ステージ４

寄存器分配：

我们采用了一种类似于内存运行方式的方法来进行寄存器分配，具体来说，每次需要使用变量时，我们会进行如下操作：

1. 如果这个变量在寄存器中，直接使用其在寄存器中的地址
2. 如果这个变量不在寄存器中，且寄存器中有空余的位置，我们会将寄存器分配在该空余位子
3. 如果这个变量不在寄存器中，且寄存器中没有空余位置，我们会找到最久没有被使用的存在于寄存器的变量，将其存回储存并将空余的位置分配给被请求的变量

但因为mips代码生成顺序和实际运行顺序有所区别，因此对于if语句这种会产生分支的语句，我们需要进行额外的处理：当遇到if语句时，我们会将寄存器中接下来可能会用到的所有变量存回储存，以规避可能出现的问题。

Register Allocation:

We employ a method similar to the way memory operates for register allocation. Specifically, the following steps are taken whenever a variable needs to be used:

1. If the variable is in a register, use its address directly from the register.
2. If the variable is not in a register, and there is an available space in the registers, allocate a register to the variable.
3. If the variable is not in a register, and there is no available space in the registers, find the least recently used variable in the registers, store it back in memory, and allocate the available space to the requested variable.

However, due to the difference between the MIPS code generation order and the actual execution order, extra handling is required for statements like if statements that introduce branching. When encountering an if statement, we store back into memory all variables that may be used in the registers shortly after, in order to mitigate potential issues.