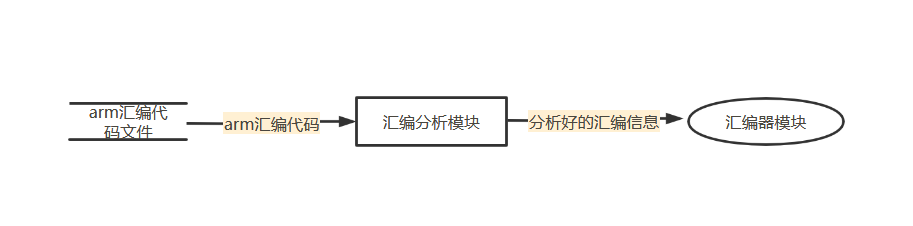
# 需求分析与复杂工程问题归纳

## 需求分析

首先我们讨论汇编分析模块的功能性需求，汇编分析模块总体功能是读取分析汇编代码得到生成目标文件所需的关键信息，并将这些信息保存在合适的数据结构中提供给汇编器模块，使得汇编器模块根据这些信息可快速构建目标文件。因此模块的输入为以.s为后缀的arm汇编代码文件，向后续模块输出分析得到的储存汇编代码关键信息的数据结构。



此功能性需求的核心目标是汇编器模块仅根据此模块产生的数据结构，并且做最少的信息处理就可以构建出完整的目标文件。

这里根据核心功能需求就引出汇编分析模块的几个设计问题，读取汇编代码后需要分析哪些信息？设计什么样的数据结构来保存这些信息？如何得到所需要的信息？对这三个问题的归纳分析也就能的得到该模块的详细需求设计。

问题1：需要分析什么信息？

ELF文件的涉及程序的关键节区有.text节区，.data节区，.bss节区和.reloc\_text节区，这些节区的内容需要从汇编代码中获得，而ELF文件其他节区的内容都可以从这些节区运算构建得到，因此汇编分析模块主要想后续模块提供的内容就是.text .data .bss .reloc\_text这几个节区中的内容，即汇编代码，全局已初始化和为初始化的变量，代码中需要重定位的符号，位置等信息。

问题2：怎么保存这些信息？

根据问题1的分析将所需的信息划分为5类，为每一类信息设计一个结构体以对信息进行保存，如汇编代码这类信息需要保存他的操作符和操作数，全局变量需要保存他的变量名和对应的值，重定位信息需要保存重定位处的类型，比如是全局变量的重定位还是函数的重定位，重定位的变量名和重定位的位置。根据如上所需内容设计5个结构体和其相应内容。然后将生成的结构体放入链表中传递给汇编器模块。

问题3：如何得到所需要的信息？

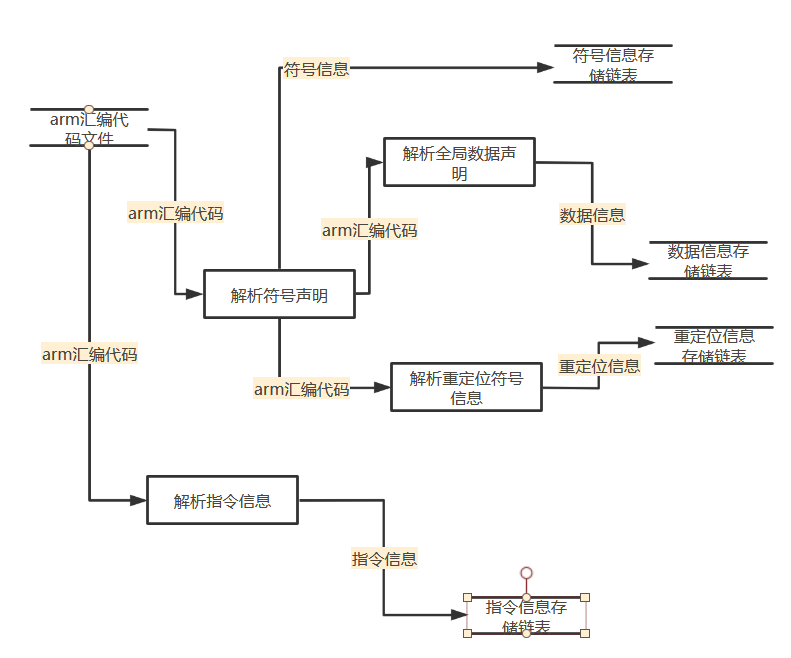
读入的汇编代码中包含伪指令，因此汇编分析模块需要对伪指令进行处理，还要将一些汇编表示与二进制表示不同的地方进行处理，比如跳转语句在汇编代码中只有标号，而处理时需要汇编分析模块计算跳转的位移。汇编分析模块线性读取汇编代码进行分析，读到不同的语句后做不同的逻辑处理。这里就需要对汇编代码进行分类，对不同类型的代码做不同的逻辑动作，生成不同的信息结构体保存在链表中。

对汇编代码的分类可大致分为全局声明类（如.globl伪指令），符号类型声明类.（如.type伪指令），数据声明类（如.word/.space伪指令），以及可与二进制代码对应的正常指令。

根据这些分类生成不同的信息结构体，就可以得到后续汇编器构建目标文件的全部信息，且汇编器模块可以很方便得访问这些数据结构。

根据上述问题归纳可以的到模块的详细需求：线性读取汇编代码，根据设计的数据结构分析代码中的符号，变量，指令，重定位符号等信息并分别产生对应的结构体链表。

详细数据流图如下：



## 可行性分析

#### 经济可行性

可与马玏结合

#### 社会可行性

可与马玏结合

#### 技术可行性

汇编分析模块需要向汇编器模块提供保存汇编代码信息的数据结构，使得汇编器模块可方便快速的构建目标文件，汇编分析模块需要将诸如符号，指令，全局变量等信息从汇编代码中分离，并且需要将一些如偏移，伪指令等需要处理的信息进行处理。

这些信息通过给出的汇编代码都可以分析得到，且设计的数据结构是通过ELF文件的节区构成进行信息数据结构设计，因此分析出的数据结构满足汇编器模块构建ELF文件的需要，因此该模块在技术上可行。

# 测试环境构建与测试驱动开发

汇编分析器测试模块将树莓派上得到的汇编代码作为测试用例，测试环境在ubuntu上运行，使用c++开发单元测试驱动，将数据结构中的内容输出作比对。

单元测试驱动采用输出五个结构体链表中的内容并与预期进行比对的方式进行单元测试。

比如测试符号表信息结构体的测试代码如下:

ARM\_analyze \*\_arm\_analyze = new ARM\_analyze(arm\_assemble);

    cout << "symbol\_list:\n";

    for (int i = 0; i < ARM\_analyze::symbol\_list.size(); i++)

    {

        cout << "\tbind:" << ARM\_analyze::symbol\_list[i]->bind << "\t";

        cout << "\tdefined:" << ARM\_analyze::symbol\_list[i]->defined << "\t";

        cout << "\tname:" << ARM\_analyze::symbol\_list[i]->name << "\t";

        cout << "\ttype:" << ARM\_analyze::symbol\_list[i]->type << "\t";

        cout << "\tvalue:" << ARM\_analyze::symbol\_list[i]->value << "\t";

        cout << "\n";

    }

对其他几个结构体链表的测试类似。

现选取一个测试用例arm.s该用例是在树莓派平台上对c语言编译得到的arm汇编代码：

.arch armv7-a

    .eabi\_attribute 28, 1

    .eabi\_attribute 20, 1

    .eabi\_attribute 21, 1

    .eabi\_attribute 23, 3

    .eabi\_attribute 24, 1

    .eabi\_attribute 25, 1

    .eabi\_attribute 26, 2

    .eabi\_attribute 30, 6

.eabi\_attribute 34, 1

    .eabi\_attribute 18, 4

    .data

    .globl a

    .type    a, %object

a:

    .word  1

    .word  1

    .space  392

    .globl b

    .type    b, %object

b:

    .word  2

    .word  1

    .space  392

    .text

    .globl main

    .syntax unified

    .type    main, %function

main:

    push  {r4, r5, fp, lr}

    sub    sp, sp, #48

    add    fp, sp, #0

    mov  r0, #0

    mov  r5,  r0

    mov r0, #2

    ldr r0, =b

    mov  r5,  r0

    str     r5, [fp, #4]

.L1:

    ldr r0, [fp, #4]

    mov r1, #0

    cmp r0,r1

    ble .L2

    ldr r0, [fp, #4]

    mov r1, #1

    sub r0, r0, r1

    str r0, [fp, #4]

    b     .L1

.L2:

    mov r0, #0

    bl putint()

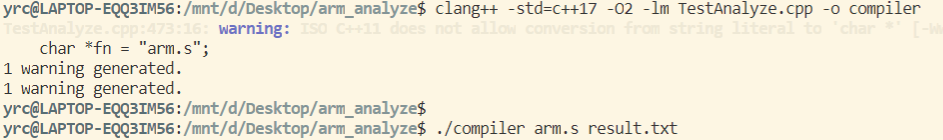
    mov r0, r0

    mov sp, fp

    add  sp, sp, #48

    pop  {r4, r5, fp, pc}

使用单元测试驱动分析该汇编代码测试用例:



分析结果如下：

symbol\_list:

bind:0 defined:1 name:a type:1 value:0

bind:0 defined:1 name:b type:1 value:400

bind:0 defined:1 name:main type:0 value:0

bind:0 defined:1 name:.L1 type:3 value:40

bind:0 defined:1 name:.L2 type:3 value:96

bind:0 defined:0 name:putint type:-1 value:0

reloc\_symbol\_list:

name:b type:1 value:28

name:putint type:0 value:100

arm\_assem\_list:

op\_name:push$$$ Operands1:$$$ Operands2:$$$ Operands3:$$$ Reg\_list:r4$$$ r5$$$ fp$$$ lr$$$

op\_name:sub$$$ Operands1:sp$$$ Operands2:sp$$$ Operands3:#48$$$ Reg\_list:

op\_name:add$$$ Operands1:fp$$$ Operands2:sp$$$ Operands3:#0$$$ Reg\_list:

op\_name:mov$$$ Operands1:r0$$$ Operands2:#0$$$ Operands3:$$$ Reg\_list:

op\_name:mov$$$ Operands1:r5$$$ Operands2:r0$$$ Operands3:$$$ Reg\_list:

op\_name:mov$$$ Operands1:r0$$$ Operands2:#2$$$ Operands3:$$$ Reg\_list:

op\_name:ldr$$$ Operands1:r0$$$ Operands2:pc$$$ Operands3:#-4$$$ Reg\_list:

op\_name:nop$$$ Operands1:$$$ Operands2:$$$ Operands3:$$$ Reg\_list:

op\_name:mov$$$ Operands1:r5$$$ Operands2:r0$$$ Operands3:$$$ Reg\_list:

op\_name:str$$$ Operands1:r5$$$ Operands2:fp$$$ Operands3:#4$$$ Reg\_list:

op\_name:ldr$$$ Operands1:r0$$$ Operands2:fp$$$ Operands3:#4$$$ Reg\_list:

op\_name:mov$$$ Operands1:r1$$$ Operands2:#0$$$ Operands3:$$$ Reg\_list:

op\_name:cmp$$$ Operands1:r0$$$ Operands2:r1$$$ Operands3:$$$ Reg\_list:

op\_name:ble$$$ Operands1:#44$$$ Operands2:$$$ Operands3:$$$ Reg\_list:

op\_name:bge$$$ Operands1:#40$$$ Operands2:$$$ Operands3:$$$ Reg\_list:

op\_name:blt$$$ Operands1:#36$$$ Operands2:$$$ Operands3:$$$ Reg\_list:

op\_name:bne$$$ Operands1:#32$$$ Operands2:$$$ Operands3:$$$ Reg\_list:

op\_name:beq$$$ Operands1:#28$$$ Operands2:$$$ Operands3:$$$ Reg\_list:

op\_name:bgt$$$ Operands1:#24$$$ Operands2:$$$ Operands3:$$$ Reg\_list:

op\_name:ldr$$$ Operands1:r0$$$ Operands2:fp$$$ Operands3:#4$$$ Reg\_list:

op\_name:mov$$$ Operands1:r1$$$ Operands2:#1$$$ Operands3:$$$ Reg\_list:

op\_name:sub$$$ Operands1:r0$$$ Operands2:r0$$$ Operands3:r1$$$ Reg\_list:

op\_name:str$$$ Operands1:r0$$$ Operands2:fp$$$ Operands3:#4$$$ Reg\_list:

op\_name:b$$$ Operands1:#-52$$$ Operands2:$$$ Operands3:$$$ Reg\_list:

op\_name:mov$$$ Operands1:r0$$$ Operands2:#0$$$ Operands3:$$$ Reg\_list:

op\_name:bl$$$ Operands1:0$$$ Operands2:$$$ Operands3:$$$ Reg\_list:

op\_name:mov$$$ Operands1:r0$$$ Operands2:r0$$$ Operands3:$$$ Reg\_list:

op\_name:mov$$$ Operands1:sp$$$ Operands2:fp$$$ Operands3:$$$ Reg\_list:

op\_name:add$$$ Operands1:sp$$$ Operands2:sp$$$ Operands3:#48$$$ Reg\_list:

op\_name:pop$$$ Operands1:$$$ Operands2:$$$ Operands3:$$$ Reg\_list:r4$$$ r5$$$ fp$$$ pc$$$

data\_element\_list:

name:word value:1

name:word value:1

name:space value:392

name:word value:2

name:word value:1

name:space value:392

bss\_element\_list:

可见分析到了相关所需的信息。汇编器模块根据这些信息可以快速的构建目标文件。