(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104679481 A
(43)申请公布日 2015.06.03

(21)申请号 201410697780.X

(22)申请日 2014.11.26

(66)本国优先权数据
201310625156.4 2013.11.27 CN

(71)申请人 上海芯豪微电子有限公司
地址 200092 上海市杨浦区四平路1398号B座1202

(72)发明人 林正浩

(51)Int.Cl.
G06F 9/32(2006.01)
G06F 12/10(2006.01)

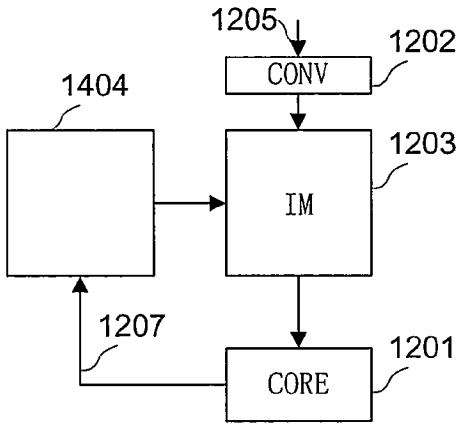
权利要求书8页 说明书61页 附图22页

(54)发明名称

一种指令集转换系统和方法

(57)摘要

本发明提供了一种指令集转换系统和方法，能将外部指令转换为内部指令供处理器核执行，且可以通过配置，很方便地扩展处理器系统支持的指令集；本发明还提供了内部指令地址和外部指令地址的实时转换方法，使得处理器核能直接在较高层次缓存中读取内部指令，减少流水线深度。



CN 104679481 A

1. 一种指令集转换方法，其特征在于，包括：

将外部指令转换为内部指令，并建立外部指令地址和内部指令地址之间的映射关系；

将所述内部指令存储在处理器核能直接访问的缓存中；和

直接根据该内部指令地址对缓存寻址读出相应的内部指令供处理器核执行；或

根据所述映射关系将处理器核输出的外部指令地址转换为内部指令地址后，对缓存寻址读出相应的内部指令供处理器核执行。

2. 如权利要求 1 所述的指令集转换方法，其特征在于，根据程序执行流及处理器核执行指令的反馈向处理器核提供后续指令；所述处理器核执行指令的反馈可以是处理器核执行分支指令时产生的分支转移是否发生的信号。

3. 如权利要求 1 所述的指令集转换方法，其特征在于，对于需要被转换的外部指令：

提取出外部指令中包含指令类型在内的各个指令域；

根据提取出的指令类型查找对应的内部指令的指令类型和指令转换控制信息；

根据所述指令转换控制信息对提取出的相应指令域进行移位；和

对所述内部指令类型及移位后的指令域进行拼接，构成相应的内部指令，从而将外部指令转换为内部指令。

4. 如权利要求 3 所述的指令集转换方法，其特征在于，

一条外部指令被转换为一条内部指令；其中，该外部指令的指令地址对应内部指令的指令地址；或

一条外部指令被转换为多条内部指令；其中，该外部指令的指令地址对应所述多条内部指令中第一条内部指令的指令地址。

5. 如权利要求 4 所述的指令集转换方法，其特征在于，

多条外部指令被转换为一条内部指令；其中，所述多条外部指令中第一条外部指令的指令地址对应该内部指令的指令地址。

6. 如权利要求 3 所述的指令集转换方法，其特征在于，建立外部指令地址和内部指令地址之间的映射关系。

7. 如权利要求 6 所述的指令集转换方法，其特征在于，所述外部指令地址和内部指令地址之间的映射关系包括：

外部指令地址和内部指令块地址之间的映射关系；

外部指令块内地址和内部指令块内地址之间的映射关系。

8. 如权利要求 7 所述的指令集转换方法，其特征在于，可以用一种数据结构表示外部指令地址和内部指令块地址之间的映射关系；

所述数据结构中存储了内部指令块地址，且所述内部指令块地址同时按外部指令块地址和外部指令块内地址进行排序。

9. 如权利要求 8 所述的指令集转换方法，其特征在于，在所述数据结构中，如果一个外部指令地址对应的内部指令块地址存在，则可以根据所述外部指令地址中的外部指令块地址和外部指令块内地址，在该数据结构中找到对应的位置，读出其中存储的内部指令块地址。

10. 如权利要求 8 所述的指令集转换方法，其特征在于，在所述数据结构中，如果一个外部指令地址对应的内部指令块地址不存在，则可以根据所述外部指令地址中的外部指令

块地址和外部指令块内地址，找到其插入位置，并在位置中存储该外部指令地址对应的内部指令块地址。

11. 如权利要求 7 所述的指令集转换方法，其特征在于，根据所述外部指令块地址和内部指令块地址之间的映射关系，可以对外部指令地址进行转换得到对应的内部指令块地址。

12. 如权利要求 11 所述的指令集转换方法，其特征在于，根据所述外部指令块内地址和内部指令块内地址之间的映射关系，可以对外部指令块内地址进行转换得到对应的内部指令块内地址。

13. 如权利要求 6 所述的指令集转换方法，其特征在于，对于任意一个外部指令地址，通过正向移位逻辑，从初始值开始，对从该地址所在的外部指令块起始地址开始至该外部指令地址之间的外部指令条数进行计数；其中，每经过一条所述外部指令，正向移一位，最终得到一个移位结果；

通过反向移位逻辑，从所述外部指令块对应的内部指令块的起始地址开始对每条外部指令对应的第一条内部指令的条数进行计数；其中，每经过一条所述内部指令，反向移一位，直到移位结果恢复为所述初始值；和

此时对应的内部指令块内地址即对应所述外部指令的块内地址。

14. 如权利要求 6 所述的指令集转换方法，其特征在于，通过地址计算，将栈寄存器操作转换为对寄存器堆的操作，使得处理器核内部的寄存器堆能作为栈寄存器使用。

15. 如权利要求 6 所述的指令集转换方法，其特征在于，所述转换能将一种或多种指令集的指令转换为一种指令集的指令。

16. 一种指令集转换系统，其特征在于，包括：

处理器核，用于执行内部指令；

转换器，用于将外部指令转换为内部指令，并建立外部指令地址和内部指令地址之间的映射关系；

地址映射模块，用于存储所述外部指令地址和内部指令地址之间的映射关系，并对外部指令地址和内部指令地址之间进行转换；

缓存，用于存储转换得到的内部指令，并根据内部指令地址输出相应内部供处理器核执行。

17. 如权利要求 16 所述的指令集转换系统，其特征在于，所述转换器进一步包括：

存储器，用于存储外部指令类型与内部指令类型的对应关系，及相应外部指令和内部指令之间各个指令域的对应关系；

对齐器，用于将外部指令移位对齐，并在外部指令跨越指令块边界的情况下，将该外部指令移位到一个指令块并对齐；

提取器，用于提取出外部指令中的各个指令域；其中，提取出的指令类型被用于对所述存储器寻址，以读出所述外部指令对应的指令转换控制信息及相应的内部指令类型，并根据所述控制信息对提取出的指令域进行移位；

指令拼接器，用于对所述内部指令类型和移位后的指令域进行拼接，构成内部指令。

18. 如权利要求 17 所述的指令集转换系统，其特征在于，所述地址映射模块进一步包括：

块地址映射模块，用于存储外部指令块地址与内部指令块地址之间的映射关系，并将外部指令块地址转换为内部指令块地址；和

偏移地址映射模块，用于存储外部指令块内地址与内部指令块内地址之间的映射关系，并将外部指令块内地址转换为内部指令块内地址。

19. 如权利要求 18 所述的指令集转换系统，其特征在于，所述系统还包括一个循迹系统；所述循迹系统根据存储在其中的程序执行流及处理器核执行指令的反馈，同时对所述程序执行流及缓存寻址，并从缓存中读出后续指令送往处理器核供执行；

所述处理器核执行指令的反馈可以是处理器核执行分支指令时产生的分支转移是否发生的信号。

20. 如权利要求 19 所述的指令集转换系统，其特征在于，地址映射模块中还包含一个正向移位逻辑和一个反向移位逻辑；

对于任意一个外部指令地址，通过正向移位逻辑，从初始值开始，对从该地址所在的外部指令块起始地址开始至该外部指令地址之间的外部指令条数进行计数；其中，每经过一条所述外部指令，正向移一位，最终得到一个移位结果；

通过反向移位逻辑，从所述外部指令块对应的内部指令块的起始地址开始对每条外部指令对应的第一条内部指令的条数进行计数；其中，每经过一条所述内部指令，反向移一位，直到移位结果恢复为所述初始值；和

此时对应的内部指令块内地址即对应所述外部指令的块内地址。

21. 如权利要求 20 所述的指令集转换系统，其特征在于，处理器核内的寄存器堆可以被用做栈寄存器；所述系统还包含：

栈顶指针寄存器，用于存储当前栈顶指针，该指针指向寄存器堆中的一个寄存器；

加法器，用于计算栈顶指针加一的值，对应当前栈顶之上的寄存器的位置；

减法器，用于计算栈顶指针减一的值，对应当前栈顶寄存器之下的寄存器的位置；

栈底控制模块，用于检测栈寄存器是否即将为空或即将为满，并在栈寄存器即将为满时将栈底位置的至少一个寄存器的值送往存储器保存，并相应调整栈底指针，使得栈寄存器不会溢出；或

在栈寄存器即将为空时，相应调整栈底指针，并将之前送到存储器保存的至少一个寄存器的值存回栈底，使得栈寄存器能继续提供操作数供处理器核执行。

22. 如权利要求 1 所述的缓存方法，其特征在于，对填充到一级缓存的指令进行审查，提取出相应的指令信息；第一读指针根据所述指令信息而非指令本身的功能确定如何更新。

23. 如权利要求 1 所述的缓存方法，其特征在于，当第一读指针指向一条有条件分支指令，且其后一条是无条件分支指令时，则根据处理器核对有条件分支指令的执行结果：

若分支转移发生，第一读指针被更新为所述有条件分支指令的分支目标寻址地址值；若分支转移没有发生，第一读指针被更新为所述无条件分支指令的分支目标寻址地址值；

使得处理器核不需要单独一个时钟周期执行所述无条件分支指令。

24. 如权利要求 1 所述的指令集转换方法，其特征在于，当处理器核执行到分支指令时，根据分支预测选择顺序执行下一指令和分支目标指令中的一个作为后续指令执行，并保存另一个的寻址地址；

若分支转移结果与分支预测一致，则继续执行后续指令；

若分支转移结果与分支预测不一致，则清空流水线，并从所述保存的寻址地址对应的指令重新执行。

25. 如权利要求 19 所述的指令集转换系统，其特征在于，第一读指针根据所述指令信息而非指令本身的功能确定如何更新。

26. 如权利要求 19 所述的指令集转换系统，其特征在于，同时从轨道表中读出第一读指针指向的轨迹点及其后一个轨迹点中存储的所述指令信息。

27. 如权利要求 26 所述的指令集转换方法，其特征在于，当第一读指针指向一条有条件分支指令，且其后一条是无条件分支指令时，则根据处理器核对有条件分支指令的执行结果：

若分支转移发生，第一读指针被更新为所述有条件分支指令的分支目标寻址地址值；若分支转移没有发生，第一读指针被更新为所述无条件分支指令的分支目标寻址地址值；

使得处理器核不需要单独一个时钟周期执行所述无条件分支指令。

28. 如权利要求 19 所述的指令集转换系统，其特征在于，所述循迹系统还包括一个寄存器，用于存储顺序执行下一指令和分支目标指令中的一个寻址地址；

当处理器核执行到分支指令时，根据分支预测选择顺序执行下一指令和分支目标指令中的一个作为后续指令执行，并将另一个的寻址地址存储在所述寄存器中；

若分支转移结果与分支预测一致，则继续执行后续指令；

若分支转移结果与分支预测不一致，则清空流水线，并从所述寄存器中保存的寻址地址对应的指令重新执行。

29. 如权利要求 19 所述的指令集转换系统，其特征在于，所述轨道表中每条轨道的最后一个轨迹点之后再增加一个结束轨迹点；所述结束轨迹点的指令类型为无条件分支指令，其分支目标寻址地址为顺序执行下一轨道第一个轨迹点的寻址地址；当第一读指针指向结束轨迹点时，一级缓存输出空指令。

30. 如权利要求 29 所述的指令集转换系统，其特征在于，所述轨道表中每条轨道的最后一个轨迹点之后再增加一个结束轨迹点；所述结束轨迹点的指令类型为无条件分支指令，其分支目标寻址地址为顺序执行下一轨道第一个轨迹点的寻址地址；且

当结束轨迹点之前的轨迹点不是分支点时，可以将该结束轨迹点的指令类型及分支目标寻址地址作为该轨迹点的指令类型及分支目标寻址地址。

31. 一种能执行一种或多种指令集的处理器系统，其特征在于包括：

一个第一存储器，用于存储属于第一指令集的复数条计算机指令；

一个指令转换器，用于将所述属于第一指令集的复数条计算机指令转换为复数条内部指令，所述内部指令属于一种第二指令集；

一个第二存储器，用于存储由指令转换器转换得到的所述复数条内部指令；和

一个连接所述第二存储器的处理器核，用于在不需要访问所述复数条计算机指令、以及不需要指令转换器参与的情况下，从第二存储器中读取并执行所述复数条内部指令。

32. 如权利要求 31 所述的系统，其特征在于：

指令转换器包含一个存储器，所述存储器可以根据配置用于存储第一指令集和第二指令集之间的映射关系；和

指令转换器根据存储在其中的第一指令集和第二指令集之间的映射关系将属于第一指令集的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

33. 如权利要求 31 或 32 所述的系统，其特征在于进一步包括 ：

一个连接指令转换器和处理器核的地址转换器，用于将所述复数条计算机指令中的目标计算机指令地址转换为所述复数条内部指令中的目标指令的内部地址。

34. 如权利要求 33 所述的系统，其特征在于在地址转换器转换地址时 ：

将所述目标计算机指令地址映射为内部指令块地址 ；

将所述目标计算机指令地址映射为内部指令在所述块地址对应的指令块中的块内偏移地址 ；和

合并所述块地址和块内偏移地址，构成内部地址。

35. 如权利要求 34 所述的系统，其特征在于 ：

根据所述计算机指令块地址和所述内部指令块地址之间的块地址映射关系映射产生所述块地址。

36. 如权利要求 35 所述的系统，其特征在于 ：由地址转换器存储所述块地址映射关系。

37. 如权利要求 35 所述的系统，其特征在于 ：由硬件逻辑根据一个映射关系表映射产生所述块内偏移地址。

38. 如权利要求 34 所述的系统，其特征在于进一步包括 ：

一个结束标志存储器，用于存储内部指令块的结束指令的内部指令地址 ；所述结束指令就是转移到顺序地址的下一内部指令块前的最后一条内部指令。

39. 如权利要求 34 所述的系统，其特征在于进一步包括 ：一个下块地址存储器，用于存储顺序地址下一内部指令块的块地址。

40. 如权利要求 34 所述的系统，其特征在于进一步包括 ：一个分支目标缓冲，用于存储了分支目标的内部指令地址。

41. 如权利要求 32 所述的系统，其特征在于 ：

所述第一存储器存储了属于一个第三指令集的复数条计算机指令 ；

指令转换器根据配置在所述存储器中存储了第三指令集和第二指令集之间的映射关系 ；和

指令转换器根据存储在其中的第三指令集和第二指令集之间的映射关系将属于第三指令集的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

42. 如权利要求 41 所述的系统，其特征在于在所述系统上运行一个第一线程指令序列和一个第二线程指令序列 ；其中 ：

第一线程指令序列由第一指令集的复数条计算机指令构成 ；

第二线程指令序列由第三指令集的复数条计算机指令构成 ；

所述指令转换器根据配置在所述存储器中同时存储了第一指令集和第二指令集之间的映射关系，以及第三指令集和第二指令集之间的映射关系 ；和

指令转换器根据线程号选择所述第一指令集和第二指令集之间的映射关系及第三指令集和第二指令集之间的映射关系中的一个，将该线程的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

43. 如权利要求 32 所述的系统，其特征在于 ：

所述复数条计算机指令中的每一条均至少包含一个内容为指令类型的指令域；

所述复数条内部指令中的每一条均至少包含一个内容为指令类型的指令域；

所述复数条计算机指令和所述复数条内部指令一一对应；和

所述映射关系包括每条计算机指令的指令类型和每条内部指令的指令类型之间的映射关系，以及每条计算机指令中除指令类型之外的指令域与每条内部指令中除指令类型之外的指令域之间的映射关系。

44. 如权利要求 32 所述的系统，其特征在于：

所述复数条计算机指令中的每一条均至少包含一个内容为指令类型的指令域；

所述复数条内部指令中的每一条均至少包含一个内容为指令类型的指令域；

所述复数条计算机指令和所述复数条内部指令的总数不相等；和

所述复数条计算机指令中的每一条均被映射为所述复数条内部指令中的一条或多条。

45. 如权利要求 43 或 44 所述的系统，其特征在于：

所述计算机指令的指令域中至少包含一个指令类型；和

指令转换器至少利用所述指令类型对指令转换器中的存储器寻址读出相应的映射关系。

46. 如权利要求 45 所述的系统，其特征在于：

所述映射关系包含一个移位逻辑；和

所述复数条内部指令中至少一条指令的一个指令域通过对相应计算机指令的相应指令域移位产生。

47. 一种用于执行一种或多种指令集的处理器系统的方法，其特征在于包括：

将属于第一指令集的复数条计算机指令存储在一个第一存储器中；

由一个指令转换器将所述复数条计算机指令转换为属于一个第二指令集的复数条内部指令；

将由指令转换器转换得到的所述复数条内部指令存储在一个第二存储器中；和

由一个连接所述第二存储器的处理器核在不需要访问所述复数条计算机指令、以及不需要指令转换器参与的情况下，从第二存储器中读取并执行所述复数条内部指令。

48. 如权利要求 47 所述的方法，其特征在于：

通过将第一指令集和第二指令集映射关系存储到指令转换器的存储器中，对指令转换器进行配置；和

指令转换器根据存储在其中的第一指令集和第二指令集之间的映射关系将属于第一指令集的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

49. 如权利要求 47 或 48 所述的方法，其特征在于：

通过一个连接指令转换器和处理器核的地址转换器将所述复数条计算机指令中的目标计算机指令地址转换为所述复数条内部指令中的目标指令的内部地址。

50. 如权利要求 49 所述的方法，其特征在于在地址转换器转换地址时：

将所述目标计算机指令地址映射为内部指令块地址；

将所述目标计算机指令地址映射为内部指令在所述块地址对应的指令块中的块内偏移地址；和

合并所述块地址和块内偏移地址，构成内部地址。

51. 如权利要求 50 所述的方法，其特征在于 ：

根据所述计算机指令块地址和所述内部指令块地址之间的块地址映射关系映射产生所述块地址。

52. 如权利要求 51 所述的方法，其特征在于 ：由地址转换器存储所述块地址映射关系。

53. 如权利要求 51 所述的方法，其特征在于 ：由硬件逻辑根据一个映射关系表映射产生所述块内偏移地址。

54. 如权利要求 50 所述的方法，其特征在于进一步包括 ：

由一个结束标志存储器存储内部指令块的结束指令的内部指令地址 ；所述结束指令就是转移到顺序地址的下一内部指令块前的最后一条内部指令。

55. 如权利要求 50 所述的方法，其特征在于进一步包括 ：由一个下块地址存储器存储顺序地址下一内部指令块的块地址。

56. 如权利要求 50 所述的方法，其特征在于进一步包括 ：由一个分支目标缓冲存储了分支目标的内部指令地址。

57. 如权利要求 48 所述的方法，其特征在于 ：

将属于一个第三指令集的复数条计算机指令存储在所述第一存储器中 ；

由指令转换器根据配置在所述存储器中存储了第三指令集和第二指令集之间的映射关系 ；和

由指令转换器根据存储在其中的第三指令集和第二指令集之间的映射关系将属于第三指令集的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

58. 如权利要求 57 所述的方法，其特征在于运行一个第一线程指令序列和一个第二线程指令序列 ；其中 ：

第一线程指令序列由第一指令集的复数条计算机指令构成 ；

第二线程指令序列由第三指令集的复数条计算机指令构成 ；

由所述指令转换器根据配置在所述存储器中同时存储第一指令集和第二指令集之间的映射关系，以及第三指令集和第二指令集之间的映射关系 ；和

由指令转换器根据线程号选择所述第一指令集和第二指令集之间的映射关系及第三指令集和第二指令集之间的映射关系中的一个，将该线程的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

59. 如权利要求 48 所述的方法，其特征在于 ：

所述复数条计算机指令中的每一条均至少包含一个内容为指令类型的指令域 ；

所述复数条内部指令中的每一条均至少包含一个内容为指令类型的指令域 ；

所述复数条计算机指令和所述复数条内部指令一一对应 ；和

所述映射关系包括每条计算机指令的指令类型和每条内部指令的指令类型之间的映射关系，以及每条计算机指令中除指令类型之外的指令域与每条内部指令中除指令类型之外的指令域之间的映射关系。

60. 如权利要求 48 所述的方法，其特征在于 ：

所述复数条计算机指令中的每一条均至少包含一个内容为指令类型的指令域 ；

所述复数条内部指令中的每一条均至少包含一个内容为指令类型的指令域 ；

所述复数条计算机指令和所述复数条内部指令的总数不相等 ；和

所述复数条计算机指令中的每一条均被映射为所述复数条内部指令中的一条或多条。

61. 如权利要求 59 或 60 所述的系统，其特征在于：

所述计算机指令的指令域中至少包含一个指令类型；和

指令转换器至少利用所述指令类型对指令转换器中的存储器寻址读出相应的映射关系。

62. 如权利要求 61 所述的方法，其特征在于：

所述复数条内部指令中至少一条指令的一个指令域通过对相应计算机指令的相应指令域移位产生。

一种指令集转换系统和方法

技术领域

[0001] 本发明涉及计算机，通讯及集成电路领域。

背景技术

[0002] 目前，如果需要在某个处理器核上执行属于不同指令集的程序，最常用的方法是使用软件虚拟机（或虚拟层）。虚拟机的作用是对由处理器核不支持的指令集（外部指令集）组成的程序进行翻译或解释，生成处理器核本身支持的指令集（内部指令集）对应的指令后供执行。一般地，采用解释的方法，是在运行过程中实时地由虚拟机通过软件方法将外部指令中的包括操作码、操作数等各个域依次取出，然后用存储器中实现的栈结构，根据不同操作码对操作数进行相应操作。因此，需要执行很多条内部指令才能实现任意一条外部指令的功能，效率很低。而采用翻译的方法，在程序执行前先执行类似软件编译的过程，将该程序转换成完全由内部指令集组成的形式。这样在执行程序时，效率比较高，但软件编译本身依然有不小的开销。

[0003] 第二种解决方法是在处理器核内部包含对应不同指令集的指令译码器，在执行不同指令集的指令时用相应的指令译码进行译码并交后续流水线操作。这种方法在执行效率上几乎没有损失，但增加的指令译码器会导致硬件开销增大，提高了处理器芯片的成本。此外，由于多种指令译码器是事先用硬件在处理器核内实现的，缺乏扩展性，无法支持新的指令集。

[0004] 第三种解决方法是在处理器核的外部增加一个转换模块，将外部指令集转换为内部指令集后供处理器核执行。这种转换模块可以是用软件实现的，但一般来说，用软件进行解释虽然易于扩展，但效率太低。这种转换模块也可以是用硬件实现的，但难于扩展，且无法充分利用缓存存储转换得到的内部指令。

[0005] 具体地，若该转换模块位于缓存和处理器核之间，则缓存中存储的是外部指令，必须经过转换才能供处理器核执行。这样，无论是否缓存命中，都要经过该转换步骤，对同样的外部指令进行多次重复性的转换，不但增加了功耗，而且加深了处理器核的流水线，从而增加了硬件开销和分支预测失败时的性能损失。

[0006] 若该转换模块位于缓存之外（即缓存位于转换模块和处理器核之间），则缓存中存储的是转换得到的内部指令，即根据内部指令地址对缓存寻址，而处理器核执行分支指令计算得到的分支目标指令地址是外部指令地址。由于内部指令和外部指令并不是一一对应的（如：一条外部指令可以对应多条内部指令），因此必须记录内部指令地址与外部指令地址对应关系，才能在分支转移时，将分支目标指令的外部指令地址转换为内部指令地址并以此在缓存中找到正确的指令。记录内部指令地址与外部指令地址对应关系的难点在于如何有效地储存，以及如何有效地转换。否则，一旦发生分支转移，只能根据外部指令地址从转换模块以外的更低层次存储器中读取指令经转换模块转换后再存储到缓存中并供处理器核执行，依然严重影响执行效率。该问题的一种方法是用基于程序执行路径跟踪缓存(trace cache)代替传统的基于地址匹配的缓存。但跟踪缓存中会存储大量地址重复、但位

于不同路径上的指令，造成很大的容量浪费，导致跟踪缓存的性能不高。

[0007] 本发明提出的方法与系统装置能直接解决上述或其他的一个或多个困难。

发明内容

[0008] 本发明提出了一种指令集转换方法，所述方法包括：将外部指令转换为内部指令，并建立外部指令地址和内部指令地址之间的映射关系；将所述内部指令存储在处理器核能直接访问的缓存中；直接根据该内部指令地址对缓存寻址读出相应的内部指令供处理器核执行；或根据所述映射关系将处理器核输出的外部指令地址转换为内部指令地址后，对缓存寻址读出相应的内部指令供处理器核执行。

[0009] 可选的，在所述方法中，根据程序执行流及处理器核执行指令的反馈向处理器核提供后续指令；所述处理器核执行指令的反馈可以是处理器核执行分支指令时产生的分支转移是否发生的信号。

[0010] 可选的，在所述方法中，对于需要被转换的外部指令提取出外部指令中包含指令类型在内的各个指令域；根据提取出的指令类型查找对应的内部指令的指令类型和指令转换控制信息；根据所述指令转换控制信息对提取出的相应指令域进行移位；对所述内部指令类型及移位后的指令域进行拼接，构成相应的内部指令，从而将外部指令转换为内部指令。

[0011] 可选的，在所述方法中，一条外部指令被转换为一条内部指令；其中，该外部指令的指令地址对应内部指令的指令地址；或一条外部指令被转换为多条内部指令；其中，该外部指令的指令地址对应所述多条内部指令中第一条内部指令的指令地址。

[0012] 可选的，在所述方法中，多条外部指令被转换为一条内部指令；其中，所述多条外部指令中第一条外部指令的指令地址对应该内部指令的指令地址。

[0013] 可选的，在所述方法中，建立外部指令地址和内部指令地址之间的映射关系。

[0014] 可选的，在所述方法中，所述外部指令地址和内部指令地址之间的映射关系包括：外部指令地址和内部指令块地址之间的映射关系、外部指令块内地址和内部指令块内地址之间的映射关系。

[0015] 可选的，在所述方法中，可以用一种数据结构表示外部指令地址和内部指令块地址之间的映射关系；所述数据结构中存储了内部指令块地址，且所述内部指令块地址同时按外部指令块地址和外部指令块内地址进行排序。

[0016] 可选的，在所述数据结构中，如果一个外部指令地址对应的内部指令块地址存在，则可以根据所述外部指令地址中的外部指令块地址和外部指令块内地址，在该数据结构中找到对应的位置，读出其中存储的内部指令块地址。

[0017] 可选的，在所述数据结构中，如果一个外部指令地址对应的内部指令块地址不存在，则可以根据所述外部指令地址中的外部指令块地址和外部指令块内地址，找到其插入位置，并在位置中存储该外部指令地址对应的内部指令块地址。

[0018] 可选的，在所述方法中，根据所述外部指令块地址和内部指令块地址之间的映射关系，可以对外部指令地址进行转换得到对应的内部指令块地址。

[0019] 可选的，在所述方法中，根据所述外部指令块内地址和内部指令块内地址之间的映射关系，可以对外部指令块内地址进行转换得到对应的内部指令块内地址。

[0020] 可选的，在所述方法中，对于任意一个外部指令地址，通过正向移位逻辑，从初始值开始，对从该地址所在的外部指令块起始地址开始至该外部指令地址之间的外部指令条数进行计数；其中，每经过一条所述外部指令，正向移一位，最终得到一个移位结果；通过反向移位逻辑，从所述外部指令块对应的内部指令块的起始地址开始对每条外部指令对应的第一条内部指令的条数进行计数；其中，每经过一条所述内部指令，反向移一位，直到移位结果恢复为所述初始值；此时对应的内部指令块内地址即对应所述外部指令的块内地址。

[0021] 可选的，在所述方法中，通过地址计算，将栈寄存器操作转换为对寄存器堆的操作，使得处理器核内部的寄存器堆能作为栈寄存器使用。

[0022] 可选的，在所述方法中，所述转换能将一种或多种指令集的指令转换为一种指令集的指令。

[0023] 本发明还提出了一种指令集转换系统，所述系统包括：处理器核，用于执行内部指令；转换器，用于将外部指令转换为内部指令，并建立外部指令地址和内部指令地址之间的映射关系；地址映射模块，用于存储所述外部指令地址和内部指令地址之间的映射关系，并对外部指令地址和内部指令地址之间进行转换；缓存，用于存储转换得到的内部指令，并根据内部指令地址输出相应内部供处理器核执行。

[0024] 可选的，在所述系统中，所述转换器进一步包括：存储器，用于存储外部指令类型与内部指令类型的对应关系，及相应外部指令和内部指令之间各个指令域的对应关系；对齐器，用于将外部指令移位对齐，并在外部指令跨越指令块边界的情况下，将该外部指令移位到一个指令块并对齐；提取器，用于提取出外部指令中的各个指令域；其中，提取出的指令类型被用于对所述存储器寻址，以读出所述外部指令对应的指令转换控制信息及相应的内部指令类型，并根据所述控制信息对提取出的指令域进行移位；指令拼接器，用于对所述内部指令类型和移位后的指令域进行拼接，构成内部指令。

[0025] 可选的，在所述系统中，所述地址映射模块进一步包括：块地址映射模块，用于存储外部指令块地址与内部指令块地址之间的映射关系，并将外部指令块地址转换为内部指令块地址；偏移地址映射模块，用于存储外部指令块内地址与内部指令块内地址之间的映射关系，并将外部指令块内地址转换为内部指令块内地址。

[0026] 可选的，所述系统还包括一个循迹系统；所述循迹系统根据存储在其中的程序执行流及处理器核执行指令的反馈，同时对所述程序执行流及缓存寻址，并从缓存中读出后续指令送往处理器核供执行；所述处理器核执行指令的反馈可以是处理器核执行分支指令时产生的分支转移是否发生的信号。

[0027] 可选的，在所述系统中，地址映射模块中还包含一个正向移位逻辑和一个反向移位逻辑；对于任意一个外部指令地址，通过正向移位逻辑，从初始值开始，对从该地址所在的外部指令块起始地址开始至该外部指令地址之间的外部指令条数进行计数；其中，每经过一条所述外部指令，正向移一位，最终得到一个移位结果；通过反向移位逻辑，从所述外部指令块对应的内部指令块的起始地址开始对每条外部指令对应的第一条内部指令的条数进行计数；其中，每经过一条所述内部指令，反向移一位，直到移位结果恢复为所述初始值；此时对应的内部指令块内地址即对应所述外部指令的块内地址。

[0028] 可选的，在所述系统中，处理器核内的寄存器堆可以被用做栈寄存器；所述系统还

包含：栈顶指针寄存器，用于存储当前栈顶指针，该指针指向寄存器堆中的一个寄存器；加法器，用于计算栈顶指针加一的值，对应当前栈顶之上的寄存器的位置；减法器，用于计算栈顶指针减一的值，对应当前栈顶寄存器之下的寄存器的位置；栈底控制模块，用于检测栈寄存器是否即将为空或即将为满，并在栈寄存器即将为满时将栈底位置的至少一个寄存器的值送往存储器保存，并相应调整栈底指针，使得栈寄存器不会溢出；或在栈寄存器即将为空时，相应调整栈底指针，并将之前送到存储器保存的至少一个寄存器的值存回栈底，使得栈寄存器能继续提供操作数供处理器核执行。

[0029] 可选的，在所述方法中，对填充到一级缓存的指令进行审查，提取出相应的指令信息；第一读指针根据所述指令信息而非指令本身的功能确定如何更新。

[0030] 可选的，在所述方法中，当第一读指针指向一条有条件分支指令，且其后一条是无条件分支指令时，则根据处理器核对有条件分支指令的执行结果：若分支转移发生，第一读指针被更新为所述有条件分支指令的分支目标寻址地址值；若分支转移没有发生，第一读指针被更新为所述无条件分支指令的分支目标寻址地址值；使得处理器核不需要单独一个时钟周期执行所述无条件分支指令。

[0031] 可选的，在所述方法中，当处理器核执行到分支指令时，根据分支预测选择顺序执行下一指令和分支目标指令中的一个作为后续指令执行，并保存另一个的寻址地址；若分支转移结果与分支预测一致，则继续执行后续指令；若分支转移结果与分支预测不一致，则清空流水线，并从所述保存的寻址地址对应的指令重新执行。

[0032] 可选的，在所述系统中，第一读指针根据所述指令信息而非指令本身的功能确定如何更新。

[0033] 可选的，在所述系统中，同时从轨道表中读出第一读指针指向的轨迹点及其后一个轨迹点中存储的所述指令信息。

[0034] 可选的，在所述系统中，当第一读指针指向一条有条件分支指令，且其后一条是无条件分支指令时，则根据处理器核对有条件分支指令的执行结果：若分支转移发生，第一读指针被更新为所述有条件分支指令的分支目标寻址地址值；若分支转移没有发生，第一读指针被更新为所述无条件分支指令的分支目标寻址地址值；使得处理器核不需要单独一个时钟周期执行所述无条件分支指令。

[0035] 可选的，在所述系统中，所述循迹系统还包括一个寄存器，用于存储顺序执行下一指令和分支目标指令中的一个寻址地址；当处理器核执行到分支指令时，根据分支预测选择顺序执行下一指令和分支目标指令中的一个作为后续指令执行，并将另一个的寻址地址存储在所述寄存器中；若分支转移结果与分支预测一致，则继续执行后续指令；若分支转移结果与分支预测不一致，则清空流水线，并从所述寄存器中保存的寻址地址对应的指令重新执行。

[0036] 可选的，在所述系统中，所述轨道表中每条轨道的最后一个轨迹点之后再增加一个结束轨迹点；所述结束轨迹点的指令类型为无条件分支指令，其分支目标寻址地址为顺序执行下一轨道第一个轨迹点的寻址地址；当第一读指针指向结束轨迹点时，一级缓存输出空指令。

[0037] 可选的，在所述系统中，所述轨道表中每条轨道的最后一个轨迹点之后再增加一个结束轨迹点；所述结束轨迹点的指令类型为无条件分支指令，其分支目标寻址地址为顺

序执行下一轨道第一个轨迹点的寻址地址；当结束轨迹点之前的轨迹点不是分支点时，可以将该结束轨迹点的指令类型及分支目标寻址地址作为该轨迹点的指令类型及分支目标寻址地址。

[0038] 本发明还提出了一种能执行一种或多种指令集的处理器系统，包括：一个第一存储器，用于存储属于第一指令集的复数条计算机指令；一个指令转换器，用于将所述属于第一指令集的复数条计算机指令转换为复数条内部指令，所述内部指令属于一种第二指令集；一个第二存储器，用于存储由指令转换器转换得到的所述复数条内部指令；一个连接所述第二存储器的处理器核，用于在不需要访问所述复数条计算机指令、以及不需要指令转换器参与的情况下，从第二存储器中读取并执行所述复数条内部指令。

[0039] 可选的，在所述系统中，指令转换器包含一个存储器，所述存储器可以根据配置用于存储第一指令集和第二指令集之间的映射关系；指令转换器根据存储在其中的第一指令集和第二指令集之间的映射关系将属于第一指令集的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

[0040] 可选的，所述系统进一步包括：一个连接指令转换器和处理器核的地址转换器，用于将所述复数条计算机指令中的目标计算机指令地址转换为所述复数条内部指令中的目标指令的内部地址。

[0041] 可选的，在所述系统中，在地址转换器转换地址时：将所述目标计算机指令地址映射为内部指令块地址；将所述目标计算机指令地址映射为内部指令在所述块地址对应的指令块中的块内偏移地址；合并所述块地址和块内偏移地址，构成内部地址。

[0042] 可选的，在所述系统中，根据所述计算机指令块地址和所述内部指令块地址之间的块地址映射关系映射产生所述块地址。

[0043] 可选的，在所述系统中，由地址转换器存储所述块地址映射关系；由硬件逻辑根据一个映射关系表映射产生所述块内偏移地址。

[0044] 可选的，所述系统进一步包括：一个结束标志存储器，用于存储内部指令块的结束指令的内部指令地址；所述结束指令就是转移到顺序地址的下一内部指令块前的最后一条内部指令。

[0045] 可选的，所述系统进一步包括：一个下块地址存储器，用于存储顺序地址下一内部指令块的块地址；一个分支目标缓冲，用于存储了分支目标的内部指令地址。

[0046] 可选的，在所述系统中，所述第一存储器存储了属于一个第三指令集的复数条计算机指令；指令转换器根据配置在所述存储器中存储了第三指令集和第二指令集之间的映射关系；指令转换器根据存储在其中的第三指令集和第二指令集之间的映射关系将属于第三指令集的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

[0047] 可选的，在所述系统上运行一个第一线程指令序列和一个第二线程指令序列；其中：第一线程指令序列由第一指令集的复数条计算机指令构成；第二线程指令序列由第三指令集的复数条计算机指令构成；所述指令转换器根据配置在所述存储器中同时存储了第一指令集和第二指令集之间的映射关系，以及第三指令集和第二指令集之间的映射关系；指令转换器根据线程号选择所述第一指令集和第二指令集之间的映射关系及第三指令集和第二指令集之间的映射关系中的一个，将该线程的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

[0048] 可选的，在所述系统中，所述复数条计算机指令中的每一条均至少包含一个内容为指令类型的指令域；所述复数条内部指令中的每一条均至少包含一个内容为指令类型的指令域；所述复数条计算机指令和所述复数条内部指令一一对应；所述映射关系包括每条计算机指令的指令类型和每条内部指令的指令类型之间的映射关系，以及每条计算机指令中除指令类型之外的指令域与每条内部指令中除指令类型之外的指令域之间的映射关系。

[0049] 可选的，在所述系统中，所述复数条计算机指令中的每一条均至少包含一个内容为指令类型的指令域；所述复数条内部指令中的每一条均至少包含一个内容为指令类型的指令域；所述复数条计算机指令和所述复数条内部指令的总数不相等；所述复数条计算机指令中的每一条均被映射为所述复数条内部指令中的一条或多条。

[0050] 可选的，在所述系统中，所述映射关系包含一个移位逻辑；所述复数条内部指令中至少一条指令的一个指令域通过对相应计算机指令的相应指令域移位产生。

[0051] 可选的，在所述系统中，所述计算机指令的指令域中至少包含一个指令类型；指令转换器至少利用所述指令类型对指令转换器中的存储器寻址读出相应的映射关系。

[0052] 本发明还提出一种用于执行一种或多种指令集的处理器系统的方法，所述方法包括：将属于第一指令集的复数条计算机指令存储在一个第一存储器中；由一个指令转换器将所述复数条计算机指令转换为属于一个第二指令集的复数条内部指令；将由指令转换器转换得到的所述复数条内部指令存储在一个第二存储器中；由一个连接所述第二存储器的处理器核在不需要访问所述复数条计算机指令、以及不需要指令转换器参与的情况下，从第二存储器中读取并执行所述复数条内部指令。

[0053] 可选的，在所述方法中，通过将第一指令集和第二指令集映射关系存储到指令转换器的存储器中，对指令转换器进行配置；指令转换器根据存储在其中的第一指令集和第二指令集之间的映射关系将属于第一指令集的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

[0054] 可选的，在所述方法中，通过一个连接指令转换器和处理器核的地址转换器将所述复数条计算机指令中的目标计算机指令地址转换为所述复数条内部指令中的目标指令的内部地址。

[0055] 可选的，在所述方法中，在地址转换器转换地址时：将所述目标计算机指令地址映射为内部指令块地址；将所述目标计算机指令地址映射为内部指令在所述块地址对应的指令块中的块内偏移地址；合并所述块地址和块内偏移地址，构成内部地址。

[0056] 可选的，在所述方法中，根据所述计算机指令块地址和所述内部指令块地址之间的块地址映射关系映射产生所述块地址。

[0057] 可选的，在所述方法中，由地址转换器存储所述块地址映射关系；由硬件逻辑根据一个映射关系表映射产生所述块内偏移地址。

[0058] 可选的，所述方法进一步包括：由一个结束标志存储器存储内部指令块的结束指令的内部指令地址；所述结束指令就是转移到顺序地址的下一内部指令块前的最后一条内部指令。

[0059] 可选的，所述方法进一步包括：由一个下块地址存储器存储顺序地址下一内部指令块的块地址；由一个分支目标缓冲存储了分支目标的内部指令地址。

[0060] 可选的，在所述方法中，将属于一个第三指令集的复数条计算机指令存储在所述

第一存储器中；由指令转换器根据配置在所述存储器中存储了第三指令集和第二指令集之间的映射关系；由指令转换器根据存储在其中的第三指令集和第二指令集之间的映射关系将属于第三指令集的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

[0061] 可选的，在所述方法中，运行一个第一线程指令序列和一个第二线程指令序列；其中：第一线程指令序列由第一指令集的复数条计算机指令构成；第二线程指令序列由第三指令集的复数条计算机指令构成；由所述指令转换器根据配置在所述存储器中同时存储第一指令集和第二指令集之间的映射关系，以及第三指令集和第二指令集之间的映射关系；由指令转换器根据线程号选择所述第一指令集和第二指令集之间的映射关系及第三指令集和第二指令集之间的映射关系中的一个，将该线程的所述复数条计算机指令转换为属于第二指令集的所述复数条内部指令。

[0062] 可选的，在所述方法中，所述复数条计算机指令中的每一条均至少包含一个内容为指令类型的指令域；所述复数条内部指令中的每一条均至少包含一个内容为指令类型的指令域；所述复数条计算机指令和所述复数条内部指令一一对应；所述映射关系包括每条计算机指令的指令类型和每条内部指令的指令类型之间的映射关系，以及每条计算机指令中除指令类型之外的指令域与每条内部指令中除指令类型之外的指令域之间的映射关系。

[0063] 可选的，在所述方法中，所述复数条计算机指令中的每一条均至少包含一个内容为指令类型的指令域；所述复数条内部指令中的每一条均至少包含一个内容为指令类型的指令域；所述复数条计算机指令和所述复数条内部指令的总数不相等；所述复数条计算机指令中的每一条均被映射为所述复数条内部指令中的一条或多条。

[0064] 可选的，在所述方法中，所述复数条内部指令中至少一条指令的一个指令域通过对相应计算机指令的相应指令域移位产生。

[0065] 可选的，在所述方法中，所述计算机指令的指令域中至少包含一个指令类型；指令转换器至少利用所述指令类型对指令转换器中的存储器寻址读出相应的映射关系。

[0066] 对于本领域专业人士，还可以在本发明的说明、权利要求和附图的启发下，理解、领会本发明所包含其他方面内容。

[0067] 有益效果

[0068] 本发明所述处理器系统中最接近处理器核的缓存系统（即较高层次缓存）中存储的是处理器核本身支持的内部指令集，而存储在主存储器或较低层次缓存中的是外部指令集。通过对转换器配置，可以使得相应的外部指令集被转换为内部指令集供处理器核执行。因此，能很方便地扩展处理器系统支持的指令集。

[0069] 本发明根据程序执行流及处理器核执行指令的反馈，直接由较高层次缓存向处理器核提供内部指令，降低了流水线深度，提高了流水线效率。特别在分支预测错误时，能减少浪费的流水线周期。

[0070] 对于本领域专业人士而言，本发明的其他优点和应用是显见的。

附图说明

[0071] 图 1 是本发明所述处理器系统的一个示意图；

[0072] 图 2 是本发明所述转换器的一个实施例；

[0073] 图 3A 是本发明所述对齐器的一个实施例；

[0074] 图 3B 是本发明所述对齐器运行过程的一个实施例；

[0075] 图 4A 是本发明所述提取器的一个实施例；

[0076] 图 4B 是本发明所述提取器运行过程的一个实施例；

[0077] 图 5A 是本发明所述映射信息的一个示意图；

[0078] 图 5B 是本发明所述映射信息的另一个示意图；

[0079] 图 5C 是本发明所述映射信息存储器运行的一个实施例；

[0080] 图 5D 是本发明所述映射信息存储器运行的另一个实施例；

[0081] 图 5E 是本发明所述映射信息存储器运行的另一个实施例；

[0082] 图 5F 是本发明所述指令拼接器的一个实施例；

[0083] 图 6 是本发明所述包含多层缓存的处理器系统的一个实施例；

[0084] 图 7A 是本发明所述基于轨道表的缓存结构的实施例；

[0085] 图 7B 是本发明所述扫描转换器的一个实施例；

[0086] 图 8A 是本发明所述外部指令块与内部指令块对应关系的示意图；

[0087] 图 8B 是本发明所述偏移地址映射关系存储形式的一个实施例；

[0088] 图 8C 是本发明所述偏移地址转换器的一个实施例；

[0089] 图 8D 是本发明所述块地址映射模块的一个实施例；

[0090] 图 9A ～ 9F 是本发明所述包含多层缓存的处理器系统运行过程的示意图；

[0091] 图 10A 是本发明所述操作数栈的一个实施例；

[0092] 图 10B 是本发明所述更新栈底的一个实施例；

[0093] 图 10C 是本发明所述更新栈底的另一个实施例；

[0094] 图 11A 是本发明所述基于轨道表的缓存结构的另一个实施例；

[0095] 图 11B 是本发明支持猜测执行的实施例；

[0096] 图 12 是本发明所述包含可配置转换器的处理器系统的一个实施例；

[0097] 图 13A 是本发明所述可配置转换器的一个框图实施例；

[0098] 图 13B 是本发明所述可配置转换器中存储器的一个实施例；

[0099] 图 13C 是本发明所述可配置转换器中存储器的另一个实施例；

[0100] 图 14 是本发明所述包含可配置转换器和地址映射模块的处理器系统的一个实施例；

[0101] 图 15 是本发明所述包含可配置转换器和地址映射模块的处理器系统的另一个实施例；

[0102] 图 16 是本发明所述包含分支目标表的处理器系统的一个实施例；

[0103] 图 17 是本发明所述包含分支目标表和循迹器的处理器系统的另一个实施例；

[0104] 图 18A 是本发明所述下块地址存储器格式的一个实施例；

[0105] 图 18B 是本发明所述下块地址存储器格式的另一个实施例；

[0106] 图 18C 是所述两个存储层次处理器系统中外部指令地址格式的一个示意图；

[0107] 图 19 是本发明所述包含两层指令存储器的处理器系统的一个实施例；

[0108] 图 20 是本发明所述两个存储层次处理器系统中标签存储器结构的一个示意图；

[0109] 图 21 是本发明所述外部指令边界不对齐情况下指令存储器存储内部指令的一个

实施例；

[0110] 图 22 是本发明所述块地址映射模块的另一个实施例；

[0111] 图 23 是本发明所述包含轨道表的处理器系统的一个实施例；

[0112] 图 24 是本发明所述利用寄存器堆实现栈操作功能的处理系统的一个实施例。

具体实施方式

[0113] 以下结合附图和具体实施例对本发明提出的高性能缓存系统和方法作进一步详细说明。根据下面说明和权利要求书，本发明的优点和特征将更清楚。需说明的是，附图均采用非常简化的形式且均使用非精准的比例，仅用以方便、明晰地辅助说明本发明实施例的目的。

[0114] 需要说明的是，为了清楚地说明本发明的内容，本发明特举多个实施例以进一步阐释本发明的不同实现方式，其中，该多个实施例是列举式并非穷举式。此外，为了说明的简洁，前实施例中已提及的内容往往在后实施例中予以省略，因此，后实施例中未提及的内容可相应参考前实施例。

[0115] 虽然该发明可以以多种形式的修改和替换来扩展，说明书中也列出了一些具体的实施图例并进行详细阐述。应当理解的是，发明者的出发点不是将该发明限于所阐述的特定实施例，正相反，发明者的出发点在于保护所有基于由本权利声明定义的精神或范围内进行的改进、等效转换和修改。同样的元器件号码可能被用于所有附图以代表相同的或类似的部分。

[0116] 本发明所述的指令地址 (Instruction Address) 指的是指令在主存储器中的存储地址，即可以根据该地址在主存储器中找到这条指令。在此为简单明了起见，均假设虚拟地址等于物理地址，对于需要进行地址映射的情况，本发明所述方法也可适用。在本发明中，当前指令可以指当前正在被处理器核执行或获取的指令；当前指令块可以指含有当前正被处理器执行的指令的指令块。

[0117] 为便于说明，在本说明书中，术语“外部指令集 (Guest Instrution Set)”代表本发明所述处理器系统执行的程序对应的指令集，“外部指令集”中包含的指令即“外部指令”；术语“内部指令集 (Host Instruction Set)”代表本发明所述处理器系统中处理器核本身支持的指令集，“内部指令集”中包含的指令即“内部指令”；术语“指令块”代表指令地址高位相同的一组连续的指令；术语“指令域”是指令字中的代表同一内容的连续区域 (Field)，如第一操作码 (Op-code) 域、第二操作码域、第一源寄存器 (Source Register) 域、第二源寄存器域、目标寄存器 (Tatget Register) 域、立即数 (immediate) 域等。此外，在本发明中，内部指令集是定长指令集，即每条目标指令的字长是固定的 (如 :32 位)；而外部指令集可以是定长指令集，也可以是变长指令集。如果外部指令集是变长的，且一条变长外部指令所占的所有字节的地址高位不完全相同，即该指令跨越了两个指令块，则将该外部指令作为前一个指令块的最后一条指令，而该指令之后的一条指令作为后一个指令块的第一条指令。

[0118] 在本发明中，分支指令 (Branch Instruction) 或分支点 (Branch Point) 指的是任何适当的能导致处理器核改变执行流 (Execution Flow)(如 :非按顺序执行指令或微操作) 的指令形式。分支指令地址指分支指令本身的指令地址，该地址由指令块地址和指令

偏移地址构成。分支目标指令指的是分支指令造成的分支转移所转向的目标指令，分支目标指令地址指的是分支目标指令的指令地址。

[0119] 根据本发明技术方案，每条外部指令先被转换成单数条或复数条内部指令；或复数条外部指令被转换成单数条或复数条内部指令；再由处理器核执行，从而实现与直接执行所述外部指令相同的功能。请参考图 1，其为本发明所述处理器系统的一个示意图。其中，存储器 103 中存储了需要被执行的程序的可执行代码，且该可执行代码是由外部指令集的指令构成；每条所述外部指令先被送到转换器 200 转换为对应的单数条或复数条内部指令，再被送到处理器核 101 执行。在本发明中转换器 200 可以是固定结构的，即仅支持将特定的外部指令集转换为内部指令集；也可以是可配置的，即可以根据配置，将一种或多种外部指令集转换为内部指令集。在此，可以认为所述固定结构的转换器是可配置的转换器的特例，因此在本说明书，仅对可配置的转换器进行说明。

[0120] 请参考图 2，其为本发明所述转换器的一个实施例。在本实施例中，转换器 200 由存储器 201、对齐器 203、提取阵列 205、指令拼接器 207 和操作码拼接器 209 构成。其中，对齐器 203 将外部指令移位对齐，并在外部指令跨越指令块边界的情况下，将该外部指令移位到一个指令块并对齐。

[0121] 请参考图 3A，其为本发明所述对齐器的一个实施例。在本实施例中，对齐器 203 由控制器 301、缓冲 303、305 和循环移位器 307 构成。在此，假设一条外部指令的字长的单位为字节，且一个指令块可以容纳最长的外部指令的全部字节。因此，本实施例采用两个缓冲分别存储两个连续的指令块。这样，正被处理的一条外部指令可以完全位于缓冲 303 中一个指令块中；或跨越了指令块边界（即该指令的头在一个缓冲 303 中指令块的尾部，而剩余部分在缓冲 305 中指令块头部）。选择器 312、314、316、318 和 320 按顺序由左到右各对应一个字节，在译码器 327 控制下对其相应字节选择缓冲 303 或 305 中内容送往循环移位器 307 的输入。

[0122] 控制器 301 中有寄存器 321 及加法器 323，其位数为 m，而 2^m 等于存储器 303，305 的字节宽度。寄存器 321 存储了当前被转换的外部指令的起始偏移地址 (SA，Start Address)。该 SA 经编码器 327 编码后作为选择信号控制缓冲 303 及 305 的输出选择器 312、314、316、318 和 320，相应从缓冲 303 中选出偏移地址大于等于 SA 的字节及从缓冲 305 中选出偏移地址小于 SA 的字节，一同送往循环移位器 307。经总线 313 被送到循环移位器 307 作为移位位数 (Shift Amount)。

[0123] 这样，循环移位器 307 的输入中，偏移地址大于等于该 SA 的部分就是所述外部指令的头部 353，而偏移地址小于该 SA 的部分就是所述外部指令的尾部 355，且可能在所述尾部之后还有后续外部指令的一部分内容。因此，循环移位器 307 根据从总线 313 接收到的移位位数（即 SA）进行循环左移，即可将所述外部指令的头部 353 移至指令块的起始位置，将该外部指令的尾部 355 置于同一个指令块中头部右面的位置，并将该指令块从循环移位器 307 输出。

[0124] 该指令块中外部指令经检测后其外部指令长度会由存储器 201 经总线 325 送出。该长度经总线 325 送到控制器 301 中的加法器 323 与总线 313 上的移位位数相加，其结果就是下一条外部指令的起始偏移地址 SA 存入寄存器 321 中。此外，若加法器 323 的进位输出为‘0’，表示所述下一条外部指令的起始位置位于缓冲 303 中，可直接按上述方法进行对

齐。若加法器 323 的进位输出为‘1’，则表示所述下一条外部指令的起始位置位于缓冲 305 中。此时，在该进位输出的控制下，缓冲 305 中的内容被填充到缓冲 303 中，同时新的一个后续指令块被填充到缓冲 305 中，使得所述下一条外部指令的起始位置仍然位于缓冲 303 中，并按上述方法进行对齐。

[0125] 请参考图 3B，其为本发明所述对齐器运行过程的一个实施例。外部指令 351 跨越了指令块边界。其中，头部 353 位于指令块 357 中，尾部 355 位于指令块 359 中。根据本发明技术方案，指令块 357 和 359 分别存储在缓冲 303 和 305 中，并经选择器选择、拼接后形成如指令块 361 的形式作为循环移位器 307 的输入。此时，指令块 361 由三部分组成，从左到右依次是外部指令 351 的尾部 355、外部指令 351 的后续指令的一部分 363，以及外部指令 351 的头部 353。

[0126] 移位器 307 根据所述外部指令头部 353 起始字节在指令块中的偏移地址 313 为移位位移进行循环左移，使得外部指令 351 的起始地址与指令块的起始位置对齐。在本实施例中，循环移位得到的指令块 365 中除了外部指令 351 外，还有其后续指令的一部分 363，该部分对后续操作没有任何影响，可以忽略。

[0127] 回到图 2，经对齐器 203 移位对齐后的外部指令被送到提取阵列 205 中根据指令类型提取出各个指令域。提取阵列 205 由若干个结构相同的提取器构成。在此，提取器的数目大于等于外部指令集中任意指令包含的指令域的最大数目。在本发明所述处理器系统支持的所有外部指令集中，若指令最多包含 n 个指令域，则提取阵列 205 由 n 个提取器构成，且每个提取器均接收同样的外部指令作为输入，并根据存储器 201 送来的控制信号输出需要提取出的信息。

[0128] 请参考图 4A，其为本发明所述提取器的一个实施例。在本实施例中，提取器由循环移位器 401 和掩码器 403 构成。其中，循环移位器 401 根据接收到的移位位数，将输入的外部指令字循环移位，从而将指令中的特定指令域移至相应位置。掩码器 403 则对移位后的指令与掩码字进行按位与 (Bit AND) 操作，使得提取器的输出中除了所述特定指令域之外，其他部分为全‘0’。这样，就可以将外部指令的指令域移至内部指令对应的指令域的位置。

[0129] 请参考图 4B，其为本发明所述提取器运行过程的一个实施例。本实施例对外部指令 451 中的指令域 453 的移位、掩码进行说明。其中，循环移位器 401 的移位位数等于指令域在内部指令和外部指令中的差值。例如，指令域 453 位于外部指令 451 的第 10、11、12 位 (Bit)，而在对应的内部指令中该指令域应位于第 6、7、8 位，则对应的移位位数为左移 4 位 (即‘10’减‘6’)。这样，外部指令 451 经循环移位器 401 移位后得到如图 4B 中的移位后指令 455 的形式。

[0130] 在本实施例中，由于该指令域位于内部指令的第 6、7、8 位。因此，掩码字 457 的第 6、7、8 位均为‘1’，其他位均为‘0’。这样，移位后指令 455 在掩码器 403 中与掩码字 457 按位与后作为提取器的输出，即如图 4B 中的提取器输出 459 的形式。

[0131] 回到图 2，提取阵列 205 中的一部分提取器用于提取外部指令的操作码域，另一部分提取器则提取外部指令的其他指令域。例如，假设外部指令集的指令中操作码域最多有三个，那么在提取阵列 205 中，提取器 211、213 和 215 用于提取操作码域 (称为操作码提取器)，剩余的提取器 (如提取器 221、223、225 和 227) 用于提取其他指令域 (称为其他域提取器)。在此，提取器 211、213 和 215 提取出的操作码分别被移位至不同位置互不重叠，并

被送到操作码拼接器 209 进行按位或 (Bit OR) 操作，从而得到完整的操作码。该完整操作码被作为寻址地址送到存储器 201。

[0132] 当一个提取器被用于提取操作码域时，该提取器的控制信号（如：移位位数、掩码字等）均来源于相应的寄存器。例如，在图 2 中，寄存器 212 中的控制信号经选择器 222 选择后用于控制提取器 211；寄存器 214 中的控制信号经选择器 224 选择后用于控制提取器 213；寄存器 216 中的控制信号经选择器 226 选择后用于控制提取器 215。

[0133] 当一个提取器被用于提取其他指令域时，该提取器的控制信号均来源于存储器 201。存储器 201 由若干行映射信息构成，分为直接访问区及间接访问区。每行映射信息对应一个寻址地址。由于每个寻址地址对应一个完整的内部指令操作码，因此，一行或多行映射信息对应所述外部指令集之中的一条或多条外部指令，其中存储了相应的提取信息。所述提取信息包括该外部指令对应的内部指令的操作码、该外部指令除操作码域以外的各个指令域起始位置及宽度、所述指令域与相应的内部指令的指令域的位置关系等。

[0134] 在本发明中，可以根据外部指令的操作码直接对存储器 201 的直接访问区寻址，找到对应的那行映射信息。具体地，可以将操作码拼接器 209 输出的完整操作码本身作为对直接访问区的寻址地址，读出相应行中的映射信息。而存储器 201 的间接访问区则必须根据其他行映射信息中的索引值（即行地址信息）访问。例如，当一条外部指令对应多条内部指令时，可以先以该外部指令的完整操作码作为寻址地址在直接访问区中读出所述多条内部指令中第一条内部指令对应的映射信息，从而转换出第一条内部指令。而该映射信息中包含了所述多条内部指令中第二条内部指令对应的映射信息在间接访问区中的索引值。因此，根据该索引值，即可在间接访问区中找到所述第二条内部指令对应的映射信息，从而转换出第二条内部指令。如此反复，直到转换得到所述多条内部指令中的最后一条内部指令为止。

[0135] 请参考图 5A，其为本发明所述映射信息的一个示意图。图 5A 所示的映射信息一行即对应一条外部指令，即该外部指令对应一条内部指令。映射信息 501 由内部指令操作码 503、外部指令长度 505、若干个提取器配置信息（如提取器配置信息 507、509、511、513）和结束标志 515 构成。其中，内部指令操作码 503 就是所述外部指令对应的内部指令的操作码。外部指令长度 505 是所述外部指令本身的指令字长度，被送到对齐器 203 作为外部指令长度值 325 与当前指令起始点相加用于计算下一外部指令的起始点。结束标志 515 中存储了全‘0’，用以表示该行是外部指令对应的最后一行内部指令映射信息。

[0136] 映射信息 501 中提取器配置信息的数目与提取器的个数相同，且一一对应。每个提取器配置信息由三部分构成：移位位数 (R)、掩码值中‘1’的开始位置 (B) 和掩码值中‘1’的个数 (W)。其中，移位位数 R 被送到相应提取器用于控制循环移位器 401 的移位；开始位置 B 和个数 W 则被用于确定掩码值中‘1’的位置，即从 B 开始的连续 W 个掩码位的值为‘1’，其余掩码位的值为‘0’。

[0137] 请参考图 5B，其为本发明所述映射信息的另一个示意图。图 5B 所示的映射信息多行对应一种外部指令，即该外部指令对应多条内部指令。在此以一条外部指令对应三条内部指令为例，这三条内部指令相应信息对应的映射信息分别为映射信息 551、561 和 571。其中映射信息 551 位于存储器 201 中的直接访问区可由外部指令中提取的操作码译码后直接寻址的地址区域。而映射信息 561 和 571 位于存储器 201 中的间接访问区，必须根据直

接访问区内映射信息（如映射信息 551）中存储的索引值寻址访问。与图 5A 中的映射信息 501 类似，映射信息 551 也由内部指令操作码 503、外部指令长度 505、若干个提取器配置信息（如提取器配置信息 507、509、511、513）和结束标志 515 构成。而映射信息 561 和 571 中也包含了内部指令操作码 503、若干个提取器配置信息（如提取器配置信息 507、509、511、513）和结束标志 515，但可以不包含外部指令长度 505。其中，三行映射信息中的内部指令操作码 503 分别对应了所述外部指令对应的三条内部指令的操作码。映射信息 551 中的外部指令长度 505 是所述外部指令本身的指令字长度，被送到对齐器 203 作为外部指令长度值 325 用于计算下一外部指令的起始点。在本实施例中，映射信息 551 和 561 的结束区标示并非结束而是指向下一条映射信息的地址。对于其他情况也可以此类推。映射信息 551 和 561 的指令字长度 505 中各存储了指向后续映射信息的索引。即，映射信息 551 的指令字长度 505 中存储了映射信息 561 在存储器 201 中的索引值，映射信息 561 的指令字长度 505 中存储了映射信息 571 在存储器 201 中的索引值。而映射信息 571 中作为多条内部指令对应一条外部指令的最后一条内部指令信息，其指令字长度 505 才给出该外部指令的指令长度。作为所述外部指令对应的最后一行映射信息，映射信息 571 的结束标志 515 中存储了全‘0’。这样，根据操作码提取器提取到的完整操作码可以找到第一行映射信息，之后在各行映射信息的结束标志 515 的控制下，存储器 201 可以正确地输出一条外部指令对应的所有内部指令的映射信息，从而正确地进行指令集转换。

[0138] 回到图 2，对于对齐器 203 输出的任意一条外部指令，由操作码提取器提取出的完整操作码作为寻址地址可以从存储器 201 中读出相应的内部指令操作码 503 经总线 230 送往指令拼接器 207，及读出对应该外部指令各个指令域的提取信息并分别送到各个其他域提取器。每个其他域提取器根据所述提取信息中的域起始位置、域宽度及移位位数信息，将外部指令相应的指令域移至特定位置，并进行掩码操作，使得其他域提取器的输出除了所述移位后的指令域之外均为‘0’。

[0139] 这样，所述外部指令中除操作码域以外的所有指令域在各个其他域提取器中被移至内部指令所需的指令域后，输出到指令拼接器 207 进行按位或操作，并拼接在存储器 201 输出的内部指令操作码之后，形成符合内部指令集格式的内部指令。该内部指令被送往处理器核执行，从而实现相应的外部指令的功能。

[0140] 请参考图 5C、5D 和 5E，其为本发明所述映射信息存储器运行的三个实施例。在这些实施例中，存储器 201 分为直接访问区 531 和间接访问区 533。其中，间接访问区的地址比直接访问区高，例如外部指令操作码所形成的地址为 n 位，存储器 201 的地址为 n+1 位。当地址的最高位为‘0’访问直接访问区 531，当地址的最高位为‘1’访问间接访问区 533。

[0141] 存储器 201 中每行映射信息都包含了一个两位的结束标志（如图由 Y 位和 Z 位构成），用于表示与该行映射信息对应的外部指令及内部指令之间的转换关系，即是一条外部指令对应一条内部指令，还是一条外部指令对应多条内部指令，还是多条外部指令对应一条内部指令，控制转换器以哪一种方式处理下一条指令。具体地，图 5C 中的标志 535 的值‘00’表示该行映射信息对应当前的外部指令，即一条外部指令对应一条内部指令；图 5D 中的标志 545 的值‘10’表示该行映射信息不但对应当前的外部指令，还对应其后一条外部指令，即多条外部指令对应一条内部指令；图 5E 中的标志 555 的值‘01’表示该行映射信息及该行映射信息中索引值指向的映射信息一同对应当前的外部指令，即一条外部指令对应多

条内部指令。所述标志中的 Y 位被用于表示是否对下一条外部指令进行转换。具体地，若 Y 位为‘0’，表示对当前外部指令（或包含当前外部指令在内的若干连续外部指令）的转换已经完成，下一周期开始对下一外部指令的转换。若 Y 位为‘1’，表示当前外部指令的转换尚未完成，下一周期将继续进行相关转换，不能开始下一外部指令的转换。

[0142] 在本实施例中，所述标志被存储到寄存器 537，同时该行映射信息中索引值被存储到寄存器 539 供下一条指令转换使用。寄存器 537 中存储的前一外部指令的所述标志就可以在处理当前外部指令时被用于控制选择器 541（由所述标志中的 A 位控制）和地址拼接逻辑 543（由所述标志中的 B 位控制）。

[0143] 图中寄存器 537 的 Y 输出控制一个二路选择器。当 Y 值为‘0’时，选择来自于从外部指令中操作码作为存储器 201 的地址，当 Y 值为‘1’时，选择存在寄存器 539 中的从存储器 201 中前一条指令的索引值作为当前指令转换时的存储器 201 地址。Z 值是作为一个地址高位拼接到来自于从外部指令中操作码而形成的地址。当 Z 值为‘0’时，存储器 201 上的地址指向直接访问区，当 Z 值为‘1’时，存储器 201 上的地址指向间接访问区。图中圆圈表示总线拼接。

[0144] 在图 5C 实施例中，前一外部指令对应的结束值 YZ 为‘00’，且该内部指令可以根据相应的映射信息按之前所述方法产生，那么当前外部指令应对应至少一条新的内部指令。此时地址拼接逻辑 543 的一个输入是来源于操作码拼接器 209 的当前外部指令的完整操作码，另一个输入是寄存器 537 中标志的 Z 位（‘0’），即在所述完整操作码之前拼接全‘0’，因此地址拼接逻辑 543 的输出依然是当前外部指令的完整操作码，对应直接访问区 531 的地址。而选择器 541 受标志中的 Y 位（‘0’）控制，选择来源于或逻辑的输出作为存储器 201 的寻址地址。这样，即可从存储器 201 的直接访问区 531 中读出当前外部指令对应的映射信息，按之前所述方法对相应指令域移位及掩码，并送往指令拼接器 207。又因为标志中的 Y 位为‘0’，因此下一周期即可开始对下一外部指令进行转换。

[0145] 请参考图 5F，其为本发明所述指令拼接器的一个实施例。其中，寄存器 563 中存储的是已经转换完毕得到的内部指令或尚位转换完毕得到的中间转换结果。所述标志中的 Z 位被存储在寄存器 561 中，并在下个周期送往与逻辑 567，以及经反相器反相后输出作为表示寄存器 563 中的内部指令是否已转换完成的信号。与逻辑 567 的另一个输入来源于寄存器 563 中存储的值，其输出被送往或逻辑 569。或逻辑 569 的另一个输入是来源于总线 559 的从各个提取器送来的移位掩码得到的结果。寄存器 563 的输出就是指令拼接器 207 的输出 667。

[0146] 对于图 5C 实施例所述情况，由于标志的 Z 位为‘0’，因此在下一周期，与逻辑 567 的输出为‘0’，则或逻辑 569 的输出就是各个提取器移位掩码得到的结果。这些结果在寄存器 563 中拼接成一条完整的内部指令。此时反相器 565 输出的值为‘1’（即上述 Z 位的反相值），表示转换完毕，且寄存器 563 中存储的内容就是转换得到的内部指令。这样，就完成了一条外部指令向一条内部指令的转换，并在下一周期输出，同时转换器开始读下一条外部指令的转换。

[0147] 在图 5D 实施例中，前一外部指令对应的标志值为‘10’，表示该外部指令对应多条内部指令，且上一次映射信息对应的内部指令尚不足以完成该转换，那么在完成对前一外部指令的转换之前，不能对当前外部指令进行转换。此时，寄存器 539 中存储的就是所述上

一次映射信息中包含的索引值，即所述上一次映射信息的后一个映射信息（这两个映射信息均对应所述前一外部指令）在间接访问区 533 中的寻址地址。选择器 541 则受标志中的 Y 位（‘1’）控制，选择寄存器 539 输出的所述索引值。由于该索引值对应的存储器 201 地址空间位于间接访问区 533，因此可从间接访问区 533 中读出所述前一外部指令对应的映射信息，按之前所述方法对相应指令域移位及掩码，并送往指令拼接器 207。由于标志中的 Y 位为‘1’，因此下一周期继续对当前外部指令转换，不能开始下一外部指令的转换。

[0148] 此时，在指令拼接器 207 中，标志的 Z 位为‘0’，因此在下一周期，与逻辑 567 的输出为‘0’，则或逻辑 569 的输出就是各个提取器移位掩码得到的结果。这些结果在寄存器 563 中拼接成一条完整的内部指令。此时反相器 565 输出的值为‘1’（即上述 Z 位的反相值），表示转换完毕，且寄存器 563 中存储的内容就是转换得到的内部指令。这样，在一条外部指令向对应多条内部指令转换过程中，生成了所述多条内部指令中的一条，并在下一周期输出。同时，从下一周期开始，重复上述过程，直到对应的映射信息中的标志的 Y 位为‘0’，表示该映射信息对应的内部指令是所述多条内部指令中的最后一条，并在下一周期输出该内部指令，完成了一条外部指令向多条内部指令的转换，同时转换器开始读下一条外部指令的转换。

[0149] 在图 5E 实施例中，前一外部指令对应的标志值为‘01’，表示该外部指令及其后一条外部指令（即当前外部指令）对应同一条内部指令，那么应该继续对当前外部指令进行转换，直至产生所述多条外部指令对应的同一条内部指令。此时，地址拼接逻辑 543 的一个输入是来源于操作码拼接器 209 的当前外部指令的完整操作码，另一个输入是寄存器 537 中标志的 Z 位（‘1’），即在所述完整操作码之前拼接一个额外地址，使得地址拼接逻辑 543 的输出是对应间接访问区 533 的地址。而选择器 541 受标志中的 Y 位（‘0’）控制，选择来源于或逻辑的输出作为存储器 201 的寻址地址。这样，即可从存储器 201 的间接访问区 533 中读出相应的映射信息，即根据所述前一外部指令及当前外部指令共同对应的映射信息。之后，按之前所述方法对相应指令域移位及掩码，并送往指令拼接器 207。

[0150] 此时，在指令拼接器 207 中，标志的 Z 位为‘1’，因此在下一周期，与逻辑 567 的输出是存储在寄存器 563 中的值（即转换的中间结果），则或逻辑 569 的输出就是当前各个提取器移位掩码得到的结果与所述中间结果的经组合（如按位或操作）得到的结果。这些结果在寄存器 563 中进一步被拼接成新的中间结果。此时反相器 565 输出的值为‘0’（即上述 Z 位的反相值），表示转换仍未完毕。又由于标志中的 Y 位为‘0’，转换器开始对下一外部指令开始转换，则重复上述过程，连续多条外部指令相应指令域的移位掩码结果经或逻辑 569 被组合到一起，将多条外部指令转换为一条内部指令，直到所述 Z 位为‘0’，表示当前外部指令是所述内部指令对应的多条外部指令中的最后一条。此时，反相器 565 输出的值为‘1’（即上述 Z 位的反相值），表示转换完毕，且寄存器 563 中存储的内容就是转换得到的内部指令。这样，完成了多条外部指令向一条内部指令的转换，

[0151] 需要说明的是，在本发明中，存储器 201 可以由可重复擦写的随机存储器 (RAM) 构成，根据所需支持的不同外部指令集向该随机存储器写入不同的映射信息；也可以由只读存储器 (ROM) 构成，即固定支持一种或多种外部指令集；还可以由能够实现同样功能的逻辑电路构成，固定支持一种或多种外部指令集。可以将缓冲器的一部分指定作为存储器 201 使用而不做缓存使用。

[0152] 此外，如果外部指令是定长的，且规定提取器的长度等于指令字的长度，则转换器200中可以省去对齐器203。根据本发明技术方案，转换器200可以根据配置支持不同的外部指令集。那么当其中一种外部指令集的指令长度与提取器的长度相同时，可以由选择器204直接选择外部指令送往各个提取器；否则选择器204选择对齐器203的输出送往各个提取器。其他操作与之前实施例所述相同，在此不再赘述。

[0153] 根据本发明技术方案，可以在处理器系统不同层次的缓存中存储不同的指令集的指令，以提高处理器系统的性能。例如，可以在处理器系统的二级缓存中存储外部指令，在一级缓存中存储内部指令，并在所述外部指令被填充到一级缓存的过程中进行指令集转换。请参考图6，其为本发明所述包含多层缓存的处理器系统的一个实施例。

[0154] 在图6中，处理器系统由处理器核601、主动表604、扫描转换器608、轨道表610、替换模块611、循迹器614、块地址映射模块620、偏移地址映射模块618、偏移地址转换器622、减法器928、一级缓存602，二级缓存606和选择器640、660、680、638、692、694及696构成。图6中的空心圆圈表示总线的拼接。图6中没有显示的还有一个控制器，该控制器接收从块地址映射模块620、扫描转换器608、主动表604、轨道表610及替换模块611的输出控制各功能模块的操作。

[0155] 在本发明中，二级缓存606中存储的是外部指令，而一级缓存602中存储的是相应的内部指令。可以用第一地址和第二地址来表示指令在一级缓存或二级缓存中的位置信息。在此，第一地址和第二地址可以是一级缓存的寻址地址，也可以是二级缓存的寻址地址。

[0156] 当一条内部指令已经被存储在一级缓存602中时，可以用BN1X表示该内部指令所在指令块的一级块号（即指向一级缓存中相应的一个一级指令块），并用BN1Y表示该内部指令的一级块内偏移量（即该内部指令在一级指令块中的相对位置）。当一条外部指令已经被存储在二级缓存606中时，可以用BN2X表示该外部指令所在指令块的二级块号（即指向二级缓存中相应的一个二级指令块），并用BN2Y表示该外部指令的二级块内偏移量（即该外部指令在二级指令块中的相对位置）。为了便于说明，可以用BN1代表BN1X和BN1Y，用BN2代表BN2X和BN2Y。由于本发明中凡在一级缓存中的内部指令对应的外部指令在二级缓存中均有存储，因此对于一级缓存中存储的内部指令，可以用BN1或BN2表示。

[0157] 主动表604中的表项与二级缓存606中的存储块一一对应。主动表604中的每个表项存储了一个二级指令块地址与一个二级块号BN2X的匹配对，指明了该指令块地址对应的二级指令块存储在二级缓存606中的哪个存储块中。在本发明中，可以根据一个二级指令块地址在主动表604中进行匹配，并在匹配成功的情况下得到一个BN2X；也可以根据一个BN2X对主动表604寻址，以读出对应的二级指令块地址。

[0158] 当外部指令从二级缓存608向一级缓存602填充时，扫描转换器608计算外部指令中分支指令的分支目标地址，外部指令且由608中的指令转换器200转换成内部指令。计算得到的分支目标地址被送至主动表604与其中存储的指令块地址匹配确定该分支目标是否已经存储在二级缓存606中。如果匹配不成功，则分支目标指令所在的指令块尚未被填充到二级缓存606中，那么在将该指令块更低层存储器填充到二级缓存606中的同时，在主动表604中建立相应的二级指令块地址与二级块号的匹配对。

[0159] 扫描转换器608对从二级缓存606向一级缓存602填充的指令块（外部指令）进

行转换和审查，并提取出对应内部指令的轨迹点信息填充到轨道表 610 的相应表项中，从而建立该二级指令块对应的至少一个一级指令块的轨道。具体地，在建立轨道时，首先由替换模块 611 产生一个 BN1X 指向一条可用轨道。在本发明中，替换模块 611 可以根据替换算法（如 LRU 算法）确定可用轨道。

[0160] 具体地，扫描转换器 608 对从二级缓存 606 填充到一级缓存 602 的每一条外部指令进行审查，并提取出某些信息，如：指令类型、指令源地址和分支指令的分支增量，并基于这些信息计算出分支目标地址。对于直接分支指令，可以通过对该指令所在指令块的块地址、该指令在指令块中的偏移量和分支增量 (Branch Offset) 三者相加得到分支目标地址。所述指令块地址可以是从主动表 604 中读出并被直接送往扫描转换器 608 中加法器的。也可以在扫描转换器 608 中增加用于存储当前指令块地址的寄存器，这样主动表 604 就不需要实时地送出指令块地址。在本实施例中，直接分支指令的分支目标地址由扫描转换器 608 产生，而间接分支指令的分支目标地址由处理器核 601 产生，且这两者对应的都是外部指令地址。此外，扫描转换器 608 还将所述每一条外部指令转换为对应的一条或多条内部指令，且在转换过程中不改变分支指令的分支增量，即外部分支指令中的分支增量与其对应的内部分支指令中的分支增量相等，保证处理器核 601 产生的间接分支指令的分支目标地址的正确性。

[0161] 块地址映射模块 620 中对应每个二级缓存块的每一行有复数个表项，每个表项中存储与此二级缓存块中一部分（称为二级缓存块的子块）对应的一级缓存块的一级块号 (BN1X) 以及该二级缓存子块在二级缓存块内的起始偏移量 (BN2Y)。其中各表项中的 BN2Y 由左向右递增排列。当一个新的表项被加入块地址映射模块 620 中的一行时，其 BN2Y 会由比较器 924 与该行上现有其他表项的 BN2Y 值比较，并由移位器 926 将 BN2Y 值大于新表项的 BN2Y 值的表项右移，空出位置供新表项存放。

[0162] 块地址映射模块 620 中的行与主动表 604 中的行及二级缓存 606 中的存储块一一对应，并由同一个 BN2X 指向。块地址映射模块 620 用于存储对应二级块号与一级块号的对应关系，如图 6 所示，其表项格式 680 包括一级块号 BN1X 和二级块内偏移量。这样，对于一个 BN2，可以根据其中的 BN2X 找到块地址映射模块 620 中的一行，再用其中的 BN2Y 在该行各个表项中存储的有效的 BN2Y 进行比较，即可读出相应比较成功的表项中的 BN1X（即该 BN2Y 对应的外部指令的相应内部指令对应的 BN1X），从而将 BN2X 转换为相应的 BN1X，或得到比较不成功的结果（即该 BN2Y 对应的外部指令的相应内部指令尚未存储在一级缓存 602 中）。

[0163] 本实施例中，轨道表 610 的格式为 686 或 688。686 由三部分构成：格式 (TYPE)，二级块号 (BN2X) 及二级块内偏移 (BN2Y)。其中格式中含指令类型地址，包括非分支指令，无条件直接分支指令，有条件直接分支指令，无条件间接分支指令，有条件间接分支指令。在此，有条件直接分支指令、无条件直接分支指令、有条件间接分支指令和无条件间接分支指令可以统称为分支指令，其对应的轨迹点为分支点。格式中还包含地址类型，其在 686 格式中是二级缓存地址 BN2。688 的格式也由三部分构成：格式 (TYPE)，一级块号 (BN1X) 及一级块内偏移 (BN1Y)。688 格式中指令类型与 686 的相同，但是地址类型在 688 中固定为一级缓存地址 BN1。本实施例中，块地址映射模块 620 中的存储器 920 的格式如 684 所示，其为一级缓存块地址 BN1X 与二级缓存块内偏移地址 BN2Y 的组合。

[0164] 轨道表 610 含有复数个轨迹点 (track point)。一个轨迹点是轨道表中的一个表项，可含有至少一条指令的信息，比如指令类别信息、分支目标地址等。在本发明中轨迹点本身的轨迹点地址与该轨迹点所代表指令的指令地址相关 (correspond)；而分支指令轨迹点中含有分支目标的轨迹点地址，且该轨迹点地址与分支目标指令地址相关。与一级缓存 602 中一系列连续内部指令所构成的一个一级指令块相对应的复数个连续的轨迹点称为一条轨道。该一级指令块与相应的轨道由同一个一级块号 BN1X 指示。轨道表含有至少一条轨道。一条轨道中的总的轨迹点数可以等于轨道表 610 中一行中的表项总数。这样，轨道表就成为一个以轨道表项地址对应分支源地址、表项内容对应分支目标地址来代表一条分支指令的表。此外，在轨道表 610 的每一行中还可以额外增加一个二级块号表项，用于记录该行第一个轨迹点对应的外部指令的 BN2。这样，就可以在某个一级指令块被替换时，将以该行为分支目标的其他轨道表行中的该 BN1 转换为相应的 BN2，使该行可以被其他指令行写入而不致引起错误。

[0165] 轨道表 610 中记录了程序运行的可能路径或程序执行流的可能流向，因此循迹器 614 可以根据轨道表 610 中的程序流和处理器核 601 的反馈沿程序流循迹。因为在一级缓存器 602 中存有与轨道表表项相应的内部指令，一级缓存器 602 以循迹器 614 的输出总线 631 为读地址，跟随循迹器 614 所遵循的程序流而通过总线 695 送出指令供处理器核 601 执行。轨道表 610 中某些分支目标是用二级缓存器地址 BN2 记录的，其目的是仅将需用的外部指令转换成内部指令存到一级缓存器，使得一级缓存器可以有较二级缓存器更小的容量和更快的速度。当循迹器 614 读出的表项中分支以 BN2 记录时，此时要将该 BN2 送往块地址映射模块 620 等模块匹配或扫描转换模块 608 转换获得 BN1 地址，将指令填充到一级缓存 602 中 BN1 地址，也将该 BN1 地址填回轨道表中该表项中，循迹器 614 沿该 BN1 并根据处理器核 601 反馈的指令执行结果（如：分支指令的执行结果），控制一级缓存 602 向处理器核 601 输出指令供执行。

[0166] 在本发明中，可以用所述第一地址和第二地址来表示轨迹点在轨道表中的位置信息。而直接分支点的指令类型中则还可以包含分支目标寻址地址是以 BN1 表示（即分支目标为 BN1 的直接分支指令）还是以 BN2 表示（即分支目标为 BN2 的直接分支指令）的信息。当一个分支点中存储的是 BN1 时，说明该分支点的分支目标内部指令所在的内部指令块已经被存储在一级缓存 602 中由该 BN1X 指向的存储块中，且根据该 BN1Y 可以从中找到所述分支目标内部指令。当一个分支点中存储的是 BN2 时，说明该分支点的分支目标外部指令所在的外部指令块已经被存储在二级缓存 606 中由该 BN2X 指向的存储块中，且根据该 BN2Y 可以从中找到所述分支目标外部指令，但无法直接确定该分支目标外部指令对应的内部指令是否已经存储在一级缓存 602 中。

[0167] 偏移地址映射模块 618 中的行与轨道表 610 中的行及一级缓存 602 中的存储块一一对应，并由同一个 BN1X 指向。偏移地址映射模块 618 用于存储二级缓存 606 中的外部指令偏移地址与一级缓存 602 中的内部指令偏移地址之间的对应关系。偏移地址转换器 622 则可以根据偏移地址映射模块 618 送来的由 BN1X 指向的映射关系（即 BN2Y 和 BN1Y 的映射关系），将接收到的 BN2Y 转换为相应的 BN1Y，或将接收到的 BN1Y 转换为相应的 BN2Y。

[0168] 因此，当需要将 BN2 转换为 BN1 时，首先根据 BN2X 和 BN2Y，在块地址映射模块 620 中转换得到 BN1X，再根据偏移地址映射模块 618 中由该 BN1X 指向的行中的映射关系，将

BN2Y 转换为 BN1Y，从而完成 BN2 向 BN1 的转换。

[0169] 当需要将 BN1 转换为 BN2 时，首先从轨道表 610 中由 BN1X 指向的行中的所述额外表项中读出相应的 BN2，其中 BN2X 就是所述 BN1X 指向的内部指令块对应的外部指令块号，BN2Y 就是所述 BN1X 指向的内部指令块对应的外部指令在其所在外部指令块中的起始位置。根据偏移地址映射模块 618 中由该 BN1X 指向的行中的映射关系及所述起始位置，即可将 BN1Y 转换为 BN2Y，从而完成 BN1 向 BN2 的转换。

[0170] 在图 6 中，主要的总线有三类：外部指令地址总线、BN1 总线和 BN2 总线。其中，外部指令地址总线主要有总线 657、683 和 675；BN1 总线主要有总线 631 和 693；BN2 总线主要有总线 633 和 687。此外，还有其他一些总线，如 BN2X 总线 639、BN2Y 总线 637，以及映射关系总线 691。

[0171] 具体地，总线 675 上的内容是主动表 604 中由 BN2X 指向的行中存储的外部指令块地址（即二级缓存块地址）。该地址被送回扫描转换器 608 以计算直接分支指令的分支目标地址。

[0172] 总线 657 上的内容是扫描转换器 608 在审查发现分支指令时输出的直接分支指令的分支目标指令地址，总线 683 上的内容是处理器核 601 在执行间接分支指令时输出的分支目标指令地址。总线 657 和 683 的格式均与外部指令地址格式相同。其中，块地址部分（高位部分）被选择器 680 选择后经总线 681 送往主动表 604 与其中储存的外部指令块地址匹配以获得一个二级块号 BN2X 并经总线 671 从二级缓存器 606 中读取外部指令。总线 671 的格式为 BN2X，与总线 657 上外部指令地址偏移部分（低位部分）的 BN2Y 拼接成一个完整的 BN2 地址送往轨道表 611 存储。总线 671 上 BN2X 也被送到选择器 640。选择器 640 选择总线 671 及轨道表 610 输出经总线 633 而来的 BN2X 中的一个作为 BN2X 放上总线 639，用以读取块地址映射模块 620 中的一行数据进行 BN2 到 BN1 的映射。

[0173] 总线 637 是三输入选择器 638 的输出，三输入选择器 638 选择总线 633、657 或 683 上的 BN2Y 送到块地址映射模块 620，在由总线 639 上的 BN2X 指向的行中匹配出相应的 BN1X

[0174] 总线 633 是轨道表 610 的输出，其格式可以是 BN1 或 BN2。当其格式为 BN2 时被送到块地址映射模块 620 及偏移地址映射模块 618 中将 BN2X 映射为 BN1X。其映射还需要将该 BN2 中的 BN2Y 经总线 637 送往减法器 928 与块地址映射模块 620 输出的相应的二级子存储块的起始地址相减以获得正确的净块内偏移地址供偏移地址转换器 622 使用，将 BN2Y 转换为 BN1Y。所述 BN1X 和 BN1Y 合并为 BN1 被写回轨道表 610。总线 633 上的 BN2X 还可以被送到主动表 604 读出对应的外部指令块地址经总线 657 送往扫描转换器 608，与总线 633 直接送往扫描转换器 608 的 BN2Y 一同构成外部指令地址。此外，总线 633 上的 BN2X 还可以经总线 673 送往二级缓存 606 读出对应的外部指令块。

[0175] 总线 631 是循迹器 614 的输出，其格式为 BN1。该输出被送到一级缓存 602 作为地址以读取指令供处理器核 601 使用。

[0176] 总线 693 是替换模块 611 的输出，格式为 BN1X，其意义为向扫描转换器 608 提供下一个可用的一级块号 BN1X（或轨道号），供扫描转换器 608 填充转换所得的内部指令。总线 693 上的 BN1X 也与来自总线 657 的 BN2Y 共同放上总线 665（及构成块地址映射模块 620 中表项内容）送往选择器 940 以供按地址顺序存放于块地址映射模块 620。因此，665 总线上的格式为 BN1X 和 BN2Y。总线 693 控制一级缓存 602 的写入块地址与来自扫描转换模块

608 输出的 BN1Y 总线 669 作为写入块内偏移地址，控制将扫描转换器 608 转换得到的内部指令经总线 667 填充到一级缓存 602。同时，总线 693 与总线 669 还共同寻址将与内部指令相应的格式（由扫描转换模块 608 经总线 687 送出）、分支目标（由总线 671 上的 BN2X 与总线 657 上的 BN2Y 拼接到总线 687）经总线 687 同步写入轨道表 610。

[0177] 总线 687 将其上的指令类型、BN2Y 与来自总线 671 的 BN2X 拼接成一个完整的轨迹点内容送往轨道表 610 存储。

[0178] 总线 954 是块地址映射模块 620 的输出，其中的 BN1X 用于从偏移地址映射模块 618 中读取相应的偏移地址映射信息送往偏移地址转换器 622；其中的 BN2Y 输出送往减法器 928 与总线 633 上送来的 BN2Y 值相减，其结果被送往偏移地址转换器 622。偏移地址转换器 622 根据输入将总线 954 上的 BN2Y 映射成 BN1Y 地址。来自总线 954 的 BN1X 地址与偏移地址转换器 622 输出的 BN1Y 地址被拼接成完整 BN1，经总线 685 送往三输入选择器 692 的一个输入端。

[0179] 选择器 692 选择总线 685 上的 BN1、总线 687 上的 BN2 或总线 693 上的 BN1（其中总线 693 送来的 BN1X 被加上为‘0’的 BN1Y 拼接成完整的 BN1）送往轨道表 610 作为写入的轨迹点内容。

[0180] 请参考图 7A，其为本发明所述基于轨道表的缓存结构的实施例。为了便于说明，在图 7A 中只显示了部分器件或部件。如之前实施例所述，轨道表 610 的行与一级缓存 602 的存储块一一对应，且轨道表行（即轨道）中的表项（即轨迹点）数目比一级存储块中的指令数目多一个。其中，轨道的最后一个轨迹点中存储了指向顺序执行的下一轨道的位置，其余表项与一级存储块中的指令一一对应，并存储了程序执行流信息（如指令类型、分支目标地址等），且轨道中从左向右的每个轨迹点对应的地址递增。

[0181] 轨道表 610 的读端口在循迹器 614 输出的读指针 631 的寻址下，输出相应轨迹点内容并放上总线 633，而控制器则检测所述总线 633 上的内容。

[0182] 如果该内容中的指令类型是非分支指令，选择器 738 选择增量器 736 的输出使得循迹器向右移动达到下一个地址（即更大的地址）。

[0183] 如果该内容中的指令类型是无条件分支，则选择器 738 选择总线 633 上的分支目标地址，使得读指针 631 转到由总线 633 上分支目标地址对应的轨迹点位置。

[0184] 如果该内容中的指令类型是有条件分支，则循迹器 614 暂停更新并等待，直到处理器核 601 产生分支转移是否发生的 TAKEN 信号 635。如果分支转移没有发生，则如之前非分支指令的做法运行，如果分支转移发生，则如之前无条件分支指令的做法运行。

[0185] 轨道表 610 写端口对应的写入地址有两个来源，分别是选择器 694(BN1X) 和 696(BN1Y)。当建立轨道时，替换模块 611 输出行地址 BN1X，而扫描转换器 608 输出列地址 BN1Y。当循迹器 614 读出的轨迹点内容中存储的是 BN2 时，该 BN2 被送往块地址映射模块 620 或扫描转换器 608 等产生 / 生成 BN1，该 BN1 需要被写回该轨迹点中（即读出、修改及写回，read modify write）；当循迹器 614 读出的轨迹点内容中的指令类型是间接分支指令时，将处理器核 601 产生的间接分支目标地址送往主动表 604、块地址映射模块 620 等产生 / 生成 BN1，该 BN1 也需要被写回该轨迹点中。在这两种情况下，轨道表 610 的写入地址均是当时的读出地址。

[0186] 轨道表 610 写端口本身有三个来源：总线 685、687 和 693，经选择器 692 选择后作

为写入内容。其中总线 685 上的值是块地址映射模块 620 和偏移地址转换器 622 输出的 BN1，总线 687 上的值是二级缓存地址形式 (BN2) 的分支目标地址，而总线 693 上的值是将被写入轨道最后一个表项中的指向顺序执行下一轨道的 BN1。

[0187] 在本实施例中，在外部指令被转换为内部指令的同时，扫描转换器 608 审查、提取出相应信息。本实施例中，轨道表内容共有三部分：若该内部指令是非分支指令或间接分支指令，则选择器 694 选择由替换模块 611 产生的该内部指令对应的 BN1X 693 作为轨道表 610 写地址中的第一地址，选择器 696 选择扫描转换器 608 输出的该分支内部指令在其所在指令块中的块内偏移量 669 作为轨道表 610 写地址中的第二地址，将该指令类型（即非分支指令或间接分支指令）作为写入内容写入轨道表 610 中，完成该轨迹点的建立。

[0188] 若该内部指令是直接分支指令，则扫描转换器 608 计算分支目标地址。所述分支目标地址中的块地址经总线 657 被送到主动表 604 匹配。若匹配成功，得到匹配成功项对应的 BN2X 经总线 671、639 送往块地址映射模块 620，并将所述分支目标地址中的块内偏移量（即 BN2Y）经总线 657、637 送往块地址映射模块 620。在块地址映射模块 620 中由所述 BN2X 指向的行中查找对应的 BN1X。若存在有效的 BN1X，则从偏移地址映射模块 618 中读出该 BN1X 指向的行中的映射关系并送往偏移地址转换器 622 将所述 BN2Y 转换为 BN1Y。选择器 694 选择由替换模块 611 产生的该内部指令对应的 BN1X 693 作为轨道表 610 写地址中的第一地址，选择器 696 选择扫描转换器 608 输出的该分支内部指令在其所在指令块中的块内偏移量 669 作为轨道表 610 写地址中的第二地址，而所述 BN1X 和 BN1Y 被合并为 BN1 放上总线 693 并经选择器 692 选择后与所述提取出的指令类型一同作为轨迹点内容写入轨道表 610 中，完成该轨迹点的建立。此时该轨迹点中包含的是 BN1。

[0189] 若在块地址映射模块 620 不存在该 BN2X 及 BN2Y 对应的有效 BN1X，则选择器 694 选择由替换模块 611 产生的该内部指令对应的 BN1X 693 作为轨道表 610 写地址中的第一地址，选择器 696 选择扫描转换器 608 输出的该分支内部指令在其所在指令块中的块内偏移量 669 作为轨道表 610 写地址中的第二地址，将总线 671 上的该 BN2X 和扫描转换器 608 输出的该 BN2Y 拼接成 BN2 放上总线 687 并经选择器 692 选择后与所述提取出的指令类型一同作为轨迹点内容写入轨道表 610 中，完成该轨迹点的建立。此时该轨迹点中包含的是 BN2。

[0190] 若所述分支目标地址中的块地址在主动表 604 中不成功，表示该分支目标地址对应的外部指令尚未存储在二级缓存 606 中，则根据替换算法（如 LRU 算法）分配一个二级存储块的块号 BN2X，并将该分支目标地址送往更低层次的存储器取回相应指令块存储到二级缓存 606 由所述 BN2X 指向的存储块中。选择器 694 选择由替换模块 611 产生的该内部指令对应的 BN1X 693 作为轨道表 610 写地址中的第一地址，选择器 696 选择扫描转换器 608 输出的该分支内部指令在其所在指令块中的块内偏移量 669 作为轨道表 610 写地址中的第二地址，直接将该 BN2X 及所述分支目标地址中的块内偏移地址（及 BN2Y）合并为 BN2 放上总线 687 并经选择器 692 选择后与所述提取出的指令类型一同作为轨迹点内容写入轨道表 610 中，完成该轨迹点的建立。此时该轨迹点中包含的是 BN2。

[0191] 在上述过程中，轨道表 610 的写地址中的第一地址 (BNX) 还经总线 745 指向偏移地址映射模块 618 中的相应行，使得每个内部指令块与相应的外部指令的映射关系被存储到所述行中。此外，若被转换填充的外部指令对应的内部指令多于一个一级存储块可以容

纳的数目时，超出部分依次填充到由替换模块611新产生的BN1X指向的一级存储块并建立相应轨道。重复上述过程，即可实现从二级缓存向一级缓存转换、填充指令并建立相应的轨道。

[0192] 循迹器614则由寄存器740、增量器736和选择器738组成，其读指针631（即寄存器740的输出）指向轨道表110中处理器核601即将执行的指令（即当前指令）对应的轨迹点，并读出轨迹点内容经总线633送往选择器738。同时，读指针631对一级缓存602寻址，读出该当前指令并送往处理器核601供执行。

[0193] 若所述轨迹点内容中的指令类型显示该指令为非分支指令，则选择器738选择来源于增量器736的对寄存器740的值增一的结果作为输出送回寄存器740，使得下一周期寄存器740的值增一，即读指针631指向下一个轨迹点并从一级缓存602中读出对应内部指令供处理器核601执行。

[0194] 若所述轨迹点内容中的指令类型显示该指令为分支目标为BN1的无条件直接分支指令，则选择器738选择该BN1作为输出送回寄存器740，使得下一周期寄存器740的值被更新为该BN1，即读指针631指向分支目标内部指令对应的轨迹点并从一级缓存602中读出该分支目标内部指令供处理器核601执行。

[0195] 若所述轨迹点内容中的指令类型显示该指令为分支目标为BN1的有条件直接分支指令，则选择器738根据处理器核执行该分支指令时产生的表示分支转移是否发生的TAKEN信号635进行选择，同时暂停寄存器740的更新直至处理器核601送来有效的TAKEN信号635。此时，若TAKEN信号635的值为‘1’，表示分支转移发生，选择轨道表输出的BN1作为送回寄存器740，使得下一周期寄存器740的值被更新为该BN1，即读指针631指向分支目标内部指令对应的轨迹点并从一级缓存602中读出该分支目标内部指令供处理器核601执行。若TAKEN信号635的值为‘0’，表示分支转移没有发生，则选择增量器736的对寄存器740的值增一的结果作为输出送回寄存器740，使得下一周期寄存器740的值增一，即读指针631指向下一个轨迹点并从一级缓存602中读出对应内部指令经总线695供处理器核601执行。

[0196] 若所述轨迹点内容中的指令类型显示该指令为分支目标为BN2的直接分支指令（包括有条件、无条件两种情况），则该BN2被送往块地址映射模块620。在块地址映射模块620中，若存在该BN2对应的有效BN1X，则输出该BN1X，且由偏移地址转换器622将该BN2中的BN2Y转换为对应的BN1Y，并将所述BN1X和BN1Y合并为BN1放上总线685。此时，选择器694选择读指针631值（即分支指令本身对应的分支点BN1）中的BN1X作为写地址中的第一地址，选择器696选择读指针631值中的BN1Y作为写地址中的第二地址，选择器692选择总线685上的BN1作为写入内容回该分支点中。若不存在该BN2对应的有效BN1X，则由替换模块611产生一个BN1X，在轨道表610（及一级缓存602）中指定一条可用轨道（及对应的一个存储块）。同时，将二级缓存606中从所述BN2对应的外部指令开始直至其所在二级指令块结束的所有外部指令经扫描转换器608转换及审查，提取出对应内部指令的轨迹点信息填充到轨道表610中由所述BN1X指向的行，并将产生的BN1X和BN2X之间的映射关系存储到偏移地址映射模块618中，以及将转换得到的内部指令填充到一级缓存602中由所述BN1X指向的存储块中。需要说明的是，由于从分支目标外部指令开始转换、填充，因此该分支目标外部指令对应的内部指令在其所在一级存储块中必定是第一条指令，即BN1Y

的值为‘0’。这样，所述分支点的分支目标指令就被存储在一级缓存602中，且所述BN2中的BN2X被转换为分支目标内部指令对应的BN1X（由替换模块611产生），与BN1Y（值为‘0’）一同合并为BN1放上总线693。此时，选择器694、696选择读指针631的值（即分支指令本身对应的分支点）作为写地址，选择器692选择总线693上的BN1作为写入内容写回该分支点中。如此，轨道表610输出的轨迹点内容包含的是BN1。之后的操作与上述分支目标为BN1的直接分支指令中的情况相同，在此不再赘述。

[0197] 若所述轨迹点内容中的指令类型显示该指令为间接分支指令（包括有条件、无条件两种情况），则将处理器核601对该分支指令执行时产生的分支目标地址中的块地址送往主动表604匹配。若匹配成功，则可以得到匹配成功项对应的BN2X，并以分支目标地址中的块内偏移量作为BN2Y，并以该BN2X和BN2Y值送往块地址映射模块620匹配，如命中获得相应的BN1值，则之后的操作与上述分支目标为BN1的直接分支指令中的情况相同；若不命中，则之后的操作与上述分支目标为BN2的直接分支指令中的情况相同，在此不再赘述。若匹配不成功，表示该分支目标地址对应的外部指令尚未存储在二级缓存606中，则根据替换算法（如LRU算法）由主动表604分配一个二级存储块的块号BN2X，并将该分支目标地址送往更低层次的存储器取回相应指令块存储到二级缓存606由所述BN2X指向的存储块中。再按之前所述方法，将该外部指令块转换后填充到一级缓存602中并建立相应轨道、记录映射关系，以及将所述BN2被转换为BN1填回该分支点中（在此过程中产生的BN2并不会被填充到轨道表610中，而直接将对应的BN1填充到轨道表610中），使得轨道表610输出的轨迹点内容包含的是BN1。之后的操作与上述分支目标为BN1的直接分支指令中的情况相同，在此不再赘述。

[0198] 若下一次循迹器重新读出含该间接分支目标的表项时，该表项的指令类型是间接分支指令，但是地址类型是BN1，控制器据此认定该间接分支指令此前已访问过，可以用该BN1地址猜测执行，但是经过BN1地址反求出相应的外部指令地址（如：通过该BN1X对应的轨道中存储的BN2X对主动表604寻址读出外部指令块地址，并通过618转换得到外部指令块内地址，从而得到完整的外部指令地址），待处理器核601执行该间接分支指令产生分支目标地址时将该分支目标地址与反求出的外部指令地址比较。如果相同，则继续执行。如果不相同，则清空分支点后的指令，不保存其结果，从处理器核601提供的分支目标地址开始执行并将该地址如前例映射成BN1后存入该分支点。

[0199] 回到图6，扫描转换器608负责将外部指令转换成内部指令填充到一级缓存。过程中扫描转换器608也计算外部指令的分支目标地址，提取指令的类型并将目标地址与类型信息填充到与一级缓存内部指令填充的相应轨道表表项。请参考图7B，其为本发明所述扫描转换器的一个实施例。

[0200] 在本实施例中，扫描转换器608接受来自两个来源的输入。第一个来源是当轨道表610经总线633送出一个直接分支外部指令地址BN2，此BN2在块地址映射模块620中匹配未命中，此时所需的外部指令块已经存储在二级缓存606中，主动表604中也有相应的外部指令(PC)高位地址，但尚未被转换成内部指令存储在一级缓存602中。总线633上的BN2X地址被送往主动表604中读出相应的PC高位，经总线675送往扫描转换器608，总线633上的块内偏移量BN2Y也被送到扫描转换器608中。此时，选择器660也选择将总线633上的BN2X放上总线673向二级缓存606提供块地址。

[0201] 第二个来源是当轨道表610经总线633送出一个间接分支外部指令类型且其地址格式为外部指令地址格式，表示该间接分支指令的目标需由处理器核601计算。此时，控制器将处理器核601执行相应间接条件分支指令时得到的外部分支目标地址经总线683、选择器680、总线681送往主动表604匹配。如果不匹配，表示分支目标的外部指令块尚不在二级缓存606中，此时主动表将总线681上外部指令地址送往低层存储器读取相应指令块并填充到二级缓存606中由主动表604分配，经选择器660、总线673指向的二级缓存606中的二级缓存块。同时，该外部指令的高位被存入主动表中的对应标签域。如果匹配，主动表604经选择器660和总线673指向与匹配标签相应的二级缓存606中的二级缓存块。同时，总线683上的PC地址被送进扫描转换器608。

[0202] 请参照图7B的扫描转换器608内部结构。扫描转换器608中包含转换器200，直接分支目标地址计算器792，块内偏移映射生成器796，控制器790与输入选择器798，799。其中，控制器790接受各模块来的状态信号并控制各模块协同工作。

[0203] 选择器798选择来自总线675或总线683的PC高位地址存入寄存器788。选择器799选择来自总线633或683的PC低位地址(BN2Y)存入寄存器321。其中，来自总线675及总线633的地址用于将轨道表中的BN2转换成BN1地址，过程中将相应的外部指令翻译成内部指令并存储到一级缓存602。而来自总线683的地址是用于将间接分支目标相应的外部指令翻译成内部指令并存储到一级缓存器并将该一级缓存器块号BN1X连同块内偏移量BN1Y存储到轨道表610中间接分支指令相应的表项。不管来自哪个来源，选择器798、799选择之后，其操作是相同的。以下以BN2转换成BN1地址为例。

[0204] 二级缓存606的地址为BN2，此例中其格式为‘8XYY’。其中‘8X’为块地址BN2X，其值为‘80’～‘82’。二级缓存606中每个二级缓存块（图中一行）有32个字节，其块内偏移量BN2Y为其块内字节(byte)地址‘YY’，其值为‘0’～‘31’，字节中储存变长的外部指令。一级缓存602地址为BN1，其格式为‘7XY’，其中‘7X’为块地址BN1X，其值为‘70’～‘75’。一级缓存602中的每个一级指令块（图中一行）有4条定长内部指令，其块内偏移量BNY1为其块内字(word)地址‘Y’，为易于理解及与BN2Y区分，其值在此实施例中以字母为A～D标注；在此实施例中一条内部指令的长是一个字(word)，内部指令也可以有其他的长度。轨道表610中每行有A～E五个表项，其中A～D四个表项对应于一级缓存602中A～D四条内部指令，表项E用于存放该行顺序下一个一级缓存块的地址。

[0205] 直接分支目标地址计算器792中有三输入加法器760用以计算直接分支目标地址。直接分支目标地址计算器792中还有一个边界比较器772，其输入与总线679相连。边界比较器772中存储了一个二级缓存块中的最大地址（此实施例中为‘31’），总线679上的BN2Y值越过二级缓存块的边界时（大于‘31’），边界比较器772会产生一个二级缓存地址越界信号通知控制器790。直接分支目标地址计算器792还有一个选择器774，控制器790可以控制该选择器选择转换器200输出的分支偏移量或者全‘0’，送往加法器760。选择全‘0’时，计算顺序下一外部指令块地址。

[0206] 请参照图6，设循迹器614指向轨道表中某表项并从该表项中读出其类型为直接分支指令，其分支目标为BN2地址‘8024’，其意义为二级缓存606中第‘80’号二级缓存块中块内偏移量为‘24’的外部指令。该BN2地址经总线633送往块地址映射模块620匹配。其BN2X值经选择器640选择后经总线639选择块地址映射模块620中块地址存储模块920

存储的‘80’行表项内容中的BN2Y，与总线633上经选择器638选择后经总线637送进的BN2Y比较。比较结果为不命中，即该分支指令是存储于二级缓存器的外部指令，但尚未转换成内部指令存入一级缓存602。控制器接收到该不命中信号，即控制以总线633上的BN2X在主动表604中第‘80’行读取其中标签（假设为‘9132’）经总线675送至扫描转换模块608。请参考图7B，控制器也控制608中选择器798选择总线675，选择器799选择总线633，也通知扫描转换器608中控制器790开始转换指令。

[0207] 控制器790控制寄存器756存入选择器798的输出（‘9132’），也控制寄存器321存入选择器799的输出（‘24’，二进制为‘1100’）。即该分支目标的PC地址为‘913224’，存储在二级缓存中第‘80’行，故其BN2地址为‘8024’。假设二级缓存606一次读取16个字节，寄存器321上的4位块内偏移地址仅有最高位被从总线679送往二级缓存606，与来自总线673的块地址合成地址‘8016’，从二级缓存606中读取相应字节经总线677送进转换器200中的对齐器203。此时对齐器203输入的最低字节为字节‘16’，转换器200即以寄存器321上的低3位二进制‘100’作为原始移位量将第‘24’字节移到对齐器203输出的最低字节，开始指令转换。转换器200中的存储器201中对应每条指令会给出一个信号786控制块内偏移映射生成器796记录相应指令的块内偏移量。存储器201另外送出总线788用以控制796中逻辑门780与764以禁止某些块内偏移量的记录，以实现多条内部或外部指令与一条内部或外部指令对应时的映射。

[0208] 寄存器321上的二进制值‘1100’经总线679被送到译码器762译成独热码(one-hot-code)‘00000000000000000000000100000000’，经与或门764存储进存储器766。相应地，计数器776在一段外部指令开始被转换时，被置位为‘0’，其输出总线669上的值‘000’也被译码器778译成独热码‘1000’经逻辑门780送往存储器782存储。块内偏移映射生成器796中还有移位器768及寄存器770，总线679上的值在一个外部指令段开始转换时被存入寄存器770，以控制移位器768的移位。在此例中，‘1100’被存入寄存器770控制移位器768左移24位，使寄存器766中对应于字节‘24’的信息被移位到字节‘0’的位置放上总线691。

[0209] 替换模块611给正在转换生成的内部指令按置换规则分配了一级缓存602中的‘72’号一级缓存块。控制器控制选择器692选择总线693上的BN1X地址‘72’连同BN1Y地址A(‘00’)写入轨道表610中。此时选择器694、696选择总线631上地址，所以BN1地址‘72A’被写入某表项取代原来的BN2地址‘8026’，但不改变原来的指令类型。若循迹器614在该某表项处根据指令类型和/或处理器核601的控制信号决定分支，则会以该‘72A’放上总线631指向轨道表610中‘72’行第一个表项继续执行。

[0210] 替换模块611通过总线693送出BN1X地址‘72’选择602中的‘72’号一级缓存块，也选择了轨道表610中与偏移地址映射模块618中的第‘72’行供扫描转换器608产生的内部指令及相应程序流，块内偏移信息填充。总线669被送出扫描转换器608，送到一级缓存602及轨道表610作为一级缓存块的块内偏移地址BN1Y供填充一级缓存块及相应轨道表使用。位于二级缓存606中从BN2地址为‘8024’开始的分支指令经转换器200转换生成了一条非分支内部指令，从总线667送往一级缓存器602填充进‘72’号一级缓存块的A项（块内偏移为‘00’），其相应的指令类型（非分支指令）也由存储器201输出经总线687送往轨道表610存储在‘72A’项。

[0211] 控制器也控制将总线 633 上的 BN2Y 值‘24’经选择器 698 选择后与总线 693 上的 BN1X 地址‘72’拼接成 BN1X,BN2Y 的形式‘7224’经总线 665 写入块地址映射模块 620 中的块地址存储模块 920 中由总线 633 上的 BN2X 经选择器 640 选择后由总线 639 寻址的‘80’行中最左面的表项。该表项由总线 633 上的 BN2Y ‘24’值经选择器 638 选择后经总线 637 送入块地址映射模块 620 中与该行中各表项的 BN2Y 值‘32’比较决定而确定。该值及其位置表示二级缓存器中‘80’号二级缓存块第‘24’字节开始的外部指令段被存于‘72’号一级缓存块中,且二级缓存器‘80’行中字节地址小于‘24’的外部指令还未被转换成内部指令。具体结构与操作见图 8 实施例。

[0212] 转换器 201 在转换过程中检测到上述外部非分支指令的长度为 2 个字节,经总线 325 控制对齐器 203 将经总线 677 输入的外部指令继续左移 2 位开始指令转换。此字节长度也被送往加法器 323 与寄存器 321 的内容相加,其和‘26’再次存入寄存器 321。寄存器 321 的输出再次由译码器 762 译为独热码‘00000000000000000000000000100000’,并与寄存器 766 中的内容经与或门 764 进行按位‘或’操作,其结果‘00000000000000000000000010100000’再次被存入寄存器 766,其意义为‘80’号二级缓存块中第‘24’字节与‘26’字节各是一条外部指令的起始字节。

[0213] 转换器 200 转换开始于‘26’字节的外部指令,转换过程中发现该指令为一条 4 个字节长的直接分支指令,转换器对其分支偏移量不作任何修改与转换得到的内部指令的其他部分一并直接放上总线 667。其分支指令类型也如前例由总线 687 输出。计数器 776 也在总线 786 的控制下增‘1’,总线 669 值为‘001’。控制器 790 根据该指令为分支指令控制加法器 760 将存储器 756 中的 PC 高位地址与寄存器 321 中的块内偏移量及从总线 667 中对应分支偏移量的部分 798(假设此时该值为‘24’)相加,其和 (sum) 即为分支目标的 PC 地址‘913316’放上总线 657 输出。该和中的低位 BN2Y(不大于二级缓存块内字节数的部分)被拼接到总线 687 上输出。

[0214] 总线 657 上的 PC 地址的高位经选择器 680,总线 681 被送往主动表 604 匹配,其结果为不命中。主动表 604 即将该‘9133’PC 高位地址经总线 681 送往低层存储器读取相应的外部指令块。主动表 604 也分配二级缓存器中‘81’号二级缓存块供此外部指令块存放。二级缓存块号 BN2X(‘81’)也经总线 671 送出与总线 687 上的低位 BN2Y(‘18’)拼接成完整的 BN2 与在总线 687 上的直接分支指令类型,经选择器 692 送往轨道表 610,写入由替换模块 611 经总线 693 指向的‘72’行中由总线 669 指向的 B 项(地址‘001’)。同时,转换所得的内部分支指令经由总线 667 被写进一级缓存器 602 中‘72B’项。

[0215] 总线 669 上的值‘001’也被译码器 778 译为独热码‘0100’与寄存器 782 中的值作‘或’操作值‘1100’存回寄存器 782,代表内部指令块中第一条和第二条都各自对应一条外部指令。如果一条内部指令对应的不是一条外部指令的起始字节(即一条外部指令转换成多条内部指令时相应的第一条内部指令后的内部指令),则存储器 201 的内容通过总线 788 送出的信号(如图 5D 中的结束值 YZ 为‘10’的情况)会控制与或门 780,使得寄存器 782 中的信号与全‘0’进行‘或’操作,使寄存器 782 中相应该指令的位被记录成‘0’,表示该内部指令不对应一条外部指令,使得该指令不会成为一个分支目标。另一方面,当有多条外部指令被融合成一条内部指令时,存储器 201 的内容通过总线 788(如图 5E 中的结束值 YZ 为‘01’的情况)送出的信号会控制与或门 764 ‘擦去’该多条外部指令第一条指令后的

其他指令的相应记录，使得外部指令和内部指令的条数能够一致。当一段外部指令转换成相应的内部指令后，寄存器 782 与 766 中‘1’的数目是一样的，虽然所处的位置不相同。在寄存器 766 中‘1’的位置是代表外部指令起始字节的字节地址。在寄存器 782 中‘1’的位置是代表内部指令起始指令的指令地址。

[0216] 存储器 201 在转换过程中检测到上述始于 26 字节的外部指令的长度为 4 个字节，经总线 325 控制对齐器 203 将经总线 677 输入的外部指令再左移 4 位开始指令转换。此字节长度也被送往加法器 323 与寄存器 321 的内容相加，其和‘30’再次存入寄存器 321。寄存器 321 的输出再次由译码器 762 译为独热码，并与寄存器 766 中存储的内容进行按位‘或’操作，得到的结果‘00000000000000000000000010100010’再次被存入寄存器 766。计数器 776 也依前例增‘1’，使总线 669 指向 C 项。

[0217] 转换器 200 在转换过程中从存储器 201 中经总线 325 读出上述始于‘30’字节的外部指令的长度为 4 个字节，此字节长度也被送往加法器 323 与寄存器 321 的内容相加，其和‘34’再次存入寄存器 321。寄存器 321 的输出 679 与比较器 772 中存储的二级缓存块字节数‘31’比较，此时根据比较结果通知控制器 790 已越过二级缓存块边界。控制器 790 据此控制选择器 774 选择全‘0’，也控制加法器 760 将存储器 756 中的 PC 高位地址与寄存器 321 中的块内偏移量及从总线 667 中送来的全‘0’相加以求顺序下一个外部指令块地址。其结果 PC 地址‘913302’由总线 657 送出，其中 PC 地址中高位‘9133’被送往主动表 604 匹配，得到 BN2X 值‘81’（之前因 PC 地址‘913326’匹配未命中，由主动表 604 分配）。该 BN2X 值经选择器 660、总线 673 选择二级缓存 606 中‘81’号二级缓存块，依前例转换器 200 依前例读取‘81’号二级缓存块中‘0’～‘15’字节进转换器 200，从中提取‘0’～‘1’字节移位拼接到已在转换器 200 中的‘80’号二级缓存块‘30’～‘31’字节后完成该外部指令的转换。转换得到的内部指令从总线 667 送入一级缓存 602 中‘72C’项存储。寄存器 782 中的内容也依前例更新为‘1110’。

[0218] 因在转换指令时已越过二级缓存块边界。控制器 790 此时据此控制转换器 200 停止转换指令，也控制计数器 776 再增一位，使总线 669 上的地址指向‘72D’项。控制器也使总线 671 上的 BN2X 值‘81’，经选择器 640 及总线 639 送至块地址映射模块 620 中块地址存储模块 920，选择其中‘81’行的内容读出与经总线 657、选择器 638、总线 637 送进块地址映射模块 620 的 BN2Y 地址‘02’进行比较。如果匹配命中则将匹配所得的 BN1 连同控制器产生的无条件分支指令类型经总线 685，选择器 692 存进轨道表 610 中‘72D’表项。现在匹配结果为不命中，其意义为相应的外部指令块已在二级缓存中，但尚未被转换为内部指令。此时控制器 790 产生一个直接分支指令类型放上总线 687 与来自加法器 760 的低位 BN2Y（对应于块内偏移字节数）‘02’一同由总线 687 输出。控制器使总线 671 上的 BN2X 值与已在总线 687 上的 BN2Y 地址拼合成 BN2 地址‘8102’，连同无条件分支指令类型一起经选择器 692 写入轨道表‘72D’项。此时，并无相应的内部指令，所以一级缓存 602 中的‘72D’项没有被填充。

[0219] 此时，控制器 790 也控制将寄存器 766 中内容经移位器 768 左移 24 位后，其值为‘10100010’，此格式即图 8B 中行 751 的数据格式。放上总线 691；控制器 790 也控制将寄存器 782 中内容‘1110’放上总线 691。寄存器 782 中的格式即如图 8B 中行 771 的数据格式。总线 691 上的内容被送往偏移地址映射模块 618 中由一级缓存置换器 611 指向的第

‘72’行写入，以供以后对该行进行外部与内部指令的块内偏移映射时用。

[0220] 至此，扫描转换器 608 协同其他模块完成了对一段外部指令的转换，提取该段指令中的程序流 (program flow) 信息，并将程序流信息与转换得到的内部指令存入轨道表 610 及一级缓存 602 中的相应表项。使得本实施例可以经由循迹器 614 读取并遵循轨道表 610 中程序流将相应内部指令供给寄存器核执行。此时块地址映射模块 620 与轨道表 610 中的数值可参考图 9A。

[0221] 在一段外部指令的转换过程中，也有可能一级缓存块先于二级指令段被填满。计数器 776 中也有相当于边界比较器 772 的比较器，在越过一级缓存块的边界的情况下通知控制器 790。控制器 790 在此情况下向一级缓存块置换器 611 请求一个新的一级缓存块并控制将此新缓存块的 BN1X 地址连同为‘0’的 BN1Y 地址经总线 693 和选择器 692 写入轨道表中填充满了的行的最后一项中。轨道表中每行都比相应的一级缓存块中多一项，以在一级缓存块写满的情况下程序流可以延续到下一新增轨道。因为新增的一级缓存块是从第一项开始填充，因此其 BN1Y 地址固定为‘00’。此后，计数器 776 被重置。替换模块 611 通过总线 693 指向新的一级缓存块及相应轨道表中的行。之后转换出的内部指令及相应程序流信息就从总线 963 指向的缓存块及轨道表行的 A 表项开始填充。

[0222] 请参考图 8A，其为本发明所述外部指令块与内部指令块对应关系的示意图。在本发明中，外部指令集可以是定长指令集，也可以是变长指令集。为了不失一般性，在本说明书中主要以变长的外部指令集为例进行说明，定长外部指令集可以作为变长外部指令集的一种特例。

[0223] 在本实施例中，假设一个外部指令块的长度为 16 个字节（从字节 0 到字节 15），且每条内部指令的长度为 4 个字节。如图 8A 所示，外部指令块 701 中包含了 6 条变长指令。如之前实施例所述，外部指令块中的字节 0 是上一条指令的最后一个字节，因此属于上一外部指令块，即本外部指令块中的外部指令从指令块的字节 1 开始。其中，外部指令 703 占 3 个字节（字节 1、2 和 3），外部指令 705 占 5 个字节（字节 4、5、6、7 和 8），外部指令 707 占 2 个字节（字节 9 和 10），外部指令 709 占 1 个字节（字节 11），外部指令 711 占 3 个字节（字节 12、13 和 14），外部指令 713 在本外部指令块中占 1 个字节，其余部分在下一外部指令块中。

[0224] 在本实施例中，假设外部指令 705 可以被转换为 2 个内部指令（即内部指令 725 和 727），外部指令 703、707、709、711 和 713 均可以被转换为 1 个内部指令，分别为内部指令 723、729、731、733 和 735，则经扫描转换器 608 转换后得到的内部指令块 721 中包含了 7 条内部指令（从内部指令 0 到内部指令 7）。此外，在扫描转换器 608 进行指令块转换的同时，也产生了外部指令块内偏移地址 BN2Y 和内部指令块内偏移地址 BN1Y 的对应关系。该对应关系被存储在偏移地址映射模块 618 中。

[0225] 需要说明的是，在本发明中，一条外部指令可能被转换为一条或多条内部指令。为了不失一般性，在本说明书中主要以一条外部指令对应多条内部指令为例进行说明，而一条外部指令对应一条内部指令的情况是一种特例。即，当一条外部指令对应一条内部指令时，该外部指令对应的第一条内部指令和最后一条内部指令均是所述外部指令对应那一条内部指令。

[0226] 请参考图 8B，其为本发明所述偏移地址映射关系存储形式的一个实施例。在本实

施例中，行 751 和 771 构成一组映射关系，分别对应外部指令块和内部指令块，以存储图 8A 实施例中的外部指令和内部指令之间的偏移地址映射关系。其中，行 751 有 16 个表项，每个表项内只存储一位 (bit) 数据（即‘0’或‘1’），其中‘0’表示该表项对应的外部指令偏移地址不是一条外部指令的起始位置，‘1’表示该表项对应的外部指令偏移地址是一条外部指令的起始位置。

[0227] 每组映射关系中的第二行（即行 771）中的每个表项对应一个内部指令偏移地址，即表项数目与内部指令块最大可能包含的内部指令个数相同。且每个表项内也只存储一位数据（即‘0’或‘1’），其中‘0’表示该表项对应的内部指令不是其相应外部指令的第一条内部指令，‘1’表示该表项对应的内部指令是其相应外部指令的第一条内部指令。

[0228] 这样，通过分别对行 751 和 771 中的‘1’进行操作就可以将外部指令偏移地址转换为内部指令偏移地址。请参考图 8C，其为本发明所述偏移地址转换器 622 的一个实施例。在本实施例中，以外部指令偏移地址转换为内部指令偏移地址为例进行说明。其中，从偏移地址映射模块 618 送来的映射关系格式如图 8B 实施例所述。

[0229] 选择器阵列 801 中选择器的列数与外部指令块包含的偏移地址个数相等而行数为列数加一，即 17 行和 16 列。为了清晰起见，在图 8C 中只显示了 4 行、3 列，分别为自左向右的最初 4 行和自下向上的最初 3 列。行号以最下一行为第 0 行，以上各行的行号依次递增。列号以最左面一列为 0 列，其右方各列的列号依次递增，每列对应一个外部指令中的偏移地址。第 0 列各选择器的输入 A 和 B 均为‘0’，除了第 0 行选择器的 A 输入为‘1’。第 0 行所有选择器的输入 B 均为‘0’。其他列选择器的输入 A 来源于前一列同一行选择器的输出值，输入 B 来源于前一列下一行选择器的输出值。

[0230] 选择器阵列 803 的结构与选择器阵列 801 类似，具有相同的行数。不同之处在于选择器阵列 803 中选择器的列数与内部指令块包含的指令条数相等。同样地，为了清晰起见，在图 8C 中只显示了 4 行、5 列，分别为自左向右的最初 4 行和自下向上的最初 5 列。行号与列号的设置与 801 相同。此外，选择器阵列 803 中的第 0 行所有选择器的输入 B 均为‘0’；最后一行（16 行）所有选择器的输入 A 均为‘0’，且第 0 行各选择器的输出均被送到编码器 809 按输出的列的位置编码。其他选择器的输入 A 来源于前一列上一行选择器的输出值，输入 B 来源于前一列同一行选择器的输出值；且第 0 列的输入 A 来源于选择器阵列 801 上一行选择器的输出值，输入 B 来源于选择器阵列 801 同一行选择器的输出值。

[0231] 译码器 805 对外部指令偏移地址进行译码，得到的掩码值送往掩码器 807。由于一个外部指令块包含 16 个偏移地址，因此该掩码值的宽度为 16 位，其中该外部指令偏移地址对应的掩码位及其之前的掩码位的值均为‘1’，该外部指令偏移地址对应的掩码位之后的掩码位的值均为‘0’。之后，将该掩码值与从偏移地址映射模块 618 送来的映射关系中的行 751 进行按位与操作，从而保留行 751 中该外部指令偏移地址对应的掩码位及其之前掩码位对应的值，并将其余值清零，得到一个 16 位的控制字送往选择器阵列 801。

[0232] 该控制字的每一位控制选择器阵列 801 中的一列选择器。当该位为‘1’时，相应列的选择器全部选择输入 B；当该位为‘0’时，相应列的选择器全部选择输入 A。即，对于选择器阵列 801 中的每一列选择器，若对应的控制位为‘1’，则选择来源于前一列下一行的输出值作为输入，使得前一列的输出值整体上移一行，并在最下一行补‘0’，作为本列的输出；若对应的控制位为‘0’，则选择来源于前一列同一行的输出值作为输入，保持前一列的输出

值作为本列的输出。这样，控制字中有多少个‘1’，选择器阵列 801 第一列的输入就会被上移多少行，即选择器阵列 801 的输入中的唯一一个‘1’被上移了相应行数。由于选择器阵列 801 的行数和列数与外部指令块包含的偏移地址个数相等，因此选择器阵列 801 的输出中包含且仅包含一个‘1’，且这个‘1’所在的行的位置由控制字确定。

[0233] 同时，从偏移地址映射模块 618 送来的映射关系中的行 771 直接作为控制字被送往选择器阵列 803。与选择器阵列 801 中类似，该控制字的每一位控制选择器阵列 803 中的一列选择器。当该位为‘1’时，相应列的选择器全部选择输入 A；当该位为‘0’时，相应列的选择器全部选择输入 B。即，对于选择器阵列 803 中的每一列选择器，若对应的控制位为‘1’，则选择来源于前一列上一行的输出值作为输入，使得前一列的输出值整体下移一行，并在最上一行补‘0’，作为本列的输出；若对应的控制位为‘0’，则选择来源于前一列同一行的输出值作为输入，保持前一列的输出值作为本列的输出。这样，每经过控制字中的一个‘1’，选择器阵列 803 的输入就会被下移一行，即所述输入中的唯一一个‘1’被下移了一行。因此，当编码器 809 接收到从选择器阵列 803 最下一行送来的‘1’时，即可根据该‘1’所在的列的位置生成对应的内部指令偏移地址。

[0234] 以图 8B 中的映射关系为例，若外部指令偏移地址为‘9’（即对应外部指令块中的第十个字节也即第三条指令），则掩码器 807 输出的掩码值为‘1111111111000000’，与行 751 中的值‘0100100001011001’进行按位与操作后得到‘0100100001000000’，即控制字中有三个‘1’。这样，选择器阵列 801 的输入中的‘1’被上移三行，即输出的‘1’位于的第 3 行。因此，所述‘1’在选择器阵列 803 中经 3 个值为‘1’的控制位对应的选择器列之后到达编码器 809，因为行 771 中的值为 1101111，使得选择器阵列 803 在第 0，第 1 及第 3 列对输入的‘1’各降一行，最后在第 3 列向编码器 809 输出的值为‘1’，对应内部指令块中的第四条指令（偏移地址为‘3’）。编码器 809 按此编码得到‘3’，从而将外部指令偏移地址值‘4’转换为内部指令偏移地址值‘3’。

[0235] 根据本发明技术方案，可以通过将待排序的 BN2Y 值与块地址映射模块 620 各表项中存储的 BN2Y 值比较以将当前被写入的 BN1X 和 BN2Y 存储到正确位置。请参考图 8D，其为本发明所述块地址映射模块的一个实施例。

[0236] 在本实施例中，块地址映射模块 620 包含块地址存储模块 920、比较模块 924、移位器 926、多路选择器 940、多路选择器 942 及一些选择器控制逻辑。各个功能模块又分成基本相同的复数个列（如：R、S 和 T）。其每列中各有其自有的块地址存储模块 920、比较模块 924、移位器 926、多路选择器 940 及多路选择器 942。其中块地址存储模块 920 为一个由复数个表项组织成复数行与复数列组成（如图 8D 中存储模块 970、971 及 972）的存储器阵列。其每个表项中有两部分：一级缓存块号（BN1X）及二级缓存块内位移（BN2Y）。存储器阵列由地址 639 选出其中一行由总线 950 输出；也同样由总线 639 选出一行将总线 952 上的数据写入该行。块地址存储模块 920 中某列被排序功能模块中每一列各有其相应的比较模块 924 用于比较块内偏移 BN2Y。除比较模块 924 外各功能模块与总线的位宽均等于块地址存储模块 920 表项宽度用于传输表项。比较模块 924 是位宽为 BN2Y 的大于比较器，当某列中总线 950 上的 BN2Y 大于从总线 635 上送入的 BN2Y 时，该列比较器输出为‘1’；当总线 950 上的 BN2Y 小于等于总线 635 上的 BN2Y 时，该列比较器输出为‘0’。当比较器输出为‘0’时，选择器 940 选择本列总线 950 上的表项内容放上总线 952。当比较器输出为‘1’时，其右面

一列的选择器选择比较器所在列 950 上数据经移位器 926 移位后的数据放上总线 952。即当比较器输出为‘1’时，控制器将本列 950 上数据右移一列。当某列的比较器输出为‘1’，而其左面一列比较器输出为‘0’时，则该某列选择总线 665 上数据放上总线 952。总线 952 按列将选择器 940 的输出送至块地址存储模块 920。例如：选择器 976 的输出只送回存储模块 970、选择器 977 的输出只送回存储模块 971。当某列的比较器输出为‘0’，而其右面的一列比较器的输出为‘1’时，则控制逻辑选择该列总线 950 上数据放上总线 954 送往轨道表 610 与块内偏移映射器逻辑 618 等。

[0237] 假设每行二级指令块中的最大偏移地址为‘31’（即偏移地址范围为‘0’～‘31’），则当一个二级指令块被写入二级缓存 606 时，其二级块内偏移地址 (BN2Y)982 均被设为‘32’，其意义是本行最大偏移地址加‘1’。现假设总线 639 上的高位 (BN2X) 为‘81’选出 620 中的一行，其中存储模块 970、971 和 972 列中表项内的 BN2Y 均为‘32’。从总线 637 送进的是 BN2Y，此时值为‘18’。其意义为以 BN2 地址‘8118’匹配排序。比较模块 924 比较的结果为比较器输出 973、974 和 975 均为‘1’（输出 973 为‘1’即表示在块地址存储模块 920 中尚无对应总线 637 上的 BN2Y 的有效表项），控制选择器 940 中选择器 977 和 978 选择 C 输入，即移位器 926 的输出放上总线 952；而选择器 976 选择总线 665 上的数据放上总线 952。总线 952 上的数据被写入块地址存储模块 920 中刚才读出的同一行。其结果是存储模块 970 中表项存储了从总线 665 送进的数据，存储模块 971 中表项存储了原来存储模块 970 中表项数据，存储模块 972 中表项存储了原来存储模块 971 中表项数据。图中未显示的右方各列相应的比较器的来自总线 950 的 BN2Y 输入都为‘32’大于‘18’，所以比较结果都为‘1’，各自控制相应列的数据右移。即 BN2Y 值大于新来的数据的 BN2Y 值的都被右移使得包括新数据在内的各表项按 BN2Y 值的升序排列。控制器检测比较模块 924 中最左边的一个比较器的输出 973 以判定输入的 BN2Y 值有无对应的一级缓存块。如比较器输出 973 为‘1’，表示输入的 BN2Y 无对应的一级缓存块。如比较器输出 973 为‘0’，表示输入的 BN2Y 有对应的一级缓存块。

[0238] 假设上述行又被总线 639 上‘81’号地址读出，此时存储模块 970、971 和 972 表项中相应 BN2Y 值为‘18’、‘32’和‘32’，与从总线 637 送来的 BN2Y 值‘27’由比较模块 924 中的相应比较器相比。其结果为比较器输出 973 为‘0’，比较器输出 974 及 975 均为‘1’。比较器输出 973 使得选择器 976 选 A 输入将本列总线 950 上数据放上总线 952；比较器输出 974 使得选择器 978 选 C 输入，即移位器 926 的输出；比较器输出 975 使得其右方一列的选择器选 C 输入，即移位器的输出。而比较器输出 973 为‘0’与比较器输出 974 为‘1’使得选择器 977 选择 B 输入即总线 665 上的数据。写回块地址存储模块 920 之后存储模块 970 中表项数据的 BN2Y 值为‘18’，存储模块 971 中表项数据的 BN2Y 值为‘27’，存储模块 972 中表项数据的 BN2Y 值为‘32’，在其他右方各项中均为‘32’。如此则表项中数据是根据其 BN2Y 值排序，其相应的一级缓存块号也被按二级存储块内偏移排序，使得可以根据一个外部指令的 BN2 地址映射得到相应的内部指令的 BN1 地址。

[0239] 假设一个新的 BN2 地址‘8123’从总线 639 及 637 送入。此时‘81’行被读出，存储模块 970、971 和 972 表项中的 BN2Y 值分别为‘18’、‘27’和‘32’。总线 637 中送入的 BN2Y 值为‘23’。经比较模块 924 比较得到比较器输出 973 为‘0’、输出 974 和 975 均为‘1’。此时选择器 954 的控制上只有信号 979 为‘1’（信号 979 是比较器输出 973 与输出 974 的异

或)，存储模块970中表项上的内容被放上总线954送往块内偏移映射逻辑（包含块内偏移映射模块618，偏移地址转换器622和减法器928)。表项内容中的一级缓存块号BN1X被作为地址从块内偏移映射模块618中读出与该一级缓存块相应行中的映射关系送往偏移地址转换器622。总线637上的BN2Y（二级缓存块内偏移量）由减法器928减去总线954上的BN2Y（其为该二级缓存块内的与该一级缓存器对应的二级子缓存块的起始地址），其差(23-18＝5)即为总线637上的BN2Y在该二级子缓存块的净地址偏移量。偏移地址转换器622根据该偏移量及上述映射关系即可求出相应的一级缓存块内偏移量BN1Y（该对应关系中二级缓存偏移量为字节5处必为‘1’标识一条外部指令的第一个字节开始，由偏移地址转换器622求出与该外部指令对应的内部指令一级缓存偏移量）。由总线954上的BN1X与此BN1Y拼接即获得指向与上述二级缓存地址‘8123’对应的一级缓存地址BN1。该BN1可被放入轨道表611中表项以便循迹器查找。

[0240] 以下结合图6、图8D、图9A～图9F进行说明，其中图9A～9F为图6实施例运行过程的示意图。

[0241] 在图9A～图9F中显示了运行时块地址存储模块920、二级缓存606、偏移地址映射模块618、轨道表610及一级缓存602中的相应内容。其中，块地址存储模块920中每一行与二级缓存606中一个二级缓存块对应，也与主动表604中一个外部指令块地址对应。偏移地址映射模块618与轨道表610的一行对应于一级缓存602中一个一级缓存块。图6中主动表604还负责按置换规则为新取进的外部指令块在二级缓存606中分配二级缓存块，替换模块611负责按置换规则为内部指令在一级缓存602中分配一级缓存块。图中一级缓存器601中的阴影部分表示已填充的内部指令。

[0242] 二级缓存606的寻址地址为BN2，其格式为‘8XYY’。其中‘8X’为块地址BN2X。为便于说明，此例中二级缓存606是一个路组缓存，其块地址即为索引地址(index)，其值为‘80’～‘82’，其相应标签（即块地址）存放在主动表中相同索引地址的行。二级缓存606中每个二级缓存块（图中一行）有32个字节，其块内偏移量BN2Y为其块内字节(byte)地址‘YY’，其值为‘0’～‘31’。其中储存变长的外部指令，图中每个分隔代表一条不同长度的外部指令，在此实施例中外部指令的长度从2个字节到8个字节不等。

[0243] 一级缓存602则是在轨道表610与块地址存储模块920协同控制下的一个全相连缓存，其地址为BN1，其格式为‘7XY’，其中‘7X’为块地址BN1X，其值为‘70’～‘75’。一级缓存602中的每个一级指令块（图中一行）有4条定长内部指令，其块内偏移量BNY1为其块内字(word)地址‘Y’，为易于理解及与BN2Y区分，其值在此实施例中以字母为A～D标注；在此实施例中一条内部指令的长是一个字(word)，内部指令也可以有其他的长度。轨道表610中每行也有A～D四个表项对应于一级缓存602中A～D四条内部指令。轨道表610中每行还有一个E表项，用于存放其下一指令块的地址。轨道表610中的每个表项储存一个类型，循迹器根据类型决定下一步的地址。表项还可以存储一个指针指向该表项所代表的指令的目标地址，其格式既可以为BN2，也可以为BN1。偏移地址映射模块618每行与一个一级缓存块及其相应的轨道表中一行对应。

[0244] 块地址存储模块920中每一行与二级缓存606中一个二级缓存块对应。二级缓存620中每一行中有复数个表项（如：R、S、T、U、V)。每个表项可与一级缓存中的一个一级指令块相对应。块地址存储模块920中各表项内容含有其相应的一级缓存块的块地址BN1X，

及该一级缓存块中的第一个内部指令在二级缓冲块中相应的外部指令在该二级缓存块中的地址 BN2Y。当一个二级缓存块被写入时，其块地址存储模块 920 中相应的行中的 BN2Y 地址全被重置为‘32’，其意义为其顺序下一个二级缓存块中的第一个字节。

[0245] 图 9A 为开始状态，其时二级缓存 606 中二级缓存块‘80’已被填充，而二级缓存块‘81’和‘82’尚未填充。‘80’号块中从字节‘24’开始的外部指令正被扫描转换器 608 转换为内部指令格式经总线 667 向一级缓存 602 中一级缓存块‘72’按顺序填充。‘80’块中字节‘24’～‘25’是一条外部指令，其相应内部指令被填充进‘72’号块 A 项；‘80’块中字节‘26’～‘29’是一条外部指令，其相应内部指令被填充进‘72’号块 B 项；‘80’块中从字节‘30’开始是一条 4 个字节长的外部指令，其相应内部指令将被填充进‘72’号块 C 项。

[0246] 转换格式过程中，扫描转换器 608 发现开始于‘80’号块‘26’字节的外部指令是一条分支指令，并以该缓存块在主动表 604 中存储的缓存块地址加块内偏移量‘26’，再加上分支偏移量计算出其分支目标。该分支目标高位经总线 657 送往主动表 604 中匹配未命中，经主动表 604 按分配新的二级缓存块‘81’号缓存块（即 BN2X 为‘81’）；主动表 604 也向低层存储器送出该分支目标高位以读取相应外部指令块存入‘81’号缓存块。相应扫描转换器 608 中‘81’行的 BN2Y 全被重置为‘32’。该新分配的二级缓存块号由主动表 604 经总线 671 送出，与扫描转换器 608 输出的总线 657 上的分支目标低位（‘18’号字节）在总线 687 上拼接成一个 BN2 地址。扫描转换器 608 也得出与外部指令‘8026’（即‘80’块‘26’字节）相应的内部指令地址为‘72B’（即‘72’号一级存储块中第二个字），于是扫描转换器 608 的地址总线 669 指向轨道表 610 中‘72’行 B 列中表项，写入经总线 687 传来的表项内容。因此，轨道表 610 中‘72B’表项内容为 BN2 地址‘8118’。

[0247] 总线 657 上的分支目标的低位 (BNY2 值‘18’）被选择器 638 选择后放上比较模块 924 的一个输入 637 与来自块地址存储模块 920 的‘81’行（由主动表 604 分配的 BN2X 值‘81’被选择器 640 选择及经总线 639 送达）的各表项内容比较，发现该为‘18’的值小于所有表项内容（即‘18’＜‘32’），因此 BN1X 值‘72’与 BN2Y 值‘18’（地址在第‘18’号字节的分支目标外部指令被写入了‘72’号第一存储块）被写入块地址存储模块 920 中的‘81’行中的 R 项。此时，R 项的值为‘7218’。

[0248] 扫描转换器 608 继续转换格式到二级缓存 606 中‘80’块的字节‘30’，发现该指令长度为 4 个字节，超出本块 2 个字节，于是在本二级缓存块地址加‘30’（块内偏移）加‘4’（指令字节数），产生下一外部指令块地址。该下一缓存块地址也被总线 657 送往主动表 604 匹配，发现该外部指令块已在（或正在从低层存储器读入）‘81’号二级缓存块，扫描转换器 608 即从‘81’号缓存块中读取需要的数据以完成开始于‘80’号二级缓存块字节‘30’的外部指令的转换，并将转换所得的内部指令按顺序填充进一级缓存器‘72’块 C 项。因为这是‘80’号二级存储块上最后一条外部指令，扫描转换器 608 要向轨道表 610 提供顺序下一条指令地址。此时，匹配所得的 BN2X 地址被主动表 604 从总线 671 送出，与总线 657 上的低位 BN2Y(30+4 ＝ 34，抛弃超出 32 个字节宽的部分，得值‘2’）在总线 687 上合成一个 BN2 地址‘8102’。本实施例处理指令流从一个指令块的最后一条指令转移到其顺序下一条指令的方式是将其视为一条无条件分支指令，即把总线 687 上的 BN2 地址作为一个目标地址，放到轨道表中一个指令块最后一道指令（地址‘72C’）之后的表项，且类型设为无条件分支。因此，扫描转换器 608 经总线 661 送出其值为‘72D’的地址，控制轨道表在‘72’

行 D 项写入 BN2 地址‘8102’。

[0249] 循迹器 614 从轨道表‘72’行 A 项开始读取轨道表中内容，因该行中 A 项不是分支指令，循迹器继续往右读取。循迹器 614 从‘72’行 B 项读出‘8118’判断是一个 BN2 地址，即将该地址经总线 631 送往块地址存储模块 920 及二级缓存 606。该 BN2 地址从块地址存储模块 920 中读出其‘81’行的表项内容。控制逻辑发现块地址存储模块 920 中‘81’行所有的一级缓存块号都为无效，据此判断该 BN2 地址的相应外部指令尚未被转换成内部指令，即控制二级缓存 606 顺序读出地址从‘8118’开始的部分‘81’号二级缓存块直到‘8131’（‘81’块最后一个字节）上的外部指令提供给扫描转换器 608 进行格式转换。

[0250] 扫描转换器 608 也因此向替换模块 611 请求一个可被替换的一级指令块号。替换模块 611 遵循一定的规则，比如 LRU 替换算法，以确定可替换的一级存储块，此时按顺序是‘70’、‘71’、‘73’、‘74’、‘75’。因此，按顺序提供‘70’号一级存储块供填充。扫描转换器 608 即据此将从二级缓存 606 中‘8118’开始的外部指令转换成的内部指令按顺序填入一级缓存 602 中‘70’号存储块中 A、B、C、D 各项，并将 BN1 地址‘70A’写入轨道表 610 中‘72B’表项，代替原 BN2 地址‘8118’。这是基于在二级缓存器中‘8118’地址开始的外部指令的相应内部指令被存储在从‘70A’开始的一级缓存块内。请见图 9B。

[0251] 扫描转换器 608 发现‘70’号一级存储块的 D 项被填充后，‘81’号二级存储块中地址为‘8118’～‘8131’的指令尚未被转换完毕，只转换到地址为‘8126’的外部指令。于是向替换模块 611 请求一个可被替换的一级指令块号。替换模块 611 按顺序提供‘71’号一级存储块。于是控制器将替换模块 611 产生的 BNX 值‘73A’连同控制器产生的无条件分支指令类型‘71A’（‘71’号一级缓存块中的第一条指令的地址）依前例写入轨道表 610 中‘70’行中 E 项以供循迹器 614 执行到此时跳转到‘71’号缓存块的第一条指令。扫描转换器 608 也继续转换外部指令并按顺序填入‘71’号一级存储块。扫描转换器 608 也将地址为‘8118’～‘8126’中每条外部指令的第一个字节的块内偏移地址 BN2Y 以及相应内部指令的块内偏移地址 BN1Y 以图 7B 例中的格式存入块内偏移映射器 618 中循迹器指针 631 指向的‘70’行。

[0252] 从总线 657 中送出的 BNY2 值‘27’被送往比较模块 924 与‘81’行各表项比较。结果发现该 BNY2 值大于 R 表项中的 BNY2 值‘18’，但小于 S 表项及其他表项中的 BNY2 值（均为‘32’）。值‘7127’被填入块地址存储模块 920 中‘81’行的 S 表项，原 R 表项值‘7018’不变，原 T、U、V 表项的值均右移一个表项。

[0253] 因为扫描转换器 608 在‘8118’～‘8131’的外部指令中未发现分支指令，所以轨道表 610 中‘70’行中的 A、B、C、D 各项中没有分支目标的记录。扫描转换器 608 发现‘81’行中从‘26’字节开始的外部指令结束于‘31’字节，没有延伸到下一个指令块，而且该外部指令相应的内部指令结束于‘71’号存储块 B 项。因此，将如前例计算，匹配而分配得到的下一外部指令地址‘8200’存入轨道表 610 中‘71’行 C 项。主动表 604 如前例向低层存储器读取‘82’号二级缓存块的相应外部指令块以填充‘82’号二级缓存块。请见图 9C。

[0254] 处理器核执行轨道表中‘72B’项中的分支指令，其判断结果经信号 635 送往循迹器 614。此时，该结果为不分支。循迹器 614 据此移向轨道表中同一行中下一轨迹点‘72C’读出后，发现为非分支指令，移向下一个表项‘72D’。读出后发现是一条目标为‘8102’的无条件分支地址。控制器判断此为 BN2 地址，经总线 633 送出。总线 633 中高位被送到块

地址存储模块 920，读出其中‘81’行各表项内容送入比较模块 924 的一组输入端，而总线 633 中低位（其值为‘02’）经选择器 638 选择后送到比较模块 924 的另一个输入端 637 相比较。比较结果为 637 上的 BNY2 值小于所有表项中的值，控制逻辑据此判断 BN2 地址为‘8102’的外部指令尚未有相应的内部指令存于一级指令块中。控制逻辑控制二级缓存 606 从总线 633 送来的 BN2X 地址‘81’及总线 679 上送来的地址‘00’开始送外部指令到扫描转换器 608 以转换为内部指令。

[0255] 扫描转换器 608 如前例请求并获得‘73’号一级缓存块以顺序填充转换所得的内部指令。同时，因为总线 637 上的 BNY2 地址‘02’小于所有‘81’行中所有表项内容，如前例，值‘7302’（代表 BNY2 为‘02’的外部指令的相应内部指令被放入‘73’号一级指令块）被放入‘81’行 R 表项中，而原‘81’行各表项都各右移一个表项。并且新值被写入的表项（此时为 R 表项）中的 BNY2 值‘18’被送往扫描转换器 608 以通知扫描转换器 608 只需转换到‘18’字节前一个字节，即‘17’字节即可。

[0256] 在转换所得的内部指令被填充入‘73’号一级缓存块的同时，替换模块 611 产生的 BNX 值‘73A’连同控制器产生的无条件分支指令类型被写入轨道表 610 中‘72D’，将其中的 BN2 值‘8102’替换为 BN1 值‘73A’。循迹器 614 读指针 631 此时仍指向‘72D’项，所以在总线 633 上读出了‘73A’的值。控制逻辑判断这是 BN1 值，据此控制一级缓存用 73A’地址读出相应内部指令供处理器核 601 使用。

[0257] 扫描转换器 608 转换到‘81’行第‘9’字节结束的外部指令时，发现第‘73’号一级指令块已填到 D 项，据此请求得到‘74’号一级指令块继续转换并填充从第‘10’字节开始的外部指令。如前例替换模块 611 产生的 BNX 值‘74A’连同控制器产生的无条件分支指令类型填入轨道表 610 中‘73’行 E 项。从总线 657 中送出的 BNY2 值‘10’如前例被送往比较模块 924 与‘81’行各表项比较。结果发现该 BNY2 值大于 R 表项中的 BNY2 值‘02’，但小于 S 表项中的 BNY2 值‘18’及其他表项中的 BNY2 值。依前例，值‘7410’被填入块地址存储模块 920 中‘81’行的 S 表项，原 R 表项值不变，原 T、U、V 表项的值均右移一个表项。

[0258] 扫描转换器 608 继续转换外部指令并填充到一级缓存 602。在字节‘17’结束的外部指令是填充到‘74’号一级缓存块中 B 项。此时，扫描转换器 608 发现已遇到此前比较模块 924 送来的限度‘18 ‘，并以该限度在块地址存储器 920 中 81 行匹配得到‘70’，即以‘70A’即无条件分支指令类型存入轨道表 610 中‘74’行 C 项存储。另一种实施方式可以将 BN2 地址‘8118’存入轨道表 610 中‘74’行 C 项存储留待循迹器将其读出时在映射为。请见图 9D。

[0259] 在上述指令转换与一级缓存 602 填充的同时，循迹器 614 在继续沿‘73’号轨道前行，因为轨道表中‘73B’、‘73C’、‘73D’表项都为非分支指令，循迹器在这些表项处都不停留，从‘73E’表项读出了无条件分支指令目标‘74A’，即转移到‘74’行从 A 项开始前行。循迹器在‘74C’表项读出无条件分支指令目标‘70A’。即转移到‘70’行继续前行，在‘70E’表项读出无条件分支转移指令，目标‘71A’。循迹器 614 即转移到‘71’行继续前行在‘71C’表项读出表项内容为无条件分支指令，目标‘8200’。控制器判断该目标为二级缓存块地址，于是通过总线 631 将该地址送往块地址存储模块 920，匹配发现‘82’号二级缓存块并无有效的一级缓存块。该匹配结果使扫描转换器 608 开始将‘82’号缓存块中所有外部指令转换为内部指令，从替换模块 611 提供的‘75’号一级存储块开始填充进一级缓存器 602。同

时，扫描转换器 608 也将转换时提取的指令类型及计算得到的分支目标同步填充进轨道表 610 中相应表项。控制器也控制将置换模块 911 产生的 BN1 地址‘75A’连同无条件分支指令类型，写入轨道表 610 中循迹器 614 正指向的表项‘71C’。该表项新内容被从轨道表中读出，经总线 631 直接送往一级缓存 602 读出内部指令供处理器核 601 使用。

[0260] 请见图 9E. 循迹器 614 沿‘75’行前行在‘75B’处遇到一条条件分支指令，其目标为‘8116’，该值为‘8116’的 BN2 被送往块地址存储模块 920 匹配，发现其 BN2Y 值‘16’大于‘81’行 S 表项中 BN2Y 值‘10’，但小于 T 表项中 BN2Y 值‘18’。

[0261] 经图 8D 中比较模块 924 比较得到比较器输出 973 和 974 均为‘0’、输出 975 为‘1’。此时选择器 954 的控制上只有信号 981 为‘1’（信号 981 是输出 974 与输出 975 的异或），存储模块 971 中表项上的内容‘7410’被放上总线 954 送往块内偏移映射逻辑（包含块内偏移映射模块 618，偏移地址转换器 622 和减法器 928）。表项内容中的一级缓存块号 BN1X 被作为地址从块内偏移映射模块 618 中读出第 74 行中的映射关系送往偏移地址转换器 622。总线 637 上的 BN2Y（二级缓存块内偏移量）由减法器 928 减去总线 954 上的 BN2Y（其为该二级缓存块内的与该一级缓存器对应的二级子缓存块的起始地址），其差 (16-10 = 6) 即为总线 633 上的 BN2Y 在该二级子缓存块的净地址偏移量。偏移地址转换器 622 根据该净偏移量及上述映射关系即可求出相应的一级缓存块内偏移量 BN1Y。由总线 954 上的 BN1X 与此 BN1Y 拼接即获得指向与上述二级缓存地址‘8116’对应的一级缓存地址 BN1 值‘74B’。该 BN1 值可被放入轨道表 611 中‘75B’表项取代原有的‘8116’以便循迹器 614 根据此 BN1 值及处理器核 601 的反馈控制一级缓存 602 读取指令。扫描转换器 608 继续转换二级缓存器 606 上‘82’行上的外部指令，在填完‘75’号一级缓存块后获分配‘77’号缓存块作为下一顺序缓存块。请参照图 9F。

[0262] 在轨道表中循迹器需用的分支指令地址都由 BN2 转换为 BN1 后，循迹器 614 读出该等值后即可直接无间断（除等待处理器核 601 经总线 635 送来的条件分支决定外）控制一级指令缓存向处理器核 601 提供指令。

[0263] 进一步地，根据本发明技术方案，所述处理器系统不但可以支持对应不同处理器平台的各种外部指令集（二进制码指令集），也可以支持对应虚拟机的字节码指令集，如作为 JAVA™解释器输入的字节码指令。此时，可以采用与之前实施例相同的方法将一条字节码指令转换为一条或多条内部指令供处理器核执行。鉴于字节码指令的特殊性，还可以在转换过程中做一些改进以提高性能。例如，对于一条需要常数进行运算的字节码指令，因为该常数是存储在存储器中的常量池内，因此按之前实施例所述方法会被转换为一条数据读取指令及相应的运算指令。然而，可以在扫描转换器审查发现该字节码指令是读取常数的指令时，提前将该常数从存储器中填充到数据缓存中。这样，当处理器核执行到该字节码指令对应的第一条内部指令（即数据读取指令）时，不会发生因数据读取造成的缓存缺失。

[0264] 更进一步地，还可以在提前从存储器中获取到该常数时，直接将该常数以立即数的形式嵌入到相应的内部指令（即运算指令）中，从而可以省去所述数据读取指令。这样，当处理器核执行到该字节码指令对应的内部指令（即已嵌入该常数的运算指令）时，可以直接进行运算，从而进一步提高了处理器系统的性能。

[0265] 此外，对于字节码指令中的栈运算指令，也可以用本发明所述的方法转换为对应的内部指令供处理器核执行，从而省去将字节码指令翻译为机器码指令的过程。在本发明

中，一次栈运算被转换为一条内部指令，且该类内部指令的操作数不是寄存器堆中的寄存器值，而是操作数栈中位于栈顶的若干个寄存器值。此时，可以对处理器核中现有的寄存器堆增加相应的控制逻辑，使得该寄存器堆能用做栈寄存器。

[0266] 请参考图 10A，其为本发明所述操作数栈的一个实施例。在本实施例中，以一个栈运算最多需要两个操作数并得到一个运算结果为例进行说明。对于其他情况，也可以此类推。

[0267] 在图 10A 中，寄存器堆 1001 同时支持两个读操作和一个写操作。其中，译码器 1003、1005 分别对送来的两个寄存器号译码后分别送往第一读端口和第二读端口，从总线 1013 和 1015 读出对应的寄存器值。译码器 1007 则对将被写入的寄存器的寄存器号进行译码，并送往写端口，使得总线 1017 上的值可以被写入对应的寄存器。寄存器 1011 中存储了栈顶指针值，即该寄存器堆作为操作数栈使用时栈顶指向的寄存器号。寄存器 1011 中的值通过总线 1045 被送到选择器 1053、1055 和 1057，以及减量器 1031、增量器 1041 和控制器 1019。其中，减量器 1031 和增量器 1041 分别对总线 1045 送来的栈顶指针值进行减一和增一的操作，并将相应结果分别通过总线 1043 和 1047 送往选择器 1053、1055 和 1057。由于寄存器堆 1001 的容量有限，在作为操作数栈使用时若容量已满或接近满（即栈顶指针离栈底指针的距离达到一定程度）时，需要将栈底的一部分操作数按顺序存储到外部存储器（或缓存）中，并移动栈底指针，使得这部分寄存器可以容纳新被压入栈中的操作数，从而构成一个类似循环缓冲 (Circular Buffer) 的结构。同样地，当操作数栈已空或接近空（即栈顶指针离栈底指针的距离达到一定程度）时，需要将之前存储到外部存储器（或缓存）中的那部分操作数按逆序填充回操作数栈中，同时移动栈底指针，使得操作数栈能继续提供操作数。在本实施例中，控制器 1019 根据该栈顶指针值，产生一个新的栈底指针值经译码器 1009 译码后控制寄存器堆 1001 将原栈底指针和所述新栈底指针之间的寄存器值存储到外部存储器中，或从外部存储器将相应操作数填充到寄存器堆 1001 中原栈底指针和所述新栈底指针之间的寄存器中。

[0268] 相应地，在内部指令中有一个指令域表示该内部指令是寄存器运算指令还是栈运算指令，该指令域的值通过控制线 1021 被送到选择器 1033、1035 和 1037。当该内部指令是栈运算指令时，选择器 1033、1035 和 1037 均选择输入 A 并分别送往译码器 1003、1005 和 1007；当该内部指令是寄存器运算指令时，选择器 1033、1035 和 1037 均选择输入 B 并分别送往译码器 1003、1005 和 1007。

[0269] 这样，若一条内部指令是寄存器运算指令，则两个源寄存器号和一个目标寄存器号分别通过总线 1023、1025 和 1027 被选择器 1033、1035 和 1037 选择及经译码器 1003、1005 和 1007 译码后对寄存器堆寻址，从而读出及写入相应的寄存器值。该操作与现有技术类似，在此不再赘述。

[0270] 若一条内部指令是栈运算指令，则上述三个存储寄存器号的指令域被用于存储栈顶指针移动信息。例如，对于一条从栈顶取出两个操作数运算并将结果存回栈顶减一的栈运算指令，其中一个操作数对应的寄存器号就是寄存器 1011 中存储的栈顶指针值，另一个操作数对应的寄存器号是该栈顶指针值减一，而运算结果对应的寄存器号也是该栈顶指针值减一。即，将位于栈顶的两个操作数出栈运算后，再将运算结果压回栈顶。此时，选择器 1053 受总线 1023 上的指令域控制，选择输入 D（当前栈顶指针值），从寄存器堆中读出第一

个操作数；选择器 1055 受总线 1025 上的指令域控制，选择输入 H（当前栈顶指针值减一），从寄存器堆中读出第二个操作数；选择器 1057 受总线 1027 上指令域控制，选择输入 K（当前栈顶指针值减一），经译码后选中将被写回的寄存器。同时，选择器 1051 受总线 1029 上指令域控制，选择输入 N（当前栈顶指针值减一）作为新的栈顶指针值写回寄存器 1011，完成栈顶指针更新。

[0271] 又如，对于一条将操作数压入操作数栈的指令，选择器 1057 受总线 1027 上指令域控制，选择输入 I（当前栈顶指针值加一），经译码后选中相应寄存器，从而将操作数写入该寄存器，实现压栈操作。同时，选择器 1051 受总线 1029 上指令域控制，选择输入 L（当前栈顶指针值加一）作为新的栈顶指针值写回寄存器 1011，完成栈顶指针更新。

[0272] 又如，对于一条将操作数从操作数栈出栈的指令，选择器 1053 受总线 1023 上指令域控制，选择输入 D（当前栈顶指针值），经译码后选中相应寄存器读出操作数，实现出栈操作。同时，选择器 1051 受总线 1029 上指令域控制，选择输入 N（当前栈顶指针值减一）作为新的栈顶指针值写回寄存器 1011，完成栈顶指针更新。

[0273] 此外，控制器 1019 中存储了当前栈底指针值，并对从寄存器 1011 送来的当前栈顶指针值进行判断。若栈底指针值与栈顶指针值接近到一定程度，说明操作数栈接近空，若之前有操作数被存储到外部存储器（或缓存），则需要将一定数目的操作数从外部存储器（或缓存）填充到寄存器堆中栈底以外部分，并更新栈底指针值。相应地，若栈底指针值与栈顶指针值远离到一定程度，说明操作数栈接近满，则需要将一定数目的操作数从寄存器中栈底开始部分存储到外部存储器（或缓存）中，并更新栈底指针值。

[0274] 请参考图 10B，其为本发明所述更新栈底的一个实施例。在本实施例中，假设当栈底指针值与栈顶指针值相差‘3’的时表示操作数栈接近空，且每次填入一个操作数。在某一时刻，栈底指针指向寄存器 1073，栈顶指针指向寄存器 1079。执行一个出栈操作后，栈顶指针指向寄存器 1077。此时，栈底指针值与栈顶指针值相差‘3’，则控制器 1019 发出信号从外部存储器（或缓存）取回之前存储出去的最后一个操作数，并将该操作数填充到栈底指针值减一位置的寄存器（即寄存器 1071），同时对栈底指针值减一，使得栈底指针指向寄存器 1071，保持栈中操作数的数目大于‘3’。

[0275] 请参考图 10C，其为本发明所述更新栈底的另一个实施例。在本实施例中，假设当栈底指针值与栈顶指针值相差‘7’的时表示操作数栈接近满，且每次向外部存储器（或缓存）存储一个操作数。在某一时刻，栈底指针指向寄存器 1081，栈顶指针指向寄存器 1091。执行一个入栈操作后，栈顶指针指向寄存器 1093。此时，栈底指针值与栈顶指针值相差‘7’，则控制器 1019 发出信号将栈底指针指向的那个操作数存储到外部存储器（或缓存）中，同时对栈底指针值加一，使得栈底指针指向寄存器 1083，保持栈中操作数的数目小于‘7’。

[0276] 根据本发明技术方案，每次填充或存储多个操作数的方法与图 10B 和图 10C 的实施例中所述类似，在此不再说明。此外，在上述实施例中通过对栈顶指针值和栈底指针值之间的差值做判断以确定操作数栈是否接近空或满。然而，也可以根据栈顶指针值的变化来判断。例如，自上一次调整栈底指针值以来，若栈项指针值累计增加或减少到一定程度，即可进行相应的操作。

[0277] 在图 7A 实施例中，将结束轨迹点视为一个无条件分支点，因此当循迹器读指针

631 指向结束轨迹点之前的那个轨迹点（即指令块中的最后一条指令），且该轨迹点不是分支点，或是分支转移没有发生的分支点时，循迹器读指针 631 继续更新、移动到结束轨迹点，并输出 BN1 送往一级缓存 602。由于结束轨迹点不对应于真实的指令，循迹器读指针 631 要到下一个时钟周期才会更新为下一轨道的第一个轨迹点，因此在本时钟周期内，一级缓存 602 还需要向处理器核 601 输出一条空指令（即不会改变处理器核内部状态的指令，例如 NOP）供执行。在本发明中，可以对送到一级缓存 602 的寻址地址进行判断，一旦发现寻址地址对应结束轨迹点，则不需要访问一级缓存 602，直接输出空指令供处理器核 601 执行。然而，这样做的缺点是使得处理器核 601 多花费一个时钟周期用于执行无用的空指令。因此，可以对图 7A 进行改进，使得循迹器读指针 631 指向结束轨迹点的前一轨迹点时，根据该轨迹点的指令类型及处理器核 601 执行该指令的反馈，在下一时钟周期直接指向分支目标轨迹点或下一轨道的第一个轨迹点。

[0278] 请参考图 11A，其为本发明所述基于轨道表的缓存结构的另一个实施例。本实施例中的处理器核 601、一级缓存 602、扫描转换器 608、二级缓存 606、替换模块 611、偏移地址映射模块 618 和选择器 692、696、694 均与图 7A 实施例相同。不同之处在于，轨道表 610 每次输出两个轨迹点的内容（循迹器读指针 631 指向的轨迹点内容 1182 及其后的一个轨迹点内容 1183），而循迹器中则增加了类型译码器 1152、控制器 1154 和选择器 1116。其中控制器 1154 执行图 7A 中未显示的控制器的类似功能，此处将其显示以便于说明较复杂的功能与操作。

[0279] 在本实施例中，轨道表 610 的读端口在循迹器输出的读指针 631 的寻址下，输出两个相邻轨迹点的内容并放上总线 1117 与总线 1121，控制器 1154 则检测所述总线 1117 上的指令类型，类型译码器 1152 检测所述总线 1121 上的指令类型。在任一时刻，从轨道表 610 中读出两个表项：当前表项 1182 及其顺序下一个（右方）表项 1183。当前表项 1182 中的内容经总线 1117 读出送往选择器 738 的一个输入及控制器 1154。下一表项 1183 则经总线 1121 送出，送往类型译码器 1152 译码，其结果控制选择器 1116。选择器 1116 的一个输入来源于总线 1121，另一个输入来源于读指针 631 中的 BN1X 及增量器 736 送来的增一后的 BN1Y（即读指针 631 中的 BN1Y 值增一）。类型译码器 1152 只对无条件分支指令类型译码，若总线 1121 上的类型为无条件分支指令类型，则控制选择器 1116 选择输出总线 1121 上的内容；若任何其他类型，则选择来源于总线 631 的 BN1X 与增量器 736 输出的增一后的 BN1Y。

[0280] 以下先考虑总线 1121 上的类型（即顺序下一个表项）不是无条件分支指令类型。此时，选择器 1116 选择来自增量器 736 的输出送往选择器 738 的一个输入。

[0281] 如果控制器 1154 译出总线 1117 上（即当前表项 1182 中的内容）的指令类型是非分支指令，控制器 1154 控制选择器 738 选择由选择器 1116 选择的增量器 736 的输出作为寄存器 740 的输入。来自处理器核 601 的控制信号 1111 控制该输入存入寄存器 740，使得循迹器向右移动达到下一个地址（即顺序更大的地址 BNX1 不变，BNY1+ ‘1’）。在本实施例中，控制信号 1111 是处理器核 601 向循迹器提供的反馈信号，此控制信号 1111 在处理器核正常工作时一直为 ‘1’，使循迹器中寄存器 740 每个时钟周期都更新，使读指针 631 指向轨道表中一个新的表项及一级缓存 602 中一条新的指令以供处理器核执行。当处理器核 601 中工作异常，需要停流水线或者不能执行新的指令时，则控制信号 1111 为 ‘0’，使寄存

器 740 停止更新，循迹器及指针 631 保持原来状态不变，一级缓存 602 暂停向处理器核 601 提供新的指令。

[0282] 如果总线 1117 上该内容中的指令类型是无条件分支，则控制器 1154 控制选择器 738 选择总线 1117 上的分支目标地址，使得读指针 631 跳转到由总线 1117 上分支目标地址对应的轨迹点位置。

[0283] 如果总线 1117 上的指令类型是直接有条件分支，则控制器 1154 控制循迹器暂停更新并等待，直到处理器核 601 产生分支转移是否发生的 TAKEN 信号 635。此时寄存器 740 不仅受控制信号 1111 控制，也受处理器核 601 产生的一个表示 Taken 信号 635 是否有效的信号 1161 控制，需要信号 1161 显示 TAKEN 信号 635 有效且控制信号 1111 也有效时，寄存器 740 才更新。如果分支转移没有发生 (TAKEN 信号 635 为 '0')，则选择器 738 选择选择器 1116 的输出，如之前执行非分支指令的方式运行；如果分支转移发生 (TAKEN 信号 113 为 '1')，则选择器 738 选择总线 1117，将其上的分支目标地址存入寄存器 740，指针 631 指向轨道表中分支目标的相应表项，及一级缓存 602 中的分支目标指令，将其读出供处理器核 601 执行。

[0284] 如果总线 1117 上的指令类型是 BN2 分支类型，则控制器 1154 控制循迹器中寄存器 740 暂停更新并等待，按前例将该 BN2 转换获得 BN1 地址，并写回轨道表中的原来间接分支表项。该表项经总线 1117 读出，此后处理与前例相同。循迹器沿该 BN1 并根据处理器核 601 反馈的指令执行结果（如：分支指令的执行结果），控制一级缓存 602 向处理器核 601 输出指令供执行。

[0285] 如果分支转移没有发生，则如之前非分支指令的做法运行，如果分支转移发生，则如之前无条件分支指令的做法运行。

[0286] 如果该内容中的指令类型是间接分支，控制器 1154 控制循迹器中寄存器 740 暂停更新，并等待处理器核 601 经总线 683 送出分支目标地址，并如前例被送往主动表 604、块地址映射模块 620 匹配，以后操作与上例同。

[0287] 如果表项 1183 中是无条件分支指令，则分支类型译码器 1152 对总线 1121 上的指令类型译码，使得选择器 1116 选择总线 1121 上的分支目标而不选择经增量器 736 提供的 BN1（所述 BN1 即 BN1X、BN1Y+ '1')，如此当处理器核 601 执行完表项 1182 相应的指令后，并不执行表项 1183 对应的指令（因为表项 1183 对应的可能是结束轨迹点，在一级缓存 602 中并无指令与其对应），而是直接执行表项 1183 中所含的分支目标地址的相应指令。

[0288] 如果表项 1182 中是一条非分支指令，则如上所述执行完该指令后所执行的下一条指令就是表项 1183 中的分支目标所指向的指令。如果表项 1182 中是一条无条件分支指令，则执行完该指令后所执行的下一条指令就是表项 1182 中的分支目标所指向的指令，表项 1183 对该过程不产生影响。如果表项 1182 中是一条条件分支指令，则执行完该指令后所执行的下一条指令取决于处理器核 601 所产生的 TAKEN 信号 635。如判断为分支转移发生 (TAKEN 信号 635 为 '1')，则选择器 738 选择总线 1117 上的分支目标，表示 TAKEN 信号 635 有效的信号 1161 控制将该目标存入寄存器 740，使指针 631 指向该分支目标，下一条执行的指令就是表项 1182 中分支目标地址所指向的指令。如判断为分支转移不发生 (TAKEN 信号 635 为 '0')，则选择器 738 选择经选择器 1116 输出的总线 1121 上的分支目标，表示 TAKEN 信号 635 有效的信号 1161 与控制信号 1111 控制将来自 1183 的无条件分支目标存入

寄存器 740 使指针 631 指向该分支目标，下一条执行的指令就是表项 1183 中的无条件分支目标地址所指向的指令。

[0289] 结束轨迹点中的无条件分支目标其地址也可以是二级缓存地址 BN2。类型译码器 1152 在译码总线 1121 上读出的表项的指令类型时如果发现该地址为 BN2 格式，也可以将该总线 1121 输出的 BN2 放上总线 1117 按前例转换为 BN1 存回该表项。为了清晰及便于说明起见，这个路径在图 11A 中没有画出。

[0290] 图 11A 例中该条件分支指令的类型判断可以有四种方式。第一种方式为只有一种无条件分支类型，即对程序中原有的无条件分支指令，与本发明所添加的结束轨迹点中控制跳转到下一轨道起始表项的无条件跳转操作不加区分。这种方式会使得程序中原有的条件分支指令被跳过，不被处理器核 601 执行，但是程序流在轨道表 610 与循迹器的控制下，可以正确执行该分支指令的目标指令及其后续指令。这样，节省了原来执行该无条件分支指令所占的时钟周期。但处理器核 601 中因为未执行该指令，程序计数器 PC 值会有误差，如果需要保持精确 PC 值则需进行补偿。本发明中的缓存系统不需要 PC 即能正确向处理器核 601 提供其将要执行的指令供其不间断地执行。如果需要获得某个时刻的 PC 值时（如调试时），每行轨道表中都记载了该一级指令块相应的二级缓存块地址 BN2X 及二级缓存子块地址。由此，BN2X 从主动表 604 中可读出相应的标签，与二级缓存块地址，子块地址及指针 631 中 BNY 的数值拼接，就是正在执行的指令的 PC 值。

[0291] 第二种方式为有两种无条件分支类型。其中，一种为结束点无条件分支类型对应轨道中每条轨道的结束点。对于这种结束点无条件分支类型，类型译码器 1152 将其视为该结束点不对应程序中一条指令，由此控制选择器 1116 选择总线 1121 上的分支目标，在执行完总线 1117 上的指令后直接跳转到总线 1121 上的分支目标地址。另一类对应程序中的无条件分支类型，类型译码器 1152 在译出这种类型时不将其作为分支处理，控制选择器 1116 选择增量器 736 的输出。当执行完总线 1117 上的表项内容的相应指令后，下一条执行的指令为其顺序下一条指令，即程序中原有的无条件分支指令。这种方式下处理器核中的 PC 则一直保持正确的值。

[0292] 第三种方式为对图 11A 实施例进行改进，在扫描转换器 608 对指令块审查的过程中，如果发现一级指令块的倒数第二条指令不是有条件分支指令，且最后一条指令为非分支指令，扫描转换器 608 在这种情况下将结束轨迹点合并到该最后一条指令对应的轨迹点中。即，将该最后一条指令的指令类型标记为无条件分支指令，并将下一指令块第一条指令对应的 BN1 或 BN2（若是 BN2 则循迹器读出时会按前例将其转换为 BN1）作为轨迹点内容存储在该最后一条指令对应的轨迹点中。这样，当循迹器读指针 631 指向该指令对应的轨迹点时，除了从一级缓存 602 中读出该指令供处理器核 601 正常执行外，控制器 1154 将总线 1117 上指令类型译码发现是无条件分支类型，因此控制选择器 738 选择总线 1117，在下一时钟周期将读指针 631 更新为该无条件分支的分支目标 BN1（即下一指令块第一条指令对应的 BN1）。此时，处理器核 601 不需要浪费一个时钟周期执行空指令。

[0293] 在扫描转换器 608 对指令块审查的过程中，如果发现一级指令块的最后一条指令（对应一条轨道中最后一个轨迹点）为分支指令，扫描转换器 608 在这种情况下不将结束轨迹点合并到该指令对应的轨迹点中，而将结束轨迹点的内容放置在每条轨道最后一条指令对应的轨迹点之后（右方）的轨迹点。当该最后一条指令是无条件分支指令时，控制器

1154 按总线 1117 上的无条件分支类型控制选择器 738 选择总线 1117 上的分支目标放上指针 631，跳转至该目标，结束轨迹点不会被执行。当该最后一条指令是条件分支指令时，控制器 1154 按总线 1117 上的条件分支类型控制循迹器暂停，等待处理器核 601 产生的分支判断信号 635。此时类型译码器 1152 译出总线 1121 上的指令类型为无条件分支，控制选择器 1116 选择总线 1121。当分支判断信号 635 为‘分支’，控制器 1154 控制选择器 738 选择总线 1117 上的条件分支目标放上指针 631。当分支判断信号 635 为‘不分支’，控制器 1154 控制选择器 738 选择 1116 选择器的输出，将总线 1121 上的无条件分支目标放上指针 631。一级缓存 602 按指针 631 送出指令供处理器核 601 执行。

[0294] 上述三种方式都既适用于定长的指令或变长的指令。即并不要求结束轨迹点在轨道中的固定位置。此外，若结束轨迹点在轨道中的位置是固定的，则可以根据读指针 631 中 BN1Y 的值判断是否已经到达最后一条指令。第四种方式为轨道表中只有一种无条件分支类型，但循迹器根据该类型在轨道中所处位置将其分为两种类型。此方式中，指针 631 中的 BN1Y 被送进类型译码器 1152 而总线 1121 上的指令类型不需要译码。当所述 BN1Y 指向一条轨道中最后一个表项时，类型译码器 1152 控制选择器 1116 选择总线 1121 上的分支目标，在执行完总线 1117 上的指令后直接跳转到总线 1121 上的分支目标地址。当所述 BN1Y 指向一条轨道中除最后一个表项之外的其他表项时，类型译码器 1152 控制选择器 1116 选择增量器 736 的输出。当执行完总线 1117 上的表项内容的相应指令后，下一条执行的指令为其顺序下一条指令。这种方式下处理器核中的 PC 则一直保持正确的值。这种方式适应定长指令。

[0295] 此外，当从总线 1117 上读出的轨道表 610 表项经控制模块 1154 译码其类型为条件分支指令时，本发明可以控制处理器核 601 沿分支中的一支猜测执行 (speculate execution)，以提高处理器的执行效率。请参见图 11B，其为本发明支持猜测执行的实施例。图 11B 中循迹器中与图 11A 中循迹器相比增添了选择器 1162 及寄存器 1164，用于选择、存储分支猜测执行未选中的另一支暂存，以备猜测错误时使用。猜测执行方向可以由现有的静态预测，或动态分支预测 (branch prediction) 技术决定，也可由存于轨道表中对应分支指令的表项中的分支预测域决定。

[0296] 以猜测不分支为例，控制器 1154 在译出总线 1117 上的一个条件分支类型并获得不分支的预测值时，控制选择器 1162 及寄存器 1164 选择总线 1117 上的分支目标地址存入寄存器 1164。同时控制器 1154 控制选择器 738 选择 1116 选择器的输出（其为分支指令的顺序下一条指令）供存入寄存器 740，使指针 631 控制一级缓存 602 提供分支指令后的顺序下一条指令供处理器核 601 执行，并向处理器核标记这条指令为猜测执行。指针 631 也指向轨道表 610 中分支指令后顺序第一个表项，使其被放上总线 1117。之后控制器 1154 按总线 1117 上的指令类型决定循迹器的后续方向，继续向处理器核提供指令。所有这些指令都被标记为猜测执行。当总线 1161 通知分支判断信号 635 为有效时，控制器 1154 将预测的分支方向与 635 上的分支方向比较。若比较结果相同，则沿原猜测方向继续执行。若比较结果不同，此时控制器 1154 向处理器核 601 发送‘猜测错误’的信号，使处理器核清除所有带猜测执行标记的指令及其中间执行结果。同时控制器 1154 控制选择器 738 选择寄存器 1164 的输出，使分支未被猜测执行的一支的地址被用于控制一级缓存器 602 向处理器核 601 提供指令，并沿此继续执行。

[0297] 若猜测为分支，则控制器 1154 在译出总线 1117 上的一个条件分支类型并获得进行分支的预测值时，控制选择器 1162 及寄存器 1164 选择 1116 选择器的输出（其为分支指令的顺序下一条指令）存入寄存器 1164。同时控制器 1154 控制选择器 738 选择总线 1117 上的分支目标地址供存入寄存器 740，使指针 631 控制一级缓存 602 提供分支指令的分支目标指令供处理器核 601 执行，并向处理器核标记这条指令为猜测执行。指针 631 也指向总线 1117 上的分支目标地址指向的轨道表 610 中表项，使其被放上总线 1117。之后控制器 1154 按总线 1117 上的指令类型决定循迹器的后续方向，继续向处理器核提供指令。所有这些指令都被标记为猜测执行。当总线 1161 通知分支判断信号 635 为有效时，控制器 1154 将预测的分支方向与分支判断信号 635 上的分支方向比较。若比较结果相同，则沿原猜测方向继续执行。若比较结果不同，此时控制器 1154 向处理器核 601 发送‘猜测错误’的信号，使处理器核清除所有带猜测执行标记的指令及其中间执行结果。同时控制器 1154 控制选择器 738 选择寄存器 1164 的输出，使分支未被猜测执行的一支的地址被用于控制一级缓存器 602 向处理器核 601 提供指令，并沿此继续执行。

[0298] 现有的指令集转换技术通常用一个固定指令转换模块（有时将其称为译码器）将一种外部计算机指令集转换为内部指令集（有时称为微操作）后供执行内部指令集的处理器核执行。通常该转换模块位于存储外部指令的缓存和处理器核之间，处理器核提供的外部指令地址寻址缓存读出外部指令经转换模块转换为内部指令后供给处理器核执行。对外部指令的重复转换，不仅大幅增加功耗，而且时延较长的指令转换器在指令执行的关键路径上，需要较深的指令缓冲器 (Instruction Buffer)，大幅加深了处理器核的流水线，从而增加了硬件开销和分支预测失败时的性能损失。当转换模块位于缓存之前时，缓存内存储的是可被处理器核直接执行的内部指令，但因为内部指令（一般是定长指令）和外部指令（可以是变长指令）一般不是一一对应的，因此在分支转移时缺乏可靠地将分支目标指令的外部指令地址（一般由外部指令编译器产生的分支偏移量及外部分支指令地址相加产生，上述两者都以外部指令地址表达）转换为内部指令地址并以此在缓存中寻址正确的内部指令的方法及系统。导致现有的处理器宁愿承受上述因重复转换同一指令导致的功耗、性能、成本等损失，而将指令转换模块置于缓存与处理器核之间，而在一级指令缓存器存储外部指令的原因。虽然使用跟踪缓存、指令循环缓冲器等可以在程序执行路径 (trace) 命中或执行循环代码时避免所述实时地址转换，但跟踪缓存中会同时重复存储位于不同路径上的同一指令，造成很大的容量浪费，导致跟踪缓存的性能不高。这些存储器在某些特定条件下可以用特定的指令地址寻址，但无法让处理器核在任意条件下用指令地址可靠，高效地对存储内部指令的存储器实现如同正常缓存方式的寻址，不可避免地要经常，重复读取外部指令将其经转换器转换为内部指令，或者使用低效的软件方式将外部指令地址翻译成内部指令地址。总之，现有技术缺乏可靠，高效的方法及系统将外部指令地址转换为内部指令地址，是影响虚拟机效率的瓶颈。另外现有的指令转换器都是将固定的一种或少数种特定的外部指令集转换为内部指令集。

[0299] 采用本发明所述的指令集转换系统和方法则可以将转换得到的内部指令存储在缓存中，并由地址映射模块完成对处理器核产生的外部指令地址向内部指令地址的转换，使得处理器核可以直接对已经存储在缓存中的内部指令寻址，而不需要处理器核反复对存储外部指令的缓存寻址，读出外部指令后经指令转换器转换为内部指令后供处理器核执

行，多次反复的对一级缓存中的同一外部指令进行转换，从而避免上述功耗，关键路径上长时延，及额外的硬件开销成本问题。本发明所述的可配置指令转换器可以根据配置将任意种不特定的外部指令集转换为内部指令集。

[0300] 本发明所述的指令集转换系统主要由转换器和地址映射模块两大部分组成。本发明所述的转换器可以是固定转换也可以是可配置的。根据本发明技术方案，当一个处理器核可执行的指令集（即内部指令集）中的指令与任意需要运行的指令集（即外部指令集）中的指令一一对应时，可配置转换器就可以与处理器核共同使用，将外部指令转换为内容指令供所述处理器核执行。此时，外部指令中分支指令对应的分支目标地址与该分支指令对应的内部指令的分支目标地址是相同的，不需要进行外部地址到内部地址的映射。请参考图12，其为本发明所述包含可配置转换器的处理器系统的一个实施例。在本实施例中，外部指令 1205 经可配置转换器 1202 转换后被存储在指令存储器 1203 中，供处理器核 1201 直接执行。在此，指令存储器 1203 中存储的是内部指令，可配置转换器 1202 的功能和结构与图 2 实施例中的转换器 200 类似。由于外部指令和内部指令一一对应，因此外部指令地址与内部指令地址是相同的，当处理器核 1201 执行一条分支指令，如果不执行分支，则以分支指令地址增‘1’作为下条指令的地址送到指令存储器 1203 读取内部指令供处理器核 1201 执行；如执行分支，按外部指令的分支偏移量加上分支指令的地址产生的外部指令分支目标地址，与内部指令分支目标地址相同；因此可以直接使用该外部指令分支目标地址对指令存储器 1203 寻址，从中读出分支目标内部指令。不需要将外部指令地址转换为内部指令地址。当执行非分支指令时，其下条指令的地址产生方式与上述分支指令不执行分支时相同。

[0301] 采用本发明所述的可配置转换器的处理器系统能够根据需要进行配置，从而执行不同的外部指令集。请参考图 13A，其为本发明所述可配置转换器的一个框图实施例。在本实施例中，存储器 201 如图 2 所述存储了内部指令集和外部指令集的转换规则。提取器 1302（即图 3 中操作码提取器 211，213，215）则如前所述从总线 1205 送来的外部指令中抽取出外部指令操作码作为寻址地址经总线 1307 送到存储器 201 寻址读出对应于该外部指令的转换规则，其中的掩膜及移位控制信号经总线 1308 控制移位模块 1303（即图 2 中 221，223，225，227）对外部指令中各个指令域（如操作数的寄存器堆地址）提取，掩膜并移位到内部指令的格式规定的位置；其中的内部指令操作码也经总线 1309 送出，并按规则移位到内部指令格式规定的位置，与上述掩码、移位后的指令在合并模块 1304（与图 2 中 207 相似）中合并为内部指令，经总线 1306 输出。这样，本发明所述可配置转换器就完成了将外部指令转换为内部指令的操作；改变存储器 1301 中的转换规则就可以使指令转换器与执行内部指令的处理器核的组合执行不同的外部指令集。

[0302] 此外，还可以在所述可配置转换器中增加一个寄存器用于存储外部指令是定长（Fix Length）还是变长（Variable Length）的信息。当该寄存器被配置为定长（例如被配置为‘0’）时，则表示外部指令在外部指令块中的边界是对齐的，因此在转换时可以从外部指令块的起始地址开始转换。当该寄存器被配置为变长（例如被配置为‘1’）时，则表示外部指令在外部指令块中的边界不一定对齐，此时只能对目标指令开始直至该外部指令块中最后一条尚未被转换的指令进行转换。

[0303] 进一步，可以在存储器 1301 中存储复数种外部指令集的转换规则，其中每种外部

指令集各有其地址空间，不同的程序线程选择不同的转换规则地址空间。此时在图 2 中控制提取外部指令操作码的寄存器 212，214，216 之外再增设一个寄存器存储该线程对应的指令集转换规则的存储器 201 基地址。另将上述寄存器增设为复数组，每组对应一种外部指令集，由选择器选择。并在处理器的存储器管理器 MMU 中的线程号存储器（一般在 TLB 中）对应每条线程添加一个存储域，存储选择上述复数组寄存器的选择信号。请参考图 13B，其为本发明所述可配置转换器中存储器的一个实施例。例如寄存器组 1311 存储的是 P 指令集的操作码提取位置及其相应指令转换规则在存储器 201 中的基地址‘m’；寄存器组 1311 存储的是 Q 指令集的操作码提取位置及其相应指令转换规则在存储器 201 中的基地址‘n’。

[0304] 当线程 J 的外部指令由指令转换器转换时，MMU 中 J 线程的选择信号 316 控制选择器 315 选择寄存器组 1311 的输出。此时，操作码提取器 1302（即图 3 中操作码提取器 211，213，215）受寄存器组 1311 的控制对被转换的外部指令提取操作码；该操作码与也来自寄存器组 1311 的基地址‘m’被加法器 1318 相加后作为地址对转换规则存储器 201 寻址，控制指令转换器的操作，将 P 指令集指令转换成内部指令存入图 12 中指令存储器 1203。当线程 K 的外部指令由指令转换器转换时，MMU 中 K 线程的存储器的选择信号 316 控制选择器 315 选择寄存器组 1313 的输出。此时，操作码提取器 1302 受寄存器组 1313 的控制对被转换的外部指令提取操作码；该操作码与也来自寄存器组 1313 的基地址‘n’被加法器 1318 相加后作为地址对转换规则存储器 201 寻址，控制指令转换器的操作，将 Q 指令集指令转换成内部指令存入图 12 中指令存储器 1203。如此处理器核在从 J 线程切换到 K 线程时，实际上是从执行 P 指令集指令转换为执行 Q 指令集指令。如此可实现在一个本发明所公开的虚拟机中执行含有复数种外部指令集中的指令的程序。当然用复数个指令转换器，每个负责转换一种外部指令集，也可实现同样的功能。

[0305] 某些计算机指令集中的指令上的复数个域之间是正交的（Othogonal），即这些域之间是独立的，比如有些指令集除操作码域外还用指令中某些域中的编码来代表对特定存储器或寄存器的寻址，这些域也需要由转换规则进行映射，而非对外部指令中的地址进行移位就满足内部指令的要求。此时可以用复数个转换规则存储器及相应逻辑对应复数个正交的指令域，使得转换规则存储器的总表项数（行数）控制在一个合理的数目。请参考图 13C，其为本发明所述可配置转换器中存储器的另一个实施例。与图 13A 相比，图 13C 中增添了一个转换规则存储器 1321 及其专用的提取器 1322（与 1302 同样功能），及移位逻辑 1323（与 1303 同样功能）。另外还新增了如同图 13B 例中的寄存器组 1311 及 1313 的寄存器组（图 13C 中位显示）以控制新增的存储器 1321 及其相应逻辑。新增的逻辑中存储器 1321 及掩膜移位逻辑 1323 的输出都被送到合并其 1304 与原有存储器 201 及掩膜移位逻辑 1303 的输出合并。两套存储器及其相应逻辑可以分工协同处理同一计算机指令集，各负责外部指令上部分域的转换，在合并器 1304 中合并成内部指令。两套存储器及其相应逻辑还可以独立操作，各自独立负责将一种外部指令转换为内部指令，实现如图 13B 的功能。为此可增设一个可写的寄存器，由这个寄存器的状态决定图 13C 的指令转换器是以协作，或独立方式操作。

[0306] 此外图 13A 中合并模块 1304 还要根据内部指令的转换顺序产生与外部指令的映射关系，例如图 8A 或图 8B 所示的例子，以供填写块地址偏移映射器 YMAP 等。合并模块 1304

还产生写地址，控制将内部指令填入指令存储器 1203 等。如果内部指令是定长的则每对指令存储器 1203 写进一条指令，一级缓存写地址加上一个固定长度，如 4 个字节。如果内部指令是变长的则存储器 1301 中对应该指令的转换规则中要记载该指令的长度，每对指令存储器 1203 写进一条指令，一级缓存写地址加上从存储器 1301 输出的该指令的长度，作为下一指令的起始地址。也可以将一个内部指令块的复数条内部指令分次存入一个缓冲器，将整个内部指令块一起写入指令存储器 1203。也可以由其他模块产生上述映射关系及写地址，如在图 7A，图 7B 中由扫描转换器中负责扫描的部分负责。

[0307] 采用本发明所述可配置转换器的处理器系统可以在外部指令集与内部指令集的指令一一对应的情况下工作。然而，当两种指令集的指令不一一对应时，会有一条外部指令被转换为多条内部指令，或多条外部指令被合并为一条内部指令的情况发生；又或者外部指令或内部指令中至少一种为变长指令；从而有可能导致外部指令的分支目标地址与相应内部指令的分支目标地址不一一对应。此时，可以用本发明所述的地址映射模块结合指令转换器实现指令集转换及指令地址的映射。请参考图 14，其为本发明所述包含指令转换器和地址映射模块的处理器系统的一个实施例。在本实施例中，外部指令经转换器 1202 转换后被存储在指令存储器 1203 中，供处理器核 1201 直接执行。即指令存储器 1203 中存储的是内部指令，且指令存储器 1203 根据内部指令地址寻址输出相应的内部指令。在转换过程中，转换器 1202 还产生外部指令与相应内部指令的对应关系存储到地址映射模块 1404 中。当处理器核 1201 按指令顺序执行指令存储器 1203 中的内部指令时，其程序计数器 PC 每次增加‘1’，使得相应的内部指令地址增‘1’，从而对指令存储器 1203 寻址以读出下一条内部指令。当处理器核 1201 执行分支指令产生分支目标地址时，由于该分支目标地址是以外部指令地址形式表示的，因此被送到地址映射模块 1404 按前述方法转换为对应的内部指令地址后再送到指令存储器 1203 寻址以读出相应的内部指令（即分支目标指令）。具体地，若地址映射模块 1404 中已经存储了所述外部指令地址对应的映射关系，则说明该外部指令对应的内部指令已经被存储在指令存储器 1203 中，可以直接将所述外部指令地址转换为内部指令地址输出。若地址映射模块 1404 中尚未存储所述外部指令地址对应的映射关系，则说明该外部指令尚未被转换为内部指令。此时，由转换器 1202 将包含所述外部指令在内的至少一条外部指令转换后存储到指令存储器 1203 中，并将对应的映射关系存储到地址映射模块 1404 中，这样就可以将所述外部指令地址转换为内部指令地址输出。在此，转换器 1202 可以是固定将一种特定外部指令转换为内部指令的转换器，也可以是图 2，图 3，图 4，图 5 及图 13A、B 中公开的可配置指令转换器。

[0308] 根据本发明技术方案，地址映射模块 1404 可以由映射表构成。所述映射表可以由外部指令地址寻址，其表项内存储了相应内部指令的地址。在此基础上，所述映射表可以有多种具体实现方式。

[0309] 方式一：映射表中的每个表项均由外部指令地址的最小单位（例如：字节）寻址，每个表项中都存储了该表项对应的外部指令对应的内部指令所在的内部指令块的块地址（即内部指令块在指令存储器 1203 中的块号），以及内部指令在所述内部指令块中的块内地址偏移地址。这样，在对外部指令地址进行转换时，可以根据所述外部指令地址对映射表的表项寻址，读出相应表项中的内部指令块地址及块内偏移地址，完成地址转换。

[0310] 方式二：当外部指令的长度不固定时，可以对方式一所述映射表进行压缩以消除

空的表项。以外部指令按字节寻址为例，由于外部指令长度不固定，只有每条外部指令起始地址字节才占据一个表项，存储该外部指令的块内偏移以及相应的内部指令块内偏移地址，而其余外部地址非起始地址字节不占据表项。在此，映射表每行对应一个外部指令块，可以由外部指令块地址寻址。这样，在对外部指令地址进行转换时，可以根据所述外部指令的块地址对映射表的行寻址，读出整行内容。之后，用所述外部指令的块内偏移地址对该行所有表项中的外部指令块内偏移地址进行匹配，选择并输出匹配成功项中存储的内部指令地址，完成地址转换。

[0311] 方式三：映射表中的每行由两部分组成，第一部分包含数据的位数与一个外部指令块包含多少个最小地址单位相同（例如，数据位数与外部指令块包含的字节数相同），第二部分包含数据的位数与一个内部指令块可能包含的最多内部指令数相同。第一部分中对应各外部指令起始地址（即起始字节）的数据置为‘1’，其余均为‘0’，而第二部分中对应各外部指令相应第一条内部指令对应的数据置为‘1’，其余均为‘0’，具体格式可以参考图8B。这样，在对外部指令地址进行转换时，可以根据所述外部指令的块地址对映射表的行寻址，读出整行内容（包括两个部分）。之后，根据所述外部指令的块内偏移地址对第一部分直至该块内偏移地址字节对应的数据为止的‘1’进行加‘1’计数，再对计数结果根据第二部分中的‘1’进行减‘1’计数，直到计数结果为‘0’，此时第二部分中的计数位置对应的就是内部指令的块内偏移地址，完成地址转换。如图8C的装置可以高效完成上述映射。

[0312] 进一步，可使外部指令块与内部指令块有固定的对应关系（如一个存储外部指令的二级缓存块可以等分成两个二级缓存子块，其中每个子块对应一个存储内部指令的一级缓存块）。如此可以将外部指令与内部指令的映射操作分解为块地址的映射操作（因为有对应关系，所以易于实现），以及块内偏移地址的映射两个部分来实现以简化映射的难度。如此一级缓存块不一定每个表项含有有效的内部指令。以下以一级指令在一级指令块中从最小块内偏移地址（一般为‘0’）开始增序排列。如此对应每个指令块还需要存储其偏移地址最大的指令的偏移地址以提醒系统下个周期要提供按程序顺序下一条指令的一级缓存块地址。此外也需要一个块内偏移映射器根据该二级指令子缓存块与其对应的一级指令缓存块之间的映射关系（如上述三种方式等）提供分支目标的块内偏移映射。

[0313] 请参考图15，其为本发明所述包含可配置转换器和地址映射模块的处理器系统的另一个实施例。本实施例中转换器1202、指令存储器1203和处理器核1201均与图12、14中的相同，另外还给出了地址映射模块的一种具体实施方式。在本例中，如果指令存储器1203缺失，可将相应的外部指令地址送往更外层次存储器获取相应的外部指令块经指令转换器1202如前述转换并填充到指令存储器1203中。以下各实施例的说明均假设指令存储器1203总是命中。

[0314] 地址映射模块由标签存储器1505（相当于前述实施例中的主动表604）、块内偏移映射器1504（为了简单明了，1504中包含图6中618偏移地址映射模块和622偏移地址映射器的功能）和结束标志存储器1506构成，三者的行均与指令存储器1203中的内部指令块对应。其中，结束标志存储器1506的每行存储了指令存储器1203中对应内部指令块的最后一条内部指令的块内偏移地址。可以在处理器核1201读取内部指令的同时在结束标志存储器1506中检查该内部指令是否为当前内部指令块中的最后一条。若该内部指令不是当前内部指令块中的最后一条，则下一内部指令的块内偏移地址就是该内部指令的偏移

地址加一；否则，下一内部指令就是下一内部指令块的第一条内部指令。

[0315] 标签存储器 1505 中的每行存储了外部指令块地址（即标签），因此可以通过标签匹配找到该外部指令所在指令块对应的内部指令块在指令存储器 1203 中的位置，以及与该内部指令块同一行中块内偏移映射器 1504 中相应的映射关系、结束标志存储器 1506 中该指令块中最后一条内部指令的位置信息。与缓存结构类似，对于不同的存储器组织形式，标签存储器 1505 和指令存储器 1203 可以有不同的结构。具体地，以直接映射存储结构为例，所述外部指令的块地址可以被进一步分为标签和索引号，根据索引号对标签存储器 1505 中的行寻址读出相应行的内容后与块地址中的标签进行比较，若相等则匹配成功，否则匹配不成功。当匹配不成功时可以用该外部指令地址从更低的指令存储器中取得相应的外部指令块经指令转换器 1202 如前转换为内部指令块后按缓存替换规则写入指令存储器 1203 中，并由将外部指令中的标签写入标签存储器 1505 的同一行，将指令转换器 1202 产生的块内偏移映射关系存入块内偏移映射器 1504、将 1202 产生的该指令块最后一条指令的块内偏移量存入结束标志存储器 1506 的同一行。当然，标签存储器 1505 和指令存储器 1203 也可以被组织为其他任何合适的组织结构（例如：组相联或全相联结构），其具体匹配方法均与缓存中相应组织结构情况下的匹配方法相同，在此不再赘述。为了便于描述，在后面的实施例中均以直接映射结构为例进行说明，且假设标签匹配均成功。

[0316] 处理器核 1201 根据是否需要分支或跳转通过总线 1508 提供不同的指令地址。当一条指令地址经总线 1508 输出以控制指令存储器 1203 读取指令供处理器核 1201 执行时，1508 上的块地址也被送到结束标志存储器 1506 中寻址读出该行的结束地址，与 1508 上的内部指令块内偏移地址进行匹配以检查该内部指令是否为内部指令块中的最后一条。如该指令不是内部指令块中最后一条指令时，则结束标志存储器输出的 1507 信号控制处理器核 1201 在下一时钟周期的指令块地址不变，块内偏移量增‘1’在下个周期放上总线 1508。若是最后一条，则结束标志存储器输出的 1507 信号控制处理器核 1201 在下个周期输出下一指令块的外部指令块地址（由当前指令块地址增‘1’所得）并以‘0’作为内部指令的块内偏移地址，组合成指令地址放上总线 1508。此时 1507 也控制将 1508 上的指令块地址送往标签存储器 1505 匹配，如匹配，则总线 1508 上就是下一条指令的正确地址。在被执行的指令为非分支指令时，分支判断信号 1509 控制选择器 1510 选择总线 1508 上的块内偏移地址对指令存储器 1203 寻址读取下一周期的内部指令供处理器核 1201 执行。用于对指令存储器 1203 的块地址则在任何时候都来自总线 1508。

[0317] 但当处理器核 1201 译码上述来自指令存储器 1203 的指令发现其为一条分支指令时，则按照指令进行分支判断。如果分支判断为‘不分支’，则下一周期产生的地址如同上述一般。分支判断信号 1509 控制选择器 1510 选择总线 1508 上地址。如果分支判断为‘执行分支’，则以分支指令的外部指令地址加上分支指令中所含有的分支偏移量获得分支目标的外部指令地址经总线 1508 在下一周期送出。为了减少对地址值的存储，实际上处理器核只记录了分支指令（或其他指令）的内部指令地址。可以用例如图 8B 的映射关系，以图 8C 中的映射装置进行逆运算，即以内部指令地址送到译码器 805，及内部指令映射关系送到 807 作为输入，外部指令的映射用以控制矩阵 803，则该装置的输出即为外部指令地址。也可以在进行指令转换时，将外部分支指令的外部块内偏移量加到该分支指令的分支偏移量，将其和作为分支偏移量记录到内部分支指令中。如此在处理器核 1201 执行到分支

指令时，只需要将指令块地址（块内偏移量为‘0 ‘）加上分支指令中所记录的修正后分支偏移量，其和就是正确的外部分支目标地址，省却分支指令内部指令块内偏移量映射到外部指令块内偏移量的操作。

[0318] 该外部指令分支目标地址中的块地址经总线 1508 被送到标签存储器 1505 匹配，也被送到块内偏移映射器 1504 读出该行的映射关系将 1508 上的外部块内偏移量映射为内部指令块内偏移量 1512。分支判断信号 1509 控制选择器 1510 选择 1512 作为块内偏移量送往指令存储器 1203。1508 总线上的块地址也被送往指令存储器 1203。若标签存储器 1505 匹配成功，则以该地址取分支目标指令供处理器核执行。

[0319] 实际上在本实施例中，总线 1508 上的下一条指令的块地址（包含指令地址中标签及索引部分）一直是外部指令地址。其中索引部分被用于对所有存储器如 1505，1504，1516 及 1203 做行寻址。而 1508 上的下一条指令的块内偏移地址则根据指令的类型等可以是外部指令地址或内部指令地址。如果当前指令的类型为非分支指令或分支指令但不执行分支时，且该指令非内部指令块中最后一条指令，则下一条指令的块内偏移地址是内部指令格式（当前指令地址增‘1’，指向当前内部指令的下一条内部指令）。如果当前指令的类型为非分支指令或分支指令但不执行分支时，且该指令是内部指令块中最后一条指令，则下一条指令的块内偏移地址‘0’可以被视为外部指令格式，也可被视为内部指令格式。如果当前指令的类型为分支指令且执行分支时，则下一条指令的块内偏移地址是外部指令格式，要经过块内偏移映射器 1504 映射为内部块内偏移指令地址才可用于从指令存储器 1203 中读取指令。如果将外部地址中的索引部分视为内部指令地址的块地址，则指令存储器 1203 在任何时刻都是由内部指令地址寻址。如果指令存储器 1203 及指令地址映射模块的组织方式是多路组，则相似地内部指令的块地址由路号 (Way number) 与外部指令中的索引部分组成。即本实施例中公开的虚拟机中的地址映射模块，可以将由外部指令编译器产生的外部指令地址直接映射为内部指令地址访问存储内部指令的指令存储器，供处理器核执行。或者也可以将内部指令地址的块地址视为等同于外部指令地址的块地址（包含标签部分及索引部分）。本虚拟机既避免了现有软件虚拟机通过软件将外部指令地址映射为内部指令时的低效及储存庞大的地址映射表的开销；也避免了现有硬件虚拟机由外部指令地址对存有外部指令的指令存储器寻址，读出外部指令，将其由指令转换器转换为内部指令再由处理器核执行，因为多次重复转换同一条指令导致的高功耗。本虚拟机的一个技术特征是外部指令先经过指令转换器转换才存入指令缓存，因此指令缓存中储存的是内部指令，可不需指令转换直接执行。

[0320] 根据本发明技术方案，还可以增加分支目标表用于记录分支目标指令的内部指令地址，使得重复执行同一条分支指令发生分支转移时不必每次都需要将分支目标指令的外部指令地址转换为内部指令地址。请参考图 16，其为本发明所述包含分支目标表的处理器系统的一个实施例。在本实施例中，可配置转换器 1202、指令存储器 1203、处理器核 1201、标签存储器 1505、块内偏移映射器 1504 和结束标志存储器 1506 均与图 15 中的相同。不同之处在于增加了分支目标存储器 (BTB)1607 以及选择器 1608 的连接方式与图 15 种选择器 1510 不同。在此，分支目标存储器 1607 中存储了以内部指令地址形式记录的分支目标历史信息，即存储了该分支指令本身的内部指令地址，及其分支目标的内部指令地址，及之前执行该分支指令时是否转移的预测信息。分支目标存储器 1607 并不必要与其他存储器

行行对应。分支目标存储器 1607 输出其分支预测信号 1511 以控制选择器 1608 选择来自总线 1508 或分支目标存储器 1607 的指令地址。

[0321] 这样，在处理器核 1201 经总线 1508 输出内部指令地址到指令存储器 1203 寻址的同时，还将该内部指令地址送到分支目标存储器 1607 与存储在其中的所有分支指令本身的内部指令地址进行匹配，并输出匹配成功项包含的分支目标内部指令地址和预测信息。当前指令为非分支指令时或虽为分支指令但分支预测为不分支时，下一时钟周期分支预测选择信号 1511 控制选择器 1608 选择总线 1508 上的指令地址访问指令存储器 1203，其操作与图 15 实施例在执行同样指令时的操作相同，在此不再赘述。当前指令为分支指令且分支预测为执行分支时，分支预测选择信号 1511 控制选择器 1608 选择分支目标存储器 1607 输出的内部指令分支目标地址访问指令存储器 1203。当前指令为分支指令但在分支目标存储器 1607 中未匹配命中时，则在分支目标存储器 1607 中按置换规则分配一个表项以存储分支指令的内部指令地址。如果分支判断为‘执行分支’，则如图 15 例处理器核 1201 产生外部指令地址经总线 1508 送出。以该外部指令地址如图 15 例经标签存储器 1505 匹配确认的指令块地址，及经块内偏移映射器 1504 映射得到的内部指令块内偏移量 1512 一同作为内部指令分支目标地址，以及分支预测值存储进分支目标存储器 1607 中新分配表项中的相应域。该内部指令分支目标地址也被分支目标存储器 1607 旁路经选择器 1608 访问指令存储器 1203。如果分支判断为‘不分支’，则将分支目标存储器 1607 中新增表项置为无效，分支预测选择信号 1511 控制选择器 1608 选择总线 1508 上的指令地址（此时为分支指令的下一条顺序内部指令的地址）访问指令存储器 1203；此时 1508 上的指令地址于图 15 例中在同等条件下产生的地址相同，不再赘述。当执行分支指令判断分支预测为错误时，处理器核 1201 清除按错误预测执行的指令的中间结果，执行正确的分支，并更新分支目标存储器 1607 中存储的分支预测。

[0322] 请参考图 17，其为本发明所述包含分支目标表和循迹器的处理器系统的另一个实施例。本实施例中的转换器 1202、指令存储器 1203、处理器核 1721、标签存储器 1505、块内偏移映射器 1504、结束标志存储器 1506 和分支目标存储器 1607 均与图 16 中的相同。不同之处在于，本例中还包括下块地址存储器 1709、选择器 1711、或逻辑 1707 和循迹器，并通过循迹器产生内部指令地址，使得处理器核 1701 只需要输出外部指令地址即可。

[0323] 本实施例中新增的下块地址存储器 1709 与标签存储器 1505、块内偏移映射器 1504、结束标志存储器 1506 均行行对应，其格式请参考图 18A 的一个实施例。在本例中下块地址存储器每行包含两个部分：第一部分 1801 存储了该行对应的内部指令块的上一内部指令块的 X 地址；第二部分 1802 存储了该行对应的内部指令块的下一内部指令块的 X 地址。这样，使用当前内部指令块的块地址（即循迹器输出的 X 地址）对下块地址存储器 1709 寻址，即可读出顺序地址的上一、下一内部指令块的相应 X 地址。而选择器 1711 则根据处理器核 1201 输出的分支转移是否发生的 TAKEN 信号 1713 对下块地址存储器 1709 输出的下一内部指令块的 X 地址与 Y 地址‘0’构成的下一内部指令块第一条内部指令地址，及分支目标存储器 1607 输出的分支目标内部指令地址选择后送往选择器 1705。或逻辑 1707 则在当前内部指令为内部指令块最后一条指令、或发生分支转移时，控制选择器 1705 选择来源于选择器 1711 的输入。

[0324] 所述循迹器由寄存器 1701、增量器 1703 和选择器 1705 构成。其中，寄存器 1701

中存储，并输出由块地址（以下简称为X地址）和内部指令块内偏移地址（以下简称为Y地址）构成的当前内部指令地址1723。当前内部指令地址1723用于对指令存储器1203寻址读取其中一行中的内部指令送到处理器核1721译码，并同时访问下块地址存储器1709，结束标志存储器1506的相应一行，也被送到分支目标存储器(BTB)1607中匹配。1723中的X地址寻址结束标志存储器1506中读取相应的一行中的内容与1723上的Y地址进行匹配以检查该指令是否为内部指令块中的最后一条。若该指令不是最后一条且处理器核1721对该指令译码结果判断为不是分支指令，则或逻辑1707控制选择器1705选择寄存器1701输出的X地址及增量器1703输出的增‘1’后的Y地址存储到寄存器1701中作为下个时钟周期的当前内部指令地址。

[0325] 若该指令是最后一条内部指令或是分支指令，则选择器1705在或逻辑1707的控制下选择选择器1711的输出存储到寄存器1701中作为下个周期的当前内部指令地址。具体地，若当时分支判断信号(TAKEN)1713为‘不分支’，则控制选择器1711选择下块地址存储器1709中，由前述当前内部指令地址1723寻址提供的下一内部指令块第一条内部指令的地址，经选择器1705选择后存储到寄存器1701中。若当时分支判断信号(TAKEN)1713为‘执行分支’，则控制选择器1711选择分支目标存储器1607中，由当前内部指令地址1723匹配获得的分支目标内部指令地址，经选择器1705选择后存储到寄存器1701中。也可以用分支目标存储器1607中存储的分支预测值代替处理器和1721产生的分支判断信号1713控制选择器1711及1705。这种方式需要核实分支预测是否正确及一旦预测错误可以修正的机制。

[0326] 本实施例中，控制指令存储器1203等的内部指令地址由循迹器提供。处理器核1721仅在当前内部指令地址1723与分支目标存储器1607内容匹配不命中，或在下块地址存储器1709寻址遇到无效表项，且在分支判断及结束指令判断选择了上述不命中或无效指令地址时才需要提供外部指令地址1708作为下一周期的地址。具体地，在上述分支目标存储器1607内容匹配不命中时，处理器核1721如图16例同样方式计算外部指令分支目标地址1708送往标签存储器1505匹配，也送到块内偏移映射器1504映射。匹配映射所得的内部指令分支目标地址如图16例同样方式被存入分支目标存储器1607的表项，并被存入寄存器1701作为当前内部指令地址1723。在上述下块地址存储器1709寻址遇到无效表项时，处理器核1721如图16例同样方式计算外部指令下块地址1708送往标签存储器1505匹配。匹配所得的内部指令下块地址被存入上述无效表项中的1802域，也要将本地址块的块地址存入下块地址存储器1709中对应上述匹配所得的下块地址所指向的一行中的1801域。

[0327] 需要说明的是，由于各个顺序地址的内部指令块通过下块地址存储器1709中存储的信息联系在一起，即可以根据当前内部指令块的X地址对下块地址存储器1709寻址读出下一内部指令块的X地址1802。如果某个内部指令块被替换出指令存储器1203，可以根据该内部指令块的X地址对下块地址存储器1709寻址读出其中存储的上一内部指令块的X地址1801，再根据该1801中的X地址对下块地址存储器1709寻址找到相应行，将该行中存储下一内部指令块（即被替换的内部指令块）X地址的部分1802置为无效，从而反映替换后的地址关系。如果指令存储器以组相联方式组织，则一个指令块的下个指令块的行地址是本指令块的行地址增‘1’，可以缺省；1801及1802域记录路号(Way number)即可实现

其功能。

[0328] 进一步地，可以将上述技术扩展到包含更多层指令存储器的系统。请参考图19，其为本发明所述包含两层指令存储器的处理器系统的一个实施例。在本例中，转换器1202、指令存储器1203、处理器核1201、块内偏移映射器1504、结束标志存储器1506、分支目标存储器1607、下块地址存储器1709、选择器1711、或逻辑1707和循迹器均与图16中的相同。不同之处在于，指令存储器1203、块内偏移映射器1504、下块地址存储器1709、结束标志存储器1506和分支目标存储器1607共同构成第一级指令存储层次，而指令存储器1903、标签存储器1905和块地址映射模块1904（与图6中620类似功能）共同构成第二级指令存储层次。在此，指令存储器（以下改称一级指令缓存器以资与1903明确区分）1203中存储的是内部指令，而指令存储器1903中存储的是外部指令。指令存储器1903中的外部指令在被处理器核1201执行前先被转换器1202转换为相应的内部指令后存储在一级指令缓存器1203中供处理器核1201取用。

[0329] 在本实施例中一个外部指令块可以对应多个内部指令块。在本例中，指令存储器1903中包含了一级指令缓存器1203中的所有内部指令对应的外部指令，因此可以使用一个标签存储器1905同时为两个存储层次服务。

[0330] 在本实施例中，标签存储器1905的行与指令存储器1903中的外部指令块一一对应，其中存储有相应外部指令块的标签地址。此外，本例中还增加了块地址映射模块1904，也与标签存储器1905行行对应，每行存储了该外部指令块在一级指令缓存器1203中对应的单数个或复数个内部指令块的1X地址及有效信号（当该外部指令块对应的一个内部指令块尚未被存储到1203中时，其相应1X地址的有效信号为无效）。请参考图18C，其为所述两个存储层次虚拟机系统中外部指令地址格式的示意图。在此，外部指令地址由块地址、子块号1813和块内偏移地址1814构成。其中，块地址对应指令存储器1903中的外部指令块，可以被进一步分为标签1811和索引号1812，并可根据索引号1812对标签存储器1905的行寻址读出存储在其中的标签信息，与地址中的标签1811比较以确定外部指令块地址是否匹配成功。索引号1812也可对块地址映射模块1904中的存储器寻址选出其中一行，子块号1813选择该存储器中的一列。

[0331] 请参考图20，其为本发明所述块地址映射模块1904结构的一个示意图。所述块地址映射模块由写入模块2001、输出选择器2007，以及存储器构成。此例中每个外部指令块被分为两个子块，每个子块中的外部指令被指令转换器1202转换为内部指令存入一级指令缓存中的一个一级指令块。因此1904中的存储器每一行对应于二级指令缓存1903中的一个（二级）外部指令块，存储器也被分为两列2003及2005对应每个外部指令块中的两个子块由子块号1813选择。存储器的每个表项对应一个子块，其中存有与该外部指令子块对应的（一级）内部指令块的一级指令块地址（1X地址）。如此块地址映射模块1904可以将外部指令块地址映射为与其相应的内部指令块地址，将外部指令子块与其相应的内部指令块联系起来。并且一个外部指令子块的相应内部指令块可以被放置在一级指令缓存中任意一个一级缓存块中，因此一级指令缓存可以是全相联组织形式。

[0332] 具体地，对块地址映射模块1904中存储器写入时由外部指令地址中的子块号1813控制写驱动器2001选择驱动存储器列2003或2005，由索引地址1812选择存储器中一行供写入相应的内部指令1X地址（即图20中的1X）。对块地址映射模块1904中存储器

读出时由索引地址 1812 选择存储器中一行读出，由外部指令地址中子块号 1813 控制选择器 2007 选择存储器列 2003 或 2005 的数据输出。

[0333] 回到图 19，对于第一级指令存储层次，其工作原理和运行过程与图 17 实施例类似，不同之处仅在于当前内部指令地址 1723 与分支目标存储器 1607 内容匹配不命中，或在下块地址存储器 1709 寻址遇到无效表项，且在分支判断及结束指令判断选择了上述不命中或无效指令地址时的处理有所不同。如图 17 实施例一样，此时处理器核 1721 提供外部指令地址 1708 作为下一周期的地址。不同的是该外部指令地址不再直接由图 17 中在本层次中的标签存储器 1505 转换，而是以该外部指令地址中的索引 1812 读出标签存储器 1905 中的一个表项与外部指令地址中的标签 1811 匹配，以外部指令地址中的索引 1812 及子块号 1813 对块地址映射模块 1904 寻址。如标签匹配命中，且 1904 中读出的 1X 地址有效，说明所需的内部指令已存储在一级指令缓存器 1203 中。此时将读出的 1X 地址经总线 1906 送回第一级指令存储层次填充无效的下块地址存储器 1709 中表项；或以该 1X 地址寻址块内偏移映射器 1504 将总线 1708 上的外部指令块内偏移地址映射为内部指令块内偏移地址，1X 地址连同上述内部指令块内偏移地址构成内部指令分支目标地址（1Y 地址）存入匹配未命中的分支目标存储器 1607 表项。此后操作与图 17 例中相同。

[0334] 如标签匹配命中，且从 1904 中读出的 1X 地址无效，说明所需的内部指令尚未存储在一级指令缓存器 1203 中。此时以总线 1708 上的外部指令地址对二级缓存 1903 寻址，将相应外部指令子块送到指令转换期 1202 转换为内部指令块存储进一级指令缓存器 1203 中由缓存置换逻辑指定的一级缓存块，并将该一级缓存块的 1X 地址存入 1904 中外部指令指向的表项（即原读出无效的表项），并将该地址设为有效。指令转换过程中产生的块内偏移映射关系，以及结束标志也被写入块内偏移映射器 1504 及 1506 中由该 1X 地址指向的行。将读出的 1X 地址如前例经总线 1906 送回第一级指令存储层次存入无效的下块地址存储器 1709 中表项，或连同映射产生的内部指令块内偏移地址存入匹配未命中的分支目标存储器 1607 表项。此后操作与图 17 例中相同。

[0335] 如标签匹配未命中，说明所需的指令尚未存储在二级指令缓存 1903 中。此时将总线 1708 上的外部指令地址送到更低层的存储器取外部指令块填入二级指令缓存 1903 中由缓存置换逻辑指定的一个二级缓存块。同时将总线 1708 上外部指令中的标签 1811 存入标签存储器 1905 中与上述二级缓存块相应的表项，将块地址映射模块 1904 中与上述二级缓存块相应的两个表项均置为无效。之后按上述标签匹配命中但寻址获得的块地址映射模块中 1X 地址无效的情况执行。

[0336] 当外部指令为定长指令时，外部指令块或子块的边界与一条外部指令的起点重合。因此无论是因为顺序执行进入、还是因为分支转移进入该外部指令块（或子块）时，都可以从该外部指令块或子块的边界开始将完整的块或子块转换成相应的内部指令块存储到内部指令存储器中。当外部指令集是一种变长指令集时，一个外部指令块（或子块）中第一条外部指令的起始地址可能不一定与块（或子块）的边界重合。这种情况下，当分支转移进入一个外部指令块时，只能对从分支目标指令开始至该外部指令块结束的部分外部指令块进行转换并存储在一个内部指令缓存块以供处理器核执行；而对于该分支目标指令之前的指令则要等待下一次分支目标或顺序进入该外部指令块时其起始点落在这些指令上才能进行转换并将转换得到的内部指令添加进上述内部指令块。可以修改转换得到的内

部指令在一级指令缓存器 1203 中的存储形式以适应这种情况，并定义每条外部指令属于其开始地址所在的外部指令块。

[0337] 请参考图 21，其为本发明所述外部指令与块边界不对齐情况下指令存储器存储内部指令的一个实施例。外部指令块 2101 为指令存储器 1903 中的一行外部指令块（或子块），内部指令块 2102 为一级指令缓存 1203 中与外部指令块 2101 对应的一行内部指令块。假设第一次分支转移的目标指令为外部指令 2105，可以从目标指令 2105 开始一直到该指令块转换完毕，存入内部指令缓存块。因此可以依然按照地址递增的顺序存储内部指令，但将转换得到所有内部指令的最高地址与内部指令块 2102 的地址最高位 (MSB) 处（即图 21 中内部指令块 2102 的最右侧）对齐。这样，外部指令 2105 对应的内部指令 2106 就被存储到如图 21 所示位置，且从外部指令块 2101 中从指令 2105 开始的所有外部指令对应的内部指令均被按地址顺序存储到内部指令块 2102 中如图 21 所示的阴影部分中。

[0338] 此外，在本实施例中，指令存储器 1903 和 1203 的每行都增加了一个指针，分别用于指向外部指令块中已经被转换的第一条外部指令（如图 21 中指向内部指令 2105 的指针 2103），以及内部指令块中已经被存储的第一条内部指令（如图 21 中指向内部指令 2106 的指针 2104）。这样，当再次因顺序执行或分支转移进入该外部指令块时，就可以比较进入时外部指令块内偏移地址和所述指针 2103，确定目标指令是否已被转换。进一步地，如果确定新的目标指令尚未被转换，则对外部指令块 2101 中从该新的目标指令开始直至指针 2103 指向的外部指令之前的所有外部指令转换后，将转换得到的所有内部指令的最高地址与内部指令块 2102 中指针 2104 指向的地址的前一地址对齐，并依然按照地址递增的顺序存储内部指令。同时分别将指针 2103、2104 的值更新为指向所述新的目标指令在外部指令块 2101 中的位置，及该新的目标指令对应的内部指令在内部指令块 2102 中的位置。块内偏移映射器 1504 中的内部指令映射关系也按高位对齐的方式存放，与内部指令缓存块一致。可以将上述两个指针在块内偏移映射器 1504 中的每行中实现。

[0339] 根据本发明技术方案，当采用图 21 实施例所述内部指令存储方式时，每个内部指令块的第一条指令不一定位于该内部指令块的起始地址（即 Y 地址‘0’）。因此需要对所述处理器系统中的下块地址存储器进行相应修改。请参考图 18B，其为本发明所述下块地址存储器格式的另一个实施例。在本例中下块地址存储器每行除了包含图 18A 实施例中的第一部分 1801 和第二部分 1802 外，还增加了一个第三部分 1803，用于存储了该行对应的内部指令块的下一内部指令块中第一条内部指令的 1Y 地址。这样，第二部分 1802 和第三部分 1803 共同构成了所述下一内部指令块第一条内部指令的地址，使得在因为外部指令边界不对齐而导致内部指令没有从内部指令块 LSB 开始存储的情况下，依然能根据当前内部指令块的块地址（即 1X 地址）对下块地址存储器寻址读出相应地址以找到下一内部指令块的第一条指令。图 21 及图 18B 的格式也可以应用于图 15，图 16，图 17 实施例中以处理外部指令起始地址与外部指令块边界不对齐的情况。

[0340] 图 21 及图 18B 阐述了外部指令子块与内部指令块具有严格一一对应映射关系情况下解决指令与块边界不对齐问题的一个实施例。图 22 为本发明所述块地址映射模块的另一个实施例，描述一种外部指令块与内部指令块间弹性映射以解决指令与块边界不对齐问题的实现方式，可以应用在图 19 的实施例中。在此例中，以一个外部指令块中的指令可被转换为内部指令放入最多三个（可以是任何数目）内部指令块为例，则块地址映射模块

的主体部分被分为 3 个存储器 2201、2202 和 2203，这三个存储器的每行均与一个外部指令块对应，且每行由两个存储域组成，分别用于存储该外部指令段的起始外部指令在其所在外部指令块中的块内偏移地址（如图中的 2Y），以及该子块对应的内部指令块在一级指令缓存器 1203 中的块地址（如图中的 1X）。此外，这三个存储器的相应行之间还具有通路 2205 和 2206，可以分别将存储器 2201 任意行的内容右移到存储器 2202 的相应行中，以及将存储器 2202 任意行的内容右移到存储器 2203 的相应行中。

[0341] 当一个外部指令块第一次作为分支目标被访问时，从分支目标的外部指令块内偏移地址 (2Y) 起该外部指令块的所有完整指令都被转换为内部指令依次放入一个内部指令块。该 2Y 值以及上述内部指令块的块地址 (1X) 被存入图 20 中存储器 2201 中外部指令块地址 (2X) 所指向的一行，以记录块地址为该 1X 的内部指令块中第一条内部指令对应该 2X 外部指令块中块内偏移为该 2Y 的外部指令。如果上述内部指令块中放满了还有更多的内部指令，则分配另一个内部指令块存放这些溢出的内部指令，并将溢出内部指令中第一条对应的外部指令的块内偏移地址 (2Y) 连同新分配内部指令块的块地址 (1X) 存入存储器 2202 中 2X 指向的一行。外部指令与内部指令的块内偏移映射关系也被存入图 19 中块内偏移映射器 1504 中由 1X 寻址的行。

[0342] 进一步，分支目标的外部偏移地址 2Y 与块内偏移映射器 1504 中由相应的内部指令块地址 1X 指向的映射关系映射为内部指令块内偏移 1Y。至此，由分支目标开始的外部指令块已经被指令转换器 1202 转换为内部指令；外部指令块地址 2X 也已由块地址映射模块 1904 映射为内部指令块地址 1X，外部指令块内偏移地址 2Y 也被块内偏移映射器 1504 映射为内部指令偏移地址 1Y。更进一步，该分支目标内部指令地址 1X，1Y 可被存入分支预测模块 1607 中，供循迹器选用。

[0343] 回到图 20，下一次该外部指令块被访问时，以访问地址中的外部指令块地址 2X 对存储器 2201，2202 与 2203 寻址读出同一行送入比较器 2204。访问地址中的外部指令块内指令偏移地址 2Y 在比较器 2204 中与从各存储器读出的各 2Y 进行比较，选择第一个其 2Y 值小于访问地址中 2Y 值的存储器所存储的的 1X 值作为块地址映射模块 1904 的输出 1906。后续操作与前述相同。如果存储器 2201 中的 2Y（其值是块地址映射器 1904 中所有存储器中最小的）仍大于访问地址的 BN2Y，则访问目标指令仍未转换为内部指令，此时系统控制指令转换器 1202 将从访问目标开始一直到存储器 2201 存储的 2Y 值之前的外部指令都转换为内部指令存入一级缓存块置换逻辑指定的一级缓存块。同时图 20 中存储器 2202 中由访问地址中的外部指令块地址 2X 指向的行被右移到 2203 中同一行，存储器 2201 中由该 2X 指向的行被右移到 2202 中同一行，而访问目标的 2Y 值与新指定的 1X 值被存入存储器 2201。如此一个外部指令块以多次访问的起始点开始被转换为若干个内部指令块，其映射关系也被纪录在如图 22 结构的块地址映射模块 1904 中。图 8 实施例中对图 22 结构的块地址映射模块的操作有详细说明。当得到外部指令对应的内部指令 1X 地址与 1Y 地址后，后续操作与前述相同，在此不再赘述。

[0344] 根据本发明技术方案，还可以将轨道表结合到所述处理器系统中。请参考图 23，其为本发明所述包含轨道表的处理器系统的一个实施例。在本实施例中，由于本发明所述的轨道表本身已经包含了分支目标地址信息、下一指令块地址信息，以及结束轨迹点信息，因此可以用轨道表 2301 代替了下块地址存储器 1709、结束标志存储器 1506 和分支目标存

储器 1607。此外，标签存储器 1905、块地址映射模块 1904、转换器 1202、一级指令缓存器 1203、处理器核 1201、块内偏移映射器 1504、选择器 1711、或逻辑 1707 和循迹器均与图 19 中的相同。在本例中还增加了扫描器 2302，如前所述用于对被转换的外部指令进行审查，并对其中的分支指令计算分支目标的外部指令地址 BN2 后，转换为相应的内部指令地址 BN1。在本例中，由于所述内部指令地址 BN1 就是一级指令缓存器 1203 的地址，一级指令缓存器 1203 中的内部指令与轨道表 2301 中的轨迹点一一对应，且分支指令对应的轨迹点中包含了分支目标的内部指令地址，因此可以如前所述由循迹器对轨道表 2301 寻址读出轨迹点内容，并根据分支指令执行情况，选择当前循迹地址增‘1’或轨迹点中的分支目标循迹地址作为下一内部指令的循迹地址。

[0345] 此外，还可以根据轨道表 2301 中轨迹点的内容确定是否到达内部指令块的最后一条指令。例如，可以在轨迹点中用一个标志位表示该轨迹点是否对应所述最后一条指令，当循迹器读指针指向该轨迹点时，根据总线 2313 上读取到的该标志位值即可判定已经到达所述最后一条指令。

[0346] 在本例中，轨道表 2301 可以同时通过总线 2311 输出循迹器读指针 1723 指向的轨迹点的内容，以及通过总线 2309 输出该轨迹点所在轨道的结束轨迹点（存有顺序下一个内部指令块起始点的地址）的内容，从而如图 19 实施例那样同时向选择器 1711 提供分支目标循迹 BN1 地址和下一内部指令块 BN1 地址。

[0347] 本实施例与图 19 实施例还有一个不同之处在于增加了一个选择器 2315，用于对块地址映射模块 1904 经总线 1906 送来的 BN1X 及块内偏移映射器 1504 送来的 BN1Y 地址合并而成的 BN1 内部指令地址（也是一级指令缓存地址）和扫描器 2302 输出的 BN2 二级缓存地址选择后存入轨道表 2301。

[0348] 具体地，当扫描器 2302 对二级指令缓存 1903 送往一级指令缓存器 1203 的外部指令进行审查时，对其中的分支指令，按外部分支指令地址加指令中携带的外部分支偏移量的方式计算其分支目标的外部指令地址。计算出的外部分支指令地址索引部分对标签存储器寻址，读出内容与外部分支指令中的标签部分匹配。如不命中则以该外部指令从更低层次存储器读取该外部指令块存入二级缓存 1903 中由缓存块置换逻辑指定的二级缓存块；并在标签存储器 1905 的相应一行中存入该外部指令的标签部分，在块地址映射模块 1904 的相应一行将所有有效位置为‘无效’。如命中，即以外部指令的索引号 1812（如果二级缓存 1903 以组相连形式组织还要连同路号）为二级缓存块地址 BN2X，子块号 1813 与块内偏移地址 1814 为 BN2Y 一同组成二级缓存地址 BN2。该 BN2 被存放到轨道表 2301 中与该外部分支指令对应的内部分支指令的表项中。如此，当一条外部分支指令经转换为内部指令存入一级指令缓存器 1203 时，其分支目标已经至少以外部指令形式存入二级指令缓存 1903，且该内部分支指令的相应轨道表表项中已存有该分支目标的二级缓存地址 BN2。

[0349] 以后当循迹器读指针 1723（一级缓存地址 BN1）寻址一级指令缓存器 1203 读出内部分支指令供处理器和 1721 执行的同时，也寻址轨道表 2301 读出与该指令相应的轨道表项。当轨道表 2301 的输出 2311 是 BN2 格式时且分支判断 1713 为‘执行分支’时，选择器 1711 将该 BN2 放上总线 2304，以该 BN2 对块地址映射模块 1904 寻址，如映射输出为‘无效’，说明该分支目标指令所在的指令块还未转换为内部指令块存入一级指令缓存器 1203。此时处理器系统控制以该 BN2 寻址二级缓存 1903 读取该外部指令块送往扫描器 2302 如前

述计算块中分支指令的分支目标，也如前述送到指令转换器 1202 转换为内部指令块如前述存入一级指令缓存器中由缓存块置换逻辑给出的 BN1X 地址指向的一级指令缓存块。系统也将该 BN1X 地址存入块地址映射模块 1904 中原来‘无效’的表项，也将指令转换器 1202 产生的偏移地址映射关系存入块内偏移映射器 1504 中该 BN1X 指向的行。进一步，虚拟机系统控制将外部指令偏移地址 1814 根据上述 BN1X 指向的 1504 中的映射关系行映射为内部指令 BN1Y。由上述 BN1X 与 BN1Y 构成的分支目标内部指令的一级缓存地址 BN1 被写入对应分支指令的轨道表表项以代替原来的 BN2。至此分支目标外部指令及其后的外部指令块已被转换为内部指令块存入一级缓存 1721，同时该内部分支目标指令的一级缓存地址已被存入与其分支源指令相应的轨道表表项。

[0350] 以后当循迹器输出的一级缓存地址 1723(BN1) 寻址一级指令缓存器 1203 读出内部分支指令供处理器和 1721 执行的同时，也寻址轨道表 2301 读出与该指令相应的轨道表项。当轨道表的输出 2311 是 BN1 格式时，该 BN1 经分支判断信号 1713 等控制选择器 1711 及 1705 选择，如 1713 为‘不分支’，则循迹器读指针，一级缓存地址 1723 经增量器 1703 增‘1’后作为下一周期的一级缓存地址 1723 ；如 1713 为‘执行分支’则轨道表输出的上述 BN1 作为下一周期的循迹器读指针，一级缓存地址 1723。一级缓存地址 1723 直接对一级指令缓存器 1203 寻址，读出内部指令供处理器核 1721 执行。图 6 实施例是图 23 中结构的一个具体实现。

[0351] 轨道表中的结束轨迹点也按同样方式处理，即当外部指令被转换为内部指令存入一个一级缓存块时，扫描器 2302 也计算出其顺序下一指令块的外部地址（当前外部指令块地址增一）并将其送往标签存储器 1905 匹配。如果不匹配，则按前述方式从更低层存储器取外部指令块存入二级缓存器 1903 中缓存块置换逻辑以 BN2X 地址指定的缓存块并更新标签存储器 1905 及块地址映射模块 1904 中的相应行。如此得到的 BN2X 或匹配时所得的 BN2X 被存入轨道表 2301 中与上述一级缓存块对应行的结束轨迹点。以后当缓存读指针 1723 指向这一行时，该 BN2 从轨道表结束轨迹点经 2309 读出，其 BN2X 如同前例分支目标指令地址 BN2 经 2311 读出时一般经总线 2304 被送到块地址映射模块 1904 映射为 BN1X（如该 BN1X 地址无效，则如前例由 BN2 寻址二级缓存器 2302 将外部指令转换为内部指令并存入一级指令缓存器 1203 中由缓存块置换逻辑以 BN1X 指定的一级缓存块，并更新标签地址存储器 1905 及块地址映射模块 1904)，该 BN1X 与总线 2304 上的 BN2Y 经块内偏移映射器 1504 映射为 BN1Y。该 BN1X 与 BN1Y 构成 BN1 地址经选择器 2315 被存储进轨道表 2301 中替换原 BN2。上述分支目标地址或下块地址都可以在第一次进行标签匹配时就检查相应块地址映射模块 1904 表项内容是否有效，如有效则说明分支目标指令或下块指令已经以内部指令形式存储在一级指令缓存器 1203 中，此时即以 1904 表项中的 BN1X 如上述过程将 BN2Y 映射为 BN1Y 而将 BN1 直接存入轨道表。

[0352] 请参考图 24，其为本发明所述利用寄存器堆实现栈操作功能的处理系统的一个实施例。为便于说明，在图 24 中只显示了部分模块和器件。在本例中，处理器核中的寄存器堆 2402 可以被配置为栈使用。此时，栈控制器 2404 则根据指令的译码结果及当前寄存器堆 2402 中的存储状况，调整输出地址 2405 和 2406 分别作为栈顶指针值和栈底指针值送往寄存器堆 2402。

[0353] 栈控制器 2404 的具体结构可以采用如图 10A 中控制器 1019、寄存器 1011、减量

器 1031、增量器 1041 和选择器实现。其中寄存器 1011 存储了当前栈顶指针值。最基本的栈操作包括出栈 (POP) 和压栈 (PUSH) 两种。减量器 1031 和增量器 1041 分别对当前栈顶指针值减‘1’和增‘1’,分别对应出栈(栈顶指针值减‘1’)和压栈(栈顶指针值增‘1’)的情况。这样,根据指令译码结果,可以将从存储器 2403 读取来的操作数依次压栈到寄存器堆 2402 中(栈顶指针值依次相应增‘1’)以实现基于栈的数据读取;也可以从寄存器堆 2402 中依次出栈若干个操作数(栈顶指针值依次相应减‘1’)送到执行单元 2401 做相应算术逻辑运算后,再压栈回寄存器堆 2402 中(栈顶指针值相应增‘1’)以实现基于栈的运算;还可以从寄存器堆 2402 中出栈操作数存储到存储器 2403 中(栈顶指针值相应减‘1’)以实现基于栈的数据存储。具体地,可以由寄存器堆处理器指令集中控制每个读或写口的寄存器堆地址域中的三个位控制对该读口或写口进行栈顶指针值操作(增‘1’,不变,或减‘1’)。

[0354] 在运行过程中,可以通过对栈顶指针值和栈底指针值进行比较判断该栈是否已满(或接近满),以及是否已空(或接近空)。一旦由寄存器堆 2402 构成的栈已满(或接近满),则可以在栈控制器 2404 的控制下将靠近栈底的若干数据暂存到存储器 2403 中,同时调整栈底指针指向新的栈底,从而使寄存器堆 2402 构成的栈空出一部分存储空间供后续栈操作使用。可以通过在存储器 2403 中按栈的形式组织存储空间,并以栈操作(压栈、出栈)存储所述需要暂存的数据的方法,保持这些数据的原始顺序信息。这样,一旦由寄存器堆 2402 构成的栈已空(或接近空),则可以在栈控制器 2404 的控制下从存储器 2403 的所述栈中按出栈的顺序读出若干之前暂存的数据存储回寄存器堆 2402 的相应寄存器中,并调整栈底指针指向新的栈底,即恢复这部分数据在被暂存到存储器 2403 前的状态,从而使得寄存器堆 2402 构成的栈中依然存有一部分数据供后续栈操作使用。这样,就可以利用寄存器堆实现栈操作功能。

[0355] 为了能在不同的硬件平台下通用,某些计算机程序语言产生以栈操作指令为主的中间代码,并在执行时由软件解释器对将中间代码实时翻译成若干条机器指令再由硬件平台执行,因此中间代码的执行效率不高。使用本发明所述处理器系统,可以直接执行这种栈操作指令(即,将每条栈操作指令转换为对应的一条内部指令),从而大幅提高处理器系统的执行效率。此外,与现有技术通常使用软件实现虚拟机相比,本发明所述的多指令集处理器系统完全以硬件实现了虚拟机。

[0356] 下面以图 23 所述结构为例,对本发明技术的几种实际应用情况进行举例说明。相应的方法和操作过程也可以应用到本发明提出的其他任何合适的结构中(例如图 15、16、17、19 结构等)。此外,为了便于说明,在下面的描述中仅以变长指令集、定长指令集,及栈操作指令集作为外部指令集的例子进行说明,但是其他任何合适的计算机指令集都可以作为外部指令集应用于本发明。

[0357] 首先,该虚拟机系统被用于执行由变长指令构成的程序,即外部指令为变长指令。先将该变长指令集与内部指令集的相应指令映射转换规则导入到转换器 1202 的存储器 1301 中及将控制 1202 中寄存器 212 等的值写入。其中控制指令转换起点的寄存器其值为从(分支目标或顺序的)进入地址开始转换。据此在执行变长指令时,若处理器核 1201 所需变长指令已经存储在指令存储器 1903 中,则对指令存储器 1903 寻址读出该变长指令所在指令块送往扫描器 2302 和转换器 1202,并对从该变长指令开始直至指令块中最后一条

尚未被转换的变长指令进行扫描 / 转换，计算分支指令的分支目标地址并转换为相应的内部指令地址，同时对这些变长指令转换得到的内部指令块被依次根据替换算法存储到一级指令缓存器 1203 的相应行中，并在轨道表 2301 的相应行中建立对应的轨道。具体地，在扫描 / 转换时，若一级指令缓存器 1203 中已经存储了分支目标对应的内部指令，则该分支目标的变长指令地址可以通过地址转换（如前所述由标签存储器 1905、块地址映射模块 1904 和块内偏移映射器 1504 完成）得到对应的内部指令地址 BN1 作为轨迹点内容存储到轨道表中。若一级指令缓存器 1203 中尚未存储分支目标对应的内部指令，但指令存储器 1903 中已经存储了该分支目标，则可以将该分支目标的变长指令地址 BN2 作为轨迹点内容存储到轨道表中。若指令存储器 1903 中尚未存储该分支目标，则可以将该分支目标从更外层存储器填充到指令存储器 1903 中由替换算法确定的行中，并将该分支目标的变长指令地址 BN2 作为轨迹点内容存储到轨道表中。这样，轨道表 2301 中就包含了变长分支指令的分支目标的地址信息。

[0358] 循迹器则根据从轨道表 2301 读取到的内容和处理器核 1201 对分支内部指令的执行结果，控制一级指令缓存器 1203 输出相应内部指令供处理器核 1201 执行。当按内部指令地址顺序执行时，可以通过增量器 1703 对循迹地址（即内部指令地址）增‘1’，或选择轨道表 2301 通过总线 2309 输出的下一内部指令块地址 2309，直接从一级指令缓存器 1203 中找到相应的内部指令。

[0359] 当执行分支转移时，可以根据轨道表 2301 的 2311 输出分支目标的内部指令地址 BN1 直接从一级指令缓存 1203 中找到相应的内部指令供处理器核 1201 执行。当轨道表 2301 输出的是分支目标的变长指令地址 BN2 时，若该变长指令对应的内部指令已经在之前的运行过程中被存储到了指令存储器 1203 中，则该变长指令地址如前所述可以通过地址转换得到对应的内部指令地址 BN1，并根据该地址从一级指令缓存器 1203 中找到相应的内部指令供处理器核 1201 执行。否则，根据该变长指令地址从指令存储器 1903 中找到相应的变长指令，并如前所述将从该变长指令开始直至指令块中最后一条尚未被转换的变长指令进行扫描 / 转换，将相应的内部指令块存储到一级指令缓存器 1203 中并在轨道表 2301 中建立相应的轨道，同时将该变长指令转换得到的内部指令提供给处理器核 1201 执行。处理器核 1201 执行所述内部指令产生相应的执行结果，例如执行分支内部指令时产生分支转移是否发生的 TAKEN 信号送往循迹器。循迹器则如前所述根据 TAKEN 信号及轨道表 2301 经总线 2313 送来的表示是否到达指令块最后一条指令的信号对多个地址来源进行选择，从而控制程序流继续执行。

[0360] 在本例中，当处理器系统执行完由变长指令构成的程序后，转而执行由定长指令构成的程序。在这种情况下，可以当最后一条变长指令执行完毕后，停止处理器核的运行，并将处理器核中及各存储器中状态置为无效，将定长指令集与内部指令集的指令对应转换规则及寄存器设置导入到转换器 1202 的存储器中及寄存器中以替代原存储的变长对应转换规则。其中控制指令转换起点的寄存器其值为从外部指令块或子块的最低地址开始转换。在执行定长指令时，若处理器核 1201 所需定长指令已经存储在指令存储器 1903 中，则对指令存储器 1903 寻址读出该定长指令所在指令块送往扫描器 2302 和转换器 1202，对该定长指令块整块进行扫描 / 转换，计算分支指令的分支目标地址并转换为相应的内部指令地址，同时对转换得到的内部指令块根据替换算法存储到一级指令缓存器 1203 的相应行

中，并在轨道表 2301 的相应行中建立对应的轨道。其具体操作与前述对变长指令的扫描 / 转换、存储到一级指令缓存器 1203 及在轨道表 2301 中建立轨道基本相同，不同之处仅在于在此是对整个定长外部指令块进行扫描转换。循迹器根据从轨道表 2301 读取到的内容和处理器核 1201 对分支内部指令的执行结果，控制一级指令缓存器 1203 输出相应指令供处理器核 1201 执行的过程与前述执行变长指令时相同。

[0361] 之后，假设该处理器系统接下来执行由变长指令集和定长指令集混合编码的程序，则可以通过在指令集切换时重新配置转换器 1202 的方式实现不同外部指令集的实时切换。具体方式与上述由执行一种指令集更换到另一种指令集类似，只是过程中不需要将轨道表 2301，指令缓存器 1203，1903 等所有存储器清零。由于轨道表 2301 中不同线程轨道之间互不干扰，而其他各存储器都与轨道表相关，因此各线程之间是相互独立的，有各自独立的轨道空间。在指令集或线程切换时，只要将一个线程的循迹器读指针 1723 和处理器核中的寄存器状态保存起来，留待恢复执行该线程时将这些数据重新填回，就可以当时该线程被切换出的点开始继续 (resume) 执行。可以在循迹器中使用一个存储器保存循迹器对应每个线程的读指针，使得线程（或虚拟机）切换时可以方便地恢复相应的读指针。同样，可以为处理器核 1701 的各状态寄存器建立一个对应各线程的存储器，如此在不同线程之间切换时，其时间间隔只是读指针，处理器核状态寄存器与读指针存储器，状态存储器间交换数据所需时间。

[0362] 最后，本发明所述处理器系统还可以结合图 13B 所述方法，由转换器 1202 根据线程号的不同采用相应的指令集对应关系对外部指令进行转换，使得在不同线程对应的指令集不相同的情况下，所述处理器系统不需要通过暂停处理器核来重新配置转换器，可以不间断地执行指令。具体地，可以在执行程序前，将所有可能用到的外部指令集的对应关系按图 13B 实施例所述方法导入到转换器 1202 的存储器中由线程号寻址的存储空间中。在对外部指令进行转换时，首先用线程号对转换器 1202 的所述存储器寻址找到对应的存储空间，然后再按前述方法根据该存储空间内的对应关系将外部指令转换为内部指令。在这种情况下，其他操作过程与前述实施例相同，在此不再赘述。由于每个虚拟机包含不同的线程，因此这样采用本例所述方法就可以实现在同一处理器系统上同时运行多个不同虚拟机的功能。如前所述，由于轨道表 2301 中不同线程轨道之间互不干扰，因此这些复数个虚拟机之间也不会因为不同线程的相同或不同计算机指令集的外部指令共存在二级缓存 1903 而发生相互干扰。保存循迹器读指针及处理器核内寄存器状态的方式如前所述。在这种方式中，还可以在同一个处理器系统上运行多个执行同一指令集的虚拟机其方式是在指令转换器 1202 中只存储一种外部指令集的转换规则，而各线程的基地址都指向这一种规则。不同线程（不同虚拟机）之间相互独立，线程（虚拟机）切换时如前交换循迹器指针及处理器核寄存器状态即可。

[0363] 如同以上实施例所述在两种不同的指令集之间无缝切换，具有本发明所述可直接执行栈操作指令的处理器，还可以在执行寄存器操作的指令集与执行栈操作的指令集之间无缝切换，可以不间断地执行多个不同指令集中的指令。具体地，可以在执行程序前，将所有可能用到的寄存器操作或栈操作指令集的转换规则按图 13B 实施例所述方法导入到转换器 1202 的存储器中由线程号寻址的存储空间中。在对寄存器操作指令或栈操作进行转换时，首先用线程号对转换器 1202 的所述存储器寻址找到对应的存储空间，然后再按前述

方法根据该存储空间内的对应关系将寄存器操作指令或栈操作转换为内部指令。在定义内部指令时在普通用于控制寄存器操作指令的指令域以外另增一个位，即图 10A 中的 1021 控制信号。在转换寄存器操作指令集为内部指令时，该位被设为‘0’，使信号 1021 控制选择器 1033，1035，1037 直接选择内部指令上的寄存器堆地址域直接对寄存器堆 1001 寻址控制其读写。在转换栈操作指令集为内部指令时，该位被设为‘1’，使信号 1021 控制选择器 1033，1035，1037 选择由内部指令上控制栈顶指针增减的指令域（可以是用执行寄存器操作指令时的寄存器地址域）控制选择器 1053，1055，1057 选择的栈顶指针 1045 及其增减量对寄存器堆 1001 寻址控制其读写。如此可以在处理器运行过程中，在寄存器操作的指令集与栈操作的指令集之间无缝切换。这使得只要有适当的条件，如前述的线程号，控制指令转换器 1202 使用正确的转换规则将外部指令转换为内部指令，则栈操作的指令可以内嵌在寄存器操作的指令集的程序中无缝执行。反之亦然。其他操作过程与前述实施例相同，在此不再赘述。

[0364] 根据本发明技术方案和构思，还可以有其他任何合适的改动。对于本领域普通技术人员来说，所有这些替换、调整和改进都应属于本发明所附权利要求的保护范围。

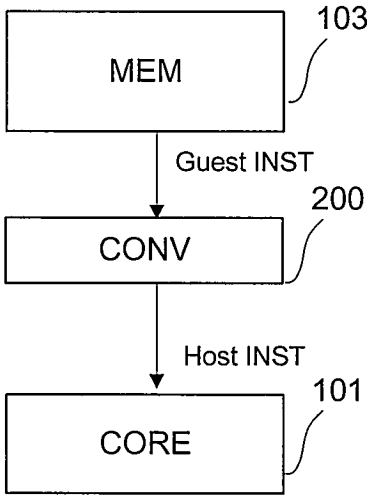


图 1

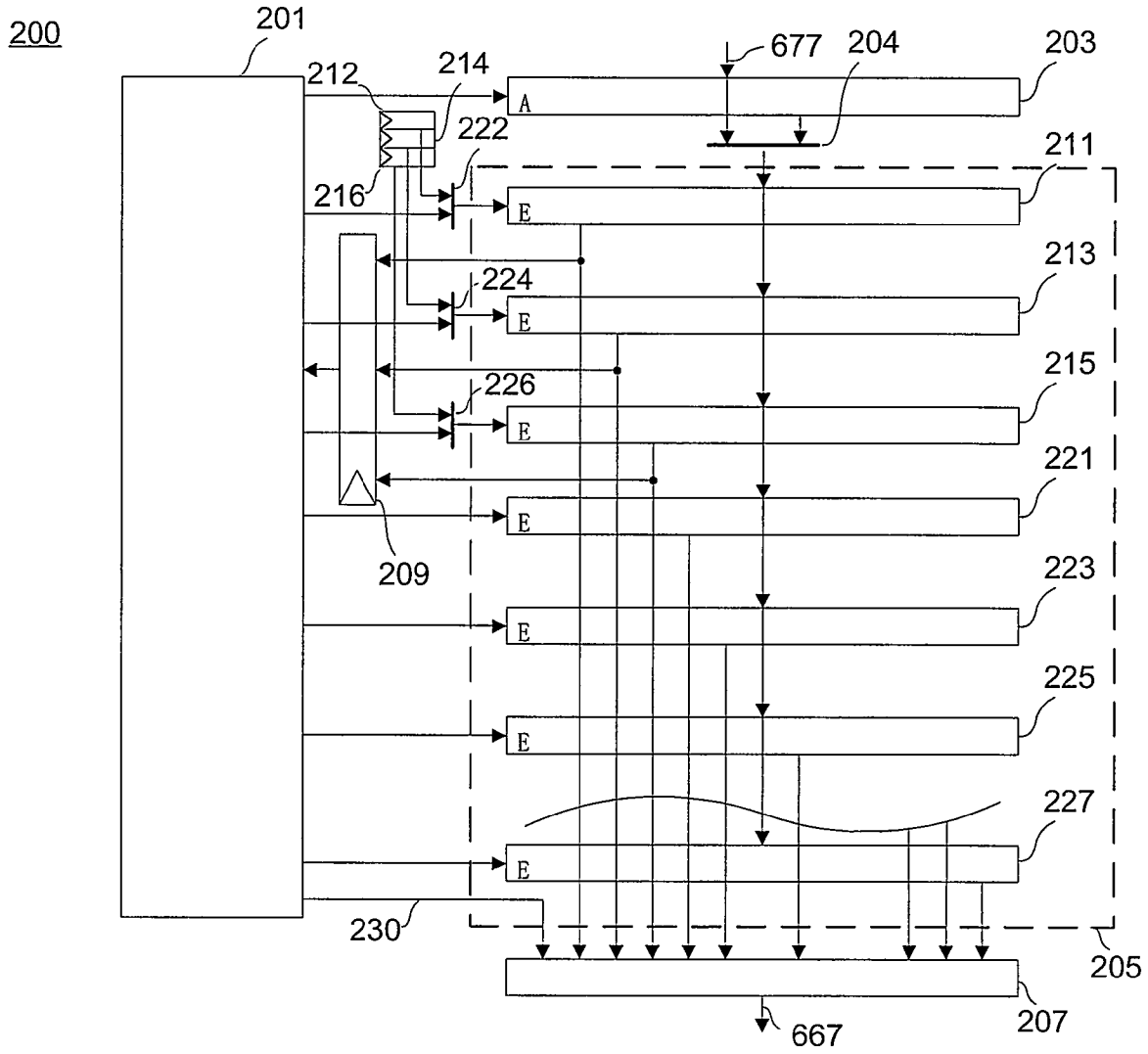

200
201
212
214
677
204
203
A
222
211
216
E
213
224
226
215
209
221
223
225
227
230
205
207
667

图 2

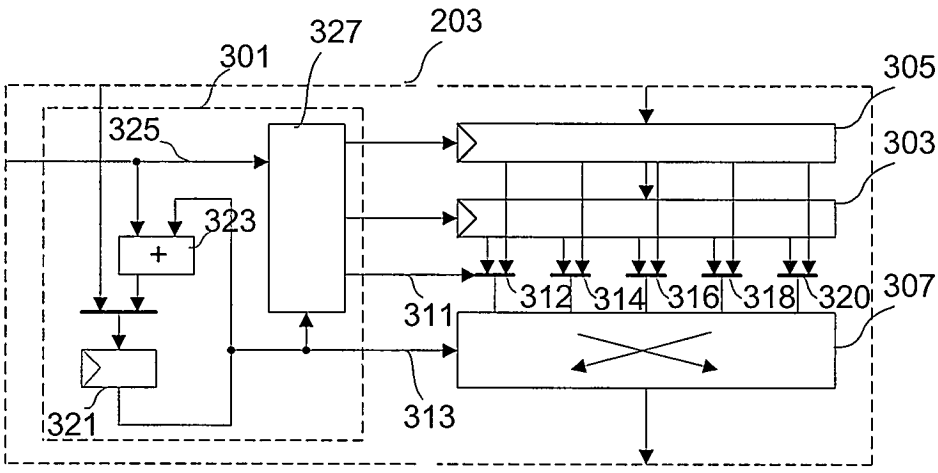

203
327
301
305
325
303
323
+
312
314
316
318
320
307
311
313
321

图 3A

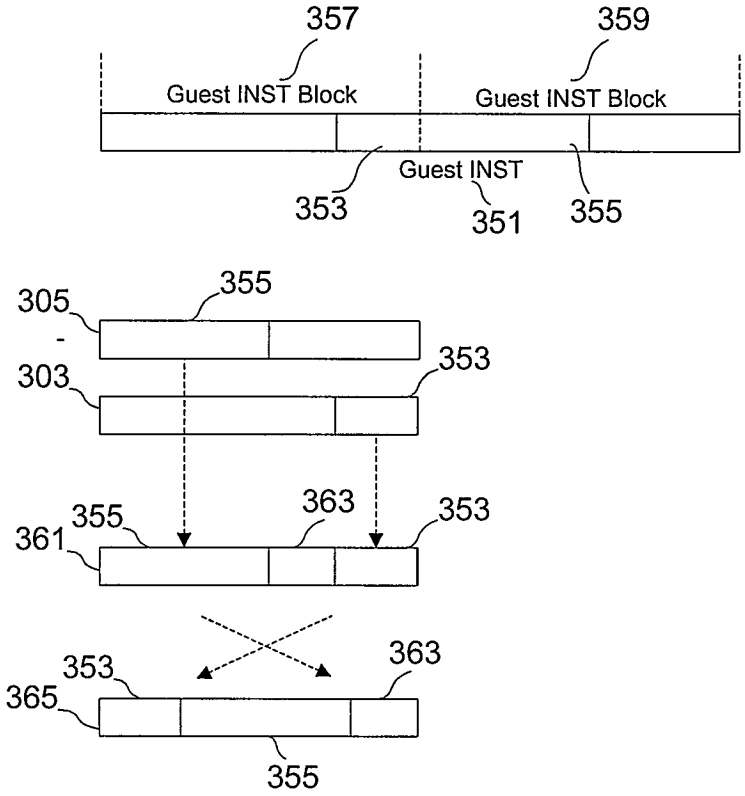

357
359
Guest INST Block
Guest INST Block
Guest INST
353
351
355
305
355
353
303
355
363
353
361
353
363
365
355

图 3B

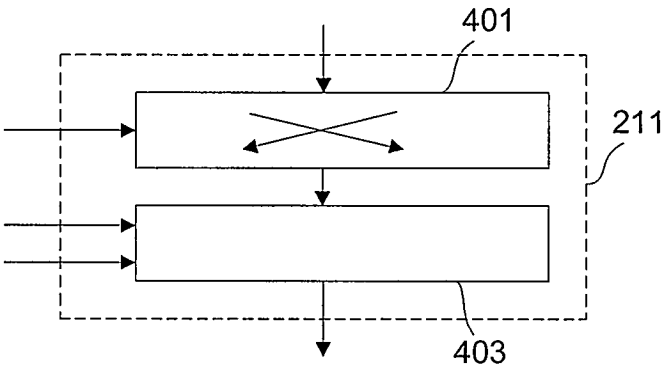

401
211
403

图 4A

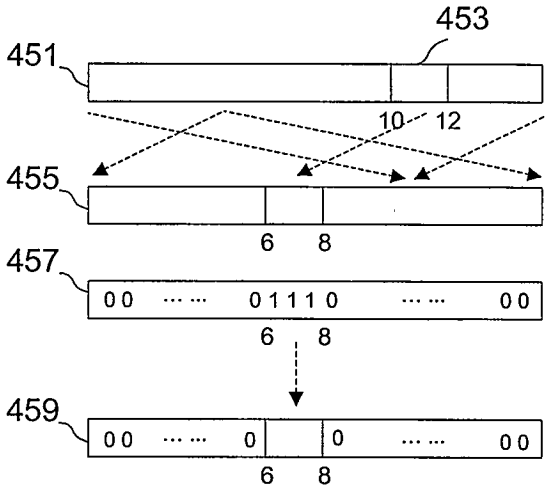

453
451
10
12
455
6
8
457
00 …… 01110 …… 00
6
8
459
00 …… 0
0 …… 00
6
8

图 4B

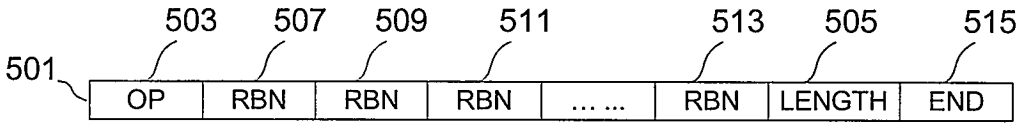

503
507
509
511
513
505
515
501
OP
RBN
RBN
RBN
… …
RBN
LENGTH
END

图 5A

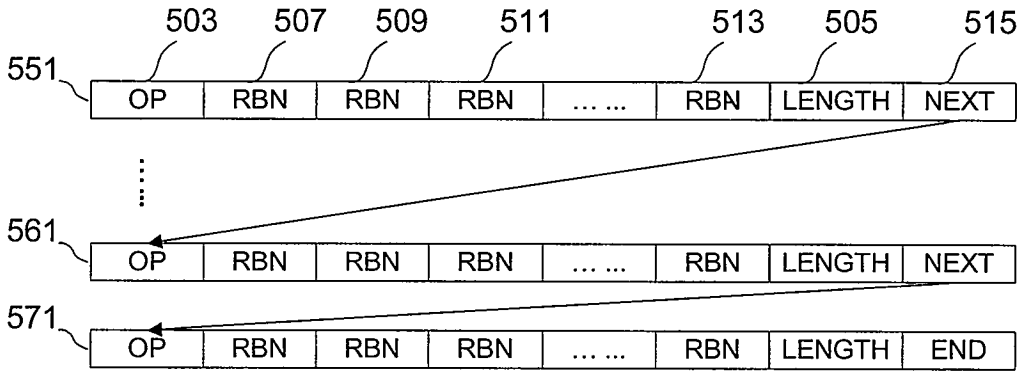

503
507
509
511
513
505
515
551
OP
RBN
RBN
RBN
… …
RBN
LENGTH
NEXT
561
OP
RBN
RBN
RBN
… …
RBN
LENGTH
NEXT
571
OP
RBN
RBN
RBN
… …
RBN
LENGTH
END

图 5B

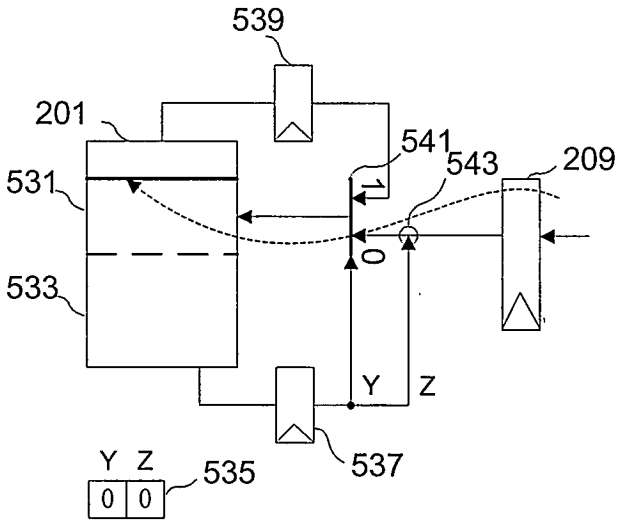

539
201
541
543
209
531
1
0
533
Y
Z
537
Y Z
0 0
535

图 5C

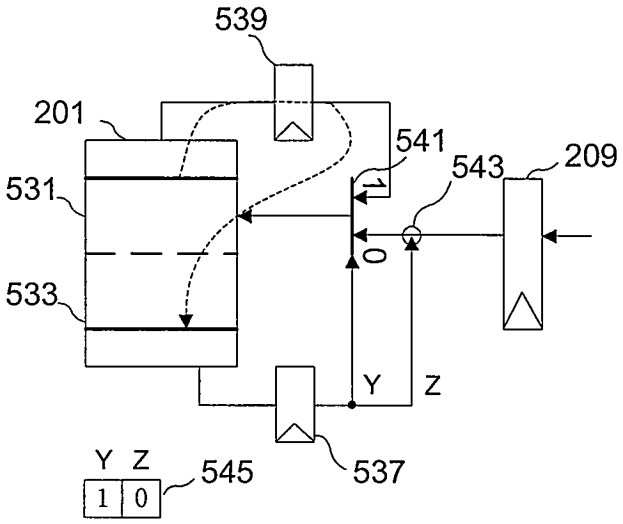

539
201
541
543
209
531
1
0
533
Y
Z
537
Y Z
1 0
545

图 5D

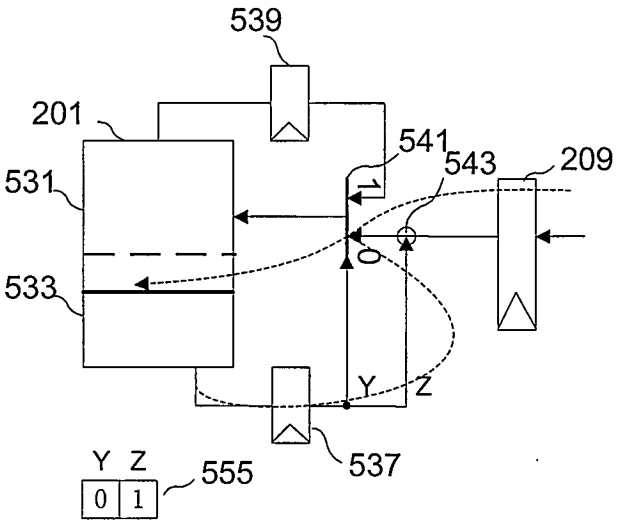

539
201
541
543
209
531
533
Y
Z
Y Z
0 1
555
537

图 5E

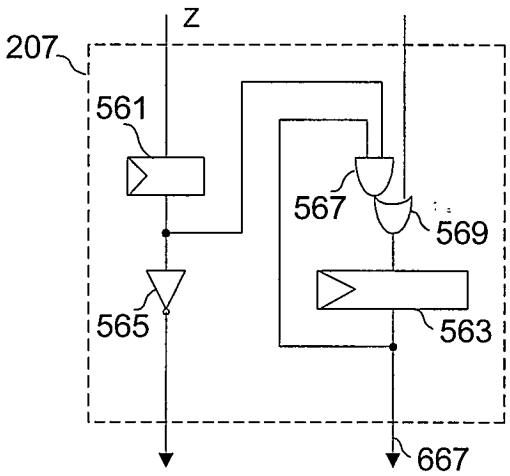

Z
207
561
567
569
565
563
667

图 5F

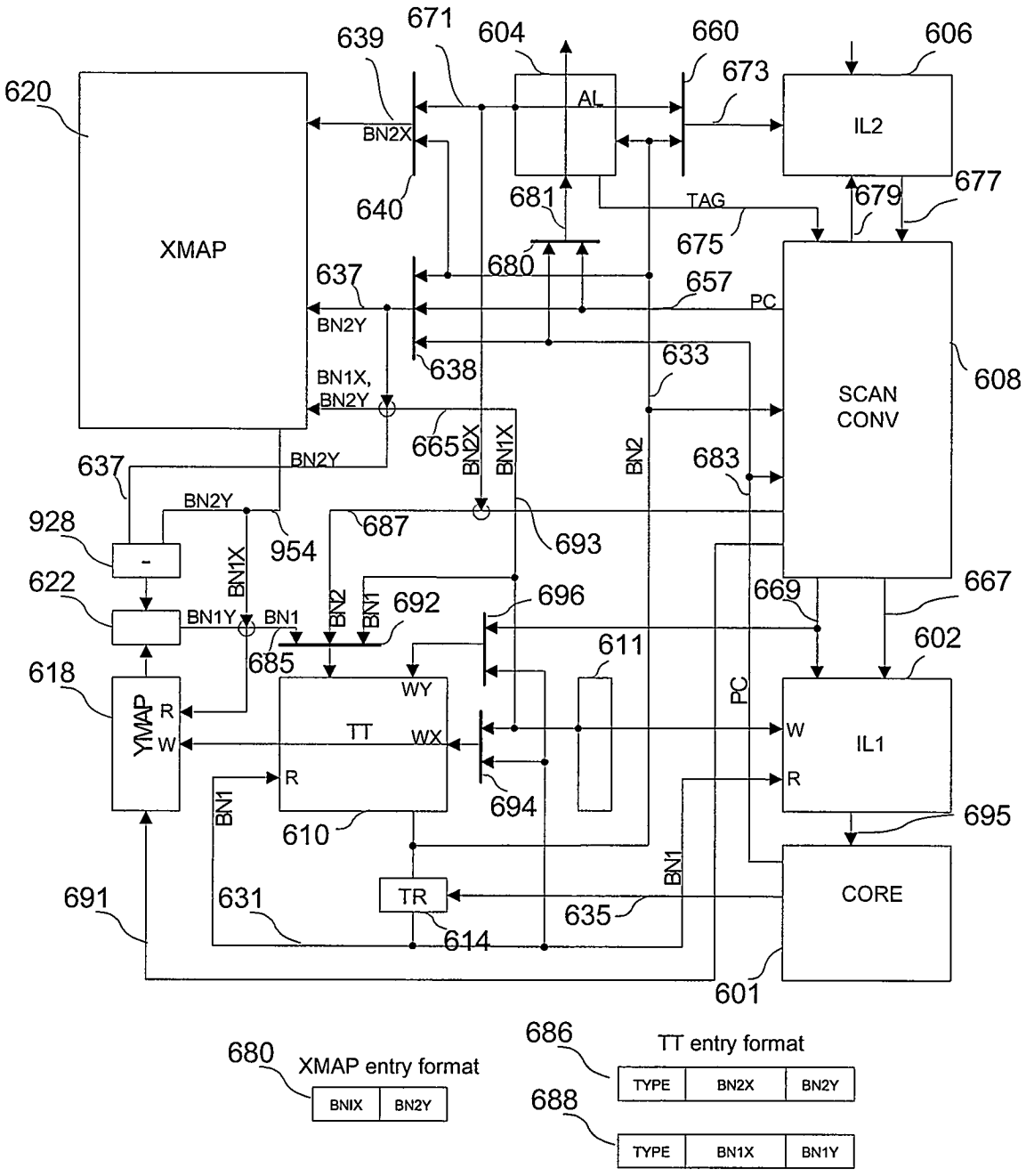

620
XMAP
639
671
604
660
606
673
AL
IL2
BN2X
640
681
TAG
679
677
675
680
637
657
PC
BN2Y
638
633
608
SCAN
CONV
BN1X,
BN2Y
665
BN2X
BN1X
BN2
683
637
BN2Y
928
954
687
693
622
BN1Y
BN1
692
696
669
667
685
611
602
618
YMAP
R
W
WY
TT
WX
IL1
694
610
695
691
631
TR
635
CORE
614
601
680
XMAP entry format
BNIX
BN2Y
686
TT entry format
TYPE
BN2X
BN2Y
688
TYPE
BN1X
BN1Y

图 6

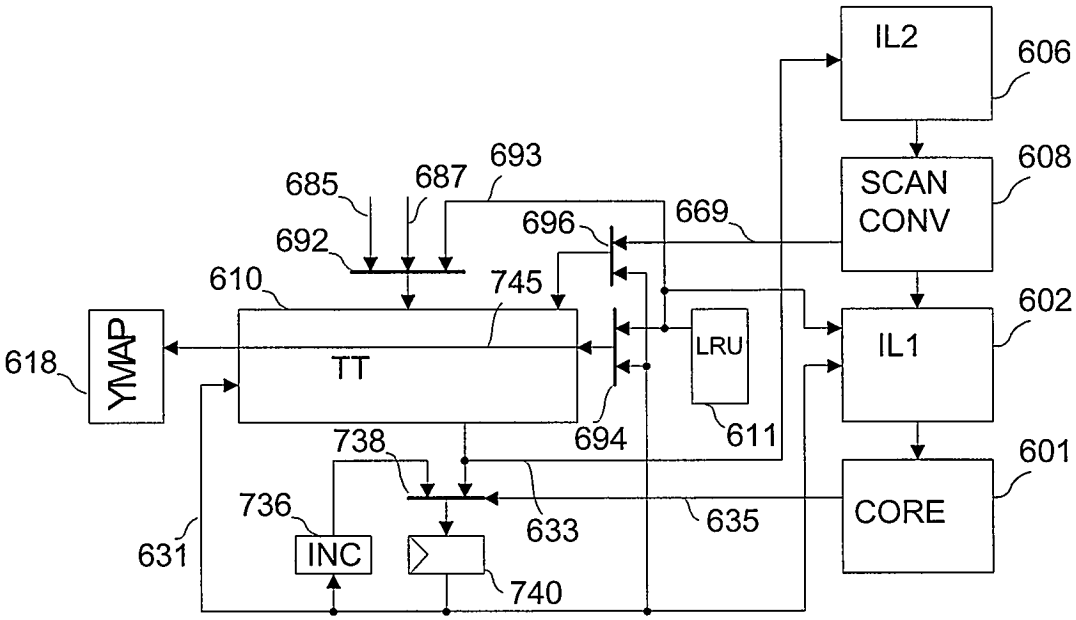

IL2
606
SCAN
CONV
608
IL1
602
CORE
601
685
687
693
692
696
669
610
745
618
YMAP
TT
LRU
611
694
738
736
INC
740
633
635
631

图 7A

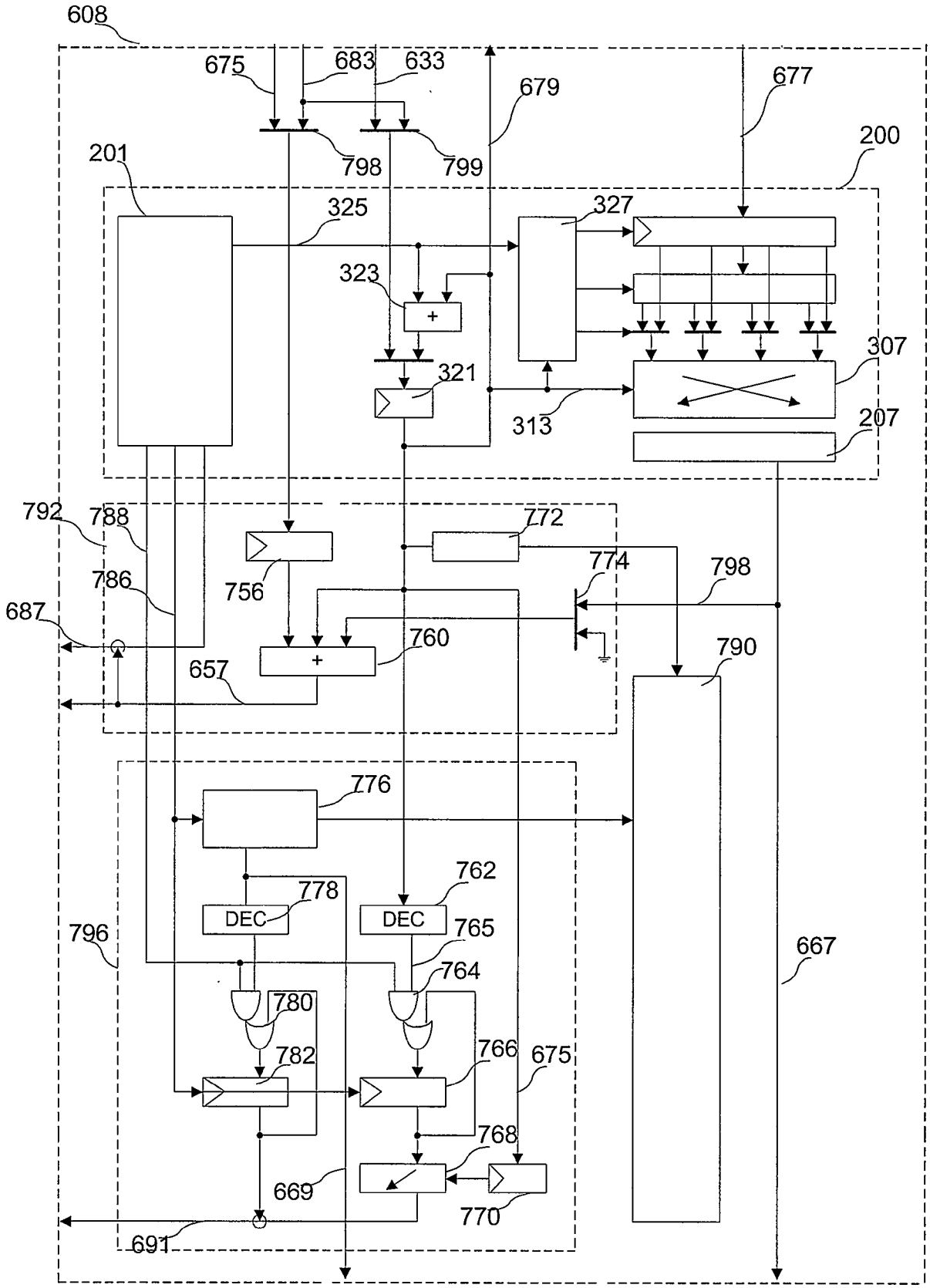

608
675
683
633
679
677
200
201
798
799
325
327
323
+
321
307
313
207
792
788
772
774
798
786
756
687
760
790
657
776
762
778
DEC
DEC
796
765
764
667
780
782
766
675
768
669
770
691

图 7B

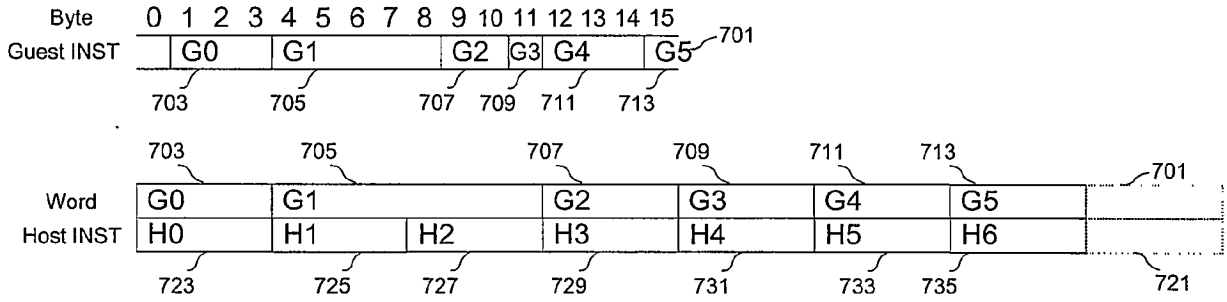

Byte
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Guest INST
G0
G1
G2
G3
G4
G5
701
703
705
707
709
711
713
Word
Host INST
H0
H1
H2
H3
H4
H5
H6
721
723
725
727
729
731
733
735

图 8A

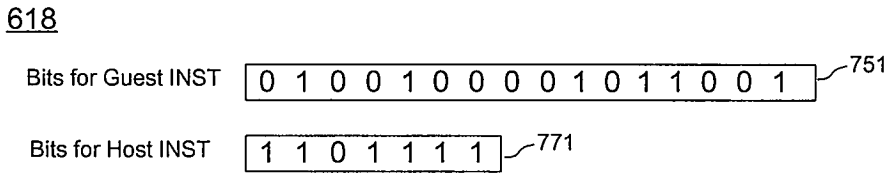

618
Bits for Guest INST
0 1 0 0 1 0 0 0 0 1 0 1 1 0 0 1
751
Bits for Host INST
1 1 0 1 1 1 1
771

图 8B

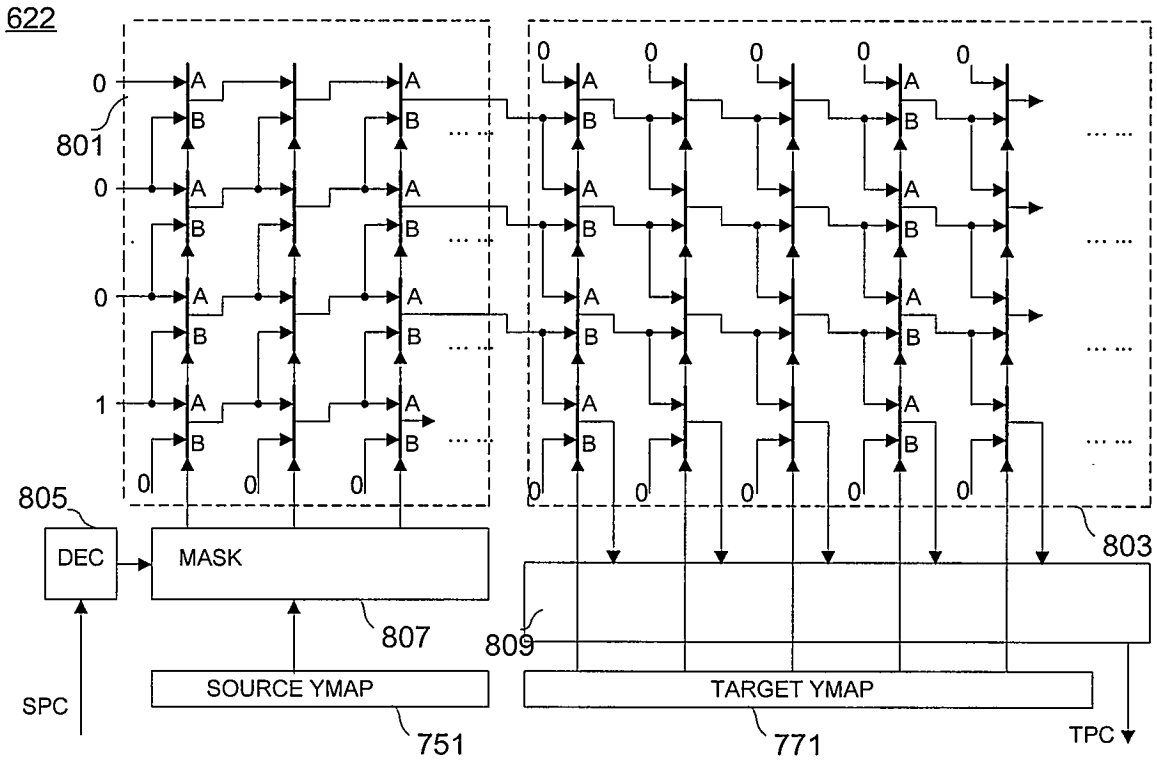

622
801
805
DEC
MASK
807
SPC
SOURCE YMAP
751
809
TARGET YMAP
771
803
TPC

图 8C

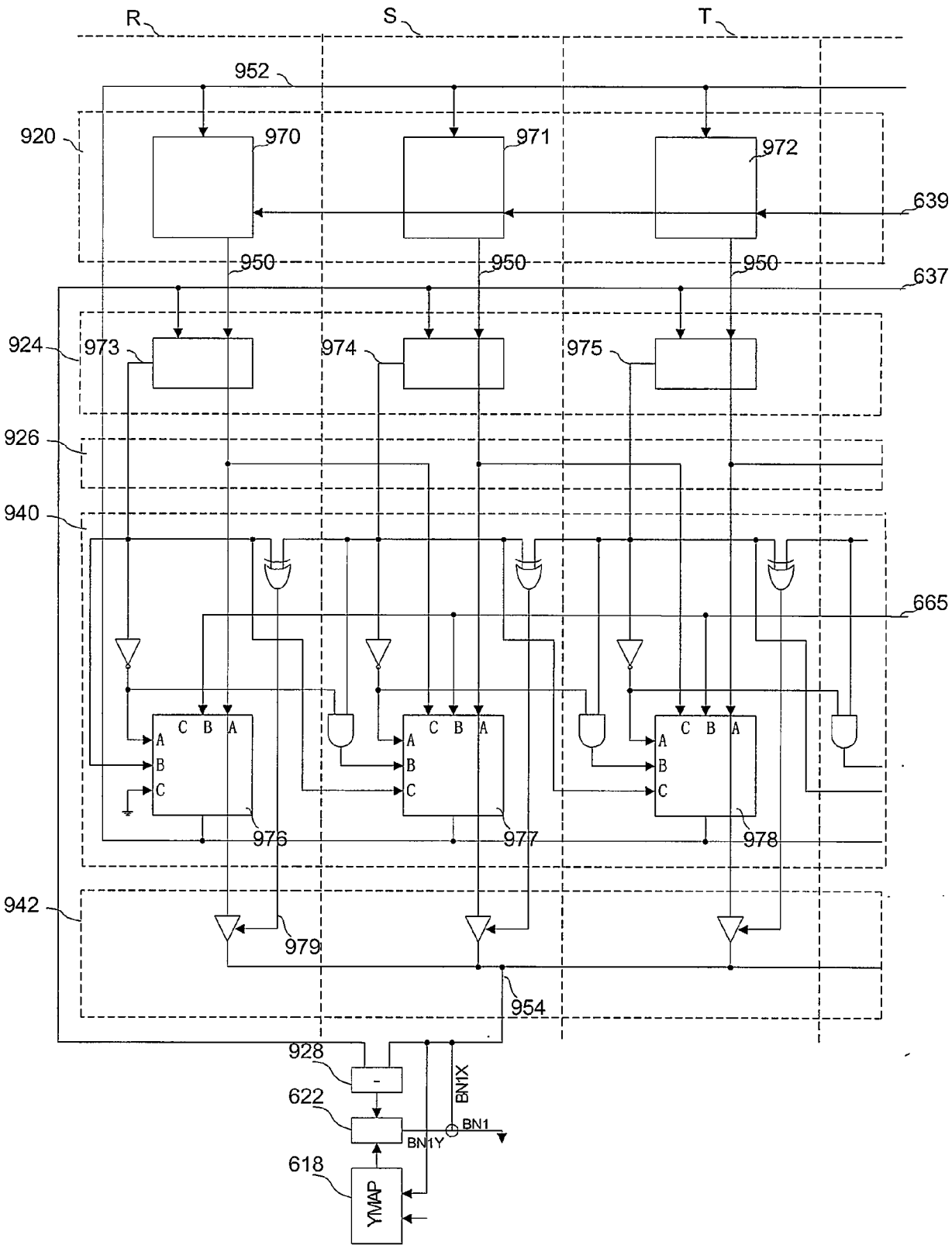

R
S
T
952
920
970
971
972
639
950
950
950
637
924
973
974
975
926
940
665
C B A
A
B
C
976
977
978
942
979
954
928
622
618
BN1X
BN1
BN1Y
YMAP

图 8D

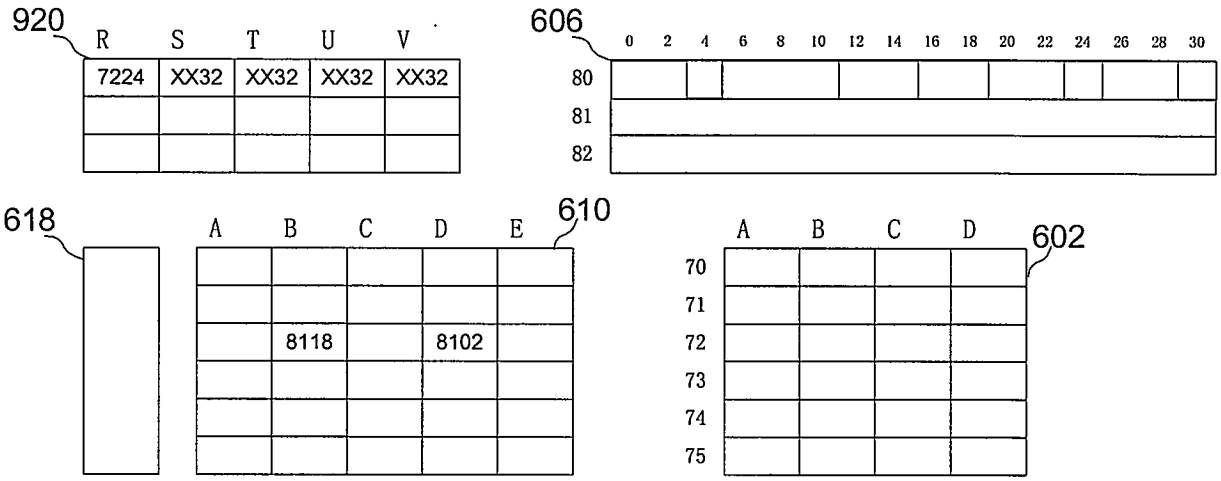

920
R S T U V
7224 XX32 XX32 XX32 XX32
606
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
80
81
82
618
A B C D E
610
8118
8102
A B C D
602
70
71
72
73
74
75

图 9A

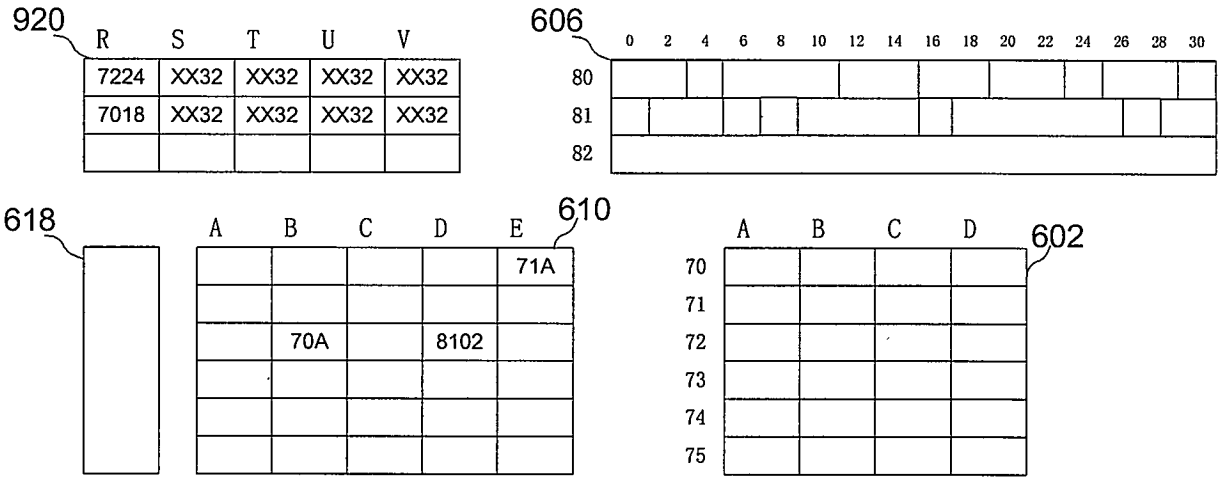

920
R S T U V
7224 XX32 XX32 XX32 XX32
7018 XX32 XX32 XX32 XX32
606
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
80
81
82
618
A B C D E
610
71A
70A
8102
A B C D
602
70
71
72
73
74
75

图 9B

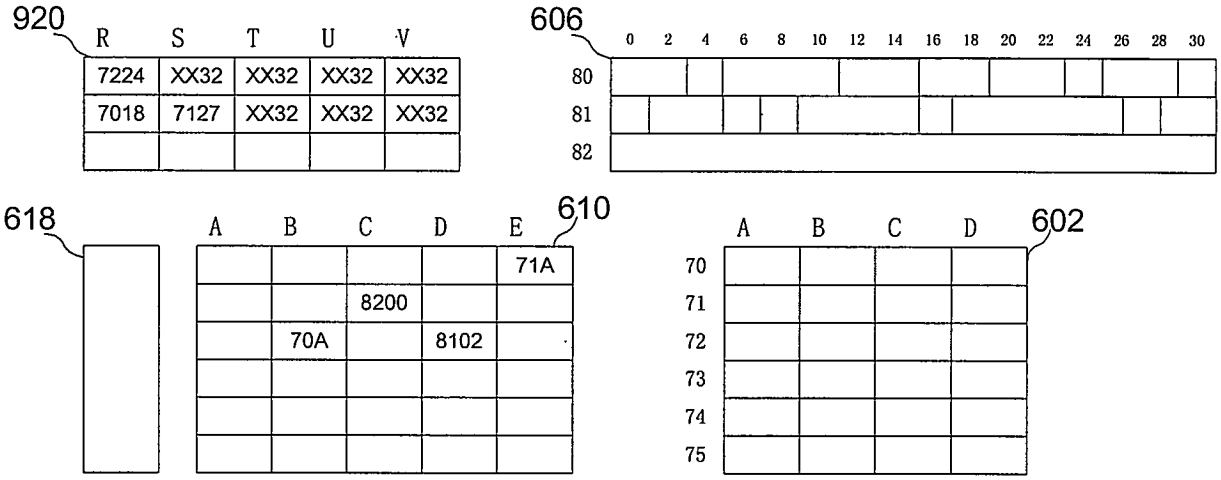

920
R S T U V
7224 XX32 XX32 XX32 XX32
7018 7127 XX32 XX32 XX32
606
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
80
81
82
618
A B C D E
610
71A
8200
70A 8102
A B C D
602
70
71
72
73
74
75

图 9C

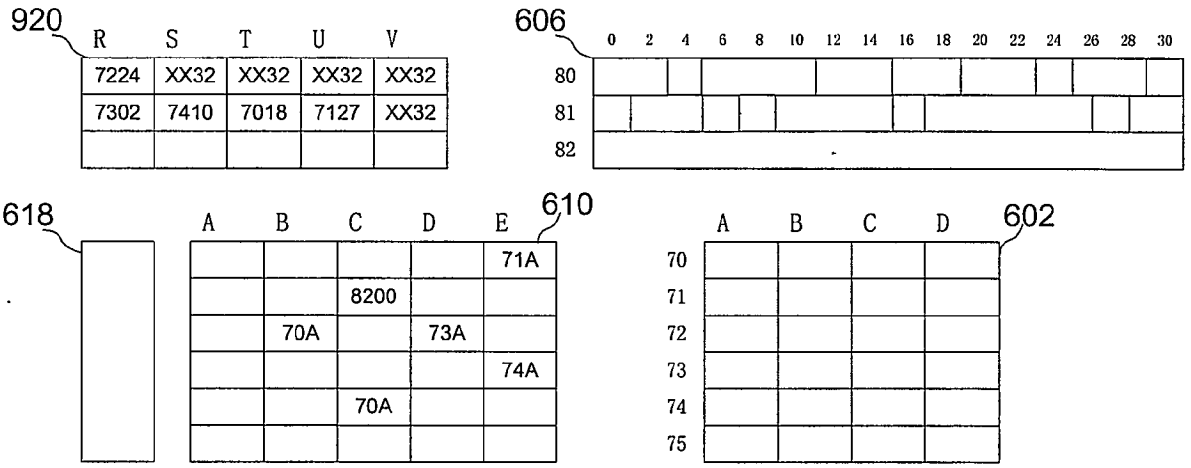

920
R S T U V
7224 XX32 XX32 XX32 XX32
7302 7410 7018 7127 XX32
606
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
80
81
82
618
A B C D E
610
71A
8200
70A 73A
74A
70A
A B C D
602
70
71
72
73
74
75

图 9D

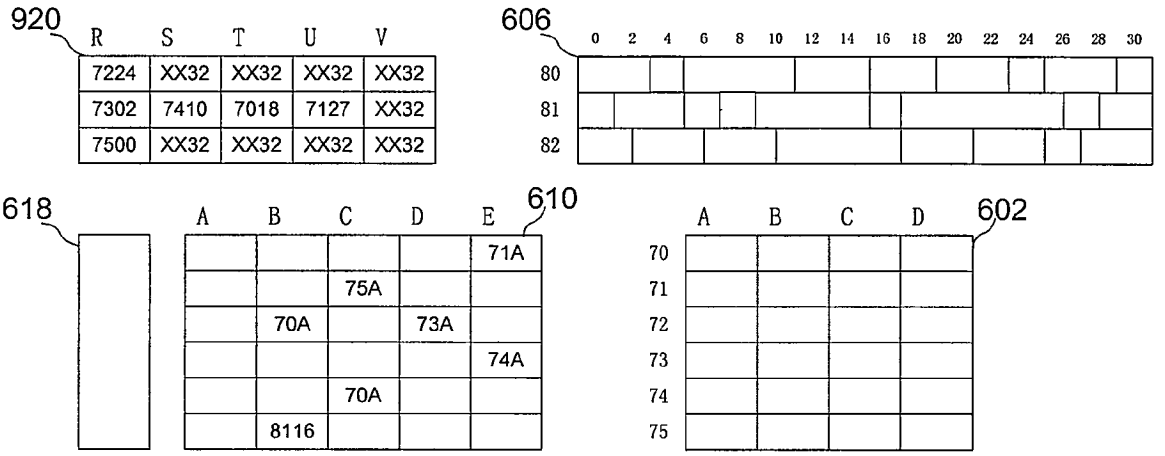

920
R S T U V
7224 XX32 XX32 XX32 XX32
7302 7410 7018 7127 XX32
7500 XX32 XX32 XX32 XX32
606
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
80
81
82
618
610
A B C D E
71A
75A
70A 73A
74A
70A
8116
602
A B C D
70
71
72
73
74
75

图 9E

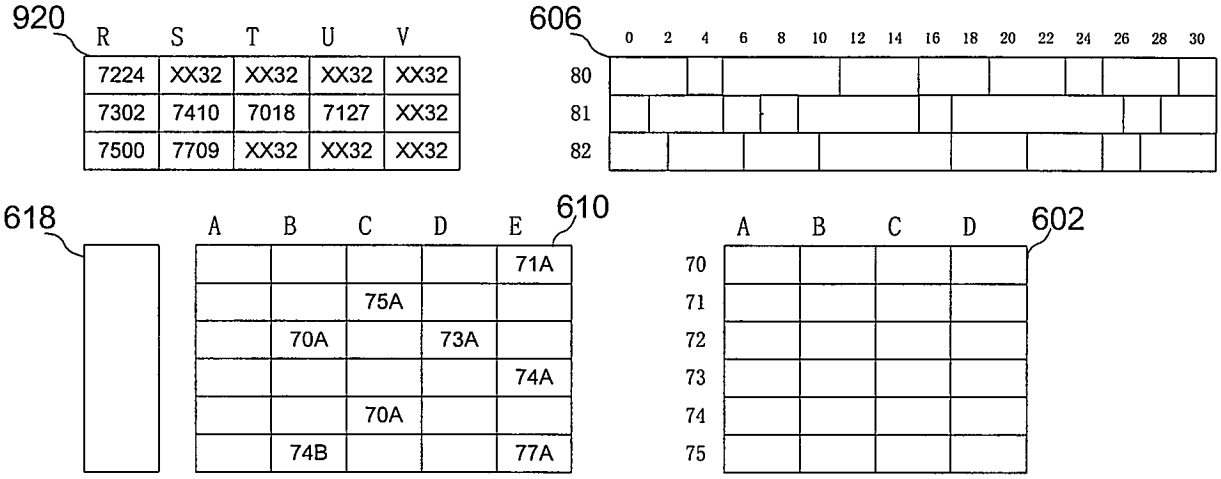

920
R S T U V
7224 XX32 XX32 XX32 XX32
7302 7410 7018 7127 XX32
7500 7709 XX32 XX32 XX32
606
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
80
81
82
618
610
A B C D E
71A
75A
70A 73A
74A
70A
74B 77A
602
A B C D
70
71
72
73
74
75

图 9F

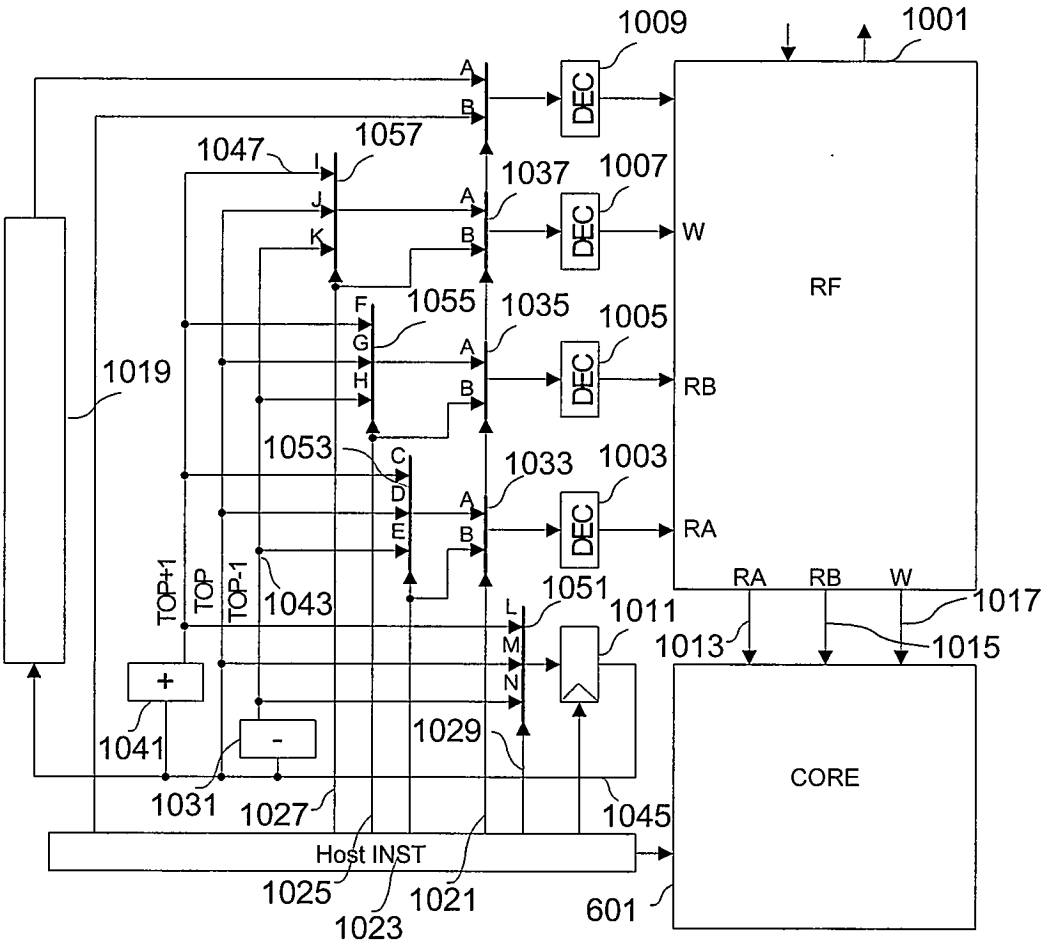

1009
1001
DEC
A
B
1057
1047
I
J
K
1037
1007
W
RF
1055
1035
1005
F
G
H
RB
1019
1053
1033
1003
C
D
E
RA
TOP+1
TOP
TOP-1
1043
1051
1011
L
M
N
RA
RB
W
1017
1013
1015
+
-
1041
1029
CORE
1031
1027
1045
Host INST
1025
1023
1021
601

图 10A

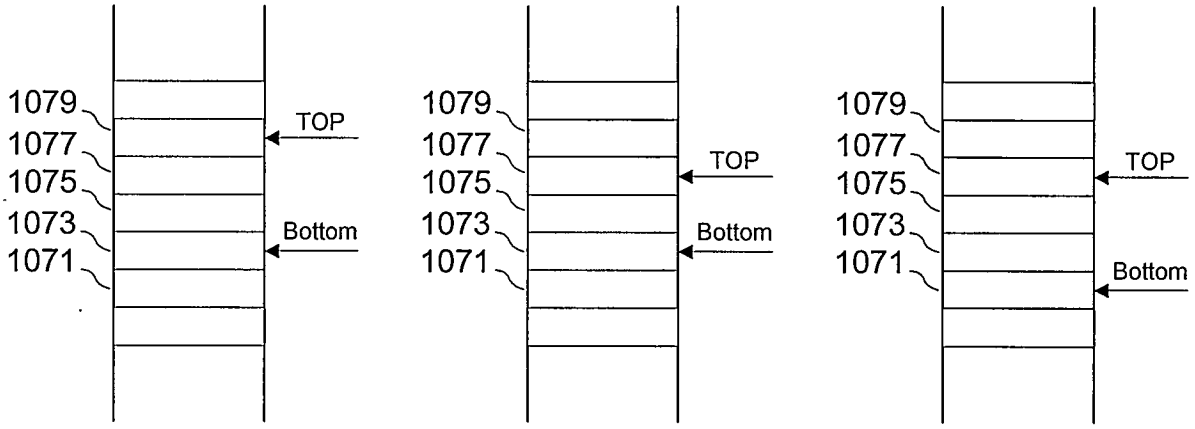

1079
1077
1075
1073
1071
TOP
Bottom
1079
1077
1075
1073
1071
TOP
Bottom
1079
1077
1075
1073
1071
TOP
Bottom

图 10B

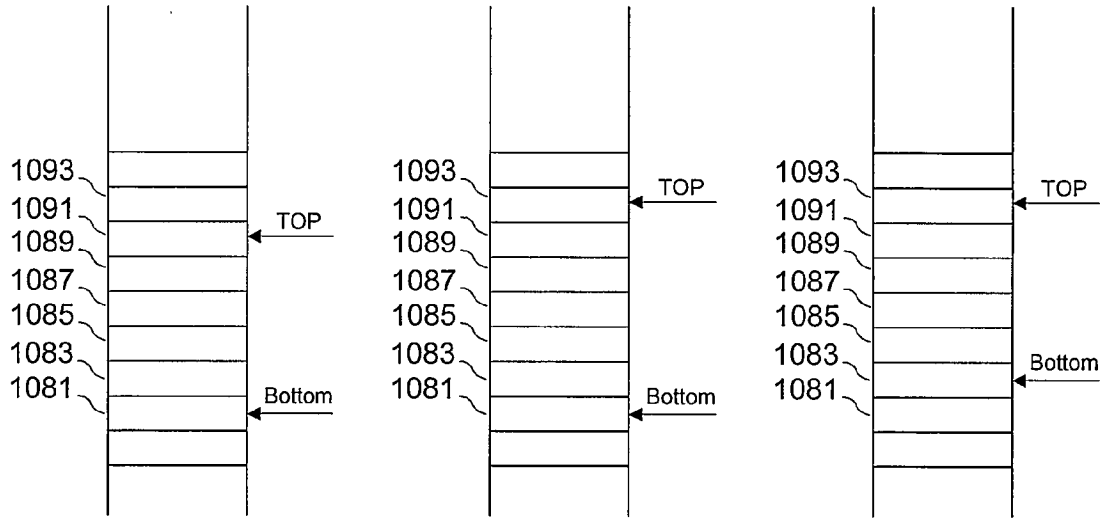

1093
1091
1089
1087
1085
1083
1081
TOP
Bottom
1093
1091
1089
1087
1085
1083
1081
TOP
Bottom
1093
1091
1089
1087
1085
1083
1081
TOP
Bottom

图 10C

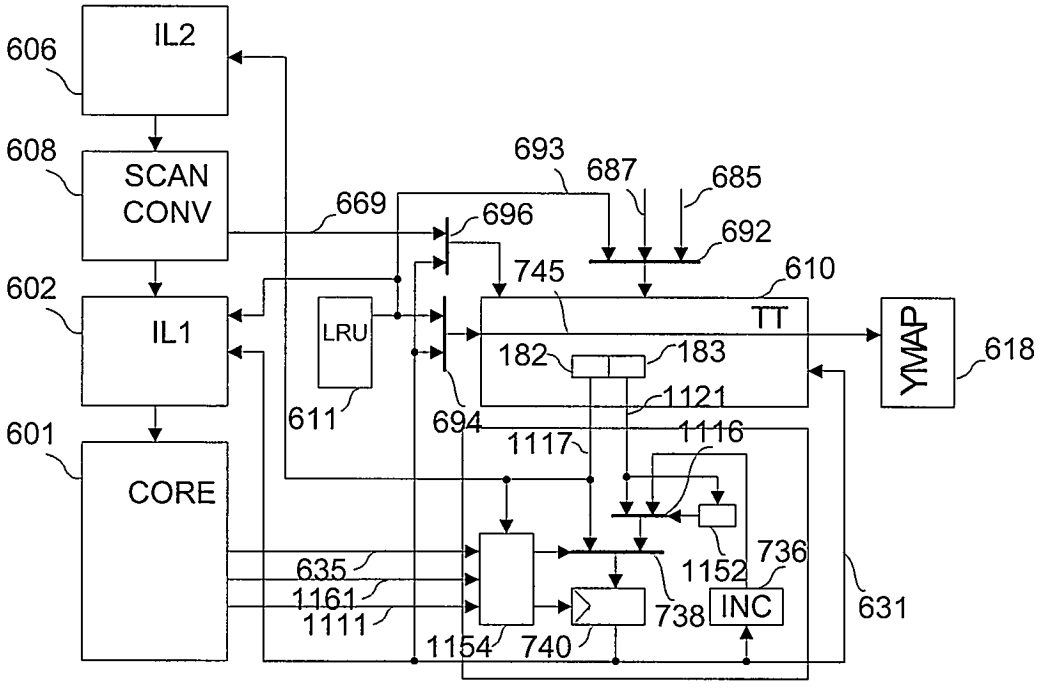

606
IL2
608
SCAN CONV
602
IL1
601
CORE
669
696
693
687
685
692
745
610
TT
YMAP
618
182
183
1121
1116
1117
611
694
LRU
635
1161
1111
1154
740
738
INC
736
1152
631

图 11A

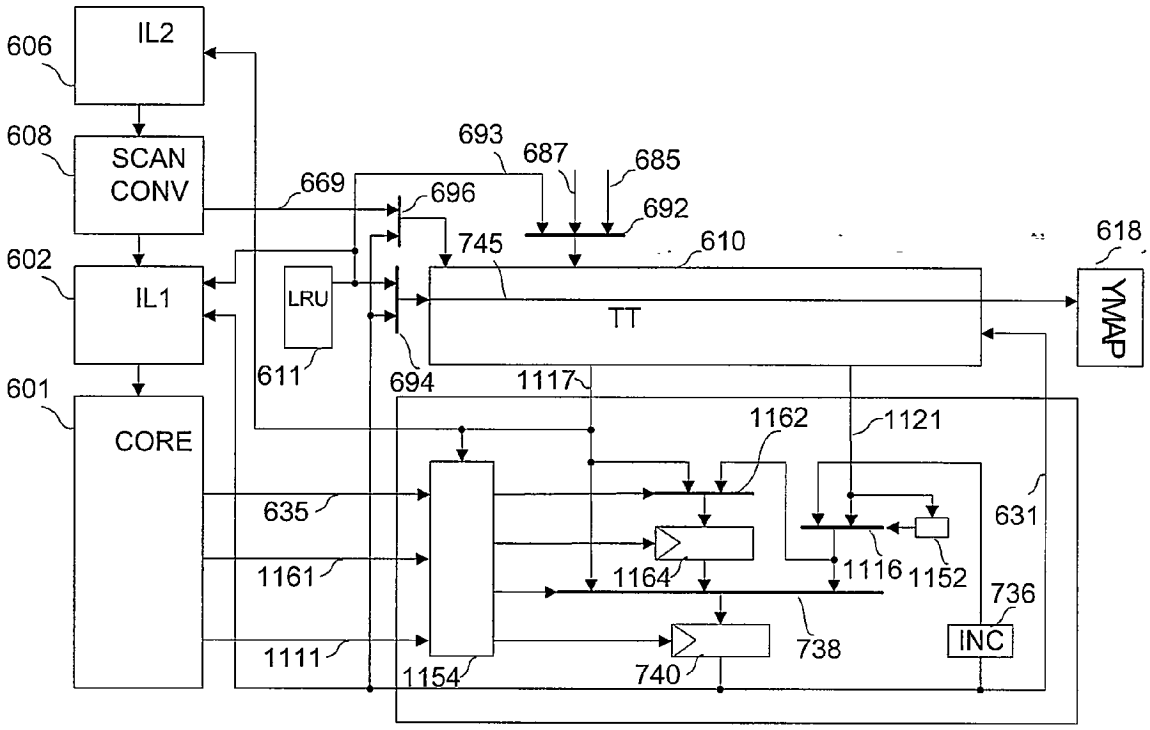


图 11B

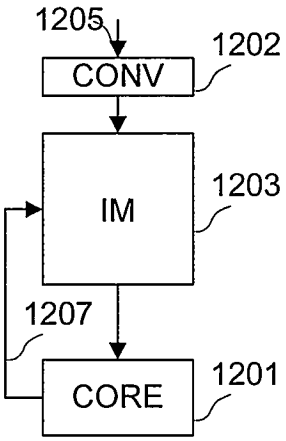


图 12

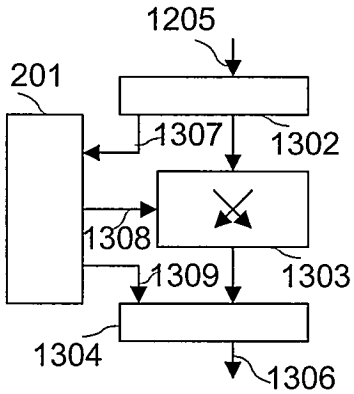


图 13A

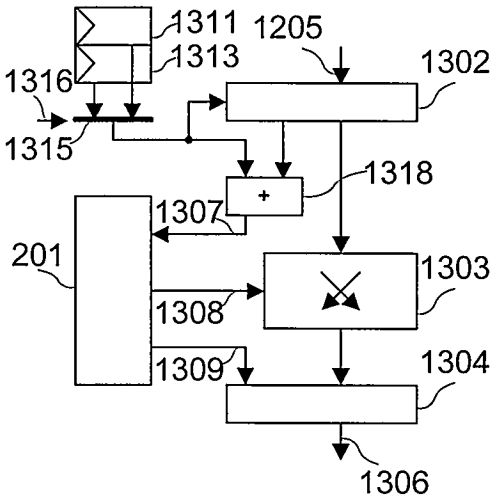

1311
1313
1205
1302
1316
1315
1318
1307
201
1303
1308
1309
1304
1306

图 13B

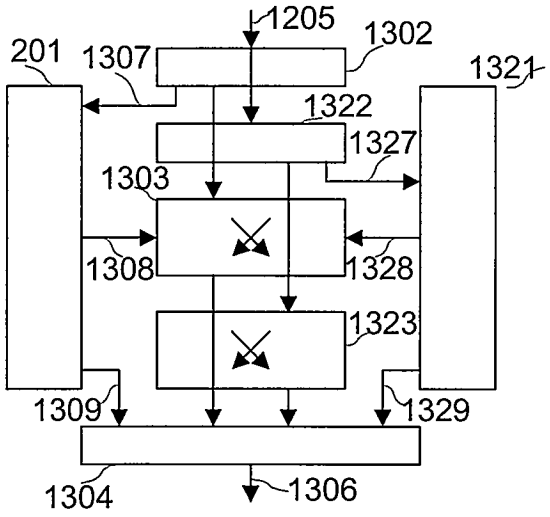

1205
1302
201
1307
1321
1322
1327
1303
1308
1328
1323
1309
1329
1304
1306

图 13C

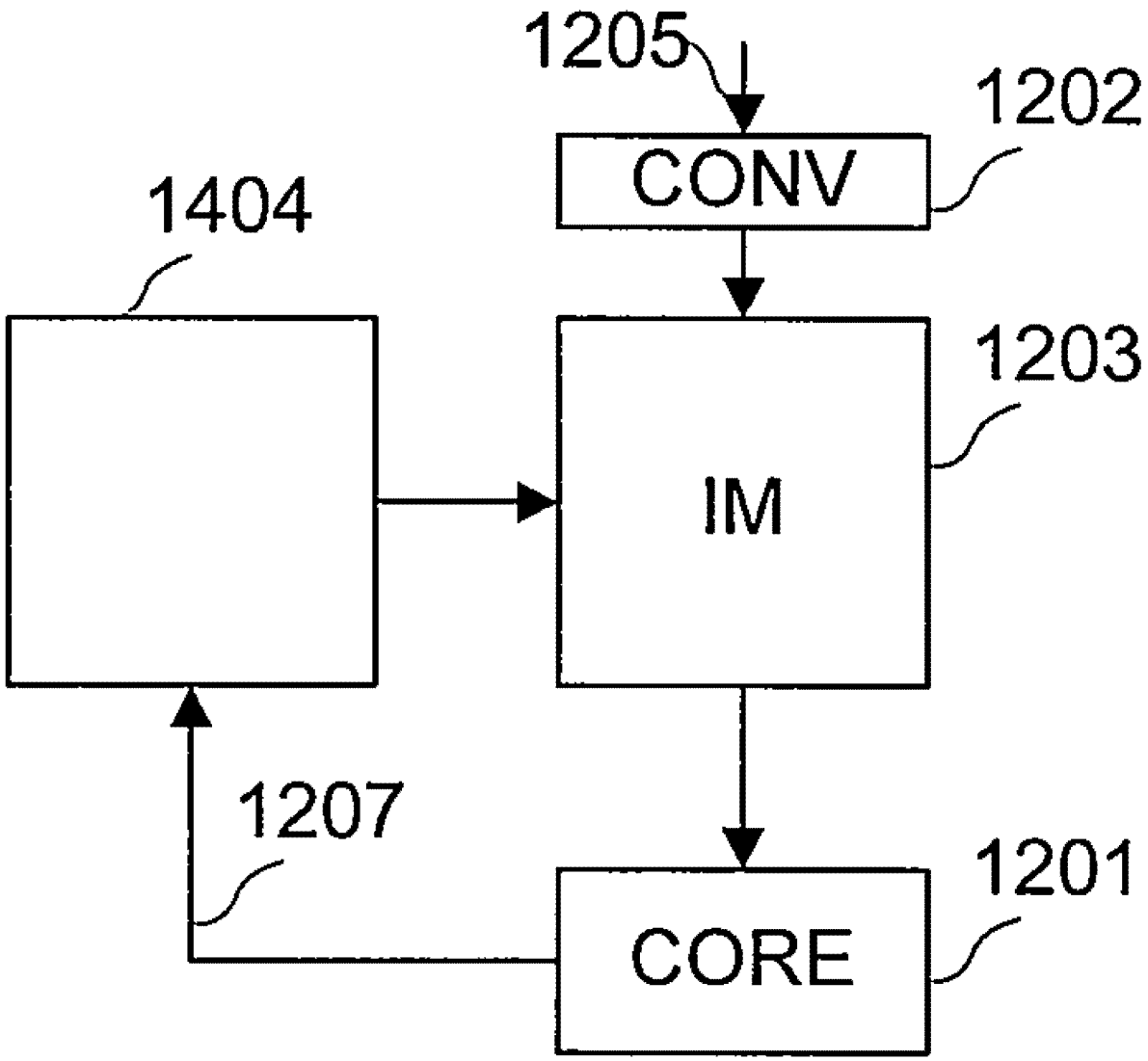

1205
1202
CONV
1404
1203
IM
1207
1201
CORE

图 14

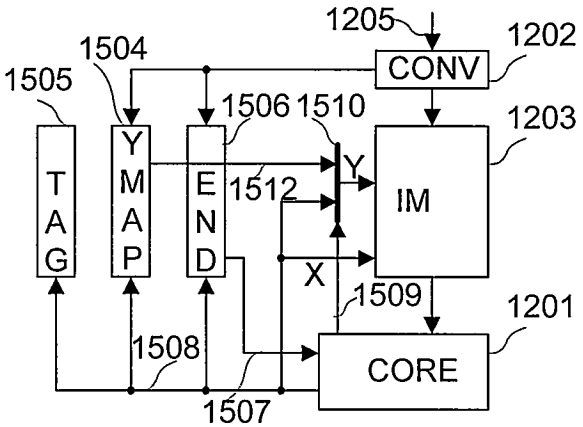

1205
1202
CONV
1504
1505
1506
1510
1203
TAG
YMAP
END
Y
1512
IM
X
1509
1201
1508
CORE
1507

图 15

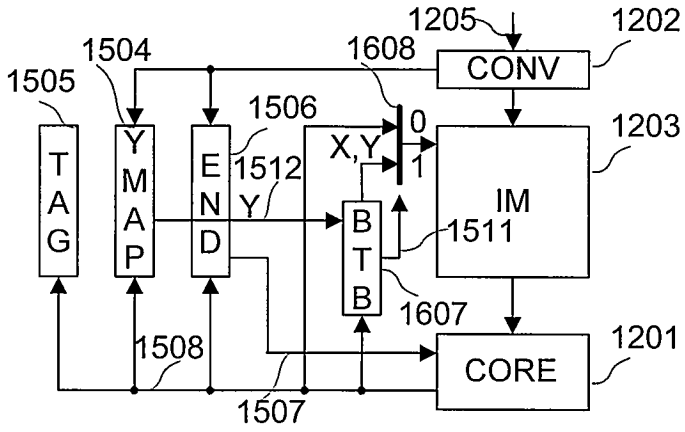


图 16

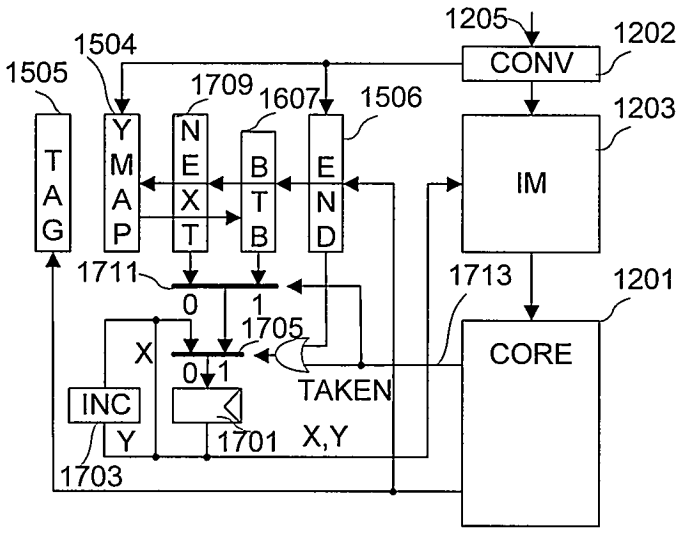


图 17

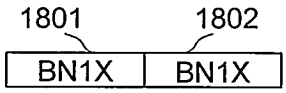


图 18A

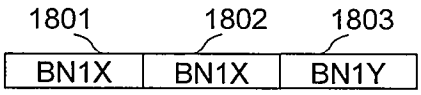


图 18B

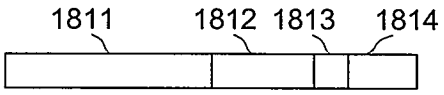


图 18C

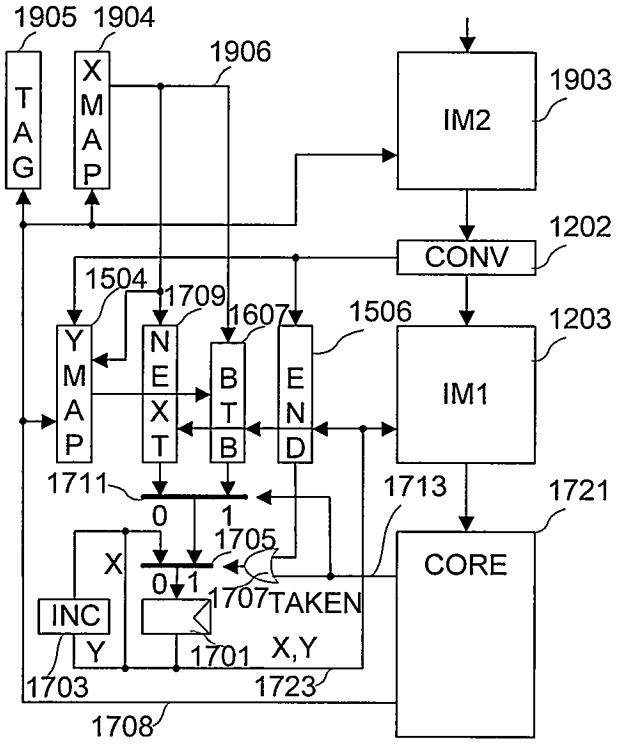


图 19

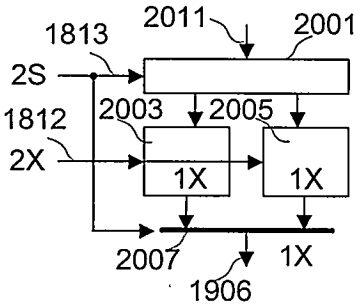


图 20

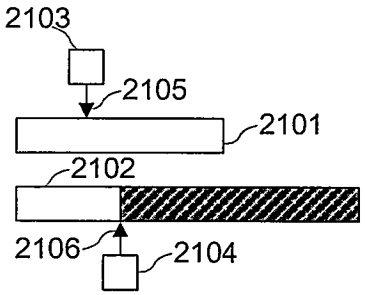


图 21

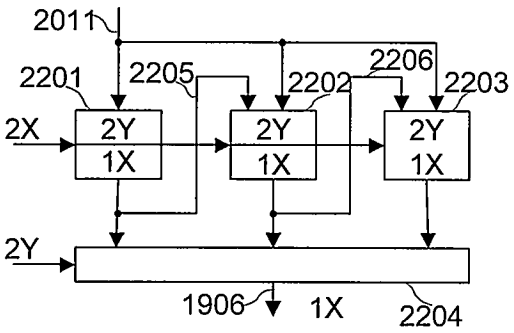


图 22

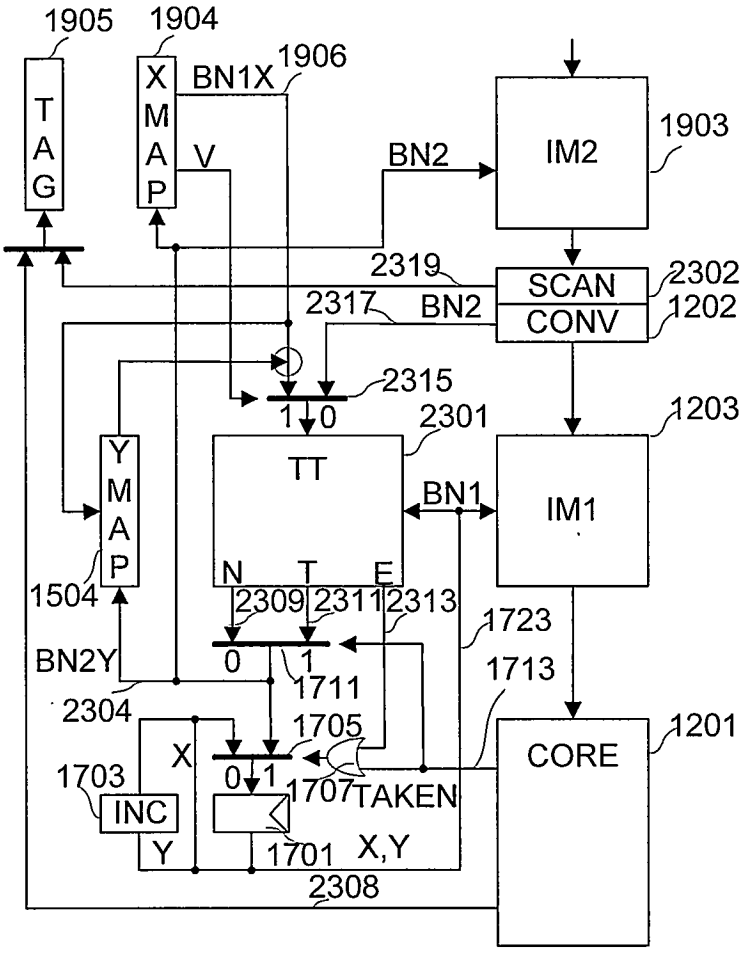

1905
1904
1906
TAG
XMAP
BN1X
V
BN2
IM2
1903
2319
SCAN
CONV
2302
1202
2317
BN2
2315
1 0
2301
1203
YMAP
TT
BN1
IM1
1504
N T E
2309 2311 2313
1723
BN2Y
0 1
1711
1713
2304
1705
X
0 1
1201
CORE
1703
INC
1707
TAKEN
Y
1701
X,Y
2308

图 23

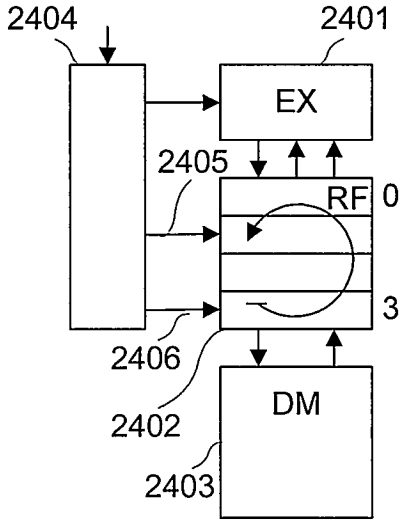

2404
2401
EX
2405
RF
0
3
2406
2402
DM
2403

图 24