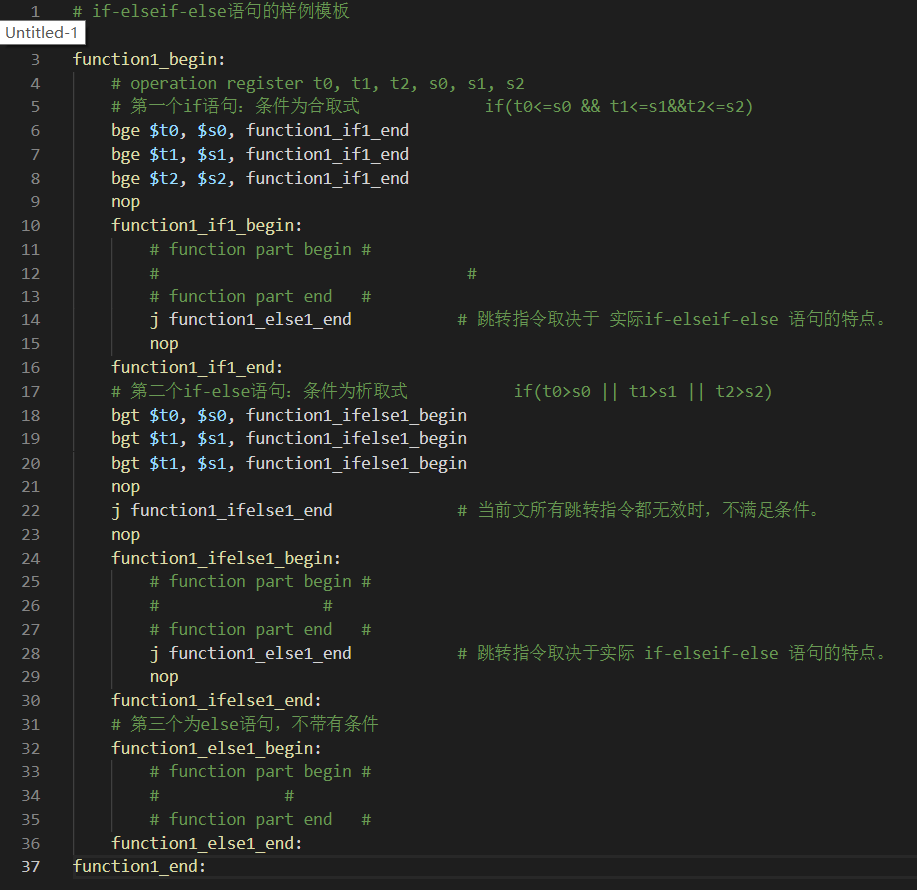
P2课下作业个人小结

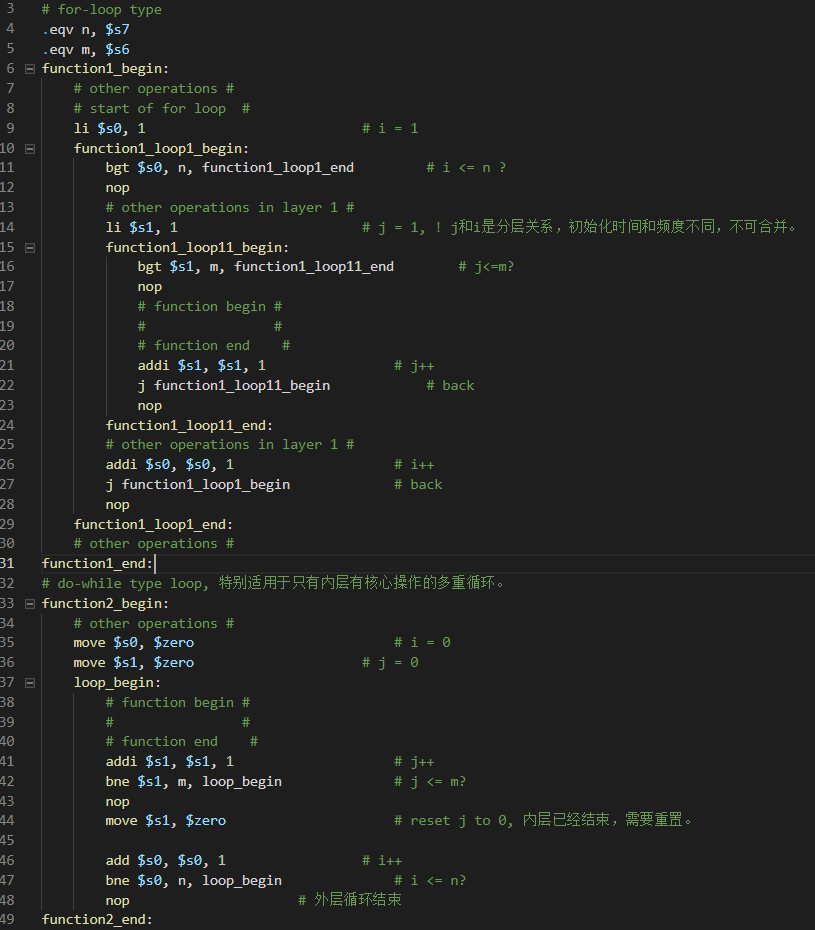
1. 写mips汇编程序的基本步骤。
   1. 仔细分析题干，明确任务。
   2. 书写C语言代码，用适宜于mips编写的思路进行书写。（**节约工作量重要因素**）
   3. 汇编程序编写：全局注释-内存空间、宏的定义-主函数与函数
   4. 静态查错（**重要**，在时间允许情况下一定要做。）
2. C语言代码编写规范。
   1. **第一要素：简洁** = 更少汇编代码 = 更少耗时、更少问题。
   2. 不轻易将“顺序功能”写在函数中。（汇编进行函数调用时花费大量前后期任务）。
   3. 子函数：不在途中return，易造成栈操作出现遗漏。
   4. 条件语句：框架清晰规整，条件至简。
3. 节省代码行数的技巧与方法。
   1. 根本问题：C语言参考模板的代码是否简洁？
   2. 分支技巧：
      1. Branch类指令参与不等式比较分支，不适用set+beqz方式跳转。
      2. 基础型的操作使用macro封装：push、pop（注意**栈取放反向**）和input。
      3. 递归函数中多次调用时首选-用s寄存器存需要保存的值（否则栈维护会很麻烦）。
      4. R型ALU运算指令一般为3寄存器，不要只将其当做2寄存器使用。
4. 矩阵取值、赋值、运算时的要点与注意事项。
   1. 伪指令.word value:space 可定义初始值不同的字宽空间。.space指令切记要在期望空间上4倍大小。
   2. 二维数据存放技巧：空间大小开成2^n \* 2^n，在计算地址的行跳跃时，减少因为错乘“行的个数”（正确的因为列宽）而造成的错误。
   3. SW和LW的基址寻址法：lw/sw $t0, label($t1) V.S. 寄存器寻址 lw/sw $t0, offset($t1)。
5. 常用的伪指令。
   1. .eqv : 更改寄存器名字以方便识别。建议：用于全局变量和重复高频变量中（如i，j，k等），并主要以s寄存器为主（及少量高码t寄存器）。
   2. .macro：封装建议操作函数，但**封装不佳易造成冲突篡改**（部分问题采用函数调用时的栈维护不可避免）。
      1. 常用：input\_int, output\_char/space/enter, push, pop.
      2. 个性化函数：建议变量声明时采用：被改变寄存器+中间t寄存器的方式。在开头和结尾用push和pop把中间寄存器内容备份，在进行操作。**（此方法可完全避免篡改问题）。**
6. If-elseif-else语句模板。

每次在搭建条件语句模块时，**至少应该将一种条件操作框架搭建好再填充内容**。

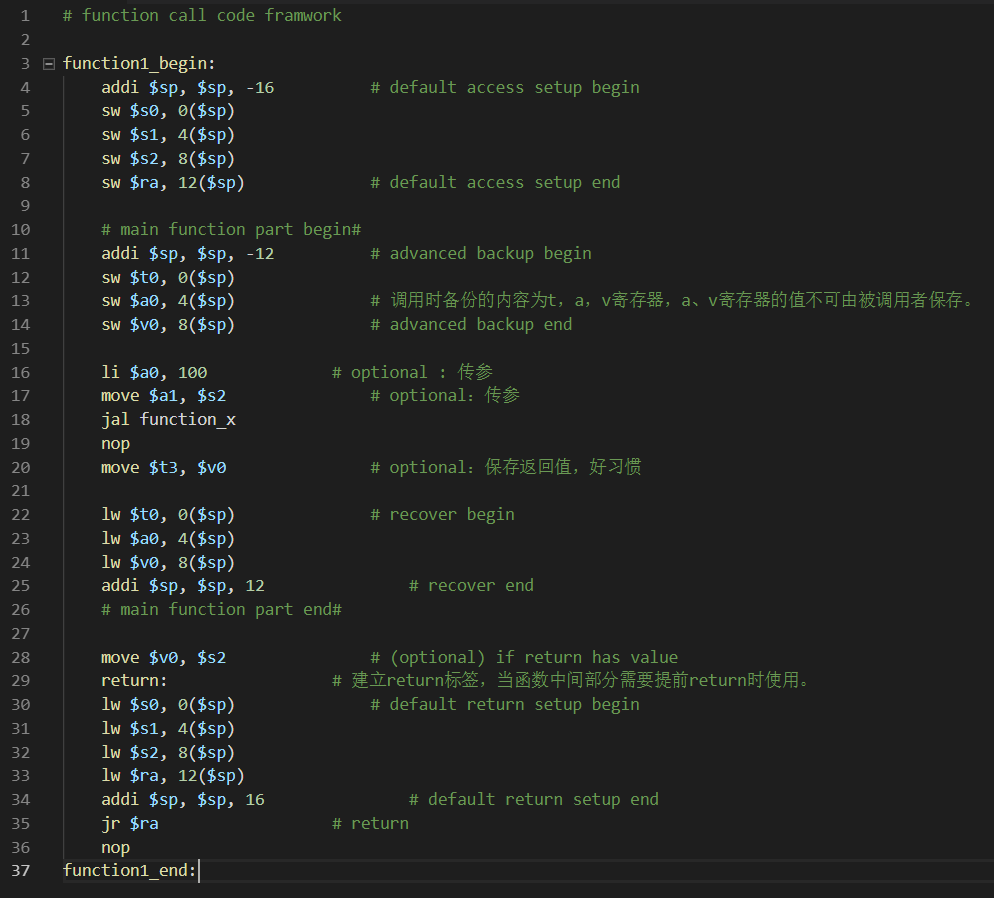


1. 循环语句模板。

每次在搭建循环语句模块时，**应将头尾（包括多层循环）都搭建好再填充内容**。



1. 调用函数模板。



1. Debug技巧与方法。
   1. 静态查错：运行前静态查错-运行错误定位更细致后静态查错。
   2. 动态查错：借助**断点/错误点**，定位错误的代码位置，结合寄存器、内存中的值，找到错误的地方-进行静态查错。
   3. 相似功能复制代码后重点查错位置：**标签名、循环变量赋值、数组名、循环界限**……
   4. 常见的bugs：
      1. S寄存器被误写成t寄存器**（静态查错时一定要查出来）**。
      2. Add和and书写失误。
      3. 栈维护变量时，采用push和pop时弄反了。
      4. 递归调用时，中途return未进行变量复位。
      5. 矩阵地址运算时，行运算应乘上列数（不是行数，此方法不建议采用。）
      6. 输出和输入时，缺少syscall指令造成漏入漏出。