计算机科学导论实验

班级快速排序实验

计算机科学导论实验课程助教组

2020年5月15日

助教：郭泓锐

guohongrui17@mails.ucas.ac.cn

内容大纲

- 实验目的
- 知识回顾
- 预备知识
- 实验内容
- 评分标准

内容大纲

- 实验目的
- 知识回顾
- 预备知识
- 实验内容
- 评分标准

实验目的

- 加深对快速排序算法的理解。
- 初步接触计算机组成原理的知识内容，理解自动计算是如何实现的。

内容大纲

- 实验目的
- 知识回顾
- 预备知识
- 实验内容
- 评分标准

知识回顾

- 快速排序算法
  - 核心思想：分治。

知识回顾

- 快速排序算法
  - 如果 数据个数 ⩽ 1，直接结束。
  - 否则
    - 从数据中随机取一个数当作“标杆”。
    - 剩余数据分成较小组（小于“标杆”）和较大组（大于“标杆”） 。
    - 把较小组放在“标杆”左侧，把较大组放在“标杆”右侧。（划分——分）
    - 分别对这两组数据再执行快速排序算法。（递归——治）

知识回顾

- 举个例子

9	4	6	7	3	8	1

知识回顾

- 举个例子

知识回顾

- 举个例子

3	1	4	9	6	7	8

知识回顾

- 举个例子

知识回顾

- 举个例子

知识回顾

- 举个例子

知识回顾

- 举个例子

知识回顾

- 举个例子

知识回顾

- 举个例子

知识回顾

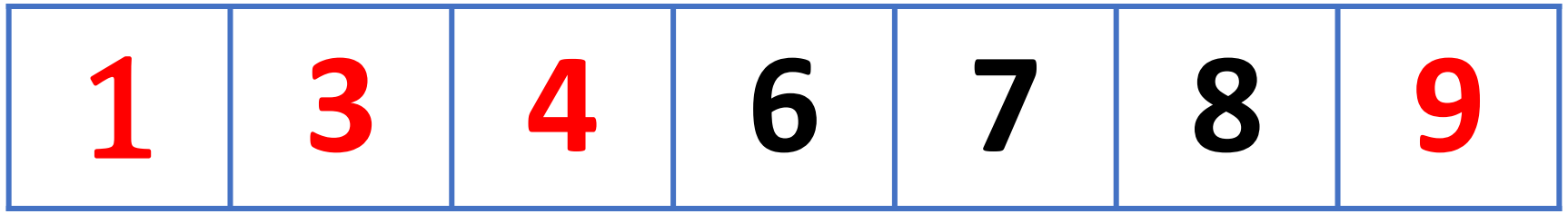
- 举个例子

知识回顾

- 举个例子

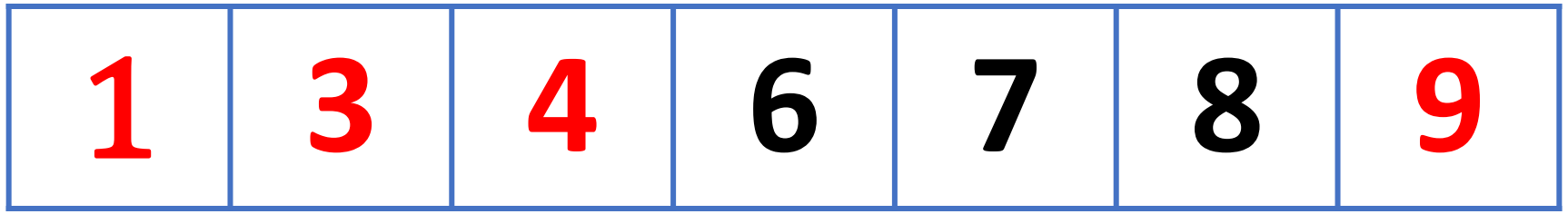
知识回顾

- 举个例子



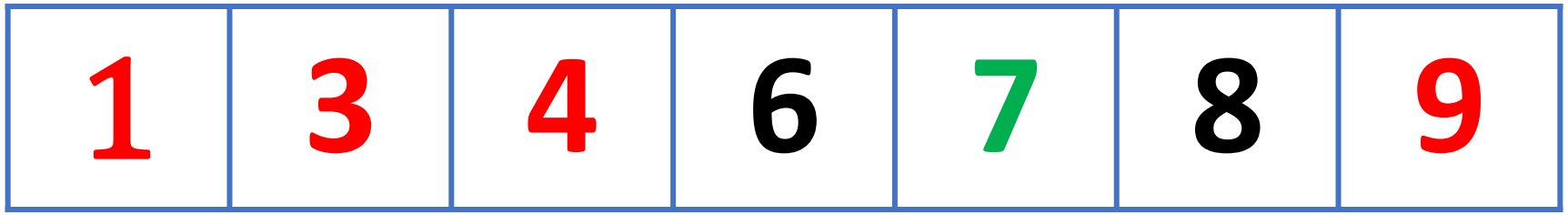
知识回顾

- 举个例子



知识回顾

- 举个例子



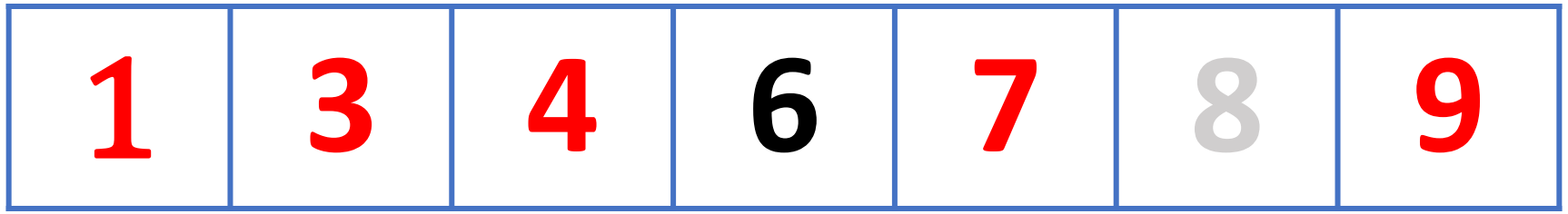
知识回顾

- 举个例子

1	3	4	6	7	8	9

知识回顾

- 举个例子



知识回顾

- 举个例子

知识回顾

- 举个例子

1	3	4	6	7	8	9

知识回顾

- 举个例子

知识回顾

- 举个例子

知识回顾

- 快速排序算法时间复杂性
  - 最坏：$\Theta(n^2)$;
  - 期望：$\Theta(\mathrm{n}\log\mathrm{n})$.

- 每次需要随机选取一个标杆，若固定标杆位置则无法达到期望时间！

内容大纲

- 实验目的
- 知识回顾
- 预备知识
- 实验内容
- 评分标准

预备知识

- 问题：如何找出30亿个数中的最大值

- 用计算机找！

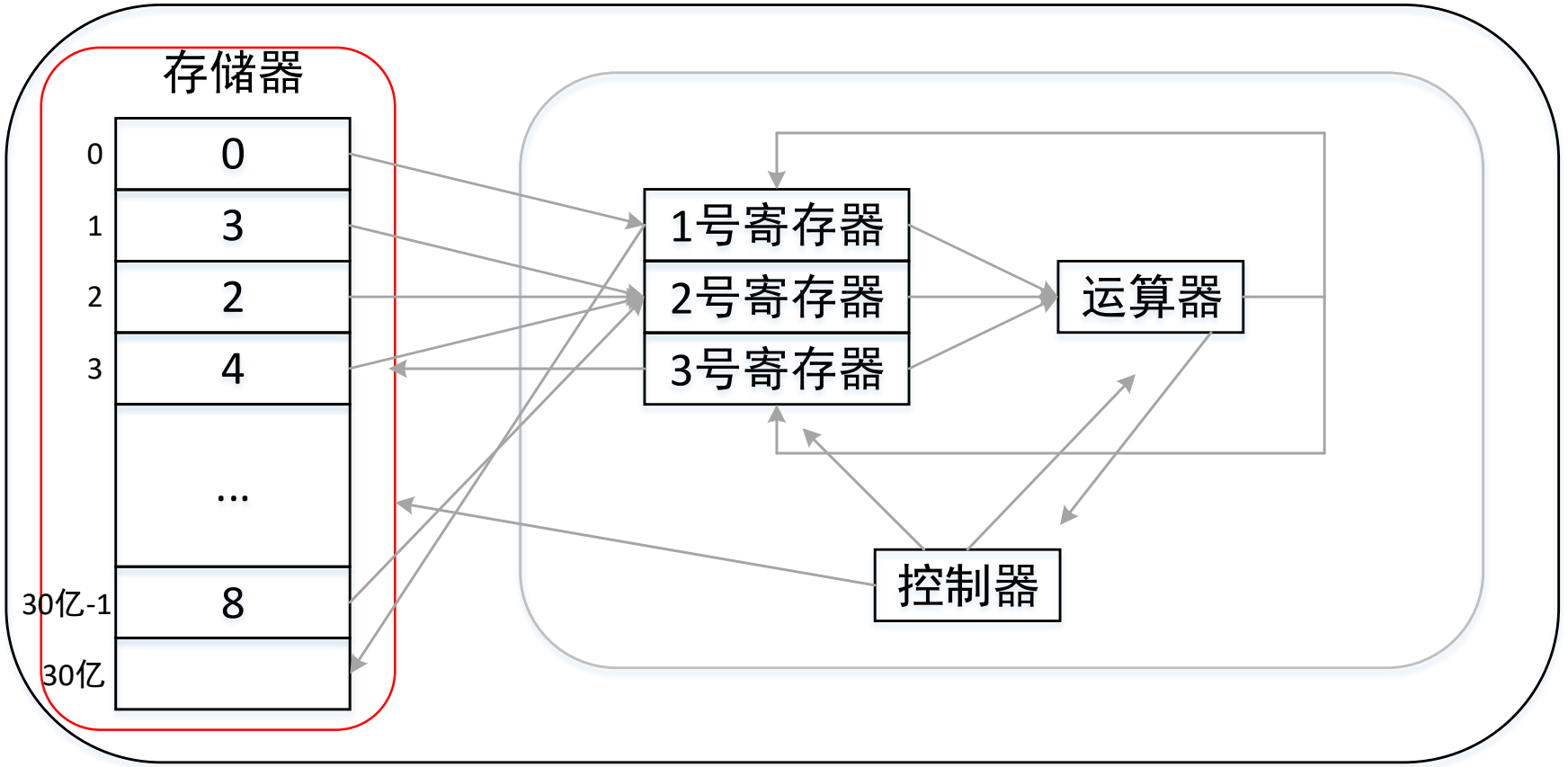
预备知识

- 自然语言描述：
  - 先将第1个数记下。
  - 遍历第2个数，第3个数，…，第30亿个数：如果当前数大于记下的数，就用当前的数替换掉记下的数。
  - 遍历完成后记下的数就是最大数。

预备知识

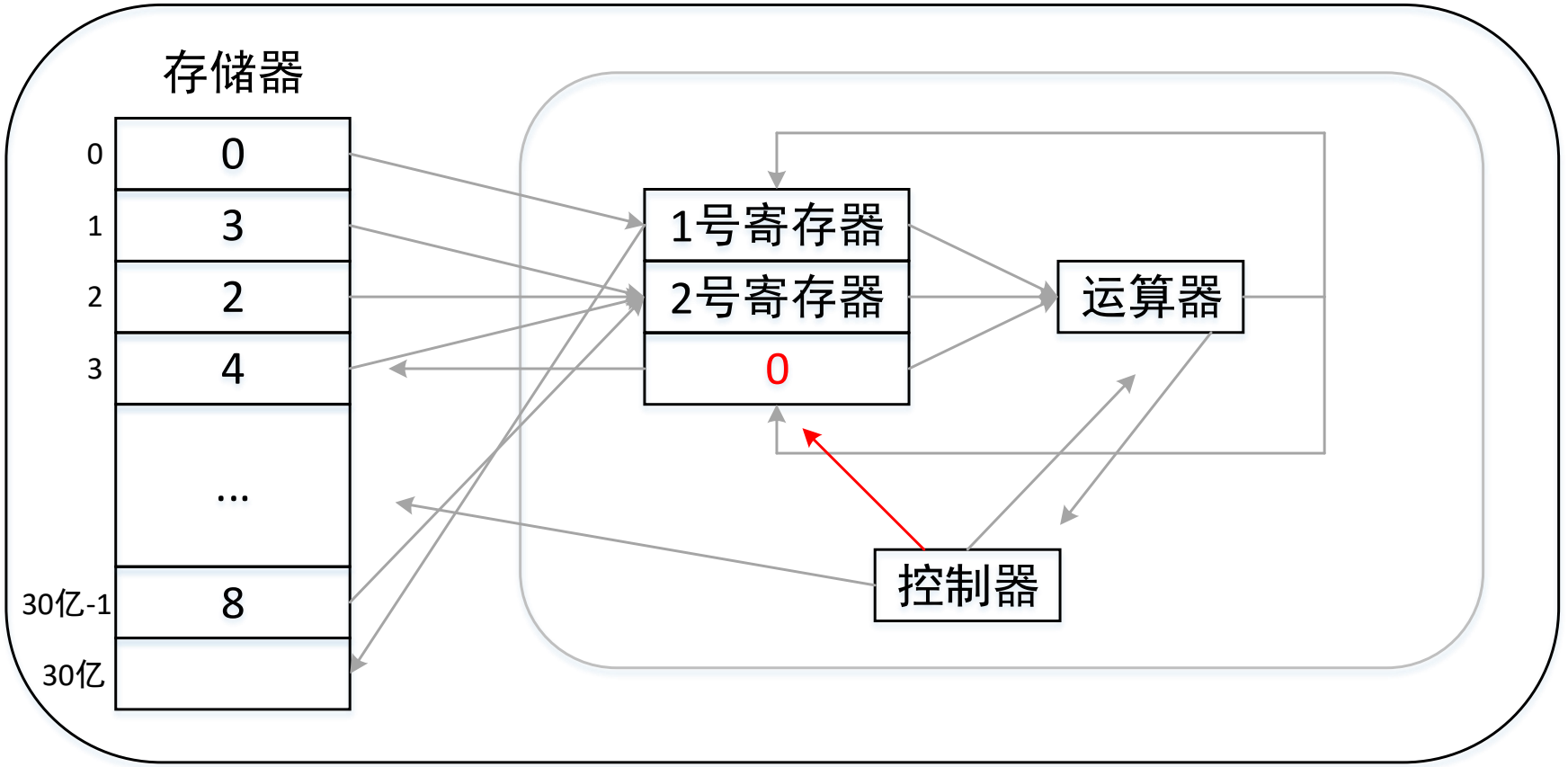
指令

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



预备知识

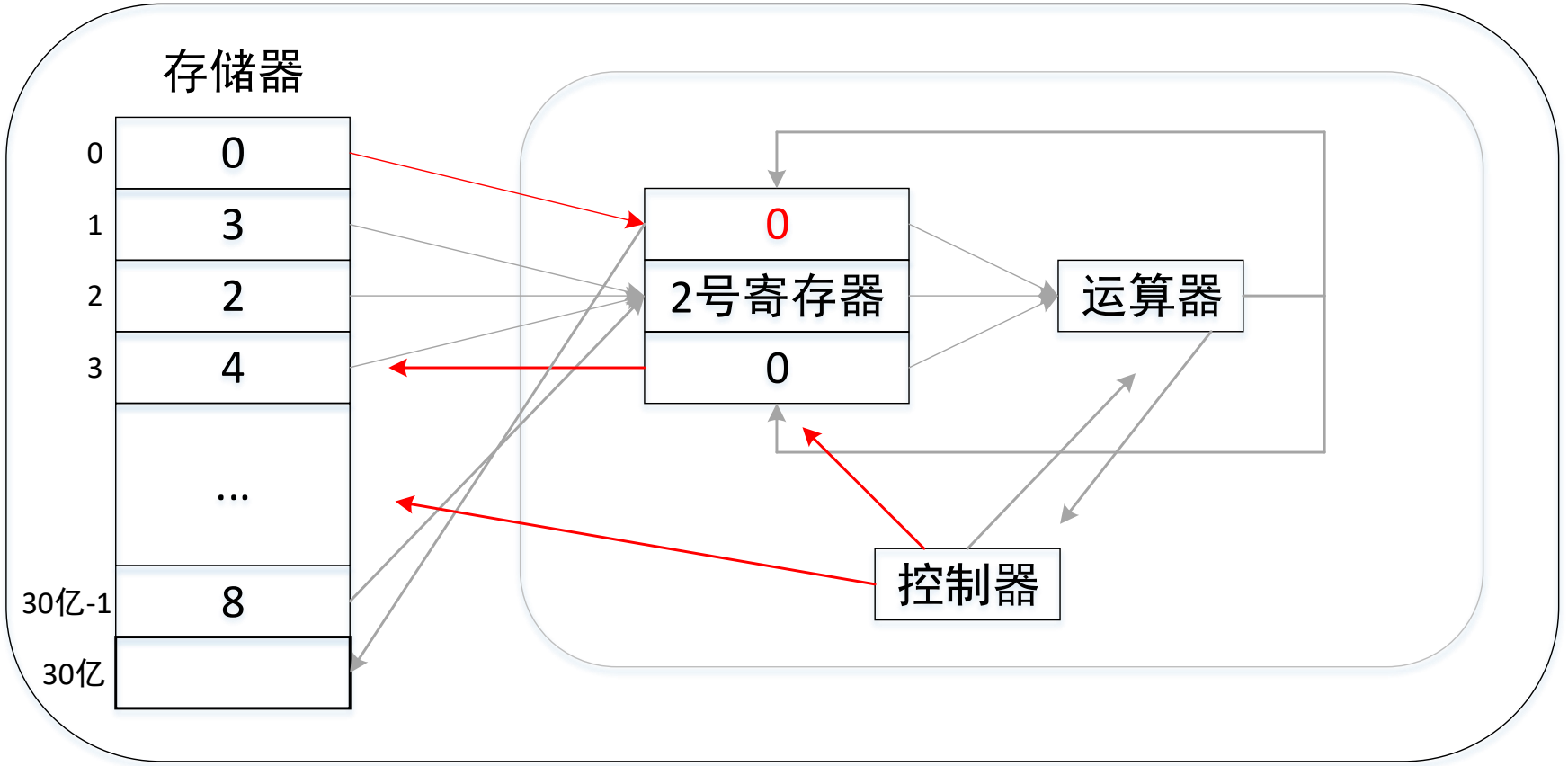
step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



预备知识

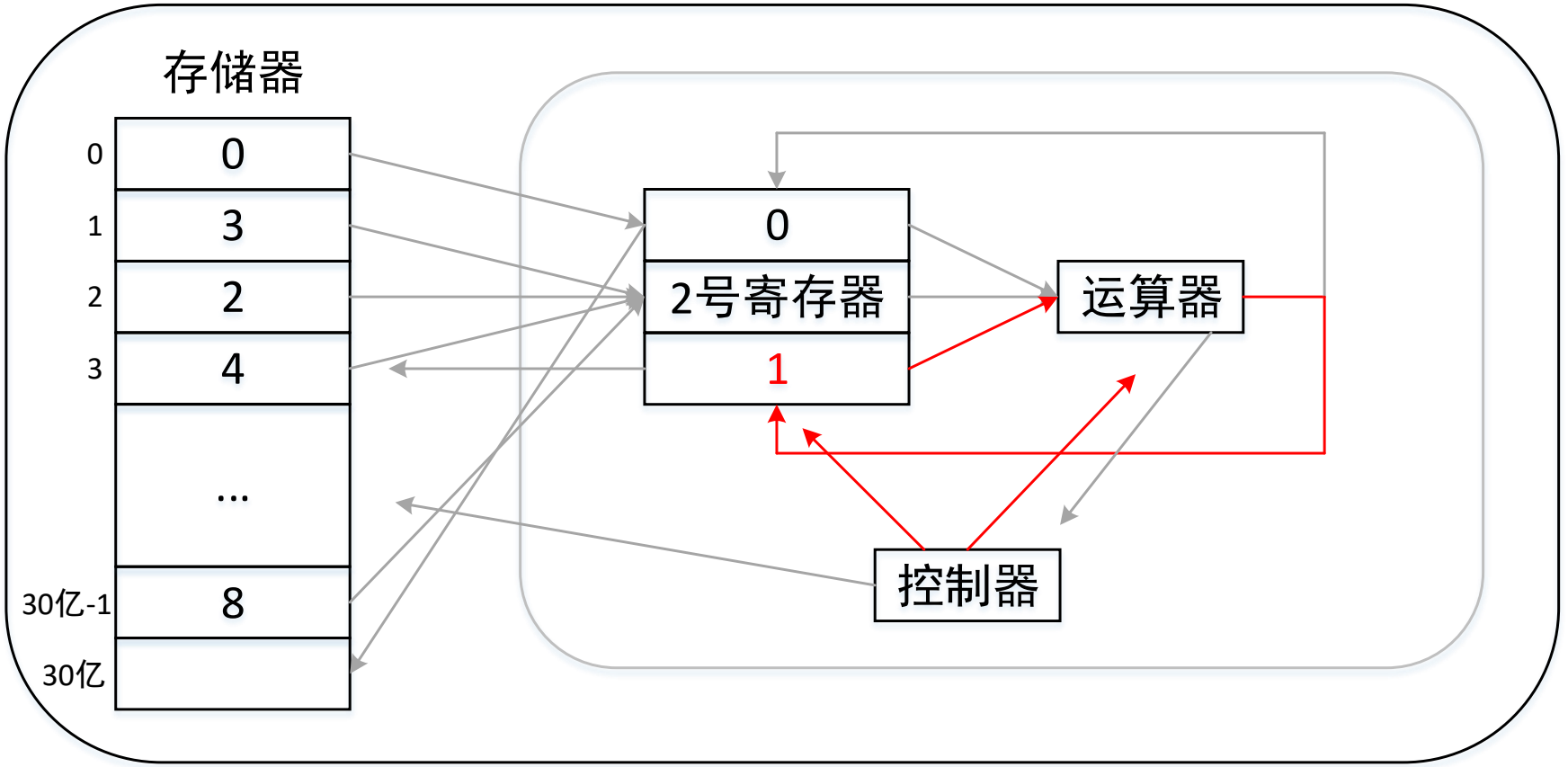
先将第1个数记下
（记在1号寄存器中）

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



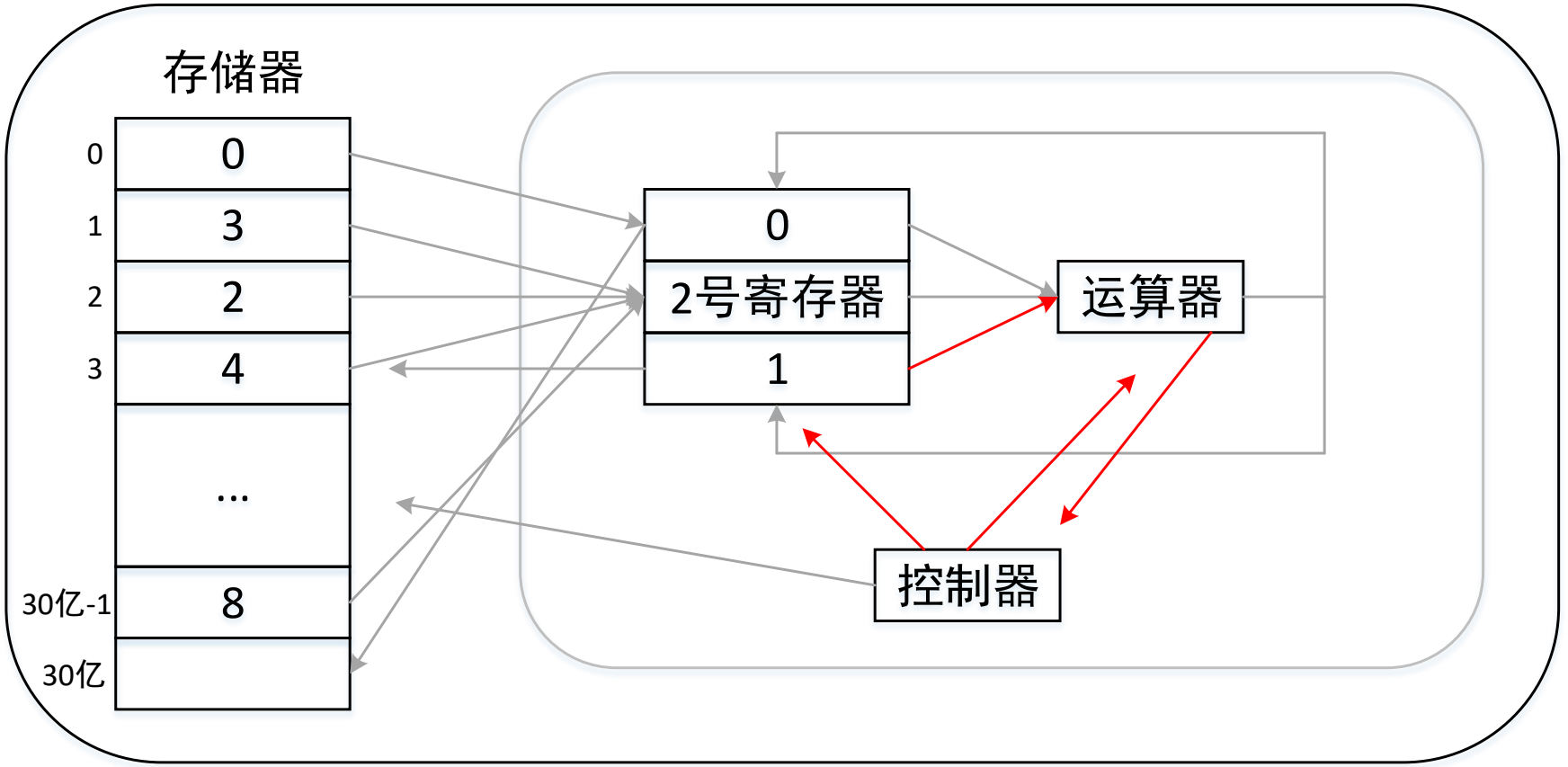
预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



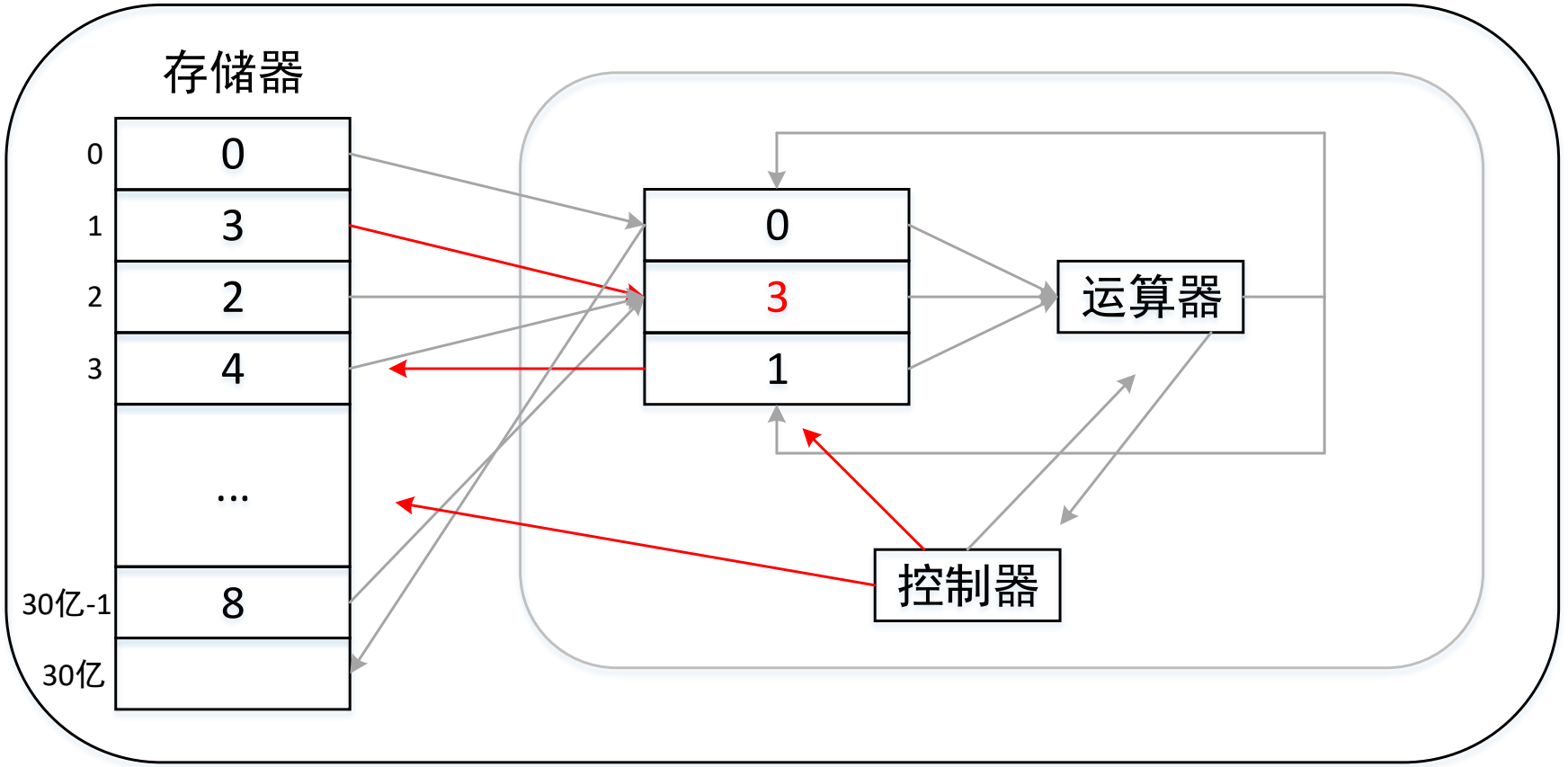
预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



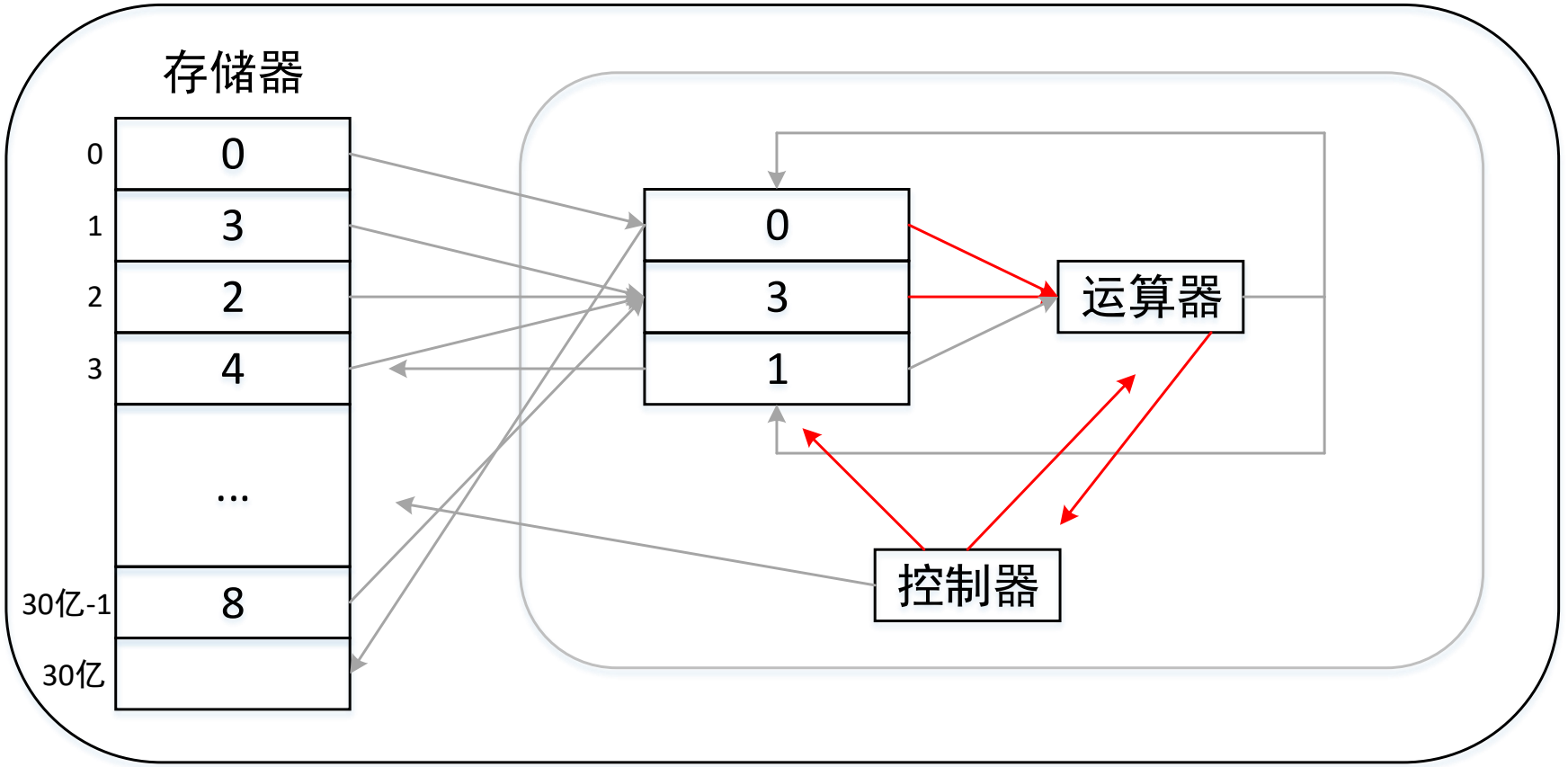
预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



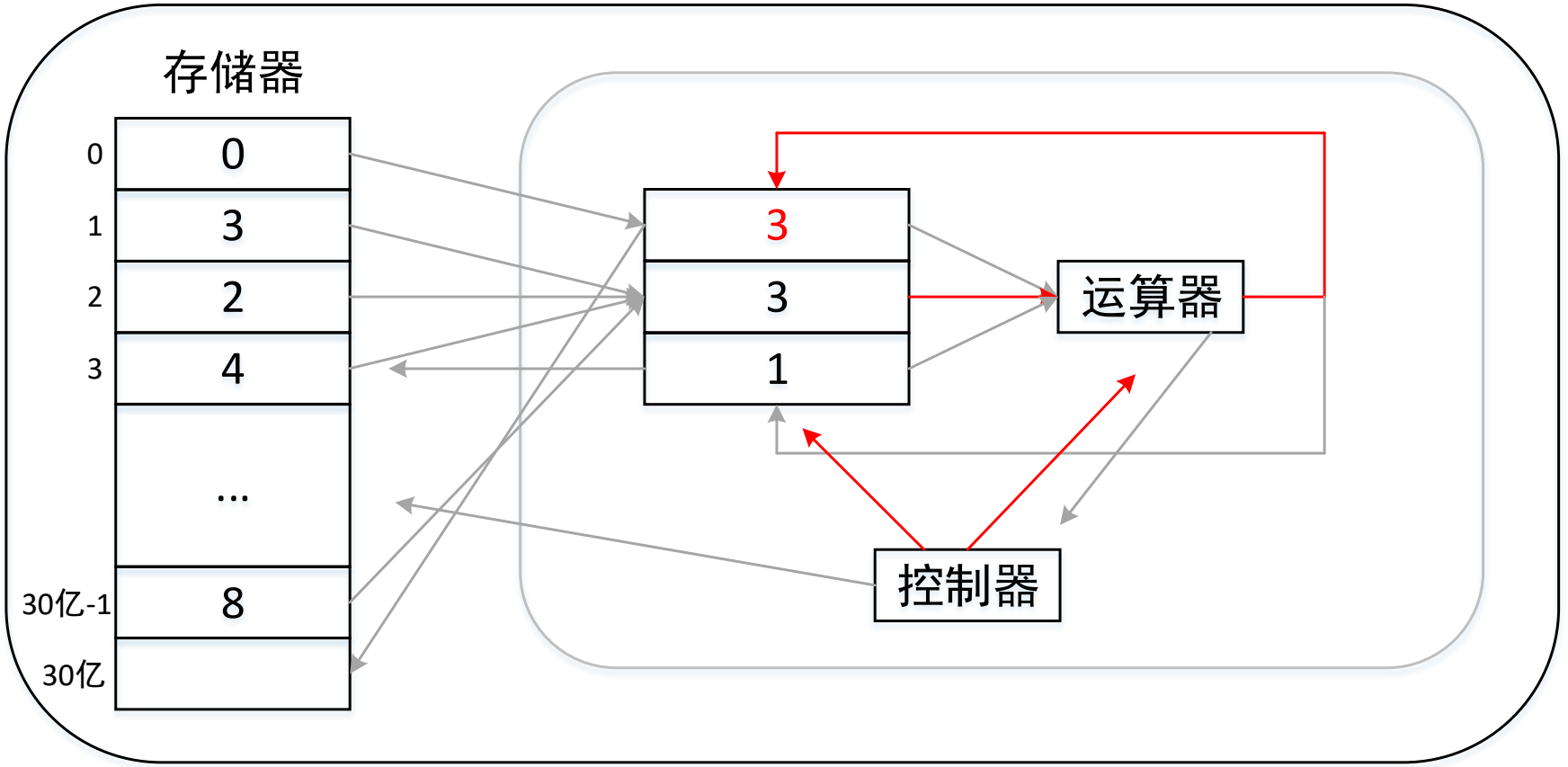
预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



预备知识

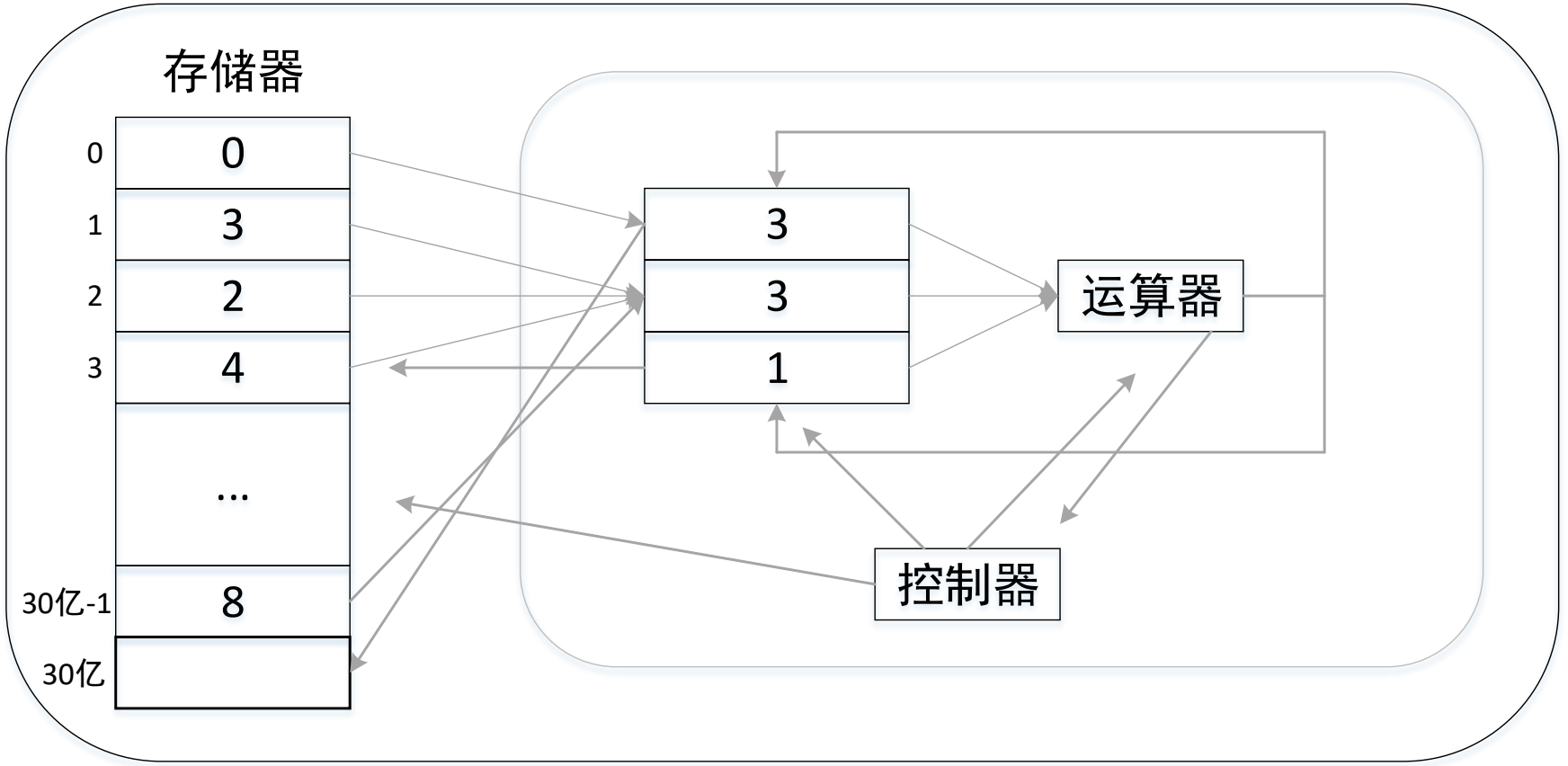
step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



预备知识

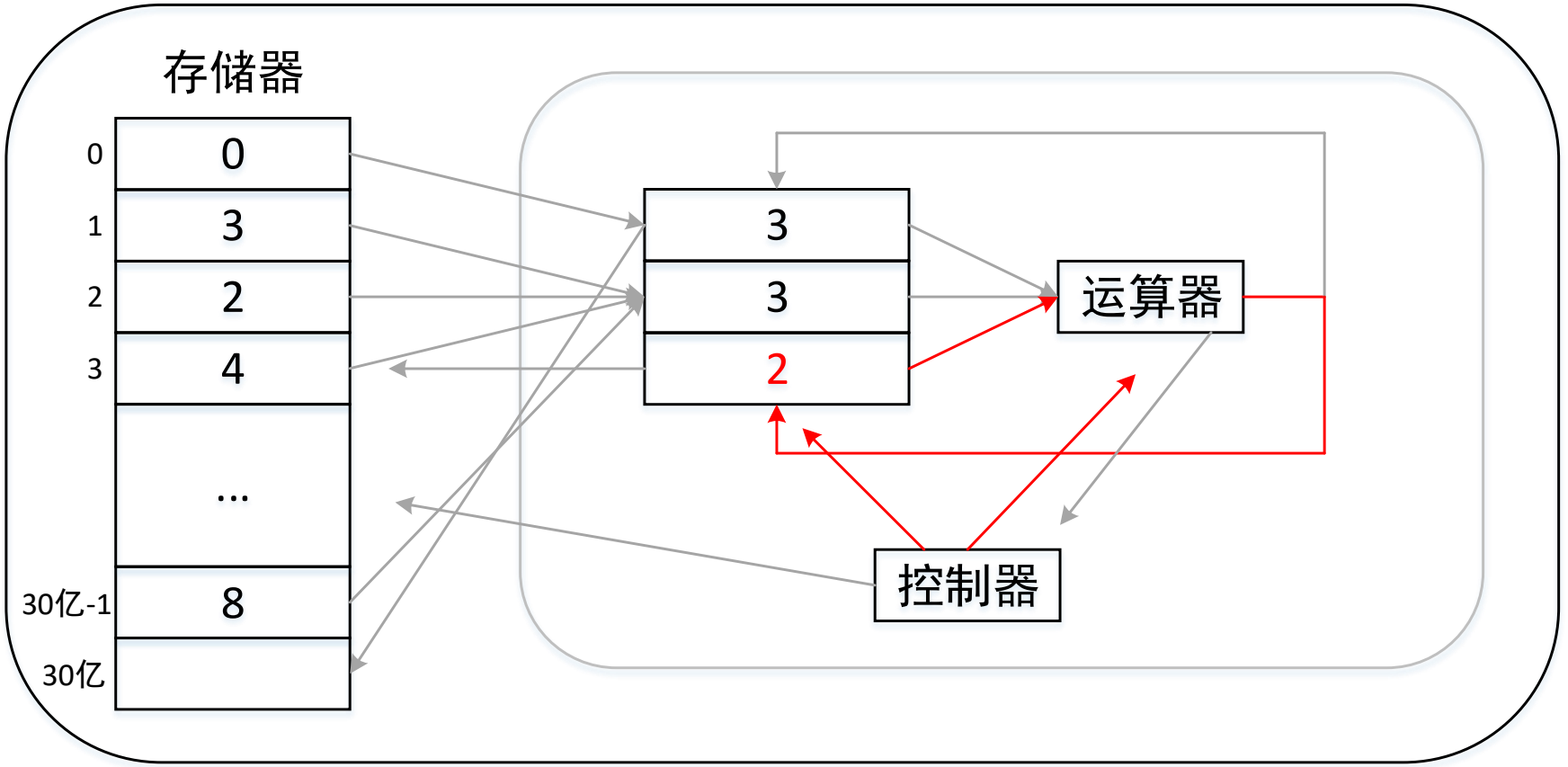
遍历第2个数，第3个数，…，第30亿个数：如果当前数大于记下的数，就用当前的数替换掉记下的数

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



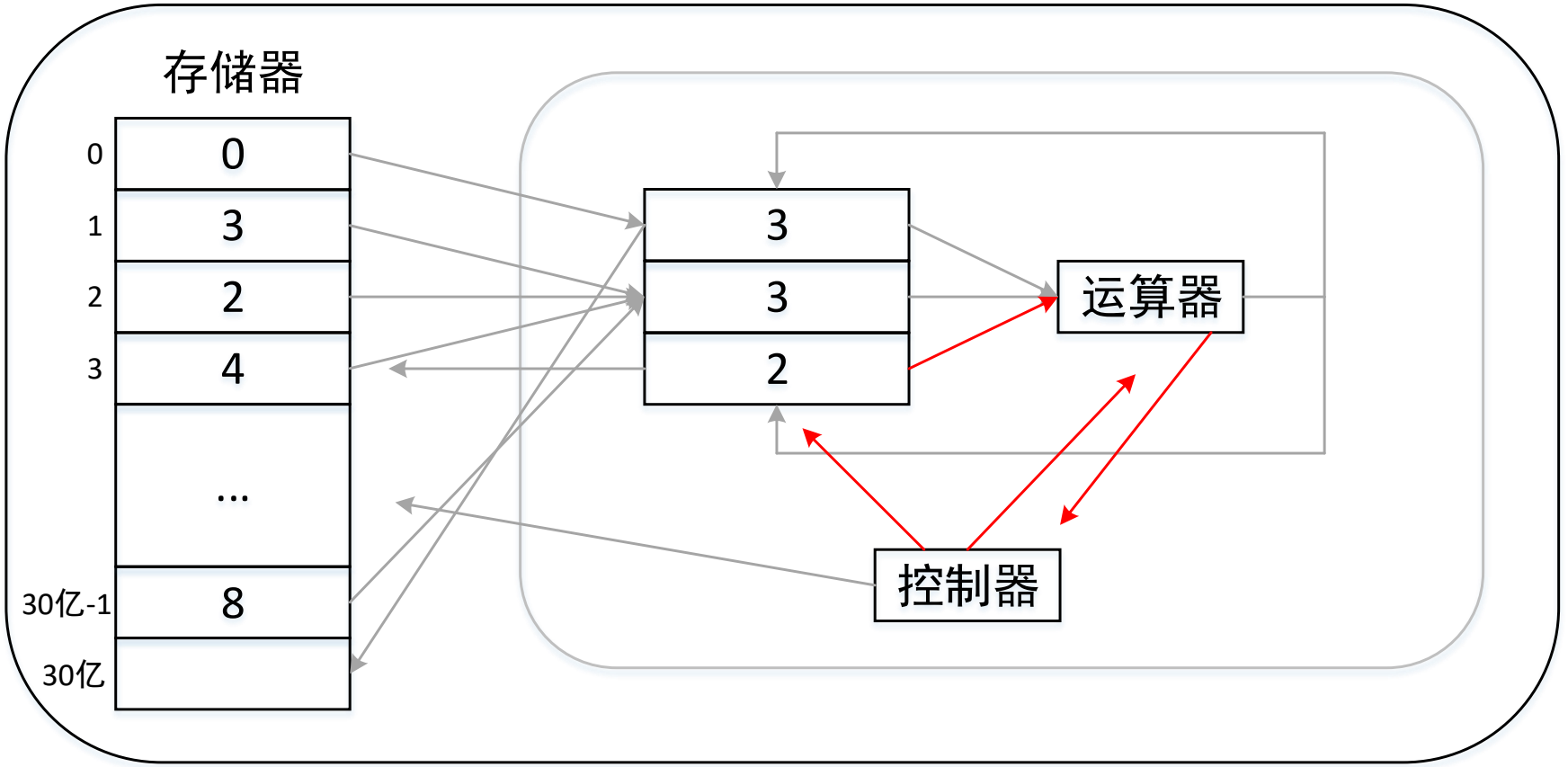
预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



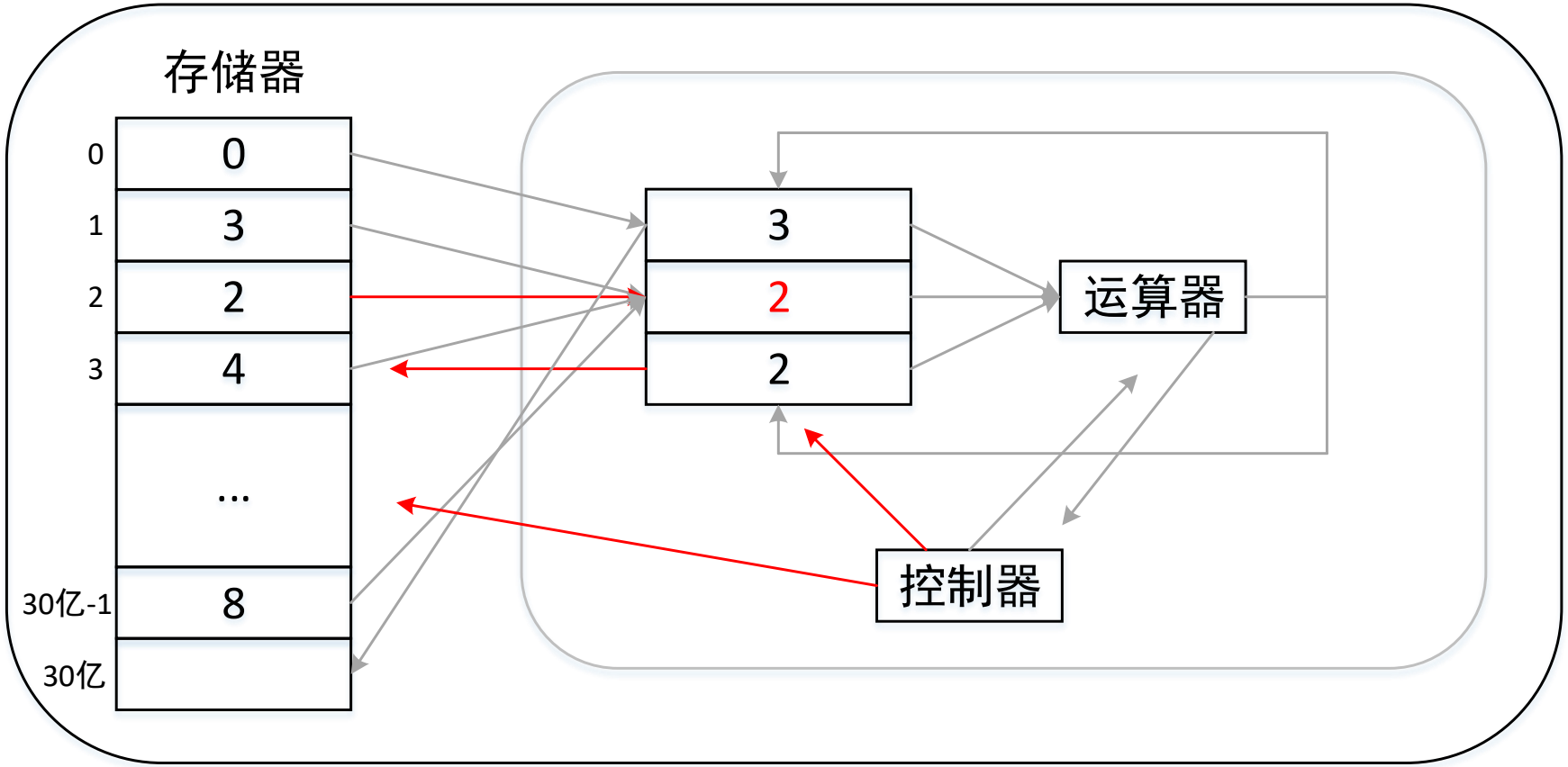
预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



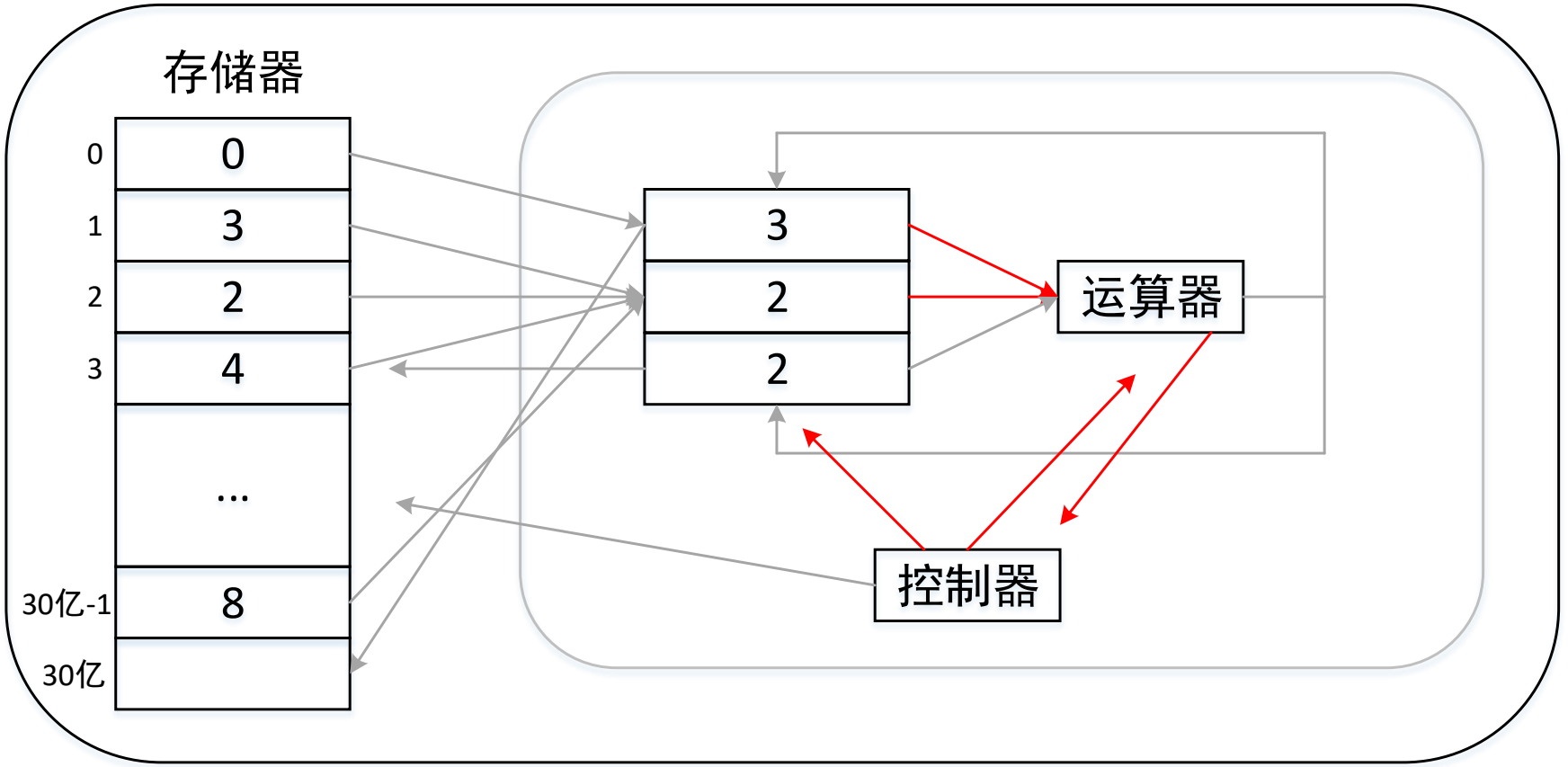
预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



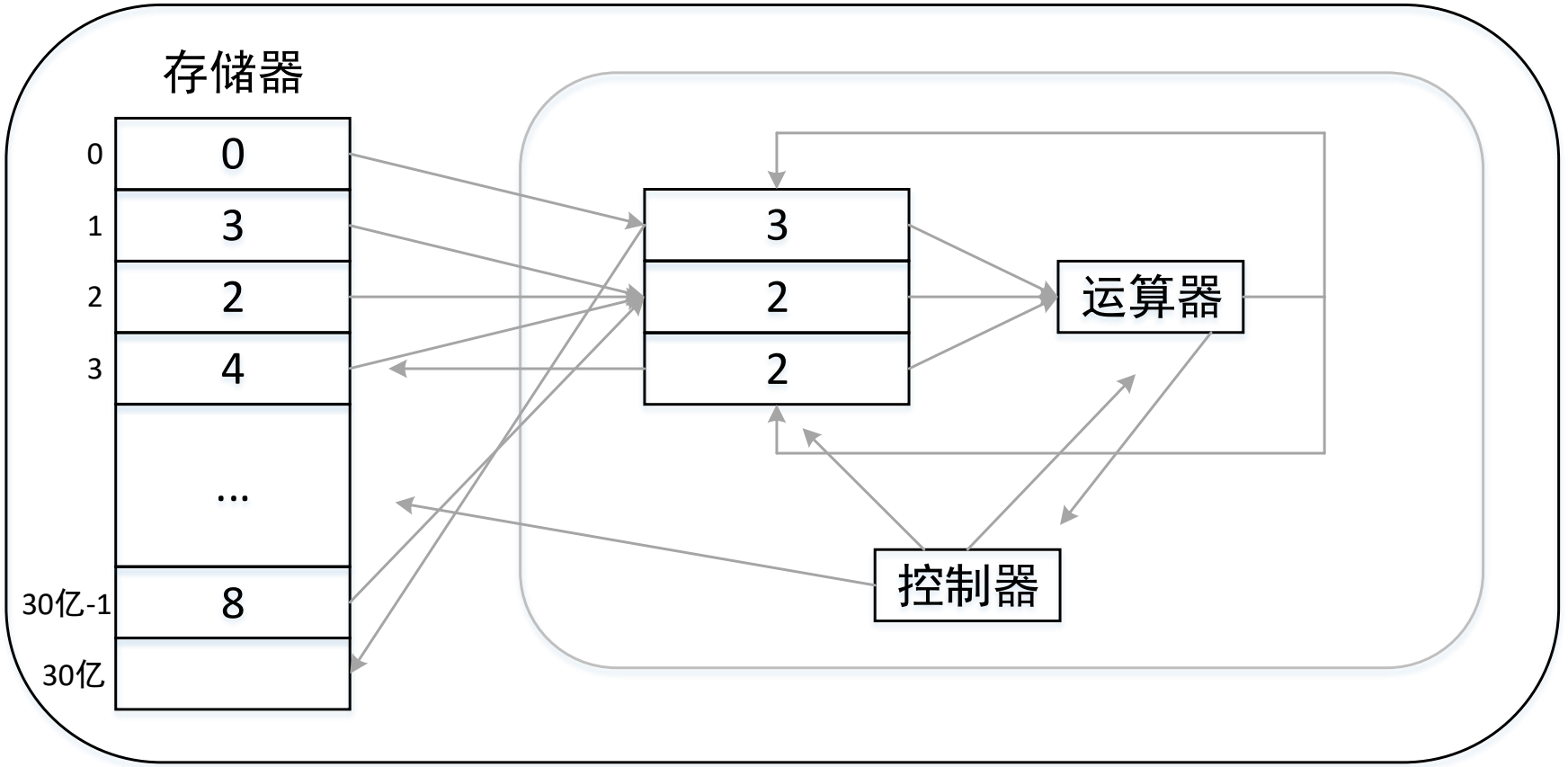
预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



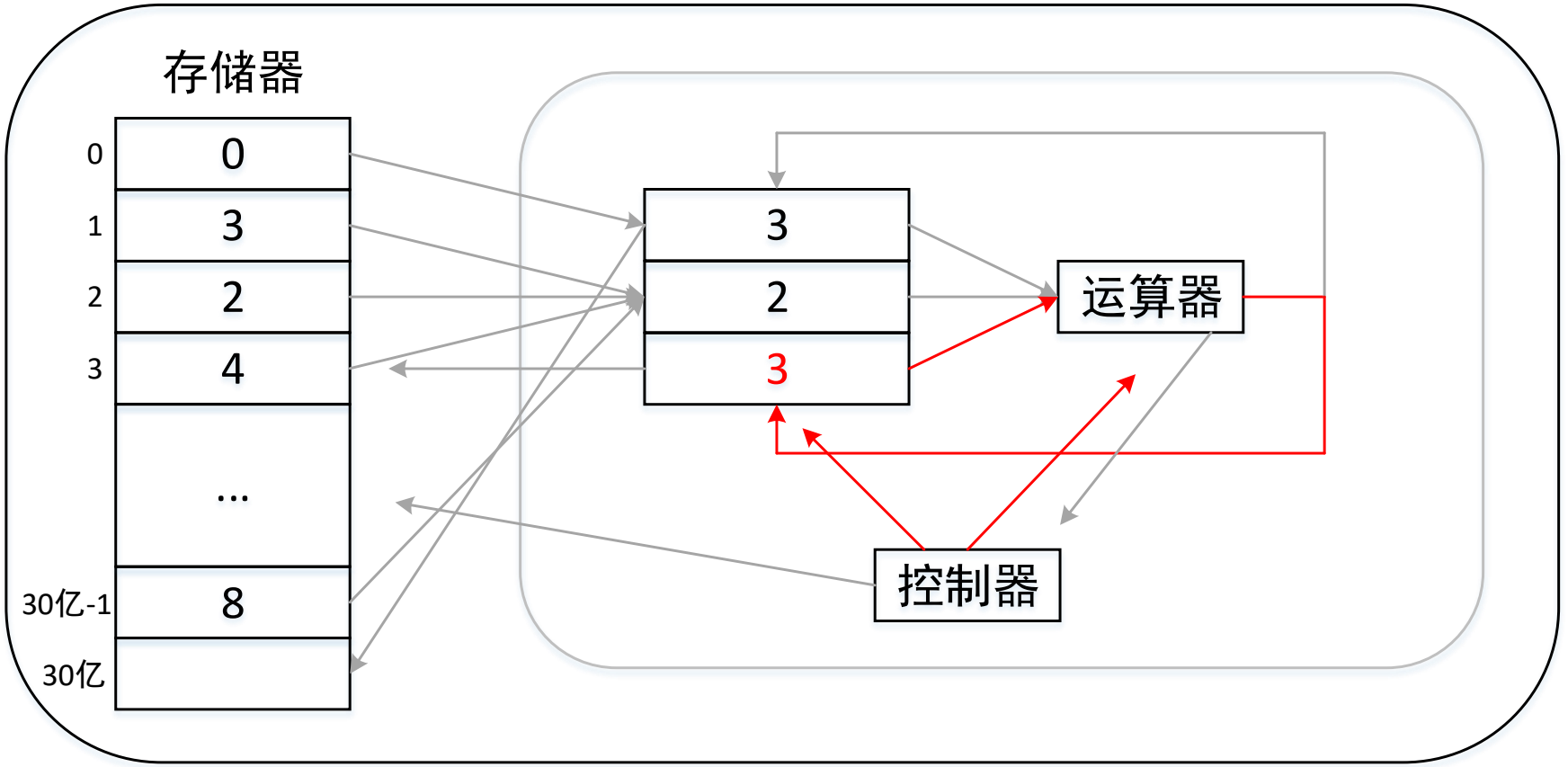
预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



预备知识

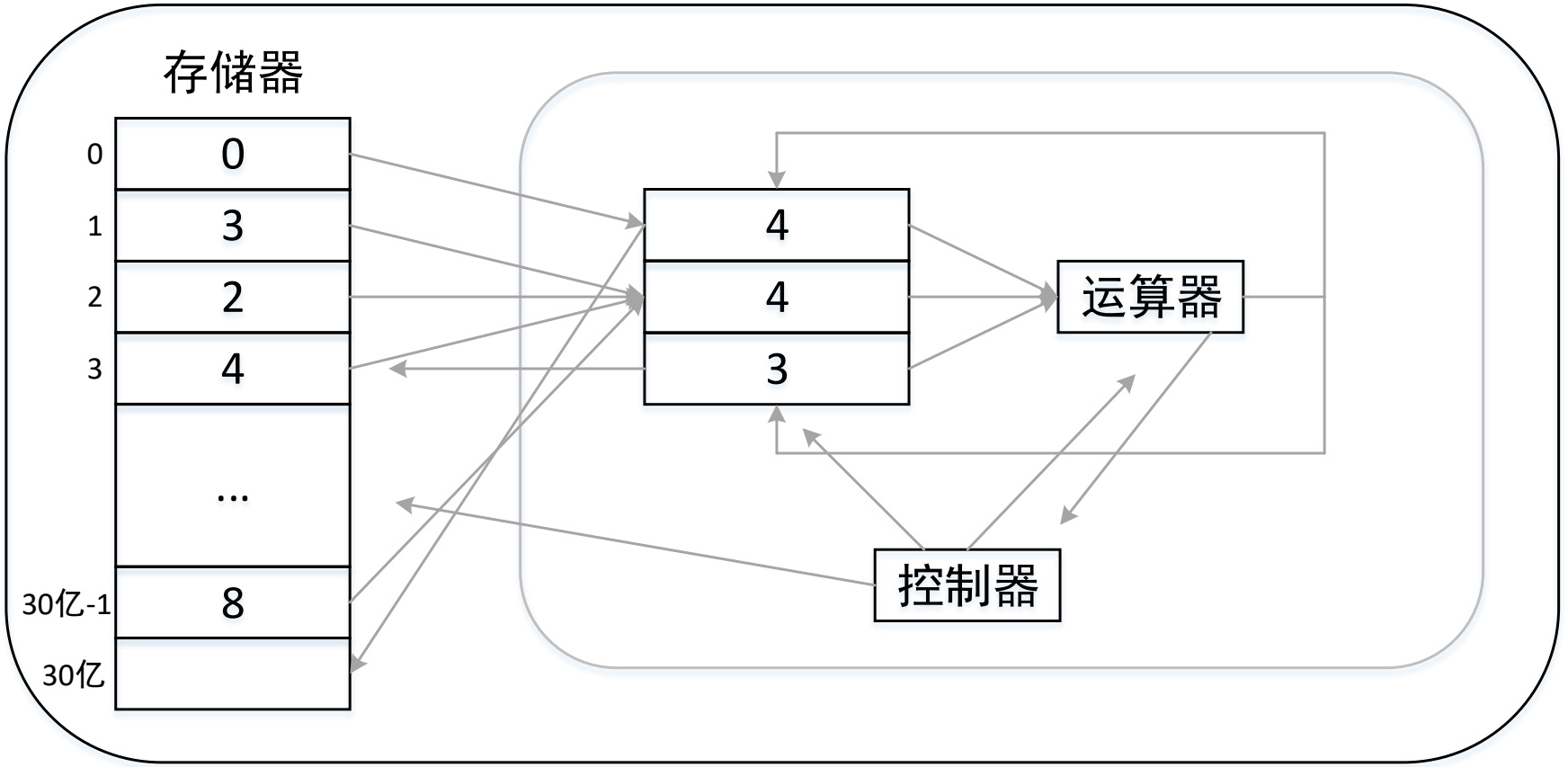
step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



预备知识

执行完step4-step7后

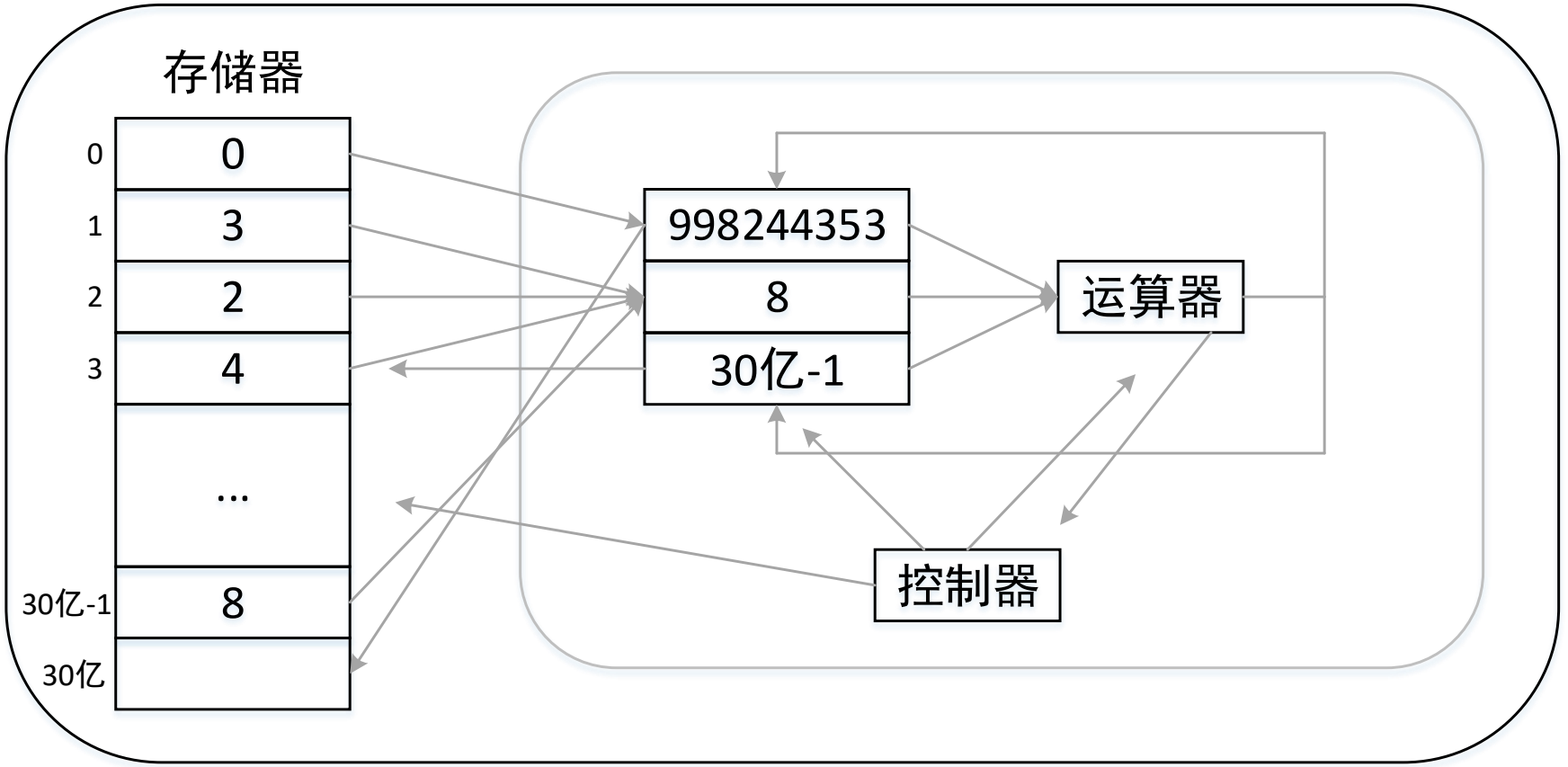
step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



预备知识

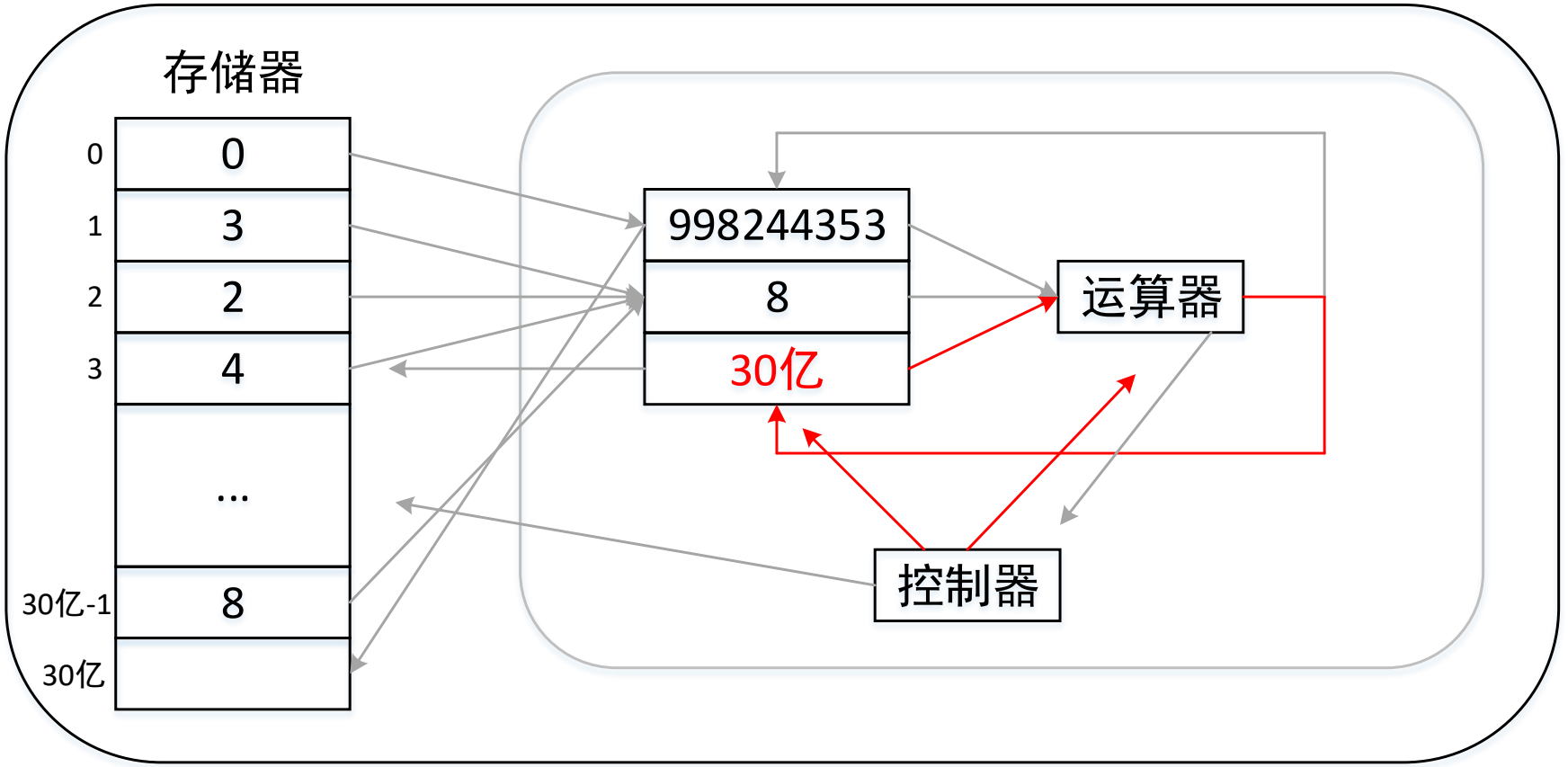
很久很久以后

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



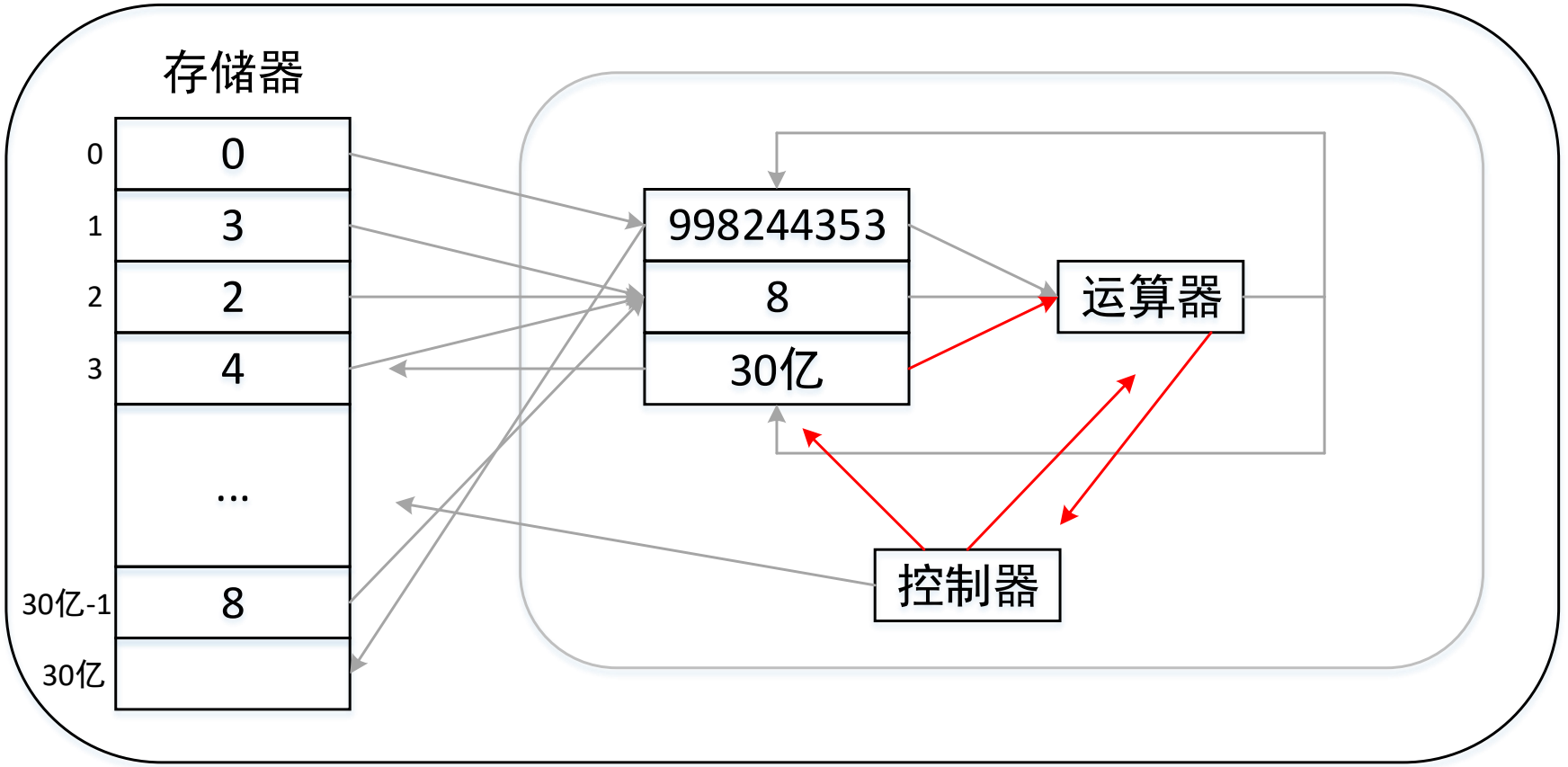
预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



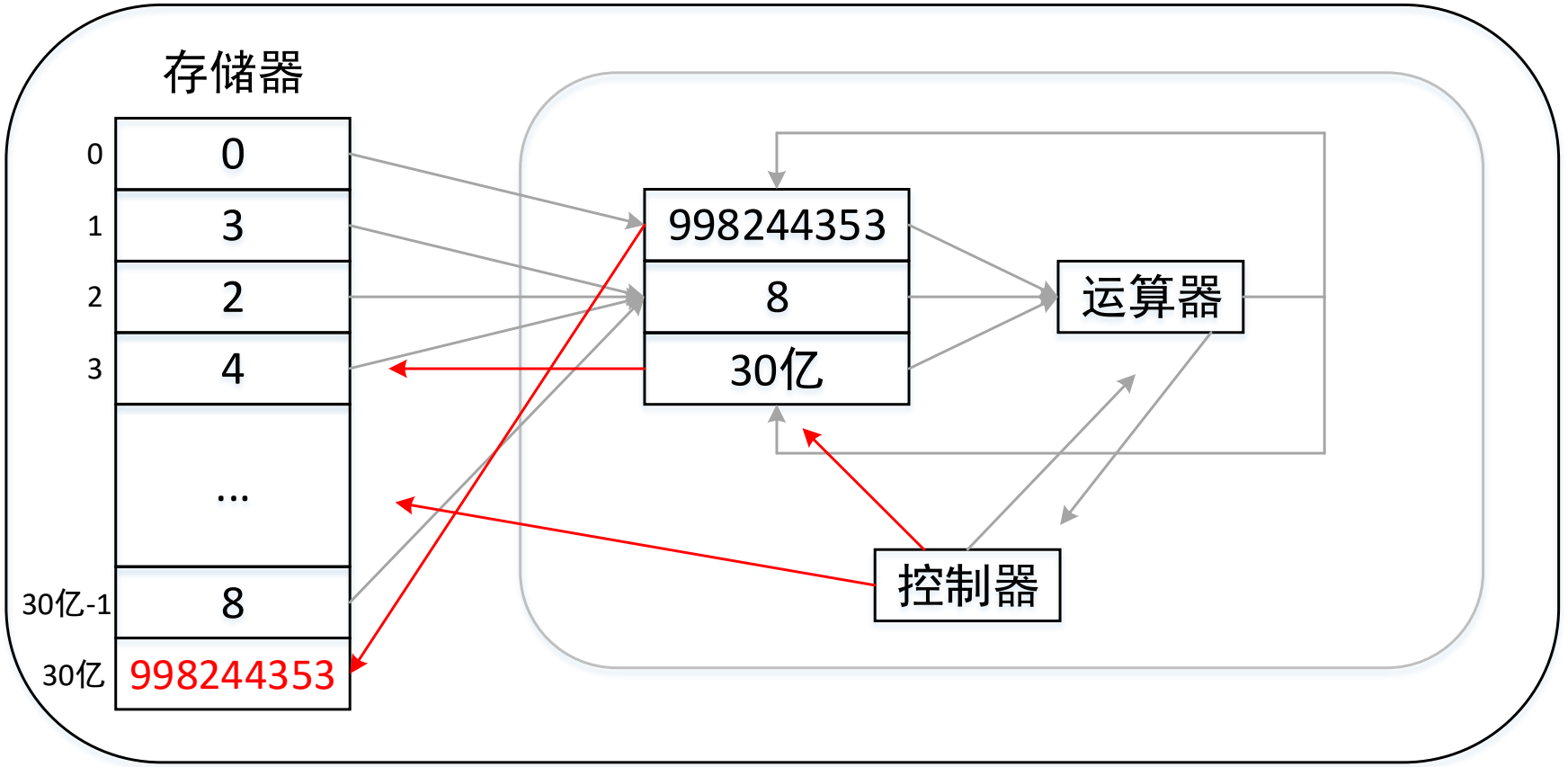
预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



预备知识

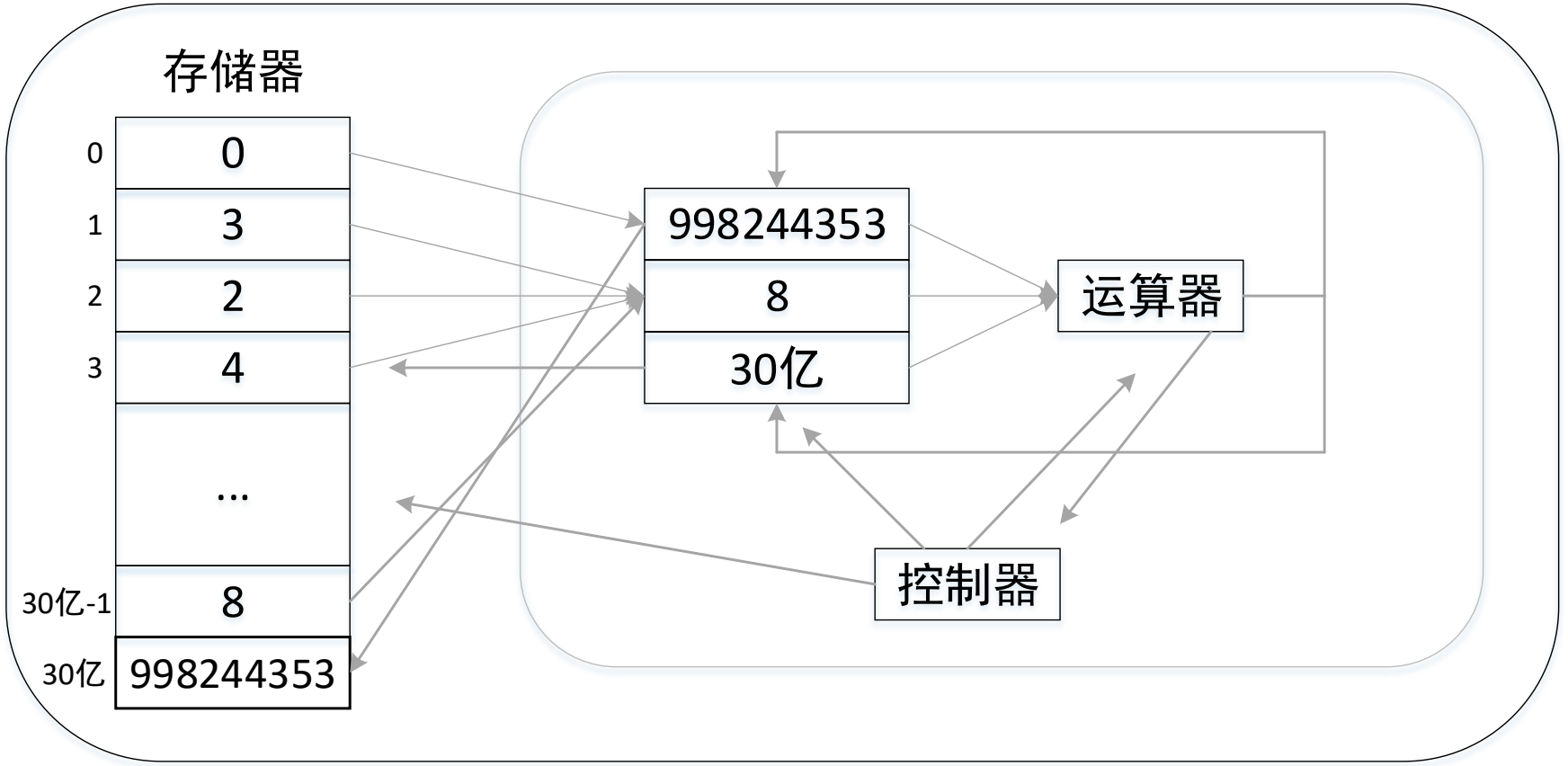
step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束



预备知识

step1: 让3号寄存器的数为0
step2: 以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
step3: 将3号寄存器的数加1
step4: 如果3号寄存器的数等于30亿，则跳转到step9继续执行
step5: 以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
step6: 如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
step7: 让1号寄存器的数等于2号寄存器的数
step8: 跳转到step3继续执行
step9: 以3号寄存器的数为地址，将1号寄存器的数存到存储器中
step10: 结束

遍历完成后记下的数就是最大数



预备知识

- 更加规范的指令
  - 问题1：计算机不理解自然语言
  - set　$3, 0　　让3号寄存器的数为0
  - load　$3, $1　　以3号寄存器的数为地址，将存储器中的数读到1号寄存器中
  - inc　$3　　将3号寄存器的数加1
  - jeqc　$3, 30亿, 9　如果3号寄存器的数等于30亿，则跳转到step9继续执行
  - load　$3, $2　　以3号寄存器的数为地址，将存储器中的数读到2号寄存器中
  - jge　$1, $2,　8　如果1号寄存器的数大于等于2号寄存器的数，则跳转到step8继续执行
  - mov　$2, $1　　让1号寄存器的数等于2号寄存器的数
  - j　3　　跳转到step3继续执行
  - store　$3, $1　　以3号寄存器的数为地址，将1号寄存器的数存到存储器中
  - halt　　结束

预备知识

- 更加规范的指令
  - 问题2：没说清控制器能识别哪些指令（计算机具体可以做哪些事情）
    - 比如会不会有指令：
    - judgehalt $(01101011...1)_2$
    - 表示判断二进制串 $(01101011...1)_2$ 对应的 <M,w> 是否有图灵机 M 在 w 上不可停机。
    - 停机问题是不可计算的，所以设计出的计算机肯定不允许有这样的指令。
    - 应该明确哪些指令是允许的。除此之外的指令认为是不合法的。

预备知识

- 更加规范的指令
  - 解决办法：设计指令集、指令规则。

load	寄存器 A，寄存器 B	以寄存器 A 的数为地址，将存储器中的数读到寄存器 B 中
store	寄存器 A，寄存器 B	以寄存器 A 的数为地址，将寄存器 B 的数存到存储器中
set	寄存器 A，常数 B	让寄存器 A 的数为 常数 B
inc	寄存器 A	将寄存器 B 的数加1
mov	寄存器 A，寄存器 B	让寄存器 B 的数等于 寄存器 A 的数
j	常数 A	跳转到第 A 条指令继续执行
jge	寄存器 A，寄存器 B，常数 C	如果寄存器 A 的数大于等于寄存器 B 的数，则跳转到第 C 条指令继续执行
jeqc	寄存器 A，常数 B，常数 C	如果寄存器 A 的数等于常数 B，则跳转到第 C 条指令继续执行
halt		结束

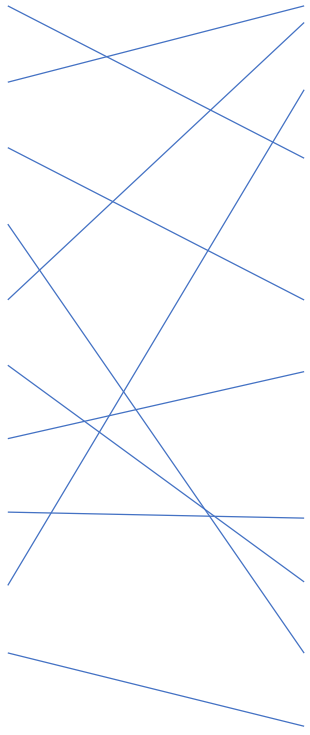
一条指令规则

所有 指令规则 构成 **指令集**

操作码　　操作数　　操作过程

预备知识

- 更加规范的指令
  - set $3, 0
  - load $3, $1
  - inc $3
  - jeqc $3, 30亿, 9
  - load $3, $2
  - jge $1, $2, 8
  - mov $2, $1
  - j 3
  - store $3, $1
  - halt



指令集（指令规则）：

load	寄存器 A,	寄存器 B	
store	寄存器 A,	寄存器 B	
set	寄存器 A,	常数 B	
inc	寄存器 A		
mov	寄存器 A,	寄存器 B	
j	常数 A		
jge	寄存器 A,	寄存器 B,	常数 C
jeqc	寄存器 A,	常数 B,	常数 C
halt			

预备知识

- 和图灵机转移规则的比较
  - 图灵机：
    - 每次根据当前状态和读到的字母去查找规则；
    - 规则之间没有顺序！
  - 指令：
    - 执行的时候用的是指令（也称指令规则实例）序列；
    - 指令序列中的顺序有意义！

内容大纲

- 实验目的
- 知识回顾
- 预备知识
- 实验内容
- 评分标准

实验内容

- 设计一台（由人+zoom平台组成的）可对同学输入数据进行排序的班级排序计算机！
- 实验步骤：
- 1. 设计硬件和指令集；
- 2. 把快速排序算法用指令写出来；
- 3. 模拟执行这些指令，完成对同学们持有数据的快速排序。

实验内容

- 具体实验步骤：
  - 1.1硬件：
    - 应至少包含存储器和控制器。可额外设计其他硬件。
    - 这些硬件均要可由人体或其他简易道具实现。

    - 存储器：至少含有1个存储器，该存储器只含参与排序的同学。
    - 控制器：应由一名同学充当，负责根据指令控制其他部件并确定下一条指令。
    - 寄存器：可有可无，看怎么设计，可在zoom平台共享方格来指代。
    - 运算器：需要单独设计出来。
    - 监督器：由其他小组同学担任，用于监督控制器是否正常运行指令（是否作弊），并统计运算器执行步数及排序总时间。

实验内容

- 具体实验步骤：
  - 1.1硬件：
    - 应至少包含存储器和控制器。可额外设计其他硬件。
    - 这些硬件均要可由人体或其他简易道具实现。

    - 硬件设计方案格式要求：

硬件名称	组成	职责/能力
控制器	一名同学	负责分析指令，控制其他部件并确定下一条指令。
…	…	…

实验内容

- 具体实验步骤：
  - 1.2指令集：
    - 同学们需根据自己设计的班级排序计算机硬件来相应地设计指令规则。
    - 指令规则格式应按照前面电子计算机的指令规则格式来设计。

操作码	操作数1	操作数2	操作数3	操作数4	操作过程
jge	寄存器A	寄存器B	常数C		如果寄存器A的数大于等于寄存器B的数，则跳转到第C条指令继续执行。
…	…	…			…

实验内容

- 具体实验步骤：
  - 1.2指令集：
    - 要注意指令集和硬件是相辅相成的。
    - 指令集到底设计成什么样，取决于你硬件能做成什么样，有什么能力。

    - 人及zoom平台 VS 电子元器件？

    - 比如定义这样人相对于电子元器件有优势的一条规则：
    - label j, ‘a’
    - 操作过程：从存储器的第j号地址开始递增，找到第一个未标记的地址，并标记为’a’。
    - 指令： label 5, #

实验内容

- 具体实验步骤：
  - 2. 根据指令集写出快速排序指令序列
    - 同学们需要根据自己设计的指令集，构建一个指令序列，最终能在自己设计的班级排序计算机上完成快速排序。
    - 可以做适当假设，比如参与排序的同学个数可认为存放在某个自己指定的位置。
    - 此外，设计的指令序列需要保证以下三个正确性：
      - 结果正确（排序结果是正确的）；
      - 算法正确（正确执行了快速排序算法）；
      - 系统正确（确实是串行执行而不是并行执行的）。

实验内容

- 具体实验步骤：
  - 3. 根据硬件、指令集和快速排序指令序列在zoom平台上进行实践
    - 在“班级快速排序实验”的第三次课程中，助教将组织同学们在zoom上进行实践。具体流程如下：
      - 每2-3个组结成一个大组，共形成3-4个大组，在zoom的不同分会场进行实践。
      - 每个大组依次使用各个小组的硬件、指令集和指令序列进行1次排序。
      - 使用某个小组的方案时，该小组中出若干同学充当控制器以及必要时充当其他硬件，同时由组长兼任监控器，录像记录排序过程。再从非该小组的同学中选一人作为监督器，监督控制器是否按照该小组的设计方案按序执行指令。其余同学充当待排序数据（7-10人，若3个组构成一个大组，可通过组内商讨最终每人执行3轮中的2轮）。
      - 监督器要记录下整个过程的运行步数及时长，待排序过程结束后汇报给该小组成员记录下来。
      - 同学记录数据，课后用于分析。
      - 注意：快排过程中需要产生随机数，因此设计中应指明由哪个部件产生随机数。

往年实验内容回顾

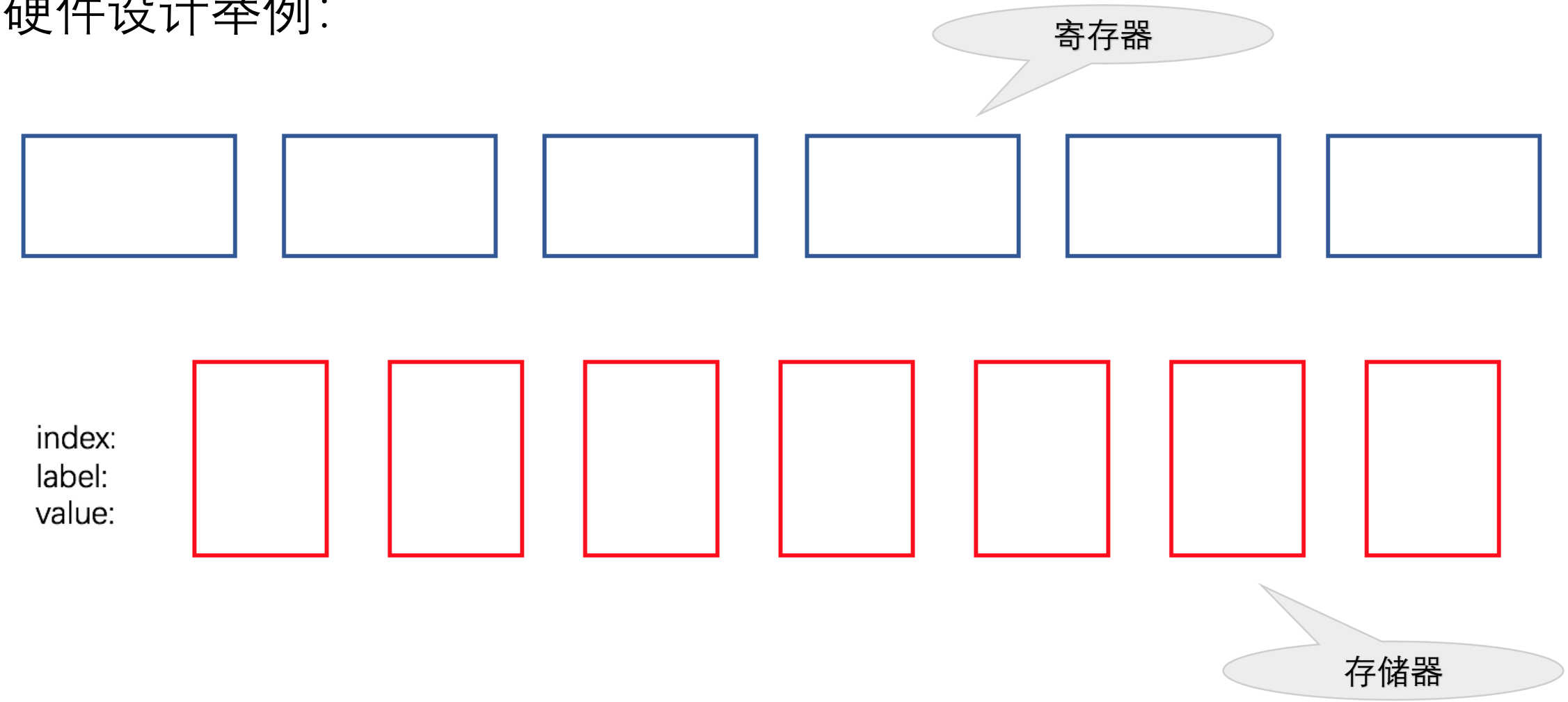
往年实验内容回顾

Input data: ordered by students' ID numbers　输入数据：同学们按学号排列

Output data: ordered by students' heights　　输出数据：同学们按身高排列

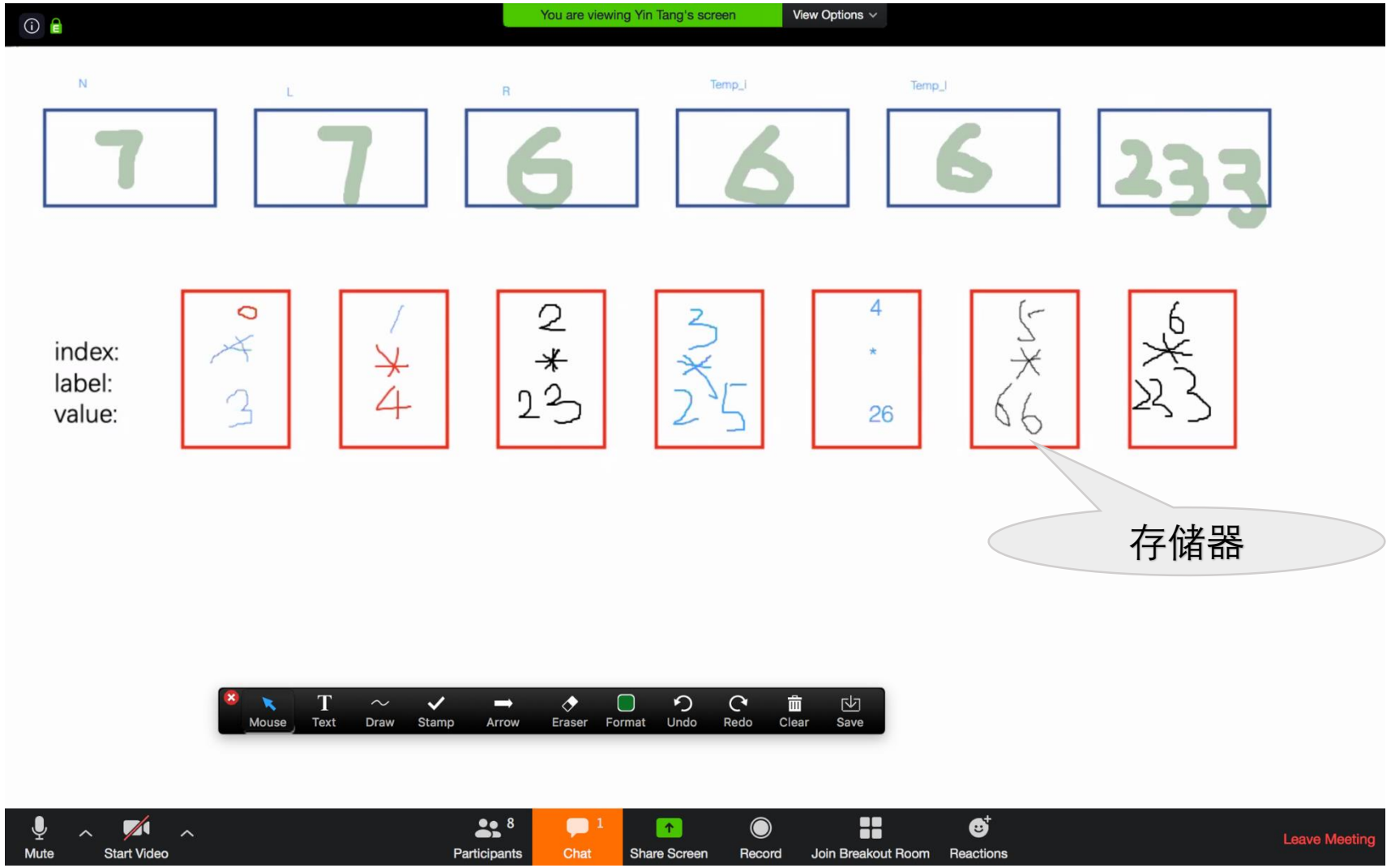
实验内容

硬件设计举例:



实验内容

排序过程举例：


You are viewing Yin Tang's screen
View Options
N
L
R
Temp_l
Temp_l
index:
label:
value:
存储器
Mouse
Text
Draw
Stamp
Arrow
Eraser
Format
Undo
Redo
Clear
Save
Mute
Start Video
Participants
Chat
Share Screen
Record
Join Breakout Room
Reactions
Leave Meeting

实验内容

- 具体实验步骤：
  - 4. 在理论课上对结果进行汇报
    - 先从每个大组的2-3个小组中先评选出一个最好方案，再由助教选择3-4个大组中较优的方案。设计该方案的小组于次周的理论课上（6月5日）进行汇报。
    - 评优依据为：
      - 先根据3个正确性进行比较；
      - 执行过程应遵从设计方案；
      - 思路新奇。

实验内容

- 实验注意事项：
  - 注意第三次实验课程前需尽量熟悉zoom的一些共享及书写操作，控制器和运算器要熟悉指令规则，以便节约排序时间。
  - 在第三次课开始前，每个小组应明确在执行你们的设计时每个人的分工，同时选出作为监督前一个小组的监督器。

实验内容

- 划分过程实现思路：
  - 思路1：两头走。

实验内容

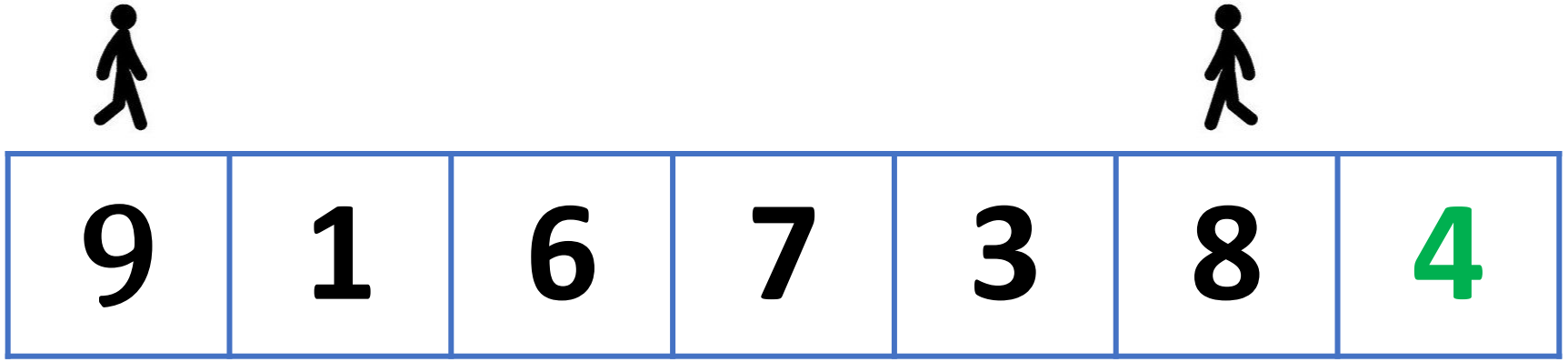
- 划分过程实现思路：
  - 思路1：两头走。

实验内容

- 划分过程实现思路：
  - 思路1：两头走。

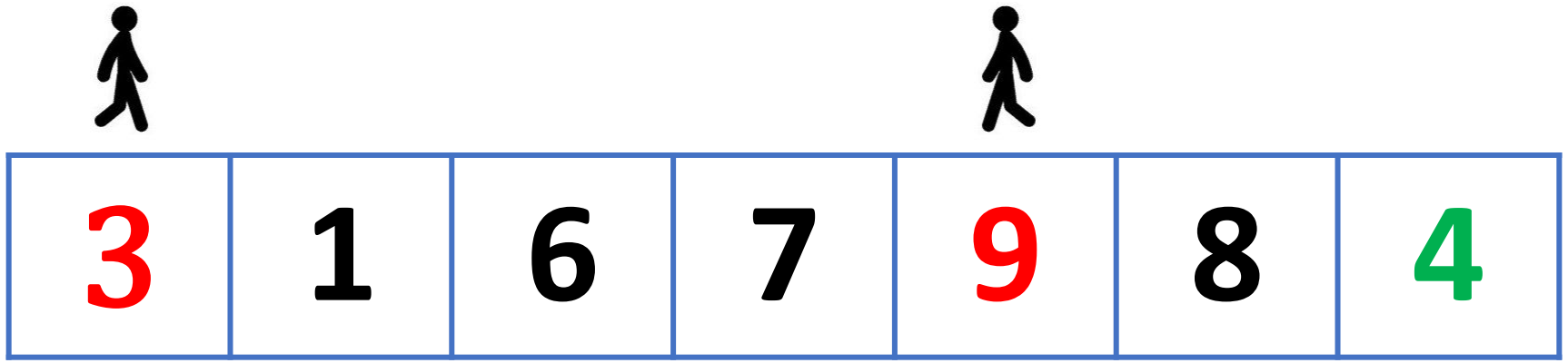
实验内容

- 划分过程实现思路：
  - 思路1：两头走。



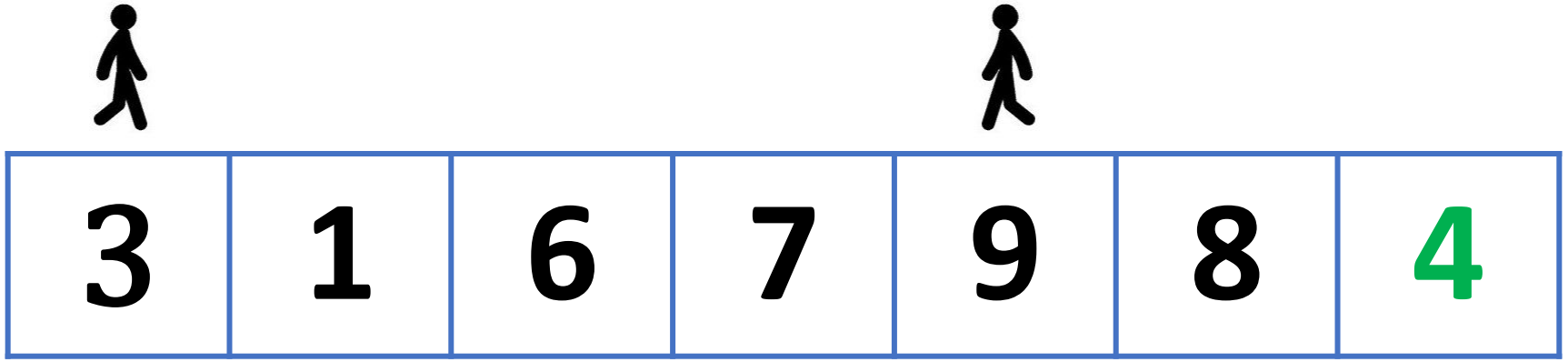
实验内容

- 划分过程实现思路：
  - 思路1：两头走。



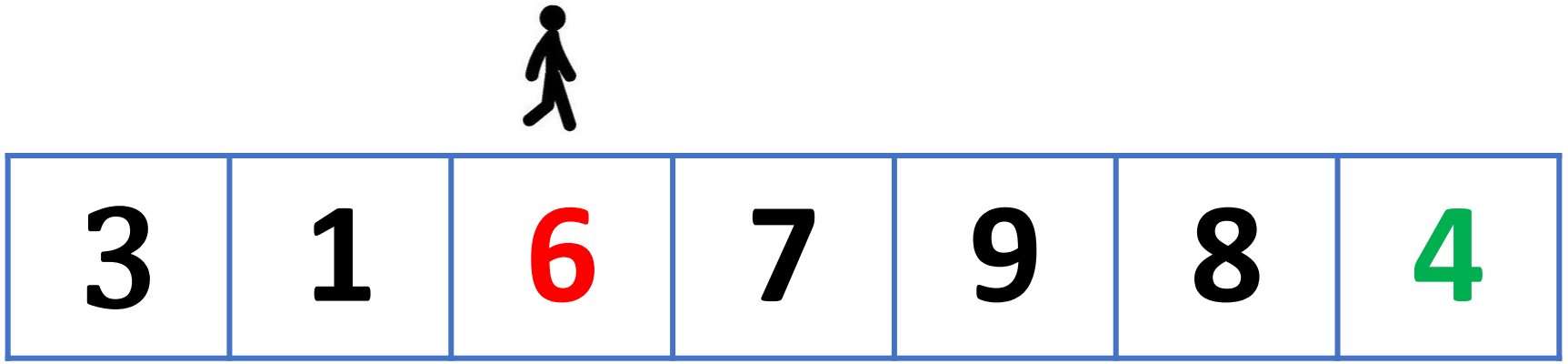
实验内容

- 划分过程实现思路：
  - 思路1：两头走。



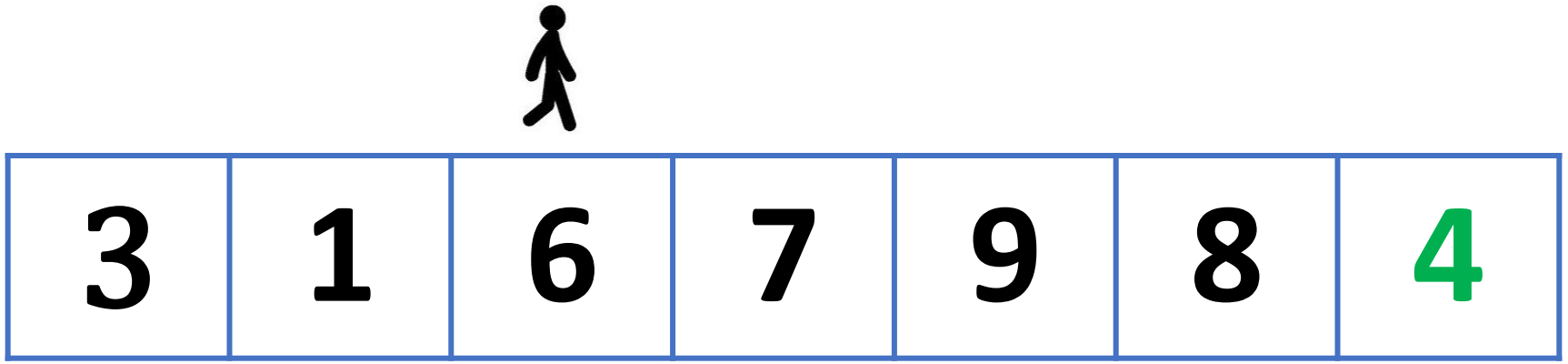
实验内容

- 划分过程实现思路：
  - 思路1：两头走。



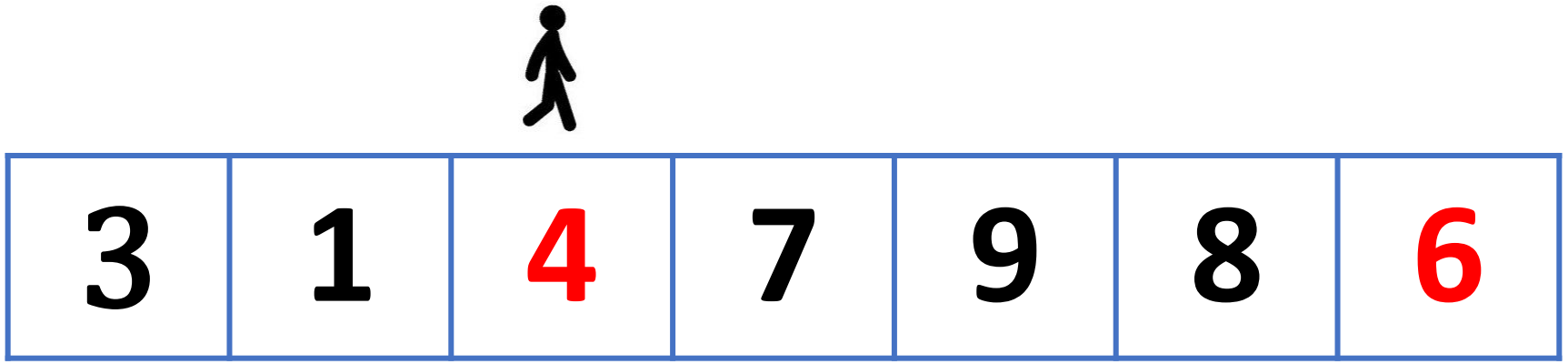
实验内容

- 划分过程实现思路：
  - 思路1：两头走。



实验内容

- 划分过程实现思路：
  - 思路1：两头走。



实验内容

- 划分过程实现思路：
  - 思路1：两头走。

实验内容

- 划分过程实现思路：
  - 思路2：从左到右扫一遍，发现小于等于标杆的就放在“左边”。

实验内容

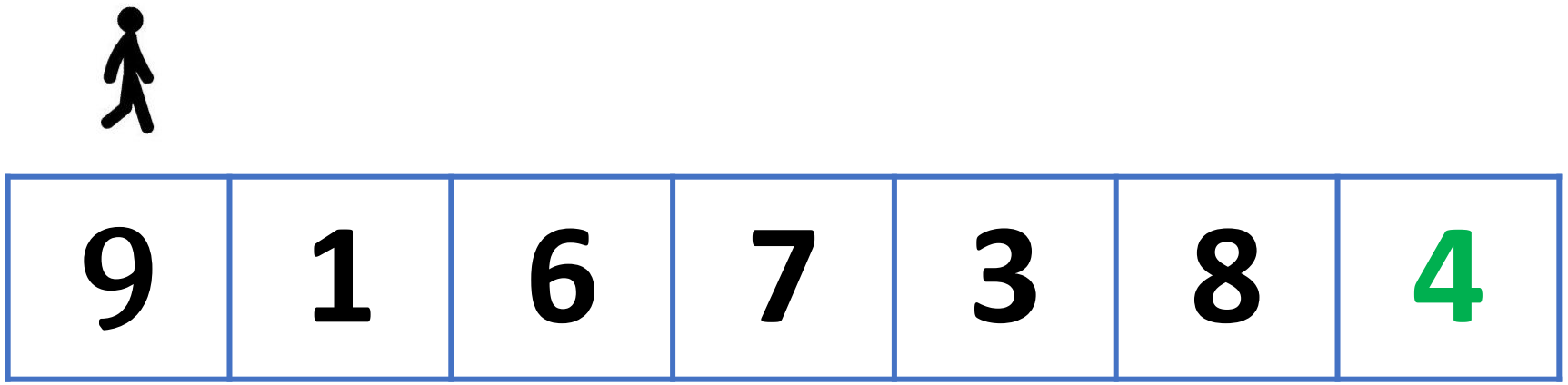
- 划分过程实现思路：
  - 思路2：从左到右扫一遍，发现小于等于标杆的就放在“左边”。

实验内容

- 划分过程实现思路：
  - 思路2：从左到右扫一遍，发现小于等于标杆的就放在“左边”。

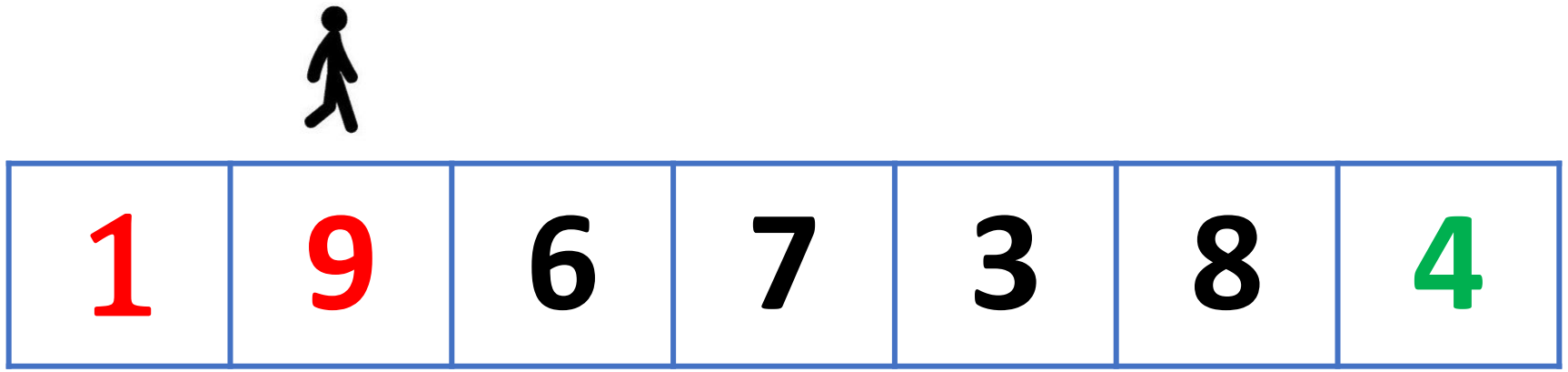
实验内容

- 划分过程实现思路：
  - 思路2：从左到右扫一遍，发现小于等于标杆的就放在“左边”。



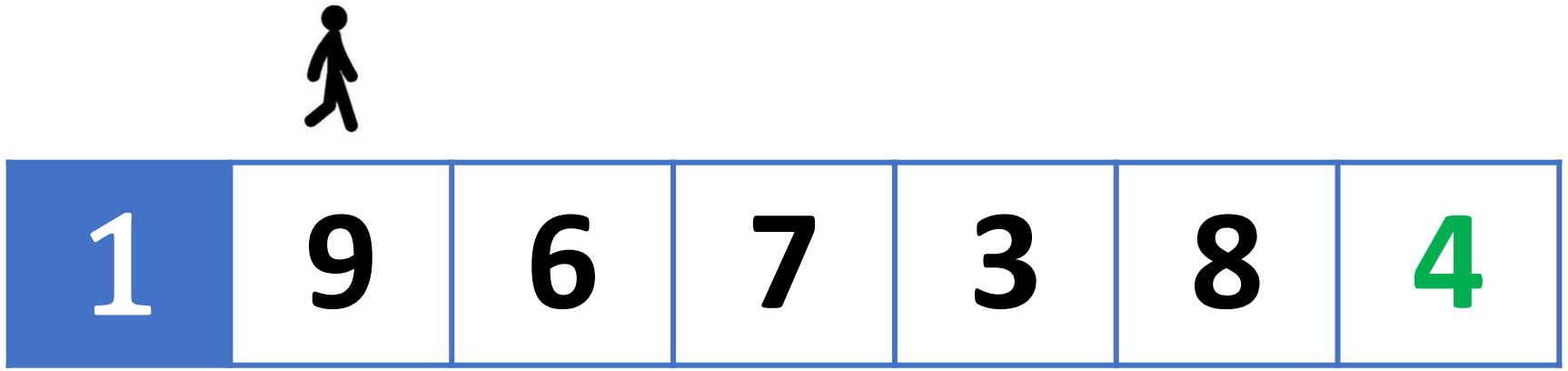
实验内容

- 划分过程实现思路：
  - 思路2：从左到右扫一遍，发现小于等于标杆的就放在“左边”。



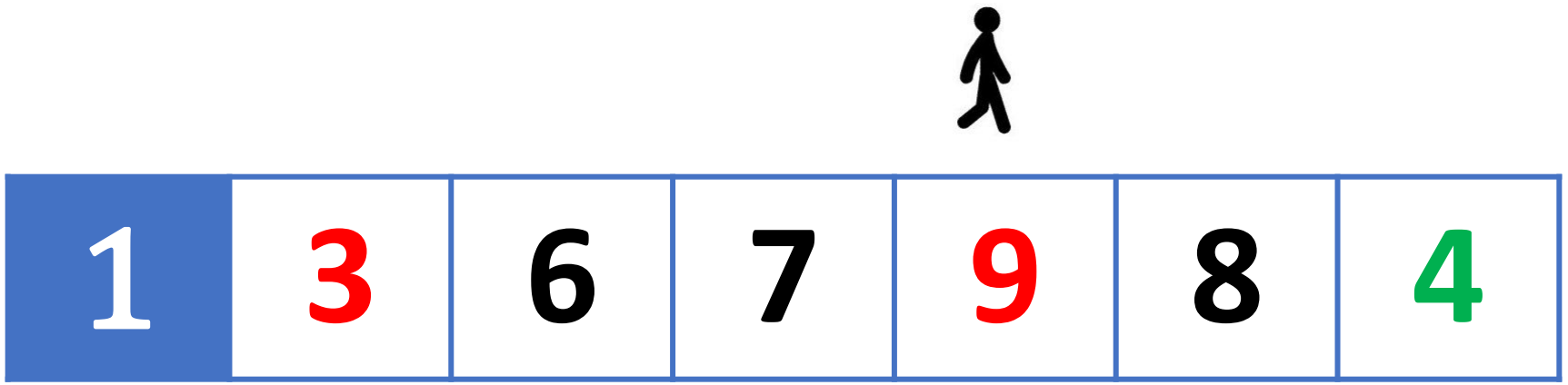
实验内容

- 划分过程实现思路：
  - 思路2：从左到右扫一遍，发现小于等于标杆的就放在“左边”。



实验内容

- 划分过程实现思路：
  - 思路2：从左到右扫一遍，发现小于等于标杆的就放在“左边”。



实验内容

- 划分过程实现思路：
  - 思路2：从左到右扫一遍，发现小于等于标杆的就放在“左边”。

实验内容

- 划分过程实现思路：
  - 思路2：从左到右扫一遍，发现小于等于标杆的就放在“左边”。

实验内容

- 划分过程实现思路：
  - 思路2：从左到右扫一遍，发现小于等于标杆的就放在“左边”。

内容大纲

- 实验目的
- 知识回顾
- 预备知识
- 实验内容
- 评分标准

评分标准

- 个人设计方案（一人一份） （40%）
  - 实验课助教提供设计方案文档，每个人独立完成自己的设计方案，并按时提交。
- PPT 展示成绩 （20%）
- 小组设计方案（一组一份）+zoom平台实验结果成绩 （ 20%）
  - 同学们进行讨论权衡利弊，每小组确定一份zoom平台实践拟用的设计方案，同时按照文档中的要求填写设计方案，并按时提交。
- 实验报告成绩 （20%）
- 额外加分上限5分
  - 课堂提问及补充
  - 被选定理论课上台展示的小组
    - 展示内容：第二次实验课上展示的内容，及对于实践的反思与总结

PPT展示内容

- PPT 展示于第二次课进行。每个小组10分钟。
- 展示应包含如下内容：
  - 硬件设计、指令集设计；
  - 设计指令序列的思路（实现划分的大致思路及某些细节如何处理）。
- 思考题的思考

思考题

- 自动计算：
  - 思考你们设计的班级排序计算机是不是图灵机？
  - 思考你们设计的班级排序计算机是否是冯诺依曼体系结构的。
  - 思考你们在实际排序过程中发生的问题电子计算机中是否会发生？如果不会，想想电子计算机为什么能避免这些错误。
  - 分析你们设计的班级排序计算机的各个硬件和电子计算机对应部件在能力上有什么区别。
- 排序算法
  - 思考如果需要排序的数据有相等的元素，你们的算法是否依旧能保证期望复杂性$O(n \log n)$?
  - 思考是否有基于比较的$o(n \log n)$时间的算法？除了快排外你还知道哪些$O(n \log n)$时间的算法？它们和快排相比有何优劣？

实验报告内容

- 实验报告（每人一份）应包含如下内容：
  - 设计指令序列的思路；
  - 人员分工及实验结果记录；
  - 实验结果分析（包括但不仅限于排序过程执行步数、产生错误的原因以及解决方法）；
  - 三个正确性的证明；
  - 对于思考题的思考。

重要的时间节点

- 班级排序实验上课时间：2020-5-15, 2020-5-22, 2020-5-29
- 个人设计方案及PPT提交截止时间为：2020-5-22 13:00
- 小组设计方案提交截止时间为：2020-5-29 13:00
- 理论课展示PPT提交（每班1个组）截止时间为：2020-6-4 13:00
- 实验报告提交截止时间为：2020-6-5 13:00

谢谢！

教学团队

姓名	负责工作	电子邮箱
徐志伟	主讲教授	zxu@ict.ac.cn
张家琳	主讲教授	zhangjialin@ict.ac.cn
唐寅	实验课助教 (1班, 5班)	tangyin16@mails.ucas.ac.cn
吴步娇	实验课助教 (2班, 6班)	wubujiao@ict.ac.cn
田国敬	实验课助教 (3班, 7班)	tianguojing@ict.ac.cn
郭泓锐	实验课助教 (4班, 8班)	guohongrui17@mails.ucas.ac.cn
何啸宇	答疑助教	hexiaoyu18s@ict.ac.cn
李龙成	答疑助教	lilongcheng18@mails.ucas.ac.cn
刘敖	答疑助教	liuao18@mails.ucas.ac.cn
徐泽凡	答疑助教	xuzefan18@mails.ucas.ac.cn