题目列表 (/contests/0) 评测记录 (/contests/0/status/)

排行榜 (/contests/0/rank_list/)

虚拟机设计（第一部分）

时间限制： 1 秒

内存限制： 2 GB

此部分分值：105分

【问题描述】

菜菜在上过汇编语言这门课后，自己设计了一套“精简指令集”。为了测试以及进一步地调整，菜菜需要一个虚拟机来将这些指令执行起来。你能帮帮他么？

菜菜需要一个16位的虚拟机，内存地址从 `0000` 到 `FFFF` ，总共有 2^{16} 字节。规定其中 $[3000, B000)$ 是数据段（区间左闭右开），即指令可以进行读写操作的内存段只有 2^{15} 字节。CPU内有四个16位寄存器可供使用： `AX` 、 `BX` 、 `CX` 、 `DX` 。CPU处理的所有数据皆为16位（二进制位）的无符号整数。

指令中的操作数总共有三种形式：立即数、寄存器和内存。

- 立即数：一个四位的16进制常数，不会省略前导零，字母使用大写，如 `02C0` ；
- 寄存器： `AX` 、 `BX` 、 `CX` 或 `DX` ，字母皆为大写。
- 内存：采用“立即数直接寻址”和“寄存器间接寻址”两种方式，给出内存地址，根据该地址去内存中存取数据。具体的形式为 `T+立即数` 和 `T+寄存器` ，如 `T02C0` 和 `TAX` ，其中立即数或相应寄存器中的值表示内存地址。

每条指令对操作数中的值进行相应地处理。对于三种不同类型的操作数，“操作数中的值”这一概念的具体含义略有不同：立即数本身即是值；寄存器中存放的整数是值；内存则是根据内存地址取出的数据是它的值。

最近递交

已递交2次, 剩余8次

递交时间	状态	得分
12:54:01	AC (/contests/0/detail/733) 最好递交	105
11:59:35	WA (/contests/0/detail/419)	77

递交

请选择语言

g++

请选择答案文件

Browse... No

递交评测(剩余8次)

数据在内存中以**小端模式**存储，即数据的高字节保存在内存的高地址中，而数据的低字节保存在内存的低地址中。菜菜的虚拟机中所有的数据均为16位（二进制位）无符号整数，在内存中将占据相邻的两个字节，其中高8位将存于高地址处，低8位则存于低地址处。

下图显示了一段内存的存储情况（ `4000` 到 `4004` ）。

4000	4001	4002	4003	4004
00	00	0C	FF	00

对于上图中的情况，如果想要根据地址 `4002` 从内存中读取数据，则将会读取 `4002` 和 `4003` 中的数据。其中高地址 `4003` 的 `FF` 作为高位，低地址 `4002` 的 `0C` 作为低位，读取的数据则为 `FF0C` 。

如果想要向地址 `4001` 处写入数据 `54AC` ，则将会把数据写入 `4001` 和 `4002` 中。其中高位的 `54` 写入高地址 `4002` 处，低位的 `AC` 则写入低地址 `4001` 处。执行写入操作后，内存情况变为下图：

4000	4001	4002	4003	4004
00	AC	54	FF	00

菜菜目前需要你来实现下面几种基本的指令：

1. `RUN` ： 标识着程序的开始。如无特殊说明，内存和寄存器均已初始化为0。
2. `STOP` ： 标识着程序的正常结束。
3. `ECHO A` ： 将操作数 A 中的值输出。
4. `ADD A B` ： 将操作数 A 中的值与 B 中的值相加，结果存回 A。相加产生溢出时，直接将溢出部分丢弃即可（截断）——无需向更高位进位，存回 A 的同样是一个16位（二进制位）无符号整数。A 不能为立即数。
5. `INC A` ： 将操作数 A 中的值加 1，结果存回 A。同样忽略溢出，A 不能是立即数。
6. `MOV A B` ： 将操作数 B 中的值写入 A，A 不能是立即数。
7. `CMP A B` ： 比较操作数 A 和 B 中的值的大小，结果将作为条件跳转指令的依据。

8. 跳转指令。

这里详细说明一下跳转指令：在没有跳转指令的情况下，虚拟机会按照程序编写的顺序执行每一条指令，而跳转指令则能够指定虚拟机接下来执行哪一条指令。具体地来说，假设总共有 n 条指令，从 1 到 n 对其进行标号。当第i 条指令执行完后，如果不是跳转指令，虚拟机将自动执行下一条即第 $i+1$ 条指令；但如果第i条指令是跳转指令，接下来将可能执行它指定的某一条指令。

跳转指令分为两种：条件跳转和无条件跳转。顾名思义，无条件跳转指令执行完后，一定会执行它指定的某一条指令；而条件跳转指令执行完后，只有在满足某些条件时，才会执行它指定的指令，如果不满足则仍然按照默认顺序执行下一条指令。

无条件跳转指令只有一种： `JMP X`

不妨假设操作数 X 中的值是 i，则该指令执行完后，将去执行第 i 条指令。

条件跳转指令则是以**上一次 `CMP A B` 指令执行时**的比较结果作为条件，根据操作数 A 和 B 中的值的大小关系，共有以下6种形式。这里同样不妨假设 A 中的值是 a，B 中的值是 b。

1. `JG X`：a 大于 b 时跳转
2. `JL X`：a 小于 b 时跳转
3. `JE X`：a 等于 b 时跳转
4. `JNG X`：a 不大于 b 时跳转
5. `JNL X`：a 不小于 b 时跳转
6. `JNE X`：a 不等于 b 时跳转

不妨假设操作数 X 中的值是 i，则该指令执行完后，如果满足相应的条件，将去执行第 i 条指令，否则按默认顺序继续执行下一条指令。需要注意的是， `CMP` 指令与条件跳转指令不是一一对应的关系。一条 `CMP` 指令的结果将作为所有条件跳转指令的依据，直到执行下一条 `CMP` 指令为止。

菜菜的虚拟机对程序的格式也有着严格的要求：

1. 每行一条指令，指令内部不同部分之间仅用一个空格进行分隔，不允许有多余空格。
2. 第一行指令是 `RUN`，最后一行指令是 `STOP`，程序中不允许有其它的 `RUN` 和 `STOP`。
3. 指令中所有的字母均为大写。

4. 出于对时钟周期的考量，一条指令中不能同时存在两个内存操作数。

菜菜已经用上述8种指令写好了一段程序。虽然每一条指令都严格符合上述约定，但由于数据的不可预知性，程序在运行过程中仍然可能会遇到以下几种问题：

1. 存取非法。程序只能对寄存器和数据段 $[3000, B000)$ 进行操作，如果当前指令试图读写数据段以外的内存，则虚拟机应当立即报错，不会执行当前以及后面的指令。
2. 耗时过长。程序本身算法复杂度较高，或者陷入死循环死递归，将无法在短时间内得出结果。所以规定，一段程序最多执行一百万条次的指令。因为**跳转指令**的存在，每条指令可能执行多次，这里按照执行指令的次数总和计算。需要注意的是，不满足条件的**条件跳转指令**同样算做一条次指令计入总和。如果第一百万条次的指令顺利执行完毕，并且本身不是 `STOP` 指令，则虚拟机应当强制终止该程序，不再执行后面的指令。
3. 跳转错误。假设总共有 n 条指令，因为第一条指令一定是 `RUN` ，所以规定**跳转指令**只能跳转到第 2 到第 n 条指令。当**无条件跳转指令**或满足条件的**条件跳转指令**试图跳转到第 i 条指令时，如果 $i < 2$ 或 $i > n$ 虚拟机应当立即报错并终止该程序。
4. CMP缺失。**条件跳转指令**是以 `CMP` 指令的结果为依据的，所以在执行**条件跳转指令**时，如果之前从未执行过 `CMP` 指令，虚拟机应当立即报错并终止该程序。

   考虑到内在的逻辑关系，除了耗时过长是在指令执行结束后判断，其余三种错误的优先级从高到低依次为CMP缺失 > 存取非法 > 跳转错误。如果某条指令执行时同时涉及多个错误，则把其中优先级最高的视为程序异常退出的原因。

希望你能按照上述要求将虚拟机实现，来执行菜菜的这段程序。

【输入格式】

给出用上述8种指令编写的一段程序，每行一条指令，保证没有格式错误。

【输出格式】

每个顺利执行的 `ECHO` 指令输出一行，一个四位十六进制整数（字母大写、不足四位用前导零补齐），表示该操作数中的值。

如果程序因**存取非法**而异常退出，则再输出一行 `ACCESS_VIOLATION` 。

如果程序因**耗时过长**而异常退出，则再输出一行 `TLE` 。

如果程序因**跳转错误**而异常退出，则再输出一行 `RUNTIME_ERROR` 。

如果程序因**CMP缺失**而异常退出，则再输出一行 `CMP_MISSING` 。

【样例1】

输入

```
RUN
MOV T4001 54AC
MOV T4003 00FF
ECHO T4002
MOV BX T4002
ECHO BX
STOP
```

输出

```
FF54
FF54
```

【样例2】

输入

```
RUN
MOV T4001 54AC
MOV T4003 00FF
MOV BX T4002
ADD BX T4001
ECHO BX
ADD BX T4000
ECHO TBX
ECHO BX
STOP
```

输出

```
5400
ACCESS_VIOLATION
```

【样例3】

输入

```
RUN
JMP 0002
ECHO T0000
STOP
```

输出

```
TLE
```

【样例4】

输入

```
RUN
MOV BX 0003
CMP AX BX
JNL 0010
ECHO AX
INC AX
JMP AX
STOP
```

输出

```
0000
RUNTIME_ERROR
```

子任务

每个测试用例中的程序均小于等于 100 行。

对于前三分之一的测试用例，程序中不涉及内存操作，即操作数中不会出现 `T`；

对于前三分之二的测试用例，程序中没有跳转指令和 `CMP` 指令。