代码优化设计

就是对课上理论指导的实践，应该很费精力的，本人垃圾的时间管理导致写完代码生成就很极限了，没时间优化了；只在生成代码时进行了很小部分的优化

静态数值计算

对于数字，可以静态的进行相关计算，减少相关代码的生成

对于Const的symbol，可通过将数值存入之前设计的符号表中，在parseLVal时检查是否有确定values（这点在数组上也十分有用）；在parseExp相关的递归子函数中

对于UnaryOp，不要一锅端的在底层处理；应在在向上传递时视情况而处理，例如负负得正之类的，又如关于`!`，连续多个的情况，会出现冗余，可以在中间代码处优化（例子：`!-a+1 || b`）

无效运算删去

得通过窥孔优化实现，最好新建一个Optimizer操作中间代码进行，本人没时间了只是想想

```
MIN @t4 b#2 9
MUL @t4 @t4 3
ADD @t4 0 @t4
ADD @t4 @t4 0   // 来自某段数组offset计算
```

关于寄存器分配

其实这个很重要。有相关的SSA、图着色等理论，没空管了qwq。

倒是一开始为了减少对全局变量lw操作，直接将全局变量分配在了.data段上，通过标签和`.word`来定位；但如此一来，虽然加载指令lw/sw可以直接使用标签看上去不错，但其一条指令要翻译成三条基本指令，最后该是可能是负优化

指令选择

除法（尤其是%，纯数（const value）的话可以优化），耗时长的指令尽量不选

关于编译器代码本身的优化：

- 看看有没有可以改为emplace_back()的vector容器操作
- 视情况，为每个类补充“拷贝构造函数”和”移动构造函数“
- 将.h的具体函数代码移至.cpp中
- 梳理头文件引用，现在是随便引