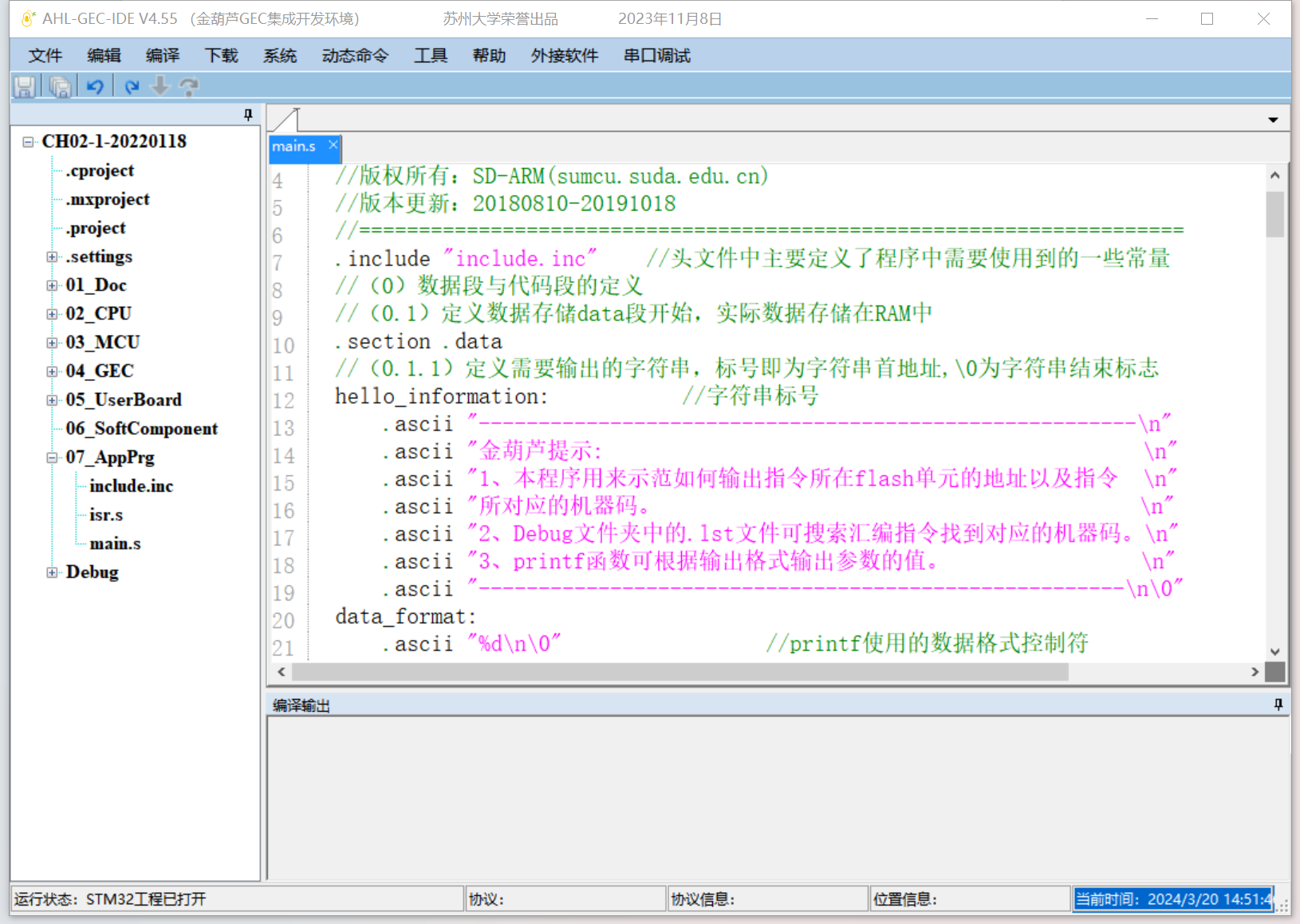
now照葫芦画瓢，开始新一个项目的流程。

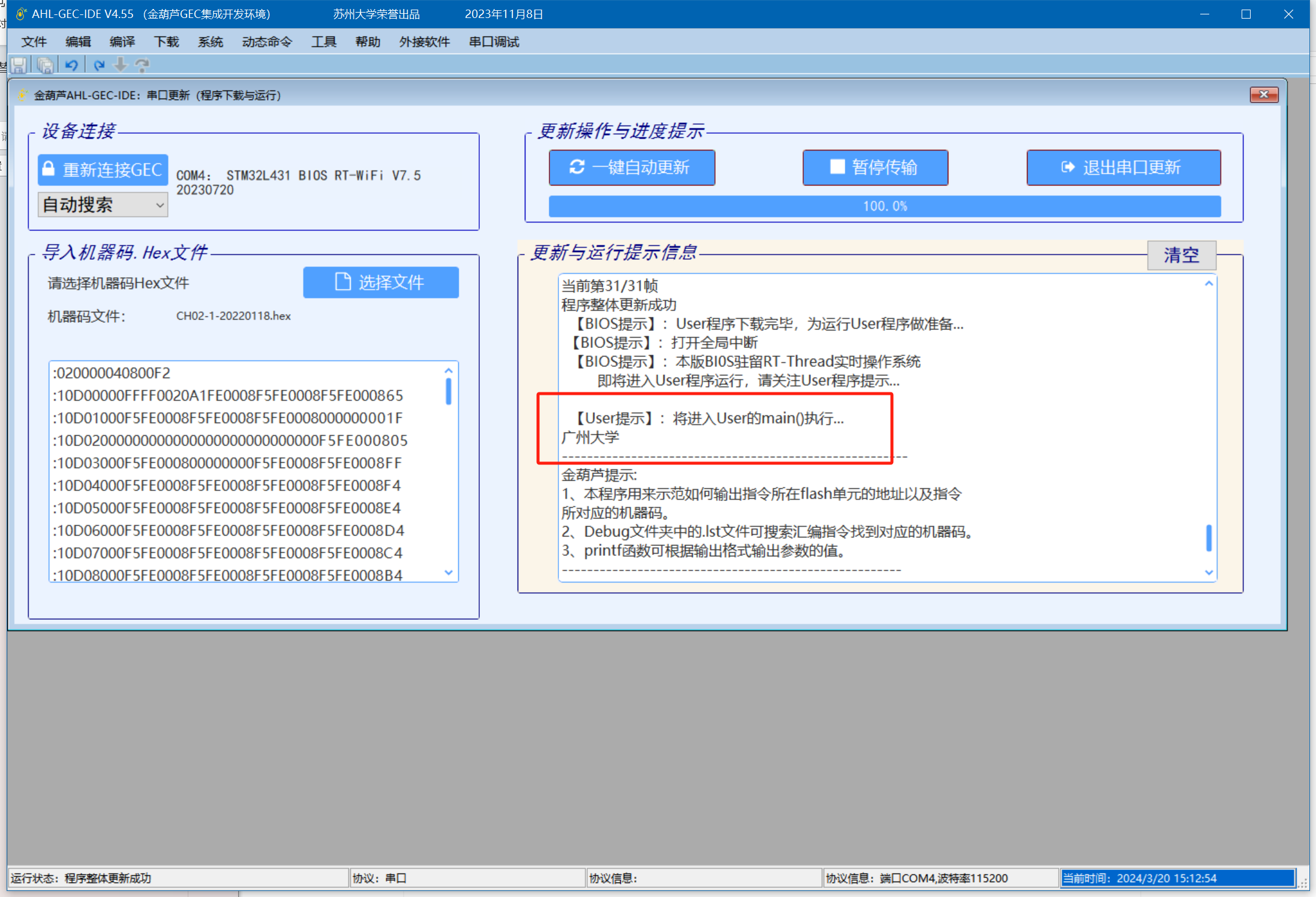
打开04-Software/ch02/CH02-1-20220118工程目录编译下载运行，单击菜单“文

件”→“导入工程”,出来的就是main.s



现在来看需求：修改main.s源文件，增加以下内容:

1. 在第一行显示“广州大学”字样。



理解一下：在汇编语言中，通常将程序的内存分为不同的段，其中最常见的是数据段（.data）和代码段（.text）。

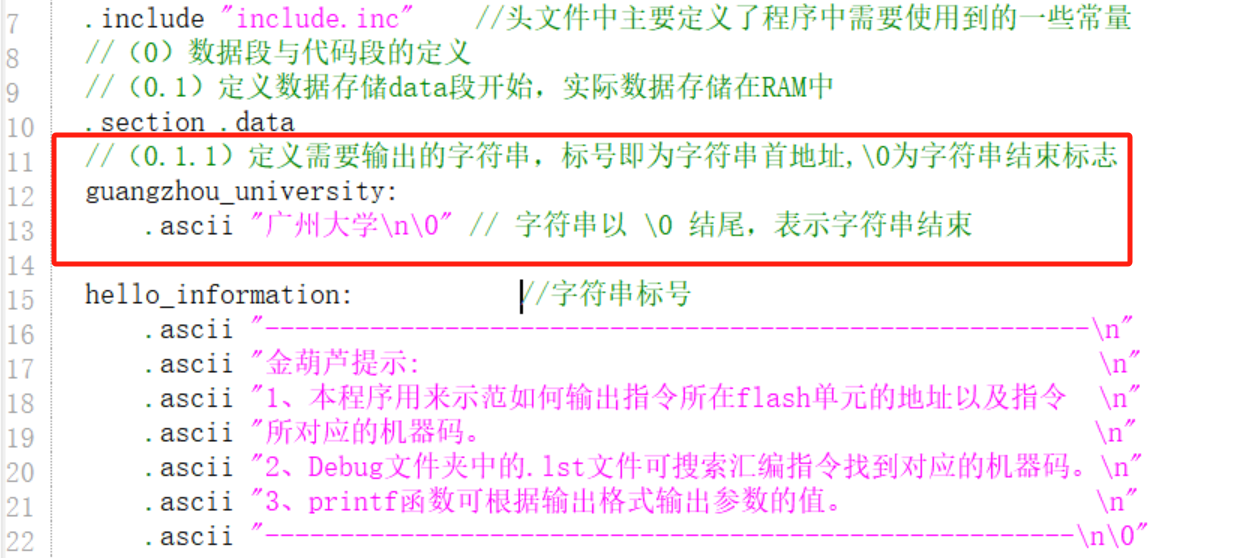
数据段（.data）：

* 数据段用于存储程序中使用的数据，这些数据通常存储在 RAM 中。
* 在数据段中，可以定义变量、常量以及字符串等数据。
* 在汇编语言中，用 .section .data 指令来定义数据段。

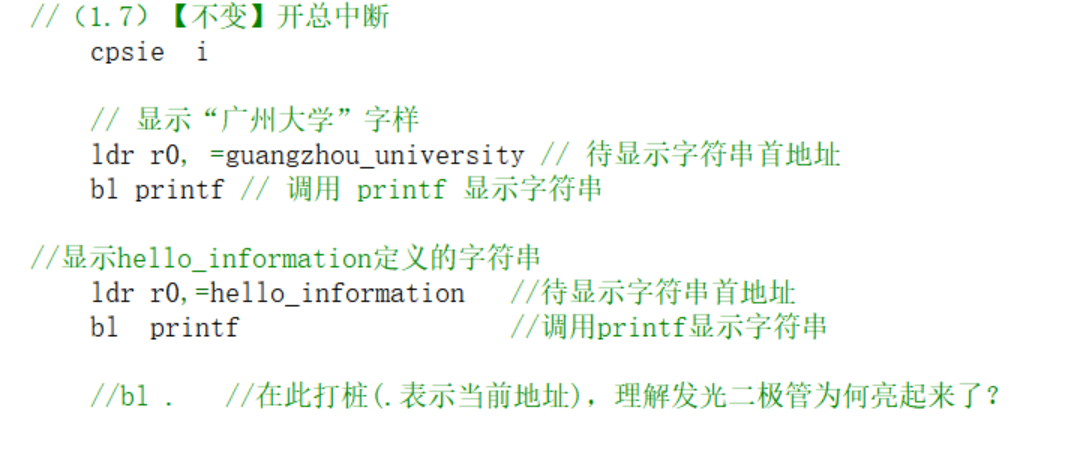
代码段（.text）：

* 代码段存储着程序的指令，通常存储在 Flash 或 ROM 中。
* 在代码段中，编写的汇编指令被存储和执行。
* 在汇编语言中，用 .section .text 指令来定义代码段。

所以找代码里面的数据段，定义一下广州大学，他怎么写我怎么写



在数据段定义了，然后就得在代码段编写相应的操作来执行，所以就去主函数的初始化部分之后调用 printf 函数来显示这个新的字符串，放在hello的前面



这段代码首先将字符串 "guangzhou\_university" 的首地址加载到寄存器 r0 中，然后通过 bl 指令调用 printf 函数，将寄存器 r0 中的地址作为参数传递给 printf 函数。

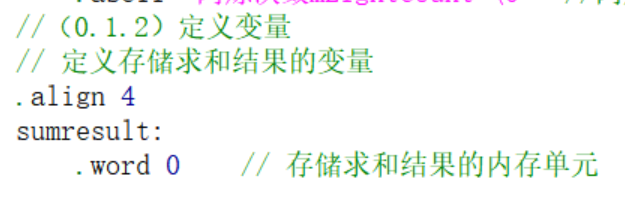
在 printf 函数内部，它会使用这个地址来访问存储在该地址处的字符串，然后将字符串打印到输出设备上

2、编写一个1+2+..+10的程序，将求和结果存入名为

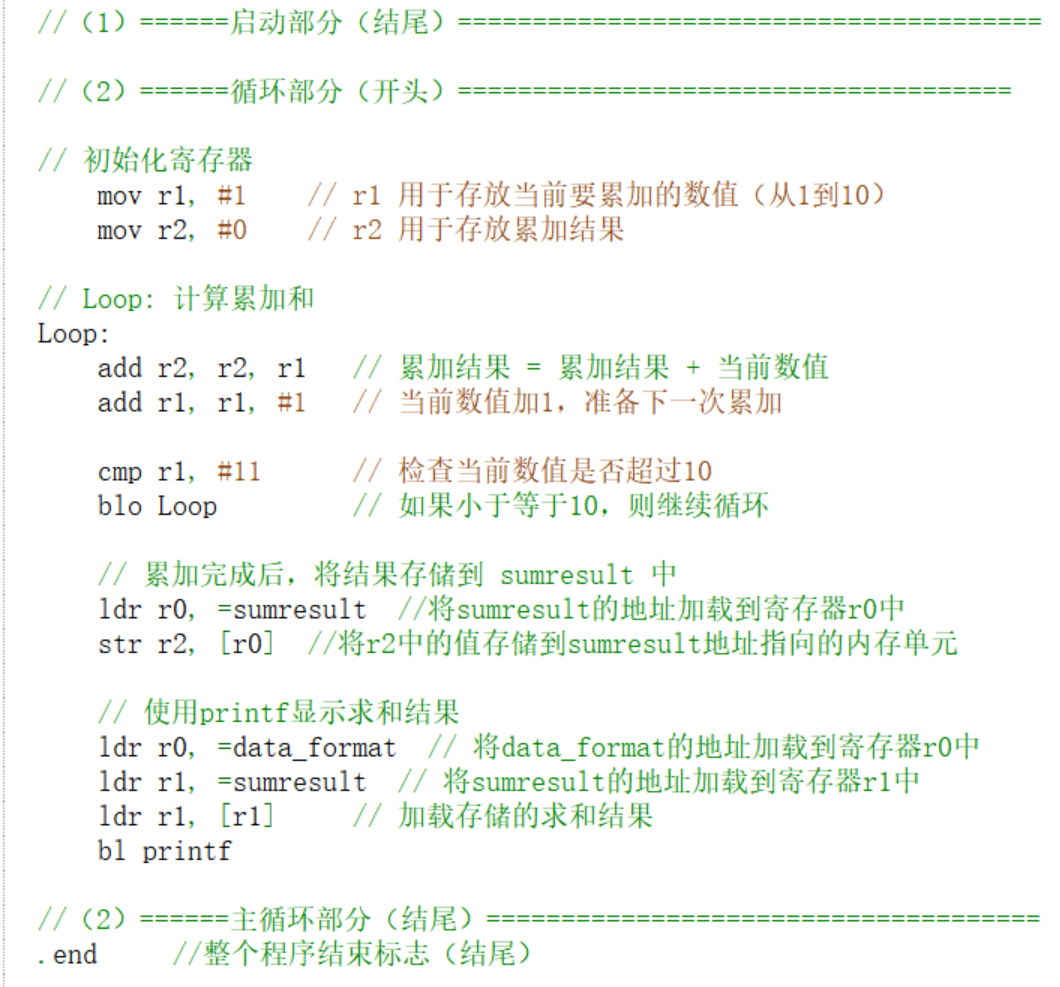
“sumresult”的内存单元中，并将求和结果用printf显示出来。



同样的操作，还是在数据段定义变量，在代码段实现操作：



初始内存单元的值为0是为了防止有其他的数据混杂导致结果错误



先初始化一下变量，然后再写操作（类似编写c语言的良好习惯）

然后是经典的累加操作，写一个循环，退出条件=11

累加完成之后就把得到的值存起来，再用printf输出

有的帅哥帅姐会问，为啥用这么怪异的格式输出，因为汇编就是这么的古板：

在ARM汇编中，函数的参数传递通常是按照一定的约定来进行的。在函数调用中，特定的寄存器用于传递参数给函数，这些约定由编译器和汇编器共同遵循。对于printf函数，通常的约定是：

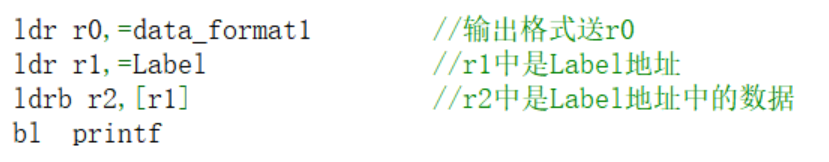
第一个参数（通常存储在 r0 寄存器中）是格式控制字符串，即包含格式化指示符的字符串，它告诉printf函数如何格式化后续参数。

后续的参数（如待打印的数值）通常存储在其他寄存器中，例如 r1、r2、r3 等。

那又有靓仔会问，累加得到的数值能不能用立即数存到sumresult 里，而不用先加载地址再加载结果呢？

那当然不行，因为因为地址通常是在运行时确定的，而立即数在汇编时就已经确定了。因此，在汇编语言中，通常需要先将地址加载到寄存器中，然后再将数据存储到该地址中。

按照我们照葫芦画瓢的规律，看下面的代码也能发现这个输出操作：



指定格式 地址 间接寻址 输出

最后我用.end结束程序，下面的LED小灯显示的就不再运行了

这样我们就完成了这个简单的需求

窗体底端