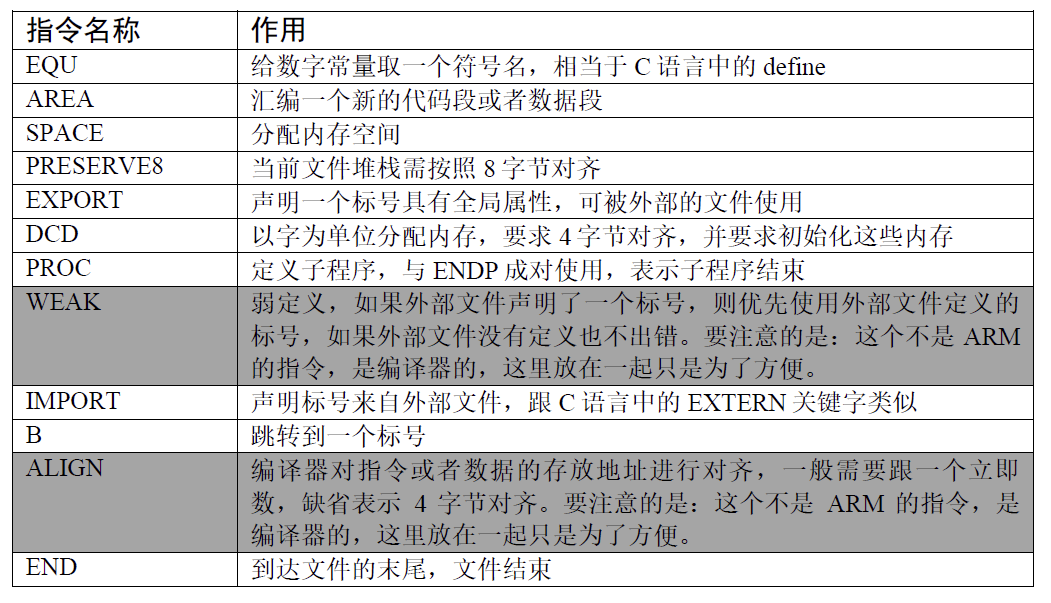
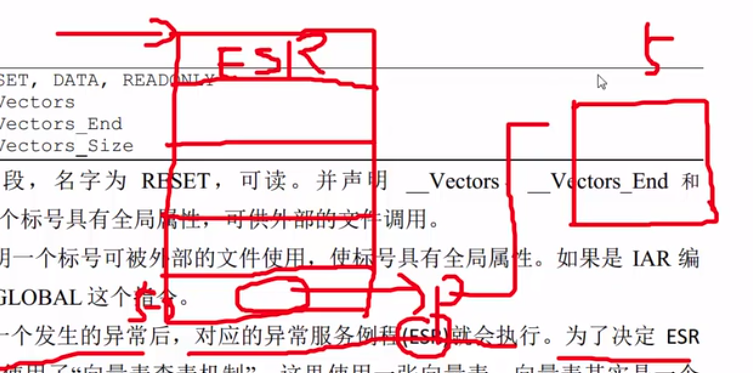
**一.启动文件汇编指令集合**

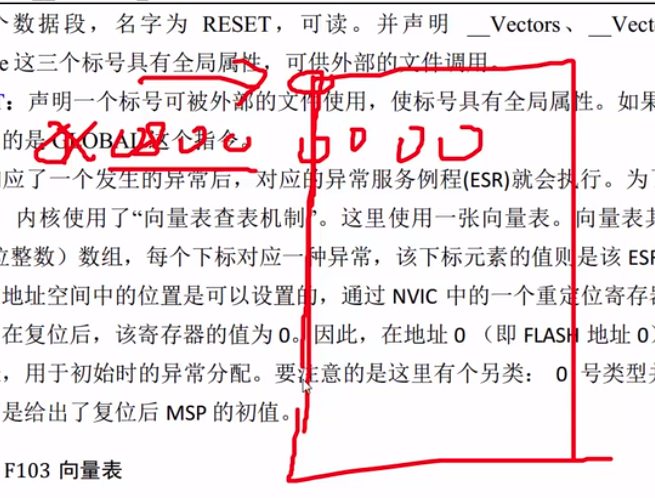
**二.中断向量表**

**1.中断的向量表的意义**

单片机会定义一张表(如下图)类似一个数组，每个元素4个字节，每个元素用来存放ESR的地址。当内核响应中断服务程序时，运用查询的方法，根据中断的函数名直接对应到相应的ESR编号(类似元素序号例如5)，ESR相应编号的4字节里就存放的一个地址，然后地址在对应执行相应的中断服务函数。这种操作会大大提高程序运行效率。详见《野火手册15.3.3》



Flash实际的是从0x00000000开始的。 32的程序从0x08000000开始存储。即把0x080映射到flash的0x00上。即把flash的0x000地址叫做0x80 所以中断表实在FLash的0x0开始，而在整个32系统的0x08开始储存



**二****,栈，堆，bss，data，code****(txet)**

**1.存储的内容不同**

栈：存放局部变量 函数参数值 函数返回地址

也就是说我们函数括弧“{}”中定义的变量（但不包括static声明的变量，static意味着在数据段中存放变量）。除此以外，在函数被调用时，其参数也会被压入发起调用的进程栈中，并且待到调用结束后，函数的返回值也会被存放回栈中。

堆：malloc()分配动态地址 free函数释放

bss：通常是指用来存放程序中**未初始化的全局变量**的一块内存区域，

bss段属于静态内存分配

data：通常是指用来存放程序中**已初始化的全局变量**的一块内存区域。

data段属于静态内存分配

code(txet)：通常是指用来存放程序**执行代码**的一块内存区域。这部分区域的大小在程序运行前就已经确定，并且内存区域通常属于只读在代码段中，也有可能包含一些只读的常数变量，例如**字符串常量等**

一个程序本质上都是由 bss段、data段、text段三个组成的。在嵌入式系统的设计中也非常重要，牵涉到嵌入式系统运行时的内存大小分配，存储单元占用空间大小的问题。

**2.区别**

栈：向低地址扩展

堆：向高地址扩展

显然如果依次定义变量

先定义的栈变量的内存地址比后定义的栈变量的内存地址要大(因为向下增长)

先定义的堆变量的内存地址比后定义的堆变量的内存地址要小(因为向上增长)

栈和堆的增长都是连续的，C语言不提供内存保护机制类似的功能，如果堆一直增长，栈一直申请，然后就会导致栈溢出，从而导致程序崩溃。

如果我们设置了堆的空间大小，但是我们程序中没有进行malloc申请，那么在程序事假运行的时候，我们栈的空间超过本身设置的空间，进入到堆里面，那么程序是不会出错的，但是超过了堆的空间了，进入到全局变量区域，就会出现莫名其妙的错误。

**三, .map文件**

**1.MAP文件的各单位的概念：**

RO：Read-Only的缩写，包括RO-data(只读数据)和RO-code(代码)。

RW：Read-Write的缩写，主要是RW-data，RW-data由程序初始化初始值。

ZI：Zero-initialized的缩写，主要是ZI-data，由编译器初始化为0。

.text：与RO-code同义。（text是执行代码大小只读）

.constdata：与RO-data同义。(const是存在flash里的数据 只读)

.bss： 与ZI-data同义 (未初始化的全局变量变量 未初始化)

.data：与RW-data同义 (初始化的全局变量变量存在RAM 可读可写)

**2.双击工程名就会出现.map文件,点击进去就能查看到各种变量的存情况。MAP文件的结构如下:**

详情:https://blog.csdn.net/qlexcel/article/details/78884379

1. Section Cross References（模块、段的交叉引用关系）
   1. 解释：主要是各个源文件生成的模块之间相互引用的关系。“refer to”是引用的意思。
   2. 举例：main.o(i.main) refers to led.o(i.LED\_Init) for LED\_Init

main函数和led被编译成了main.o和led.o函数，i.main是main.c中main函数的入口, i.LED\_Init是led.c中LED\_Init**函数的入口**。即这句话就是main中的main引用了led中的LED\_Init函数。

1. Removing Unused input sections from the image（移除未使用的段）
   1. 解释：就是将库中没有用到的函数从可执行映像中删除掉，减小程序的体积。
   2. 举例：Removing delay.o(i.delay\_ms), (76 bytes).

delay文件中的delay\_ms在函数中并没有使用，因此编译的时候移除掉减小程序体积。

* 1. 最后一栏是移除的统计结果

110 unused section(s) (total 4884 bytes) removed from the image.

110个函数 总共4884 bytes被移除

1. Image Symbol Table（映射符号表，列出了各个段所存储的对应地址）
   1. 解释：分为**Local Symbols**局部 和 **Global Symbols**全局。

**Local Symbol**下是static静态声明的变量，C文件中函数的地址，和static声明的函数的大小(静态变量定义的函数只能本文件内引用)

**Global Symbols**记录了全局变量的地址和大小，C文件中函数的地址及其代码大小，汇编文件中的标号地址（作用域全工程），

举例：Removing delay.o(i.delay\_ms), (76 bytes).

delay文件中的delay\_ms在函数中并没有使用，因此编译的时候移除掉减小程序体积。

* 1. 各单位解释：

Value：存储对应的地址； 0x0800xxx指FLASH存储存储的值和变量

0x200xx指在RAM存储的值和变量

Ov Type：符号对应的类型 全局、静态变量等位于0x2000xxxx的内存RAM中

Size：存储大小 我们怀疑内存溢出，可以查看代码存储大小来分析

Object(Section)：当前符号所在段名 即在哪个函数里

1. Memory Map of the image（映像的内存分布）
   1. 解释: 映像文件可以分为加载域（Load Region）和运行域（Execution Region）：加载域反映了ARM可执行映像文件的各个段存放在存储器中的位置关系。下面是部分截图，另外映像中的入口点就是程序开始执行的位置。
   2. 各单位解释：

Exec Addr：运行域地址

Load Addr：加载域地址

Size：存储大小

Type：类型

Data：数据类型

Code：代码类型

Zero：未初始化变量类型

PAD：这个类型在map文件中放在这个位置，其实它不能算这里的类型。要翻译的话， 只能说的“补充类型”（因为arm32位的 但有些定义8位16位的变量就会剩余一部分，则就可以自动对齐）

* 1. **加载域**就是程序在Flash中的实际存储

**运行域**是芯片上电后的运行状态

因为MCU没上电时RAM中没有数据，所以此时所有的东西（包括代码、变量、初始值等）都是存放在flash中的，当上电后又要把变量等复制到RAM中才能正常运行。

1. Image component sizes（映像组成大小）
   1. 分为三部分Object Totals ，Library Totals 和整个映像的说明

Object Totals：运行函数占的字节

Library Totals：显示已提取并作为单个对象添加到映像中的**库成员**占用了多少字节

整个映像的说明：总共字节数的统计

* 1. 各单位解释:

**第一部分：**

Code (inc. Data) ：inc代表代码占了多少字节

Data代码的字节里有多少内联数据如文字池 和短字符串。

RO Data ：显示只读数据占用了多少字节

RW Data ：显示读写数据占用了多少字节。

ZI Data ：显示零初始化的数据占用了多少字节。

Debug ：显示调试数据占用了多少字节，例如，调试输入节以及符号和字符串。

Object Totals ：显示链接到一起以生成映像的对象占用了多少字节。

incl. Generated)：链接器会生成的映像内容(连接器链接过程中生成的映像大小)例如，交互操作中间代码。 如果 Object Totals 行包含此类型的数据，则会显示在该行中。本例中共有 1016 字节的 RO 数据，其中32字节是链接器生成的 RO 数据。

incl. Padding ：链接器根据需要插入填充，以强制字节对齐。

**第二部分：**

Library Totals显示已提取并作为单个对象添加到映像中的库成员占用了多少字节。

incl. Generated)：库成员不需要连接，所以没这行

incl. Padding ：链接器根据需要插入填充，以强制字节对齐。

**第三部分：**

Grand Totals：显示映像的真实大小。

ELF Image Totals：ELF(Executable and Linking Format)可执行链接格式映像文件大小。

ROM Totals：显示包含映像所需的 ROM的最小大小。这不包括 ZI数据和存储在ROM 中的调试信息。