

[19] 中华人民共和国国家知识产权局

[51] Int. Cl⁷

G06F 9/30

G06F 9/45



[12] 发明专利申请公开说明书

[21] 申请号 02142691.0

[43] 公开日 2003 年 4 月 30 日

[11] 公开号 CN 1414468A

[22] 申请日 2002.9.17 [21] 申请号 02142691.0

[30] 优先权

[32] 2002. 5. 9 [33] US [31] 10/144,595

[71] 申请人 智慧第一公司

地址 美国加利福尼亚

[72] 发明人 G·葛兰·亨利 罗德·E·胡克

泰瑞·派克斯

[74] 专利代理机构 隆天国际专利商标代理有限公司

代理人 陈红 楼仙英

权利要求书 4 页 说明书 13 页 附图 7 页

[54] 发明名称 延伸微处理器指令集的装置及方法

[57] 摘要

本发明涉及一种延伸微处理器指令集的装置及方法。该装置包括一转译逻辑与一延伸执行逻辑。转译逻辑将一延伸指令转译为对应的微指令。该延伸指令具有一延伸前置码与一延伸指令标志。该延伸前置码用以指示于该延伸指令所指定的运算执行时，需使用一架构延伸项。该延伸指令标志则用以指出该延伸指令前置码，其中延伸指令标志是该微处理器指令集内另一架构化地指定的运算码。延伸执行逻辑耦接至转译逻辑，用以接收该对应的微指令，并于该运算执行时使用该架构延伸项。



ISSN 1008-4274

1. 一种延伸微处理器指令集的装置，其特征在于，它包括：
一 转译逻辑，用以将一延伸指令转译成对应的微指令，其中该延伸指令
5 包括：
一 延伸前置码，用以指示在该延伸指令所指定的一运算执行时，要使用
一 架构延伸项；以及
一 延伸指令标志，用以指出该延伸前置码，其中该延伸指令标志是该微
处理器指令集内另一架构化地指定的运算码；以及
10 一 延伸执行逻辑，耦接至该转译逻辑，用于接收该对应的微指令，并于
执行该运算时使用该架构延伸项。
2. 如权利要求 1 所述的装置，其特征在于，所述的延伸指令还包括依据
该微处理器指令集所架构化地指定的数个数据项。
3. 如权利要求 2 所述的装置，其特征在于，所述的些架构化地指定的数
15 据项包括一运算码项目，用以指定该运算。
4. 如权利要求 3 所述的装置，其特征在于，所述的延伸前置码指定了使用
该架构延伸项于该运算的执行时，所必需的信息。
5. 如权利要求 2 所述的装置，其特征在于，所述的延伸前置码包括数个
位，且其中这些位的逻辑状态指定了该微处理器指令集的数个架构延伸项。
- 20 6. 如权利要求 1 所述的装置，其特征在于，所述的延伸前置码包括 8 个
位。
7. 如权利要求 1 所述的装置，其特征在于，所述的微处理器指令集包括
x86 指令集。
8. 如权利要求第 7 所述的装置，其特征在于，所述的延伸指令标志包括
25 x86 指令集的运算码 F1 (ICEBKPT)。
9. 如权利要求 1 所述的装置，其特征在于，所述的对应的微指令包括一
微运算码字段与一微运算码延伸项字段。
10. 如权利要求 9 所述的装置，其特征在于，所述的延伸执行逻辑使用
该微运算码延伸项字段，以决定该架构延伸项，且其中该延伸执行逻辑使用
30 该微运算码字段以决定该运算。

11. 如权利要求 1 所述的装置, 其特征在于, 所述的转译逻辑包括:
- 一逸出指令监测逻辑, 用于监测该延伸指令标志; 以及
 - 一延伸前置码译码逻辑, 耦接至该逸出指令监测逻辑, 用以转译该延伸前置码, 并对该对应微指令内的一微运算码延伸项字段进行组态, 该微运算
- 5 码延伸项字段指定了该架构延伸项。
12. 如权利要求 11 所述的装置, 其特征在于, 所述的转译逻辑还包括:
- 一指令译码逻辑, 用以组态该对应微指令内的其它字段, 该其它字段依据该微处理器指令集指定该运算。
13. 一种用于微处理器中的指令延伸装置 (mechanism), 其特征在于,
- 10 它包括:
- 一指令延伸项, 组态为指示该微处理器于一指定运算执行时, 可使用一延伸架构特征, 其中该指令延伸项包括一指令集架构其中的一指令, 该指令后则接着一 n 位延伸特征前置码, 该指令指出该指令延伸项, 而该 n 位延伸特征前置码则指出该延伸架构特征; 以及
- 15 一转译器, 组态为接收该指令延伸项, 并产生一微指令序列, 该序列指示一延伸执行逻辑于执行该指定运算时, 应用该延伸架构特征。
14. 如权利要求 13 所述的指令延伸装置, 其特征在于, 还包括:
- 数个指令部分, 耦接至该指令延伸项, 组态为指定该指定运算。
15. 如权利要求 13 所述的指令延伸装置, 其特征在于, 所述的 n 位延伸
- 20 特征前置码的值是映像至数个补充该指令集架构的延伸架构特征。
16. 如权利要求 13 所述的指令延伸装置, 其特征在于, 所述的指令集架构是 x86 指令集架构。
17. 如权利要求 16 所述的指令延伸装置, 其特征在于, 所述的指令包括 x86 指令集架构的 ICEBKPT 指令 (即运算码 F1)。
- 25 18. 如权利要求 13 所述的指令延伸装置, 其特征在于, 所述的微指令序列包括一微运算码字段与一微运算码延伸项字段。
19. 如权利要求 13 所述的指令延伸装置, 其特征在于, 所述的转译器包括:
- 一逸出指令监测器, 用以监测该指令; 以及
- 30 一延伸前置码译码器, 耦接至该逸出指令监测器, 用以转译该 n 位延伸

特征前置码，并产生该微指令序列内一微运算码延伸项字段。

20. 一种指令集延伸模块，其特征在于，它包括：

一逸出标志，由一微处理器的转译器所接收，并指出一对应指令的附随部分是指定了一微处理器所要执行的一延伸运算，其中该逸出标志为一微处
5 理器指令集内一既有指令；以及

一延伸特征指定元，耦接至该逸出标志，且为该附随部分其中之一，用以指定该延伸运算的一补充部份，该补充部分是补充由该微处理器指令集所提供的部分。

21. 如权利要求 20 所述的指令集延伸模块，其特征在于，所述的附随部分
10 的剩余者是依据该微处理器指令集，组态为指定该延伸运算的一基本部分，且其中该基本部分与该补充部分一起指定该延伸运算。

22. 如权利要求 21 所述的指令集延伸模块，其特征在于，所述的附随部分的剩余者包括一运算码字节与选用的地址指定元字节。

23. 如权利要求 20 所述的指令集延伸模块，其特征在于，所述的微处
15 理器指令集是 x86 微处理器指令集。

24. 如权利要求 23 所述的指令集延伸模块，其特征在于，所述的既有指令包括 x86 指令集架构的 ICE BKPT 指令（即运算码 F1）。

25. 如权利要求 20 所述的指令集延伸模块，其特征在于，所述的微处理器转译器将该逸出标志与该附随部分转译成对应的微指令，该对应的微指令
20 是指示一延伸执行逻辑去执行该延伸运算。

26. 如权利要求 20 所述的指令集延伸模块，其特征在于，所述的微处理器转译器包括：

一逸出标志监测逻辑，用以监测该逸出标志，并指示该附随部分的转译动作需依据延伸转译常规（conventions）；以及

25 一译码逻辑，耦接至该逸出标志监测逻辑，用以依据该延伸转译常规执行该附随部分的转译动作，以致能该延伸运算。

27. 一种延伸微处理器指令集的方法，其特征在于，它包括：

提供一延伸指令，该延伸指令包括一延伸标志以及一延伸前置码，其中该延伸标志是该微处理器指令集的其中之一指令；

30 通过该延伸前置码与该延伸指令的其余部分指定所要执行的一延伸运

算，其中该延伸运算使用了一不能由该微处理器指令集中的指令来加以指定的微处理器的一架构特征；以及

在该延伸运算的执行中应用该架构特征。

28. 如权利要求 27 所述的方法，其特征在于，所述的指定所要执行的该延伸运算的动作包括：

首先指定该延伸运算的一常用（conventional）部分，该首先指定的动作使用了该微处理器指令集中另一个指令；以及

其次指定该延伸运算的一补充部分，该其次指定的动作使用了该延伸前置码。

29. 如权利要求 27 所述的方法，其特征在于，所述的提供延伸指令的动作包括使用 x86 微处理器指令集中一指令作为该延伸标志。

30. 如权利要求 29 所述的方法，其特征在于，所述的使用 x86 微处理器指令集的该指令的动作包括选取 x86 ICE BKPT 指令（即运算码 F1）作为该延伸标志。

31. 如权利要求 27 所述的方法，其特征在于，所述的应用该架构特征的动作包括：

将该延伸指令转译为指示该微处理器执行该延伸运算的微指令；以及于一延伸执行逻辑内，执行该微指令以进行该延伸运算。

32. 如权利要求 27 所述的方法，其特征在于，所述的转译延伸指令的动作包括：

于一转译逻辑内，监测该延伸标志；以及

依照延伸转译规则译码该延伸前置码与该延伸指令的其余部分。

延伸微处理器指令集的装置及方法

5 与相关申请的对照

(0001) 本发明是依据以下美国在先申请 (Provisional Application) 主张优先权：案号 60 / 356420，申请日为 2002 年 2 月 2 日，专利名称为“延伸微处理器指令集的装置及方法”。

10 (0002) 本发明与下列同在申请中的美国专利申请有关，其申请日与本发明相同，且具有相同的申请人与发明人。

| SERIAL NUMBER | DOCKET NUMBER | 专利名称 |
|---------------|---------------|-------------------|
| | CNTR: 2186 | 执行条件指令的装置及方法 |
| | CNTR: 2188 | 选择性地控制条件码回写的装置及方法 |
| | CNTR: 2189 | 增加微处理器的缓存器数量的装置 |
| | CNTR: 2198 | 选择性地控制结果回写的装置及方法 |

技术领域

15 (0003) 本发明涉及微电子领域，尤指一种能将微处理器指令集架构所未提供的架构特征，纳入该架构的技术。

背景技术

20 (0004) 自 70 年代初以来，微处理器的使用即呈指数般成长。从最早应用于科学与技术领域，到如今已从那些特殊领域引进商业的消费者领域，如桌上型与膝上型 (laptop) 计算机、视频游戏控制器以及许多其它常见的家用与商用装置等产品。

(0005) 随着过去三十年来使用上的爆炸性成长，在技术上也历经一相对应的提升，其特征在于对下列项目有着日益升高的要求：更快的速度、更强的寻址能力、更快的内存存取，更大的操作数、更多种运算（如浮点运算、单一

指令多重数据 (SIMD)、条件移动等) 以及附加的特殊运算 (如多媒体运算), 如此造就了该领域惊人的技术进展, 且都已应用于微处理器的设计, 像扩充管线化 (extensive pipelining)、超纯量架构 (super-scalar architecture)、快取结构、乱序处理 (out-of-order processing)、爆发式存取 (burst access)、分支预测 (branch predication) 以及假想执行 (speculative execution)。直言之, 现在的微处理器
5 比起 30 年前刚出现时, 呈现出惊人的复杂度, 且具备了强大的能力。

(0006) 但与许多其它产品不同的是, 有另一非常重要的因素已限制了, 并持续限制着微处理器架构的演进。现今微处理器会如此复杂, 一大部分得归因于这项因素, 即旧有软件的兼容性。在市场考量下, 所多制造商选择将新的
10 架构特征纳入最新的微处理器设计中, 但同时在这些新的产品中, 又保留了所有为确保兼容于较旧的, 即所谓“旧有” (legacy) 应用程序所必需的能力。

(0007) 这种旧有软件兼容性的负担, 没有其它地方, 会比在 x86-兼容的微处理器的发展史中更加显而易见。大家都知道, 现在的 32 / 16 位的虚拟模式 (virtual-mode) x86 微处理器, 仍可执行 1980 年代所撰写的 8 位真实模式
15 (real-mode) 的应用程序。而熟悉此领域技术者也承认, 有不少相关的架构“包袱”堆在 x86 架构中, 只为了支持与旧有应用程序及运作模式的兼容性。虽然在过去, 研发者可将新开发的架构特征加入既有的指令集架构, 但如今使用这些特征所凭借的工具, 即可编程的指令, 却变得相当稀少。更简单他说, 在某些重要的指令集中, 已没有“多余”的指令, 让设计者可藉以将更新的特征纳
20 入一既有的架构中。

(0008) 例如, 在 x86 指令集架构中, 已经没有任何一未定义的一字节大小的运算码状态, 是尚未被使用的。在主要的一字节大小的 x86 运算码图中, 全部 256 个运算码状态都已被既有的指令占用了。结果是, x86 微处理器的设计者现在必须在提供新特征与放弃旧有软件兼容性两者间作抉择。若要提供新的
25 的可编程特征, 则必须分派运算码状态给这些特征。若既有的指令集架构没有多余的运算码状态, 则某些既存的运算码状态必须重新定义, 以提供给新的特征。因此, 为了提供新的特征, 就得牺牲旧有软件兼容性了。

(0009) 所以, 我们所需要的是, 一种允许将新的架构特征纳入既有微处理器指令集架构的技术, 其中该指令集架构具有已完全占用的运算码结构, 而
30 该技术则仍保留旧有应用软件的兼容性。

发明内容

(0010) 本发明如同前述其它申请案，是针对上述及其它公知技术的问题与缺点加以克服。本发明提供一种更好的技术，用以扩充微处理器的指令集，使其超越现有的能力。在一具体实施例中，提供了用于扩充微处理器指令集的装置。该装置包括一转译逻辑 (translation logic) 与一延伸执行逻辑 (extended execution logic)。该转译逻辑将一延伸指令转译成对应的微指令 (micro instruction)。该延伸指令具一延伸前置码 (extended prefix) 与一延伸指令标志 (extended instruction tag)。该延伸前置码是指示在该延伸指令所指定运算的执行过程中，要使用一架构延伸项 (architectural extension)。该延伸指令标志则指出该延伸指令前置码，其中延伸指令标志是微处理器指令集内另一架构化地指定的运算码。该延伸执行逻辑耦接至转译逻辑，接收该对应的微指令，并于执行该运算时使用该架构延伸项。

(0011) 本发明的一个目的在于提出一种用于微处理器的指令延伸装置 (mechanism)。该装置具有一指令延伸项 (instruction extension) 与一转译器 (translator)。该指令延伸项指示该微处理器于一指定运算执行时，使用一延伸架构特征，其中该指令延伸项是包括一指令集架构其中一指令，其后则接着一 n 位的延伸特征前置码 (extended features prefix)，该指令指出该指令延伸项，而该 n 位的延伸特征前置码则指出该延伸架构特征。该转译器接收该指令延伸项，并产生一微指令序列。该序列指示一延伸执行逻辑于该指定运算执行时，应用该延伸架构特征。

(0012) 本发明的另一目的在于提供一种指令集延伸模块。该指令集延伸模块包括一逸出标志 (escape tag) 与一延伸特征指定元 (extended features specifier)。该逸出标志由一微处理器的转译器接收，并指出一对应指令的附随部分是指定了一微处理器所要执行的一延伸运算，其中该逸出标志为一微处理器指令集内的一既有指令。该延伸特征指定元耦接至该逸出标志，且为该附随部分其中之一，用以指定该延伸运算的补充部份。该补充部分是补充由该微处理器指令集所提供的部分。

(0013) 本发明的再一目的在于提供一种延伸微处理器指令集的方法。该方法包括提供一延伸指令，该延伸指令包括一延伸标志及一延伸前置码，其中该延伸标志是该微处理器指令集其中一指令；通过该延伸前置码与该延伸指令

的其余部分指定所要执行的一延伸运算，其中该延伸运算使用了微处理器的一架构特征，且该架构特征不能由该微处理器指令集的指令来加以指定；以及在该延伸运算执行中应用该架构特征。

5 附图说明

(0014) 本发明的前述与其它目的、特征及优点，在配合下列说明及附图后，将可获得更好的理解：

(0015) 图 1 为一相关技术的微处理器指令格式的方块图；

(0016) 图 2 为一表格，其描述一指令集架构的指令，如何对应至图 1
10 指令格式内一运算码字节的位逻辑状态；

(0017) 图 3 为本发明的延伸指令格式的方块图；

(0018) 图 4 为一表格，其显示依据本发明，延伸架构特征如何对应至一
8 位延伸前置码实施例中位的逻辑状态；

(0019) 图 5 为解说本发明用以执行延伸指令的管线化微处理器方块图；

(0020) 图 6 为图 5 微处理器内转译阶段逻辑的细部的方块图。

(0021) 图 7 为本发明用以转译与执行延伸指令的方法的运作流程图。

具体实施方式

(0022) 以下的说明，是在一特定实施例及其必要条件的脉络下而提供，
20 可使一般熟悉此项技术者能够利用本发明。然而，各种对该较佳实施例所作的修改，对熟悉此项技术者而言是显而易见的，并且，在此所讨论的一般原理，亦可应用至其它实施例。因此，本发明并不限于此处所展示与叙述的特定实施例，而是具有与此处所公开的原理与特征相符的最大范围。

(0023) 前文已针对现在微处理器内，如何扩充其架构特征，以超越相关
25 指令集能力的技术，作了背景的讨论。有鉴于此，在图 1 与图 2，将讨论一相关技术的例子。此处的讨论强调了微处理器设计者所一直面对的两难问题 (dilemma)，即一方面，他们想将最新开发的架构特征纳入微处理器的设计中，但另一方面，他们又要保留执行旧有应用程序的能力。在图 1 至 2 的例子中，一完全占用的运算码图，已把增加新运算码至该范例架构的可能性排除，因而
30 迫使设计者要不就选择将新特征纳入，而牺牲某种程度的旧有软件兼容性，要

不就将架构上的最新进展一并放弃，以便维持微处理器与旧有应用程序的兼容性。在相关技术的讨论后，于图 3 至 6，将提供对本发明的讨论。通过确认与利用一既有但未使用的运算码作为一延伸指令的前置码标志，本发明可让微处理器设计者克服已完全使用的指令集架构的限制，在允许他们提供延伸架构特征的同时，也能保留与旧有应用程序的兼容性。

(0024) 请参阅图 1，其是一相关技术的微处理器指令格式 100 的方块图。该相关技术的指令 100 具有数量可变的数据项 101—103，每一项目皆设定成一特定值，合在一起便组成微处理器的一特定指令 100。该特定指令 100 指示微处理器执行一特定运算，例如将两操作数相加，或是将一操作数从内存搬移至微处理器内的缓存器。一般而言，指令 100 内的运算码项目 102 指定了所要执行的特定运算，而选用 (optional) 的地址指定元项目 103 位于运算码 102 之后，以指定关于该特定运算的附加信息，像是如何执行该运算，操作数位于何处等等。指令格式 100 并允许程序员在一运算码 102 前加上前置码项目 101。在运算码 102 所指定的特定运算执行时，前置码 101 用以指示是否使用特定的架构特征。一般而言，这些架构特征的应用范围涵盖指令集中任何运算码 102 所指定的运算。例如，现今前置码 101 存在于一些能使用不同大小操作数（如 8 位、16 位、32 位）执行运算的微处理器中。而当许多此类处理器被程序化为一预设的操作数大小时（比如 32 位），在其个别指令集中所提供的前置码 101，仍能使程序员依据各个指令，选择性地取代 (override) 该预设的操作数大小（如为了执行 16 位运算）。可选择的操作数大小仅是架构特征的一例，在许多现代的微处理器中，这些架构特征能应用于众多可由运算码 102 加以指定的运算（如加、减、乘、布尔逻辑等）。

(0025) 图 1 所示的指令格式 100，有一为业界所熟知的范例，此即 x86 指令格式 100，其为所有现代的 x86-兼容微处理器所采用。更具体他说，x86 指令格式 100（也称为 x86 指令集架构 100）使用了 8 位前置码 101、8 位运算码 102 以及 8 位地址指定元 103。x86 架构 100 亦具有数个前置码 101，其中两个取代了 x86 微处理器所预设的地址 / 数据大小（即运算码状态 66H 与 67H），另一个则指示微处理器依据不同的转译规则来解译其后的运算码字节 102（即前置码值 0FH，其使得转译动作是依据所谓的二字节运算码规则来进行），其它的前置码 101 则使特殊运算重复执行，直至某条件满足为止（即 REP

运算码： F0H、F2H 及 F3H)。

(0026) 现请参阅图 2，其显示一表格 200，用以描述一指令集架构的指令 201 如何对应至图 1 指令格式内一运算码字节 102 的位逻辑状态。表格 200 呈现了一示范性的 8 位运算码图 200，其将一 8 位运算码项目 102 所具有的最多 256 个值，关联到对应的微处理器运算码指令 201。表格 200 将运算码项目 102 的一特定值，譬如 02H，映像至一对应的运算码指令 201 (即指令 I02)。在 x86 运算码图的例子中，为此领域中人所熟知的是，运算码值 14H 是映像至 x86 的进位累加 (Add With Carry, ADC) 指令，此指令将一 8 位的直接 (immediate) 操作数加至架构寄存器 AL 的内含值。熟悉此领域技术者也将发现，上文提及的 x86 前置码 101 (亦即 66H、67H、0FH、F0H、F2H 及 F3H) 是实际的运算码值 201，其在不同脉络下，指定要将特定的架构延伸项应用于随后的运算码项目 102 所指定的运算。例如，在运算码 14H (正常情况下，是前述的 ADC 运算码) 前加上前置码 0FH，会使得 x86 处理器执行一“解压缩与插入低压缩的单精度浮点值” (Unpack and Interleave Low Packed Single-precision Floating-Point Values) 运算，而非原本的 ADC 运算。诸如此 x86 例子所述的特征，在现代的微处理器中是部分地致能 (enabled)，此因微处理器内的指令转译 / 译码逻辑是依序解译一指令 100 的项目 101—103。所以在过去，于指令集架构中使用特定运算码值作为前置码 101，可允许微处理器设计者将不少先进的架构特征纳入兼容旧有软件的微处理器的设计中，而不会对未使用那些特定运算码状态的旧有程序，带来执行上的负面冲击。例如，一未曾使用 x86 运算码 0FH 的旧有程序，仍可在今日的 x86 微处理器上执行。而一较新的应用程序，借着运用 x86 运算码 0FH 作为前置码 101，就能使用许多 x86 架构特征，如单一指令多重数据 (SIMD) 运算，条件移动运算等等。

(0027) 尽管过去已通过指定可用 / 多余的运算码值 201 作为前置码 101 (也称为架构特征标志 / 指针 101 或逸出指令 101)，来提供架构特征，但许多指令集架构 100 在提供功能上的强化时，仍会因为一非常直接的理由，而碰到阻碍：所有可用 / 多余的运算码值已被用完，也就是，运算码图 200 中的全部运算码值已被架构化地指定。当所有可用的值被分派为运算码项目 102 或前置码项目 101 时，就没有剩余的运算码值可作为纳入新特征之用。这个严重的问题存在于现在的许多微处理器架构中，因而迫使设计者得在增添架构特征与

保留旧有程序的兼容性两者间作抉择。

(0028) 值得注意的是, 图 2 所示的指令 201 是以一般性的方式表示 (亦即 I24、I86), 而非具体指涉实际的运算 (如进位累加、减, 异或)。这是因为, 在一些不同的微处理器架构中, 完全占用的运算码图 200 在架构上, 已将纳入较新进展的可能性排除。虽然图 2 例子所提到的, 是 8 位的运算码项目 102, 熟悉此领域技术者仍将发现, 运算码 102 的特定大小, 除了作为一特殊情况来讨论完全占用的运算码结构 200 所造成的问题外, 其它方面与问题本身并不相干。因此, 一完全占用的 6 位运算码图将有 64 个可架构化地指定的运算码 / 前置码 201, 并将无法提供可用 / 多余的运算码值作为扩充之用。

(0029) 另一种做法, 则并非将原有指令集废弃, 以一新的格式 100 与运算码图 200 取代, 而是只针对一部份既有的运算码 201, 以新的指令意含取代, 如图 2 的运算码 40H 至 4FH。以这种混合的技术, 符合旧有规格的微处理器就可以兼容旧有软件模式运作, 其中运算码 40H, 4FH 是依旧有规则来解译, 或者以加强模式 (enhanced mode) 运作, 其中运算码 40H-4FH 是依加强的架构规则来解译。此项技术确能允许设计者将新特征纳入设计, 然而, 当符合旧有规格的微处理器于加强模式运作时, 缺点仍旧存在, 因为微处理器不能执行任何使用运算码 40H、4FH 的应用程序。因此, 站在保留旧有软件兼容性的立场, 兼容旧有软件 / 加强模式的技术, 还是无法接受的。

(0030) 然而, 对于运算码空间已完全占用的指令集 200, 且该空间涵盖所有于符合旧有规格的微处理器上执行的应用程序的情形, 本发明已注意到其中运算码 201 的使用状况, 且观察出, 虽然有些指令 202 是架构化地指定, 但未用于能被微处理器执行的应用程序中。图 2 所述的指令 IF1 202 即为此现象的一例。事实上, 相同的运算码值 202 (亦即 F1H) 是映像至未用于 x86 指令集架构的一有效指令 202。虽然该未使用的 x86 指令 202 是有效的 x86 指令 202, 其指示要在 x86 微处理器上执行一架构化地指定的运算, 但它却未使用于任何能在现代 x86 微处理器上执行的应用程序。这个特殊的 x86 指令 202 被称为电路内仿真断点 (In Circuit Emulation Breakpoint) (亦即 ICE BKPT, 运算码值为 F1H), 之前都是专门使用于一种现在已不存在的微处理器仿真设备中。ICE BKPT 202 从未用于电路内仿真器的外的应用程序中, 并且先前使用 ICE BKPT 202 的电路内仿真设备已不复存在。因此, 在 x86 的情形下, 本发明已在一完

全占用的指令集架构 200 内发现一样工具，借着利用一有效但未使用的运算码 202，以允许在微处理器的设计中纳入先进的架构特征，而不需牺牲旧有软件的兼容性。在一完全占用的指令集架构 200 中，本发明利用一架构化地指定但未使用的运算码 202，作为一指针标志，以指出其后的一 n 位前置码，因此允
5 许微处理器设计者可将最多 2^n 个最新发展的架构特征，纳入微处理器的设计中，同时保留与所有旧有软件完全的兼容性。本发明现将参照图 3 至 6 进行讨论。

(0031) 现请参阅图 3，其为本发明的延伸指令格式 300 的方块图。与图 1 所讨论的格式 100 非常近似，该延伸指令格式 300 具有数量可变的数据项 301
10 —305，每一项目设定为一特定值，集合起来便组成微处理器的一特定指令 300。该特定指令 300 指示微处理器执行一特定运算，像是将两操作数相加，或是将一操作数从内存搬移至微处理器的缓存器内。一般而言，指令 300 的运算码项目 302 指定了所要执行的特定运算，而选用的地址指定元项目 303 则位于运算码 302 后，以指定该特定运算的相关附加信息，像是如何执行该运算，
15 操作数位于何处等等。指令格式 300 亦允许程序员在一运算码 302 前加上前置码项目 301。在运算码 302 所指定的特定运算执行时，前置码项目 301 是用来指示是否要使用既有的架构特征。

(0032) 然而，本发明的延伸指令 300 是前述图 1 指令格式 100 的一超集 (superset)，其具有两个附加项目 304 与 305，可被选择性作为指令延伸项，
20 并置于一格式化延伸指令 300 中所有其余项目 301—303 的前。这两个附加项目 304 与 305 用以致能 / 除能 (enable / disable) 数个架构特征，其中这些特征并无法在一完全占用的指令集架构内加以指定。选用项目 304 与 305 是一延伸指令标志 304 与一延伸前置码 305。该延伸指令标志 304 是一微处理器指令集内另一架构化地指定的运算码。在一 x86 的实施例中，该延伸指令标志 304，
25 或称逸出标志 304，是用运算码状态 FIH，其为早先使用的 ICE BKPT 指令。逸出标志 304 向微处理器逻辑指出，该延伸前置码 305，或称延伸特征指定元 305 是跟随在后，其中延伸特征指定元 305 用以指示在延伸指令 300 所指定的运算执行时，要使用一架构延伸项。在一具体实施例中，延伸指令标志 304 指出，一对应延伸指令 300 的附随部分 301—303 及 305 指定了微处理器所要
30 执行的延伸运算。延伸前置码 305 则指定了该延伸运算的补充部分。也就是，

于该运算中该微处理器指令集所未提供的部分。

(0033) 此处将本发明的指令延伸技术作个概述。一指令延伸项是以一既有指令集架构其中一运算码 / 指令 304 与一 n 位延伸特征前置码进行组态。所选取的运算码指令作为一指针 304, 以指出指令 300 是一延伸指令 300 (亦即, 其指定了微处理器架构延伸项), 而该 n 位特征前置码 305 则指定一延伸架构特征, 于该延伸指令 300 执行时使用。在一具体实施例中, 延伸前置码 305 具八位大小, 最多可指定 256 个不同的架构附加项, 加入一既有指令集中现有指令的处理过程。n 位前置码的实施例, 则最多可指定 2^n 个不同的附加特征, 于一特定运算执行时使用。

(0034) 现请参阅图 4, 一表格 400 显示依据本发明, 延伸架构特征如何映像至一 8 位延伸前置码实施例的位逻辑状态。类似于图 2 所讨论的运算码图 200, 图 4 的表格 400 呈现一示范性的 8 位前置码图 400, 其将一 8 位前置码项目 305 的最多 256 个值, 关联到一符合旧有规格微处理器的对应架构特征 401 (如 E34、E4D 等), 或称架构强化功能 (architectural enhancement) 401。在 x86 的具体实施例中, 本发明的 8 位延伸特征前置码 305 是在现行 x86 指令集架构所提供的特征外, 提供给上述架构延伸项 401 (亦即 E00—EFF) 使用。在一具体实施例中, 延伸前置码 305 被映像到致能条件指令执行的特征 401。另一具体实施例则可增加微处理器中架构缓存器的数量, 如从 16 个变成 32 个。其它的实施例致能了架构特征 401, 诸如条件码回写 (write back) 的选择性控制、依据每一指令选择性地控制内存属性 (如写入保护、回写、写入结合 (write combine))。延伸地址大小 (即 64 位、128 位、256 位)、延伸操作数大小 (即 64 位、128 位、256 位)、储存检查的禁止、中断禁止以及结果回写的选择性控制。就如图 1 与图 2 所述的前置码 101, 本发明的延伸前置码 305 使一指定架构特征 401 能像上文所述般, 应用于运算码项目 302 所指定运算的执行时。本发明所提供的双运算码 (dual-opcode) 的实施例, 则选取一特定前置码状态 401, 以指出随后的运算码项目 302 具有一全新的转译含义。这样的实施例是应用于双架构的微处理器设计, 以执行不只一个指令集。

(0035) 图 4 所示的延伸特征 401 是以一般性的方式表示, 而非具体指涉实际的特征, 此因本发明的技术可应用于各种不同的架构延伸项 401 与特定的指令集架构。熟悉此领域技术者将发现, 许多不同的架构特征 401, 其中一些

已于上文提及，可依此处所述的逸出标志 304 / 延伸前置码 305 技术将其纳入一既有的指令集。图 4 的 8 位前置码实施例提供了最多 256 个不同的特征 401，而一 n 位前置码实施例则具有最多 2^n 个不同特征 401 的程序化选择。

(0036) 现请参阅图 5，其为解说本发明用以执行延伸指令 300 的管线化微处理器 500 的方块图。微处理器 500 具有三个明显的阶段类型：提取、转译及执行。提取阶段具有提取逻辑 501，可从指令高速缓存 502 或外部内存 502 提取指令。所提取的指令经由指令队列 503 送至转译阶段。转译阶段具有转译逻辑 504，耦接至一微指令队列 506。转译逻辑 504 包括延伸转译逻辑 505。执行阶段则有执行逻辑 507，其内具有延伸执行逻辑 508。

(0037) 依据本发明，于运作时，提取逻辑 501 从指令高速缓存 / 外部内存 502 提取格式化指令，并将这些指令依其执行顺序放入指令队列 503 中。接着从指令队列 503 提取这些指令，送至转译逻辑 504。转译逻辑 504 将每一送入的指令转译 / 译码为一对应的微指令序列，以指示微处理器 500 执行这些指令所指定的运算。依本发明，延伸转译逻辑 505 监测那些具有延伸前置码标志的指令，以进行对应延伸前置码的转译 / 译码。在一 x86 的实施例中，延伸转译逻辑 505 组态为监测其值为 FIH 的延伸前置码标志，其是 x86 的 ICE BKPT 运算码。微指令字段则提供于微指令队列 506 中，以致能 / 除能延伸指令中所指定的架构特征。

(0038) 微指令从微指令队列 506 被送至执行逻辑 507，由延伸执行逻辑 508 监测具有依微指令字段指示而致能的架构特征的微指令，并于微指令所指定的运算执行时，应用这些架构特征。

(0039) 熟悉此领域技术者将发现，图 5 所示的微处理器 500 是现代的管线化微处理器 500 经过简化的结果。事实上，现代的管线化微处理器 500 最多可包括有 20 至 30 个不同的管线阶段。然而，这些阶段可概括地归类为方块图所示的三个阶段，因此，图 5 的方块图 500 可用以点明前述本发明实施例所需的必要组件。为了简明起见，微处理器 500 中无关的组件并未显示出来。例如，在一实施例中，增加微处理器 500 的架构缓存器（图中未显示）至超过既有指令集架构所提供的数量，如此则必须提供一延伸缓存器档案（未显示）及对应的读 / 写逻辑（未显示）、作为延伸执行逻辑 508 的一部份。至于可处理不同大小的延伸地址 / 操作数的实施例，则包括计算延伸地址、移动延伸操作数及

在延伸操作数上执行运算等所需的延伸组件。

(0040) 现请参阅图 6, 其为图 5 的微处理器内转译阶段逻辑 600 的细部的方块图。转译阶段逻辑 600 具有一指令缓冲器 604, 依本发明, 其提供延伸指令至转译逻辑 605。转译逻辑 605 是耦接至一具有一延伸特征字段 603 的机器特定缓存器 (machine specific register) 602。转译逻辑 605 具一转译控制器 606, 其提供一除能讯号 607 至一逸出指令监测器 608 及一延伸前置码译码器 609。逸出指令监测器 608 耦接至延伸前置码译码器 609 及一指令译码器 610。延伸前置码译码器 609 与指令译码逻辑 610 存取一控制只读存储器 (ROM) 611, 其中储存了对应至某些延伸指令的样板 (template) 微指令序列。转译逻辑 605 亦包括一微指令缓冲器 612, 其具有一运算码延伸项字段 613。一微运算码字段 614, 一目的字段 615、一来源字段 616 以及一位移字段 617。

(0041) 运作上, 在微处理器通电激活期间, 机器特定缓存器 602 的延伸字段 603 的状态是通过讯号激活状态 (signal power-up state) 601 决定, 以指出该特定微处理器是否能转译与执行本发明的延伸指令。在一具体实施例中, 讯号 601 从一特征控制缓存器 (图上未显示) 导出, 该特征控制缓存器则读取一于制造时即已组态的熔丝数组 (fuse array) (未显示)。机器特定缓存器 602 将延伸特征字段 603 的状态送至转译控制器 606。转译控制逻辑 606 则控制从指令缓冲器 604 所提取的指令, 要依照延伸指令规则或是既有指令规则进行解译。提供这样的控制特征, 可允许监督应用程序 (如 BIOS) 致能 / 除能微处理器的延伸执行特征。若延伸执行被除能, 则具有被选为延伸特征标志的运算码状态的指令, 将依既有转译规则进行转译。在一 x86 的具体实施例中, 选取运算码状态 F1H 作为标志, 则在常用的转译规则下, 遇到 F1H 将造成不合法的指令异常 (exception)。然而, 在延伸转译规则下, 若遇到标志, 则会被逸出指令监测器 608 监测出来。逸出指令监测器 608 因而于延伸前置码译码器 609 转译 / 译码标志之后的延伸前置码时, 除能指令译码器 610 的运作, 并于转译 / 译码该延伸指令的剩余部分时, 致能指令译码器 610。某些特定指令将导致对控制 ROM 611 的存取, 以获取对应的微指令序列样板。微指令缓冲器 612 的运算码延伸项字段 613 由前置码译码器 609 进行组态, 而其它缓冲器字段 614—617 则由指令译码器 610 来进行组态。经过组态的微指令 612 被送至一微指令队列 (未显示于图中), 由处理器进行后续执行。

(0042) 现请参阅图 7, 其为描述本发明用以转译与执行延伸指令的方法的运作流程图 700。流程开始于方块 702, 其中一个以延伸指令进行组态后的程序被送至微处理器。流程接着进行至方块 704。

5 (0043) 于方块 704 中, 下一个指令是从高速缓存 / 内存提取。流程接着进行至判断方块 706。

(0044) 于判断方块 706 中, 对在方块 704 中所提取的下个指令进行检查, 以判断是否包括一延伸逸出标志 / 码。若否, 则流程进行至方块 712。若监测到该延伸逸出码, 则流程进行至方块 708。

10 (0045) 于方块 708 中, 转译 / 译码是在一于方块 706 所监测逸出码后的延伸特征前置码上执行。流程接着进行到方块 710。

(0046) 于方块 710 中, 一微指令序列的对应字段被组态为指出该延伸前置码指定为致能 / 除能的延伸架构特征。流程接着进行至方块 712。

15 (0047) 于方块 712 中, 该指令的其余部分 (如前置码项目、运算码、地址指定元) 被转译 / 译码, 以判断所要执行的运算及关联操作数的属性。流程接着进行至方块 714。

(0048) 于方块 714 中, 一微指令序列的其余字段被组态为指定所指定的运算及其操作数属性。流程接着进行至方块 716。

20 (0049) 于方块 716 中, 该微指令序列, 其包括方块 710 所组态的运算码延伸项字段以及方块 714 所组态的其余字段, 被送至一微指令队列, 由微处理器执行。流程接着进行至方块 718。

(0050) 于方块 718 中, 本方法完成。

25 (0051) 虽然本发明及其目的, 特征与优点已详细叙述, 其它实施例亦可包括在本发明的范围内。例如, 本发明已就如下的技术加以叙述: 利用已完全占用的指令集架构内的单一、未使用的运算码状态作为标志, 以指出其后的延伸特征前置码。但本发明的范围就任一方面来看, 并不限于已完全占用的指令集架构, 或未使用的指令, 或是单一标志。相反地, 本发明涵盖了未完全映像的指令集、具已使用运算码的实施例以及使用一个以上的指令标志的实施例。例如, 考虑一没有未使用运算码状态的指令集架构。本发明的一具体实施例包括了选取一作为逸出标志的运算码状态, 其中选取标准是依市场因素而决定。
30 另一具体实施例则包括使用运算码的一特殊组合作为标志, 如运算码状态 7FH

的连续出现。因此，本发明的本质是在于使用一标志序列，其后则为 n 位的延伸项前置码，可使延伸架构特征被应用于延伸指令所指定的运算中。

(0052) 此外，虽然上文是利用微处理器为例来解说本发明及其特征和优点，熟悉此领域技术者仍可察觉，本发明的范围并不限于微处理器的架构，而可涵盖所有形式的可编程装置，如信号处理器、工业用控制器（**industrial controller**）、数组处理机及其它同类装置。

总之，以上所述仅为本发明的较佳实施例而已，当不能以此限定本发明所实施的范围。大凡依本发明权利要求所作的等效变化与修饰，皆应仍属于本发明专利涵盖的范围内。

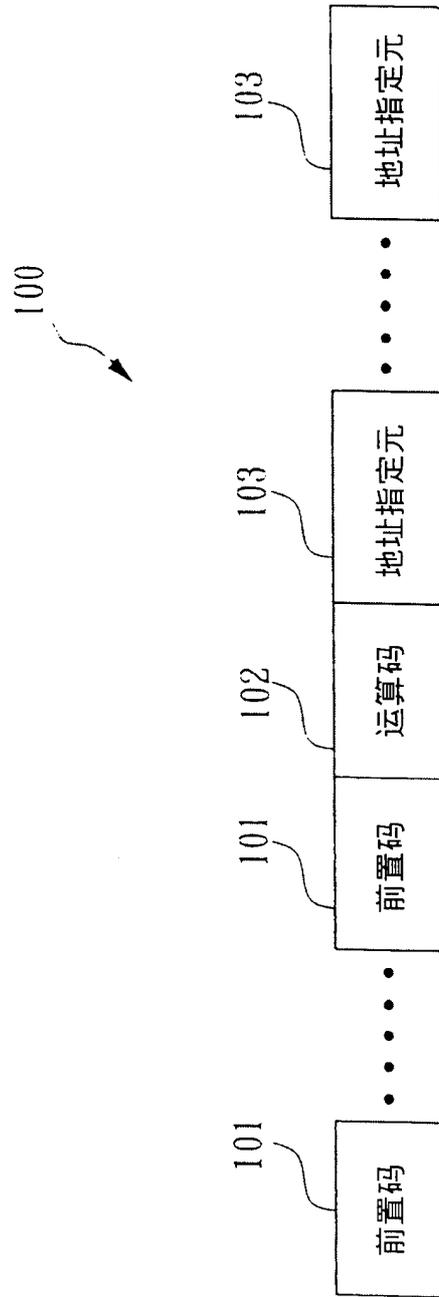


图1

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 10A | 10B | 10C | 10D | 10E | 10F |
| 1 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 11A | 11B | 11C | 11D | 11E | 11F |
| 2 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 12A | 12B | 12C | 12D | 12E | 12F |
| 3 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 13A | 13B | 13C | 13D | 13E | 13F |
| 4 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 14A | 14B | 14C | 14D | 14E | 14F |
| 5 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 15A | 15B | 15C | 15D | 15E | 15F |
| 6 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 16A | 16B | 16C | 16D | 16E | 16F |
| 7 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 17A | 17B | 17C | 17D | 17E | 17F |
| 8 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 18A | 18B | 18C | 18D | 18E | 18F |
| 9 | 190 | 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 19A | 19B | 19C | 19D | 19E | 19F |
| A | 1A0 | 1A1 | 1A2 | 1A3 | 1A4 | 1A5 | 1A6 | 1A7 | 1A8 | 1A9 | 1AA | 1AB | 1AC | 1AD | 1AE | 1AF |
| B | 1B0 | 1B1 | 1B2 | 1B3 | 1B4 | 1B5 | 1B6 | 1B7 | 1B8 | 1B9 | 1BA | 1BB | 1BC | 1BD | 1BE | 1BF |
| C | 1C0 | 1C1 | 1C2 | 1C3 | 1C4 | 1C5 | 1C6 | 1C7 | 1C8 | 1C9 | 1CA | 1CB | 1CC | 1CD | 1CE | 1CF |
| D | 1D0 | 1D1 | 1D2 | 1D3 | 1D4 | 1D5 | 1D6 | 1D7 | 1D8 | 1D9 | 1DA | 1DB | 1DC | 1DD | 1DE | 1DF |
| E | 1E0 | 1E1 | 1E2 | 1E3 | 1E4 | 1E5 | 1E6 | 1E7 | 1E8 | 1E9 | 1EA | 1EB | 1EC | 1ED | 1EE | 1EF |
| F | 1F0 | 1F1 | 1F2 | 1F3 | 1F4 | 1F5 | 1F6 | 1F7 | 1F8 | 1F9 | 1FA | 1FB | 1FC | 1FD | 1FE | 1FF |

200

201

201

201

201

202

图 2

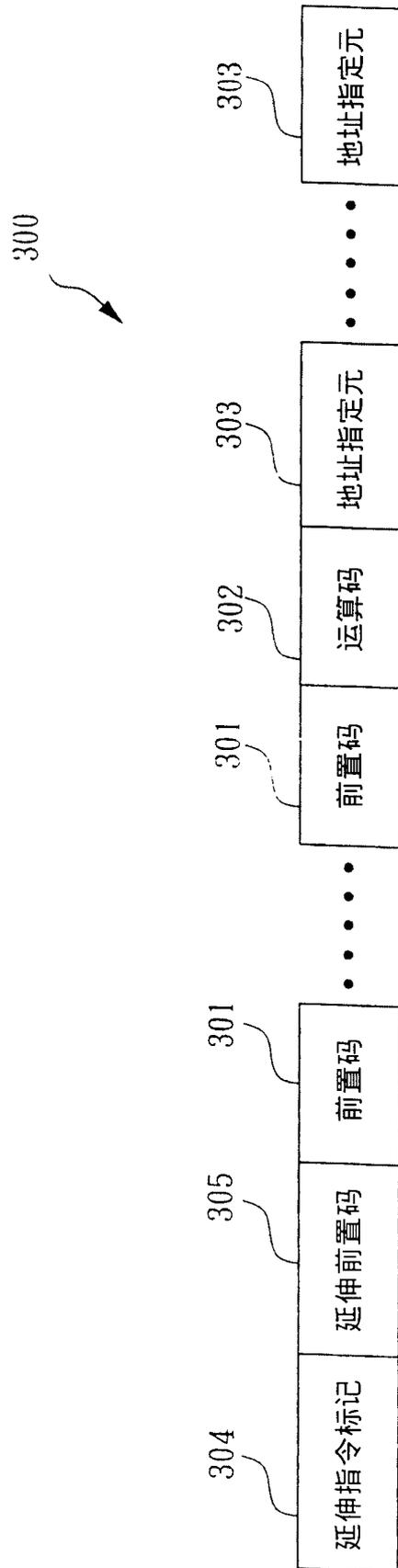


图 3

400

401

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | E00 | E01 | E02 | E03 | E04 | E05 | E06 | E07 | E08 | E09 | E0A | E0B | E0C | E0D | E0E | E0F |
| 1 | E10 | E11 | E12 | E13 | E14 | E15 | E16 | E17 | E18 | E19 | E1A | E1B | E1C | E1D | E1E | E1F |
| 2 | E20 | E21 | E22 | E23 | E24 | E25 | E26 | E27 | E28 | E29 | E2A | E2B | E2C | E2D | E2E | E2F |
| 3 | E30 | E31 | E32 | E33 | E34 | E35 | E36 | E37 | E38 | E39 | E3A | E3B | E3C | E3D | E3E | E3F |
| 4 | E40 | E41 | E42 | E43 | E44 | E45 | E46 | E47 | E48 | E49 | E4A | E4B | E4C | E4D | E4E | E4F |
| 5 | E50 | E51 | E52 | E53 | E54 | E55 | E56 | E57 | E58 | E59 | E5A | E5B | E5C | E5D | E5E | E5F |
| 6 | E60 | E61 | E62 | E63 | E64 | E65 | E66 | E67 | E68 | E69 | E6A | E6B | E6C | E6D | E6E | E6F |
| 7 | E70 | E71 | E72 | E73 | E74 | E75 | E76 | E77 | E78 | E79 | E7A | E7B | E7C | E7D | E7E | E7F |
| 8 | E80 | E81 | E82 | E83 | E84 | E85 | E86 | E87 | E88 | E89 | E8A | E8B | E8C | E8D | E8E | E8F |
| 9 | E90 | E91 | E92 | E93 | E94 | E95 | E96 | E97 | E98 | E99 | E9A | E9B | E9C | E9D | E9E | E9F |
| A | EA0 | EA1 | EA2 | EA3 | EA4 | EA5 | EA6 | EA7 | EA8 | EA9 | EAA | EAB | EAC | EAD | EAE | EAF |
| B | EB0 | EB1 | EB2 | EB3 | EB4 | EB5 | EB6 | EB7 | EB8 | EB9 | EBA | EBB | EBC | EBD | EBE | EBF |
| C | EC0 | EC1 | EC2 | EC3 | EC4 | EC5 | EC6 | EC7 | EC8 | EC9 | ECA | ECB | ECC | ECD | ECE | ECF |
| D | ED0 | ED1 | ED2 | ED3 | ED4 | ED5 | ED6 | ED7 | ED8 | ED9 | EDA | EDB | EDC | EDD | EDE | EDF |
| E | EE0 | EE1 | EE2 | EE3 | EE4 | EE5 | EE6 | EE7 | EE8 | EE9 | EEA | EEB | EEC | EED | EEE | EEF |
| F | EF0 | EF1 | EF2 | EF3 | EF4 | EF5 | EF6 | EF7 | EF8 | EF9 | EFA | EFB | EFC | EFD | EFE | EFF |

图 4

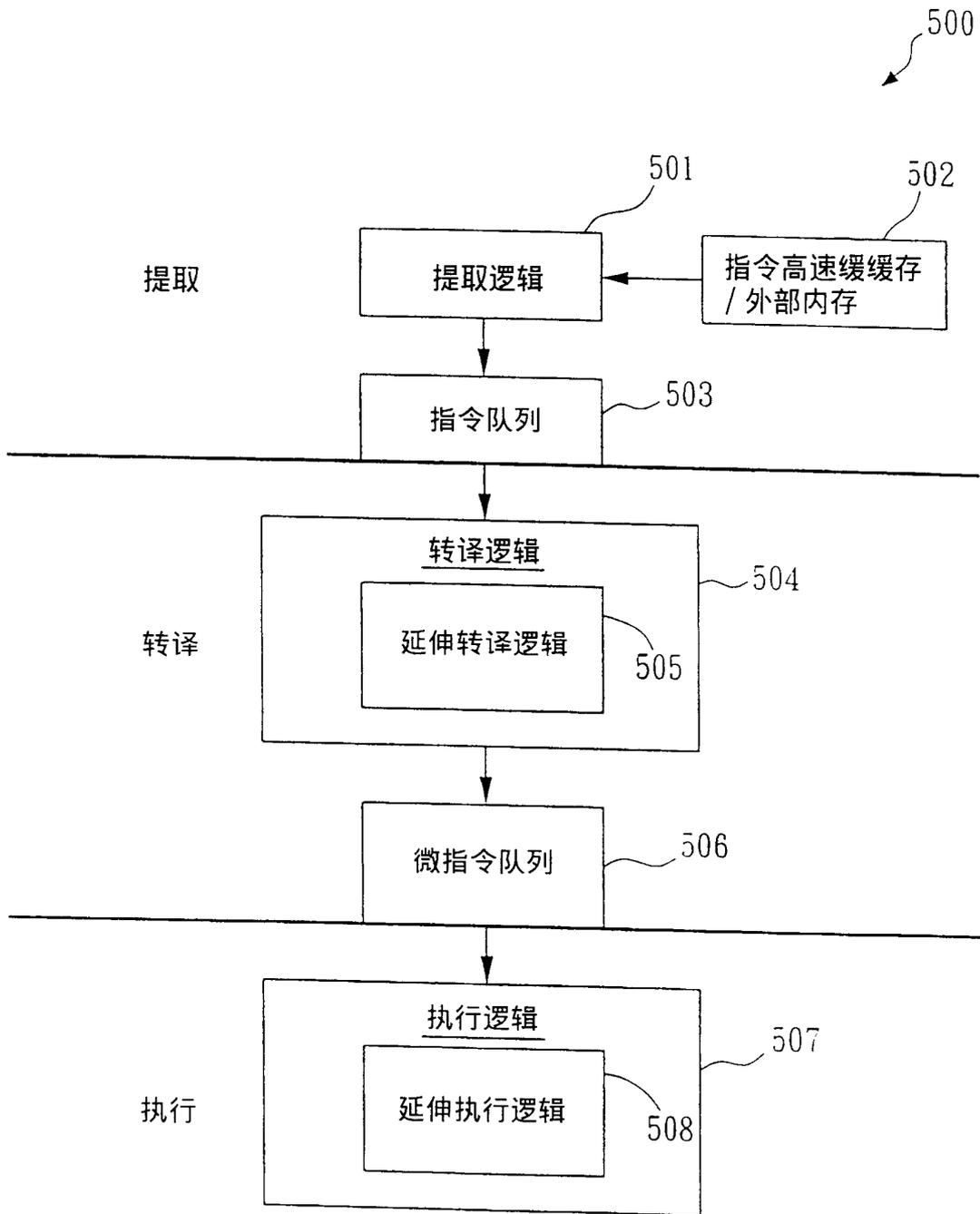


图 5

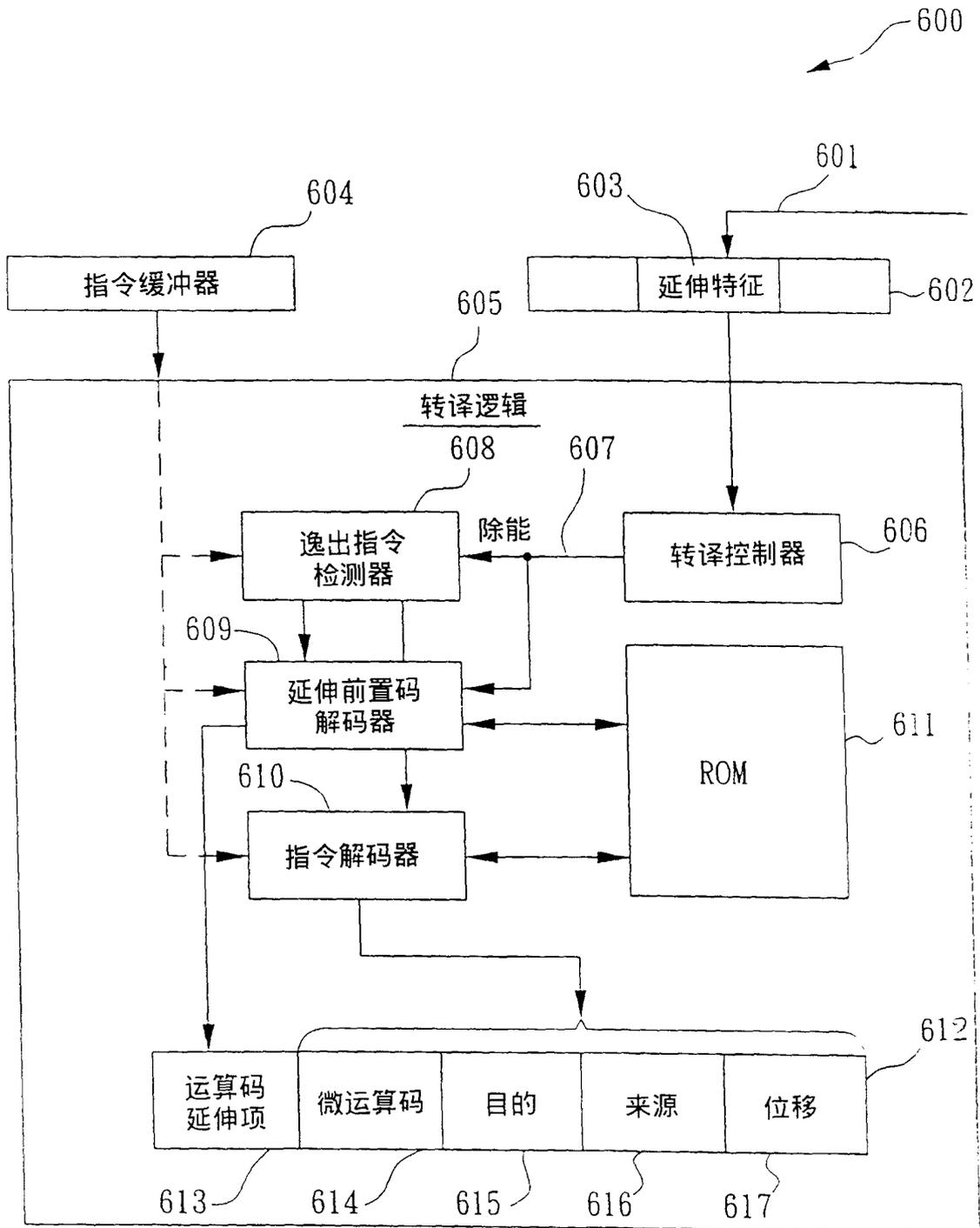


图 6

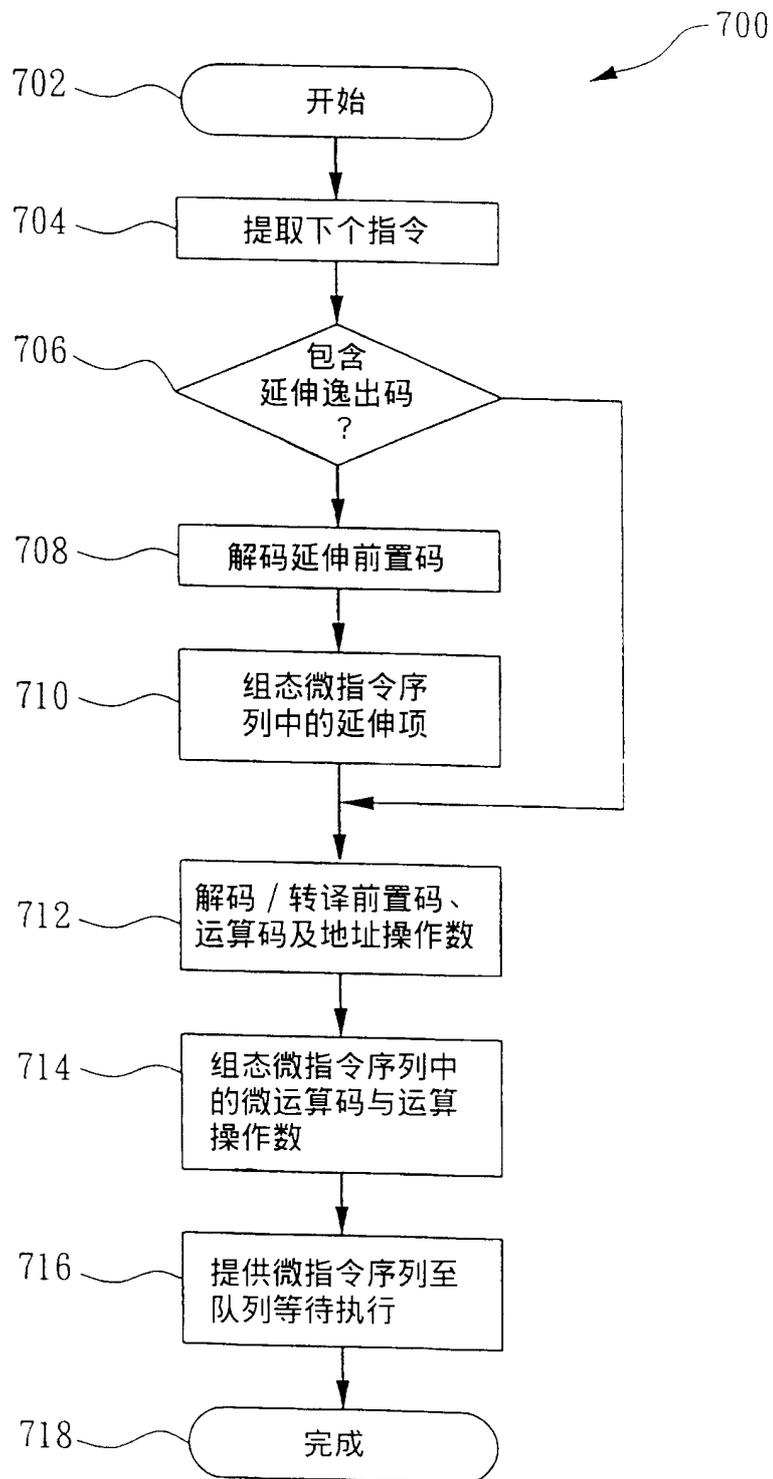


图 7