1. 假设可用的寄存器为 R0 和 R1，且所有临时单元都有是非活跃的，试对以下四元式基本块：

T1:=B-C

T2:=A*T1

T3:=D+1

T4:=E-F

T5:=T3*T4

W=T2/T5

用简单代码生成算法生成其目标代码。

解题思路：

简单代码生成算法是依次对四元式进行翻译。不妨先考虑下面单一的四元式的翻译过程：

T:=a+b

由于汇编语言的加法指令代码形式为：

ADD R, X

其中 ADD 表示为加法指令，R 为第一个操作数，且第一个操作数必须为寄存器类型，X 为第二个操作数，它可以是寄存器类型，也可以是内存型的变量类型。该指令的意义为，将第一个操作数 R 与第二个操作数 X 进行相加，再将累加结果存回到第一个操作数的寄存器 R 中。所以，要完整地翻译出四元式 T:=a+b，则可能要如下的三条汇编指令：

LD R, a

ADD R, b

ST R, T

上面三条指令的含义是：第一条指令是第一个操作数 b 从内存装入一个寄存器 R 中，第二条指令进行加法运算；第三条指令再将第二条指令的累加结果写回到内存中的变量 T 中。

那么，是不是一个四元翻译成目标代码时都有要生成三条汇指令呢？如果考虑到目标代码生成过程中的两优化问题，则答案是否定的。即，为了使生成的目标代码更短和充分利用计算机的寄存器，上面的三条指令中，第一条指令和第三条指令在某些情形下可能不是必需的。这是因为，如果在翻译此四元式之前，已经有指令将操作数 a 放在某个寄存器中，则第一条指令可以省略；同样，如果下一个四元式紧接着要引用操作数 T，则第三指令不必急于执行，可延迟到以后适当的时机再执行。

更进一步，如果必须要用到第一条指令（第一个操作数只在内存中而不在寄存器中）则此时所有寄存器全部分配完毕，该如何处理。此时要根据寄存器中所有变量的待用信息（也就是引用点）来决定淘汰一个寄存器给当前的四元式使用。寄存器的淘汰策略如下：

- 如果有一个寄存器中的变量已经没有后续的引用点，且该变量是非活跃的，则可直接将该寄存器作为空闲寄存器使用；
- 如果所有寄存器中变量在基本内都还有引用点，且都是活跃的，则将引用点最远的变量所占用寄存器中的结果存回到内存相应的变量中，再将该寄存器分配给当前的指令使用。

解答：

该基本块的目标代码如下（指令后面为相应注释）：

LD R0, B //取第一个空闲寄存器 R0

SUB R0, C //运算结束后 R0 中为 T1 结果，内存中无该结果

LD R1, A //取一个空闲寄存器 R1

MUL R1，R0 //运算结束后 R1 中为 T2 结果，内存中无该结果

LD R0，D //此时 R0 中结果 T1 已经没有引用点，且临时单元 T1 是非活跃的，所以，寄存器 R0 可作为空闲寄存器使用。

ADD R0,"1" //运算结束后 R0 中为 T3 结果，内存中无该结果

ST R1，T2 //翻译四元式 T4:=E-F 时，所有寄存器已经分配完毕，寄存器 R0 中存的 T3 和寄存器 R1 存的 T2 都是有用的。由于 T2 的下一个引用点较 T3 的下一个引用点更远，所以暂时可将寄存器 R1 中的结果存回到内存的变量 T2 中，从而将寄存器 R1 空闲以备使用。

LD R1, E

SUB R1, F //运算结束后 R1 中为 T4 结果，内存中无该结果

MUL R0， R1 //运算结束后 R0 中为 T5 结果，内存中无该结果。注意，该指令将寄存器 R0 中原来的结果 T3 冲掉了。可以这么做的原因是，T3 在该指令后不再有引用点，且是非活跃变量。

LD R1, T2 //此时 R1 中结果 T4 已经没有引用点，且临时单元 T4 是非活跃的，所以，寄存器 R1 可作为空闲寄存器使用。

DIV R1, R0 //运算结束后 R1 中为 W 结果，内存中无该结果。此时所有指令部分已经翻译完毕。

ST R1, W //指令翻译完毕时，寄存器中存有最新的计算结果，必须将它们存回到内存相应的单元中去，否则，在翻译下一个基本时，所有的寄存器被当成空闲的寄存器使用，从而造成计算结果的丢失。考虑到寄存器 R0 中的 T5 的寄存器 R1 中的 W，临时单元 T5 是非活跃的，所以，只要将结果 W 存回对应单元即可。

2. 假设可用的寄存器为 R0 和 R1，其中 T4 是活跃变量，试对以下四元式基本块：

T1:=A+B

T2:=C+D

T3:=E-T2

T4:=T1-T3

用简单代码生成算法生成其目标代码。

解答：

该基本块的目标代码如下：

LD R0, A

ADD R0, B

LD R1, C

ADD R1, D

ST R0, T1

LD R0, E

SUB R0, R1

```
LD R1, T1
SUB      R1, R0
ST  R1, T4
```