任务概述:

模拟一个简易的冯诺依曼式计算机 CPU 的工作。

该 CPU 字长为 16 位，共 11 个寄存器，其中 3 个系统寄存器，分别为程序计数器，指令寄存器，标志寄存器；8 个通用寄存器，即寄存器 1、2、3、4（数据寄存器），寄存器 5、6、7、8（地址寄存器）。该 CPU 至多支持 32K 内存。内存分两部分，一部分为代码段，从地址 0 开始。另一部分为数据段，从地址 16384 开始。其所支持的指令集如表 1-1。每条指令固定由 32 位（编号为 0 到 31）二进制数组成，其中第 0 到 7 位为操作码，代表 CPU 要执行哪种操作；第 8 到 15 位为操作对象，如寄存器，内存地址等；第 16 到 31 位为立即数。该 CPU 有一个输入端口和一个输出端口。输入端口的数据由标准输入设备（键盘）输入，输出端口的数据输出到标准输出设备（显示器）上。

程序的大致步骤:

程序开始时要从指定文件中读入一段用给定指令集写的程序至内存（从地址 0 开始顺序保存），程序计数器初始值也为 0。然后程序就开始不断重复取指令（读取程序计数器 PC 内的指令地址，根据这个地址将指令从内存中读入，并保存在指令寄存器中，同时程序计数器内容加 4，指向下一个条指令。因为我们所有的指令长度固定为 4 个字节，所以加 4）。分析指令（对指令寄存器中的指令进行解码，分析出指令的操作码，所需操作数的存放位置等信息）和执行指令（完成相关计算并将结果写到相应位置）的过程。程序每执行一条指令就要输出 CPU 当前的状态，如各寄存器的值，指定内存的值等（具体要求在后边）。当读到停机指令时，程序按要求输出后就结束了。

程序也可以分为单核版和多核版两个。

多核版说明:

- **只做两个核的版本（扩展到多核版类似）。**
- **增加了 3 条指令**

为了简化程序，增加三条功能很强的指令，具体见指令集。这与实际的 CPU 差异很大。

- **每个核有自己的一套寄存器，但两个核共享内存。核心 1 的代码段从地址 0 开始，核心 2 的代码段从地址 256 开始。**
- **每个核读入自己的指令序列。**
- **验收指令序列非常简单，两个核的指令序列是一样的，实现的就是多线程一章里的卖票程序（共 100 张票，两个线程卖）。**
- **为了简化程序，我们规定 100 这个值存在地址为 16384 的内存里。所以程序初始化时要将这块内存的值初始化为 100。程序结束时它应该变成 0。**

大作业多核版追加要求:

因为两个子线程在输出每步指令结果和输出主存时，可能因超时中断，造成输出结果混在一起，很乱。所以为了验收顺利，追加两个要求:

1. 输出主存的代码区和数据区，因为耗时较长，不要放在子线程中作，务必在子线程结束后由主线程来统一输出。

2. 输出每步指令结果，只能由子线程完成，为避免半路中断，请在程序中主动添加互斥请求，保证输出的完整性。注意：这个互斥请求，输入指令里不会有，必须同学们自己来主动保护。

表 1-1 指令集

	指令	说明

停机指令	00000000 00000000 0000000000000000	停止程序执行。
数据传送指令	00000001 00010000 0000000000000000	将一个立即数（绿色部分）传送至寄存器 1
	00000001 00010101 0000000000000000	将寄存器 5（5、6、7、8 号寄存器为地址寄存器）中地址所指向的内存单元（2 个字节）的内容传送至寄存器 1
	00000001 01010001 0000000000000000	将寄存器 1 的内容传送至寄存器 5 中地址所指向的内存单元（2 个字节）5、6、7、8 号寄存器为地址寄存器）。
算数运算指令	00000010 00010000 0000000000000000	将寄存器 1 内的数与一个立即数（绿色部分）相加，结果保存至寄存器 1
	00000010 00010101 0000000000000000	将寄存器 1 内的数与寄存器 5 中地址所指向的内存单元（2 个字节）里存的数相加，结果保存至寄存器 1
	00000011 00010000 0000000000000000	将寄存器 1 内的数减去一个立即数（绿色部分），结果保存至寄存器 1
	00000011 00010101 0000000000000000	将寄存器 1 内的数减去寄存器 5 中地址所指向的内存单元（2 个字节）里存的数，结果保存至寄存器 1
	00000100 00010000 0000000000000000	将寄存器 1 内的数与一个立即数（绿色部分）相乘，结果保存至寄存器 1
	00000100 00010101 0000000000000000	将寄存器 1 内的数与寄存器 5 中地址所指向的内存单元（2 个字节）里存的数相乘，结果保存至寄存器 1
	00000101 00010000 0000000000000000	将寄存器 1 内的数除以（C 语言的整数除法）一个立即数（绿色部分），结果保存至寄存器 1
	00000101 00010101 0000000000000000	将寄存器 1 内的数除以（C 语言的整数除法）寄存器 5 中地址所指向的内存单元（2 个字节）里存的数，结果保存至寄存器 1
逻辑运算指令	00000110 00010000 0000000000000000	将寄存器 1 内的数与一个立即数（绿色部分）做逻辑与，结果保存至寄存器 1（如果结果为真则保存 1，否则保存 0）
	00000110 00010101 0000000000000000	将寄存器 1 内的数与寄存器 5 中地址所指向的内存单元（2 个字节）里存的数做逻辑与，结果保存至寄存器 1（如果结果为真则保存 1，否则保存 0）
	00000111 00010000	将寄存器 1 内的数与一个立即数（绿色部分）做逻辑或，结果保存至寄存器 1（如果结果为真则保存 1，否则保存 0）

	0000000000000000	
	00000111 00010101 0000000000000000	将寄存器 1 内的数与寄存器 5 中地址所指向的内存单元（2 个字节）里存的数做逻辑或，结果保存至寄存器 1（如果结果为真则保存 1，否则保存 0）
	00001000 00010000 0000000000000000	将寄存器 1 内的数做逻辑非，结果保存至寄存器 1（如果结果为真则保存 1，否则保存 0）
	00001000 00000101 0000000000000000	将寄存器 5 中地址所指向的内存单元（2 个字节）里存的数做逻辑非，结果仍保存至寄存器 5 中地址所指向的内存单元（如果结果为真则保存 1，否则保存 0）
比较指令	00001001 00010000 0000000000000000	将寄存器 1 内的数与一个立即数（绿色部分）比较，如两数相等，则标志寄存器被修置为 0，如寄存器 1 大，则标志寄存器被置为 1，如寄存器 1 小，则标志寄存器被置为-1。
	00001001 00010101 0000000000000000	将寄存器 1 内的数与寄存器 5 中地址所指向的内存单元（2 个字节）里存的数比较，如两数相等，则标志寄存器被置为 0，如寄存器 1 大，则标志寄存器被置为 1，如寄存器 1 小，则标志寄存器被置为-1。
跳转指令	00001010 00000000 0000000000000000	无条件跳转指令，转移至程序计数器加一个立即数（绿色部分）处执行。也就是说要修改程序计数器。
	00001010 00000001 0000000000000000	如果标志寄存器内的值为 0 则转移至程序计数器加一个立即数（绿色部分）处执行。也就是说要修改程序计数器。
	00001010 00000010 0000000000000000	如果标志寄存器内的值为 1 则转移至程序计数器加一个立即数（绿色部分）处执行。也就是说要修改程序计数器。
	00001010 00000011 0000000000000000	如果标志寄存器内的值为-1 则转移至程序计数器加一个立即数（绿色部分）处执行。也就是说要修改程序计数器。
输入输出指令	00001011 00010000 0000000000000000	从输入端口读入一个整数并保存在寄存器 1 中。也就是从键盘读一个整数到寄存器 1 中。
	00001100 00010000 0000000000000000	将寄存器 1 中的数输出到输出端口。也就是将寄存器 1 中的数以整数的形式输出到显示器上，同时输出一个换行符。
多核版指令	00001101 00000000 0000000000000000	立即数（绿色部分）为内存地址，请求互斥对象，用于锁住立即数所指定的内存。
	00001110 00000000 0000000000000000	立即数（绿色部分）为内存地址，释放互斥对象，释放掉锁住立即数所指定的内存的互斥对象。与上一条指令对应。
	00001111 00000000 0000000000000000	休眠立即数（绿色部分）毫秒。

输入方式:

以文件的方式输入，该文件为一个以停机指令为结尾的指令序列。如:

```
00001011000100000000000000000000
00000001010100000100000000000000
00000001010100010000000000000000
00001011000100000000000000000000
00000010000101010000000000000000
00001100000100000000000000000000
00000000000000000000000000000000
```

输出方式:

每执行一条指令后都要输出各寄存器状态，格式见样例。

```
    ip = 44
    flag = -1
    ir = 20316160
    ax1 = 4 ax2 = 6 ax3 = 3 ax4 = 0
    ax5 = 16384 ax6 = 16386 ax7 = 0 ax8 = 0
```

当执行到输入指令时在用户输入前要输出:

in :\n

```
    in:
    20
```

当执行到输出时输出前要先输出:

out :

输出指令结束后要输出一个换行符。

所有指令执行完后要输出当前内存内容;

先输出代码段：每四个字节一输出（也就是每条指令当成一个整数输出），每行输出 8 条指令，共输出 16 行;

然后输出数据段：每两个字节当成一个整数输出，每行输出 16 个整数，共输出 16 行。具体格式见样例。

```
codeSegment :
185597952 22036480 22085632 23085058 17825792 23134208 152371200 167903260
36044800 34603009 20316160 36700161 23265280 167804956 203423744 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
dataSegment :
5 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

多核版输出变化说明:

为了区分每个核心的输出，增加一个类似寄存器的核心 ID（id）,该值在整个线程运行期间不变，比如核心 1 的 id 为 1，核心 2 的 id 为 2 等。大家的输出有两处要修改，一是在输出所有寄存器之前先输出核心 ID，格式见后边输出样例；二是在执行输出指令时，也是先输出核心 ID，格式见输出样例。

输出样例:

寄存器输出:

```
id = 2
ip = 268
flag = 0
ir = 277
ax1 = 10 ax2 = 0 ax3 = 0 ax4 = 0
ax5 = 16384 ax6 = 0 ax7 = 0 ax8 = 0
```

输出指令输出:

```
id = 1        out: 8
```

注：红色字体为新增内容。

常见问题解答:

问题 1: 指令集中没有设置地址寄存器内地址的指令，只有使用寄存器 5 中地址的指令。怎么办?

答: 数据传送指令对寄存器 1 到 8 通用，就是说可以把立即数直接传送至这些寄存器。

问题 2: 一个指令 4 个字节，一行 8 个指令，一共 16 行，加起来也才 512 个字节，剩下的 15872 个字节是被吃掉了吗?

答: 只显示很少的内存是因为我们的代码写不了那么多。后边的都是 0，不用显示的。但按我们的设计可以有那么多。

袁老师 3 月 27 日特别说明:

1、指令寄存器（ir，Instruction Register）为 16 位，所以只保存我们的 32 位指令的前 16 位，后 16 位的立即数部分不保存。

2、立即数部分为补码。

3、碰到输入指令时要先输出 in:回车。

4、碰到输出指令时要先输出 out: （注意，冒号后有一个空格），然后输出需要输出的内容，最后再输出一个换行符。

5、输出 codeSegment :和 dataSegment :前先输出一个换行符。

6、dict.dic 中为指令序列及解释，output.txt 中为实际运行结果（输入的整数为 5）。

多核版常见问题解答:

1. 1. id = 1 和 out 之间是空格？还是 tab 四个空格？文档里是 tab。
   答：四个空格。
2. 既然两个核的指令是一样的，那为何还要分别存到 0 和 256 中呢？存到一处的话读取也互不影响。
   答：虽然测试指令序列两个核心一样，但设计上要考虑不一样的情况。
3. 测试指令程序不是多个吗？在读写票数之前是一定会上锁吗？会不会出现不上锁直接读写的情况？
   答：如果我的指令没写错，读写之前一定上锁，不会有不上锁就读写的情况。
4. 题目要求的是锁住立即数部分指向的内存，我就不知道怎么处理了？原则上它应该能锁住任意部分的内存。
   答：请求互斥对象语句可以锁住任何后续的操作内存，直到释放互斥对象。所以，不用管锁哪里，只要执行指令调用互斥对象语句就可以。