### 深圳杯数学建模挑战赛2024C题

编译器版本的识别问题

随着程序设计语言的不断变化，编译器也会不断更新。例如，GCC（the GNU Compiler Collection）就已经更新到了13.2.0版本[1]。不同版本的编译器在编译同一程序脚本时，编译结果会存在一定的差异；相同版本的编译器在使用不同编译选项时，编译结果也会出现差异。能否利用编译结果差异区分编译器的版本？

难点：1、切换编译器得到编译结果；2、选择编译结果的主要特征。

2000年全国大学生数学建模竞赛A题DNA序列分类问题

给定20个已知类别的DNA序列，其中序列标号1-10 为A类，11-20为B类。请从中提取特征，构造分类方法，并用这些已知类别的序列，衡量你的方法是否足够好。然后用你认为满意的方法，对另外20个未标明类别的人工序列（标号21—40）进行分类 ，例如下面三个序列

atggataacggaaacaaaccagacaaacttcggtagaaatacagaagcttagatgcatatgttttttaaataaaatttgtattattatggtatcataaaaaaaggttgcga A类

gtattacaggcagaccttatttaggttattattattatttggattttttttttttttttttttaagttaaccgaattattttctttaaagacgttacttaatgtcaatgc B类

ccattagggtttatttacctgtttattttttcccgagaccttaggtttaccgtactttttaacggtttacctttgaaatttttggactagcttaccctggatttaacggc ？

1. GCC编译器的安装教程（Windows环境）：<https://blog.csdn.net/qq_45601448/article/details/109158588>；
2. GCC使用教程：<https://blog.csdn.net/qq_42475711/article/details/85224010>
3. 附件中提供的是源码，参考源码安装GCC（Linux环境），构建Linux环境可以通过创建虚拟机或者电脑上再安装Linux系统。
4. 利用mingw安装GCC：<https://files.1f0.de/mingw/?login=from_csdn>
5. 在VScode 中使用EASYX详细教程：<https://blog.csdn.net/qq_35598074/article/details/110749803>

附件中提供的.cpp 文件是C++编程的源代码文件，包含了程序员编写的程序逻辑。为了能够在计算机上运行这个程序，我们需要将源代码编译成机器可以理解的指令，这个过程就是编译。

编译过程通常由编译器完成，如GCC。编译器读取 .cpp 文件，检查语法错误，将源代码转换成机器码，并可能进行一些优化。这个过程结束后，编译器会生成一个或多个文件，其中在Windows系统上通常是一个 .exe 文件（可执行文件）。.exe 文件包含了程序运行所需的所有机器码和可能的数据，它是源代码经过编译后的最终产品。可以直接运行这个 .exe 文件来执行程序。

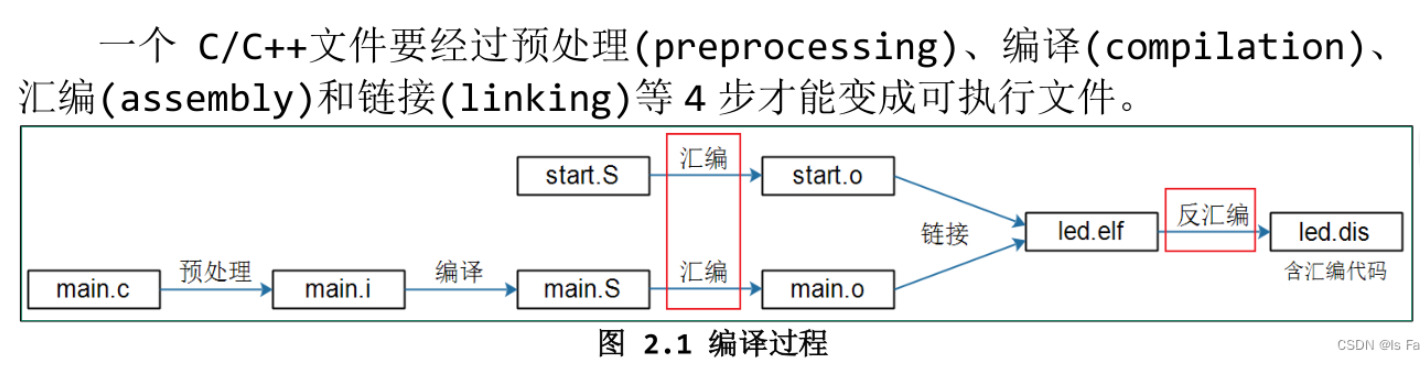
* 对于整个编译过程

步骤一：预处理（预编译）：编译处理宏定义等宏命令，生成后缀为“.i”的文件

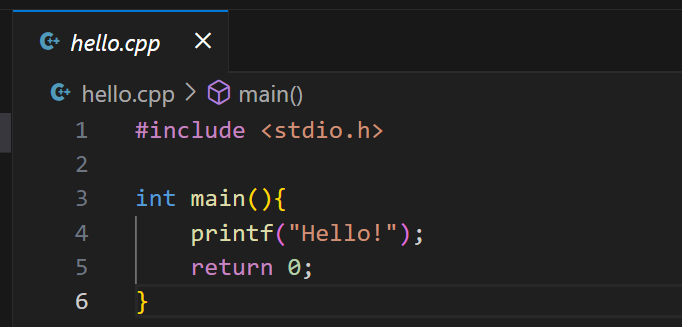
步骤二：编译：将预处理后的文件转换成汇编语言，生成后缀为“.s”的文件

步骤三：汇编：由汇编生成的文件翻译为二进制目标文件，生成后缀为“.o”的文件

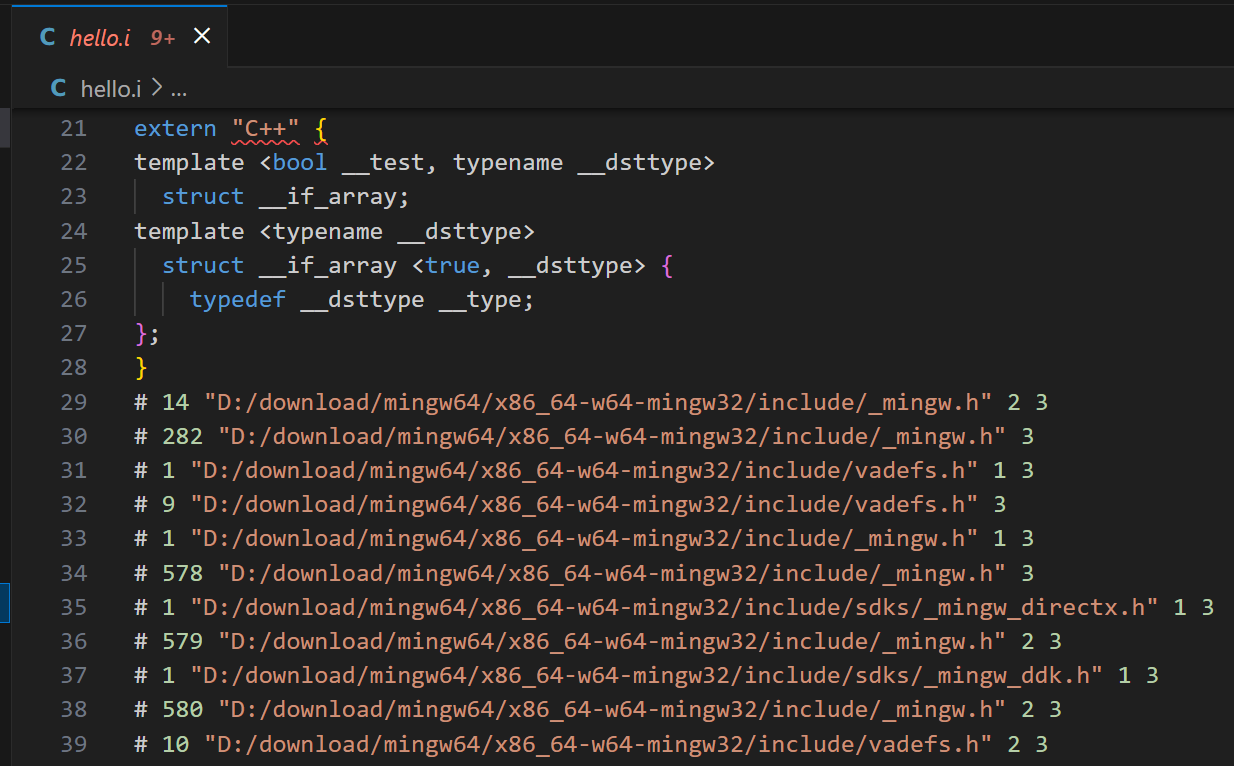
步骤四：连接：多个目标文件（二进制）结合库函数等综合成的能直接独立执行的执行文件，生成后缀为“.exe”的文件



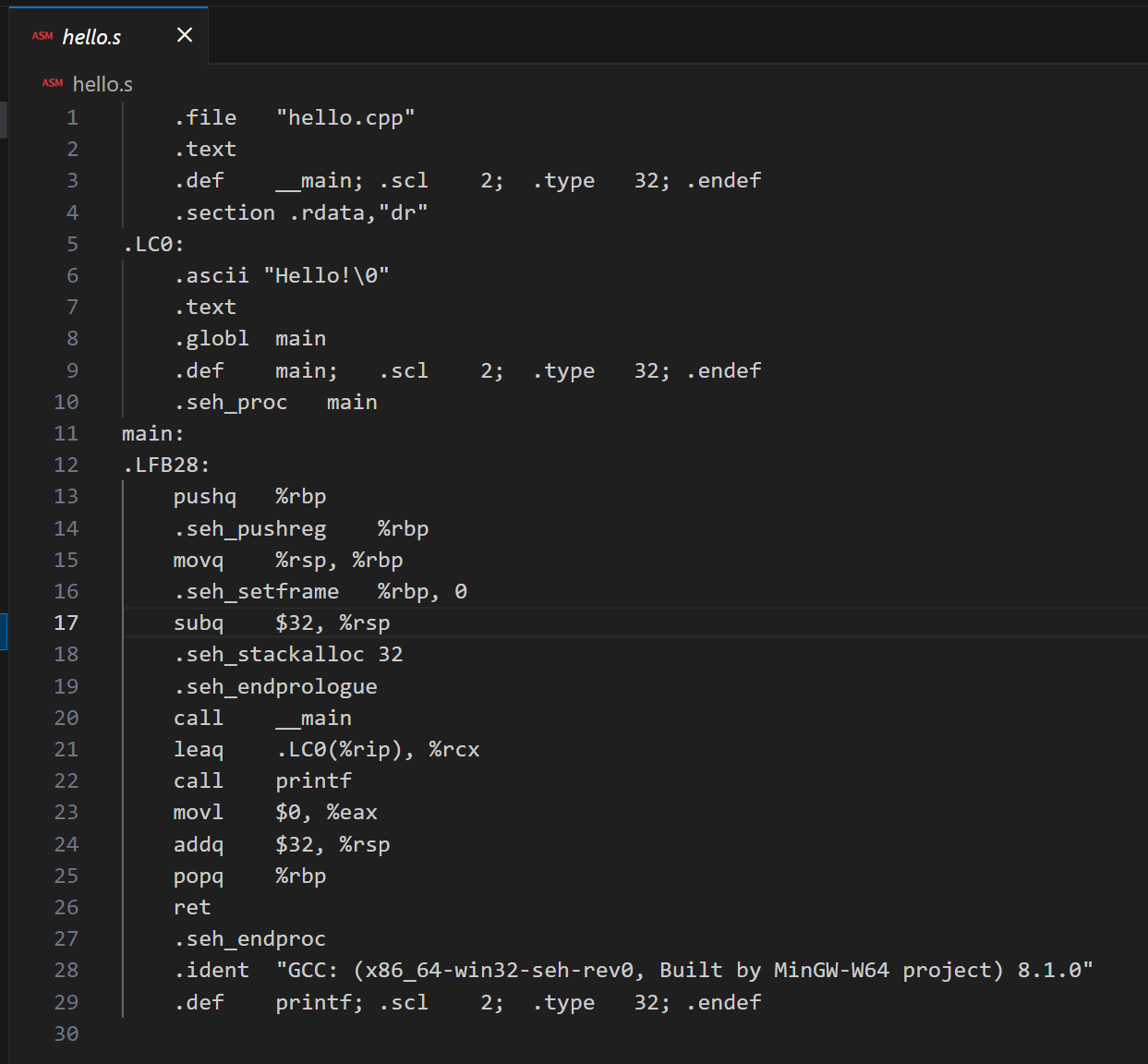
举个例子：输出“Hello！”的代码



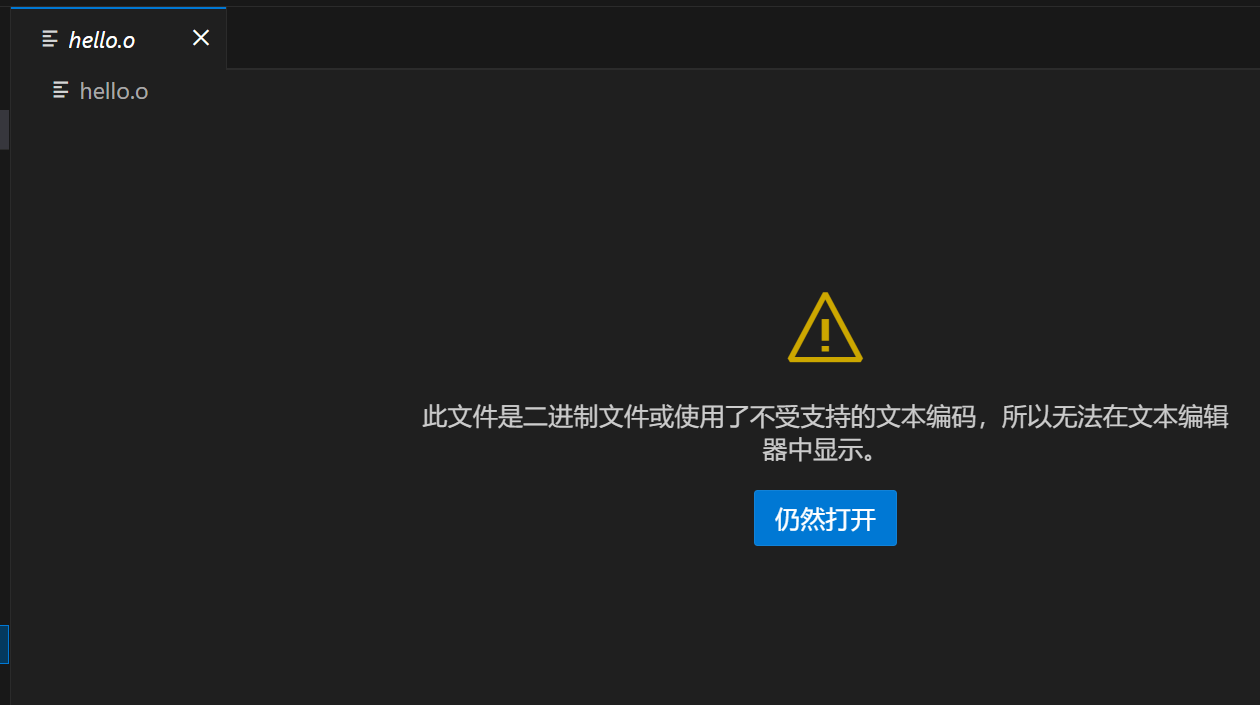
步骤一：g++ -E hello.cpp -o hello.i // 预处理



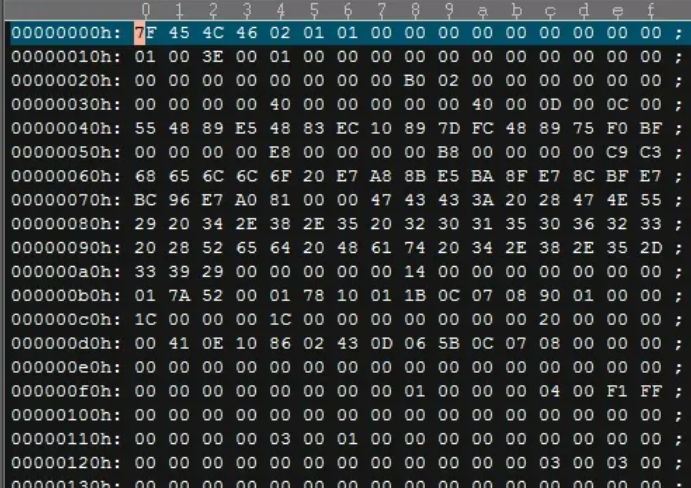
步骤二：g++ -S hello.i -o hello.s // 编译



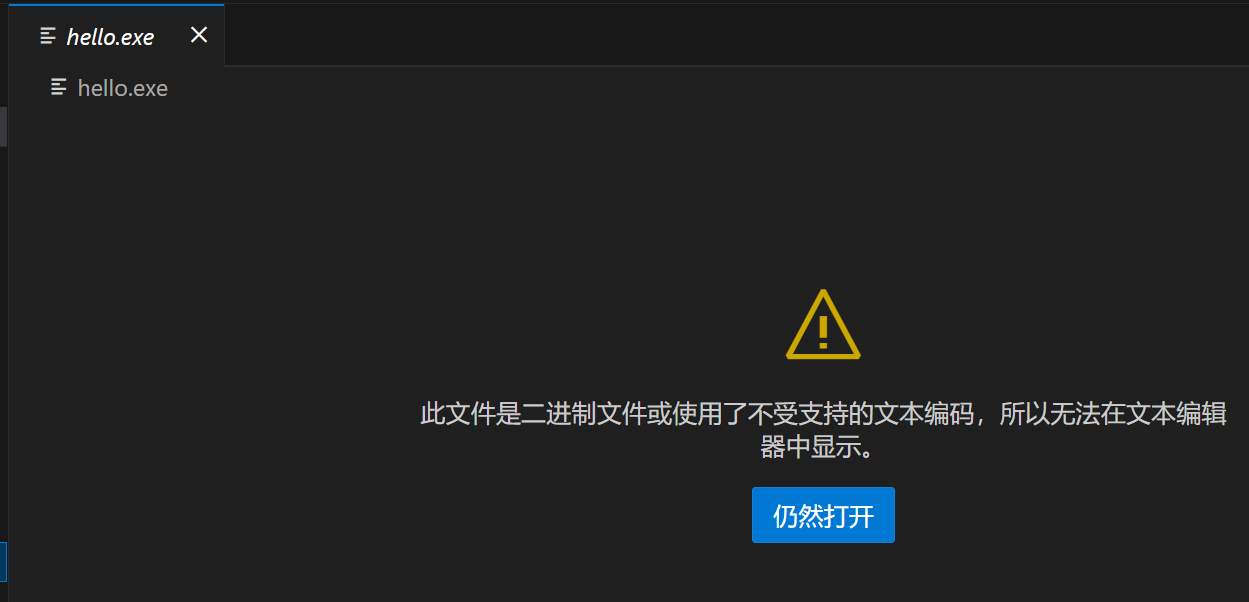
步骤三：g++ -c hello.s -o hello.o // 汇编



在VScode里无法打开，但这个文件的内容中包含的是机器码，类似于下图这种形式：



步骤四：g++ hello.o -o hello // 连接



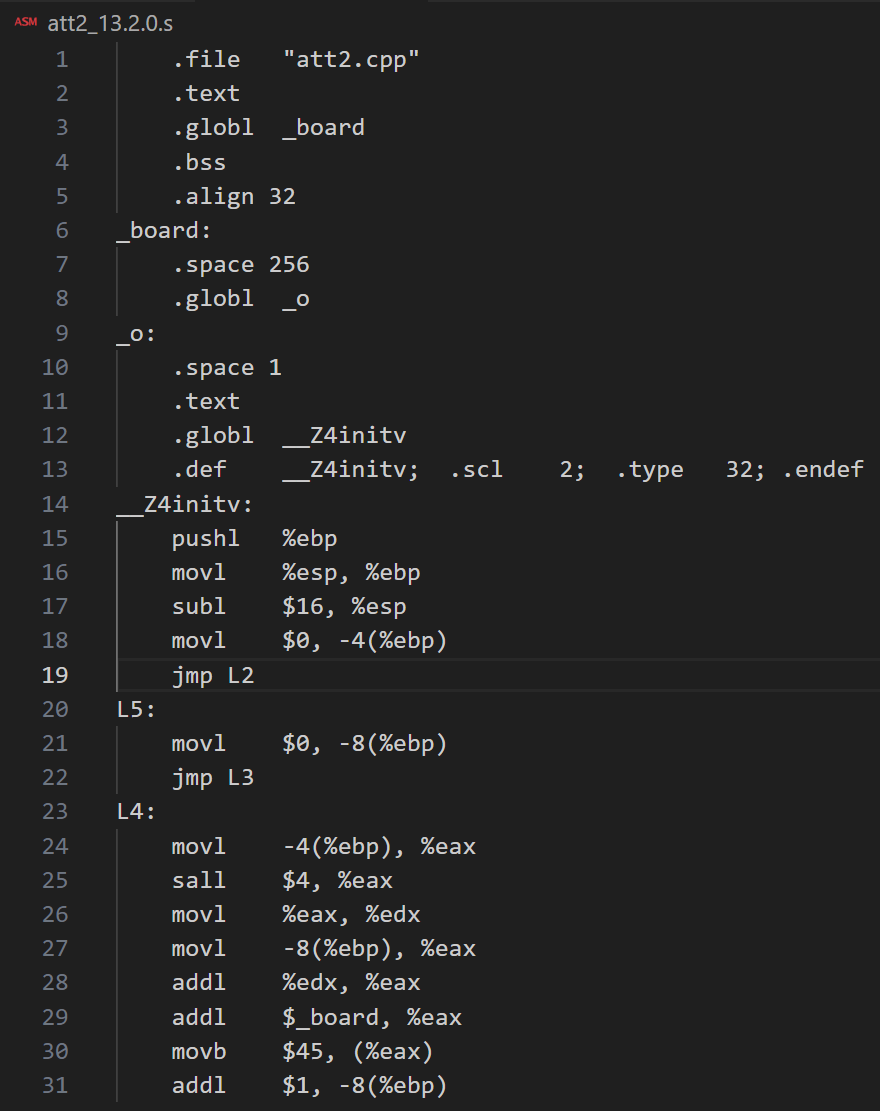
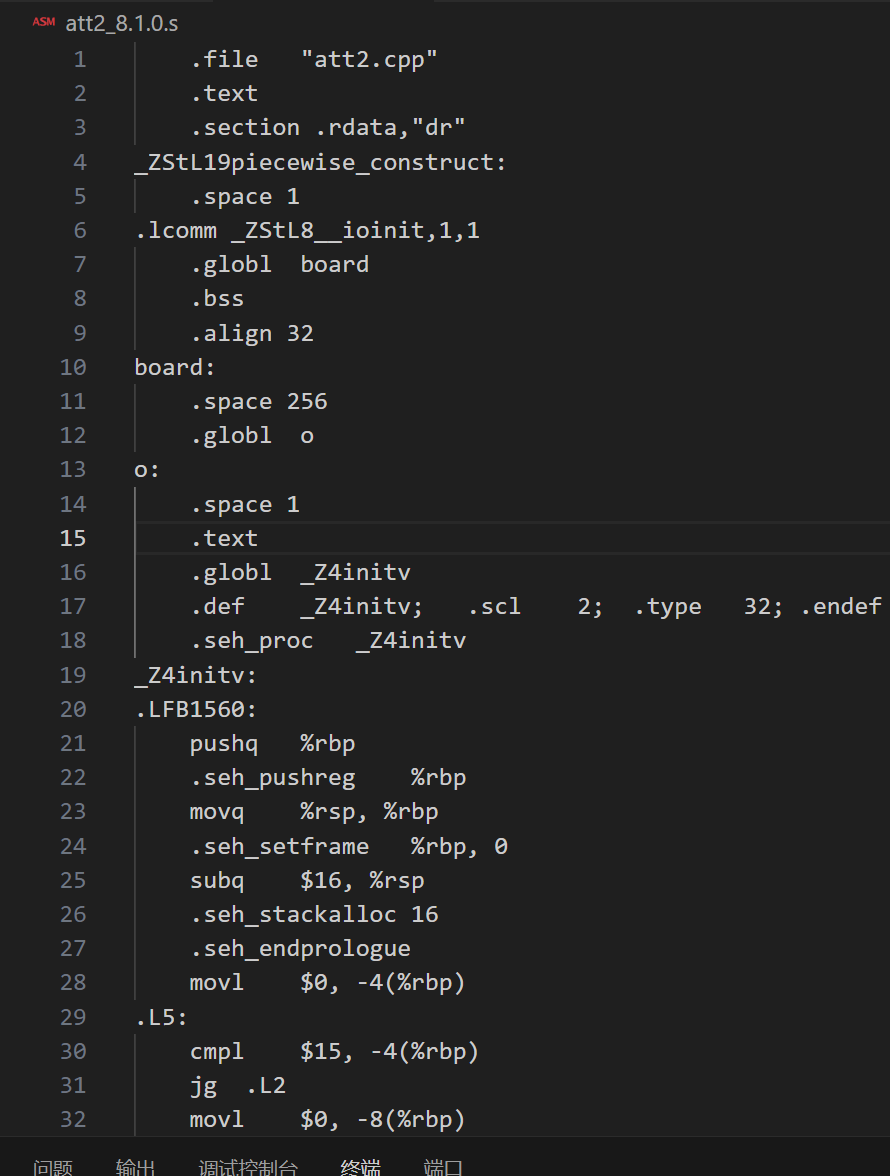
问题1 使用GCC中不同版本的C++编译器编译附件1中的程序源代码[2]，并对比使用默认编译选项时的编译结果。找出区分这些编译结果的主要特征。

我利用的是VScode软件，用g++ -v判断当前采用的GCC版本。

用g++ -S att1.cpp -o att1\_xx.x.x.s和 g++ -S att2.cpp -o att2.s命令生成对应文件

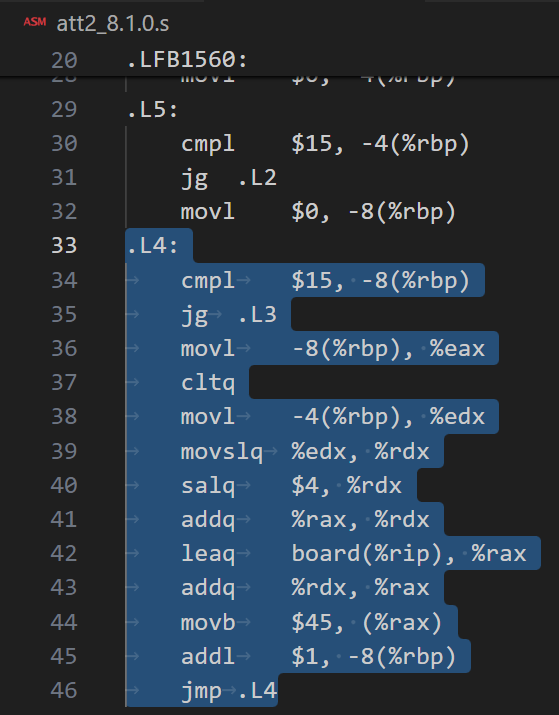
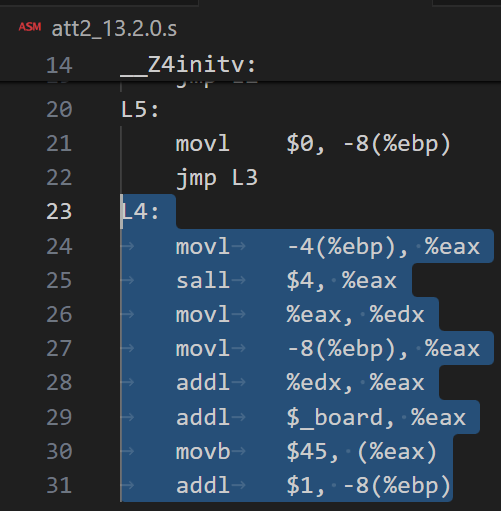
通过修改环境变量修改对应的GCC版本

展示8.1.0和13.2.0对应的结果：



通过观察上面两张图片，可以看到.s文件是按照代码的行来转换的，因此可以将不同版本的C++编译器相同行的不同之处作为区分编译结果的主要特征。

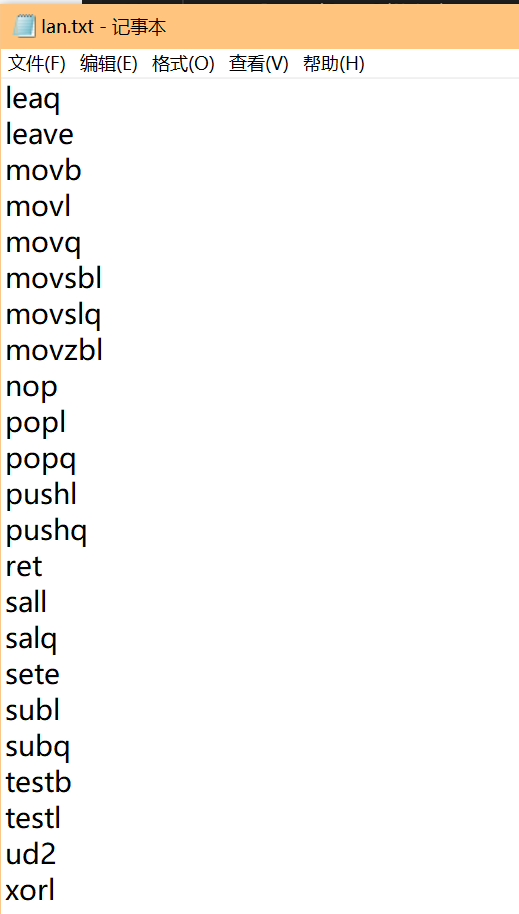
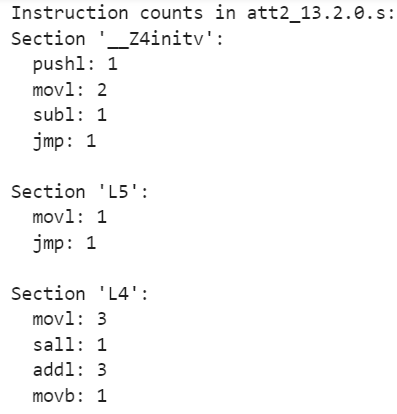
比如第4行：

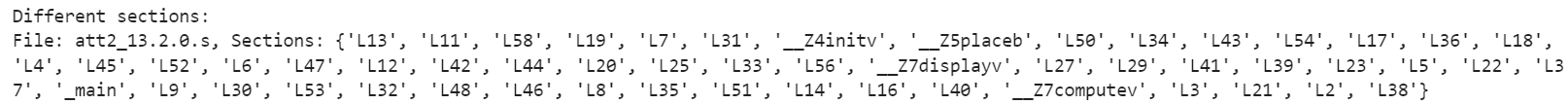
 

代码基本思想：

1. 构建命令文件，统计出现过的命令；
2. 每个部分以冒号开始，并且部分名称独占一行，随后是该部分的内容，直到遇到下一个冒号为止，统计该部分中对应的命令数量
3. 通过比较不同的部分确定主要特征

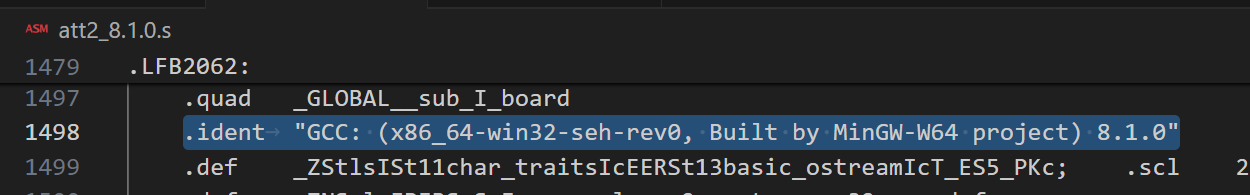
结果：

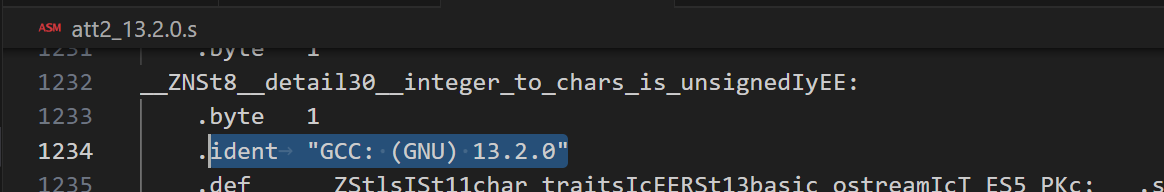
 



还可以采用其他方法确定主要特征，比如比较相同行的对应代码后面跟随的内容……

确定的有用主要特征越多，构建的判别函数就会越准确。关于编译后的内容有一个地方需要注意：





GCC8.1.0和GCC13.2.0编译的部分中其实包含了对应的GCC编译器版本，因为没有做所有版本的编译，不好确定是否每个版本都有提示，因此上文中确定的主要特征没有将这一情况考虑进去。

问题2 根据问题一中得到的特征，构造一个判别函数，使得能从各版本C++编译器使用默认编译选项时的编译结果，判别区分编译器版本。

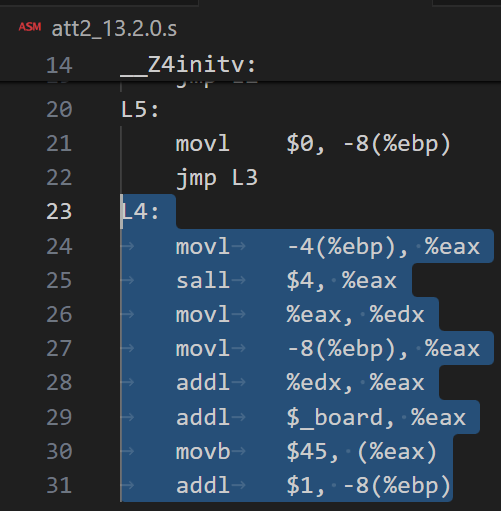
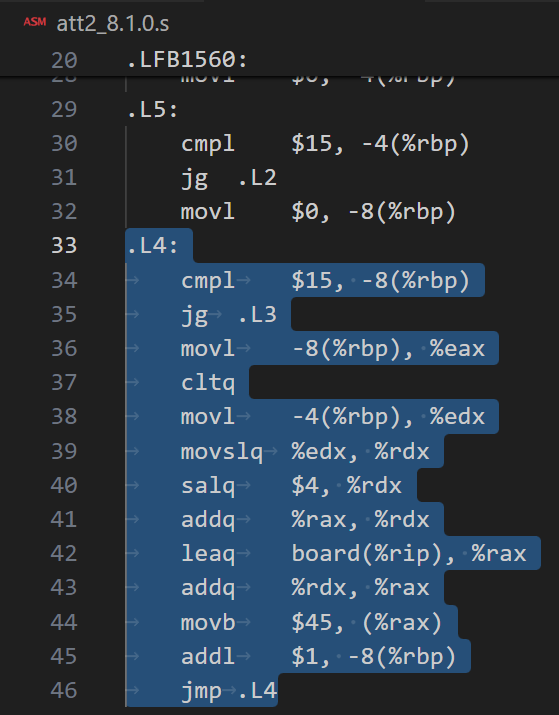
要根据得到的结果来构造一个判别函数进行分类，首先需要确定分类的依据。从问题一结果可以知道每个文件在指令部分和指令计数上的差异，可以根据这些差异来定义一个判别函数，该函数可以为每个文件分配一个类别。类别可以是基于不同的部分、不同的指令或两者的组合来定义的。

判别函数：根据是否存在不同的部分或指令计数来为每个文件分类。利用决策树算法进行划分。

问题3 用GCC中不同版本的C++编译器编译附件2中的源程序代码[3]，给出直接使用问题2中得到的判别函数区分编译器区版本的结果。

研究使用附件1、2原代码编译结果之一都能区分GCC中不同版本的C++编译器的判别函数。

相当于需要修改判别函数，更新判别规则，需要更多的或者更准确的主要特征，例如之前提到的比较相同行的对应代码后面跟随的内容：





从多个角度入手构建新的判别函数。

问题4 给出几条提高由编译结果区分编译器版本的判别函数性能的建议，包括区分度和对原代码的泛化性。

1. 特征选择：
   * 编译选项和标志：不同的编译器和版本可能支持不同的编译选项和标志。收集这些选项作为特征，可以提高区分度。
   * 警告和错误信息：编译器在编译过程中产生的警告和错误信息往往包含了关于编译器版本的线索。可以分析这些文本信息，提取关键词或模式作为特征。
   * 生成的代码特征：比较不同编译器版本生成的汇编代码或机器代码，寻找其中的差异作为特征。例如，指令集、优化级别、代码布局等。
2. 算法优化：
   * 使用机器学习算法：可以利用机器学习算法（如决策树、随机森林、神经网络等）来训练判别函数。这些算法可以自动学习从特征到编译器版本的映射关系。
   * 特征降维：如果特征数量过多，可以考虑使用主成分分析（PCA）、自编码器等技术进行特征降维，减少计算复杂度并提高泛化能力。
   * 参数调优：对于使用的机器学习算法，进行参数调优以找到最佳性能。这可以通过交叉验证、网格搜索等方法实现。
3. 数据增强：
   * 增加样本多样性：收集更多不同编译器版本和设置下的编译结果样本，以增加判别函数的泛化能力。
   * 数据扩充：通过对已有样本进行变换或扰动（如添加噪声、改变字体大小等），生成新的样本，以增加数据的丰富性。
4. 模型评估与反馈：
   * 评估指标：选择合适的评估指标（如准确率、召回率、F1分数等）来评估判别函数的性能。
   * 反馈循环：在实际应用中，不断收集新的编译结果数据，对判别函数进行迭代更新和优化，以提高其性能。
5. 考虑编译器特性：
   * 编译器特定的元信息：某些编译器可能在编译结果中包含特定的元信息，如版本字符串、时间戳等。提取这些信息可以显著提高区分度。
   * 编译器兼容性：考虑到不同编译器之间的兼容性问题，判别函数应尽可能避免依赖于特定编译器的特性，以提高泛化性。
6. 集成方法：
   * 结合多个判别函数：可以训练多个判别函数，并将它们的输出进行集成（加权平均等），以提高整体的性能和稳定性。