf0))!=0xf0)
{//该行有没有键被按下
L_val=( (L_val>>4))|0xf0;//设置行值格式
Keyval =get_val(H_val)× 4 + get_val(L_val);
return keyval;
}
else
H_val = H_val <<1;//判断下一行
}
}
return keyval;
}
//以下计算键值的程序可以不同
char get_val(unsigned char val)
{
unsigned char i,x;
x=0;
for(i=0;i<4;i++)
{
if((~val)==1)    return x;//全 1 返回
val = (val>>1)|0x80 ;
x = x+1 ;
}
}
```

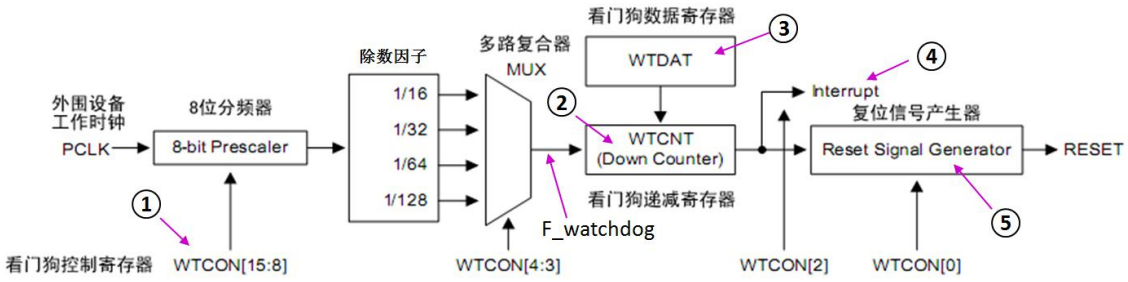
325. S3C2440 的 UART 接口具有哪些功能特性？

326.假设要将 S3C2440 的 UART1 设置为：波特率 9600b/s，7 位数据位，2 个停止位，1 位奇偶校验位，并采用流控制工作，该如何设置？给出完成该设置功能的代码段。

327.S3C2440 内部的看门狗定时器逻辑如图所示，完成下列问题：

（1）对图中标注的①-⑤的部件或信号，按序号进行原理、功能、作用方面的描述或说明。

（2）若 PCLK=50MHz，8 位预分频器的值=249，频率除数因子=16，求出 F_watchdog 的频率值。



答：（1）①WTCON：看门狗控制寄存器，保存各个功能部件的控制信息

②WTCNT：看门狗计数寄存器，16位，保存当前计数值。一旦看门狗使能，WTCNT 里的数据就开始减1计数

③WTDAT：看门狗数据寄存器，16位，保存计数初值。当 WTCNT 中的数据减1为0时，WTDAT 中的计数初值自动加载到 WTCNT 中

④Interrupt：中断信号，由 WTCON 寄存器控制是否产生中断信号。若允许，则 WTCNT 计数值减到0时，产生中断信号

⑤Reset Signal Generator：复位信号产生器，由 WTCON 寄存器控制是否产生复位信号。若允许，则 WTCNT 计数值减到0时，产生复位信号，让控制器重启

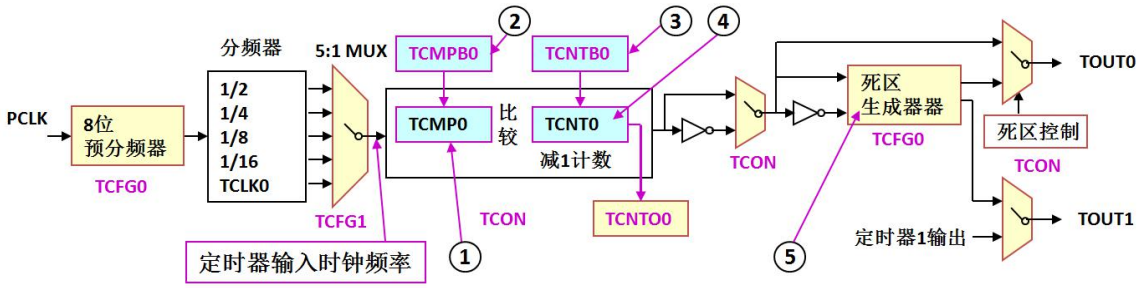
（2）F_watchdog = PCLK ÷（8位预分频器的值　+　1）÷频率除数因子

= 50MHz ÷（249　+　1）÷16 = 12.5kHz

328.. S3C2440 内部的 Timer 部件逻辑如图所示，完成下列问题：

（1）对图中标注的①-⑤的部件或信号，按序号进行原理、功能、作用方面的描述或说明。

（2）假设 PCLK=50MHz，预分频值=249，分频值=8，求出定时器输入时钟的频率值。



答：（1）①TCMP0：定时器0比较寄存器，16位，保存用于 PWM 电平翻转的比较数值。当 TCNT0 数值= TCMP0数值时，TOUT0输出的电平信号翻转。内部寄存器，不能直接进行读写操作

②TCMPB0：定时器0比较缓冲寄存器，16位，保存用于 PWM 电平翻转的比较数值初值

③TCNTB0：定时器0计数缓冲寄存器，16位，保存计数初值

计数初值 = 定时时间 / 定时器输入时钟周期-1

④TCNT0：定时器0计数寄存器，16位，保存正在计数的数值，每个时钟减1。减到0，在TOUT0输出一个电平信号。内部寄存器，不能直接进行读写操作

⑤TCFG0：定时器配置寄存器0，参数之一是确定死区长度，范围0-255。死区：关闭一个开关设备和开启另一个开关设备之间插入的时间间隔。这个时间间隔可以防止两个设备同时被启动

（2）定时器输入时钟频率 = PCLK / (预分频值+1) /(分频值) = 50MHz/(249+1)/8 = 25KHz

329.结合 S3C2440 芯片 PWM 定时器的双缓冲结构，说明 PWM 定时器的工作原理。

答：PWM：Pulse Width Modulation，脉宽调制，通过对周期性序列脉冲的脉冲宽度进行调制，等效地获得所需要的波形。广泛应用于测量、通信、功率控制与变换等许多领域。

平均电压与脉宽的关系：

$$Uave = t / T \times Umax$$

Uave：周期内平均电压

t：脉冲宽度时间

T：周期时间

Umax：脉冲最大电压

占空比：高电平时间 / 周期时间，比如，20%占空比，会有20%的高电平时间和80%的低电平时间。

S3C2440定时器具有双缓冲结构（TCNTBn、TCMPBn），

PWM 输出：定时器设置为自动加载模式。

在 TCNTn 减1计数过程中， TCNTn 与 TCMPn 进行比较，当 TCNTn = TCMPn 时，TOUTn 输出的电平会翻转。TCNTn 继续减1计数，减1到0后，TOUTn 输出再次翻转，TCNTn、TCMPn 又会重新加载 TCNTBn、TCMPBn 中的初值，重新开始计数，周而复始，产生周期性脉冲序列 PWM 输出。

330. S3C2440 的存储器 Bank0 作为引导的 ROM，Bank0 的总线宽度只能设置为 16 位和 32 位，并通过控制引脚 OM[1:0]来决定 Bank0 的总线宽度。

331.请说明 S3C2440 有几种启动方式？它们分别是如何启动的？

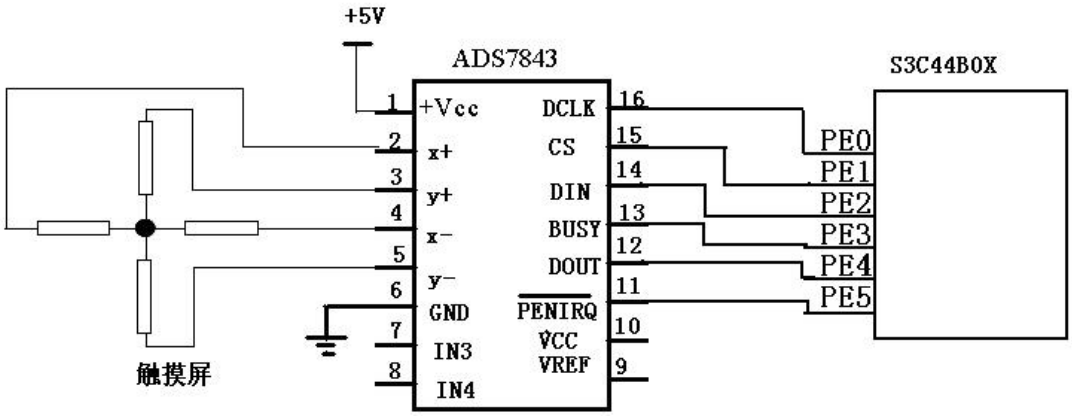
答：1.Nand Flash

OM[1:0]=00,Nand Flash 前 4KB 代码复制 sy（steping stone）区 SRAM 映射到 0X00000000 地址处执行。

2.Nor Flash

直接从低地址 0X00000000 地址处执行。

332. 触摸屏电路由触摸屏、触摸屏控制器 ADS7843、S3C44B0X 组成，电路连接原理如图所示，请回答如下 3 个问题：说明 4 线电阻式触摸屏的工作原理；给出 E 口初始化命令字，使 S3C44B0x 通过 E 口可以向 ADS7843 发送读取 X、Y 坐标命令，接收来自 ADS7843 的 X、Y 坐标值；根据 ARM 向 ADS7843 传送命令字时序，编写函数，实现向 ADS7843 传送命令字功能。



答：4 线电阻式触摸屏的工作原理：

触摸屏工作时，上下导体层相当于电阻网络。当某一层电极加上电压时，会在该网络上形成电压梯度。如有外力使得上、下两层在某一点接触，则在另一层未加电压的电极上可测得接触点处的电压，从而知道接触点处的坐标。例如，在顶层的电极（X+，X-）上加上电压，则在顶层导体层上形成电压梯度；当有外力使得上、下两层在某一点接触时，在底层（Y+，Y-）电极上就可以测得接触点处的电压；再根据该电压与电极（X+）之间的距离关系，即可知道该处的 X 坐标；然后，将电压切换到底层电极（Y+，Y-）上，并在顶层（X+，X-）电极上测量接触点处的电压，从而确定 Y 坐标。

E 口初始化命令字

E 口0,1,2输出

3,4,5输入

配置命令字：

rPCONE=0B 01 01 01　00 00 00=0x540

编写函数

Command 为命令字

```
rPDATE &= OxFC;          //CS 置低;DCLK 置低，PE1,PE0
        temp =0x80;      //设置要传送的位
        for(i＝0; i<8; i＋＋)          //发送 1 个字节
            {
    if(command＆temp)    rPDATE|＝0x04;    //将 DIN 置 1
            else         rPDATE＆＝0xFB; 将 DIN 清 0，  PE2
            rPDATE |= 0x01；          //DCLK 置高，  PE0
            delay(2)；
            rPDATE＆= 0xFE;//清除 DCLK,1 位送出，  PE0
            delay(2)；
            temp =temp>>1；                  //右移 1 位
        }
```

333.S3C2440 的 I/O 端口在配置时，一般需要配置三种寄存器，它们分别是什么寄存器？作用是什么？

答：S3C2440有9个端口，每个端口一般对应3个寄存器（端口 A 只有2个）： 控制寄存器（***CON）和数据寄存器（***DAT）、上拉电阻寄存器（***UP） 。

比如端口 B 的寄存器表示为：GPBCON、GPBDAT、GPBUP。

控制寄存器功能：配置 I/O 引脚的功能，即一个 I/O 引脚选择哪个功能。控制寄存器有自己的寻址地址。

数据寄存器功能：保存输入、或者输出的数据。数据寄存器有自己的寻址地址。

上拉电阻寄存器：确定端口 B 的 GPIO 引脚是否内部接上拉电阻。上拉电阻寄存器有自己的寻址地址。

334.简答 S3C2440 处理器分别采用 8 位、16 位、32 位数据总线时，如何设计处理器的地址线与存储器的地址连接？

	8 位	16 位	32 位
存储器	A0	A0	A0
处理器	A0	A1	A2

335. S3C2440 的存储空间由哪几部分组成？各部分有什么特点？

答：S3C2440的存储空间分成8组，最大容量是1GB，bank0---bank5为固定128MB，为只读存储器；bank6和 bank7的容量可编程改变，可以是2、4、8、16、32、64、128MB，并且bank7的开始地址与 bank6的结束地址相连接，但是二者的容量必须相等，既可以作为程序存储器，也可以作为数据存储器，一般这一部分做 RAM 使用；bank0可以作为引导 ROM，其数据线宽只能是16位和32位，复位时由 OM0、OM1引脚确定，其他存储器的数据线宽可以是8位、16位和32位。

336.编写一程序，用查询的方式，对 S3C2414 的 A/D 转换器的第 0 通道连续进行 100 次 A/D 转换，然后将其结果求平均值。 注意：A/D 转换器有独立的模拟信号输入引脚 AIN0—AIN9。

答：#define rADCCON (*(volatile unsigned *)0x58000000)

#define rADCDAT0 (*(volatile unsigned *)0x5800000c)

#define pref 49

#define ch 0

int adc(void) {

rADCCON=(1<<14)|(pref<<6)|(ch<<3)|1; //允许预分频，启动转换

while(rADCCON&0x01==1); //查询是否已经启动转换

while(rADCCON&0x8000==0); //查询转换是否结束

return rADCDAT0&0x3ff; //读取转换结果 }

void main() {

int adc_data=0, i;

for(i=0;i<100;i++)

adc_data+=adc();

adc_data=adc_data/100;

printf("adc average is: %d\n"，adc_data);

}

337.编写一程序，使用 timer0 产生并输出频率为 10KHz、占空比为 1/2 的方波。设

fpclk=50MHz.(注意对　timer0 和相关引脚初始化)

答：总分频数值=50M/10K=5000

设预分频为25（预分频值为25-1=24），分频为2，计数器初值应该为100，这样总分频数值=25*2*100=5000　　所以：

计数初值为100，

比较寄存器值为100/2=50

TCFG0=24

TCFG1=TCFG1　&　0　|　(0<<20)　|　(0<<0)

TCON=TCON　&　~(0x0F)　|　0x0a

TCON=TCON　&　~(0x0f)　|　0x09

程序如下：

```
#define rGPBCON     (*(volatile unsigned *)0x56000010)
#define rTCFG0      (*(volatile unsigned *)0x51000000)
#define rTCFG1      (*(volatile unsigned *)0x51000004)
#define rTCON       (*(volatile unsigned *)0x51000008)
#define rTCNTB0     (*(volatile unsigned *)0x5100000C)
#define rTCMPB0     (*(volatile unsigned *)0x51000010) void main(void) {  rGPBCON=rGPBC
ON & ~0x03 | 0x02;        //设置 T0 输出引脚
rTCFG0=24;        //设置预分频
rTCFG1=0;         //设置 T0(中断/DMA)模式和分频
rTCNTB0=100;        //设置 T0 初值
rTCMPB0=50;        //设置 T0 比较值
rTCON=rTCON & ~0x0f | 0x0a;   //设置控制寄存器，手动装载 T0 初值
rTCON=rTCON & ~0x0f | 0x09;   //设置控制寄存器，自动重装、启动运行 }
```

338. S3C2440 与触摸屏接口有几种接口模式？各有什么特点

答：S3C2440与触摸屏接口有5种接口模式。（1）普通的 A/D 转换模式，在普通的 A/D 转换模式，AUTO_PST=0，XY_PST=0。（2）分开的 X/Y 位置转换模式，分开的 X/Y 位置转换模式由 X 位置模式和 Y 位置模式两种转换模式组成。（3）自动（顺序）X/Y 位置转换模式，当 ADCTSC 寄存器的 AUTO_PST=1和 XY_PST=0时进入自动（顺序）X/Y 位置转换模式模式。（4）等待中断模式，当 ADCTSC 寄存器的 XY_PST=3时，进入等待中断模式模式。在等

待中断模式，等待触笔点下。（5）待机模式（Standby Mode），当 ADCCON 寄存器的 STDBM 位设置为1时，进入待机模式。进入待机模式模式后， A/D 转换停止，ADCDAT0的 XPDATA 和 ADCDAT1的 YPDATA 保持上次转换的数值。

339. ARM9 具有几个 32 位定时器？PWM 定时器是否可以作通用定时器使用？

答：两个32位定时器，PWM(脉冲宽度调制)定时器不能用作通用定时器使用

340.阅读下列与看门狗有关的寄存器描述，解释每一行代码的功能。

看门狗定时器控制寄存器（WTCON）

看门狗定时器数据寄存器（WTDAT）

```
#define rWTCON   (*(volatile unsigned *)0x53000000) // 第 1 行
#define rWTDAT   (*(volatile unsigned *)0x53000004) // 第 2 行
#define rWTCNT   (*(volatile unsigned *)0x53000008) // 第 3 行
void watchdog_test(void) {
rWTCON = ((PCLK/1000000-1)<<8)|(3<<3)|(1<<2);        // 第 4 行
rWTDAT = 7812;         // 第 5 行
rWTCNT = 7812;             // 第 6 行
rWTCON |=(1<<5);         // 第 7 行
}
```

答：第1-3　行：定义看门狗控制寄存器、数据寄存器和计数寄存器为 rWTCON、rWTDAT 和 rWTCNT。

第4行：设置看门狗的预装比例值为1000000，分频因素为1/128，并使能中断。

第5-6　行：对数据寄存器和计数寄存器赋值为7812。

第7行：启动看门狗。

341.阅读以下 S3C2440 部分用户手册．求：当 PCLK　或　UCLK　为 40　MHz 时，串口 0 的波特率为２４００　bps　，串口 1 的波特率为１１５２００ bps，相应的控制寄存器如何设置．

UART　BAUD　RATE　DIVISOR　REGISTER

There are three UART baud rate divisor registers(寄 存 器) including UBRDIV0, UBRDIV1 and UBRDIV2 in the UART block （ 模块）. The value stored in the baud rate divisor register (UBRDIVn), is used to determine the serial Tx/Rx clock rate(baud rate) as follows:

UBRDIVn = (int)(PCLK / (bps x 16)) –1 or UBRDIVn = (int)(UCLK / (bps x 16)) –1 Where, the divisor should be from 1 to (216-1) and UCLK should be smaller than PCLK.

UBRDIVn 寄存器

答: 根据 UBRDIVn = (int)(PCLK / (bps x 16)) －1

寄存器 UBRDIV0= (int)(40000000/2400*16)-1=1040=10000010000(B)

寄存器 UBRDIV1=(int)(40000000/115200*16)-1=20=10100(B)

342.获取数字声音的过程中必须进行"取样"、"量化"等处理。下面关于"量化"的叙述中错误的是：

A) 量化就是把声音样本的模拟量转换成数字量来表示

B) 量化过程往往也称为 D/A 转换

C) 量化位数增多，量化的精度可以提高，声音的保真度也更好

D) 量化位数越少，数字声音的数据量也越少

【解析】音频信息数字话的过程是取样、量化、编码。其中量化是把每个样本的模拟值转换成数字量来表示，因此量化过程往往也称为 A/D 转换（模数转换）。量化位数增多，量化的精度可以提高，声音的保真度也更好，量化位数越少，数字声音的数据量也越少。故本题选 B。

343.下面关于嵌入式系统存储器的叙述中，错误的是：

A) 目前嵌入式处理器内部的 Cache 采用 SRAM

B) 嵌入式系统使用的存储器按照其存取特性可分为随机存取存储器（RAM）和只读存储器（ROM）

C) 铁电存储器（FRAM）和磁性存储器（MRAM）是两种新型的半导体存储器

D) 通过对 DRAM 的存储控制技术进行改进，出现了 DDR2 SDRAM、DDR3 SDRAM 等新型的存储器产品

【解析】嵌入式系统的存储器以半导体存储器为主。按照其存取特性可分为 RAM 和 ROM；使用的 RAM 有 SRAM、DRAM 等多种，目前嵌入式处理器内部的 Cache 采用 SRAM，通过对 DRAM 的存储控制技术进行改进，出现了 DDR2 SDRAM、DDR3 SDRAM 等新型的存储器产品；新型存储器 FRAM 和 MRAM 均非传统的半导体存储器。故 C 项错误。

344.下面关于 ARM 嵌入式处理器的 GPIO 的叙述中，错误的是：

A) GPIO 的引脚一般是三态的，即具有 0 态，1 态和高阻状态

B)　有些 GPIO 引脚具有多种功能，通过设置相关控制寄存器的某些位来进行选择

C)　有些 ARM 芯片的 GPIO 引脚可以设置成具有中断输入功能

D)　只有几个按键的简单键盘接口，应采用专用的键盘接口芯片来实现，而不宜采用 GPIO 来设计

【解析】GPIO 一般具有三态，即 0 态、1 态和高阻状态；为了节省引脚条数，通常有些 GPIO 引脚有多种功能以供选择，可以通过设置相关控制寄存器的位来确定引脚功能；有些 ARM 芯片，如新唐科技的 Cortexc-M0 芯片每个引脚多可以设置成中断输入；在嵌入式应用系统中，少数几个按键作为简单键盘的应用非常普遍，通常可应用 GPIO 引脚构建简单键盘。故本题选 D。

345.1 下面是关于嵌入式系统中使用的无线通信接口或技术的叙述，其中错误的是：

A)　GPRS 是 GSM 用户可用的一种移动数据业务，通常支持用 AT 指令集进行呼叫、短信、传真、数据传输等业务

B)　使用 802.11 系列协议的无线局域网也称为 WiFi

C)　蓝牙是一种支持短距离通信的无线低速通信技术，它采用分散式网络结构以及快跳频和短包技术，支持点对点及点对多点通信

D)　嵌入式系统可通过扩展无线模块来实现无线通信，该模块与嵌入式处理器连接时一般只能采用 UART

【解析】GPRS 是 GSM 用户可用的一种移动数据业务，通常支持用 AT 指令集进行呼叫、短信、传真、数据传输等业务；凡使用 802.11 系列协议的无线局域网又称为 WiFi；蓝牙是一种支持短距离通信的无线低速通信技术，它采用分散式网络结构以及快跳频和短包技术，支持点对点及点对多点通信；嵌入式系统中的常用无线模块主要包括 GPS、GPRS、WiFi、蓝牙及通用射频通信模块等，通信连接接口有 UART，也有基于 USB 的。故 D 项错误。

346.1

347.1

348.1

349.1

350.1

351.1

352.1

353.

第 6 章 嵌入式接口技术应用

354.LCD 特点，LCD 基本显示原理，LCD 的 3 种显示方式，CRT 显示器的光栅扫描，LCD 控制结构

355.LCD 显示屏没有驱动电路，需要与驱动电路配合使用。√

356.属于 LCD 三种显示方式的是（　　）。D

A．投射型、反射型、透射型　　B．投射型、透反射型、透射型

C．投射型、反射型、透反射型　　D．反射型、透射型、透反射型

357.智能手机设计涉及硬件、软件、通信、安全、传感器等多方面的技术和因素，具有较高的综合性，包含多个相互关联的子问题，属于复杂工程问题。假设让你设计一个未来的智能手机，回答下面问题：

（1）请展开你的想象，说明你希望该手机具有的创新性功能（至少 3 个），并简要说明每个功能如何实现。

（2）画图说明嵌入式系统的基本组成，并根据这种组成结构，以 S3C2440 为核心画出手机的系统组成框图。

（3）分析在手机设计环节中如何考虑社会、健康、安全、法律、文化、环境等 6 个方面的因素。

358.S3C2440 的 LCD 控制器支持 4096 彩色 LCD 显示屏，这种显示方式下有几种像素颜色模式（即像素点在存储器中保存的二进制格式），具体描述这些像素颜色模式。

答：利用时间抖动算法和帧频控制算法，彩色显示模式下可产生 4096 种颜色。

12BPP： 4096 种颜色，12 个位表示一个像素点。

像素点在存储器中保存的格式为 4:4:4 模式：4 位红色、4 位绿色、4 位蓝色。由于 4 个位编码恰好对应每个颜色的 16 种，所以， 12BPP 的每种颜色的 16 个颜色级别不需要查找表。

359.S3C2440 的 LCD 控制器支持 64K 彩色 LCD 显示屏，这种显示方式下有几种像素颜色模式（即像素点在存储器中保存的二进制格式），具体描述这些像素颜色模式。

答：

像素点在存储器中保存的格式有 2 种： 5:6:5、5:5:5:1

5:6:5 模式：5 位红色、6 位绿色、5 位蓝色。

5:5:5:1 模式：5 位红色、5 位绿色、5 位蓝色、1 位亮度。数据传送时，亮度 I 位用于每个 RGB 数据的最低位，即 I 位对于每个 RGB 数据有效。

360.S3C2440 的 LCD 控制器支持 16M 彩色 LCD 显示屏，这种显示方式下有几种像素颜色模式（即像素点在存储器中保存的二进制格式），具体描述这些像素颜色模式。

答：

24BPP： 16M 种颜色，24 个位表示一个像素点。不需要查找表。

像素点在存储器中保存的格式为 8:8:8 模式：8 位红色、8 位绿色、8 位蓝色。

361.一幅没有经过数据压缩的彩色图像，其数据量是 768KB，分辨率为 1024*768,那么它每个像素的像素深度是：

A) 24 位 B) 16 位 C) 12 位 D) 8 位

【解析】数字图像数据量=分辨率*像素深度/8，故本题像素深度为 768KB*8/（1024*768）=8 位，选 D。

362.S3C2440 的 LCD 控制器支持虚拟显示，如何理解这一功能？

答：S3C2440 的 LCD 控制器支持虚拟显示，就是指可以显示比实际显示器大的图像。在参数设置时，虚拟显示窗口的宽度和高度就是实际显示器的宽度和高度，大图片的像素信息全部存放在视频缓冲区中，通过调整虚拟显示窗口在视频缓冲区的起始位置，显示大图片的不同部分内容，相当于大图片不动、虚拟显示窗口在大图片上移动。

363.下面关于嵌入式系统中常用的简单输入设备和简单输出设备的叙述中，错误的是：

A) 嵌入式系统中使用的键盘有线性键盘和矩阵键盘两类

B) 电阻式触摸屏和电容式触摸屏是嵌入式系统中常用的两种触摸屏

C) LCD 是发光二极管的简称

D) 液晶显示器是嵌入式系统常用的一种显示设备

【解析】嵌入式系统中使用的键盘有线性键盘和矩阵键盘两类；电阻式触摸屏和电容式触摸屏是嵌入式系统中常用的两种触摸屏；液晶显示器是嵌入式系统常用的一种显示设备；放光二极管是 LED，LCD 是液晶显示器，故本题选 C。

364.1

365.1

366.1

367.1

368.1

369.1

370.1

371.