需求、可行性、测试（汇编器部分）

**需求分析**

在读取了汇编代码文件后，我们得到了文件的符号、代码等相关信息，此时需要根据这些信息生成对应的可重定位文件。

*（数据流图 汇编代码文件-汇编-可重定位文件）*

我们的目标是生成一个可重定位文件，故我们需要先分析可重定位文件的各个组成部分。首先可重定位文件的结构大致由ELF文件头（ELF Header）、节区头部（Section Header）列表、节区（Section）构成。其中，ELF文件头包含了关于该文件类型的一些信息，如魔数（magic number），还包含了这个文件的大致结构信息如节区头部表格的偏移等等；一个节区头部用于描述一个节区，而对于文件中所有节区的描述组成一个连续的节区头部列表，可以通过它找到每个节区的偏移、大小等信息；节区则是文件的主要内容所在，其中又有特殊节区具有特殊的功能，例如“.text”节区中存放着机器码、“.data”节区中存放着已初始化数据等。

*（可重定位文件结构图）*

我们要生成可重定位文件，需要先生成它的各个组成部分，然后把各个部分组合成一个文件。ELF文件头中部分内容是固定的，例如魔数、文件类型等，也有部分内容是需要根据实际情况填写的，例如节区头部列表的偏移、节区名字表的偏移等。故我们需要先生成节区和节区头部，再根据实际情况填写ELF头部。而在生成节区头部又是用于描述一个节区的，所以需要先生成它对应的节区。

根据这种拓扑关系，我们可以进一步细化数据流图如下：

*（数据流图）*

其中最重要的又是汇编过程：由汇编代码翻译得到机器指令。这一步将在生成“.text“节区中完成，这也是汇编器的核心功能。为了实现这个需求，我们需要建立一个ARM汇编到机器指令的映射。

这样，我们就能完成汇编过程，得到可重定位文件，等待链接器的进一步处理。

**可行性**

经济可行性、社会可行性和马玏写的可以合并。

技术可行性：

本次项目中实现的汇编器，总体上看是完成了由汇编代码文件到目标文件的转换，其核心是能将ARM汇编语言转换为能在树莓派上运行的机器码。在前一个模块读取和分析汇编代码文件后，我们可以根据其输出的信息来生成ELF文件的各个节区内容，再进一步根据各个节区的内容来生成节区头部和ELF头部。

要实现这种生成过程，我们可以先定义好各个节区、节区头部、ELF头部、ELF文件对应的结构体，然后正确填写这些结构体里的成员，最后把结构体按合理的顺序输出到一个文件。这些结构体有的是“elf.h”中自带的，也有的是我们为了更方便地将文件各个部分有机组织起来而自己定义的。

例如对于“.data”节区，我们可以根据前一个模块输出的“data\_element\_list”获取需要的信息，再按照ELF文件所要求的格式填写到对应的结构体即可。

对于生成“.text”、“.reloc”、“.sym”节区，同理去填写对应的结构体。只不过要生成它们，依赖的是上一步读取模块的其他输出，分别是“arm\_assem\_list”、“reloc\_symbol\_list”、“symbol\_list”。

对于节区名字表、符号名字表等，需要在生成节区的过程中填写信息。

对于节区头部列表和ELF头部，把前面生成的所有内容按一定顺序组织起来，按照ELF文件格式的要求填写即可。

对于生成ELF文件的格式要求，可以查阅ELF文档，也可以对照gcc或clang生成的文件；对于ARM转换到机器码的相关规则，则可以通过ARM文档查阅。即我们可以获取关于生成目标文件所需要的相关标准。

综上，在本次项目中实现汇编器模块具有技术可行性。

**测试**

对于本项目的汇编器部分，由于ARM汇编代码本身较为庞大，一类指令中常常包含较多的处理各种字长的不同指令。而我们在实现汇编器的过程中只选取了其中高频用到的指令，即是原本ARM汇编的一个子集。故此处我们采取手动测试，将测试用例的输入限制于我们定义的ARM汇编子集内，观察以下测试用例的输出文件，并且采用ELF Reader将其与gcc输出进行对比，以直观地验证汇编器功能的正确性。

样例列表：