第13章 数据流计算机

张晨曦 刘依

www.GotoSchool.net

xzhang2000@sohu.com

13.1 数据流计算机的基本原理

13.1.1 数据驱动原理

1. 数据流计算机

- 采用数据驱动方式工作，没有程序计数器，没有常规的变量概念。
- 指令是在数据可用性的控制下并行执行的。

2. 基本原理

- 当且仅当指令所需要的数据可用时，该指令即可执行。

- 指令操作的异步性和操作结果的确定性。

  （数据流计算机所特有 ）

- 完全摆脱外界强加于它的控制，指令在数据可用性驱动下并行执行。

➢ 任何操作都是纯函数操作。

- 不设置状态，在指令之间直接传送数据，不改变机器状态。

  （具有纯函数的特点）

- 不仅有利于开发程序中各级的并行性，而且也有利于改善软件环境，提高软件的生产力。

1. 数据流计算机中的驱动方式

- 数据驱动计算
  - 一种提前计算的策略
- 需求驱动计算（需求驱动方式 ）
  - 只在当某一个函数需要用到某一个自变量时才驱动对该自变量的求值操作。
  - 按需求值，是一种滞后计算的策略。
- 两者的比较
  - 需求驱动方式可以减少许多不必要的操作，有助于提高机器的效率，但实现更为困难。

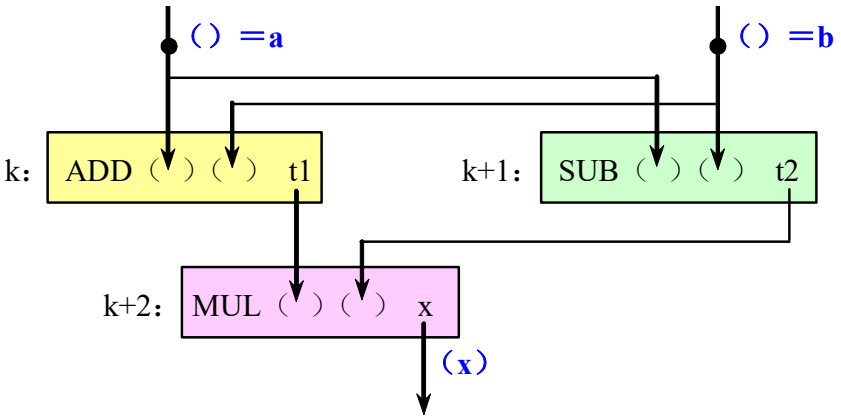
目前大多数数据流计算机都是采用数据驱动方式。

13.1.2 数据流计算机中指令的执行过程

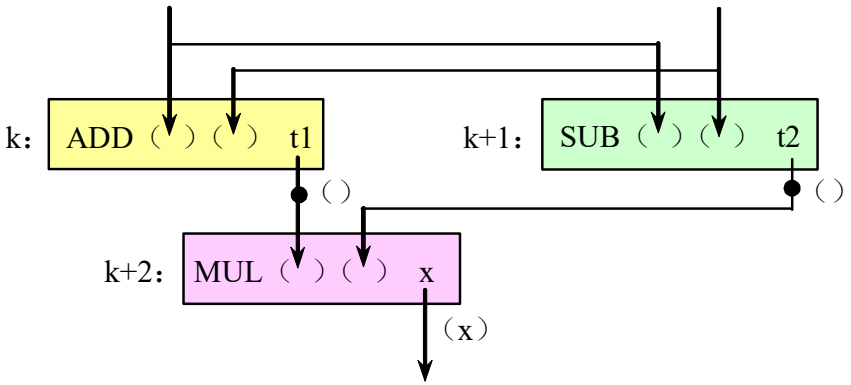
数据流计算机：

- 用数据令牌传送数据并激活指令。
- 用一种有向图来表示数据流程序。
- 一条指令的组成
  - 一个操作码
  - 一个或几个操作数
  - 一个或几个后继指令地址：用于把本指令的执行结果送往需要这个数据的指令中。

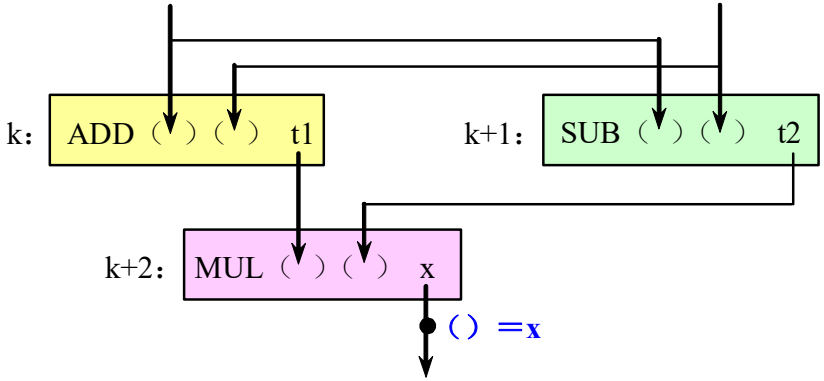
1. 计算函数z＝(a＋b)×(a－b) 指令的执行过程



（a）第一步



（b）第二步



（c）第三步

“.”：数据令牌

（）：数据令牌所携带的操作数

1. 数据驱动具有的特性

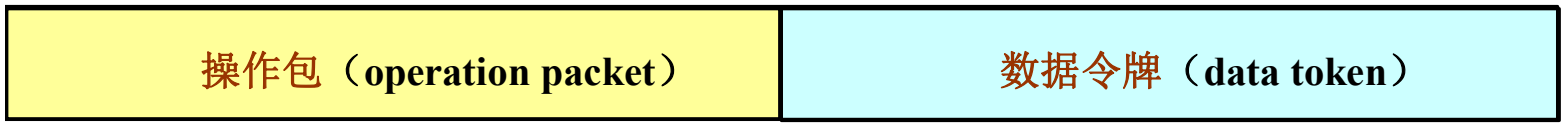
- 异步性：只要指令所需的数据令牌都已到达，指令即可独立地开始执行，而不必关心其他指令及数据的情况。
- 并行性：可同时并行执行多条指令。
- 函数性：由于不使用共享的数据存储单元，所以数据流程序不会产生诸如改变存储字这样的副作用。
- 局部性：运算过程中所产生的数据不是用操作数的地址来引用，而是作为数据令牌直接传送。

运算具有局部性。

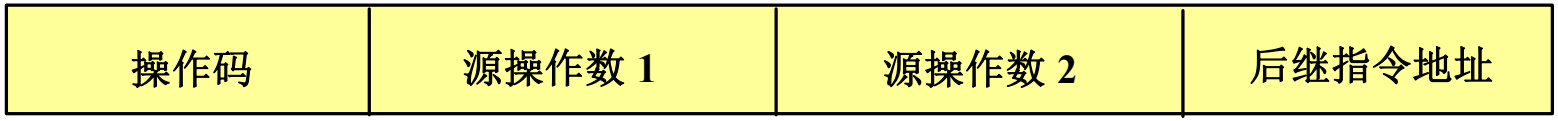
13.1.3 数据流计算机的指令结构

指令主要由两部分组成：操作包、数据令牌

- 操作包：由操作码、一个或几个源操作数以及一个或几个后继指令地址组成。
- 数据令牌
  - 由结果数值和目标地址等组成。
    - 结果数值：上条指令的运算结果
    - 目标地址：直接取自上条指令的后继指令地址
  - 如果一条指令的执行结果要送往几个目标地址，则要分别形成几个数据令牌。



（a） 指令组成



（b） 操作包组成

（c） 数据令牌组成

数据流计算机指令的组成

13.2 数据流程序图和数据流语言

最基本的数据流语言是数据流计算机的机器语言，即数据流程序图。

13.2.1 数据流程序图

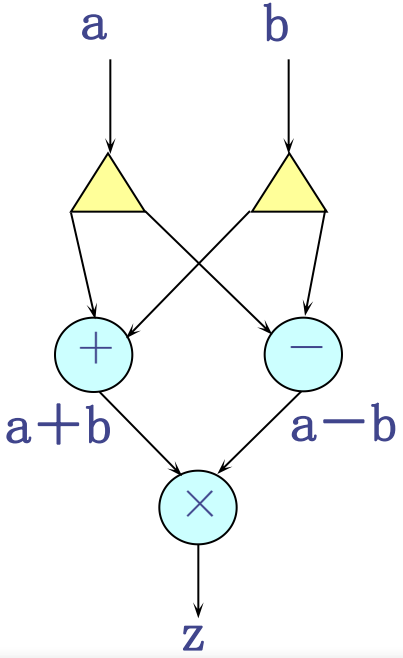
1. 数据流程序图：一种特殊的有向图，由多个结点以及连接这些结点的有向弧构成。
   - 结点用圆圈、三角形、菱形、椭圆等形状表示。
   - 符号表示进行什么操作。

- 弧代表结点之间的关系及令牌流向。
- 也称为结点分支线表示法。

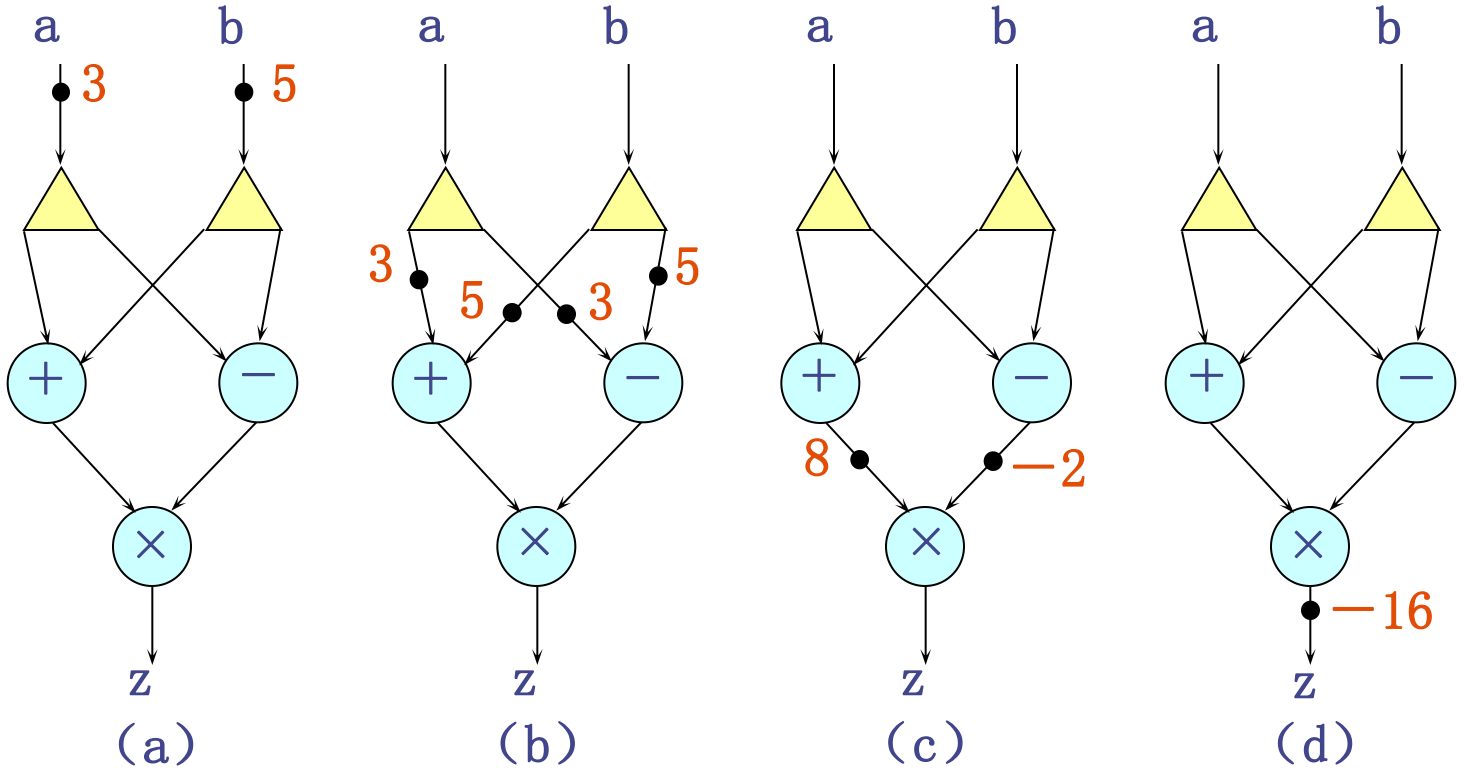
1. 举例

- 计算函数z ＝ (a＋b)×(a－b)的数据流程序图

其中：三角形表示复制结点
圆圈表示运算结点



- 数据流程序图的执行过程示意图
  - 用实心的圆点代表数据令牌沿弧移动
  - 旁边的数字代表数据的值。
  - 对于任一个结点来说，其输入弧上有实心圆点代表相应的输入数据已准备就绪。
  - 当其所有输入弧上的数据都已经就绪，且输出弧上没有数据令牌时，便可“点火”执行。



a=3，b=5

1. 用于数据流程序图的各种符号（即结点）

- 实心箭头：表示数据值
- 空心箭头：表示逻辑值

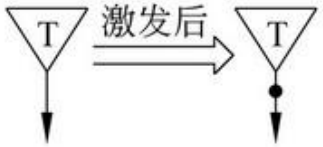
- 复制操作结点（copy）：数据或控制量（布尔量）的复制。
- 运算操作结点（operator）：主要包括常用的加、减、乘、除、乘方、开方等算术运算及与、或、非、异或、或非等布尔逻辑运算。
- 常数发生器结点（identity）：没有输入端，只有一条输出分支。
  - 用于产生常数，激发后输出带常数的令牌。

- 判定操作结点（decider）：它对输入数据按某种关系进行判断和比较，然后在输出端给出带逻辑值真（T）或假（F）的控制令牌。
- 控制类操作结点：控制类操作结点的激发条件需要加入布尔控制端。

常用的控制类操作结点有4种：

- T门控结点
- F门控结点
- 开关门控结点
- 归并门控结点

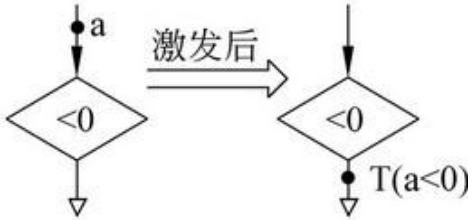
常用非控制类操作结点及其激发规则

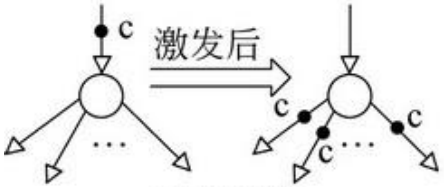


(a) 常数产生结点

(b) 复制操作结点

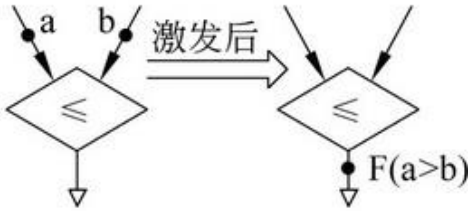
数据连接



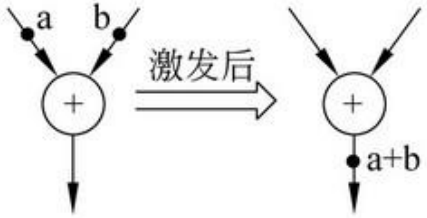


控制连接

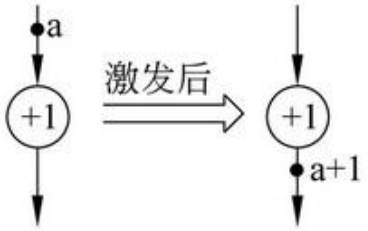
(c) 连接操作结点



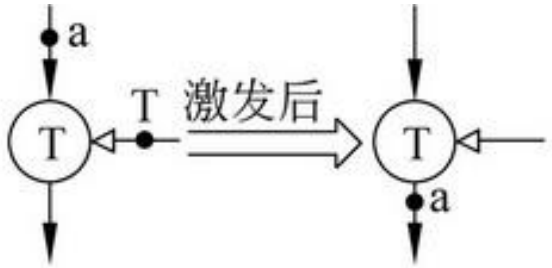
(d) 判定操作结点



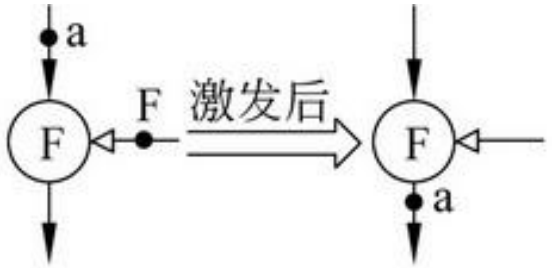
(e) 加法运算结点



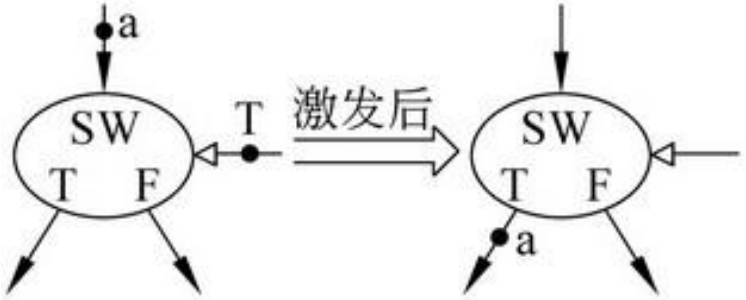
(f) 加 1 结点



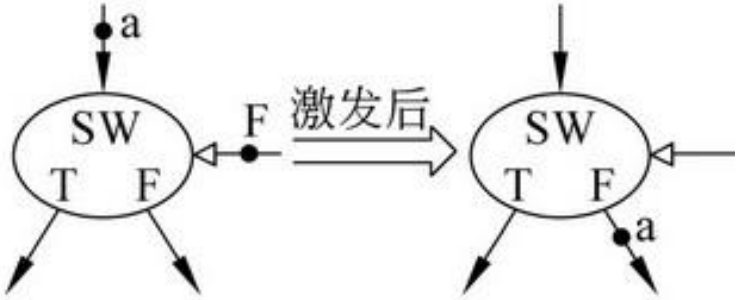
(a) T门控结点



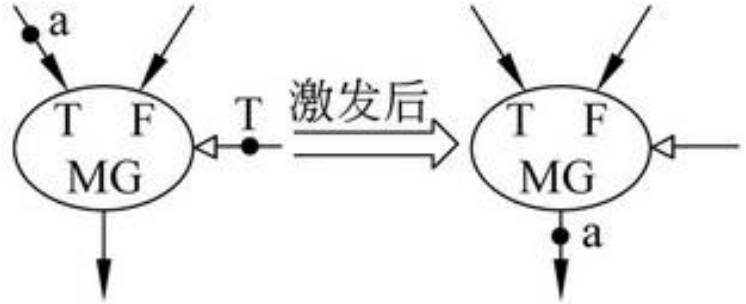
(b) F门控结点



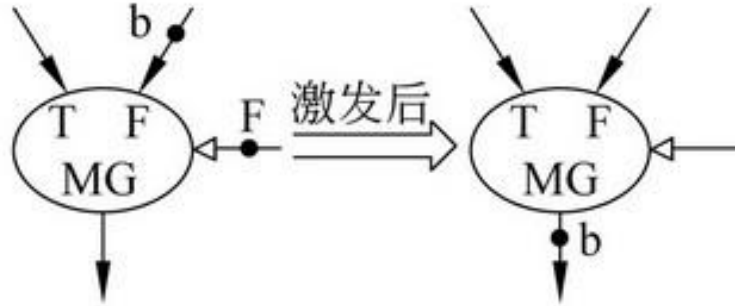
(c) 开关门控结点 1



(d) 开关门控结点 2



(e) 归并门控结点 1



(f) 归并门控结点 2

常用控制类操作结点及其激发规则

例13.1 利用上述单功能操作结点实现一般高级语言中的条件语句：

if true then G1 else G2

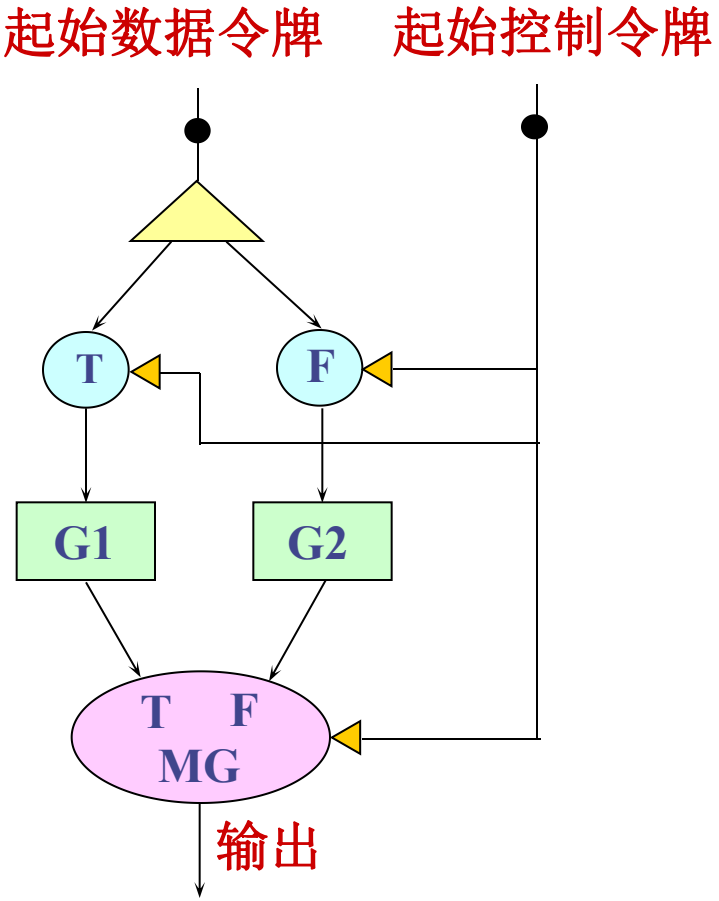
试画出数据流程序图，其中的G1和G2都是各自独立的数据流程序图。

解 如图所示。

利用一个复制结点，一个T门控结点和一个F门控结点实现起始数据令牌的两路传送；

它根据起始控制令牌所携带的是真值还是假值把起始数据令牌分别送往G1数据流程序图或G2数据流程序图；

并利用一个归并门控结点选择G1或G2数据流程序图中的一个结果作为输出，选择的依据仍然是起始控制令牌携带的是真值还是假值。



条件结构的数据流程序图

例13.2 利用上述单功能操作结点实现一般高级语言中的循环语句：

while P do G 或 until P do G

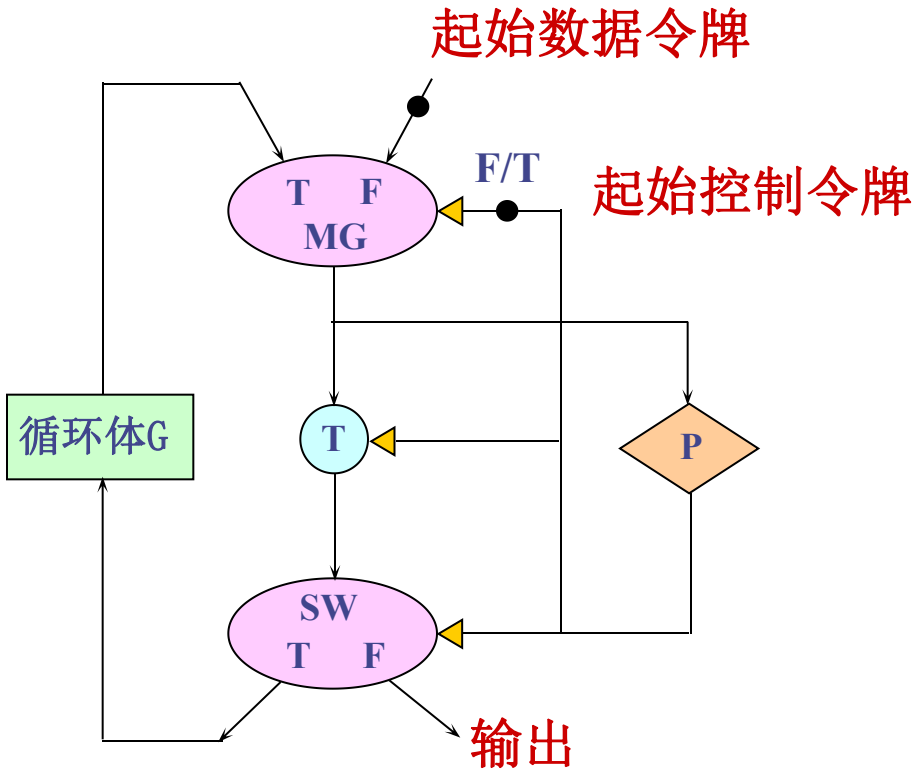
试画出数据流程序图，其中，P是循环条件，G是循环体。

解 如图所示。

在一开始要输入一个起始数据令牌和一个起始控制令牌，并用一个归并门控结点取得循环体G的输入数据令牌。

另外，用一个判定操作结点根据循环结束条件P产生的控制令牌来控制循环的执行。

最后用一个开关门控结点分配每次循环产生的结果数据令牌。



循环结构数据流程序图

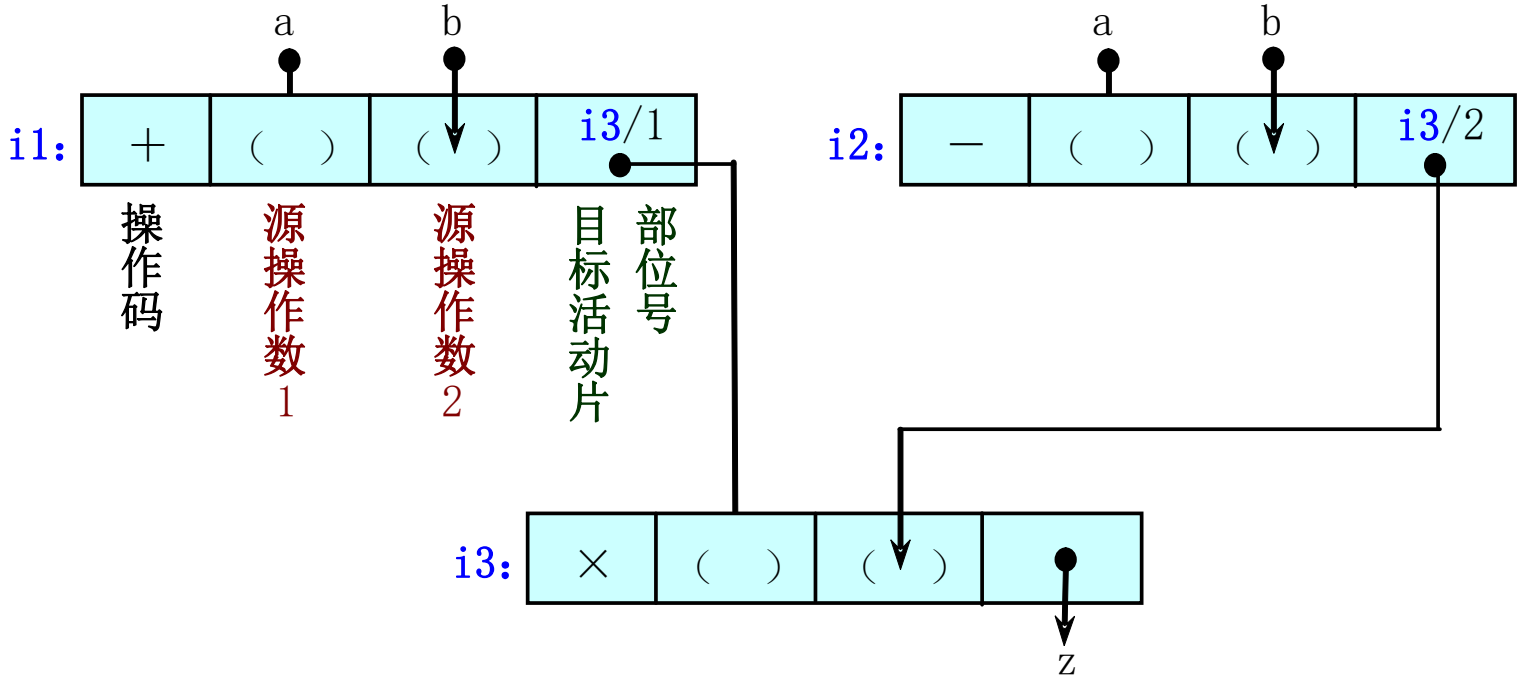
1. 活动模片表示法

- 表示数据流程序图的另一种方法
- 组成数据流程序图的基本单位是活动模片。
  - 每个活动模片相当于结点分支线表示法中的一个或几个操作结点。
- 一个活动模片通常由4个字段组成。

活动片标记	操作码	操作数 1	操作数 2	目标活动片/部位号

例13.3 请给出函数z ＝ (a＋b)×(a－b)的活动模片表示法。

解 该函数的活动模片表示法如下图所示。



13.2.2 数据流语言及其性质

1. 数据流程序图实际上是数据流计算机的机器语言。

   优点是直观易懂，但编程效率很低，难以被一般用户所接受。

2. 已经出现的比较适合于表述数据流程序的高级语言有两种：

   - 单赋值语言
     - 包括美国加州大学Irvine分校研制的ID语言，美国MIT实验室的VAL，法国的LAU语言，英国曼切斯特大学的SISAL语言等。
   - 函数类语言
     - 比较著名的有美国犹他大学研制的FP语言。

1. VAL语言的优点

- 并行性好
  - 易于开发程序中隐含的和显式的并行性，
  - 提供了相应的语句结构来表达算法中的并行成分，从而能够高效地编写数据流程序。
- 遵循单赋值规则
  - 没有传统计算机上所用的变量的概念，仅有数值的名称，运算不产生副作用。
  - 单赋值使程序清晰，易于理解，为程序的并行执行提供了一种新方法。

- 有丰富的数据类型
  - 基本数据类型有：整型、实型、布尔型和字符型等，结构类型有数组和记录等。
  - 而且允许数组和记录之间互相嵌套调用，嵌套的深度不限。
- 是一种强类型语言
  - 任何函数的自变量和计算结果的数据类型都要在函数的首部加以定义。
  - 编译程序在编译过程中能够很方便地检测出函数和表达式中数据类型发生的错误。

- 在源程序中不规定语句的执行顺序，没有GOTO之类的程序控制语句。语句的执行顺序不影响最终运算结果。
- 编制的程序具有模块化结构。
  - VAL语言编写的程序是一组模块的集合。
  - 每一个模块包括一个外部函数，该函数又可以被其他模块调用。
  - 在一个模块内部往往包含有许多内部函数，这些内部函数仅仅供本模块内部调用。

1. VAL语言存在以下缺点：
   - 没有输入输出手段，特别是交互式输入输出手段。
   - 程序的表达式还不够自然和方便。
   - 实现的效率还很低。

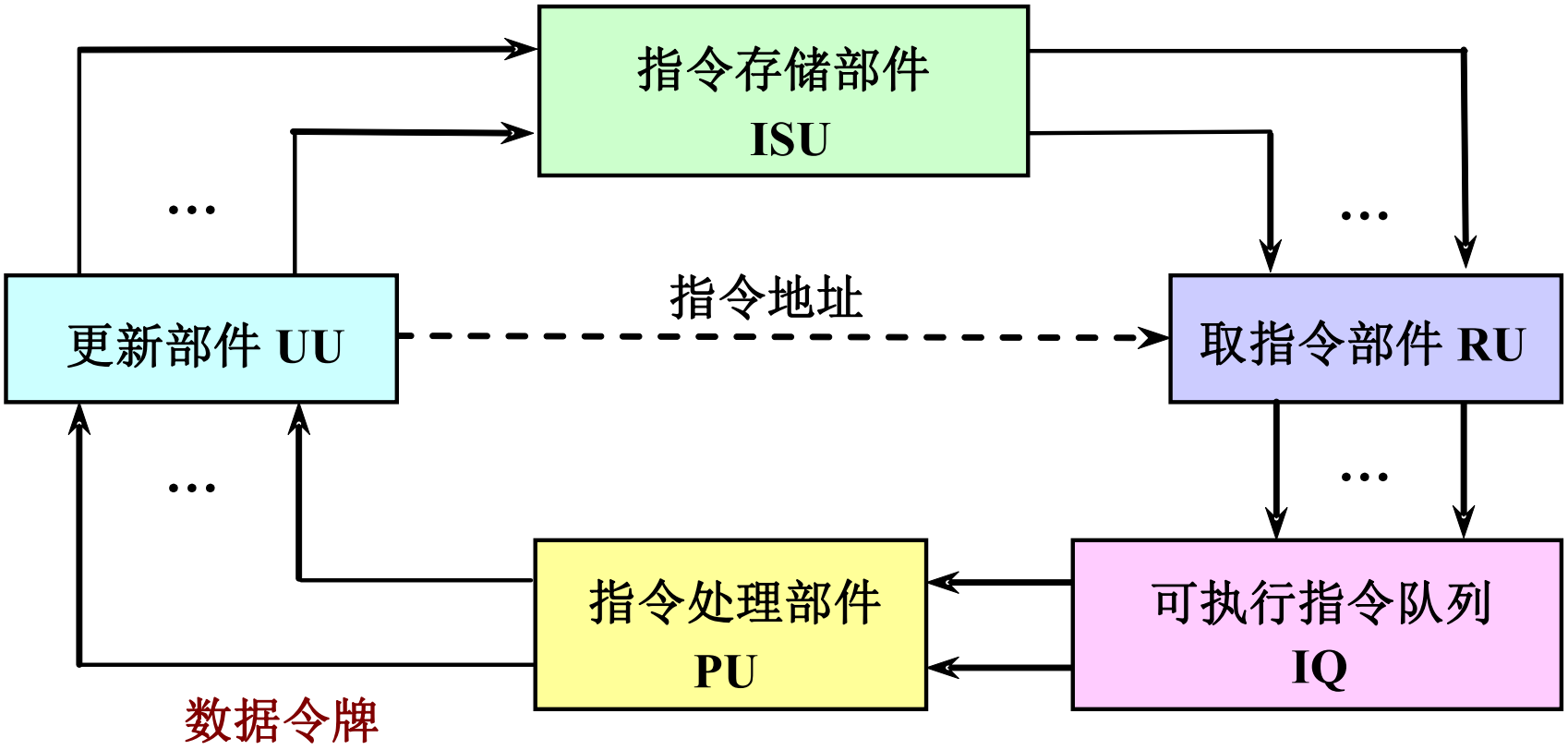
13.3 数据流计算机结构

按照对数据令牌处理方法的不同，可把数据流计算机分为两种：

- 静态数据流计算机
- 动态数据流计算机

13.3.1 静态数据流计算机

1. 静态数据流计算机的结构

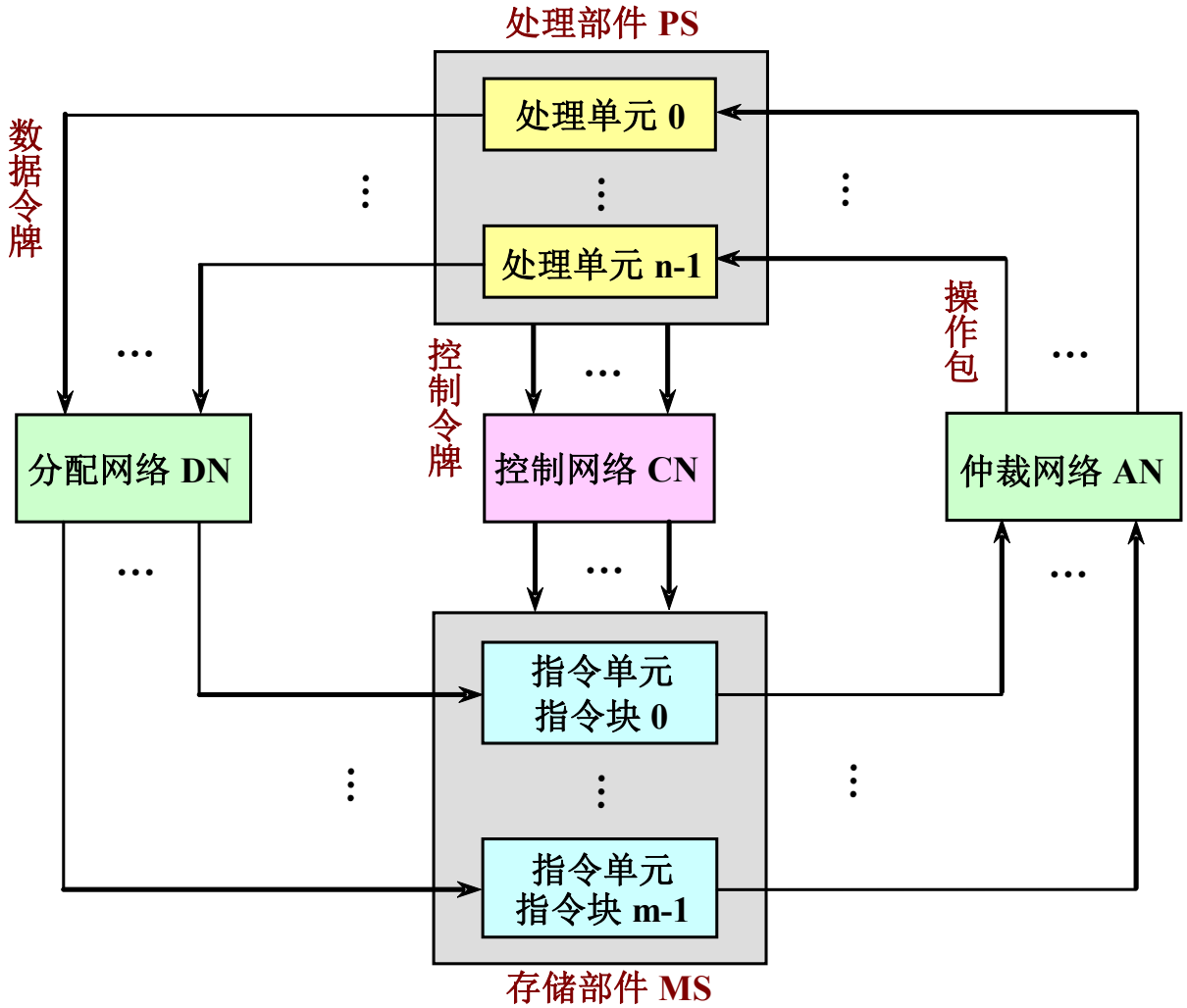

指令存储部件
ISU
…
…
更新部件 UU
指令地址
取指令部件 RU
…
…
指令处理部件
PU
可执行指令队列
IQ
数据令牌

1. 在静态数据流计算机中，数据令牌是沿数据流程序图中的有向分支流向操作结点的。
   - 当一个结点的所有输入分支线上的数据令牌都到达，且输出分支线上没有数据令牌时，就可以执行该结点的操作。这称为点火。
   - 规定：在任何一个时钟节拍内，在任何一条分支线上只允许传送一个数据令牌。
     - 好处：不必给数据令牌附加标号，使得静态数据流计算机的结构比较简单。
     - 但对程序并行性的支持不够。

1. Jack Dennis在上述模型的基础上，研制了MIT静态数据流计算机。

系统主要由5个部分组成：

- 存储部件
  - 由指令单元组成，每个指令单元保存数据流程序中的一条指令，与数据流程序图中的结点对应。通过地址来访问指令。
- 处理部件
  - 由多个处理单元组成，可以并行执行已被激活的指令所要求的操作。

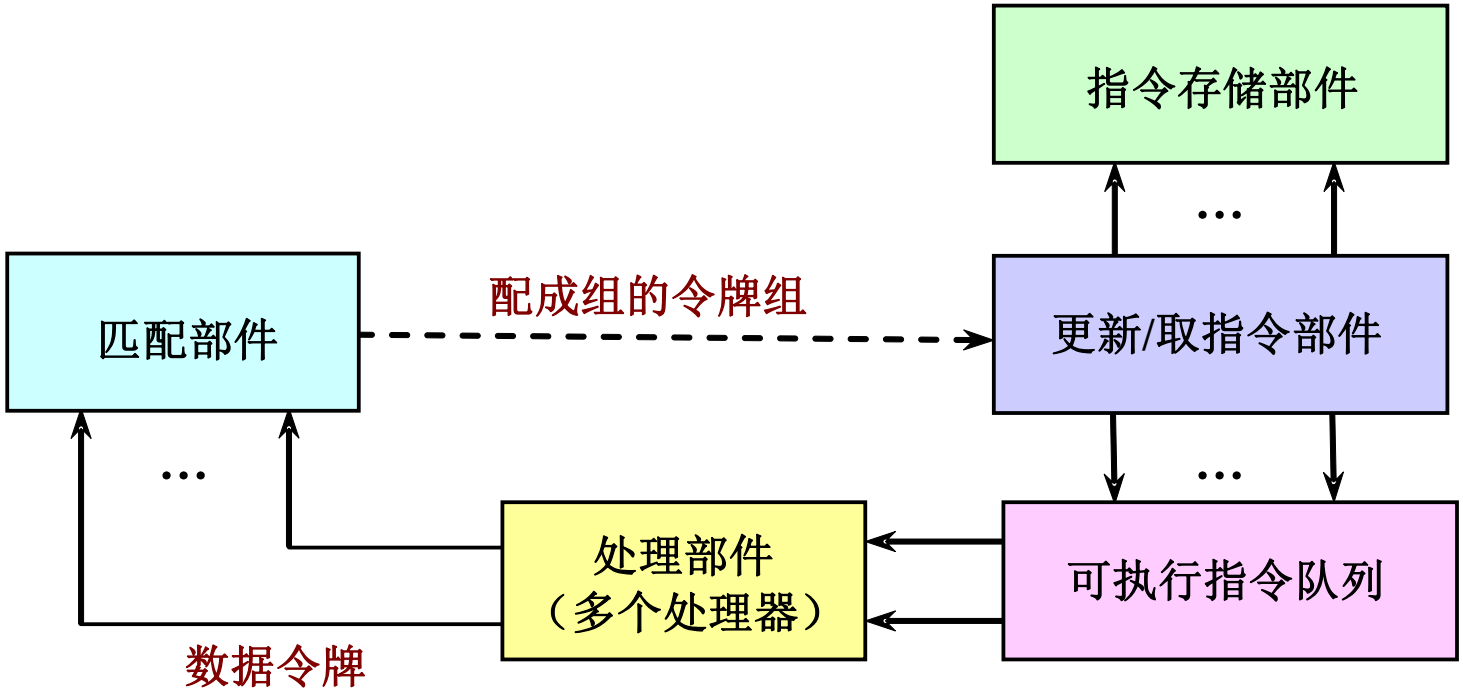


MIT静态数据流计算机的结构图

- 仲裁网络
  - 将可执行的操作包由存储器传送到处理部件。允许多个操作包同时传输。
- 控制网络
  - 将控制令牌由处理部件发送到存储部件相应的指令单元中。
- 分配网络
  - 将处理部件产生的多个结果数据令牌按其目标地址分别传送至存储部件相应的指令单元中。

13.3.2 动态数据流计算机

1. 在动态数据流计算机中，数据令牌可以带有标记，称为带标记的数据令牌。
   - 唯一地确定了令牌的状态及其他相关信息。
   - 当数据令牌在数据流程序图的有向分支线上流动时，同一条分支线上可以同时有几个数据令牌在移动。
2. 典型的动态数据流计算机的基本结构



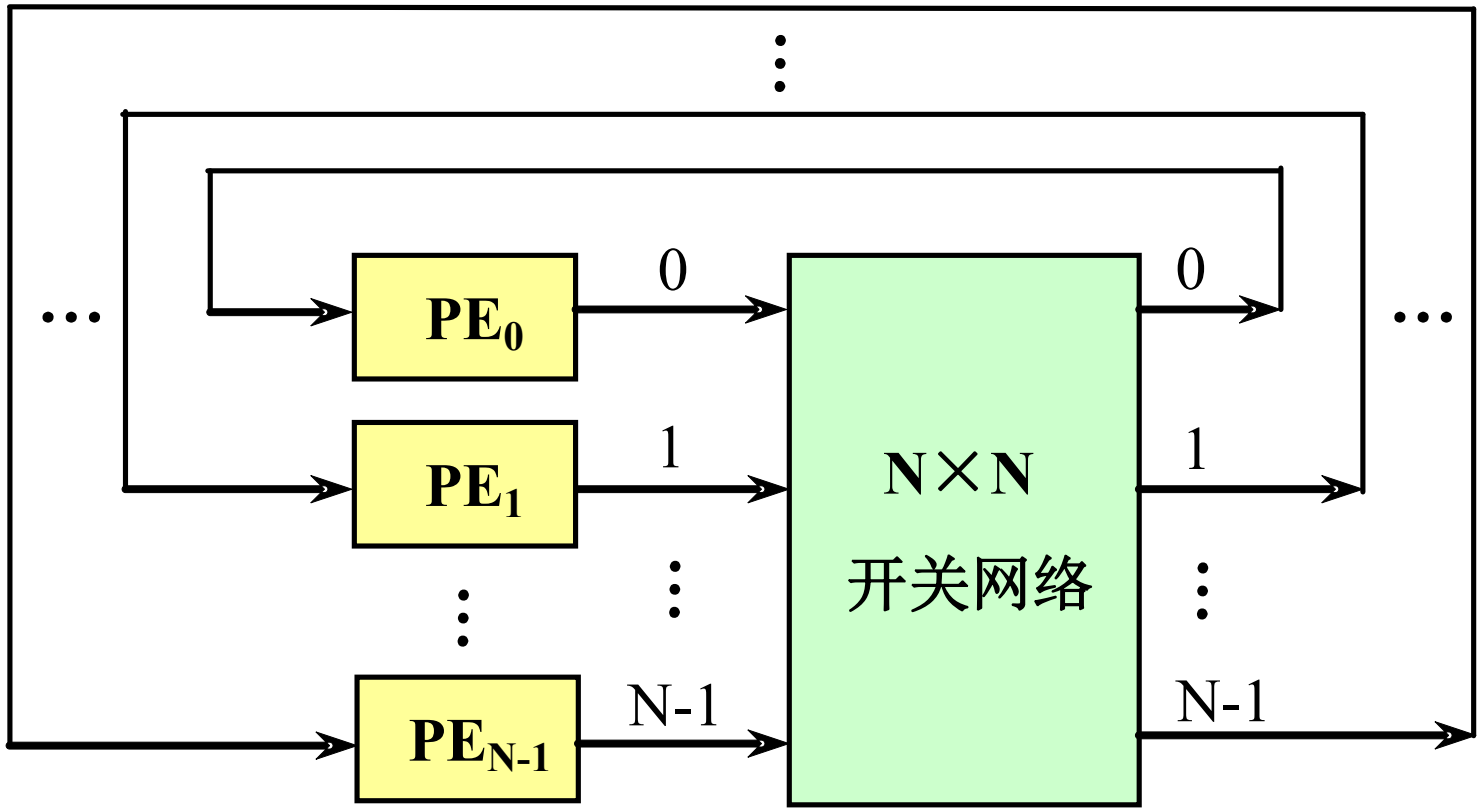
动态数据流计算机的结构

1. 网络型结构动态数据流计算机

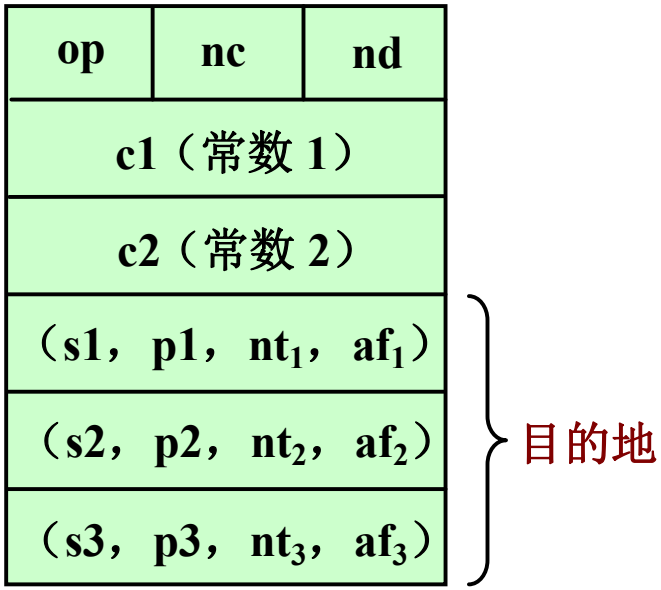
典型代表：MIT动态数据流计算机

- 由N个处理单元PE和一个N×N的包交换开关网络组成。
- PE之间通过这个开关网络进行信息交换。
- 每个PE基本上就是一台完整的处理机，有自己的存储器、算术逻辑运算部件、标记匹配部件等。

➢ 结构图


PE0
PE1
PEN-1
0
1
N-1
N×N
开关网络
0
1
N-1

- MIT动态数据流计算机的指令格式

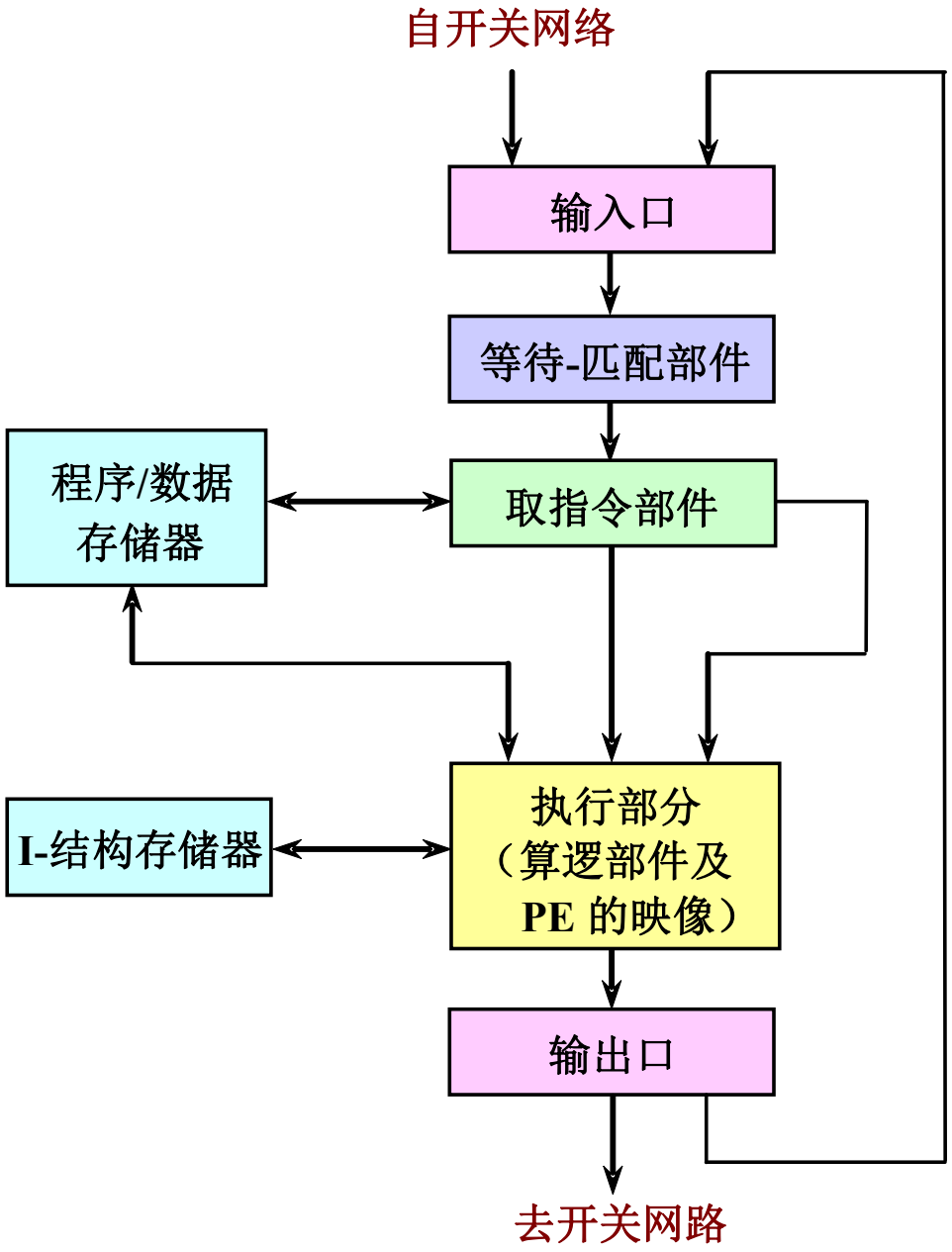


- op：操作码
- nc：存放在指令中的常数的个数（不超过两个）
- nd：操作结果送往目的地的个数
- 每个目的地由s、p、nt、af 4个字段组成

其中：

- s：目的地地址
- p：用于目的地指令的第几个输入端
- nt：激活目的地指令所需要的令牌个数
- af：用于选择执行目的地指令的PE是哪一个的赋值函数
- c1、c2：依附于该指令的常数

➢ PE的内部结构

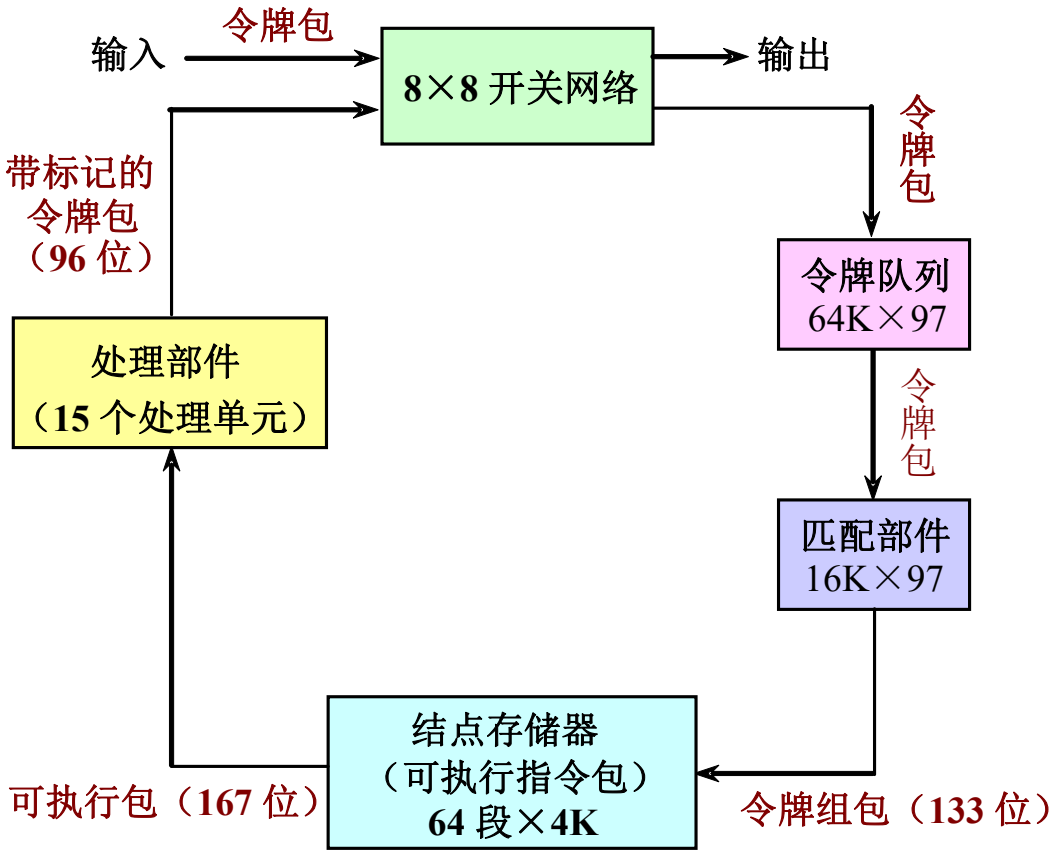

自开关网络
输入口
等待-匹配部件
程序/数据
存储器
取指令部件
I-结构存储器
执行部分
（算逻部件及
PE 的映像）
输出口
去开关网路

- N×N的开关网络
  - 用$1og_2N$个2×2的开关单元组成的网络，它有$1og_2N$级。
  - 在每个开关入口有一个异步控制器，用来控制不等长的包交换。
  - 开关出口处有仲裁电路，用来解决可能出现的路径冲突 。

1. 环形结构动态数据流计算机

典型代表：Manchester动态数据流计算机

- 结构图



Manchester动态数据流计算机的结构

- 由5个功能部件组成，按顺时针方向进行连接，形成一个环形流水线。

  （交换开关网络，令牌队列，匹配部件，结点存储器，处理部件）

- 允许多个令牌以先进先出的队列形式同时存在于数据流程序图的一个弧上。
- 采用令牌包通信。

➢ 令牌：主要由数值、标记以及目标结点地址等几部分组成。

➢ 指令：由操作码、操作数、标记、数据令牌的目的指令（两个）等组成。

系统/计数标志	1 位
特征值（tag）	36 位
目的地址	32 位
数 值	37 位

（a）数据令牌格式

系统/计数标志	1 位
特征值（tag）	36 位
操作码	12 位
操作数 1	37 位
操作数 2	37 位
目的地址 1	22 位
目的地址 2	22 位

（b）指令格式

Manchester动态数据流计算机的指令格式及数据令牌格式

- 标记中包含3部分信息
  - 令牌所属进程的标识符
  - 令牌所在的数据流程序图中的弧
  - 代表弧上第几个令牌的迭代序号

➢ Manchester动态数据流计算机采用高级数据流语言Lapes编程，这是一种单赋值的程序设计语言，其语法规则类似于PASCAL语言。

13.4 数据流计算机的评价

13.4.1 数据流计算机的优点

1. 高度并行运算

- 不仅能够开发程序中有规则的并行性，而且可以开发程序运行中隐含的并行性。
- 只要硬件资源充分，就能获得最大的并行性。
- 以前的研究和实验表明，数据流计算机对许多问题的加速比随着处理机数目的增加而线性地增长。

1. 流水线异步操作
   - 数据流计算机实现的是无副作用的纯函数型程序设计方法。
   - 可以在过程级和指令级充分开发程序中的异步并行性，可以把串行计算问题用简单的方法展开成并行计算问题来处理。
2. 与VLSI技术相适应
3. 有利于提高程序设计效率
   - 使用基于纯函数操作的程序设计语言

13.4.2 数据流计算机的缺点

1. 系统开销大

- 指令格式与传统计算机不同
  - 在空间和时间上的开销要比传统的计算机多3～6倍。
- 基于函数操作的程序设计语言使数据流计算机中存在由大量中间结果形成的数据令牌。
- 过长的流水线需要大量的并行操作才能填满。
- 操作开销大的原因
  - 主要：把并行性完全放在了指令级上
  - 另一个原因是：完全采用异步操作，没有集中控制。

1. 不能有效地利用传统计算机的研究成果
2. 数据流语言尚不完善

13.4.3 数据流计算机设计中需解决的问题

1. 研制易于使用、易于用硬件实现的高级数据流语言；
2. 研究程序如何分解，研究如何把程序模块分配给各处理部件的算法；
3. 设计出性能价格比高的信息包交换网络，实现资源冲突的仲裁和数据令牌的分配等大量通信工作；
4. 研究智能化的数据驱动机制；

1. 研究如何在数据流环境中高效率地处理复杂的数据结构；
2. 研究支持数据流运算的存储系统和存储分配方案；
3. 在广泛的应用领域里，对数据流计算机的硬件和软件作出性能评价，估计各种系统开销；
4. 研究数据流计算机的操作系统；
5. 开发数据流语言的跟踪调试工具。