

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.

G06F 9/46 (2006.01)

G06F 15/76 (2006.01)



[12] 发明专利说明书

专利号 ZL 200410049759.5

[45] 授权公告日 2009 年 5 月 20 日

[11] 授权公告号 CN 100489783C

[22] 申请日 2004.6.28

[21] 申请号 200410049759.5

[73] 专利权人 李晓波

地址 430103 湖南省长沙市韶山路 154 号
32 栋 102 号

[72] 发明人 李晓波

[56] 参考文献

CN1012409A 1987.7.29

US5237667A 1993.8.17

CN1387641A 2002.12.25

审查员 胡徐兵

[74] 专利代理机构 中科专利商标代理有限责任公
司

代理人 周国城

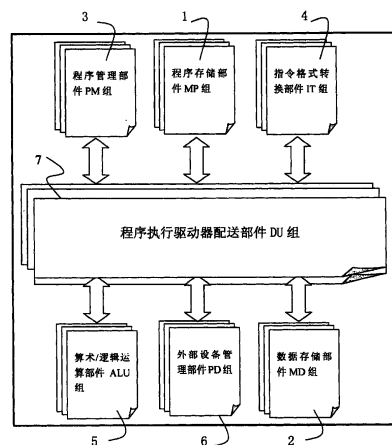
权利要求书 12 页 说明书 19 页 附图 4 页

[54] 发明名称

在单计算机上可在同一时刻执行多道程序的方法及系统

[57] 摘要

本发明公开一种在同一时刻在单台计算机处理器上能够并行地执行多道程序的方法及系统，其特征之一在于：该类计算机能够在同一时刻在单台计算机处理器上同时并行地执行多道程序；其特征之二在于：为每一道程序引入了一个程序执行驱动器，多个程序执行驱动器可相互独立地管理与控制多道程序在上述计算机处理器中同时并行地执行。本发明可用于指导设计各类用途的计算机处理器。在该类新型的计算机处理器中，程序执行驱动器的个数仅与系统提供资源的总量成正比。在系统资源足够多的条件下，程序执行驱动器的数量可足够的大，即系统可支持足够多道的程序在同一时刻在单台计算机处理器上同时并行地执行。



1、一种在单台计算机上并行执行多道程序的方法，其特征在于：
使用任一计算机处理器；

在该计算机处理器上，在同一时刻并行地执行多道程序，

所述每一道程序引入一个程序执行驱动器，使多个程序执行驱动器在同一台计算机处理器上相互独立地管理与控制多道程序在同一时刻并行地执行；

所述每道程序生成可执行代码，同时生成该道程序的程序执行驱动器，一个程序执行驱动器包含有用于说明与定义该道程序属性特征、以及管理与控制该道程序执行所需要的全部相关信息；组成一道程序的多道程序或子程序有各自的程序执行驱动器；

所述程序执行驱动器被设计成为由一组信息单元组成的信息块；在系统硬件的支持下，程序执行驱动器按照当前所携带的指令的操作码的要求逐次前往相应的各功能部件对指令进行相应的加工、处理。

2、根据权利要求1所述的方法，其特征在于：所述每道程序还执行如下步骤：

1) 一道程序生成可执行代码及其程序执行驱动器，系统管理模块将该道程序的程序执行驱动器写入到它的程序调度表的相关栏目中；该道程序存放在某个存储装置中处于等待执行状态；

2) 该道程序满足被调入执行的各项条件，系统管理模块将激活或活化该道程序的程序执行驱动器，使得该道程序进入执行状态，该道程序的程序执行驱动器管理、控制该道程序独立地执行；

3) 程序执行驱动器依据保存的该道程序的驻存信息控制到相应的程序存储部件取当前要执行的指令；

4) 该程序执行驱动器控制到某操作控制部件对已取出的指令进行译码；控制到相应的数据存储部件取操作数、控制到相应的操作控制部件完成指定的处理，以及控制将处理的结果写回到相应的数据存储部件中；

5) 该程序执行驱动器将它所保存的属于程序驻存信息类中的程序存储部件内地址 PC 项按当前指令进行修正，转步骤 3)；

6) 系统管理模块按指令和条件卸载某道程序的程序执行驱动器。

3、根据权利要求1所述的方法，其特征在于：所述程序执行驱动器的具体工作步骤如下：

步骤 S1，系统初启时，至少有一道程序的程序执行驱动器存放在某个程序管理部件 PM 的输出队列中；

步骤 S2，当某个程序执行驱动器配送部件 DU 检测到某个程序管理部件 PM 的输出队列为非空、且当前处于队首的程序执行驱动器的 NEXT 单元所指向的目的程序存储部件 MP 的输入队列为非满时，将该程序管理部件 PM 的输出队列队首的程序执行驱动器配送到相应的程序存储部件 MP 的输入队列中等待处理；

步骤 S3，当某个程序存储部件 MP 检测到其输入队列为非空时，按输入队列队首处的程序执行驱动中 PC 单元所指定的 PC 地址到它所管理的程序存储部件中读取指令，取出的指令存放到该程序执行驱动的某个单元中；如果该程序存储部件 MP 的输出队列为满状态，则等待其空；当有空位时，将处理结束的程序执行驱动器写入到它的输出队列中等待程序执行驱动器的配送部件 DU 的配送处理；

步骤 S4，当某个程序执行驱动器的配送部件 DU 检测到某个程序存储部件 MP 的输出队列为非空、且当前处于队首的程序执行驱动的 NEXT 单元所指向的目的部件是指令格式转换部件 IT、且至少有一个指令格式转换部件 IT 的输入队列为非满时，将该程序存储部件 MP 的输出队列队首的程序执行驱动器配送到相应的指令格式转换部件 IT 的输入队列中等待处理；

步骤 S5，当某个指令格式转换部件 IT 检测到其输入队列为非空时，对输入队列队首处的程序执行驱动器中的指令进行指令的格式变换，0 变换后的结果存放到该程序执行驱动器的相应单元中；按上述指令的约定置该程序驱动器的 NEXT 单元；如果该指令格式转换部件 IT 的输出队列为满状态，则等待其空；当有空位时，将处理结束的程序执行驱动器写入到它的输出队列中等待程序执行驱动器配送部件 DU 的配送处理；

步骤 S6，当某个程序执行驱动器的配送部件 DU 检测到指令格式转换部件 IT、算术/逻辑运算部件 ALU 和数据存储部件 MD 中的任何一个部

件的输出队列为非空、且当前处于队首的程序执行驱动器的 NEXT 单元所指向的目的部件的输入队列为非满时，将该程序执行驱动器配送到相应的目的部件的输入队列中等待处理，对该目的部件的处理包括：

若是某个算术/逻辑运算部件 ALU 则转步骤 S7；

若是某个数据存储部件 MD 则转步骤 S8；

若是某个程序管理部件 PM 则转步骤 S9；

步骤 S7，当某个算术/逻辑运算部件 ALU 检测到其输入队列为非空时，按输入队列队首处的程序执行驱动器中的 NEXT 单元中的命令对该程序执行驱动中的相应单元所保存的操作数进行指定的算术/逻辑运算，运算后的结果存放到该程序执行驱动器的相应单元中；按上述程序驱动器中命令的约定置该程序驱动器的 NEXT 单元；如果该 ALU 部件的输出队列为满状态，则等待其空；当有空位时，将处理结束的程序执行驱动器写入到它的输出队列中等待程序执行驱动驱动器配送部件 DU 的配送处理，下转步骤 S6；

步骤 S8，当某个数据存储部件 MD 检测到其输入队列为非空时，按输入队列队首处的程序执行驱动中的 NEXT 单元中的读或写命令以及该程序执行驱动中的某单元所指定的部件内地址对该程序执行驱动中的某单元的数据在它所管理的数据存储部件 MD 中进行读/写操作；如果是读操作，将读出的数据存放到该程序执行驱动的某单元中；按上述程序驱动器中命令的约定置该程序驱动器的 NEXT 单元；如果该数据存储部件 MD 的输出队列为满状态，则等待其空；当有空位时，就将处理结束的程序执行驱动器写入到它的输出队列中等待程序执行驱动驱动器的配送部件 DU 的配送处理，下转步骤 S6；

步骤 S9，当程序管理部件 PM 检测到其输入队列为非空时，按其输入队列队首处的程序执行驱动器中的命令要求进行相应的操作：

1) 非 END 指令，按要求处理该程序执行驱动器，将处理结束的程序执行驱动写入输出队列，等待程序执行驱动驱动器配送部件 DU 的配送处理，转步骤 S2；

2) END 指令，置相应结束标志，从输入队列中删除该程序执行驱动器，激活系统管理模块调度下一个等待执行的程序进入执行；

3) 当前指令执行结束后满足被中断的条件置相应中断标志, 保存该程序执行驱动器相应单元的内容, 并启动系统管理模块的中断处理。

4、一种在单台计算机上并行执行多道程序的系统, 其特征在于, 包括: 计算机处理器, 用于在同一时刻并行地执行多道程序;

所述计算机处理器包括: 系统的硬件和系统管理模块, 其中:

系统的硬件包括:

一个或多个可并行工作的存储装置, 用于存储读/写程序和数据信息;

一个或多个可并行工作的操作控制装置, 用于程序运行管理、指令转换、运算、控制服务;

一个或多个可并行工作的在部件间起连接作用的程序执行驱动器配送装置 DU;

一个系统管理模块, 用于程序调度, 并维护着一个称为程序调度表的数据结构;

系统在初启时将系统管理模块的程序执行驱动器加载到系统的某操作控制部件中, 该程序执行驱动器将驱动系统管理模块逐个激活多个用户程序的程序执行驱动器, 各用户程序的程序执行驱动器将相互独立地在同一时刻驱动各自的用户程序在同一台计算机处理器中同时并行地执行。

5、根据权利要求 4 所述的系统, 其特征在于: 所述程序执行驱动器包括:

第 0 号单元至第 6 号单元为运行时工作单元: 用于本道程序在执行过程中存放临时/中间结果; 其中:

第 0 号单元为 NEXT 单元, 是所述程序执行驱动器当前将要前往请求服务的功能部件的编号;

第 1 至第 6 号单元, 其用途由当前所驱动的指令确定, 其中: 当第 1 号单元用于作 ΔPC 单元时: $\Delta PC=0$ 时表示当前指令是一条 END 指令, $\Delta PC=1$ 时表示当前指令是一条非转移型指令, $\Delta PC>1$ 或 $\Delta PC<0$ 时表示当前指令是一条转移型指令;

第 7 号单元, 程序存储部件的部件编号 MPno: 用于存放本道程序驻

存的程序存储部件的部件编号；

第 8 号单元，程序存储部件的部件个数 MPnu：用于存放本道程序所占用的连续的程序存储部件的个数；如果 MPnu 的值大于 1，则 MPno 的值是第一个程序存储部件的部件编号；

第 9 号单元，程序存储部件的部件内地址 PC：用于存放本道程序当前执行指令的程序存储部件的部件内地址；

第 10 号单元，数据存储部件的部件编号 MDno：用于存放本道程序所占用的数据存储部件的部件编号；

第 11 号单元，数据存储部件的部件个数 MDnu：用于存放本道程序所占用的连续的数据存储部件的个数；如果 MDnu 的值大于 1，则 MDno 的值是第一个数据存储部件的部件编号；

第 12 号单元，执行指令条数 Ins：用于记录本道程序本次调入后累计执行的指令条数；

第 13 号单元，程序道号 PrNo：当两道以上正在执行的程序共同存放在同一个程序存储部件中时，用本参数标识各道程序；

第 14 号 至 第 29 号单元，16 个通用寄存器 R0 - R15。

6、根据权利要求 4 所述的系统，其特征在于：所述一个或多个可并行工作的存储装置，包括下述部件：

●设置一个或多个独立并行工作的程序存储部件 MP：

用于当前队首的程序执行驱动器请求读操作服务；

用于当前队首的程序执行驱动器请求写操作服务；

●设置一个或多个独立并行工作的数据存储部件 MD：

用于当前队首的程序执行驱动器请求读操作服务；

用于当前队首的程序执行驱动器请求写操作服务。

7、根据权利要求 4 所述的系统，其特征在于：所述一个或多个可并行工作的操作控制装置，包括下述部件：

●设置一个或多个独立并行工作的程序管理部件 PM：

用于启动一个程序执行驱动器的运行；

用于中断一个程序执行驱动器的运行；

用于终止一个程序执行驱动器的运行；

●设置一个或多个独立并行工作指令格式转换部件 IT:

用于请求指令格式转换的程序执行驱动器中存放的当前指令进行格式转换, 将其写入到输出队列中;

●设置一个或多个独立并行工作的算术/逻辑运算部件 ALU, 用于提供有关算术/逻辑运算的服务。

8、根据权利要求 4 所述的系统, 其特征在于:

一个或多个在所述可并行工作的部件间起连接作用的程序执行驱动器配送部件 DU, 用于提供将程序执行驱动器从所述一个部件的输出队列配送到所述另一个部件的输入队列的运输与配送服务。

9、根据权利要求 4 或 6 所述的系统, 其特征在于: 所述存储装置、操作控制装置分别含有一个输入缓存装置与一个输出缓存装置, 按请求/服务的方式工作, 输入缓存装置非空时, 启动完成指定的服务, 结果存入输出缓存装置。

10、根据权利要求 6 所述的系统, 其特征在于: 所述的存储装置还包括:

●程序存储部件 MP:

用于维护着一个输入队列, 所有请求本程序存储部件 MP 处理的程序执行驱动器都等待在该队列中, 用于维护着一个输出队列, 所有已处理的程序执行驱动器都等待在该队列中等待程序执行驱动器配送部件 DU 的配送服务;

用于读请求: 如果当前队首的程序执行驱动器请求的是读操作, 则从该程序执行驱动器中取出该道程序的部件内地址 PC 送程序存储部件 MP 的地址寄存器, 并启动程序存储部件 MP 的读操作读取该道程序的当前指令, 读出的指令写入到当前的程序执行驱动器的相应单元中, 置程序执行驱动器的 NEXT 单元为指令格式转换部件 IT, 然后将该程序执行驱动器写入到程序存储部件 MP 的输出队列中;

用于写请求: 如果当前队首的程序执行驱动器请求的是写操作, 需完成的操作与数据存储部件的写操作;

●数据存储部件 MD:

用于维护着一个输入队列, 所有请求本数据存储部件 MD 处理的程

序执行驱动器都等待在该队列中；数据存储部件 MD 维护着一个输出队列，所有已处理的程序执行驱动器都等待在该队列中等待程序执行驱动器配送部件 DU 的配送服务；

用于读操作：如果当前队首的程序执行驱动器请求的是读操作，则从该程序执行驱动器中取出该读操作的部件内地址送数据存储部件 MD 的地址寄存器，并启动数据存储部件 MD 的读操作读取相应存储单元的内容，读出的数据写入到当前的程序执行驱动器的某单元中，置 NEXT 单元为 PM，置 ΔPC 单元为 1，然后将该程序执行驱动器写入到 MD 的输出队列中；

用于写操作：如果当前队首的程序执行驱动器请求的是写操作，则从该程序执行驱动器中取出该写操作的数据及部件内地址分别送 MD 的数据寄存器和地址寄存器，并启动数据存储部件 MD 的写操作将数据写入到相应数据存储部件 MD 的存储单元中去，置 NEXT 单元为 PM，置 ΔPC 单元为 1，然后将该程序执行驱动器写入到数据存储部件 MD 的输出队列中。

11、根据权利要求 7 所述的系统，其特征在于：所述操作控制装置还包括：

●程序管理部件 PM：

用于维护着一个输入队列，所有请求本程序管理部件 PM 处理的程序执行驱动器都等待在该队列中，程序管理部件 PM 也维护着一个输出队列，所有已处理的程序执行驱动器都等待在该队列中等待程序执行驱动器配送部件 DU 的配送服务；程序管理部件 PM 的处理步骤还包括：

启动一个程序执行驱动器的运行：系统管理程序激活一道程序的程序执行驱动器实际上就是将该道程序的程序执行驱动器写入程序管理部件 PM 的输入队列，当程序管理部件 PM 处理到该程序执行驱动器时，将其 NEXT 单元置为程序存储部件 MP，然后将该程序执行驱动器写入到程序管理部件 PM 的输出队列中；

继续一个程序执行驱动器的运行：每个程序执行驱动器驱动一道程序的当前指令执行的最后一步是返回到程序管理部件 PM，如果当前指令不是一条结束指令，则程序管理部件 PM 将控制该程序执行驱动器继续运

行：对单元 $9 \leftarrow \text{单元 } 9 + \text{单元 } 1$ 用于作 $PC \leftarrow PC + \Delta PC$ 单元时，置 NEXT 单元为程序存储部件 MP，然后将该程序执行驱动器写入到程序管理部件 PM 的输出队列中；

中断一个程序执行驱动器的运行：当某个程序执行驱动器驱动的某道程序执行到当前指令满足被中断的条件时，程序管理部件 PM 置相应的中断标志、保存该程序执行驱动器相应单元的内容、并启动系统管理程序的中断处理例程，将该例程的程序执行驱动器置入程序管理部件 PM 的输出队列，由该程序执行驱动器驱动中断处理例程去完成有关的中断处理工作，该中断处理例程将把被中断的程序的程序执行驱动器写回到系统的程序调度表中，然后再根据调度规则选择和激活下一道程序的程序执行驱动器以执行该道程序；

终止一个程序执行驱动器的运行：当某道程序执行到一条结束指令时，程序管理部件 PM 将通过采用删除该道程序的程序执行驱动器的方式来终止该道程序的执行，该道程序在执行结束指令前，须执行一条或多条资源释放型指令；

●指令格式转换部件 IT：

用于维护着一个输入队列，所有请求本指令格式转换部件 IT 处理的程序执行驱动器都等待在该队列中，指令格式转换部件 IT 也维护着一个输出队列，所有已处理的程序执行驱动器都等待在该队列中等待程序执行驱动器配送部件 DU 的配送服务；

对于请求指令格式转换的程序执行驱动器，指令格式转换部件 IT 将对该程序执行驱动器中存放的当前指令进行格式转换，并据此设置 NEXT 单元，然后将该程序执行驱动器写入到指令格式转换部件 IT 的输出队列中；

●算术/逻辑运算部件 ALU，

用于维护着一个输入队列，所有请求本算术/逻辑运算部件 ALU 处理的程序执行驱动器都等待在该队列中，算术/逻辑运算部件 ALU 也维护着一个输出队列，所有已处理的程序执行驱动器都等待在该队列中等待程序执行驱动器配送部件 DU 的配送服务；

在请求算术/逻辑运算部件 ALU 进行算术/逻辑运算的服务之前，程

序执行驱动器已控制将该指令所需的操作数准备好且存放在程序执行驱动器的相应单元中,算术/逻辑运算部件 ALU 对存放在程序执行驱动器中的相应的操作数进行所要求的算术/逻辑运算,结果存放在程序执行驱动器的相应的单元中,置 NEXT 单元为程序管理部件 PM,按指令的规定置 ΔPC 单元,然后将该程序执行驱动器写入到算术/逻辑运算部件 ALU 的输出队列中。

12、根据权利要求 8 所述的系统,其特征在于:所述程序执行驱动器配送装置 DU 还包括:

用于按预定的顺序逐个查询各部件组中程序管理部件 PM、程序存储部件 MP 和数据存储部件 MD 的各部件的输出队列是否为非空,对于某非空的输出队列,程序执行驱动器配送部件 DU 从队中取出一程序执行驱动器,并按照此程序执行驱动器的 NEXT 单元所指示的目的部件组编号以及目的部件编号查询该目的部件组中的目的部件的输入队列是否为非满,如果是非满,则将该程序执行驱动配送到该目的部件组中的目的部件的输入队列中去;对于指令格式转换部件 IT 和算术/逻辑运算部件 ALU,在该部件中任选一输入队列为非满的部件。

13、根据权利要求 5 或 10 所述的系统,其特征在于:所述的存储装置还包括:

- 在程序存储部件 MP 的输入队列中程序执行驱动器的第 0 号单元的当前值为 MPr, MPr 表示目的部件是程序存储部件 MP 且要求进行“读”操作;

- 在程序存储部件 MP 的输出队列中程序执行驱动器的第 0 号单元的当前值为指令格式转换部件 IT 的编码;第 1 号单元的当前值为 I, I 为读出的当前指令;

- 在数据存储部件 MD 的输出队列中发往程序管理部件 PM 的程序执行驱动器的第 0 号单元的当前值为程序管理部件 PM 的编码;第 1 号单元的当前值为 1。

14、根据权利要求 5 或 11 所述的系统,其特征在于:所述的操作控制装置还包括:

- 在程序管理部件 PM 的输入队列中接收到的程序执行驱动器的第 0

号单元的当前值为程序管理部件PM的编码;第1号单元的当前值为 ΔPC , ΔPC 或是0或是1或是转移的偏移量;

●在程序管理部件PM的输出队列中发往程序存储部件MP的程序执行驱动器第0号单元的当前值为MP_r, MP_r表示目的部件是程序存储部件MP且要求进行“读”操作;

●在指令格式转换部件IT的输入队列中的程序执行驱动器的第0号单元的当前值为指令格式转换部件IT的编码;第1号单元的当前值为I, I为读出的当前指令;

●在指令格式转换部件IT的输出队列中的程序执行驱动器组成单元的当前值由当前指令I确定:

当前指令I为“LDI Rd, C”时,由指令格式转换部件IT发往在程序管理部件PM的程序执行驱动器的第0号单元的当前值为程序管理部件PM的编码;第1号单元的当前值为1;

当前指令I为“MV Rd, Rs”时,由指令格式转换部件IT发往程序管理部件PM的程序执行驱动器的第0号单元的当前值为程序管理部件PM的编码;第1号单元的当前值为1;

当前指令I为“LD Rd, Rs”时,由指令格式转换部件IT发往数据存储部件MD的程序执行驱动器的第0号单元的当前值为MD_r, MD_r表示目的部件是数据存储部件MD且要求进行“读”操作;第1号单元的当前值为程序管理部件PM的编码;第2号单元的当前值为[Rd], [Rd]表示寄存器Rd的当前值;第3号单元的当前值为[Rs], [Rs]表示寄存器Rs的当前值;

当前指令I为“STI Rd, C”时,由指令格式转换部件IT发往数据存储部件MD的程序执行驱动器第0号单元的当前值为MDI_w, MDI_w表示目的部件是数据存储部件MD且要求进行“立即数写”操作;第1号单元的当前值为程序管理部件PM的编码;第2号单元的当前值为[Rd];第3号单元的当前值为立即数C;

当前指令I为“ST Rd, Rs”时,由指令格式转换部件IT发往数据存储部件MD的程序执行驱动器的第0号单元的当前值为MD_w, MD_w表示目的部件是数据存储部件MD且要求进行“写”操作;第1号单元的当前

值为程序管理部件 PM 的编码；第 2 号单元的当前值为[Rd]；第 3 号单元的当前值为[Rs]；

当前指令 I 为“ADDI Rd, C”时，由指令格式转换部件 IT 发往算术/逻辑运算部件 ALU 的程序执行驱动器的第 0 号单元的当前值为 ALUI+，ALUI+表示目的部件是算术/逻辑运算部件 ALU 且要求进行“立即数的加”操作；对于 SUBI, MULI, DIVI 指令相应的程序执行驱动器的第 0 号单元的当前值分别是 ALUI-、ALUI*、ALUI÷；第 1 号单元的当前值为程序管理部件 PM 的编码；第 2 号单元的当前值为 Rd, Rd 表示寄存器 Rd 的编号；第 3 号单元的当前值为[Rd]；第 4 号单元的当前值为 C；

当前指令 I 为“ADD Rd, Rs1, Rs2”时，由指令格式转换部件 IT 发往算术/逻辑运算部件 ALU 的程序执行驱动器的第 0 号单元的当前值为 ALU+，ALU+表示目的部件是算术/逻辑运算部件 ALU 且要求进行“加”操作；SUB, MUL, DIV 指令相应的程序执行驱动器的第 0 号单元的当前值分别是 ALU-、ALU*、ALU÷；第 1 号单元的当前值为程序管理部件 PM 的编码；第 2 号单元的当前值为 Rd；第 3 号单元的当前值为[Rs1]；第 4 号单元的当前值为[Rs2]；

当前指令 I 为“JEQI Rs, C”时，由指令格式转换部件 IT 发往算术/逻辑运算部件 ALU 的程序执行驱动器的第 0 号单元的当前值为 ALUIjeq, ALUIj 表示目的部件是算术/逻辑运算部件 ALU 且要求进行“立即数的转移判断”操作；JNEI 指令的程序执行驱动器的第 0 号单元的当前值是 ALUIjne；第 1 号单元的当前值为程序管理部件 PM 的编码；第 2 号单元的当前值为 C；第 3 号单元的当前值为[Rs]；

当前指令 I 为“JEQ Rs1, Rs2”时，由指令格式转换部件 IT 发往算术/逻辑运算部件 ALU 的程序执行驱动器的第 0 号单元的当前值为 ALUjeq, ALUjeq 表示目的部件是算术/逻辑运算部件 ALU 且要求进行“转移判断”操作；JNE 指令的程序执行驱动器的第 0 号单元的当前值是 ALUjne；第 1 号单元的当前值为程序管理部件 PM 的编码；第 2 号单元的当前值为[Rs2]；第 3 号单元的当前值为[Rs1]；

当前指令 I 为“SLT Rd, Rs1, Rs2”时，由指令格式转换部件 IT 发往请求算术/逻辑运算部件 ALU 的程序执行驱动器的第 0 号单元的当前值为

ALUslt, ALUslt 表示目的部件是 ALU 且要求进行“小于设置”操作; SLE、SGT、SGE、SEQ、SNE 指令相应的程序执行驱动器的第 0 号单元的当前值是 ALUsle、ALUsgt、ALUsge、ALUseq、ALUsne; 第 1 号单元的当前值为程序管理部件 PM 的编码; 第 2 号单元的当前值为 Rd; 第 3 号单元的当前值为[Rs1]; 第 4 号单元的当前值为[Rs2];

在请求算术/逻辑运算部件 ALU 的输出队列中的程序执行驱动器的第 0 号单元的当前值为程序管理部件 PM 的编码; 第 1 号单元的当前值为 ΔPC , ΔPC 或是 1 或是转移的偏移量。

15、一种计算机处理器, 其特征在于, 包括: 程序存储部件 (1)、数据存储部件 (2)、程序管理部件 (3)、指令格式转换部件 (4)、算术/逻辑运算部件 (5)、外部设备管理部件 (6) 程序执行驱动器配送部件 (7); 程序存储部件 (1)、数据存储部件 (2)、程序管理部件 (3)、指令格式转换部件 (4)、算术/逻辑运算部件 (5)、外部设备管理部件 (6) 分别通过各自的一组或两组信号线将各自的输入队列的尾端与输出队列的首端连接到程序执行驱动器配送部件 (7);

程序执行驱动器配送部件 (7) 由多个数据输入端口、多个数据输出端口、连通网络装置与控制管理装置组成; 程序执行驱动器配送部件 (7) 的控制管理装置与多个数据输入端口连接, 用于数据选通的控制与管理; 程序执行驱动器配送部件 (7) 的连通网络装置用于将多个数据输入端口连到多个数据输出端口;

程序执行驱动器配送部件 (7) 的连通网络装置采用交叉互连方式: 任何一个输入端口都与所有或部分输出端口相连;

程序执行驱动器配送部件 (7) 的连通网络装置采用多总线方式: 所有的输入端口与输出端口被分成几组, 分别连到一条总线上, 总线与总线之间再交叉互连。

在单计算机上可在同一时刻执行多道程序的方法及系统

技术领域

本发明涉及计算机技术范畴中的计算机体系结构技术领域。具体说是一种在同一时刻在单台计算机处理器上能够并行地执行多道程序的方法及系统。

背景技术

最早的计算机是由著名数学家冯·诺依曼等人在1946年总结并明确提出来的，因此经常称之为冯·诺依曼结构计算机。冯·诺依曼结构计算机的发明奠定了现代计算机发展的基础，有着十分重大的意义。

按照冯·诺依曼计算机原理设计的计算机在体系结构上的特点有许多的归纳与总结。但本发明作者认为冯·诺依曼结构计算机处理器在体系结构上的最重要的特点是：面向单道程序执行的体系结构，即在任何时刻，都仅有而且仅能有一道程序在计算机处理器中执行。

在现代计算机的实际应用中，一个用户在一台计算机（PC）上调用多个应用、运行多道程序的情况比比皆是：在用编辑器编写某个文档的同时，从网上下载一个、甚至同时下载多个图片、也许还在播放着一段音乐、防病毒程序正在实时监控、敲击键盘、移动鼠标等。为满足该用户的需求，一个PC不得不花费大量的时间用于在所有这些程序之间频繁地来回切换，致使系统的总体性能大大降低。所有这一切都是由于根据冯·诺依曼计算机原理设计的冯·诺依曼结构计算机处理器不能在同一时刻在同一台计算机处理器上同时并行地执行多道程序。

发明内容

有鉴于此，本发明提出一种在同一时刻在单台计算机处理器上能够并行地执行多道程序的方法及系统，其特征在于在任一台该类计算机处

理器上能够在同一时刻并行地执行多道程序。

如上所述的一种在同一时刻在单台计算机处理器上能够并行地执行多道程序的方法及使用这种方法的计算机处理器，其特征在于：为每一道程序引入了一个程序执行驱动器，从而多个程序执行驱动器可在同一台该类计算机处理器上相互独立地管理与控制多道程序在同一时刻并行地执行。注，组成一道程序的多道程序或子程序亦可有各自的程序执行驱动器。

每道程序在生成可执行代码的同时，生成了该道程序的程序执行驱动器。一个程序执行驱动器至少包含有用于说明与定义该道程序属性特征以及管理与控制该道程序执行所需要的全部相关信息。

计算机处理器的基本构成：

1、系统的硬件部分至少由下述部件组成：一个或多个可并行工作的存储部件，如程序存储部件、数据存储部件、寄存器组等，一个或多个可并行工作的操作控制部件，如控制部件、“译码”部件、算术/逻辑运算部件、外部设备部件等，一个或多个可并行工作的在部件间起连接作用的连通部件，如总线及其控制部件等。上述各类部件通常有一个输入缓存装置与一个输出缓存装置。注：

- 系统中的所有部件可按请求/服务的方式工作：收到请求后，即输入缓存非空时，启动完成指定的服务，结果存入输出缓存。

- 如某被请求服务的部件重复设置了多个，则可任选一个闲者。如果全忙，则等其闲。

- 系统中的部件分为独占型与非独占型。对于独占型部件，或由系统预设或在第一次请求服务时申请。预设或申请的结果保存到该道程序对应的程序执行驱动器的某些项目中。

2、系统的软件部分至少由一个类似于现代计算机系统中的操作系统的系统管理程序模块所组成。该系统管理模块存放在某个存储部件中，至少具有程序调度的功能，并至少维护着一个称为程序调度表的数据结构。系统在初启时就将系统管理模块的程序执行驱动器加载到系统的某操作控制部件中去，称为激活或活化该程序执行驱动器。该程序执行驱

驱动器将驱动系统管理模块逐个激活多个用户程序的程序执行驱动器。此后，各用户程序的程序执行驱动器将相互独立地在同一时刻驱动各自的用户程序在同一台上述计算机处理器中同时并行地执行。

新型计算机处理器的工作方法，即程序执行驱动器驱动程序执行的方法：

1、一道程序一旦生成了可执行代码及其程序执行驱动器，系统管理模块就将该道程序的程序执行驱动器写入到它的程序调度表中。此时，该程序存放在某个存储部件中并处于等待执行状态。

2、该道程序一旦满足被调入执行的各项条件，系统管理模块就将激活或活化该道程序的程序执行驱动器。此后，该道程序进入执行状态，并在它的程序执行驱动器的管理与控制下独立地执行。

3、程序执行驱动器依据它所保存的该道程序的驻存信息（如程序存储部件号、部件内地址 PC 等）控制到相应的存储部件去取当前要执行的指令。

4、指令取出后，该程序执行驱动器即控制到某操作控制部件对已取出的指令进行“译码”。并据此，控制到相应的存储部件去取操作数、控制到相应的操作控制部件完成指定的处理，以及控制将处理的结果写回到相应的存储部件中去。注：

- 此处的“译码”与传统的译码有些许差别，详见后续章节。
- 取出后的指令、指令“译码”的结果、取出的操作数、以及操作的结果均被暂时地保存在该程序执行驱动器的某些单元中。

5、最后，在该程序执行驱动器将它所保存的属于程序驻存信息类中的 PC 项按当前指令规定的方式进行修正后，转步骤 3 以控制执行本道程序的下一条指令。

6、至少在下述情况下，系统管理模块将卸载某道程序的程序执行驱动器：1) 执行到该道程序的结束指令：这将导致 PM 置相应的结束标志并激活系统管理模块的程序执行驱动器驱动系统管理模块对该道程序进行相应的结束处理，最终从程序调度表中删除该道程序的程序执行驱动器。2) 当某个程序执行驱动器驱动的某道程序执行完当前指令后满足

被中断的条件:这将导致 PM 置相应中断标志后保存该程序执行驱动器相应单元的内容,并启动系统管理模块的中断处理例程,即将该例程的程序执行驱动器置入 PM 的输入队列;由该程序执行驱动器驱动中断处理例程去完成有关的中断处理工作:该中断处理例程将把被中断的程序的程序执行驱动器写回到系统的程序调度表中,然后再根据调度规则选择和激活下一道程序的程序执行驱动器以执行该道程序。

很明显,依据本发明提出的计算机处理器工作方法设计的一类新型的计算机处理器可实现指令级的并行。事实上,在同一时刻在同一台该类新型计算机处理器上多个程序执行驱动器相互独立地管理与控制着多道程序同时并行地执行。

在依据本发明提出的计算机处理器工作方法设计的一类新型的计算机处理器中可实现指令级的资源共享。系统中的任何部件,如果当前未被占用,都可被任意一道程序的程序执行驱动器驱动的当前指令所使用。

在依据本发明提出的计算机处理器工作方法设计的一类新型的计算机处理器中,活化程序执行驱动器的个数仅与系统提供资源的总量成正比。在系统资源足够多的条件下,活化程序执行驱动器的数量可足够的大,即系统可支持足够多道的程序在同一时刻在同一台计算机处理器上同时并行地执行。注,当等待执行的程序总数超过系统允许的最大值时,可采用中断切换技术使所有的程序都有机会执行。

依据本发明提出的计算机处理器工作方法设计的一类新型的计算机处理器与传统的多计算机系统、多处理机系统、以及多处理器系统等有着本质的不同,也与采用其它并行处理技术(如重叠技术、先行控制技术、多操作部件技术、流水线技术、超流水线技术、超长指令字技术、超标量技术、多线程技术等)的计算机系统有着本质的不同。特别值得指出的是依据本发明提出的计算机处理器工作方法设计的一类新型的计算机处理器与传统的多计算机系统的差别在于:存在许多这样的情况:多道程序只能在一台计算机处理器上执行而不能分配给多个计算机处理器执行,如鼠标、键盘、显示等系统程序、病毒监控等程序与使用该台计算机处理器的用户程序就不能分配给多个计算机处理器执行而只能在一台

计算机处理器上执行。在这种情况下，传统的多计算机系统只好采用中断切换方式将多道程序转换成一道“宏”程序在一台计算机处理器上执行。而在依据本发明提出的计算机处理器工作方法设计的一类新型的计算机处理器中，上述各道程序均可在同一时刻在单台计算机处理器上并行地执行。

附图说明

图 1 是本发明的一种实施方案的体系结构的原理性示意框图；

图 2 是本发明的上述实施方案中的程序执行驱动器的配送部件的结构框图；

图 3 是本发明的上述实施方案中的配送部件的连通网络装置的交叉互连设计方式的结构框图；

图 4 是本发明的上述实施方案中的配送部件的连通网络装置的多总线互连设计方式的结构框图；

图 5 是本发明的上述实施方案的工作流程图；

具体实施方式

依据本发明提出的计算机处理器工作方法可设计出一类新型结构的计算机处理器（以下简称 L 结构处理器）。图 1 是其中一种实施方案的体系结构的原理性示意框图。鉴于本文的目的在于阐述 L 结构处理器的技术原理而不是设计一台实用的 L 结构处理器，因此对该原理性体系结构示意框图作了许多的简化与省略，如省略了时钟部分、总线部分等；I/O 接口及外部设备部分也仅仅是象征性的说明。

出于同样的原因，本文亦仅设计了一个十分简化的指令系统。

L 结构处理器系统的一个最重要的特征在于：为每一道程序引入了一个程序执行驱动器用于驱动程序的执行。在本示例中，程序执行驱动器被设计成为由一组信息单元组成的信息块。在系统硬件的支持下，这些信息单元组，即程序执行驱动器，按照当前所携带的指令的操作码的要求逐次前往相应的各功能部件对指令的操作数进行指定的功能操作，

如取操作、运算操作、存操作等。程序执行驱动器的组成如下所示：

							7	8	9	10	11	12	13	14-29
							MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

在本示例中，一个程序执行驱动器由 30 个单元组成，各组成单元的定义与说明如下所示。

第 0 号单元 - 第 6 号单元：运行时工作单元：用于本道程序在执行过程中存放临时/中间结果。其中第 0 号单元（称为 NEXT 单元）是本程序执行驱动器当前将要前往请求服务的功能部件的编号；第 1 - 第 6 号单元依当前所驱动的指令的不同而不同，其具体用法在描述各部件功能时有详细的说明。特别需要指出的是第 1 号单元有时被用作 ΔPC 单元，当 $\Delta PC=0$ 时表示当前指令是一条 END 指令，当 $\Delta PC=1$ 时表示当前指令是一条非转移型指令，当 $\Delta PC>1$ 或 $\Delta PC<0$ 时表示当前指令是一条转移型指令。

第 7 号单元：程序存储部件的部件编号 (MPno)：用于存放本道程序驻存的程序存储部件的部件编号。

第 8 号单元：程序存储部件的部件个数 (MPnu)：当一道程序所需要占用的程序空间大于一个程序存储部件的容量时，可按顺序存放在邻接的多个连续的程序存储部件中。程序存储部件个数单元用于存放本道程序所占用的连续的程序存储部件的个数。如果 MPnu 的值大于 1，则 MPno 的值是第一个程序存储部件的部件编号。

第 9 号单元：程序存储部件的部件内地址 (PC)：用于存放本道程序当前执行指令的程序存储部件的部件内地址。

第 10 号单元：数据存储部件的部件编号 (MDno)：用于存放本道程序所占用的数据存储部件的部件编号。

第 11 号单元：数据存储部件部件个数 (MDnu)：当一道程序所需要占用的数据空间大于一个数据存储部件的容量时，可按顺序占用邻接的多个连续的数据存储部件。数据存储部件个数单元用于存放本道程序所

占用的连续的数据存储部件的个数。如果 MDnu 的值大于 1, 则 MDno 的值是第一个数据存储部件的部件编号;

第 12 号单元: 执行指令条数(Ins): 用于记录本道程序本次调入后累计执行的指令条数。本参数在本实施例中用在中断处理中作为中断条件。

第 13 号单元: 程序道号(PrNo): 当两道以上正在执行的程序共同存放在同一个程序存储部件中时, 用本参数标识各道程序。

第 14 号单元 - 第 29 号单元: 16 个通用寄存器(R0 - R15): 在本实施例中, 将 16 个通用寄存器(R0 ... R15)设计成为程序执行驱动器的组成部分。

在传统计算机中, 寄存器组通常被设计成为一种硬件部件, 由硬件的存储电器装置和读/写电器装置构成。为区别起见, 称此种寄存器组设计方式为固定式寄存器组设计方式。对于 L 结构处理器, 寄存器组可按固定式硬件方式进行设计。然而在本实施例中, 寄存器组被设计成为程序执行驱动器的一个组成部分、且随着程序执行驱动器来往于各部件之间。称此种寄存器组设计方式为动态寄存器组设计方式。这样设计的最大好处是不仅消除了读/写寄存器所花费的传送时间, 而且也消除了各道程序同时访问寄存器组时造成的等待延时。

图 1 列出了本发明的上述实施例的系统各硬件功能部件。这些功能部件的主要功能及其输入/输出的程序执行驱动器的内容如下所述。

1、 存储功能部件

1) 部件 1: 称为程序存储部件 MP, 用于提供有关读/写程序存储部件的服务。可设置多个可独立并行工作的 MP 部件。每个 MP 部件的主要功能如下:

- MP 维护着一个输入队列, 所有请求本 MP 处理的程序执行驱动器都等待在该队列中。MP 也维护着一个输出队列, 所有已处理的程序执行驱动器都等待在该队列中等待 DU 部件的配送服务。

- 读请求: 如果当前队首的程序执行驱动器请求的是读操作, 则从该程序执行驱动器中取出该道程序的部件内地址 PC 送 MP 的地址寄存

器，并启动 MP 的读操作读取该道程序的当前指令；读出的指令写入到当前的程序执行驱动器的相应单元中；置程序执行驱动器的 NEXT 单元为 IT；然后将其写入到 MP 的输出队列中。

- 写请求：如果当前队首的程序执行驱动器请求的是读操作，需完成的操作与数据存储部件的写操作类似，此处予以省略。

- 重复上述处理过程；如果本 MP 的输入队列为空或输出队列为满，则进行一次空操作。

- 在 MP 部件的输入队列中的程序执行驱动器的内容如下所示。

0	7	8	9	10	11	12	13	14-29
MPr	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 0 号单元中的 MPr 表示目的部件是 MP 且要求进行“读”操作；注，限于篇幅，凡当前未涉及到的程序执行驱动器的组成单元，如第 1 号单元至第 6 号单元，均予以省缺，以下相同。

- 在 MP 部件的输出队列中的程序执行驱动器的内容如下所示。其中第 1 号单元中的 I 为读出的当前指令，如 LDI R0, C。

0	1	7	8	9	10	11	12	13	14-29
IT	I	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

注，在本实施例中略去了如何将程序本身写入程序存储部件的过程，也因此略去了有关的程序执行驱动器的内容；事实上，该过程与将数据写入相应的数据存储部件的过程类似。

2) 部件 2：称为数据存储部件 MD，用于提供有关读/写数据存储部件的服务。可设置多个可独立并行工作的 MD 部件。每个 MD 部件的主要功能如下：

- MD 维护着一个输入队列，所有请求本 MD 处理的程序执行驱动器

都等待在该队列中。MD 也维护着一个输出队列，所有已处理的程序执行驱动器都等待在该队列中等待 DU 部件的配送服务。

- 读操作：如果当前队首的程序执行驱动器请求的是读操作，则从该程序执行驱动器中取出该读操作的部件内地址送 MD 的地址寄存器，并启动 MD 的读操作读取相应存储单元的内容；读出的数据写入到当前的程序执行驱动器的某单元中；置 NEXT 单元为 PM；置 Δ PC 单元为 1；然后将其写入到 MD 的输出队列中。

- 写操作：如果当前队首的程序执行驱动器请求的是写操作，则从该程序执行驱动器中取出该写操作的数据及部件内地址分别送 MD 的数据寄存器和地址寄存器，并启动 MD 的写操作将数据写入到相应的 MD 存储单元中去；置 NEXT 单元为 PM；置 Δ PC 单元为 1；然后将其写入到 MD 的输出队列中。

- 重复上述处理过程，如果本 MD 的输入队列为空或输出队列为满；则进行一次空操作。

- 在 MD 部件的输入队列中的程序执行驱动器的内容可参见 IT 部件的输出队列中发往 MD 的程序执行驱动器的内容。

- 在 MD 部件的输出队列中发往 PM 的程序执行驱动器的内容如下所示：

0	1	7	8	9	10	11	12	13	14-29
PM	1	MPno	MPnu	PC	MDno	MDnu	IIIns	PrNo	R0-R15

其中第 1 号单元为 Δ PC，当前指令的当前值是 1。

2、 操作功能部件

1) 部件 3：称为程序管理部件 PM，用于提供有关管理程序执行驱动器运行的服务。可设置多个可独立并行工作的 PM 部件。每个 PM 部件的主要功能如下：

- PM 维护着一个输入队列，所有请求本 PM 处理的程序执行驱动器都等待在该队列中。PM 也维护着一个输出队列，所有已处理的程序执行驱动器都等待在该队列中等待 DU 部件的配送服务。

- 启动一个程序执行驱动器的运行：系统管理模块激活一道程序

的程序执行驱动器实际上就是将该道程序的程序执行驱动器写入 PM 的输入队列。当 PM 处理到该程序执行驱动器时,将其 NEXT 单元置为 MP(取指令),然后将其写入到 PM 的输出队列中。

- 继续一个程序执行驱动器的运行: 每个程序执行驱动器驱动一道程序的当前指令执行的最后一步是返回到 PM。如果当前指令不是一条结束指令,则 PM 将控制该程序执行驱动器继续运行: 第 9 号单元 $\leftarrow$ 第 9 号单元 + 第 1 号单元, 即 $PC \leftarrow PC + \Delta PC$; 置 NEXT 单元为 MP; 然后将其写入到 PM 的输出队列中。

- 中断一个程序执行驱动器的运行: 当某个程序执行驱动器驱动的某道程序执行到当前指令满足被中断的条件时, PM 置相应的中断标志、保存该程序执行驱动器相应单元的内容、并启动系统管理模块的中断处理例程,即将该例程的程序执行驱动器置入 PM 的输入队列,由该程序执行驱动器驱动中断处理例程去完成有关的中断处理工作。该中断处理例程将把被中断的程序的程序执行驱动器写回到系统的程序调度表中,然后再根据调度规则选择和激活下一道程序的程序执行驱动器以执行该道程序。

- 终止一个程序执行驱动器的运行: 当某道程序执行到一条结束指令时, PM 将通过采用删除该道程序的程序执行驱动器的方式来终止该道程序的执行。注,该道程序在执行结束指令前,须执行一条或多条资源释放型指令。

- 重复上述处理过程;如果本 PM 的输入队列为空或输出队列为满,则进行一次空操作。

- 在 PM 部件的输入队列中接收到的程序执行驱动器的内容如下所示:

0	1	7	8	9	10	11	12	13	14-29
PM	PC	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 1 号单元为 ΔPC , 其值或是 0 或是 1 或是转移的偏移量。

- 在 PM 部件的输出队列中发往 MP 部件的程序执行驱动器的内容如下所示:

0	7	8	9	10	11	12	13	14-29
MPr	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 0 号单元中的 MPr 表示目的部件是 MP 且要求进行“读”操作；

2) 部件 4: 称为指令格式转换部件 IT, 用于提供有关指令译码和指令格式转换的服务。可设置多个可独立并行工作的 IT 部件。每个 IT 的主要功能如下:

- IT 维护着一个输入队列, 所有请求本 IT 处理的程序执行驱动器都等待在该队列中。IT 也维护着一个输出队列, 所有已处理的程序执行驱动器都等待在该队列中等待 DU 部件的配送服务。

- 对于请求指令格式转换的程序执行驱动器, IT 将对该程序执行驱动器中存放的当前指令进行格式转换, 并据此设置 NEXT 等单元, 然后将其写入到 IT 的输出队列中。

- 重复上述处理过程; 如果本 IT 的输入队列为空或输出队列为满, 则进行一次空操作。

- 在 IT 部件的输入队列中的程序执行驱动器的内容如下所示。其中第 1 号单元中的 I 为读出的当前指令, 如 LDI R0, C。

0	1	7	8	9	10	11	12	13	14-29
IT	I	MPno	MPnu	PC	Mdno	MDnu	Ins	PrNo	R0-R15

- 在 IT 部件的输出队列中的程序执行驱动器的内容根据当前指令 I 的不同而不同, 如下所示。

△ 由 IT 发往 PM 的程序执行驱动器的内容:

LDI Rd, C

0	1	7	8	9	10	11	12	13	14-29
PM	1	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 1 号单元为△PC, 当前指令的当前值是 1。

MV Rd, Rs

0	1	7	8	9	10	11	12	13	14-29
---	---	---	---	---	----	----	----	----	-------

PM	1	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15
----	---	------	------	----	------	------	-----	------	--------

其中第 1 号单元为 $\triangle$ PC，当前指令的当前值是 1。

$\triangle$ 由 IT 发往 MD 的程序执行驱动器的内容：

LD Rd, Rs

0	1	2	3	7	8	9	10	11	12	13	14-29
MDr	PM	[Rd]	[Rs]	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 0 号单元中的 MDr 表示目的部件是 MD 且要求进行“读”操作；第 2, 3 号单元中的[Rd]与[Rs]表示寄存器 Rd 与寄存器 Rs 中的内容。

STI Rd, C

0	1	2	3	7	8	9	10	11	12	13	14-29
MDIw	PM	[Rd]	C	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 0 号单元中的 MDIw 表示目的部件是 MD 且要求进行“立即数写”操作；第 2 号单元中的[Rd]表示寄存器 Rd 中的内容。

ST Rd, Rs

0	1	2	3	7	8	9	10	11	12	13	14-29
MDw	PM	[Rd]	[Rs]	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 0 号单元中的 MDw 表示目的部件是 MD 且要求进行“写”操作；第 2, 3 号单元中的[Rd]与[Rs]表示寄存器 Rd 与寄存器 Rs 中的内容。

$\triangle$ 由 IT 发往 ALU 的程序执行驱动器的内容：

ADDI Rd, C

0	1	2	3	4	7	8	9	10	11	12	13	14-29
ALUI+	PM	Rd	[Rd]	C	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 0 号单元中的 ALUI+表示目的部件是 ALU 且要求进行“立即数的加”操作。对于 SUBI, MULI, DIVI 等指令，其相应的程序执行驱动器的内容与此类似，仅是第 0 号单元分别置为是 ALUI-、ALUI*、ALUI÷。

ADD Rd, Rs1, Rs2

0	1	2	3	4	7	8	9	10	11	12	13	14-29
ALU+	PM	Rd	[Rs1]	[Rs2]	MPno	Mpnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 0 号单元中的 ALU+表示目的部件是 ALU 且要求进行“加”操作；SUB, MUL, DIV 等与此类似。

JEQI Rs, C

0	1	2	3	7	8	9	10	11	12	13	14-29
ALUIj	PM	C	[Rs]	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 0 号单元中的 ALUIj 表示目的部件是 ALU 且要求进行“立即数的转移判断”操作；JNEI 等与此类似。

JEQ Rs1, Rs2

0	1	2	3	7	8	9	10	11	12	13	14-29
ALUj	PM	[Rs2]	[Rs1]	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 0 号单元中的 ALUj 表示目的部件是 ALU 且要求进行“转移判断”操作；JNE 等与此类似。

SLT Rd, Rs1, Rs2

0	1	2	3	4	7	8	9	10	11	12	13	14-29
ALUslt	PM	Rd	[Rs1]	[Rs2]	MPno	MPnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第 0 号单元中的 ALUslt 表示目的部件是 ALU 且要求进行“小于设置”操作；SLE、SGT、SGE、SEQ、SNE 等与此类似。

3) 部件 5: 称为算术/逻辑运算部件 ALU, 用于提供有关算术/逻辑运算的服务。可设置多个可独立并行工作的 ALU 部件。每个 ALU 部件的主要功能如下:

- ALU 维护着一个输入队列, 所有请求本 ALU 处理的程序执行驱动器都等待在该队列中。ALU 也维护着一个输出队列, 所有已处理的程序执行驱动器都等待在该队列中等待 DU 部件的配送服务。

- 在请求 ALU 进行算术/逻辑运算的服务之前, 程序执行驱动器已控制将该指令所需的操作数准备好且存放在程序执行驱动器的相应单元中。ALU 对存放在程序执行驱动器中的相应的操作数进行所要求的算术/逻辑运算, 结果存放在程序执行驱动器的相应的单元中; 置 NEXT 单元 PM; 按指令的规定置第 1 号单元, 即 Δ PC 单元; 然后将其写入到 ALU 的

输出队列中。

- 重复上述处理过程;如果本ALU的输入队列为空或输出队列为满,则进行一次空操作。

- 在ALU部件的输入队列中的程序执行驱动器的内容可参见IT部件的输出队列中的发往ALU的程序执行驱动器的内容。

- 在ALU部件的输出队列中的程序执行驱动器的内容如下所示。

0	1	7	8	9	10	11	12	13	14-29
PM	Δ PC	MPno	Mpnu	PC	MDno	MDnu	Ins	PrNo	R0-R15

其中第1号单元为 Δ PC,其值或是1或是转移的偏移量。

4) 部件6:称为外部设备管理部件PD,用于提供管理外部设备,如显示、键盘、鼠标、打印、通信等设备,与主机之间进行数据输入/输出的服务。

3、 部件7:称为程序执行驱动器配送部件DU,用于提供将程序执行驱动器从一个部件的输出队列配送到另一个部件的输入队列的运输与配送服务。可设置多个可独立并行工作的DU部件。每个DU部件的主要功能如下:

- DU按预定的顺序逐个查询各部件组的各部件的输出队列是否为非空。对于某非空的输出队列,DU从队中取出一程序执行驱动器,并按照此程序执行驱动器的NEXT单元所指示的目的部件组编号以及目的部件编号查询该目的部件组的目的部件的输入队列是否为非满。如果是非满,则将该程序执行驱动器配送到该目的部件组的目的部件的输入队列中去。注,以上的处理方式是针对PM、MP和MD而言。对于IT和ALU,只需在该部件中任选一输入队列为非满的部件即可。

- 重复上述处理过程;如果所有部件组的所有部件的输出队列全为空,或者虽有输出队列为非空,但相对应的输入队列全为满,则进行一次空操作。

图1亦给出了本发明的上述实施例中系统各部件的连接关系:部件1到6分别通过各自的一或两组信号线将各自的输入队列的尾端与输出队列的首端连到部件7;

图2是本发明的上述实施例的部件7的结构框图。部件7由多个数据输入端口、多个数据输出端口、连通网络装置与控制管理装置等所组成。部件7的控制管理装置用于选通的控制与管理，如各端口忙闲的探测、优先级的确定、路径的选择等。部件7的连通网络装置用于将多个数据输入端口连到多个数据输出端口。该连通网络装置的设计有多种方法，以下列举其中较为典型的两种。

图3是本发明的上述实施例中的配送部件的连通网络装置的交叉互连设计方式的结构框图。部件7的连通网络装置采用交叉互连结构，任何一个输入端口都与所有或部分输出端口相连。

图4是本发明的上述实施例中的配送部件的多总线互连设计方式的结构框图。部件7的连通网络装置采用多总线结构，所有的输入端口与输出端口被分成几组，分别连到一条总线上；总线与总线之间再交叉互连。

图5是本发明的上述实施例的工作流程图，也是在新型计算机中程序执行驱动器的工作流程图。其具体步骤如下：

步骤S1，系统初启时，至少有一道程序，如系统管理模块，的程序执行驱动器存放在某个程序管理部件PM的输出队列中；下转步骤S2；

步骤S2，当某个DU部件检测到某个PM部件的输出队列为非空、且当前处于队首的程序执行驱动器的NEXT单元所指向的那个特定的目的MP部件的输入队列为非满时，将该PM部件的输出队列队首的程序执行驱动器配送到相应的MP部件的输入队列中等待该部件的处理；下转步骤S3；

步骤S3，当某个MP部件检测到其输入队列为非空时，按输入队列队首处的程序执行驱动器中PC单元所指定的PC地址到它所管理的程序存储部件中读取指令，取出的指令存放到该程序执行驱动器的第1号单元中；如果此时该MP部件的输出队列为满状态，则等待其空；当有空位时，就将处理结束的程序执行驱动器写入到它的输出队列中等待DU的配送处理；下转步骤S4；

步骤S4，当某个DU部件检测到某个MP部件的输出队列为非空、且

当前处于队首的程序执行驱动器的 NEXT 单元所指向的目的部件是 IT、且至少有一个 IT 部件的输入队列为非满时，将该 MP 部件的输出队列队首的程序执行驱动器配送到相应的 IT 部件的输入队列中等待该部件的处理；下转步骤 S5；

步骤 S5，当某个 IT 部件检测到其输入队列为非空时，对输入队列队首处的程序执行驱动器中的指令进行指令的格式变换，变换后的结果存放到该程序执行驱动器的相应单元中；如果此时该 IT 部件的输出队列为满状态，则等待其空；当有空位时，就将处理结束的程序执行驱动器写入到它的输出队列中等待 DU 的配送处理；下转步骤 S6；

步骤 S6，当某个 DU 部件检测到某个部件，这里的某个部件是指 IT、ALU 和 MD 中的任何一个部件，的输出队列为非空、且当前处于队首的程序执行驱动器的 NEXT 单元所指向的目的部件的输入队列为非满时，将该部件的输出队列队首的程序执行驱动器配送到相应的目的部件的输入队列中等待该部件的处理。此时的目的部件有 3 种，一是某个 ALU 部件，此时的目的 ALU 部件可以是任意一个输出队列有空位的 ALU 部件；以下转步骤 S7；二是某个 MD 部件，此时的目的 MD 部件必须是上述程序执行驱动器的 NEXT 单元所特指的哪个 MD 部件；以下转步骤 S8；最后一个是某个 PM 部件，此时的目的 PM 部件必须是上述程序执行驱动器的 NEXT 单元所特指的哪个 PM 部件；以下转步骤 S9；

步骤 S7，当某个 ALU 部件检测到其输入队列为非空时，按输入队列队首处的程序执行驱动器中的 NEXT 单元中的命令对该程序执行驱动器中的相应单元所保存的操作数进行指定的算术/逻辑运算，运算后的结果存放到该程序执行驱动器的相应单元中；如果此时该 ALU 部件的输出队列为满状态，则等待其空；当有空位时，就将处理结束的程序执行驱动器写入到它的输出队列中等待 DU 的配送处理；此时的目的部件是 PM 部件，下转步骤 S6；

步骤 S8，当某个 MD 部件检测到其输入队列为非空时，按输入队列队首处的程序执行驱动器中的 NEXT 单元中的读或写命令以及该程序执行驱动器中的某单元所指定的部件内地址对该程序执行驱动器中的某单

元的数据在它所管理的数据存储部件中进行读/写操作；如果是读操作，将读出的数据存放到该程序执行驱动器的某单元中；如果此时该 MD 部件的输出队列为满状态，则等待其空；当有空位时，就将处理结束的程序执行驱动器写入到它的输出队列中等待 DU 的配送处理；此时的目的部件是 PM 部件，下转步骤 S6；

步骤 S9，当 PM 的某个部件检测到其输入队列为非空时，按其输入队列队首处的程序执行驱动器中的命令要求进行相应的操作：1) 非 END 指令，此时 ΔPC 单元为非 0 值：PM 按要求修正其 PC 等单元的值，结果存放到该程序执行驱动器的相应单元中；如果此时该 PM 部件的输出队列为满状态，则等待其空；当有空位时，就将处理结束的程序执行驱动器写入到它的输出队列中等待 DU 的配送处理；下转步骤 S2。2) END 指令，此时 ΔPC 单元为 0：PM 置相应结束标志，从输入队列中删除该程序执行驱动器，激活系统管理模块调度其它程序执行。3) 当某个程序执行驱动器驱动的某道程序执行完当前指令后满足被中断的条件时：PM 置相应中断标志后保存该程序执行驱动器相应单元的内容，并启动系统管理模块的中断处理例程，即将该例程的程序执行驱动器置入 PM 的输入队列；由该程序执行驱动器驱动中断处理例程去完成有关的中断处理工作：该中断处理例程将把被中断的程序的程序执行驱动器写回到系统的程序调度表中，然后再根据调度规则选择和激活下一道程序的程序执行驱动器以执行该道程序。

系统的软件部分由一道简化的系统管理程序模块所组成。为重点说明本发明的技术原理，该系统管理模块仅设置了简化的程序调度功能和简化的中断处理功能，并为此维护一张程序调度表。在系统初启时，该系统管理模块的程序执行驱动器就被加载到系统的 PM 部件中。该程序执行驱动器驱动系统管理模块逐个激活用户程序的程序执行驱动器。当等待执行的程序总数超过系统允许的最大值时，系统管理模块将通过中断机制切换程序的执行。

本文所述仅为本发明技术构思下的一些基本说明。存在多种依据本发明提出的计算机处理器工作方法设计计算机处理器的方式与方法，也

存在多种实现程序执行驱动器以及实现用程序执行驱动器来管理与控制程序执行的方式与方法。而依据本发明的技术构想所做的任何等效变换，均应属于本发明的保护范围。

本发明作者为论证本发明提出的计算机处理器工作方法以及按此方法设计计算机处理器的可行性，分别用 C++ 开发了两个模拟系统，一是冯·诺依曼结构计算机处理器，一是使用本发明提出的计算机处理器工作方法的计算机处理器，即 L 结构处理器。两种计算机处理器系统采用相同的系统参数，如存储器读/写周期、ALU 运算周期等。执行 100 万条指令的模拟结果显示 L 结构处理器比冯·诺依曼结构处理器快 2 到 10 倍，而且进一步提高的余地很大。

下面是本发明的上述实施方案的指令系统设计；

简化的指令系统

MOVE 指令: MOV Rd, Rs $Rd \leftarrow Rs$

LOAD 指令: LD Rd, Rs $Rd \leftarrow DM[Rs]$

 LDI Rd, C $Rd \leftarrow C$

STORE 指令: ST Rd, Rs $DM[Rd] \leftarrow Rs$

 STI Rd, C $DM[Rd] \leftarrow C$

转移指令: JEQ Rs1, Rs2 IF ($Rs1=0$) $PC \leftarrow PC+Rs2$ ELSE $PC \leftarrow PC+1$

 JNE Rs1, Rs2 IF ($Rs1 \neq 0$) $PC \leftarrow PC+Rs2$ ELSE $PC \leftarrow PC+1$

 JEQI Rs1, C IF ($Rs1=0$) $PC \leftarrow PC+C$ ELSE $PC \leftarrow PC+1$

 JNEI Rs1, C IF ($Rs1 \neq 0$) $PC \leftarrow PC+C$ ELSE $PC \leftarrow PC+1$

运算指令: ADD Rd, Rs1, Rs2 $Rd \leftarrow Rs1+Rs2$

 SUB Rd, Rs1, Rs2 $Rd \leftarrow Rs1-Rs2$

 MUL Rd, Rs1, Rs2 $Rd \leftarrow Rs1*Rs2$

 ADDI Rs, C $Rs \leftarrow Rs+C$

 SUBI Rs, C $Rs \leftarrow Rs-C$

 MULI Rs, C $Rs \leftarrow Rs*C$

逻辑指令: OR Rd, Rs1, Rs2 $Rd \leftarrow Rs1 \text{ OR } Rs2$

AND Rd, Rs1, Rs2 $Rs \leftarrow Rs1 \text{ AND } Rs2$

ORI Rs, C $Rs \leftarrow Rs \text{ OR } C$

ANDI Rs, C $Rs \leftarrow Rs \text{ AND } C$

设置指令: SLT Rd, Rs1, Rs2 IF($Rs1 < Rs2$) $Rd \leftarrow 1$ ELSE $Rd \leftarrow 0$

SLE Rd, Rs1, Rs2 IF($Rs1 \leq Rs2$) $Rd \leftarrow 1$ ELSE $Rd \leftarrow 0$

SGT Rd, Rs1, Rs2 IF($Rs1 > Rs2$) $Rd \leftarrow 1$ ELSE $Rd \leftarrow 0$

SGE Rd, Rs1, Rs2 IF($Rs1 \geq Rs2$) $Rd \leftarrow 1$ ELSE $Rd \leftarrow 0$

SEQ Rd, Rs1, Rs2 IF($Rs1 = Rs2$) $Rd \leftarrow 1$ ELSE $Rd \leftarrow 0$

SNE Rd, Rs1, Rs2 IF($Rs1 \neq Rs2$) $Rd \leftarrow 1$ ELSE $Rd \leftarrow 0$

资源管理指令: GETMD Rd, Rs 申请占用[Rs]个连续的 MD 部件,
 $Rd \leftarrow$ 获准的 MD 部件编号。

GETMDI Rd, C 申请占用 C 个连续的 MD 部件
 $Rd \leftarrow$ 获准的 MD 部件编号。

RETMD Rs1, Rs2 申请释放由[Rs1]开始的[Rs2]个
连续的 MD 部件。

GETMDI Rs, C 申请释放由[Rs1]开始的 C 个连续
的 MD 部件。

结束指令: END 程序结束。

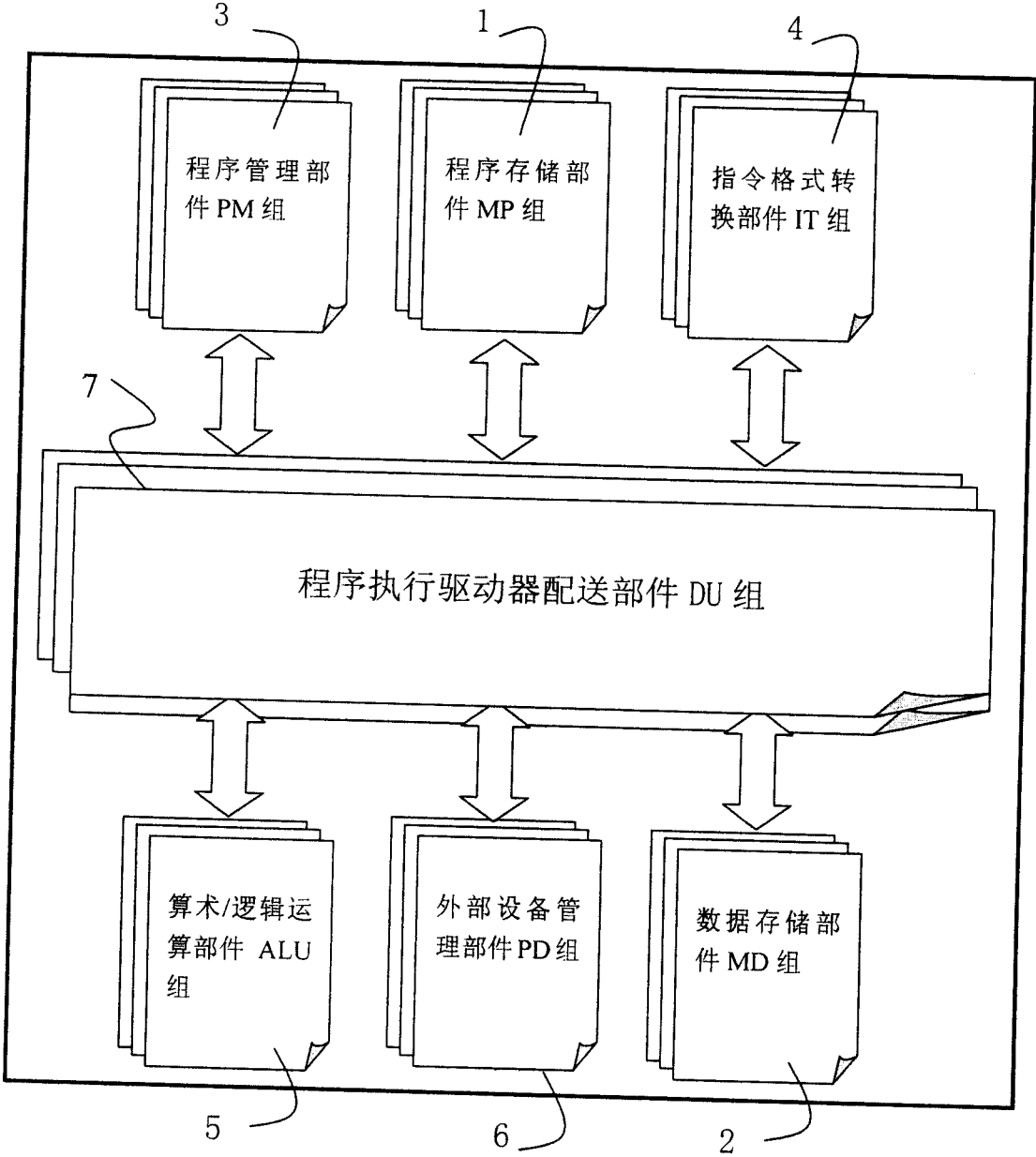


图 1

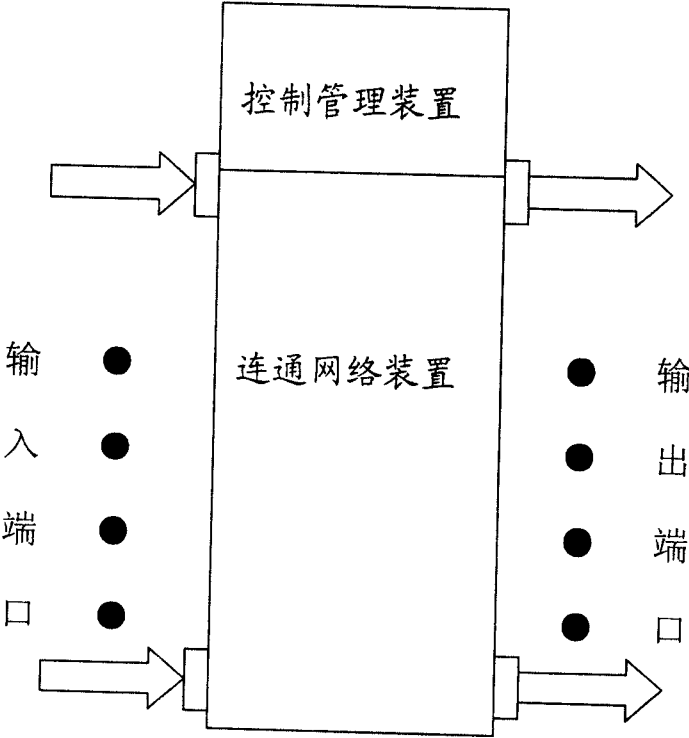


图 2

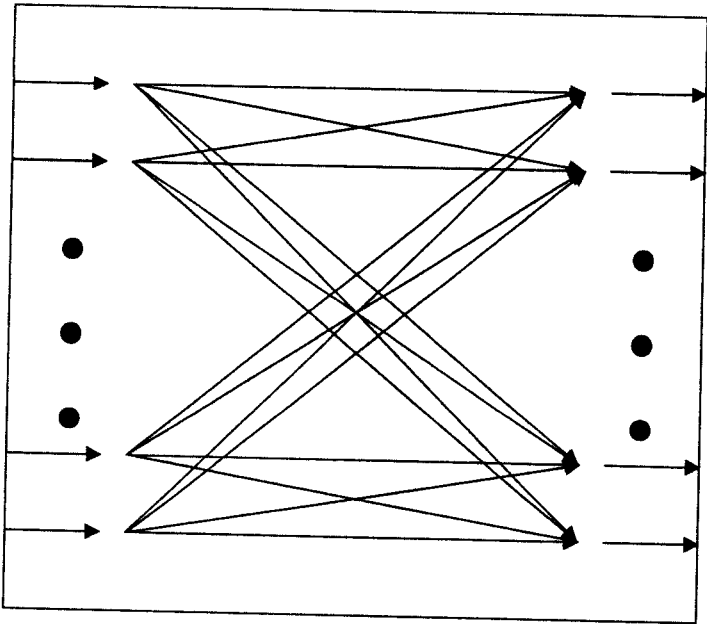


图 3

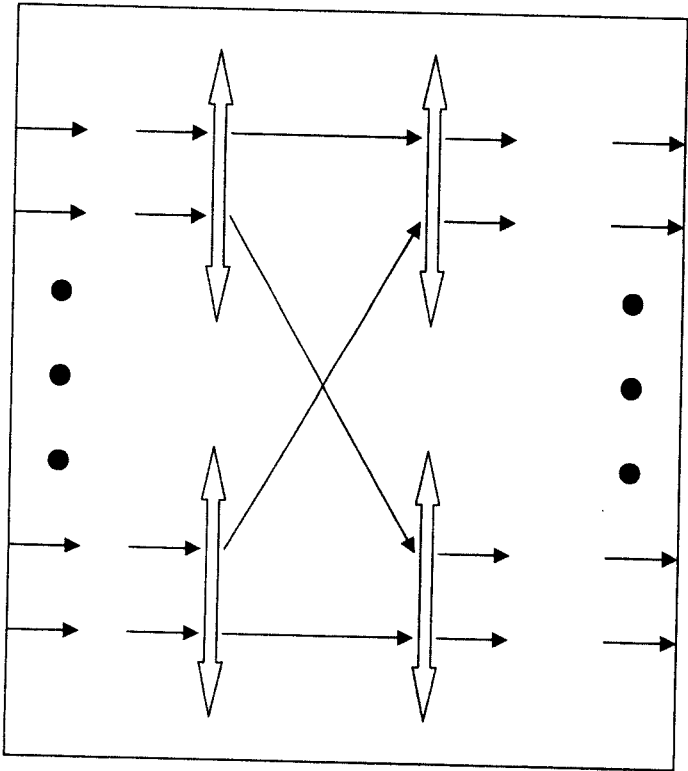


图 4

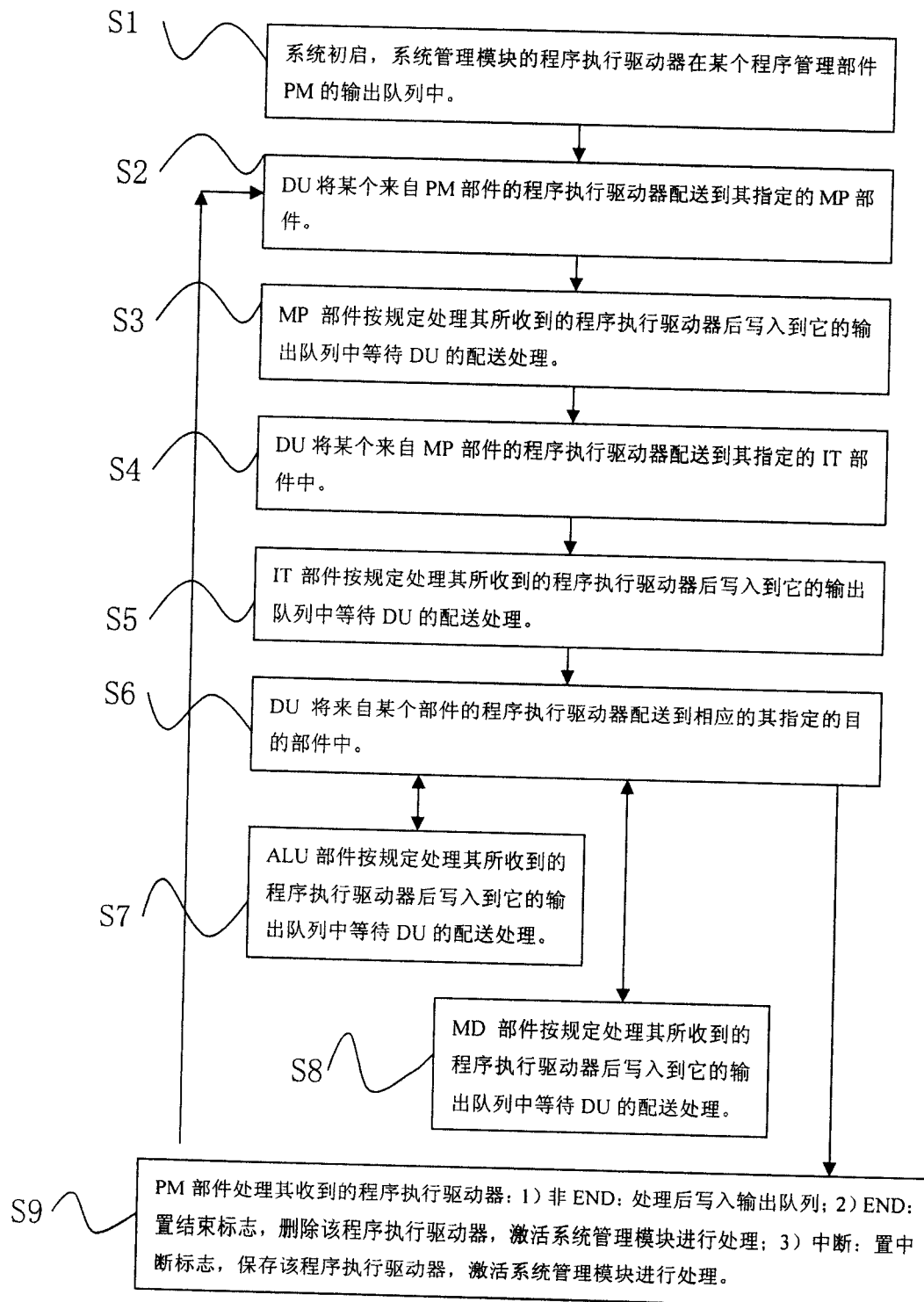


图 5