1.文件结构

新增文件：

spim.h

spim.c 将中间代码翻译为目标代码

2.寄存器分配与栈管理

寄存器分配采用朴素寄存器分配的方法，采用到了$a0传递函数调用的值

采用$t0与$t1进行传递。

其中中间代码中的临时变量，变量，地址都在栈上进行空间的分配

栈的管理与实验手册基本一致，记录$ra与$fp等内容

3.实验过程

1.对于中间代码中的所有变量，地址，临时变量都求出他需要的栈的空间与传递的参数的大小。并且求出函数所有的变量空间，为每一个变量在栈中找到相对于$fp的偏移量

2.根据前一步的偏移量与中间代码，生成目标代码

对于传递的参数，如果其对于$fp的偏移量为正，则在预处理过程中将其偏移量设置为负数

对于一般的算数运算处理逻辑是将用到的变量存到寄存器中，再去对寄存器进行对应的操作再写回栈中，写回栈中的位置是根据$fp加上偏移量来确定对应的地址

4.函数调用时寄存器，栈的处理方法

在处理参数arg时，不断地将$sp-4，并且将参数写入对应的内存，之后call到相应的地址即可

函数开始的阶段，需要先存起来$fp和$ra的值，然后把$fp置成$sp，最后把$sp - 函数中所有变量的offset总和。

Return与call的过程相反，先把$a0置为返回值，再恢复$fp和$ra，最后删除被调用者的栈。

5.问题

在对结构体数组与一维数组的处理中出现来offset求解的错误，需要进行特殊处理

不同的中间代码之间生成不同的目标代码，需要仔细地一步步比较