厦门大学《嵌入式系统》课程期末试卷

信息学院　软件工程系　2020 级　软件工程专业

主考教师：曾文华 试卷类型：（B 卷）答案 考试时间：2023. 2. 15

一、填空题（30 个空，每 1 空 1 分，共 30 分；在答题纸填写答案时请写上每个空格的对应编号）

1. ARM 指令格式为 <opcode> {<cond>} {S} <Rd>, <Rn> {, <shift_op2>}，其中，cond 为 执行条件（条件码）（1），S 为 可选后缀（2），shift 为 位移操作（3）。

2. μCLinux 中的 μ 表示 Micro（微）（4），C 表示 Control（控制）（5），μCLinux 是专门针对没有 MMU（存储管理单元）（6）的处理器设计的。

3. NAND Flash 的特点是容量 大（7）、价格 低（8）、访问方式是 顺序（9）读写；NOR Flash 的特点是容量 小（10）、价格 高（11）、访问方式是 随机（12）读写。

4. 嵌入式系统主要的存储设备为 Flash Memory（闪存）（13）和 SDRAM（14）。

5. Xshell2.0 是 Windows 环境下的 超级终端（15）；minicom 是 Linux（16）环境下的串口通信工具。

6. 远程调试时在实验箱的“Xshell 2.0”上要运行 gdbserver（17）程序，在 Ubuntu 的“终端”上要运行 arm-linux-gdb（18）程序。

7. Boot Loader 的操作模式有两种，分别是 启动加载模式（19）和 下载模式（20）。

8. ioremap()函数是将设备所处的 物理（21）地址映射到内核 虚拟（22）地址空间。

9. Android 系统的底层是由 Linux（23）操作系统作为内核。

10. 块设备驱动程序中没有读写操作函数，对块设备的读写操作都是通过 请求（24）实现的。

11. IMX6 嵌入式教学科研平台（实验箱）有二个 CPU，一个是主 CPU，采用飞思卡尔公司生产的基于 ARM Cortex-A9（25）的最新单核的 IMX6DL 嵌入式微处理器（Freescale IMX6DL）；另一个是从 CPU，采用 ARM Cortex-M3（26）内核架构的意法半导体公司生产的 STM32F103 芯片。

12. RS-232 双工方式通信时，至少需用 3 根线，这 3 根线分别是：TxD（27）、RxD（28）、GND（29）。

13. MobaXterm 是一款增强型 远程连接（30）工具。

二、名词解释（请写出下列英文缩写的中文全称，10 小题，每 1 小题 1 分，共 10 分；在答题纸填写答案时请写上每小题的对应编号）

1. SPSR：Saved Program Status Register，程序状态保存寄存器
2. TFTP：Trivial File Transfer Protocol，简单文件传输协议
3. ELF：Executable and Linkable Format，可执行与可链接格式
4. POSIX：Portable Operating System Interface of UNIX，可移植操作系统接口
5. JFFS2：Journalling Flash File System Version2，闪存日志型文件系统第 2 版
6. Ramfs：基于 RAM 的文件系统。
7. Cramfs：Compressed ROM File System，只读压缩的文件系统。
8. SMBus：System Management Bus，系统管理总线
9. Android JNI：Java Native Interface
10. OBS：Object Storage Service，对象存储服务

三、简答题（10 小题，共 30 分；在答题纸填写答案时请写上每小题的对应编号）

1. （3 分）ARM Cortex-A、ARM Cortex-R、ARM Cortex-M 系列处理器分为针对什么应用场合？

答：

（1）ARM Cortex-A 系列又称“高性能处理器”（Highest Performance），它是面向移动计算如智能手机、平板电脑和服务器市场定制的高端处理器内核，支持了包括 Linux、Android、Windows 和 iOS 等系统必须的内存管理单元（MMU），而且也是与我们平时接触最为密切的存在。

（2）ARM Cortex-R 系列实时处理器（Real-Time Processing）为要求可靠性、高可用性、容错功能、可维护性和实时响应的嵌入式系统提供高性能计算解决方案。Cortex-R 系列处理器通过已经在数以亿计的产品中得到验证的成熟技术提供极快的上市速度，并利用广泛的 ARM 生态系统、全球和本地语言以及全天候的支持服务，保证快速、低风险的产品开发。

（3）Cortex-M 系列针对成本和功耗敏感（Lowest Power, Lower Cost）的 MCU 和终端应用（如智能测量、人机接口设备、汽车和工业控制系统、大型家用电器、消费性产品和医疗器械）的混合信号设备进行过优化。

2. （3 分）请写出 ARM 指令的格式。

答：

<opcode> {<cond>} {S} <Rd>, <Rn> {, <shift_op2>}

< >内的项是必须的，{ }内的项是可选的

opcode：指令助记符（操作码），如 LDR，STR 等

cond：执行条件（条件码），如 EQ，NE 等

S：可选后缀，加 S 时影响 CPSR 中的条件码标志位，不加 S 时则不影响

Rd：目标寄存器

Rn：第 1 个源操作数的寄存器
op2：第 2 个源操作数
shift：位移操作

3. （3 分）什么是 Boot Loader？其作用是什么？常见的 Boot Loader 有那几个？

答：
（1）Bootloader：引导加载程序
（2）嵌入式系统（实验箱）启动后（打开电源，或者按 Reset 键），先执行 Bootloader，进行硬件和内存的初始化工作，然后加载 Linux 内核和根文件系统，完成 Linux 系统的启动。
（3）常见的 Boot Loader 有：U-Boot、vivi、Blob

4. （2 分）MMU（Memory Management Unit）的主要作用是什么？

答：
（1）地址映射
（2）对地址访问的保护和限制

5. （4 分）SPI 总线有 4 条接口线：（1）SDI（MISO）；（2）SDO（MOSI）；（3）SCLK（SCK）；（4）CS（SS）。请说明每条接口线的具体含义。

答：
SDI（MISO）：Serial Data In，串行数据输入；
SDO（MOSI）：Serial Data Out，串行数据输出；
SCLK（SCK）：Serial Clock，时钟信号，由主设备产生；
CS（SS）：Chip Select，从设备使能信号，由主设备控制。

6. （2 分）I2C 总线有两根接口线：（1）SDA；（2）SCK（或 SCL）。请说明每条接口线的具体含义。

答：
数据线：SDA
时钟线：SCK，或 SCL

7. （3 分）请简述设备驱动程序与应用程序的区别。

答：（1）应用程序一般有一个 main 函数，从头到尾执行一个任务。
（2）设备驱动程序却不同，它没有 main 函数，通过使用宏 module_init()，将初始化函数加入内核全局初始化函数列表中，在内核初始化时执行驱动的初始化函数，从而完成驱动的初始化和注册，之后驱动便停止等待被应用软件调用；驱动程序中有一个宏 module_exit()注册退出处理函数，它在驱动退出时被调用。
（3）应用程序可以和 GLIBC 库连接，因此可以包含标准的头文件，比如<stdio.h>、<stdlib.h>。
（4）在设备驱动程序中是不能使用标准 C 库的，因此不能调用所有的 C 库函数，比如输出打印函数只能使用内核的 printk 函数，包含的头文件只能是内核的头文件，比如<linux/module.h>。

8. （3 分）嵌入式系统开发中通常有两种方式运行应用程序，请说明这两种方式的名称和具体过程。

答：
（1）下载的方式：使用 FTP、TFTP 等软件，利用宿主机（虚拟机，电脑）与目标机（实验箱，平台）的网络硬件进行，此种方法通常是将宿主机端编译好的目标机可执行的二进制文件通过网线或串口线下载固化到目标机的存储器(FLASH)中。
在目标机嵌入式设备存储资源有限的情况下受到存储容量的限制，因此，在调试阶段通常的嵌入式开发经常使用 NFS 挂载的方式进行，而在发布产品阶段才使用下载方式。

（2）NFS 挂载方式：利用宿主机端（虚拟机，电脑）NFS 服务，在宿主机端创建一定权限的 NFS 共享目录，在目标机端（实验箱，平台）使用 NFS 文件系统挂载该目录，从而达到网络共享服务的目的。
这样做的好处是不占用目标机存储资源，可以对大容量文件进行访问。缺点是由于实际并没有将宿主机端文件存储到目标机存储设备上，因此掉电不保存共享文件内容。通常在嵌入式开发调试阶段，采用 NFS 挂载方式进行。

9. （3 分）IMX6 嵌入式教学科研平台（实验箱）从 CPU（第二个 CPU）的软件开发工具是什么？请简述该软件开发工具的安装过程。从 CPU 的 LED 灯实验程序是怎么执行的？

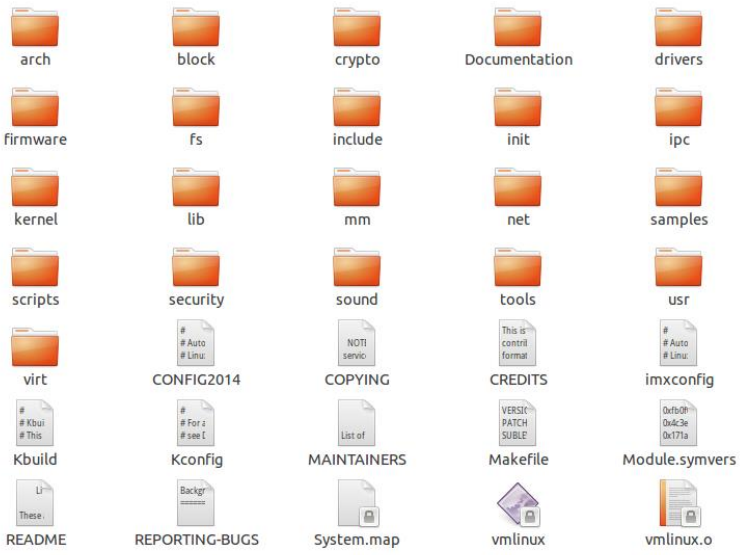
答：从 CPU 采用 MDK 嵌入式软件开发工具进行开发。
MDK 的安装过程包括：（1）安装 MDK5 主体；（2）安装对应 MCU 的 Pack 支持包；（3）破解；（4）安装调试工具 J-Link。
LED 灯实验程序的执行：首先打开 MDK5；然后打开 LED 灯工程，编译生成目标文件；下载目标文件到实验箱；按实验箱从 CPU 的 Reset 键，即可执行程序。

10. （4 分）简述云（ModelArts）+ 端（Atlas 200 DK）协同猫狗识别实验的具体步骤。

答：
第一步：在 ModelArts 中创建 Notebook 训练作业
第二步：在 Notebook 的 Jupyter 中运行训练作业，生成模型（.pb 格式）
第三步：将生成的模型从 OBS 下载到本地电脑硬盘
第四步：在 MobaXterm 环境下，完成模型的转换
第五步：从互联网上下载一些猫狗图片
第六步：在 MobaXterm 环境下，运行猫狗识别程序

四、综合题（9 小题，共 30 分；在答题纸填写答案时请写上每小题的对应编号）

1. （3 分）ARM-Linux 内核的代码目录结构如下（位于/home/uptech/fsl-6dl-source/kernel-3.14.28/目录下）：



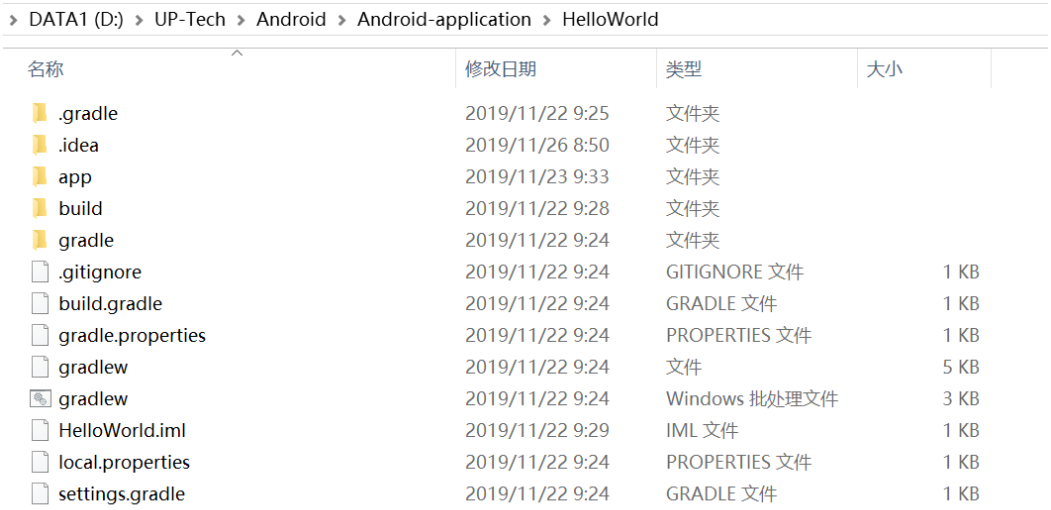
请问，该目录结构中的 arch、block、drivers、fs、mm、net 子目录中分别存放什么内容？

答：

1) arch ：包含和硬件体系结构相关的代码
2) block：块设备驱动程序
3) drivers ：设备驱动程序

4) fs：Linux 所支持的各种文件系统
5) mm：内存管理代码
6) net：网络相关代码

2. （3 分）Android Hello World 程序的工程文件夹如下：



请问该文件夹的 5 个子文件夹中存放什么内容？

答：

（1）.gradle 和.idea：这两个目录下放置的都是 Android Studio 自动生成的一些文件，我们无须关心，也不要去手动编辑。

（2）app：项目中的代码、资源等内容几乎都是放置在这个目录下的，我们后面的开发工作也基本都是在这个目录下进行的，待会儿还会对这个目录单独展开进行讲解。

（3）build：这个目录你也不需要过多关心，它主要包含了一些在编译时自动生成的文件。

（4）gradle：这个目录下包含了 gradle wrapper 的配置文件，使用 gradle wrapper 的方式不需要提前将 gradle 下载好，而是会自动根据本地的缓存情况决定是否需要联网下载 gradle。Android Studio 默认没有启动 gradle wrapper 的方式，如果需要打开，可以点击 Android Studio 导航栏 --> File --> Settings --> Build，Execution，Deployment --> Gradle，进行配置更改。

3. （2 分）以下的程序为汇编语言调用 C 语言的程序，请补充程序中 2 个划线处的内容。

```
int add(int x, int y)
{
    return(x+y);
}
```

IMPORT add　　@声明要调用的 C 函数
MOV r0, 1
MOV r1,2　　@通过 r0、r1 传递参数（参数传递规则）
BL add　　@调用 C 函数 add；返回结果由 r0 带回（子程序返回结果规则）

答：
IMPORT
BL

4. （2 分）我们在做实验时，通常采用挂载的方式，在实验箱的“超级终端（Xshell 2.0）”下，执行存放在 Ubuntu 中的可执行文件。此时运行实验箱的“超级终端（Xshell 2.0）”后，我们首先需要设置实验箱的 IP 地址，执行挂载命令，然后再运行可执行文件。设实验箱的 IP 地址为 59.77.5.120，Ubuntu 的 IP 地址为 59.77.5.122，需要将 Ubuntu 的“/imx6”目录挂载到实验箱的“/mnt”目录下，可执行文件（hello）存放在 Ubuntu 的/imx6/whzeng/hello 目录下。请写出设置实验箱的 IP 地址的命令，实现挂载功能的命令，以及运行 hello 可执行文件的命令。

答：

ifconfig eth0 59.77.5.120
mount -t nfs 59.77.5.122:/imx6 /mnt
cd /mnt/whzeng/hello
./hello

5. （4 分）如果我们不采用挂载的方式，而是采用下载的方式运行程序，即将 Ubuntu 中的可执行文件下载到实验箱中，再运行程序，请写出操作步骤（包括下载程序、运行程序）。设可执行文件（hello）存放在 Ubuntu 的/imx6/whzeng/hello/目录下，tftpd32.exe 文件（TFTP 服务）在 Windows 的 D:\UP-Tech\Linux 目录下，需要将可执行文件（hello）下载到实验箱的/home/root 目录中，Windows 系统的 IP 地址为 59.77.5.121。

答：
第一步：将 Ubuntu 下的/imx6/whzeng/hello/hello 文件，复制到 Windows 的 D:\UP-Tech\Linux 目录下（使用 Samba 服务）

第二步：在 Windows 下运行“tftpd32.exe”（TFTP 服务），将 tftpd32 的 Current Directory 设为 D:\UP-Tech\Linux，Server interface 设为 59.77.5.121。

第三步：在实验箱的“超级终端（Xshell 2.0）”下，执行：
cd /home/root
tftp -gr hello 59.77.5.121

第四步：在实验箱的“超级终端（Xshell 2.0）”下，执行：
chmod 777 hello

./hello

6. **（2 分）以下是 RS-485 驱动程序的头文件和全局变量，请问该程序的第 17)、第 18）行分别是做什么事情？**

```
1)  #include <linux/kernel.h>
2)  #include <linux/module.h>
3)  #include <linux/init.h>
4)  #include <linux/errno.h>
5)  #include <linux/fs.h>
6)  #include <linux/cdev.h>
7)  #include <linux/types.h>
8)  #include <linux/device.h>
9)  #include <asm/system.h>
10) #include <asm/uaccess.h>
11) #include <linux/platform_device.h>
12) #include <asm/irq.h>
13) #include <linux/of.h>
14) #include <linux/of_device.h>
15) #include <linux/of_gpio.h>
16) #define DRVNAME "UART485"
17) #define UART485_MAJOR 30
18) #define UART485_MINOR 0
19) #define UART485_TX 1
20) #define UART485_RX 0
```

答：

第 17）行：定义主设备号

第 18）行：定义次设备号

7. **（2 分）以下是 RS-485 驱动程序的模块初始化和模块退出函数，请填写程序中的 2 个空格部分的内容。**

```
static int __init gpio_uart485_init(void)
{
    printk("\n\n\nkzkuan___%s\n\n\n",__func__);
    return platform_driver_register(&gpio_uart485_device_driver);
}
```

```
static void __exit gpio_uart485_exit(void)
{
    printk("\n\n\nkzkuan___%s\n\n\n",__func__);
    platform_driver_unregister(&gpio_uart485_device_driver);
}
```

（1）_____(gpio_uart485_init);
（2）_____(gpio_uart485_exit);

答：
（1）module_init
（2）module_exit

8. （3 分）以下为 RS-485 双机通讯程序的一部分，请问该程序中的第 7)、8)、14）行分别是做什么事情？

```
1)  void* receive(void * data)
2)  {
3)    int c;
4)    printf("RS-485 Receive Begin!\n");
5)    for(;;)
6)    {
7)        ioctl(fd485, UART485_RX);
8)        read(fdCOMS1,&c,1);
9)        write(1,&c,1);
10)       if(c == 0x0d)
11)           printf("\n");
12)       if(c == ENDMINITERM)
13)           break;
14)       ioctl(fd485, UART485_TX);
15)   }
16)   printf("RS-485 Receive End!\n");
17)   return NULL;
18) }
```

答：
第 7）行：设置 RS-485 为接收模式
第 8）行：从 RS-485 中读 1 个字符
第 14）行：设置 RS-485 为发送模式

9. （4 分）以下为小键盘控制电子钟的主程序中的关键代码，请说明程序中的第 5)、12)、13)、19）行

的具体功能是什么？

```
1)  int main(int argc, char *argv[])
2)  {
3)      keys_fd = open(KEYDevice, O_RDONLY);
4)      mem_fd = open("/dev/mem", O_RDWR);
5)      cpld = (unsigned char*)mmap(NULL,(size_t)0x10,PROT_READ | PROT_WRITE |
    PROT_EXEC,MAP_SHARED,mem_fd,(off_t)(0x8000000));
6)      pthread_create(&th_time, NULL, time_counter, 0);
7)      pthread_create(&th_key, NULL, key_input, 0);
8)      while(1)
9)      {
10)         for(i=0; i<8; i++)
11)         {
12)             *(cpld+(0xe6<<1)) = addr[i];
13)             *(cpld+(0xe4<<1)) = tube[number];
14)             usleep(1000);
15)         }
16)     }
17)     pthread_join(th_time, &retval);
18)     pthread_join(th_key, &retval);
19)     munmap(cpld,0x10);
20)     close(mem_fd);
21)     close(keys_fd);
22)     return 0;
}
```

答：

第 5）行：内存映射操作，将数码管的内容映射到用户空间

第 12）行：设置数码管的位地址（即哪一个数码管）

第 13）行：设置数码管的段值（即显示什么内容，七段码或八段码）

第 19）行：解除内存映射

10. （5 分）以下为 CAN 总线双机通信中接收程序的主函数，请什么程序中第 10）、12）、15）、18）、21）行的含义。

```
21)   int main(int argc, char *argv[])
22)   {
23)     int s, nbytes, nbytes_send;
24)     struct sockaddr_can addr;
25)     struct ifreq ifr;
26)     struct can_frame frame_rev;
27)     struct can_frame frame_send;
28)     struct can_filter rfilter[1];
```

```
29)     int len = sizeof(addr);
30)     s = socket(PF_CAN, SOCK_RAW, CAN_RAW);
31)     strcpy(ifr.ifr_name, "can0" );
32)     ioctl(s, SIOCGIFINDEX, &ifr);
33)     addr.can_family = AF_CAN;
34)     addr.can_ifindex = ifr.ifr_ifindex;
35)     bind(s, (struct sockaddr *)&addr, sizeof(addr));
36)     rfilter[0].can_id = 0x00;
37)     rfilter[0].can_mask = CAN_SFF_MASK;
38)     setsockopt(s, SOL_CAN_RAW, CAN_RAW_FILTER, &rfilter, sizeof(rfilter));
39)   while(1)
40)     {
41)         nbytes = read(s, &frame_rev, sizeof(frame_rev));
42)         if(nbytes > 0)
43)         {
44)             printf("ID=0x%X                                                    DLC=%d
      data[0]=%X\n",frame_rev.can_id,frame_rev.can_dlc,frame_rev.data[0]);
45)         }
46)     }
47)     close(s);                                                    /
48)     return 0;
49)   }
```

答：第 10）创建套接字
第 12）行指定 can0 设备
第 15）行将套接字与 can0 绑定
第 18）行设置过滤规则，只接收表示符等于 0x00 的报文
第 21）行接收报文