**微型计算机技术学习札记**

**Learning notes of microcomputer technology**

|  |  |
| --- | --- |
| 学 号 | 1727405175 |
| 姓 名 | 魏军杰 |
| 任课教师 | 王宜怀 |
| 实验教师 |  |
| 教 材 | 王宜怀、刘一清、冯德旺. 微型计算机原理及应用—基于Arm微处理器，人民邮电出版社，2020年1月 |
| 时 间 | 2019-2020第2学期（2020.2-2020.7） |
| 单 位 | 苏州大学计算机科学与技术学院 |

说明

（1）本学习札记为苏州大学计算机科学与技术学院微型计算机技术课程教学过程使用，包括课前预习、课堂笔记整理、课后学习等记录，还包括习题作业、实验报告等内容。

（2）严格遵守本模板格式，不能新增排版格式

（3）本文档作为平时及实验成绩计入期末总评

目 录

[第1部分 学习备忘录 1](#_Toc33125823)

[2020年2月 1](#_Toc33125824)

[第1周（2020年2月24日-3月1日） 1](#_Toc33125825)

[2020年3月 1](#_Toc33125826)

[第2周（2020年3月2日-3月8日） 1](#_Toc33125827)

[第3周（2020年3月9日-3月15日） 1](#_Toc33125828)

[2020年4月 2](#_Toc33125829)

[2020年5月 2](#_Toc33125830)

[2020年6月 2](#_Toc33125831)

[2020年7月 2](#_Toc33125832)

[第2部分 学习记录整理归纳 3](#_Toc33125833)

[课本第1章 微型计算机基本结构及信息表示 3](#_Toc33125834)

[课本第2章 微型计算机的硬件系统 3](#_Toc33125835)

[课本第3章 指令系统 3](#_Toc33125836)

[课本第4章 汇编语言框架 3](#_Toc33125837)

[课本第5章 基于构件的汇编程序设计方法 3](#_Toc33125838)

[课本第6章 存储器 3](#_Toc33125839)

[课本第7章 串行通信接口 3](#_Toc33125840)

[课本第8章 中断系统及定时器 4](#_Toc33125841)

[课本第9章 模数转换与数模转换 4](#_Toc33125842)

[课本第10章 直接存储器存取DMA 4](#_Toc33125843)

[课本第11章 外接组件综合实践 4](#_Toc33125844)

[课本第12章 通用计算机的基本结构及启动过程 4](#_Toc33125845)

[课本第13章 微型计算机的发展方向 4](#_Toc33125846)

[第3部分 习题作业 5](#_Toc33125847)

[课本第1章 微型计算机基本结构及信息表示 5](#_Toc33125848)

[课本第2章 微型计算机的硬件系统 5](#_Toc33125849)

[课本第3章 指令系统 6](#_Toc33125850)

[课本第4章 汇编语言框架 6](#_Toc33125851)

[课本第5章 基于构件的汇编程序设计方法 6](#_Toc33125852)

[课本第6章 存储器 7](#_Toc33125853)

[课本第7章 串行通信接口 7](#_Toc33125854)

[课本第8章 中断系统及定时器 7](#_Toc33125855)

[课本第9章 模数转换与数模转换 8](#_Toc33125856)

[课本第10章 直接存储器存取DMA 8](#_Toc33125857)

[课本第11章 外接组件综合实践 8](#_Toc33125858)

[课本第12章 通用计算机的基本结构及启动过程 9](#_Toc33125859)

[课本第13章 微型计算机的发展方向 9](#_Toc33125860)

[第4部分 实验报告 10](#_Toc33125861)

[实验一：理解汇编程序框架及运行 10](#_Toc33125862)

[实验二：基于构件方法的汇编程序设计 10](#_Toc33125863)

[实验三：存储器实验 10](#_Toc33125864)

[实验四：基于串行通信构件的汇编程序设计 10](#_Toc33125865)

[实验五：理解中断与定时器 10](#_Toc33125866)

[实验六：ADC-DAC模块实验 10](#_Toc33125867)

[实验七：通过DMA实现内存间数据的搬运 10](#_Toc33125868)

[实验八 综合实验 11](#_Toc33125869)

[第5部分 杂记 12](#_Toc33125870)

# 第1部分 学习备忘录

## 2020年2月

### 第1周（2020年2月24日-3月1日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
| 2020年2月27日 | 微型计算机基本结构及信息表示 | 2h |
|  |  |  |

## 2020年3月

### 第2周（2020年3月2日-3月8日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
| 2020年3月4日（星期三） | 1.3数制以及数制转换  1.4 计算机信息表示方式  1.5 字符编码 | 2h |
| 2020年3月5日（星期四） | 复习上次课的内容 | 45min |

### 第3周（2020年3月9日-3月15日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
| 2020年3月11日（星期三） | 微型计算机的硬件系统2.1-2.3 | 2h |
| 2020年3月12日（星期四） | 复习周三内容，做笔记 | 1h |

### 第4周（2020年3月16日-3月22日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
| 2020年3月18日（星期三） | 2.4 - 2.5  Arm-Cortex-M4内核的微型计算机芯片实例  由STM32L431构建的通用嵌入式计算机 | 2h |
| 2020年3月18日（星期三） | 复习上午内容，整理笔记做练习题 | 2h |

### 第5周（2020年3月23日-3月29日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
| 2020年3月24日（星期二） | 预习ch3 指令系统 | 1h |
| 2020年3月25日（星期三） | 下午整理笔记，完成练习 | 2.5h |

### 第6周（2020年3月30日-4月5日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
| 2020年4月1日（星期三） | 下午整理笔记，完成作业1,2题 | 2.5h |

## 2020年4月

### 第7周（2020年4月6日-4月12日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
| 2020年4月8日（星期三） | 整理笔记，完成第四章作业 | 3h |
| 2020年4月12日（星期日） | 预习第五章 | 1h |

### 第8周（2020年4月13日-4月19日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
| 2020年4月16日（星期四） | 整理札记，完成作业1-5 | 3h |
| 2020年4月17日（星期五） | 阅读软件重用NATO标准，完成作业第6题 | 3h |

### 第9周（2020年4月20日-4月26日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
| 2020年4月23日（星期四） | 整理期中报告 | 3h |
| 2020年4月24日（星期五） | 整理期中报告 | 4h |

### 第10周（2020年4月27日-5月3日）

|  |  |  |
| --- | --- | --- |
| 日期（星期） | 内容 | 用时 |
| 2020年4月29日（星期三） | 整理第六章札记 | 2h |
| 2020年4月30日（星期四） | 完善期中报告，写第六章习题，整理第七章札记 | 3h |

## 2020年5月

## 2020年6月

## 2020年7月

# 第2部分 学习记录整理归纳

## 课本第1章 微型计算机基本结构及信息表示

学习内容：

1. 计算机种类：微型计算机（个人电脑，手机，汽车控制，空调控制等）、超级计算机
2. 冯诺依曼结构：

由控制器，运算器，存储器，输入设备，输出设备组成

信息由二进制表示

指令和数据同时放在存储器中

1. 三总线：地址总线，数据总线，控制总线

地址总线：

决定了能访问的地址空间大小

数据总线：

与存储单元进行信息交换，总线宽度等于计算机的字长

控制总线：

传送控制信号和时序信号来控制CPU和存储单元的信息交换

1. 执行指令的过程：取指令，译码，取操作数，执行指令，写回数据，执行下一条指令
2. 数制转换:二进制，十进制，十六进制等之间的转换
3. 计算机中信息的表示：
   1. 位，字节，字长
   2. 机器数，真值
   3. 补码表示方法
      1. 求补码方法：正数等于原码，负数除符号位取反+1
      2. 理解：时钟例子
      3. 一般求法：正数等于原码，负数等于2^n-原码
      4. 取值范围：8位补码（-128 ~ 127）
   4. 浮点数的表示方法（32位为例子）

·符号位 1位

·阶码 8位

·尾数 23位

1. 字符编码
   1. ASCII码

·常用ASCII码

30H-39H 0-9

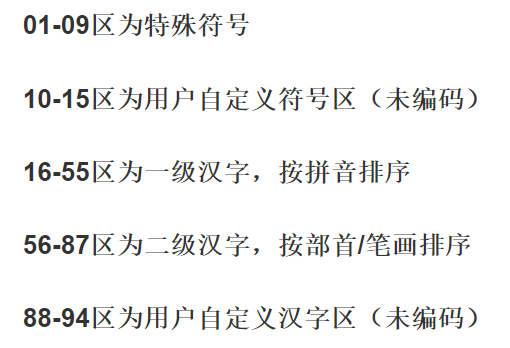
41H-5AH A-Z

61H-7AH a-z

* 1. 中文编码GBK

·国标字库：一级字库（3755个），二级字库（3088个）

* 1. 区码和位码
     1. 由四位十进制数表示，前两位区码，后两位位码
     2. 编码规则



* + 1. 所有汉字按照拼音排序每个数表示一个字

## 课本第2章 微型计算机的硬件系统

1. 微型计算机的硬件共性结构及基本性能指标
2. 字长 决定了计算机内存访问的空间
3. 主频
   1. 指令执行是以时钟周期执行的，主频越高，时钟周期越短，计算机运行越快
   2. 计算机频率的产生：**晶体振荡器**（纯净二氧化硅晶体的压电效应）
4. 存储容量
5. 外设扩展能力
6. 软件配置情况
7. 存储器
   1. RAM

RAM（Random Access Memory ）随机存取存储器，主要作用是存储代码和数据供CPU 需要时调用，跟计算机里面的内存差不多,主要是用来存放程序运行中的过程数据,掉电后就会丢失之前的数据,所以程序在上电时需要进行初始化,否则上电后的数据是一个随机数,可能导致程序奔溃

* 1. ROM

ROM掉电后数据不会丢失,但在程序运行过程中其数据不会改变.

* 1. Flash

Flash即Flash Memory，全名叫Flash EEPROM Memory，又名闪存，是一种长寿命的非易失性（在断电情况下仍能保持所存储的数据信息）的存储器，数据删除不是以单个的字节为单位而是以固定的区块为单位，区块大小一般为256KB到20MB，Flash存放代码和常数。

1. CPU内部寄存器与存储器映像

**寄存器分类**

1. 通用寄存器 作用：存放运算数据或者计数数据等CPU运算时需要的数据，用于数据缓冲
2. 栈指针 作用：访问RAM中的栈区
3. 连接寄存器 作用：用于保存函数或子程序调用的返回值
4. 程序计数寄存器 作用：放下一条指令所在单元的地址
5. 程序状态字寄存器 作用：存放状态字信息
6. 特殊功能寄存器
   * 1. 中断屏蔽寄存器 ：使用特殊指令可访问寄存器
     2. 错误屏蔽寄存器 ：能屏蔽硬件错误异常
     3. 基本优先级寄存器 ： 屏蔽特定优先级的中断
     4. 控制寄存器 ： 控制和确定处理器的操作模式和当前执行任务的特性

**Arm Cortex-M4存储器映像**

存储器映像的含义：

把4GB**空间**当做**存储器**来看，分成若干个空间，以安排一些实际的物理资源



**10位： 2^10 = 1K**

**20位： 2^20 = 1M**

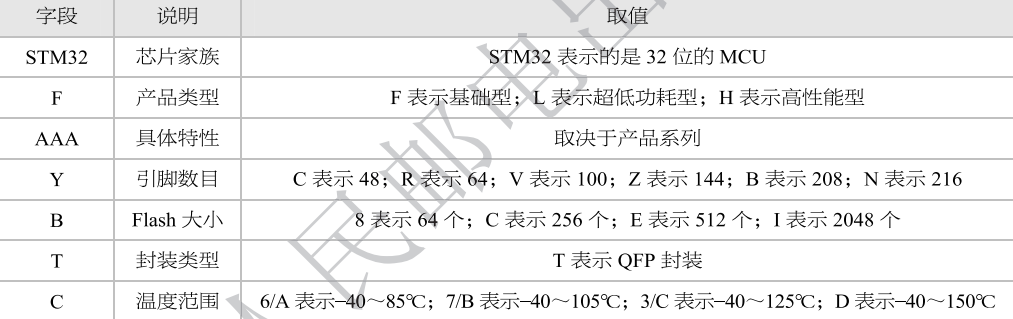
**30位： 2^30 = 1G**

**40位： 2^40 = 1T**

**Arm-Cortex-M4内核的微型计算机芯片实例**

**STM32L4系列MCU**

（1）命名格式：



比如：STM32L431RCT6

STM32：32位的MCU

L ：超低功耗型

431 ：产品系列

R ：64个引脚

C ：Flash大小：256个

T ：封装类型为QFP封装

6 ：温度范围为 -40~85度

**集成块的封装类型**：

BGA 封装 (ball grid array)

BQFP 封装 (quad flat package with bumper)

PGA 封装 (butt joint pin grid array)

C－(ceramic) 封装

Cerdip 封装

Cerquad 封装

DFP(dual flat package)

DIL(dual in-line)

DICP(dual tape carrier package)

DIP(dual tape carrier package)

1. 中断

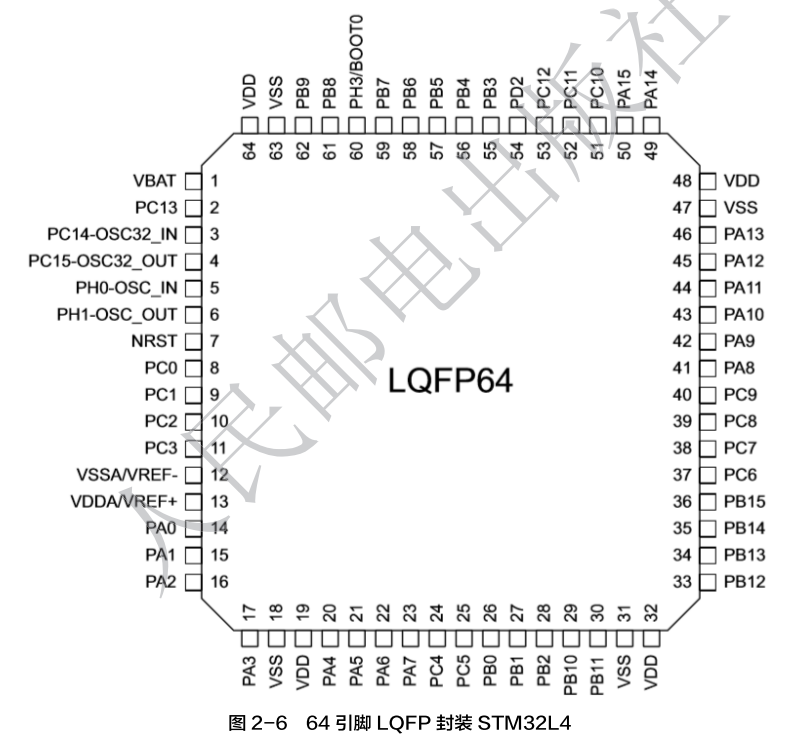
中断向量表的查看

中断分类：内核中断和外部中断

由中断向量号得到相应的中断

1. 引脚功能

一个例子：



分类：

1. 硬件最小系统引脚
   1. 电源输入
   2. 复位
   3. 晶振
   4. SWD接口
2. 对外提供服务的引脚
3. 硬件最小系统原理图
   1. 电源和滤波电路：

多处供电能保证电流平衡

电容滤波原理：通交流隔直流

* 1. 复位电路
     1. 冷复位（无电到带电），热复位（带电到无电）
     2. 外部复位，内部复位
     3. 异步复位（立即有效），同步复位（下周期有效）
  2. 晶振电路：提供工作时钟，原理：晶振
  3. SWD接口电路：实现程序下载和调试功能

**通用嵌入式计算机**

概念：

从硬件方面：把MCU硬件最小系统及面向具体应用的共性电路封装成一个整体，按照硬件构件要求进行原理图绘制、文档撰写、以及硬件测试用例设计。

从软件方面：把嵌入式软件分为基本输入输出系统（BIOS）与User程序两部分，BIOS程序先于User程序固化与MCU的非易失存储器上，BIOS先运行，然后User程序再运行。BIOS提供工作时钟以及面向知识的底层驱动构件，并为User提供函数调用接口。

特点：

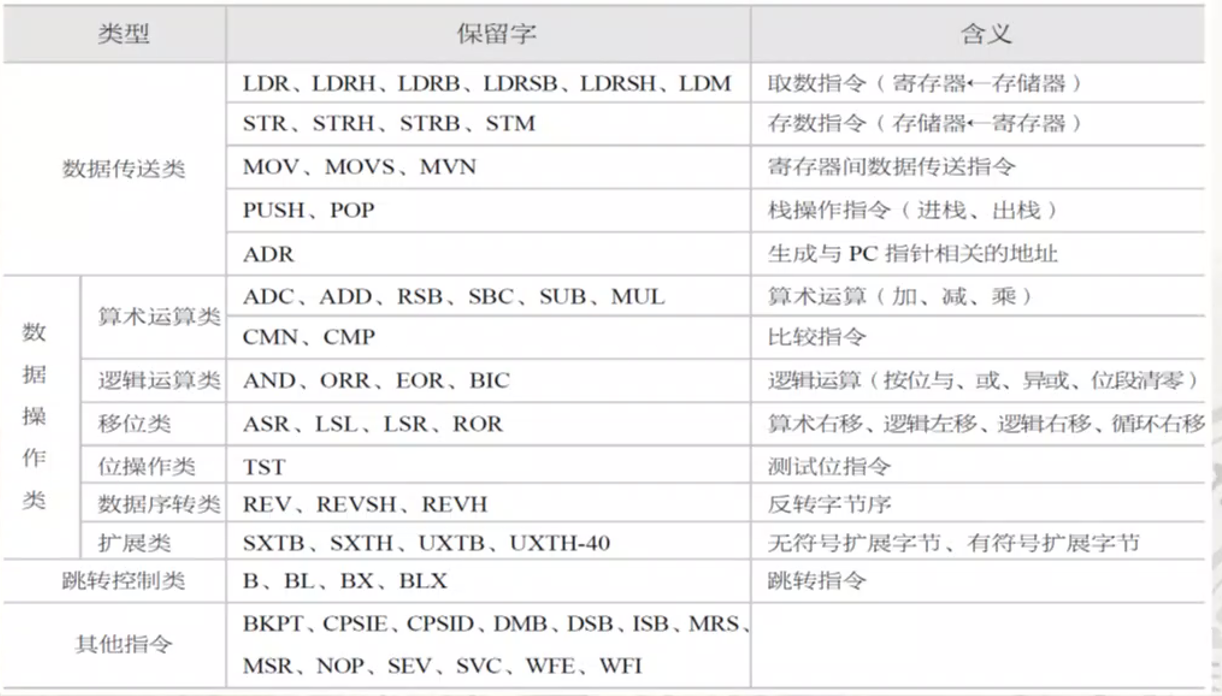
1. 移植性强
2. 稳定性强，交互性差
3. 具备统一接口
4. 实时性比较强

## 课本第3章 指令系统

指令保留字与寻址方式

记忆指令的方法：

指令保留字简表（记忆）



LDR：load 取存储器的数据到寄存器（H：半个 B：一个Byte8位）

STR：store 把寄存器内容存入存储器中

MOV：CPU内部寄存器移动

PUSH，POP：进出栈操作

ADC，ADD | SBB，SUB | MUL：算术运算

CMP：比较

CMN：负数比较

AND，OR，EOR，BIC：与，或，异或，位清零

ASR，LSR，ASL，LSL，ROR：算术右移，逻辑右移，算术左移，逻辑左移，循环右移

TST，CMP：只影响标志位，前者是按位与，后者算术减法，结果影响的都是标志Z

REV：一个寄存器中的字节逆向放入寄存器中

SXTB，SXTH：提取0:7位 | 0:15位 放入寄存器中并符号扩展

B，BL，BX，BLX：无条件跳转 | 将下一条指令地址放入寄存器后跳转 | 带状态切换的跳转（ARM或者Thumb指令） | 结合BL与BX的带链接与状态切换的跳转

寻址方式

（1）立即数寻址

MOV R0, #0x1F （带#为立即数）

1. 寄存器直接寻址

MOV R2, R1

1. 直接地址寻址

LDR R1,label ;从label读取4字节到R1

1. 寄存器间接寻址

LDR R3,[R4] ; 把R4内地址的东西取到R3中

寻找10条指令的机器码（编译后寻找）

编译后的文件如下，每个指令的机器码用颜色标注：

MOV R0,#0xFE

800d840: f04f 00fe mov.w r0, #254 ; 0xfe

ADD R0,#0x33

800d844: f100 0033 add.w r0, r0, #51 ; 0x33

AND R1,R0

800d848: ea01 0100 **and**.w r1, r1, r0

EOR R1,R0

800d84c: ea81 0100 eor.w r1, r1, r0

BX R2

800d850: 4710 bx r2

SUB R1,R0,#1

800d852: f1a0 0101 sub.w r1, r0, #1

CMP R0,0

800d856: 2800 cmp r0, #0

ASR R1,R0

800d858: fa41 f100 asr.w r1, r1, r0

REV R1,R0

800d85c: ba01 rev r1, r0

REVSH R1,R0

800d85e: bac1 revsh r1, r0

NZCV标志：

N：运算结果为正或0，N=0

C：进位标志，有进位，C=1

Z：零标志 运算结果为0，Z=1

V：溢出标志，适用于加减法运算结果，如果符号位溢出，V=1（比如两个正数相加等于负的情况，V=1）

与 或 异或 实例：

1. 与：

（1）判断一个数的奇偶：

x&1，结果为1为奇数，结果为0是偶数

1. 判断一个数是否为2的幂：

x & (x-1) 结果为0就是2的幂

1. 或

给某些特定位赋值为1：

例如给eax的低4位赋值为1：eax | 0x1111

1. 异或

（1）密文加密解密:

假设密文为：10101110，密钥为：10101

加密后密文为： 10101110

^ 10101

= 10111011

解密：

10111011

^ 10101

=10101110

（2）不用额外变量实现两个数的交换：

x y

x^y y

x^y x^y^y

x^y^x x

y x

其他的转移指令：

jmp 无条件跳转

无符号

Je 等于跳转

Jne 不等于跳转

Ja = jnbe 大于跳转 = JNBE（不小于或等于跳转）

Jb= jnae 小于跳转 = JNAE（不大于或等于跳转）

Jna 不大于跳转 = JBE（小于或等于跳转）

Jnb 不小于跳转 = JAE（大于或等于跳转）

Js 为负跳转

Jz 为0跳转

Jc 进位跳转

Jo 溢出跳转

Jp 奇偶位置位 跳转

有符号

Jg 有符号大于 跳转 = JNLE

Jge 有符号大于等于 跳转 = JNL

Jl 有符号小于 跳转 = JNGE

Jle 有符号小于等于 跳转 = JNG

·常用伪指令：

1. EQU

一般格式为：  标号：         EQU          操作数

指令功能为将操作数赋予标号，两边的值完全相等。使用EQU伪指令给一个标号赋值后，此标号在整个源文件中值固定。

Label:       EQU          1000H                ;将标号Label赋值为1000H

1. ORG

一般格式为：         ORG  1234H（绝对地址或标号）

1234H决定此语句后第一条指令（或数据）的地址。该段源程序或数据被连续存放在此后的地址内，直到下一条ORG指令为止。

ORG          8000H                ；此后目标代码存储在存储器中以0x8000h开始的地址空间中。

1. DB，DW

一般格式为：         标号：     DB         字节常数或字符或表达式

此伪指令的功能是把字节常数或字节串存放至内存连续的地址空间中。

DATA1：  DB    43H,09H,08H

DATA2：  DB    07H

伪指令DB指定了43H，09H，08H      顺序存放在8000H开始的存储单元中，DATA2中的07H紧挨着DATA1的地址空间存放，即07H存放在8003H单元中。

DW（Define Word）指令定义与DB类似，区别在于DW定义一个字，DB定义一个字节。

1. END

一般格式为： 标号： END 地址或标号（可以忽略）

此伪指令用于指示汇编语言程序段结束。

·GNU汇编常用伪指令：

GNU汇编伪指令都以 . 开头

1. 系统预定义的段

.data // 可读可写的已初始化的数据段

.bss // 可读可写未初始化的数据段

.text // 只读代码段

1. 常量定义

格式： .equ 常量名, 表达式

如：.equ var1, 0xFFFE

1. 变量定义

格式：

变量名:

.word 0x12341234

数据类型：

.word 4字节

.hword 2字节

.byte 1字节

.ascii 字符串

.asciz 以“\0”结尾的字符串

1. 条件伪指令

格式：

.ifdef 表达式

代码段1

.else

代码段2

.endif

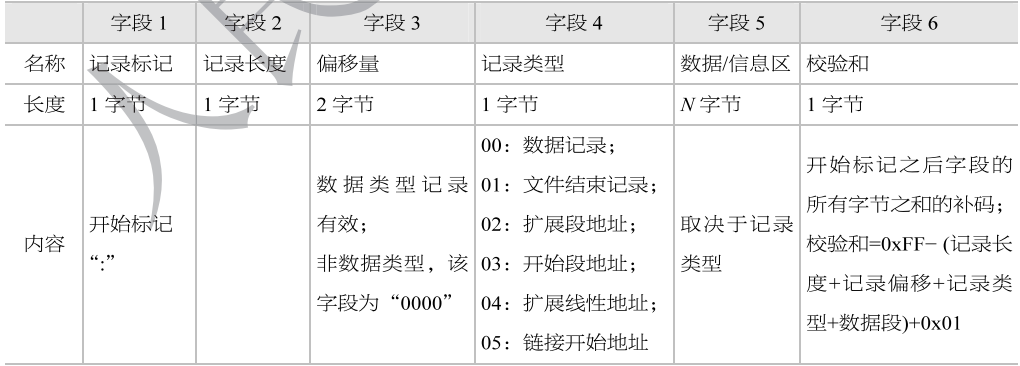
1. Include伪指令

格式：

.include “filename”

## 课本第4章 汇编语言框架

·十六进制文件行语义：



需要记忆：

记录类型：

00：数据记录（大部分）

01：文件结束记录

02：扩展段地址

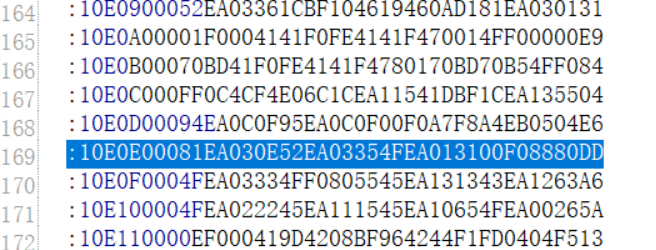
03：开始段地址

04：扩展线性地址

05：链接开始地址

校验和部分，每8位相加，溢出不算，结果取补码

取其中一行进行分析：



: 10 E0E0 00 81EA030E 52EA0335 4FEA0131 00F08880 DD

各字段分析：

10 ： 记录长度为0x10 = 16个字节

E0E0 ：偏移量为E0E0

00：记录类型为 ：数据记录

81EA030E 52EA0335 4FEA0131 00F08880：4个字数据

DD：校验和为DD，运算过程如下：

FF-((10+E0+E0+00+81+EA+03+0E+52+EA+03+35+4F+EA+01+31+00+F0+88+80) &FF)+1=DD

TTL电平标准：

高电平：大于电源电压的百分之八十

低电平：小于电源电平的百分之二十

通讯时，传输一般在5m以内

补充：

CMOS电平：

接近电源电压表示1

电压接近0V表示0

CMOS电路是电压控制器件，输入电阻极大，对干扰信号十分敏感，优点是噪声容限宽，静态功率小

232电平：

+12V表示0

-12V表示1

传输30m左右

485电平：

差分电路，两根导线传一个电平，根据差值确定电平，抗干扰能力强

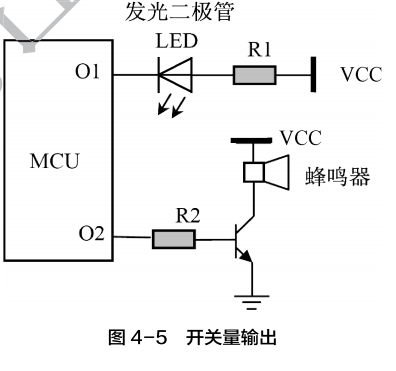
传输1km左右

不同电平标准**编程上是一样的**

三极管作用：

1. 电流放大，把微弱电流变成一定强度的信号，把电源能量转换为信号能量，有放大系数
2. 开关作用

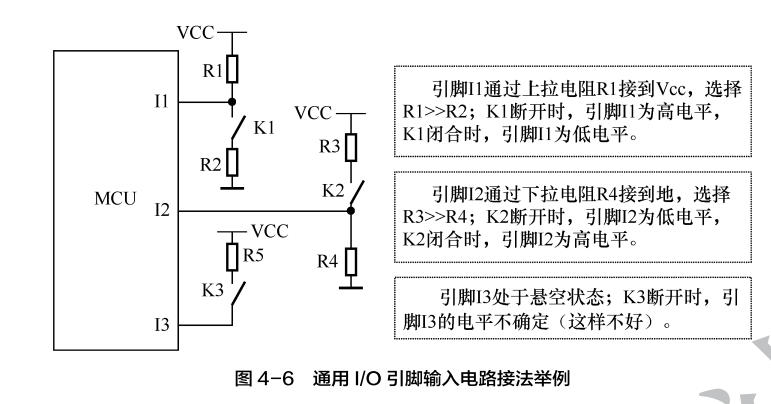
输出引脚接法：



O1处，高电平不亮，低电平亮

O2处，高电平三极管导通，蜂鸣器响，低电平蜂鸣器不响

带上拉电阻与下拉电阻的连接：



上拉电阻：

引脚与电源之间有R1，引脚与接地之间有开关和R2，开关**断开**时引脚**高**电平，**闭合低**电平，R1>>R2, R1为上拉电阻，该引脚平时为高电平

下拉电阻：

引脚与电源之间有开关与R1，引脚与接地之间有R2，开关**断开**时引脚**低**电平，**闭合高**电平，R1>>R2, R2为下拉电阻，该引脚平时为低电平

GPIO

GPIO是 通用输入输出端口，是STM32芯片的可控引脚，所有gpio引脚都有基本输入输出功能

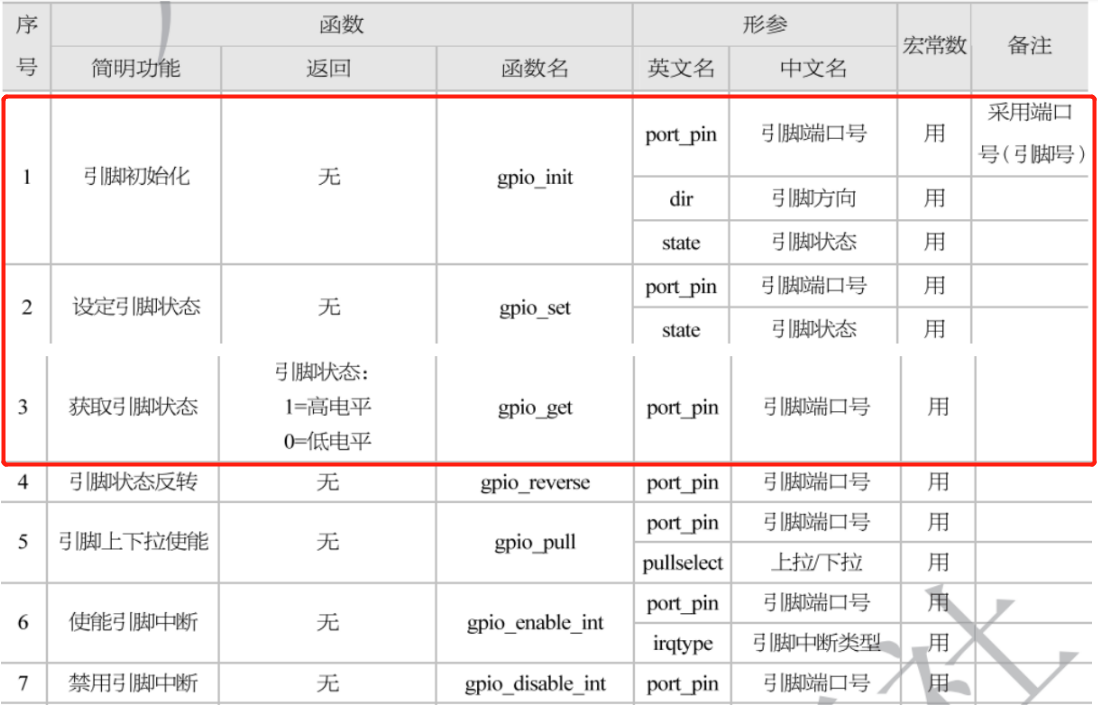
具体功能：

输入：能知道外部是高电平或者低电平

输出：能输出高电平或者低电平

GPIO的函数：

前三个为主要函数



调用gpio函数前需要把相应参数传入寄存器中

使用方法举例：

1. 在user.inc文件中，给灯取名，并确定引脚，给亮暗进行宏定义
2. 在main.s文件中，初始化小灯：通过寄存器r0,r1,r2传递参数，调用gpio\_init函数
3. 在main.s文件中，点亮小灯：通过寄存器r0，r1传递参数，调用gpio\_set

## 课本第5章 基于构件的汇编程序设计方法

**软件构件的意义在于可复用性，可重用性，提高开发人员的开发效率，通过使用具有鲁棒性的软件构件，开发人员既能够保证质量又能保证开发速度**

**构件设计的基本思想：**

在设计底层构件时，最关键的工作是要对构件的共性和个性进行分析，设计出合理的、必要的对外接口函数及其形参。尽量做到：当一个底层构件应用到不同系统中时，仅需修改构件的头文件，对于构件的源程序文件则不必修改或改动很小

**构件设计的基本原则：** 好的底层驱动构件需要具备：封装性，描述性，可移植性，可复用性 的基本要求

为了达到这些要求，需要满足以下基本原则：

1. 层次化原则：

针对应用场景和服务对象，分层组织构件

在构件的层次模型中，上层构件可以调用下层构件提供的服务，同一层次的构件不存在相互依赖关系，不能相互调用。

1. 易用性原则：

底层驱动构件需要做到：函数名简洁且达意；接口参数清晰，范围明确；使用说明语言精炼规范，避免二义性。若一个函数功能复杂，则需要拆分为多个函数

1. 鲁棒性原则：

遵循鲁棒性原则设计底层驱动构件需要做到：在明确函数输入输出的取值范围、提供清晰接口描述的同时，在函数实现的内部要有对输入参数的检测，对超出合法范围的输入参数进行必要的处理；使用分支判断时，确保对分支条件判断的完整性，对缺省分支进行处理；不能忽略warning。

1. 内存可靠使用原则：

对内存的可靠使用是保证系统安全、稳定运行的一个重要的考虑因素。遵循内存可靠使用原则设计底层驱动构件需要做到：

① 优先使用静态分配内存。相比于人工参与的动态分配内存，静态分配内存由编译器维护，更为可靠。

② 谨慎地使用变量。可以直接读写硬件寄存器时，不使用变量替代；避免使用变量暂存简单计算所产生的中间结果；使用变量暂存数据将会影响到数据的时效性。

③ 检测空指针。定义指针变量时必须初始化，防止产生“野指针”。

④ 检测缓冲区溢出，并为内存中的缓冲区预留不小于 20%的冗余。使用缓冲区时，对填充数据长度进行检测，不允许向缓冲区中填充超出容量的数据。

⑤ 对内存的使用情况进行评估。

**构件分类：**  
（1）基础构件：

基础构件是根据MCU内部功能模块的基本知识要素，针对MCU引脚功能或MCU内部功能，利用MCU内部寄存器，所制作的直接干预硬件的构件。  
  
（2）应用构件：

应用构件是使用基础构件并面向对象编程的构件。例如 printf 构件，它调用串口构件完成输出显示功能。printf 函数调用的一般形式为：printf(“格式控制字符串”, 输出表列)，本书使用的 printf函数可通过 uart 串口向外传输数据。

（3）软件构件：

软件构件是一个面向对象的具有规范接口和确定的上下文依赖的组装单元，它能够被独立使用或被其他构件调用。

**程序流程控制：**

程序流程控制主要有顺序结构、分支结构、循环结构。  
 （1）顺序结构：顺序结构程序的执行方式是“从头到尾”，逐条执行指令语句，直到程序结束，这是程序的最基本形式。  
 （2）分支结构：分支结构程序是利用条件转移指令，使程序执行到某一指令后，根据条件是否满足，来改变程序执行的次序，这类程序使计算机有了判断作用。分支结构分为单分支结构和多分支结构。  
 （3）循环结构：循环结构程序是强制CPU重复执行某一指令系列（程序段）的一种程序结构形式，凡是要重复执行的程序段都可以按循环结构设计。循环程序一般由四部分组成：初始化、循环体、循环控制和循环结束处理。

**注意：在编写软件构件的时候，不能出现调用例如输入输出的具体的底层硬件，这样不能保证构件的可移植性，可复用性**

## 课本第6章 存储器

存储器的分类：

1. 按存储介质分类：半导体存储器、磁存储器、光存储器、Flash存储器
2. 按功能分类：主存储器（内存）、辅助存储器（外存）、高速缓冲存储器（Cache）、控制存储器
3. 按存取方式分类：随机访问存储器（RAM）、顺序访问存储器（SAM）

RAM与ROM：

1. SRAM（Static Random Access Memory）：静态随机存储器，断电即失，性能稳定不需要外加额外电路，工作速度快；缺点是功耗较大，因为它由6个MOS管组成，集成度低
2. DRAM（Dynamic Random Access Memory）：动态随机存储器，读取是破坏性的，需要周期性的重写；优点是它只由一个MOS管与一个电容组成，功耗低，芯片集成度高
3. PROM，可编程只读存储器，由双极型电路和熔丝构成的基本电路，熔丝断和未断表示1和0，所以它只能一次性编程
4. EPROM，可擦除可编程只读存储器，可以进行任意次的改写
5. EEPROM：带电可编程可擦除存储器，可在专用设备上擦除已有信息重新编程

CMOS电路是电压控制器件，而TTL电路是电流控制器件

高速缓存

（1）作用：

Cache是高速缓冲存储器，是位于CPU和主存之间，规模较小，但速度很高的存储器，通常由SRAM组成，直接刻在CPU中。它是位于CPU与内存间的一种容量较小但速度很高的存储器。Cache可以保存CPU刚用过或循环使用的一部分数据，如果CPU需要再次使用该部分数据时可从Cache中直接调用，这样就避免了重复存取数据，减少了CPU的等待时间，提高了系统的效率。

（2）结构以及工作原理：

主存块和 cache 行之间有以下三种映射方式：

1. 直接映射：每个主存块映射到 cache 的固定行中。
2. 全相连映射：每个主存块映射到 cache 的任意行中。
3. 组相连映射：每个主存块映射到 cache 的固定组的任意行中。

（3）工作流程：

在 CPU 执行程序过程中，需要从主存取指令或写数据时，先检查 cache 中有没有要访问的信息，若有，就直接在 cache 中读写，而不用访问主存储器。若没有，再从主存中把当前访问信息所在的一个一个主存块复制到 cache 中。

Flash存储器

1. Flash在线编程

主要包括擦除、写入操作

1. Flash的特点

电可擦除可编程，不会断电丢失数据，功耗低，成本较低，存储密度高等

1. Flash驱动构件

Flash具有初始化、擦除、写入、读取、保护、解除保护、判空等基本操作，以构件的思想包装成多个函数，他们定义的位置在flash.c文件中

## 课本第7章 串行通信接口

串行通信接口简称串口、UART、SCI，USB未普及之前，串口是PC机必备的通信接口之一。

在硬件上一般只需要三根线：发送线，接收线，地线。

串行通信的特点：数据以字节为单位，按位的顺序从一条传输线上发送出去。

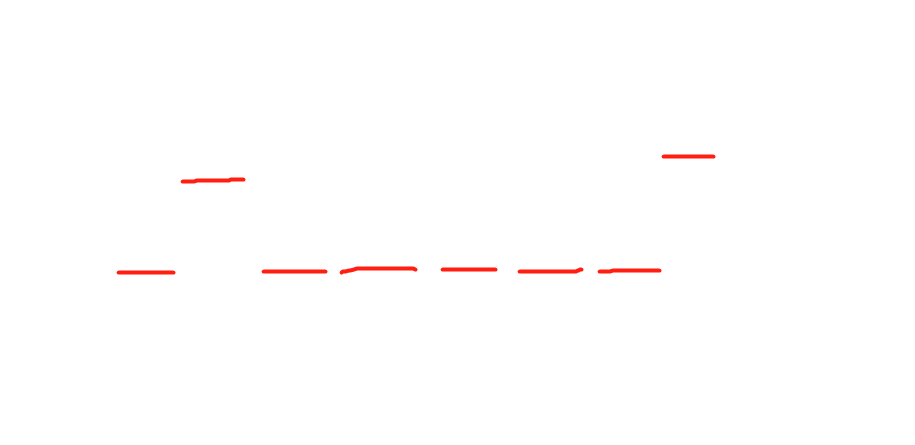
串行通信分为异步通信和同步通信。

波特率：每秒传送的位数叫波特率，单位bit/s

异步串行通信的格式：

例：字符“A”的时序图：

“A”的ascii码为41H，二进制表示为：0100 0001，即下图：



串行通信传输方式术语：

全双工（可以同时接受和发送数据）、半双工（可以接受和发送数据，但是在同一时间里只能进行一项）、单工（只能接受或只能发送数据）

RS232与RS485总线标准：同第四章232电平与485电平

TTL电平到RS232电平转换电路：

在发送端需要用驱动电路将TTL电平转换成RS232电平，接收端需要用接收电路将RS232电平转换为TTL电平，电平转换器目前广泛使用MAX232芯片

UART驱动构件使用方法：

1. include头文件“uart.inc”
2. 对构件进行初始化，把串口号与波特率放在r0 r1寄存器中，调用uart\_init函数
3. 在isr.s中编写使能中断程序
4. 开放使能中断，在main.s中调用使能中断函数来打开中断
5. 调用具体方法，比如调用uart\_send\_string函数

**对于计算机语言的学习，需要在精通一门语言的基础上，快速掌握其他语言，用中学（照葫芦画瓢，按照好的模板编程），学中用。**

Exam7-1主要功能：接收一个字节，发送一个字节

Exam7-2主要功能：使用串口调试工具，实现对串口UARTA发送的数据进行封装成帧并接收

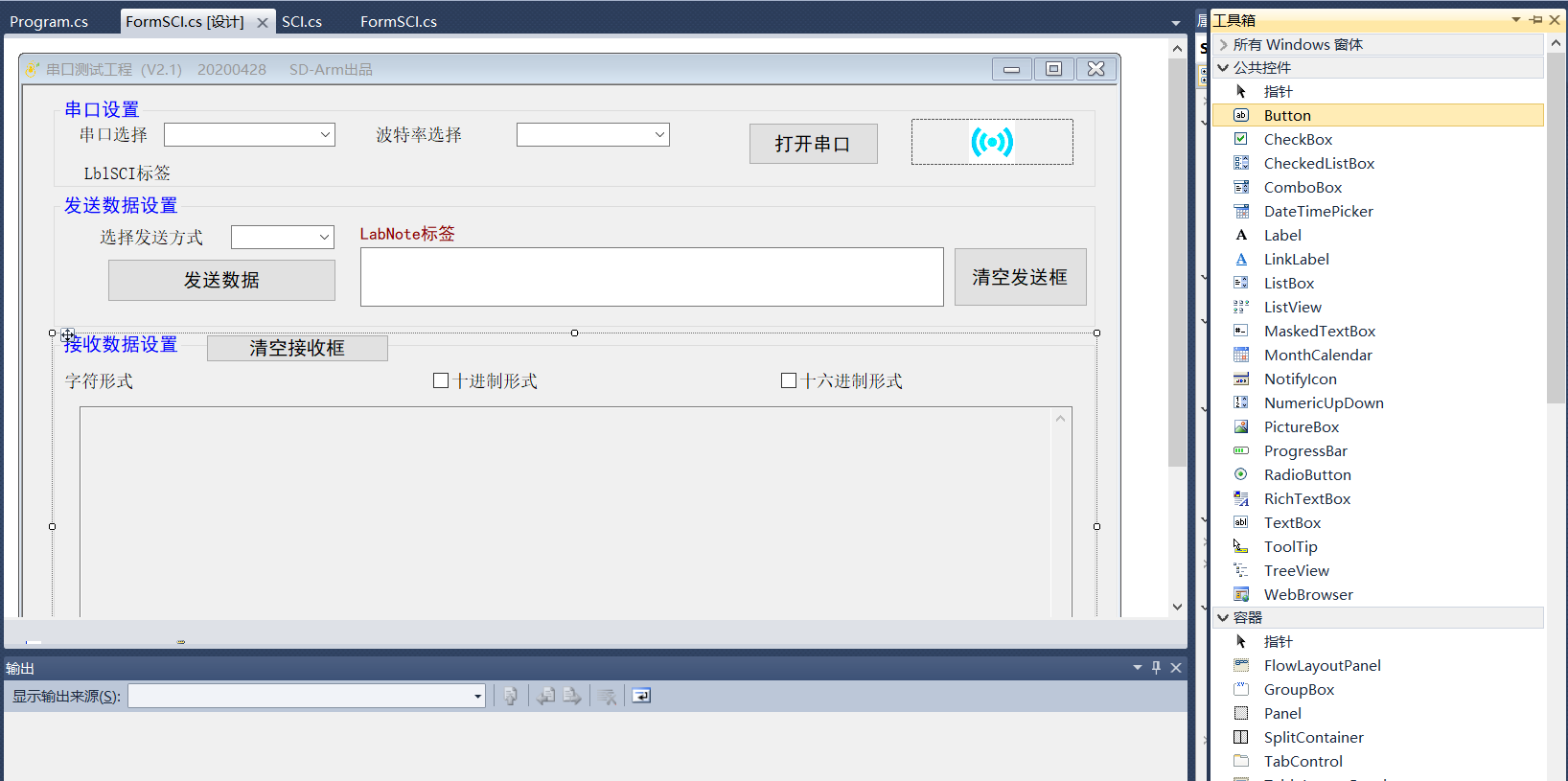
Exam7-3主要功能：每隔一秒接收一个字符串数据，并回发字符串，用户通过上位机使用串口调试工具实现通过串口UARTA向开发套件的串口模块发送一个字符串的数据

C#面向对象编程

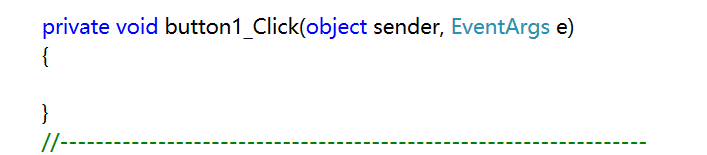
C#串口测试程序分析：

1.在Program.cs文件中通过Application.run(new FrmSCI());来初始化窗口类

2.设计窗口：通过拖动按钮，文本框等组件来设计窗口



每次拖动一个组件，就会自动生成一段代码。例如添加一个button，就会生成一段代码来，可以双击按钮来查看对应的代码，补充代码，添加需要的功能



3.主要功能具体分析：

（1）FrmSCI\_Load 加载窗体函数



功能是加载窗体，初始化时，显示“打开串口”按钮，设置波特率选择框，串口选择框为可用状态，向串口选择框添加搜索到的串口号，设置各项初始值

（2） 串口选择与波特率选择组合框

分别调用PublicVar中定义的两个全局变量g\_SCIComNum 与g\_SCIBaudRate，并显示出来

（3） BtnSCISwitch 打开串口按钮

调用SCI.cs中定义的SCIInit() 函数进行串口初始化，如果初始化成功则串口打开成功，此时设置串口选择框与波特率选择框为禁用状态，并给出提示文字，否则给出初始化失败的提示文字

如果打开串口成功，把按钮文字变成“关闭串口”，如果点关闭按钮，则调用SCI.cs中的SCIClose函数来关闭串口，关闭成功给出提示文字，失败则给出失败提示文字

（4）BtnSCISend 发送数据按钮

先判断串口有没有打开，如果没有打开则给出提示文字

判断发送数据是否为空，如果为空则给出提示文字

判断发送数据方式，把将要发送的数据进行编码，并存放在PublicVar中的g\_SendByteArray全局变量中

通过SCI.cs中的SCISendData函数来发送全局变量中的数据，返回结果

给出发送成功或者失败的提示信息

（5）btnClearRec 清空接收框按钮

把TbShowString, TbShowDec, TbShowHex这三个文本框的内容设置为空字符串

## 课本第8章 中断系统及定时器

## 课本第9章 模数转换与数模转换

## 课本第10章 直接存储器存取DMA

## 课本第11章 外接组件综合实践

## 课本第12章 通用计算机的基本结构及启动过程

## 课本第13章 微型计算机的发展方向

# 第3部分 习题作业

## 课本第1章 微型计算机基本结构及信息表示

1. 从性能、体积、主要用途三个角度，简要阐述超级计算机与微型计算机的不同点。

超级计算机：

性能：超高性能，超大容量

体积：超大体积

主要用途：科学与工程计算，天气预报，卫星图像处理，大数据处理等等需要大规模计算的领域

微型计算机：

性能：性能够支撑某具体的应用，不能用来进行大规模运算

体积：体积小，灵活性高

主要用途：个人计算机，手机，数字电视，汽车控制，空调控制等

1. 微型计算机出现以后，为何会分化为通用计算机与嵌入式计算机两个发展方向。

通用计算机主要用途是科学计算，数据分析，图像处理等，需要较高的性能，而嵌入式计算机是以测量，控制为目的，降低功耗成本，减小体积，改善开发的环境

1. 简要阐述冯·诺依曼结构的特点，并说明与哈佛结构的区别。

冯·诺依曼结构特点：

由控制器，运算器，存储器，输入设备，输出设备组成

信息由二进制表示

指令和数据同时放在存储器中

区别：

冯诺依曼结构是将指令和数据合并在一起的存储器结构，哈佛结构将指令和数据分开存放，位于不同的存储空间

1. 从功能与编程两个角度分别阐述CPU的基本知识要点。

功能：

从外部设备获取数据，进行加工处理，再把结果输出到CPU外部，CPU包括运算器、寄存器和控制器，运算器执行算术和逻辑运算，移位，地址转化等，寄存器用来暂存数据、指令、地址控制器负责产生指令所需的控制信号

编程：

CPU包含寄存器和可执行的指令系统，每一个寄存器有特定的功能，如ECX寄存器大多数用来计数，PC是程序计数器。PC指定了指令在存储器中的地址，CPU能执行的指令在设计阶段就已经确定，一个CPU能执行的所有指令构成了指令系统。

1. 简要阐述计算机中的三总线概念，给出向RAM 中地址为9856地址单元写入一个字节数213的过程描述。

三总线包括：地址总线，数据总线，控制总线

地址总线：

决定了能访问的地址空间大小

数据总线：

与存储单元进行信息交换，总线宽度等于计算机的字长

控制总线：

传送控制信号和时序信号来控制CPU和存储单元的信息交换

向RAM 中地址为9856地址单元写入一个字节数213的过程描述：

CPU通过地址总线定位9856地址单元，通过数据总线发出213这个数，然后通过控制总线将数据总线的数放在9856地址单元里

6．将下列各十进制数分别转为十六进制数、二进制数。

（1）256 （2）4098 （3）65532 （4）32769

1. 十六进制：100 二进制：100000000
2. 十六进制：1002 二进制：1000000000010
3. 十六进制：FFFC 二进制：1111111111111100
4. 十六进制：8001 二进制：1000000000000001

7．将下列十六进制数分别转为十进制数、二进制数。

（1）0x6A （2）0xFFFE （3）0xFCDE8C （4）0x86ECFD6A

1. 十进制：106 二进制：1101010
2. 十进制：65534 二进制：1111111111111110
3. 十进制：16572044 二进制：111111001101111010001100
4. 十进制：2263678314 二进制：10000110111011001111110101101010

8．将下列二进制数分别转为十六进制数、十进制数。

（1）0b1011101 （2）0b11011011 （3）0b101010011101 （4）0b1011001111001010011

1. 十六进制：5D 十进制：93
2. 十六进制：DB 十进制：219
3. 十六进制：A9D 十进制：2717
4. 十六进制：59E53 十进制：368211
5. 分别给出8字长、64位字长补码能表示的真值范围，并举例给出由真值求补码的方法，以及由补码求真值的方法。

8位字长补码能表示的真值范围：-128 ~ 127

64位字长补码能表示的真值范围：-2^63 ~ 2^63-1

原码求补码方法：

1. 如果符号位为0，补码等于原码，如00000001的补码为00000001
2. 如果符号位为1，补码数值位等于原码取反+1，如10000001的补码为11111111

补码求原码方法：

1. 如果符号位为0，原码等于补码，如00000011的原码为00000011
2. 如果符号位为1，原码数值位等于补码取反+1，如10000011的原码为11111100
3. 给出32位单精度浮点数的表示范围，说明缘由。

正浮点数范围：1.0\*2^-126 ~ (2-2^-23)\*2\*127

负浮点数范围：-(2-2^-23)\*2^127 ~ -1.0\*2\*-126

所以32位浮点数范围：-(2-2^-23)\*2^127 ~ (2-2^-23)\*2\*127

1. 在计算机编程中，如何正确区分一段文字中的汉字与英文字母？

根据字符编码规则和编码来区分，比如采用的是GBK编码，每个编码代表不同的字符，根据编码来区分汉字和英文字母

1. 为什么说程序执行过程中，CPU对内部寄存器的操作速度比对RAM中的变量操作速度快？

访问内部寄存器不需要经过三总线直接访问，对RAM变量进行操作需要通过三总线来操控

## 课本第2章 微型计算机的硬件系统

1. 所学芯片的内部微处理器有哪些寄存器？简述各寄存器的作用。

（1）通用寄存器 作用：存放运算数据或者计数数据等CPU运算时需要的数据，用于数据缓冲

（2）栈指针 作用：访问RAM中的栈区

（3）连接寄存器 作用：用于保存函数或子程序调用的返回值

（4）程序计数寄存器 作用：放下一条指令所在单元的地址

（5）程序状态字寄存器 作用：存放状态字信息

（6）浮点寄存器

（7）特殊功能寄存器

1)中断屏蔽寄存器 ：使用特殊指令可访问寄存器

2)错误屏蔽寄存器 ：能屏蔽硬件错误异常

3)基本优先级寄存器 ： 屏蔽特定优先级的中断

4)控制寄存器 ： 控制和确定处理器的操作模式和当前执行任务的特性

1. RAM存储区和Flash存储区的访问特点？简述某一芯片的RAM存储区和Flash存储区的大小及地址范围。

RAM断电即失去存储内容，随机存取，存取的速度与存储单元的位置无关，存取速度比非常快，可通过单个指令读或写

Flash断电后不会丢失数据，可以快速存取，具有电可擦除性，存储密度高，存取时有擦除和写入两种操作

假设某RAM芯片存储容量为16K\*8位，存储器的地址范围为：0000H-3FFFH

1. 简述微处理器中存储器映像的含义，给出某一芯片某一接口模块的存储器映像地址（范围）。

把4GB空间当做存储器来看，分成若干个空间，以安排一些实际的物理资源

比如下图：



代码块的存储器映像地址范围为：00000000H - 1FFFFFFFH

1. 什么是芯片的硬件最小系统，它由哪几个部分组成？

硬件最小系统是包括电源，晶振，复位，写入调试器接口等可使内部程序得以运行的规范的，可复用的核心构件系统。

由电源电路，晶振电路，复位电路，与写入器相连的SWD接口电路组成

1. 在芯片的电源电路中，一般靠近芯片引脚接有滤波电容，简述滤波电容的作用及工作原理，说明电容大小与滤波频率的关系。

滤波电容作用：抑制高频噪音

滤波电容工作原理：电容通交流隔直流

电容大小与滤波频率的关系：大电容过滤低频率杂波，小容值过滤高频波，一般成反比

1. 简述栈指针SP的基本作用，举例说明所学芯片初始SP应设置的值。

SP是存放堆栈栈顶的寄存器

假设堆栈区域为0- 3FFFFH ，初始SP设置的值为3FFFFH

1. 简要阐述通用嵌入式计算机的概念。

从硬件方面：把MCU硬件最小系统及面向具体应用的共性电路封装成一个整体，按照硬件构件要求进行原理图绘制、文档撰写、以及硬件测试用例设计。

从软件方面：把嵌入式软件分为基本输入输出系统（BIOS）与User程序两部分，BIOS程序先于User程序固化与MCU的非易失存储器上，BIOS先运行，然后User程序再运行。BIOS提供工作时钟以及面向知识的底层驱动构件，并为User提供函数调用接口。

8．为什么基于通用嵌入式计算机的概念，提高了智能终端软件设计的可移植性。

嵌入式计算机有良好的硬件适应性，也就是可移植性好，因为嵌入式计算机把MCU硬件最小系统封装起来，如实验用的STM32L431，提供了许多引脚，通过引脚只要遵循基本命名规范就可实现主程序及中断处理程序在微处理器之间的移植。

## 课本第3章 指令系统

1．通过汇编环境，给出以下指令的机器码。

（1）ADD r5,#20 （2）AND r6,r7 （3）BX r2

（1）



（2）



（3）



2．若r2=0x1234，r3 =0x5678，r5=0x9A4B，当源寄存器采用大端格式或小端格式时，分别给出以下指令执行后目标寄存器的值。

（1）REV r1,r2 r1 = 0x34120000

（2）REV16 r3,r3 r3 = 0x3412

（3）REVSH r4,r5 r4 = 0x00004B9A

3．给出把存储器的一个地址（label1）中的数据存储到另一个地址（label2）的指令代码。

LDR r1, [label1]

STR r1, [label2]

4．给出指令代码判断r1第3位是否为0，若为0转到label1。

AND r1, 0x100

JZ label1

5．若r2=80000001H，r3=90000005H，N、Z、C和V标志位的值都为0，给出以下指令的执行后r2的结果，并指出N、Z、C和V标志位的值。

（1）ADC r2,r3 （2）SUB r2,r3 （3）LSR r2,r3,#3

N Z C V r2

1. 0 0 1 1 10000006H
2. 1 0 1 0 EFFFFFCH
3. 0 0 0 0 12000000H

6．若sp=20002000H，r1=1234H，r2=5678H，r3=9ABCH，r4=DEF0，试说明执行下面指令后，sp=?，r5=？，r6=？并画图指出堆栈中各单元的内容。

push {r1-r4}

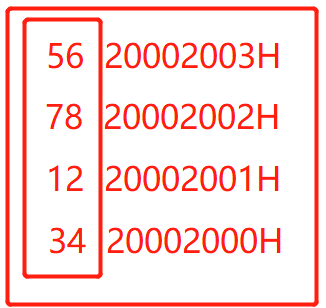
pop {r5,r6}

r5 = DEF0H

r6 = 9ABCH

sp = 20002004H

假设以小端格式存放，堆栈中各单元内容如下：



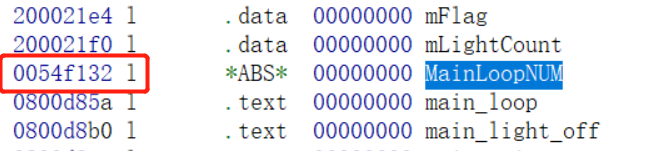
## 课本第4章 汇编语言框架

1. 简述立即数的前缀“=”和“#”的区别以及适用场合。

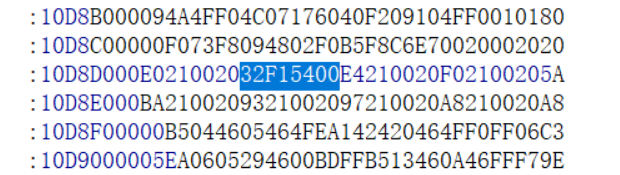
若立即数的值小于256，可使用“#”，否则只能使用“=”

1. 针对“Exam4-1”工程，利用.lst和.hex文件，分别找出MainLoopNUM 常量和printf函数的机器码在.hex文件中的存储地址。
2. MainLoopNUM：

在.lst文件中搜索MainLoopNUM



得到机器码为0054F132，大小端切换后为32F15400，在.hex文件中搜索



:10 D8D0 00 E0210020 32F15400 E4210020 F0210020 5A

偏移为D8D0

1. printf函数



跳到这个地址，是.lst文件中myprintf()函数的地址



1. 为了实现主程序和中断处理程序的可移植性，在编程中应考虑哪些问题？
2. 数据对齐问题，尽量保证所有类型都能自然对齐（比如4字节的数据在内存的地址正好被4整除）
3. 大小端问题，尽量保证大小端一致
4. 考虑在不同的机器中字长不一样的问题
5. 写高质量的注释，方便别人调用，方便别人维护
6. 芯片上电或复位后程序是直接从main函数开始运行吗？请阐述芯片上电后的启动过程。

不是

芯片上电后，运行MCU文件夹内的启动文件，完成复位MCU，从Flash中将已初始化的数据复制到RAM中，清零bss段数据，初始化系统时钟、标准库函数，最后转到main函数执行

5．当要在汇编程序中采用printf按"%d%d%d%x%x\n\0"格式输出五个数据时，参数应如何进行设置？

arg:

.asciz "%d%d%d%x%x\n"

ldr r0 , =arg

ldr r1 , =v1

ldr r2 , =v2

ldr r3 , =v3

ldr r4 , =v4

ldr r5 , =v5

bl printf

6．利用“Exam4-1”工程进行修改编程：先定义字、半字、字节三个变量，初值分别为0x8d12f6ab、0x9e2d、0x3a，然后在程序中对这三个变量都加1并存回变量中，最后通过printf进行输出这三个变量的值。

v1:

.word 0x8d12f6ab

v2 :

.hword 0x9e2d

v3:

.byte

add v1,#1

add v2,#1

add v3,#1

arg:

.asciz "%d%d%d\n"

ldr r0,=arg

ldr r1,=v1

ldr r2,=v2

ldr r3,=v3

bl printf

## 课本第5章 基于构件的汇编程序设计方法

1. 构件在设计的过程中应遵循哪些基本原则。

好的底层驱动构件需要具备：封装性，描述性，可移植性，可复用性 的基本要求

为了达到这些要求，需要满足以下基本原则：

1. 层次化原则：

针对应用场景和服务对象，分层组织构件

在构件的层次模型中，上层构件可以调用下层构件提供的服务，同一层次的构件不存在相互依赖关系，不能相互调用。

1. 易用性原则：

底层驱动构件需要做到：函数名简洁且达意；接口参数清晰，范围明确；使用说明语言精炼规范，避免二义性。若一个函数功能复杂，则需要拆分为多个函数

1. 鲁棒性原则：

遵循鲁棒性原则设计底层驱动构件需要做到：在明确函数输入输出的取值范围、提供清晰接口描述的同时，在函数实现的内部要有对输入参数的检测，对超出合法范围的输入参数进行必要的处理；使用分支判断时，确保对分支条件判断的完整性，对缺省分支进行处理；不能忽略warning。

1. 内存可靠使用原则：

对内存的可靠使用是保证系统安全、稳定运行的一个重要的考虑因素。遵循内存可靠使用原则设计底层驱动构件需要做到：

① 优先使用静态分配内存。相比于人工参与的动态分配内存，静态分配内存由编译器维护，更为可靠。

② 谨慎地使用变量。可以直接读写硬件寄存器时，不使用变量替代；避免使用变量暂存简单计算所产生的中间结果；使用变量暂存数据将会影响到数据的时效性。

③ 检测空指针。定义指针变量时必须初始化，防止产生“野指针”。

④ 检测缓冲区溢出，并为内存中的缓冲区预留不小于 20%的冗余。使用缓冲区时，对填充数据长度进行检测，不允许向缓冲区中填充超出容量的数据。

⑤ 对内存的使用情况进行评估。

1. 给出汇编语言主程序文件main.s的基本结构。

- 导入头文件

- 定义需要的字符串，变量等

- 主函数

- 启动部分

- 主循环部分（无限循环）

- 程序结束

1. 给出汇编语言中断处理程序文件isr.s的基本结构。

- 函数定义

- 定义代码段

- 中断处理程序（无循环）

4．读程序：

|  |
| --- |
| mov r2,#97  sub r2,r2,#32  mov r1,r2  ldr r0,=string\_test  bl printf |

其中printf函数用于输出数据；string\_test的定义如下：

|  |
| --- |
| string\_test:  .string " %c" |

请问：上述程序实现了什么功能。

输出ascii码为97-32的结果的字符

5．编程实现二进制与十六进制之间的转换。

string\_system\_convert:

.string "system\_convert : "

string\_system\_mesg:

.string "%d"

string\_system\_mesg2:

.string "%c"

.type convert\_hexade, function

convert\_hexade:

push {r0-r7,lr}

mov r2,#8

mov r4,#15

loop\_hexade:

mov r5,r7

sub r2,r2,#4

lsr r5,r5,r2

and r5,r5,r4

mov r6,r2

cmp r5,#10

bcs display\_string

ldr r0,=string\_system\_mesg

bl printf

mov r2,r6

cmp r2,#0

bne loop\_hexade

display\_string:

mov r3,r5

add r3,r3,#7

mov r1,r3

ldr r0,=string\_system\_mesg2

bl printf

mov r2,r6

cmp r2,#0

bne loop\_hexade

pop {r0-r7,pc}

6．查找文件：NATO Communications and Information Systems Agency. NATO Standard for Development of Reusable Software Components, 1991。认真阅读该文件，从中总结出500字左右的要点。

软件复用具有 提升软件的质量和生产效率，减少维护代价，支持快速原型，减少培训开销等优越性。

以下从开发流程上来说明软件重用NATO标准：

1. 需求分析
2. 建立一个鼓励复用现有软件的机制
3. 将软件的复用列入开发需求
4. 评估领域分析的必要性和可行性
5. 概要设计原则
6. 设计阶段应该支持前面建立的可复用需求，并为这些需求建立良好的框架，高层设计活动将不同需求划分给软件架构中的不同构件
7. 利用模型，构架和接口设计得到可复用的系统框架。

模型用于声明软件需求，将需求和现有的功能匹配，用分层的架构来分离耦合点，为所有构件划分并细化接口

1. 在设计时候考虑构件被修改的可能性
2. 采用适合的软件设计方法，能有效地组织软件系统并提高软件复用性。应该将构件封装成自包含的抽象实体，尽可能减少构件的外部依赖，做到低耦合、高内聚，保持模块之间有清晰的接口。
3. 选择CASE工具帮助贯彻设计原则，实现一致的软件工程方法学。
4. 详细设计与实现
5. 从设计到代码的转换必须遵循此前的系统结构并建造其上，必须意识到代码质量对复用至关重要，构件必须具有鲁棒性，结构清晰且易于维护。要采用合适的映射规则指导由设计元素到编程语言的映射
6. 程序结构的选择。程序结构的选择决定的个体构件的可复用性，应该将一起被复用的实体组织在一起，尽量减少与外部的依赖关系，控制构件的实现与底层平台的依赖性（如果不可避免，则需要将此模块独立出来）
7. 构件的接口定义：构件设计接口时候应该贯彻落实低耦合、高内聚的原则。
8. 参数化和异常处理：在构件执行之前确定参数取值，对象需要有初值，并且包括Exception机制报错给开发者
9. 测试
10. 测试标准：为软件增加可复用性带来的附加成本，构件的复用次数，各种构件的复用比例，构件被修改后的复用次数，复用对项目进度的影响等等
11. 测试作用：度量构件的可复用性，评估复用项目并估测复用的收益，为复用库提供信息
12. 测试过程：保证整个项目的复用需求被满足，对所有接口进行测试，测试每一条显式声明的可复用性需求。
13. 文档
14. 应用传统的文档标准：文档必须能帮助复用这理解和使用构件。构件的文档本身也可作为文档构件，应该采用一致的组织结构和文档格式，具备清晰完整的风格和机器可读的形式
15. 为复用库准备文档和复用者手册

## 课本第6章 存储器

1. 阐述CPU对内部寄存器、外设模块寄存器和存储器的访问方式有何不同？

CPU访问内部寄存器是直接访问，速度极快；

CPU访问外设寄存器通过IO接口中的寄存器（端口）；

CPU对存储器寻址是通过地址映射，通过比如页表的映射，硬件自动寻址到对应的存储器；

1. 查阅Flash存储器的发展历史，阐述Flash在线编程方式的演变过程。

Flash的在线编程方式包括：擦除和写入

擦除和写入的操作都是通过设置或者清除Flash存储器的控制寄存器FLCR中的某些位来完成的，后来封装成构件，包括多个基本操作的函数

历史上对闪存器件进行编程的传统方法有三种：

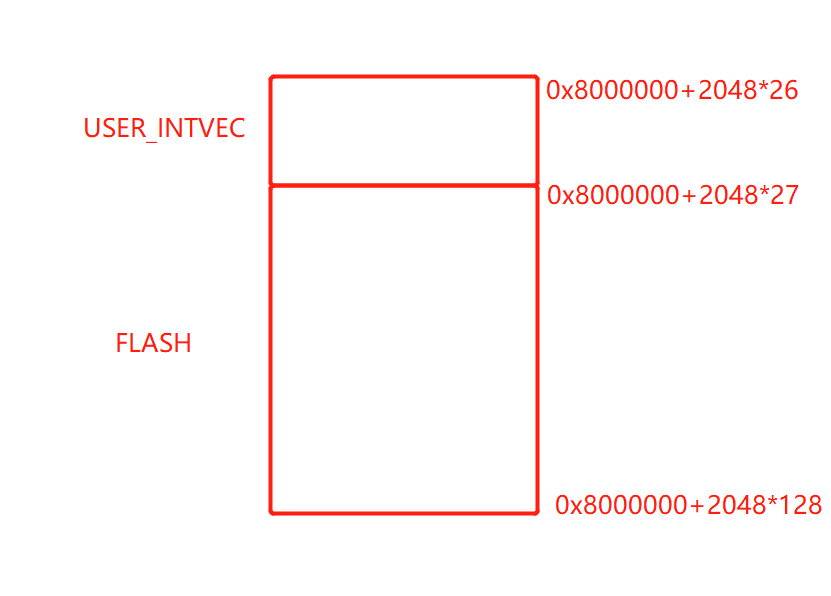
1. 在器件插入之前进行预编程，以后不能再修改
2. 将闪存器件安装到印刷电路板之后进行编程
3. 采用支持JTAG边界扫描链的ISP来控制与闪存连接的引脚
4. 芯片热启动和冷启动后，Flash存储器和RAM中的数据保持情况。

冷启动时，RAM初始化，热启动，RAM保持复位前的数据

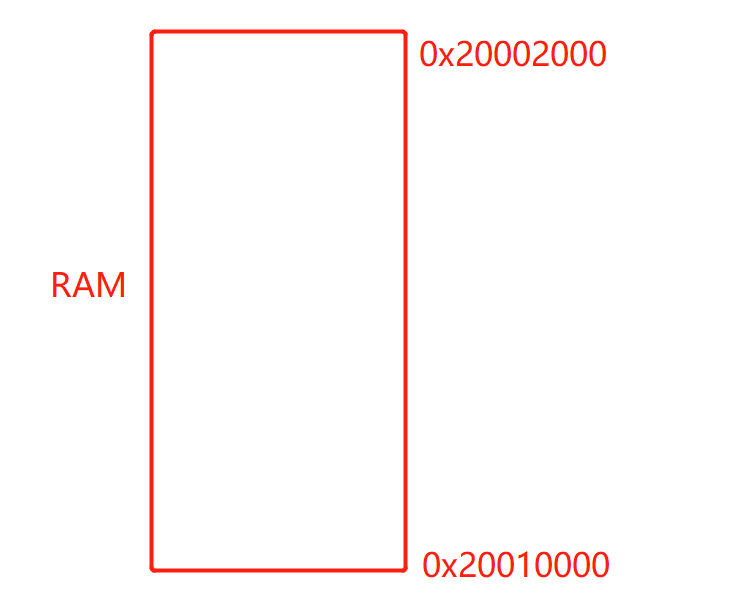
而Flash存储器在热启动和冷启动时，数据都保持不变

4．针对“Exam6\_2”工程，利用STM32L433RCTXP\_FLASH.ld和Exam6\_2.lst文件，给出Flash和RAM各段的存储空间使用情况。

Flash的存储空间：



RAM存储空间：



## 课本第7章 串行通信接口

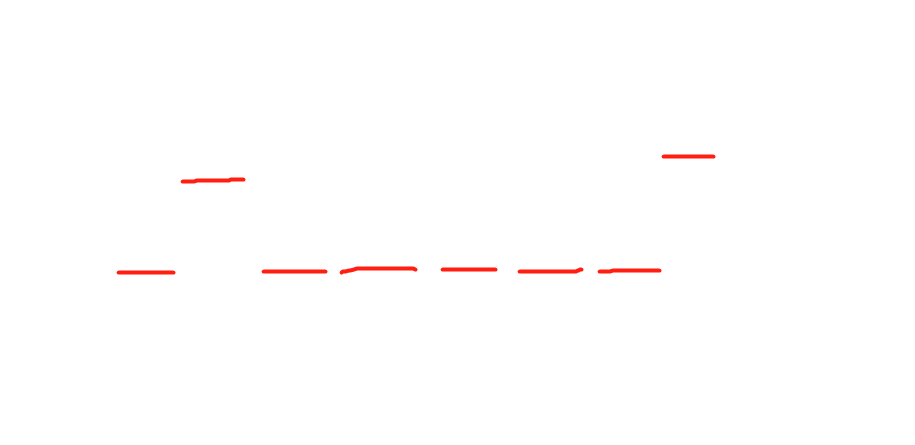
1. 简述MCU与PC之间进行串行通信，为什么要进行电平转换？如何进行电平转换？

因为MCU与PC之间的电平不同，如果要通信就要进行电平转换

在发送端需要用驱动电路将TTL电平转换成RS232电平，接收端需要用接收电路将RS232电平转换为TTL电平，电平转换器目前广泛使用MAX232芯片

1. 画出通过串口传送一个字符“A”的数据格式。

“A”的ascii码为41H，二进制表示为：0100 0001，即下图：



1. 设波特率为115200，使用NRZ格式的8个数据位、没有校验位、1位开始位、1位停止位，传输3K字节的文件最少需要多少时间?

3K Byte = 3\*2^10\*8 bit

3K字节2^13+2 bit

T = (3\*2^13+2) / 115200 = 21.33ms

1. 举例说明汇编语言和C语言在调用UART驱动构件上的区别。

传递参数的方式不同：汇编通过寄存器传递参数，C语言可通过堆栈或者寄存器传递参数

5．根据“Exam7\_2”工程的数据帧格式，给出解帧的流程。

“Exam7\_2”工程的帧格式：

帧头(2字节) + 数据长度(1-2字节) + 有效数据(N字节 N=数据长度) + 帧尾

解帧的流程：

1. 判断帧头前两个字符为‘$$’
2. 读取数据长度
3. 根据数据长度读取有效数据
4. 如果读到‘##’即读到帧尾

## 课本第8章 中断系统及定时器

1．简述中断源、中断向量表、中断处理程序和中断向量号的关系。

2．简述新增一个定时器的编程步骤。

3．如何验证程序触发了定时器中断？

4．如何验证初始化时定时器中断的时间间隔的正确性？

5．重载寄存器可设置的最大值和最小值是多少？对应的时间分别是多长？

6．若时钟频率选择48MHz，当要设置100ms、1ms产生一次Systick中断，那么重载寄存器的值分别要设置成多少？简述中断产生的时间与重载寄存器值之间的关系。

## 课本第9章 模数转换与数模转换

1．若ADC转换的参考电压为5V，要能区分0.05mV的电压，则采样位数为多少？

2．什么是量化误差，能否消除量化误差？请解释一下原因。

3．简述几种滤波方式，若AD采样十次的值为{859，897，0，834，857，856，849，900，857，866}，那么适合使用什么滤波方式？写出C程序。

4．简述如何读取STM32L4系列的内部温度传感器的值。

5．假设VREF+为3.3V，D/A转换位数n=12，试求当DOR=2000时，输出的模拟电压V0为多少？

6．对于一个12位DAC，若输出电压最小增量为0.001V，输入数字量为010110011101时，输出电压为多少？分辨率是多少？

## 课本第10章 直接存储器存取DMA

1．DMA相对于传统的传输方式来说，它的优点在哪里？

2．DMA可编程的数据传输数量最大是多少？

3．如何查询当前剩余数据量大小？

4．DMA传输中的循环模式与普通模式有什么区别？

5．通过使用串口的DMA接收方式来实现通过串口接收数据。

## 课本第11章 外接组件综合实践

1．在“Exam11\_1”工程的基础上，修改程序的颜色数据，使彩灯能呈现出多种颜色。灯颜色提示：青色为绿蓝混合，黄色为红绿混合，紫色为红蓝混合，白色为红蓝绿混合。

2．在“Exam11\_2”工程的基础上，修改程序采用PWM波的方式来控制蜂鸣器的声音的强弱。提示：通过改变PWM的占空比的方式来实现。

3．在“Exam11\_3”工程的基础上，修改程序当接收到串口发来的字符串“high”实现开启马达，当收到“low”时关闭马达。提示：可以参照7.3.2的数据帧结构进行组帧发送字符串，当收到字符串时要进行解帧判断。

4．在“Exam11\_4”工程的基础上，修改程序通过在数据段开辟一段包含8个数据的存储空间（类似定义一个包含8个元素的数组），将这8个数据作为参数传递给TM1637\_Display函数，实现在LED中显示这些数据。

5．在“Exam11\_5”工程的基础上，修改中断处理程序显示左侧和右侧物体出现的次数。

6．在“Exam11\_6”工程的基础上，修改中断处理程序显示检测到有人的次数。

7．在“Exam11\_7”工程的基础上，修改程序通过按钮控制蜂鸣器的强弱或马达的开与关。

8．在“Exam11\_8”工程的基础上，修改程序当声音传感器的AD采样值超过1000时，彩灯显示为红色。

9．在“Exam11\_9”工程的基础上，修改程序采用Systick或Timer中断实现每3秒采集一次x、y、z三个方向的加速度值。

## 课本第12章 通用计算机的基本结构及启动过程

1．简述PC机的基本结构，它与Arm体系的微型计算机有什么区别？

2．CPU有哪些参数？如何选择与CPU相匹配的主板？

3．什么是启动固件？有哪些启动固件？

4．请写出三种硬盘接口，并说明它们各自的特性。

5．简述UEFI+GPT的启动流程。

6．PC机有哪些操作系统？它们有什么共性？

## 课本第13章 微型计算机的发展方向

1．谈谈影响CPU的速度的原因。

2．新一代的存储器有着怎样的共通点，或者说人们对新一代的存储器有着怎样的期望？

3．什么是计算机指令系统？

4．为何交叉编译技术近些年发展迅速？

5．对比传统计算机，量子计算机最独特的优势是什么？

# 第4部分 实验报告

## 实验一：理解汇编程序框架及运行

## 实验二：基于构件方法的汇编程序设计

## 实验三：存储器实验

## 实验四：基于串行通信构件的汇编程序设计

## 实验五：理解中断与定时器

## 实验六：ADC-DAC模块实验

## 实验七：通过DMA实现内存间数据的搬运

## 实验八 综合实验

# 第5部分 杂记

# 期中总结报告

**第一章总结**

* 1. 微型计算机的介绍

以微处理器为核心，以地址总线，数据总线，控制总线为基础，连接存储器，输入输出接口电路而形成的一种体积小、方便使用的计算机。

常用的微型计算机有：个人电脑，手机，超级计算机，微控制器（MCU），单片机等

* 1. 冯诺依曼结构

三个要点：

（1）组成部分：由控制器，运算器，存储器，输入设备，输出设备组成

（2）信息由二进制表示

（3）指令和数据同时放在存储器中，计算机工作时能自动从存储器中取指令并执行指令。

一次计算的执行过程：

程序和数据通过输入设备进入存储器，控制器发出地址给存储器，获得程序中对应的指令，还可以得到相应的数据，控制器发命令给运算器，指挥运算器对数据进行相应的运算，将运算的结果存放到存储器中。最后需要通过输出设备将结果输出到计算机外部的某些存储介质上。这样才能完成整个计算的过程。

* 1. 三总线

三总线包括：地址总线，数据总线，控制总线

地址总线：决定了能访问的地址空间大小

数据总线：与存储单元进行信息交换，总线宽度等于计算机的字长

控制总线：传送控制信号和时序信号来控制CPU和存储单元的信息交换

* 1. 一条指令的执行过程

取指令，译码，取操作数，执行指令，写回数据，执行下一条指令

* 1. 计算机中信息的表示方式

（1）位（bit），字节（1 byte = 8 bits），字（1 word = 16 bits）

（2）机器数（数在计算机中的二进制表示），真值（带符号位的机器数对应的数值）

（3）补码表示方法

求补码方法：正数等于原码，负数除符号位取反+1

理解补码：时钟（差1补一圈减1）

一般求法：正数等于原码，负数等于2^n-原码

取值范围：8位补码（-128 ~ 127）

（4）浮点数的表示方法（32位为例子）

·符号位 1位

·阶码 8位

·尾数 23位

举例：

（-3.75）10 = （-1）\*（11.11）2 = （-1）\*1.111\*2^2

符号位：1

阶码：2+127 = 1000 0001

尾数：11100000000000000000000

合起来 -3.75表示为：1100 0000 1111 0000 0000 0000 0000 0000

* 1. 字符编码

（1）ASCII码

·常用ASCII码的记忆 ☆

30H-39H 0-9

41H-5AH A-Z

61H-7AH a-z

（2）中文编码GBK

·国标字库：一级字库（3755个），二级字库（3088个）

（3）区码和位码:由四位十进制数表示，前两位区码，后两位位码，每区100个字,分区管理汉字，每个十进制数代表一个字

**第二章总结**

2.1 微型计算机的性能指标

（1）字长：决定了计算机内存访问的空间，如64位，32位，字长影响了运算的精度

（2）主频：CPU的时钟频率

指令执行是以时钟周期执行的，主频越高，时钟周期越短，计算机运行越快

\*计算机频率的产生：晶体振荡器（纯净二氧化硅晶体的压电效应）

（3）存储容量：内存和外存大小，内存以字节为单位

（4）外设扩展能力：一台微型计算机可配置[外部设备](https://www.baidu.com/s?wd=%E5%A4%96%E9%83%A8%E8%AE%BE%E5%A4%87&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)的数量以及配置[外部设备](https://www.baidu.com/s?wd=%E5%A4%96%E9%83%A8%E8%AE%BE%E5%A4%87&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)的类型，对整个系统的性能有重大影响，如显示器的分辨率、多媒体接口功能和打印机型号等

1. 软件配置情况：软件配置情况直接影响[微型计算机系统](https://www.baidu.com/s?wd=%E5%BE%AE%E5%9E%8B%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%B3%BB%E7%BB%9F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)的使用和性能的发挥

通常应配置的软件有：[操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao" \t "https://zhidao.baidu.com/question/_blank)、计算机语言以及工具软件、数据库管理系统等

2.2 存储器

2.2.1 分类

（1）RAM

RAM（Random Access Memory ）随机存取存储器，主要作用是存储代码和数据供CPU 需要时调用，跟计算机里面的内存差不多,主要是用来存放程序运行中的过程数据,掉电后就会丢失之前的数据,所以程序在上电时需要进行初始化,否则上电后的数据是一个随机数,可能导致程序奔溃

（2）ROM

ROM掉电后数据不会丢失,但在程序运行过程中其数据不会改变.

（3）Flash

Flash即Flash Memory，全名叫Flash EEPROM Memory，又名闪存，是一种长寿命的非易失性（在断电情况下仍能保持所存储的数据信息）的存储器，数据删除不是以单个的字节为单位而是以固定的区块为单位，区块大小一般为256KB到20MB，Flash存放代码和常数。

2.2.2存储器的性能指标

1. 存储容量

存储容量 = 存储字数\*字长

存储字数表示存储器的地址空间大小，字长表示一次存取操作的数据量

1. 单位成本：每位价格 = 总成本/总容量
2. 存储速度：数据传输率 = 数据宽度/存取周期

一个存储周期包括 存取时间 和 恢复时间

1. 主存带宽：表示每秒从主存进出信息的最大数量

2.3 CPU内部寄存器

2.3.1 寄存器的分类

（1）通用寄存器 作用：存放运算数据或者计数数据等CPU运算时需要的数据，用于数据缓冲，也叫数据缓冲寄存器

（2）栈指针寄存器，即SP（Stack Point） 作用：访问RAM中的栈区

（3）程序指针类寄存器，如指令指针寄存器（存放下一条要读的指令在代码段中的偏移地址）

（4）程序运行状态类寄存器，如标志寄存器，程序状态字寄存器

（5）常用的其他功能寄存器，如表示浮点数运算、中断屏蔽等功能寄存器

2.3.2 常用的状态标志

CF：进位标志位，最高位产生进（借）位时，该位为1

PF：奇偶标志位，低8位中有偶数个1时，PF为1，否则为0

AF：辅助进（借）位标志，执行加减指令时，若结果低字节的低4位向高字节有进（借）位，AF=1，否则为0

ZF：零标志位，若结果为0，则ZF=1，否则为0

SF：符号标志位，若结果最高位为1，SF为1，否则为0

OF：溢出标志位，有符号数加减运算结果溢出OF=1，否则为0

NZCV状态标志：

N：运算结果为正或0，N=0

C：进位标志，有进位，C=1

Z：零标志 运算结果为0，Z=1

V：溢出标志，适用于加减法运算结果，如果符号位溢出，V=1（比如两个正数相加等于负的情况，V=1）

2.4 Arm Cortex-M4存储器映像

2.4.1存储器映像的含义

Arm Cortex-M4处理器直接寻址空间为4GB，地址范围为0x00000000-0xFFFFFFFF把4GB空间当做存储器来看，分成若干个空间，以安排一些实际的物理资源

☆常用的表示：

10位： 2^10 = 1K

20位： 2^20 = 1M

30位： 2^30 = 1G

40位： 2^40 = 1T

32位空间大小：2+30 = 4 \* G = 4G

2.4.2 Arm Cortex-M4内核的微型计算机实例

- STM32L4系列

（1）命名格式

STM32系列的命名格式为：

芯片家族 产品类型 具体特性 引脚数目 Flash大小 封装类型 温度范围

比如：STM32 L 431 R C T 6

（2）中断

中断是指MCU正常运行程序时，由于内核异常或MCU各模块发出请求事件引起MCU停止正在运行的程序，并运行中断处理程序

中断源分为：内核中断（主要是出现异常中断，出现后会复位芯片），外部中断（可屏蔽中断，可通过编程控制开启关闭，如进程调度）

1. 引脚功能分类：

1. 硬件最小系统引脚（包括电源输入，复位，晶振，SWD接口）

2. 对外提供服务的引脚

（4）硬件最小系统原理图

1. 电源和滤波电路：

多处供电能保证电流平衡

电容滤波原理：通交流隔直流

2. 复位电路

* + 1. 冷复位（无电到带电），热复位（带电到无电）
    2. 外部复位，内部复位
    3. 异步复位（立即有效），同步复位（下周期有效）

3. 晶振电路：提供工作时钟，原理：晶振

4. SWD接口电路：实现程序下载和调试功能

第三章总结

3.1 常用的指令保留字及含义

数据传输类

取数（从存储器取到寄存器）：

LDR（取相应地址的值-字），

LDRH（半字），

LDRB（字节），

LDRSB（有符号字节数据），

LDRSH（有符号半字数据），

LDM（多数据加载）

存数（从寄存器放到存储器）：

STR（将源寄存器中的32位字数据存放到存储器中），

STRH（半字数据），

STRB（字节数据），

STM（多数据存）

寄存器间数据传送：

MOV（相同大小寄存器间移动），

MOVS（字符串操作指令），

MVN（将源寄存器中的值取反加1，然后保存到目的寄存器）

栈操作：

PUSH（入栈），

POP（出栈）

得到PC指针加偏移的地址：ADR

数据操作类

算术运算类：

ADC（带进位加），

ADD（加），

RSB（逆向减法，即操作数2减去操作数1），

SBC（带借位减法），

SUB（减法），

MUL（乘法），

CMN（负向比较，将一个数和另一个数的二进制补码比较，更新标志位），

CMP（两个数做减法，并只修改符号位）

逻辑运算类：

AND（与），

ORR（或），

EOR（异或），

BIC（位清零，第二操作数为掩码，计算时将其取反和寄存器中数做与运算，用于清零某些位）

移位类：

ASR（算术右移，带符号），

LSL（逻辑左移），

LSR（逻辑右移），

ROR（循环右移）

位操作类：

TST（执行按位与操作，更新标志）

数据序转换类：

REV（一个寄存器中的字节逆向，并复制到另一个寄存器中），

REVSH（将低半字16位内的字节逆向，并将符号位扩展复制到另一个寄存器中），

REVH（将两个半字内的字节分别逆序，并复制到另一个寄存器中）

扩展类：

SXTB（带符号扩展一个字节到32位），

SXTH（带符号扩展一个半字到32位），

UXTB（无符号扩展一个字节到32位），

UXTH（无符号扩展一个半字到32位）

跳转控制类

B（无条件转移），

BL（转移并连接，用于call子程序），

BX（跳转并切换微处理器状态，若bit[0]为0，则目标地址代码被解释为ARM代码，否则为Thumb代码），

BLX（从ARM指令集跳转到目标地址，并将处理器状态切换为Thumb状态，用于调用使用Thumb指令集的子程序）

其他指令

BKPT （断点指令），

CPSIE（开总中断），

CPSID（关总中断），

DMB（数据内存屏蔽），

DSB（数据同步屏蔽），

ISB（指令同步屏蔽），

MRS（加载特殊寄存器的值到通用寄存器，仅在特权模式下生效），

MSR（把通用寄存器中的值存到特殊寄存器中，仅在特权模式下生效），

SEV（发送事件指令，向所有处理器发送一个事件），

SVC（操作系统服务调用，触发SVC异常），

WFE（休眠，在发生事件时候被唤醒），

WFI（休眠，在发生中断时候被唤醒）

3.2 寻址方式

（1）立即数寻址

MOV R0, #0x1F （带#为立即数）

1. 寄存器直接寻址

MOV R2, R1

1. 直接地址寻址

LDR R1,label ;从label读取4字节到R1

1. 寄存器间接寻址

LDR R3,[R4] ; 把R4内地址的东西取到R3中

LDR R3,[PC，#100] ; 从地址PC+100处读4字节到R3

3.3 常用的伪指令

（1）符号定义伪指令

用于定义ARM汇编程序的变量，对变量进行赋值，以及定义寄存器名称等

包括GBLA、GBLL、GBLS、LCLA、CN、CP、DN、FN、RLIST、SETA

（2）数据定义伪指令

包括LTORG、MAP、DCB、FIELD、SPACE、DCQ、DCW等，主要用于数据表定义、文字池定义、数据空间分配等

（3）汇编控制伪指令

包括IF、ELSE、ENDIF、WHIL、WEND、MACRO、MEND等，主要用于条件控制、宏定义等操作

第四章总结

4.1 各种电平标准

（1）TTL电平标准

高电平：大于电源电压的百分之八十

低电平：小于电源电平的百分之二十

通讯时，传输一般在5m以内

（2）CMOS电平

接近电源电压表示1

电压接近0V表示0

CMOS电路是电压控制器件，输入电阻极大，对干扰信号十分敏感，优点是噪声容限宽，静态功率小

（3）232电平

+12V表示0

-12V表示1

传输30m左右

（4）485电平

差分电路，两根导线传一个电平，根据差值确定电平，抗干扰能力强

传输1km左右

工业上用的比较多

**注：不同电平标准对写程序无影响**

4.2 上拉电阻与下拉电阻的接法

芯片输入引脚的接法

（1）上拉电阻

引脚与电源之间有R1，引脚与接地之间有开关和R2，开关断开时引脚高电平，闭合低电平，R1>>R2, R1为上拉电阻，该引脚平时为高电平

（2）下拉电阻

引脚与电源之间有开关与R1，引脚与接地之间有R2，开关断开时引脚低电平，闭合高电平，R1>>R2, R2为下拉电阻，该引脚平时为低电平

4.3 输入引脚的接法

（1）直接驱动发光二极管，低电平亮，高电平不亮

（2）通过三极管驱动蜂鸣器，其中三极管的作用为开关，输出高电平三极管导通，低电平截止

4.4 GPIO

GPIO为通用输入输出端口（General-purpose input/output），是STM32芯片的可控引脚，所有gpio引脚都有基本输入（能知道外部是高电平或者低电平）和输出（能输出高电平或者低电平）功能

GPIO主要的函数：

1. gpio\_init 引脚初始化
2. gpio\_get 获取引脚状态
3. gpio\_set 设定引脚状态

调用gpio函数时候，参数放在寄存器中传递

第五章总结

概述

软件构件的意义在于可复用性，可重用性，提高开发人员的开发效率，通过使用具有鲁棒性的软件构件，开发人员既能够保证质量又能保证开发速度。

在编写软件构件的时候，不能出现调用例如输入输出的具体的底层硬件，这样不能保证构件的可移植性，可复用性。

5.1 构件设计的基本思想

在设计底层构件时，最关键的工作是要对构件的共性和个性进行分析，设计出合理的、必要的对外接口函数及其形参。尽量做到：当一个底层构件应用到不同系统中时，仅需修改构件的头文件，对于构件的源程序文件则不必修改或改动很小

5.2 构件设计的基本原则

好的底层驱动构件需要具备：封装性，描述性，可移植性，可复用性等基本要求，为了达到这些要求，需要满足以下基本原则：

（1）层次化原则

（2）易用性原则

（3）鲁棒性原则

（4）内存可靠使用原则

5.3构件分类

（1）基础构件：

基础构件是根据MCU内部功能模块的基本知识要素，针对MCU引脚功能或MCU内部功能，利用MCU内部寄存器，所制作的直接干预硬件的构件。

（2）应用构件：

应用构件是使用基础构件并面向对象编程的构件。

（3）软件构件：

软件构件是一个面向对象的具有规范接口和确定的上下文依赖的组装单元，它能够被独立使用或被其他构件调用。

5.4程序流程控制结构

（1）顺序结构  
（2）分支结构  
（3）循环结构