

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/46 (2006.01)



[12] 发明专利说明书

专利号 ZL 200380103508.4

[45] 授权公告日 2009 年 8 月 5 日

[11] 授权公告号 CN 100524224C

[22] 申请日 2003.10.27

WO0146800A 2001.6.28

[21] 申请号 200380103508.4

ADVANCED RISC MACINES LTD (ARM) :

[30] 优先权

“ARM710 Data Sheet”. 第 10 页—第 18 页;
第 79 页—第 94 页, ONLINE DATA SHEET.
1994

[32] 2002.11.18 [33] GB [31] 0226874.6

审查员 李 菲

[32] 2003.2.14 [33] GB [31] 0303494.9

[86] 国际申请 PCT/GB2003/004615 2003.10.27

[87] 国际公布 WO2004/046924 英 2004.6.3

[85] 进入国家阶段日期 2005.5.18

[73] 专利权人 ARM 有限公司

地址 英国剑桥郡

[72] 发明人 S·C·瓦特 C·B·多尔南

L·奥里安 N·朝斯萨德

L·贝内特 S·E·S·布罗奇尔

[56] 参考文献

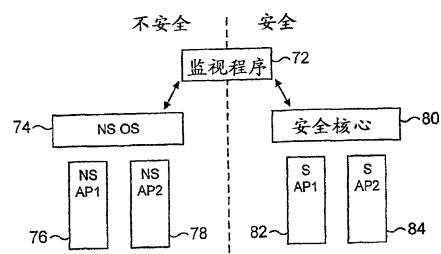
US5615263A 1997.3.25

[54] 发明名称

用于在安全模式和非安全模式间切换处理器
的装置和方法

[57] 摘要

提供一种数据处理系统，包括：可在多种模式
和安全域或非安全域中操作的处理器，包括：为所
述域中的模式的至少一个安全模式；为所述非安全
域中的模式的至少一个非安全模式；以及监视模
式，其中当所述处理器正在执行安全模式中的程序
时，所述程序访问当所述处理器正在非安全模式中
操作时不能访问的安全数据；以及经所述监视模
式，发生所述安全模式和所述非安全模式间的切
换，所述处理器至少部分在所述监视模式中操作来
执行监视程序以便管理所述安全模式和所述非安全
模式间的切换。



权利要求书 8 页 说明书 83 页 附图 56 页

1. 一种用于处理数据的装置，所述装置包括：

一个处理器，被配置成工作在多种模式和多个域中，所述多个域包括安全域和非安全域，所述多种模式包括：

为所述安全域中的模式的至少一个安全模式；

为所述非安全域中的模式的至少一个非安全模式；以及

监视模式，其特征在于，

当所述处理器正在执行安全模式中的程序时，所述处理器被配置成以便所述程序访问当所述处理器正在非安全模式中操作时不能访问的安全数据；以及

所述处理器被配置成以便经所述监视模式，发生所述安全模式和所述非安全模式间的切换，所述处理器至少部分在所述监视模式中操作来执行监视程序以便管理所述安全模式和所述非安全模式间的切换。

2. 如权利要求 1 所述的装置，其特征在于，所述处理器包括寄存器组以及所述监视程序可用来当从所述安全模式切换到所述非安全模式时，至少刷新在所述安全模式和所述非安全模式间共享的所述寄存器组部分，以便除了所述监视程序所允许的，保存在所述寄存器组中的安全数据不会从所述安全模式传递到所述非安全模式。

3. 如权利要求 1 所述的装置，其特征在于，所述处理器包括当所述处理器正在所述非安全模式中操作时使用的非安全寄存器组和所述处理器正在所述安全模式中操作时使用的安全寄存器组。

4. 如权利要求 1、2 和 3 的任何一个所述的装置，其特征在于，当在所述非安全模式中操作时，通过切换到所述监视程序内的一个或一个或多个固定点来切换到所述安全模式。

5. 如权利要求 1 所述的装置，其特征在于，所述安全域提供至少一个安全特许模式和至少一个安全用户模式，以及当在所述安全特许模式中操作时，使用与切换到安全用户模式相同的机制，可以切换到所述监视模式。

6. 如权利要求 5 所述的装置，其特征在于，所述切换使用由正在所述非安全模式中执行的程序产生的调用。

7. 如权利要求 6 所述的装置，其特征在于，执行对所述监视程序中的固定点的所述调用。

8. 如权利要求 4 所述的装置，其特征在于，除在所述监视模式中外，由程序尝试触发所述切换以便在所述非安全模式和所述安全模式中切换。

9. 如权利要求 8 所述的装置，其特征在于，对除在所述监视模式中外，通过程序在所述非安全模式和所述安全模式间的切换的任何尝试设中断并触发调用所述监视程序中的固定点。

10. 如权利要求 1 所述的装置，其特征在于，当从所述非安全模式启动进入所述监视模式时，所述处理器存储用于在停止所述非安全模式中的处理时的点重启所述非安全模式中的处理的程序计数器值。

11. 如权利要求 1 所述的装置，其特征在于，当从所述非安全模式启动进入所述监视模式时，所述处理器存储用在使所述处理器恢复到与在停止所述非安全模式中的处理时的点匹配的状态的处理器状态值。

12. 如权利要求 1 所述的装置，其特征在于，当停止所述安全模式中的处理以便能进行所述非安全模式中的异常处理时，经所述监视模式，切换到所述非安全模式以及当结束时，所述异常处理能使控制返回到所述监视模式。

13. 如权利要求 1 所述的装置，其特征在于，所述处理器包括表示所述处理器正在所述安全域还是所述非安全域中操作的安全状态标记，当所述处理器正在所述监视模式中操作时，所述安全状态标记可写，以及当所述处理器正在除所述监视模式外的模式中操作时，所述安全状态标记不可写。

14. 如权利要求 1 所述的装置，其特征在于，当正在所述至少一个非安全模式中操作时，非安全操作系统控制所述处理器。

15. 如权利要求 1 所述的装置，其特征在于，当正在所述至少一个安全模式中操作时，安全操作系统控制所述处理器。

16. 如权利要求 15 所述的装置，其特征在于，所述监视程序是所述安全操作系统的一部分。

17. 如权利要求 1 所述的装置，其特征在于，所述处理器响应应用于触发异常处理的一个或多个异常条件。

18. 如权利要求 17 所述的装置，其特征在于，在所述监视模式中，禁止所述异常条件的至少一个。

19. 如权利要求 2 所述的装置，其特征在于，所述处理器具有多个专用寄存器，当进入所述监视模式时，代替所述寄存器组内的相应的通用寄存器，以便所述多个专用寄存器可用于由所述监视程序使用，而不覆盖所述相应通用寄存器内的数据。

20. 如权利要求 17 所述的装置，其特征在于，异常中断屏蔽寄存器存储当异常发生时，指定所述异常的哪一个应当由正在所述监视模式中执行的异常处理程序处理以及所述异常的哪一个应当由正在所述安全域和所述非安全域的当前一个内的模式中执行的异常处理程序处理的一个或多个参数。

21. 如权利要求 17 所述的装置，其特征在于，所述处理器响应可用来触发进入所述监视模式和执行所述监视程序的安全异常条件。

22. 如权利要求 1 所述的装置，其特征在于，所述处理器响应进入所述监视模式和以预定位置，开始执行所述监视程序的监视模式进入指令。

23. 如权利要求 22 所述的装置，其特征在于，所述监视模式进入指令是模式切换软件中断指令以及所述预定位置是与所述模式切换软件中断指令有关的模式切换软件中断矢量。

24. 如权利要求 23 所述的装置，其特征在于，根据所述处理器是在安全模式还是在非安全模式中，不同模式切换软件中断矢量与所述模式切换软件中断指令有关。

25. 如权利要求 1 所述的装置，其特征在于，所述至少一个安全模式包括下述的一个或多个：

安全监督模式；

安全系统模式；

安全中止异常模式；

安全未定义异常模式；

安全中断模式；

安全快速中断模式；以及

安全用户模式。

26. 如权利要求 1 所述的装置，其特征在于，所述至少一个非安

全模式包括下述的一个或多个：

非安全监督模式；
非安全系统模式；
非安全中止异常模式；
非安全未定义异常模式；
非安全中断模式；
非安全快速中断模式；以及
非安全用户模式。

27. 如权利要求 1 所述的装置，其特征在于，所述处理器可在所述安全域内的至少一个特许安全模式中操作，该安全模式不能从所述安全域改变到所述非安全域，所述特许安全模式具有允许从所述特许安全模式改变成所述监视模式的特权，不必重定向程序执行点。

28. 如权利要求 27 所述的装置，其特征在于，所述处理器响应安全标记并且当在所述监视模式中时，所述处理器可操作用于改变所述安全标记以便在所述安全域和所述非安全域中改变，所述安全标记除所述监视模式外，不能访问写入。

29. 如权利要求 27 和 28 的任何一个所述的装置，其特征在于，所述处理器响应软件中断指令来通过重定向如由与所述软件中断指令有关的中断矢量指定的程序执行点，执行切换到所述监视模式。

30. 如权利要求 27 所述的装置，其特征在于，所述非安全域包括至少一个特许非安全模式，不能执行在没有重定向程序执行点的情况下，从所述至少一个特许非安全模式改变成所述监视模式。

31. 如权利要求 27 所述的装置，其特征在于，在复位时，所述处理器进入所述特许安全模式。

32. 如权利要求 27 所述的装置，其特征在于，在从所述安全域改变到所述非安全域前，正在所述监视模式中执行的监视程序切换到所述安全特许模式以允许保存在所述安全特许模式切换到所述监视模式前，从所述监视模式内不能访问的数据，用于到所述非安全域的所述改变。

33. 如权利要求 27 所述的装置，其特征在于，在所述监视模式中能访问的存储区也能在所述特许安全模式中访问。

34. 一种处理数据的方法，所述方法包括步骤：

通过可在多种模式和多个域中操作的处理器执行程序，所述多个域包括安全域和非安全域，所述多种模式包括：

为所述安全域中的模式的至少一个安全模式；

为所述非安全域中的模式的至少一个非安全模式；以及
监视模式，其特征在于，

当所述处理器正在执行安全模式中的程序时，所述程序访问当所述处理器正在非安全模式中操作时不能访问的安全数据；以及

经所述监视模式，作所述安全模式和所述非安全模式间的切换，
所述处理器执行至少部分在所述监视模式的监视程序来管理所述安全模式和所述非安全模式间的切换。

35. 如权利要求 34 所述的方法，其特征在于，所述处理器包括寄存器组以及当从所述安全模式切换到所述非安全模式时，所述监视程序至少刷新在所述安全模式和所述非安全模式间共享的所述寄存器组部分，以便除了所述监视程序所允许的，保存在所述寄存器组中的安全数据不会从所述安全模式传递到所述非安全模式。

36. 如权利要求 34 所述的方法，其特征在于，所述处理器包括当所述处理器正在所述非安全模式中操作时使用的非安全寄存器组和所述处理器正在所述安全模式中操作时使用的安全寄存器组。

37. 如权利要求 34 所述的方法，其特征在于，当在所述非安全模式中操作时，通过切换到所述监视程序内的一个或多个固定点的一个来切换到所述安全模式。

38. 如权利要求 34 所述的方法，其特征在于，所述安全域提供至少一个安全特许模式和至少一个安全用户模式，以及当在所述安全特许模式中操作时，使用与切换到安全用户模式相同的机制，可以切换到所述监视模式。

39. 如权利要求 37 所述的方法，其特征在于，所述切换使用由正在所述非安全模式中执行的程序产生的调用。

40. 如权利要求 39 所述的方法，其特征在于，执行对所述监视程序中的固定点的所述调用。

41. 如权利要求 37 所述的方法，其特征在于，除在所述监视模式中外，由程序尝试触发所述切换以便在所述非安全模式和所述安全模式之间切换。

42. 如权利要求 41 所述的方法，其特征在于，对除在所述监视模式中外，通过程序在所述非安全模式和所述安全模式间的切换的任何尝试设中断并触发调用所述监视程序中的固定点。

43. 如权利要求 34 所述的方法，其特征在于，当从所述非安全模式启动进入所述监视模式时，所述处理器存储用于在停止所述非安全模式中的处理时的点重启所述非安全模式中的处理的程序计数器值。

44. 如权利要求 34 所述的方法，其特征在于，当从所述非安全模式启动进入所述监视模式时，所述处理器存储用在使所述处理器恢复到与在停止所述非安全模式中的处理时的点匹配的状态的处理器状态值。

45. 如权利要求 34 所述的方法，其特征在于，当停止所述安全模式中的处理以便能进行所述非安全模式中的异常处理时，经所述监视模式，切换到所述非安全模式以及当结束时，所述异常处理能使控制返回到所述监视模式。

46. 如权利要求 34 所述的方法，其特征在于，所述处理器包括一个表示所述处理器正在所述安全域还是所述非安全域中操作的安全状态标记，当所述处理器正在所述监视模式中操作时，所述安全状态标记可写，以及当所述处理器正在除所述监视模式外的模式中操作时，所述安全状态标记不可写。

47. 如权利要求 34 所述的方法，其特征在于，当在所述至少一个非安全模式中操作时，非安全操作系统控制所述处理器。

48. 如权利要求 34 所述的方法，其特征在于，当在所述至少一个安全模式中操作时，安全操作系统控制所述处理器。

49. 如权利要求 48 所述的方法，其特征在于，所述监视程序是所述安全操作系统的一部分。

50. 如权利要求 34 所述的方法，其特征在于，所述处理器响应应用于触发异常处理的一个或多个异常条件。

51. 如权利要求 50 所述的方法，其特征在于，在所述监视模式中，禁止所述异常条件的至少一个。

52. 如权利要求 35 所述的方法，其特征在于，所述处理器具有多个专用寄存器，当进入所述监视模式时，代替所述寄存器组内的相应的通用寄存器，以便所述多个专用寄存器可用于由所述监视程序使用，

而不覆盖所述相应通用寄存器内的数据。

53. 如权利要求 50 所述的方法，其特征在于，异常中断屏蔽寄存器存储当异常发生时，指定所述异常的哪一个应当由正在所述监视模式中执行的异常处理程序处理以及所述异常的哪一个应当由正在所述安全域和所述非安全域的当前一个内的模式中执行的异常处理程序处理的一个或多个参数。

54. 如权利要求 50 所述的方法，其特征在于，所述处理器响应可用来触发进入所述监视模式和执行所述监视程序的安全异常条件。

55. 如权利要求 34 所述的方法，其特征在于，所述处理器响应进入所述监视模式并且以预定位置，开始执行所述监视程序的监视模式进入指令。

56. 如权利要求 55 所述的方法，其特征在于，所述监视模式进入指令是模式切换软件中断指令，而所述预定位置是与所述模式切换软件中断指令有关的模式切换软件中断矢量。

57. 如权利要求 56 所述的方法，其特征在于，根据所述处理器是在安全模式还是在非安全模式中，不同模式切换软件中断矢量与所述模式切换软件中断指令有关。

58. 如权利要求 34 所述的方法，其特征在于，所述至少一个安全模式包括下述的一个或多个：

安全监督模式；

安全系统模式；

安全中止异常模式；

安全未定义异常模式；

安全中断模式；

安全快速中断模式；以及

安全用户模式。

59. 如权利要求 34 所述的方法，其特征在于，所述至少一个非安全模式包括下述的一个或多个：

非安全监督模式；

非安全系统模式；

非安全中止异常模式；

非安全未定义异常模式；

非安全中断模式；
非安全快速中断模式；以及
非安全用户模式。

60. 如权利要求 34 所述的方法，其特征在于，所述处理器可在所述安全域内的至少一个特许安全模式中操作，该特许安全模式不能从所述安全域改变到所述非安全域，所述特许安全模式具有允许从所述特许安全模式改变成所述监视模式的特权，不必重定向程序执行点。

61. 如权利要求 60 所述的方法，其特征在于，所述处理器响应安全标记并且当在所述监视模式中时，所述处理器可用来改变所述安全标记以便在所述安全域和所述非安全域中改变，所述安全标记除所述监视模式外，不能访问写入。

62. 如权利要求 60 所述的方法，其特征在于，所述处理器响应软件中断指令来通过重定向如由与所述软件中断指令有关的中断矢量指定的程序执行点，执行切换到所述监视模式。

63. 如权利要求 60 所述的方法，其特征在于，所述非安全域包括至少一个特许非安全模式，不能执行在没有重定向程序执行点的情况下，从所述至少一个特许非安全模式改变成所述监视模式。

64. 如权利要求 60 所述的方法，其特征在于，在复位时，所述处理器进入所述特许安全模式。

65. 如权利要求 60 所述的方法，其特征在于，在从所述安全域改变到所述非安全域前，正在所述监视模式中执行的监视程序切换到所述安全特许模式以允许保存在所述安全特许模式切换到所述监视模式前，从所述监视模式内不能访问的数据，用于到所述非安全域的所述改变。

66. 如权利要求 60 所述的方法，其特征在于，在所述监视模式中能访问的存储区也能在所述特许安全模式中访问。

用于在安全模式和非安全模式间切换处理器的装置和方法

技术领域

本发明涉及数据处理系统。更具体地说，本发明涉及数据处理系统中的安全和非安全处理模式间切换的控制。

背景技术

数据处理装置通常包括用于运行加载在数据处理装置上的应用程序的处理器。该处理器将在操作系统的控制下操作。运行任何特定的应用程序所需的数据通常存储在数据处理装置的存储器中。将意识到数据可以由包含在应用程序内的指令和/或在处理器上执行那些指令期间使用的实际数据值组成。

许多例子当中，由至少一个应用程序使用的数据是不应当被处理器上运行的其他应用程序访问的敏感数据。其中一个例子诸如数据处理装置是智能卡，以及一个应用程序是使用敏感数据，诸如例如安全密钥（secure key）来执行确认、验证、解密等等的安全应用程序。在这些情况下，确保那些敏感数据安全以便不能由可以加载到数据处理装置上的其他应用程序，例如为试图访问那个安全数据的目的，而加载在数据处理装置上的黑客应用程序访问是非常重要的。

在已知的系统中，确保操作系统提供足够的安全性以确保一个应用程序的数据不能由在操作系统的控制下运行的其他应用程序访问，已然成为操作系统开发员的典型任务。然而，随着系统变得更复杂，通常的趋势是使操作系统变得更大以及更复杂，而在这样的情况下，确保操作系统本身内的足够的安全性，变得日益困难。

在美国专利申请 US2002/0007456A1 和美国专利 US6, 282, 657 和 US6, 292, 874B 中公开了试图提供敏感数据的安全存储和提供防恶意程序代码的保护的系统的例子。

因此，将期望提供用于试图保持包含在数据处理装置的存储器内的那些安全数据的安全性的改进技术。

发明内容

从一个方面可以看出，本发明提供用于处理数据的装置，所述装置包括：

处理器，可在多种模式和多个域中操作，所述多个域包括安全域或非安全域，所述多种模式包括：

为所述安全域中的模式的至少一个安全模式；

为所述非安全域中的模式的至少一个非安全模式；以及
监视模式，其特征在于，

当所述处理器正在执行安全模式中的程序时，所述程序访问当所述处理器正在非安全模式中操作时不能访问的安全数据；以及

经所述监视模式，发生所述安全模式和所述非安全模式间的切换，所述处理器至少部分在所述监视模式中操作来执行监视程序以便管理所述安全模式和所述非安全模式间的切换。

本发明意识到控制可在安全和非安全模式中操作的系统的安全性中的关键方面在于如何控制那些安全和非安全模式间的切换。特别地，执行这种切换的方法越灵活，那么，安全性中的潜在的易受攻击性越大。本发明通过提供任何安全模式和非安全模式间的所有切换必须经监视模式进行的系统来解决这一问题。完全可以在监视模式中执行切换或者可以使用从监视模式切换到一些安全模式（例如安全特许模式）来执行与切换有关的一些任务，诸如上下文保存/恢复。因此，假如监视模式或安全核心的安全性不是被直接或间接破坏的，则不可能进行非安全模式和安全模式间的未授权切换。该结构还允许监视模式设计的相对简单，这是因为其责任被限定在支持安全域和非安全域间的切换所需的范围内。通常，监视模式越简单，越不易破坏安全性。监视程序（内核）至少部分在监视模式中操作以便控制安全和非安全域间的切换—一部分监视核心可在安全特许模式而不是监视模式中操作。

作为安全域和非安全域间的切换的管理的一部分，由监视模式提供的操作的例子是监视程序（至少部分在监视模式中运行，但是要求切换到安全特许模式以便执行上下文保存/恢复或其他操作）可用来刷新在安全域和非安全域间共享的处理器的寄存器组的至少一部分。这防止了存在于在安全域的寄存器中的任何数据，当切换到非安全域时，仍然驻留在那些寄存器中，从而不利地使数据变成可在非安全域中访

问。

作为另一可能的实施例，可以提供单独的非安全和安全寄存器组。该方法在降低安全域和非安全域间的切换所需的时间方面具有某些优点，但它还与另外的硬件需求不利有关，同时还使它不利于将数据从非安全域传送到安全域。

可以用各种不同方式启动安全域和非安全域间的切换，但这些将期望共享发送到在用于提供服务的监视模式中操作的监视程序内的一个或多个固定位置的特征。将进入监视程序的可能进入点控制到较少数目的固定点内，增加了安全性。

相反，在安全域中，使用以方便上下文切换的方式，应用于切换到安全用户模式的正常机制，安全特许模式可以切换到监视模式，以及允许有利地简化在监视模式中运行的软件。安全特许模式具有与监视模式类似的安全状态。

在本发明的一个优选形式中，切换采用调用监视程序中的固定点的形式。

在本发明的另一优选形式中，除在监视模式中时，通过程序的任何尝试触发切换以便在安全域和非安全域间改变系统，诸如通过写入存储安全状态变量或程序状态变量。这些尝试可以通过硬件来触发并用来触发调用监视程序中的固定点。

当从非安全模式启动进入监视模式时，那么可以配置处理器存储程序计数值，当返回到非安全模式时，将恢复该程序计数值，以便在中断点恢复执行处理。对当返回到非安全模式时需要恢复的一个或多个处理器状态，执行类似的状态存储。

可能引发的一个问题是当处于安全模式中的处理器在要求切换到非安全模式以便能适当地处理异常时，会产生异常。在这种情况下，通常，安全模式和非安全模式间的切换是经监视模式的，以及监视模式负责在适当点重启安全处理，而不是依赖于非安全模式来恢复安全模式处理，因为这能允许非安全模式获得有关当系统处于安全模式中时，在发生异常的点的例如程序计数值或处理器状态寄存器值的安全区域的一些知识。

提供安全域和非安全域间的清楚划分的一种方法是将安全状态标记与系统关联。通过提供安全状态标记仅在监视模式中可写入，可以

将安全状态标记的控制集中在监视模式中。

可以使用本发明的系统的优选方法是提供一种非安全操作系统，当在非安全模式先操作时控制处理器，以及提供一种安全操作系统（在某些情况下，可以是删节的安全核心，）当在安全模式下操作时，控制处理器。监视程序由部分安全操作系统组成是比较有利的。

本发明的优选实施例意识到系统的安全性的潜在的弱点可以通过正在控制安全域和非安全域间的切换时的监视程序的中断产生。因此，在本发明的优选实施例中，当处于监视模式时，禁止能触发异常处理（正常处理流程的中断）的异常条件的至少一个。

为改进经监视模式，进行安全域间的切换的速度，当太考虑成本/利润时，本发明的一些实施例提供专用寄存器（影像寄存器），当进入监视模式时，替代处理器的寄存器组内的相应的通用寄存器。因此，这些专用寄存器可以直接由监视程序使用，从而允许监视程序在执行其自己的操作前，避免对现有的寄存器内容的保存，因此，降低了在监视模式中实现切换所花的时间。当在监视模式中禁止异常时，降低在监视模式中所花的时间很显著。

整体上，在系统的上下文中处理异常的方法与异常，诸如中断一样重要，总的来说，对系统的安全性存在潜在的弱点。为允许系统响应适合于特定环境的异常的方式，本发明的优选实施例提供中断屏蔽寄存器，存储一个或多个参数，用于指定哪些异常应当由正在模式中执行的异常处理程序处理以及哪些异常应当由当异常发生时，正在当前域中的模式中执行的异常处理程序处理。因此，根据异常产生的类型，可以使系统切换到安全域以便处理该异常或可以保持当前的、可能的非安全域中。监视模式有能力切换安全域以及可以视为安全域，因为可在该模式中设置安全状态标记以及可以将系统的整个安全性视为驻留在监视模式的安全性。在一些方式中，系统的安全状态可以视为安全状态标记和系统是否在监视模式中的 OR。可以将某些异常配置成迫使在监视模式中处理。另外，可以将异常留给在不管在系统的安全或非安全模式中操作的异常处理程序中处理。

本发明的优选实施例提供专用监视模式进入指令，硬件响应这些指令，进入监视模式以及开始执行在预定位置处的监视程序。这些监视模式进入指令采用具有相关模式切换软件中断矢量的模式切换软件

中断的形式。配置成进入监视模式的主要方式是通过使用这一指令以及配置指令的行为来使硬件以清楚和紧密定义的方式响应，诸如在固定点开始执行监视程序特别有利。

将意识到在安全域和非安全域内，可以提供各种不同处理模式。不一定要求在两个域中提供相同的模式。适当的模式的例子是监督模式、系统模式、中止异常模式、未定义异常模式、中断模式、快速中断模式和用户模式。

在优选实施例中，处理器可以在所述安全域中的至少一种特许安全模式下操作，该安全模式不能从安全域改变到非安全域，所述具有特权的特许安全模式允从所述特许安全模式改变到所述监视模式，而无需重新定向程序的执行点。

监视程序主要负责管理安全域和非安全域之间的来回改变。监视程序至少部分在监视模式中执行。例如，监视程序可以至少部分在监视模式中以及部分在安全模式中执行。安全核心在特许安全模式中执行。通过允许特许安全模式切换到监视模式，监视程序的编写可以通过传送到安全核心的一些函数而变得更加简单，即使可能会增加安全核心的复杂度。使监视程序简单化而增加其安全性在于其更易于证明其安全：即，仅执行所期望做的很重要，因为它横跨安全和非安全域。然而，安全核心仅在安全域中执行。监视程序能通过切换到那些模式来保存安全特许模式的状态，因为它可以随后切换回监视模式。

在优选实施例中，处理器用来改变安全标记以便在安全域和非安全域间改变，该安全标记为除监视模式外不可写入访问的。将意识到安全标记可以或不可以从监视模式之外读取访问的。

本技术允许从特许安全模式改变成监视模式，而无需重定向程序执行的同时，优选实施例还响应软件中断指令来执行切换到监视模式，但在这种情况下，需要由与软件中断指令相关的中断矢量指定的程序执行点的重定向。该软件中断机制与从特许安全模式提供的相比，灵活较低，因为通过软件中断机制，使程序流通过中断机制。能从安全或非安全特许模式调用软件中断机制，同时降低安全性风险，因为能更密切地控制中断事务的代码。

可能安全域仅包括安全特许模式的同时，优选实施例通常将包括至少一个特许非安全模式，不重定向程序执行点就不能执行从该特许

非安全模式到监视模式的切换。安全域内的模式的多样性便于正常结构和组织在安全域内执行的操作系统。

在本发明的优选实施例中，一旦复位，处理器进入特许安全模式。复位到特许安全模式具有多种不同优点，诸如允许更易于与非安全环境中的非安全识别操作系统的向后兼容性。通过复位到特许安全模式，这种安全不敏感系统能保留在安全域中，即使它不要求安全操作。当系统意识到安全性，在特许安全模式中引导还允许大量代码的可能性以及支持由允许在监视模式内执行监视模式的特许安全模式内的安全核心提供的引导操作以便以简单的形式保存，从而能更容易确保其安全性。

为允许安全模式和特许安全模式间的适当等同，优选实施例是可在监视模式中访问的存储区也可在特许安全模式中访问。

从本发明的另一方面可以看出，提供处理数据的方法，所述方法包括步骤：

通过可在多种模式和多个域中操作的处理器执行程序，所述多个域包括安全域或非安全域，所述多种模式包括：

为所述安全域中的模式的至少一个安全模式；

为所述非安全域中的模式的至少一个非安全模式；以及

监视模式，其特征在于，

当所述处理器正在执行安全模式中的程序时，所述程序访问当所述处理器正在非安全模式中操作时不能访问的安全数据；以及

经所述监视模式，进行所述安全模式和所述非安全模式间的切换，所述处理器执行至少部分在所述监视模式的监视程序来管理所述安全模式和所述非安全模式间的切换。

附图说明

参考如在附图中所述的优选实施例，仅通过举例来进一步描述本发明，其中：

图 1 是示意性地示例说明根据本发明的优选实施例的数据处理装置的框图；

图 2 示意性地示例说明在非安全域和安全域（secure domain）中操作的不同程序；

图 3 示意性地示例说明与不同安全域有关的处理模式矩阵；

图 4 和 5 示意性地示例说明处理模式和安全域间的不同关系；

图 6 示例说明根据处理模式，处理器的寄存器组的一个程序员模型；

图 7 示例说明提供用于安全域和非安全域的单独的寄存器组的例子；

图 8 示意性地示例说明具有经单独的监视模式进行的安全域间的切换的多个处理模式；

图 9 示意性地示例说明使用模式切换软件中断指令，用于安全域切换的情形；

图 10 示意性地示例说明如何由系统执行非安全中断请求和安全中断请求的一个例子；

图 11A 和 11B 示意性地示例说明根据图 10，非安全中断请求处理的例子和安全中断请求处理的例子；

图 12 示例说明与图 10 所示相比，用于处理非安全中断请求信号和安全中断请求信号的另一方案；

图 13A 和 13B 示例说明根据图 12 所示的方案，用于处理非安全中断请求和安全中断请求的示例情况；

图 14 是矢量中断表的例子；

图 15 示意性地示例说明与不同安全域有关的多个矢量中断表；

图 16 示意性地示例说明异常控制寄存器；

图 17 是示例说明以修改安全域设置的方式，试图修改处理状态寄存器的指令如何生成反过来触发进入监视模式和运行监视程序的单独的模式改变异常的流程图；

图 18 示意性地表示在多个模式中操作的处理器的控制的线程，其中中断监视模式中的任务；

图 19 示意性地表示在多个模式中操作的处理器的控制的不同线程；

图 20 示意性地表示在多个模式中操作的处理器的控制的另一线程，其中在监视模式中允许中断；

图 21 至 23 示例说明根据另一示例性实施例，用于在安全和非安全域间切换的不同处理模式和情形的视图；

图 24 示意性地示例说明将安全处理选项增加到传统的 RAM 内核上的原理；

图 25 示意性地示例说明具有安全和非安全域和复位的处理器；

图 26 示意性地示例说明使用软件伪中断，将处理请求递送到暂停的操作系统；

图 27 示意性地示例说明经软件伪中断，将处理请求递送到暂停的操作系统的另一例子；

图 28 是示意性地示例说明在接收在图 26 和 27 中所产生的类型的软件伪中断后执行的处理的流程图；

图 29 和 30 示意性地示例说明在安全操作系统跟踪由非安全操作系统执行的可能任务切换后的任务；

图 31 是示意性地示例说明在接收图 29 和 30 的安全操作系统处的呼叫时执行的处理的流程；

图 32 是示意性地示例说明在具有多个操作系统的系统中出现的中断优先级反转的问题的图，其中可以由不同操作系统处理不同中断；

图 33 是示意性地示例说明使用插桩中断（处理器来避免在图 32 中所示的问题的图；

图 34 示意性地示例说明如何处理不同类型和优先级的中断，取决于它们是否能够由采用不同操作系统来实现的中断进行中断；

图 35 示例说明当处理器在监视模式中操作时，如何用监视程序模式特定处理器结构数据来覆盖处理器结构数据；

图 36 是示例说明根据本发明的一个实施例，当在安全域和非安全域间转变时，如何切换处理器结构数据的流程图；

图 37 是示例说明控制访问存储器的用在本发明的一个实施例中的存储器管理逻辑的图；

图 38 是示例说明用来控制存储器的访问的本发明的第二实施例的存储器管理逻辑的框图；

图 39 是示例说明处理指定虚拟地址的存储器访问请求的存储器管理逻辑内，在本发明的一个实施例中执行的过程的流程图；

图 40 是示例说明在处理指定物理地址的存储器访问请求的存储器管理逻辑内，在本发明的一个实施例中执行的过程的流程图；

图 41 示意性地示例说明当发出存储器访问请求的设备正在非安全

模式中操作时，优选实施例的分区校验器（partition checker）如何用来防止访问物理地址；

图 42 是示例说明在本发明的优选实施例中，非安全页表和安全页表的用法的图；

图 43 是示例说明用在优选实施例的主翻译后援缓冲器（TLB）内的两种形式的标记的图；

图 44 示例说明在本发明的一个实施例中，在引导级后如何分区存储器；

图 45 示例说明根据本发明的实施例，根据引导分区的性能，通过存储器管理单元映射非安全存储器；

图 46 示例说明根据本发明的实施例，如何将一部分存储器的权利修改成允许安全应用程序与非安全应用程序共享存储器；

图 47 示例说明根据本发明的一个实施例，如何将设备连接到数据处理装置的外部总线；

图 48 是示例说明根据本发明的第二实施例，如何将设备耦合到外部总线的框图；

图 49 示例说明在使用单个页表集的实施例中，物理存储器的配置；

图 50A 示例说明使用两个 MMUs 来经中间地址执行从虚拟地址到物理地址的转换的配置；

图 50B 通过举例，示例说明使用两个 MMUs 来经中间地址，执行从虚拟地址到物理地址的转换的另一配置；

图 51 通过举例，示例说明用于安全域和非安全域的物理地址空间和中间地址空间的对应关系；

图 52 示例说明通过操作与第二 MMU 有关的页表，在安全和非安全域间交换存储区；

图 53 是示例说明使用单个 MMU 的实现的实施例，以及其中，主 TLB 中的差错将导致被调用来确定虚拟到物理地址转换的异常；

图 54 是示例说明由处理器内核执行以便对在图 53 的 MMU 的主 TLB 中出现未命中时发出的异常起使用的过程的流程图；

图 55 是示例说明在一个实施例的数据处理装置内提供的元件的框图，其中高速缓存具有有关存储在单个高速缓存线中的数据是安全数

据还是非安全数据的信息；

图 56 示例说明在图 55 中所示的存储器管理单元的结构；

图 57 是示例说明在图 55 的数据处理装置内执行的、处理非安全存储器访问请求的处理的流程图；

图 58 是示例说明在图 55 的数据处理装置内执行的以便处理安全存储器访问请求的处理的流程图；

图 59 示意性地表示用于不同模式在处理器上运行的程序的监视功能的可能粒度；

图 60 表示启动不同监视功能的可能方式；

图 61 表示用于控制不同监视功能的可用性的控制值的表；

图 62 表示正沿触发的触发器视图；

图 63 表示扫描链单元 (scan chain cell)；

图 64 表示在扫描链中的多个扫描链单元；

图 65 表示调试 TAP 控制器；

图 66A 表示具有 JADI 输入的调试 TAP 控制器；

图 66B 表示具有旁通寄存器 (bypass register) 的扫描链单元；

图 67 示意性地示例说明包括内核、扫描链和调试状态和控制寄存器的处理器；

图 68 示意性地示例说明控制调试或跟踪初始化的因素；

图 69A 和 69B 表示调试粒度的概述；

图 70 示意性地示例说明在运行时的调试粒度；以及

图 71A 和 71B 分别表示在安全区域中允许调试时和不允许调试时的监视调试。

具体实施方式

图 1 是示例说明根据本发明的优选实施例的数据处理装置的结构图。数据处理装置包含处理器内核 10，在其内提供用来执行指令序列的算术逻辑单元 (ALU) 16。ALU16 所需的数据存储在寄存器组 14 中。内核 10 具有各种监视功能以允许俘获表示处理器内核的活动性的诊断数据。例如，提供嵌入跟踪模块 (ETM) 22，用于根据定义将跟踪的活动性的 EMT22 内的某一控制寄存器 26 的内容，产生处理器内核的某些活动性的实时跟踪。通常，将跟踪信号输出到跟踪缓冲器，随后可从

跟踪缓冲器对其进行分析。提供矢量中断控制器 21，用于管理由各种外设（未示出）引发的多个中断的维护。

另外，如图 1 所示，内核 10 内所能提供的另一监视功能是调试功能，数据处理装置的外接调试应用程序能够经由联合测试访问组（JTAG）控制器 18 与内核 10 通信，联合测试访问组（JTAG）控制器 18 与一个或多个扫描链 12 耦合。能经扫描链 12 和 JTAG 控制器 18，将有关处理器内核 10 的各个部分的状态的信息输出到外部调试应用程序。使用内置在线仿真电路（ICE）20 来将识别何时启动和停止调试功能的条件存储在寄存器 24 内，因此，例如，将用来存储断点、观察点等等。

经配置成管理由内核 10 发出的访问数据处理装置的存储器中的单元的存储器访问请求的存储器管理逻辑 30，将内核 10 耦合到系统总线 40。通过直接连接到系统总线 40，例如图 1 中所示的紧耦合存储器（TCM）36 和高速缓存 38，可以嵌入存储器的某些部分。也可以提供另外的设备，用于访问这些存储器，例如直接存储器访问（DMA）控制器 32。通常，将提供各种控制寄存器 34，用于定义芯片的各个元件的某些控制参数，这些控制寄存器在此也称为协处理器（CP）15 寄存器。

经外部总线接口 42，包含内核 10 的芯片也可以耦合到外部总线 70（例如根据由 ARM Limited 开发的“高级微控制器总线体系结构”（AMBA）规格操作的总线），以及也可以将各种设备连接到外部总线 70。这些设备可以包括主设备，诸如数字信号处理器（DSP）50，或直接存储器访问（DMA）控制器 52，以及各种从设备，诸如引导 ROM44、屏幕驱动器 46、外部存储器 56、输入/输出（I/O）接口 60 或键存储单元 64。在图 1 中所示的这些各种从设备也均能视为数据处理装置的整个存储器的包含部件。例如，引导 ROM44 将形成数据处理装置的可访问存储器的一部分，如外部存储器 56。另外，诸如屏幕驱动器 46、I/O 接口 60 和键存储单元 64 的设备均分别包括内部存储元件，诸如寄存器或缓冲器 48、62、66，可单独地寻址为数据处理装置的整个存储器的一部分。如稍后更详细地论述，存储器的一部分，例如外部存储器 56 的一部分，将用来存储定义与存储器访问控制有关的信息的一个或多个页表 58。

如本区域的技术人员将意识到的，外部总线 70 通常将具有判优器

和解码器逻辑 54、使用判优器在由多个主设备，例如内核 10、DMA32、DSP50、DMA52 等等发出的多个存储器访问请求间判优，而使用解码器来确定外部总线上的哪个从设备将处理任何特定的存储器访问请求。

尽管在一些实施例中，可以在包含内核 10 的芯片外提供外部总线，在其他实施例中，也可以在具有内核 10 的芯片上提供外部总线。其优点在于与外部总线为芯片外时相比，外部总线上的安全数据更容易保密。当外部总线是芯片外时，可以使用数据加密技术来增加安全数据的安全性。

图 2 示意性地示例说明在具有安全域和非安全域的处理系统上运行的各种程序。该系统具有至少部分在监视模式中执行的监视程序 72。在该示例性实施例中，安全状态标记仅可在监视模式中访问的写入，以及可以通过监视程序 72 写入。监视程序 72 负责管理安全域和非安全域间的来回改变。从核心外的观点看，监视程序模式总是安全以及监视程序位于安全存储器中。

在非安全域内，提供非安全操作系统 74 和与非安全操作系统 74 协同执行的多个非安全应用程序 76、78。在安全域中，提供安全核心程序 80。安全核心程序 80 能视为形成安全操作系统。通常，这些安全核心程序 80 将设计成仅提供对必须在安全域中提供的处理活动性很关键的那些功能，以便安全核心 80 能尽可能小和简单，因为这将使其更趋安全。示出了结合安全核心 80 执行的多个安全应用程序 82、84。

图 3 示例说明与不同安全域有关的处理模式的矩阵。在该示例性例子中，相对于安全域，处理模式是对称的，因此，模式 1 和模式 2 以安全和非安全形式存在。

监视模式在系统中具有最高安全访问级，以及在该示例性实施例中，是有权在任一方向中，在非安全域和安全域间切换系统的唯一模式。因此，经切换，发生到监视模式的所有域切换以及在监视模式内执行监视程序 72。

图 4 示意性地示例说明另一组非安全域处理模式 1、2、3 和 4 和安全域处理模式 a、b 和 c。与图 3 的对称排列相反，图 4 表示一些处理模式可能不存在于安全域的一个或另一个中。监视模式 86 同样示例说明为横跨于非安全域和安全域。监视模式 86 能视为安全处理模式，因为在该模式中可以改变安全状态标记以及监视模式中的监视程序 72

具有自己设置安全状态标记的能力，总的来说，有效地提供系统内最大安全级。

图 5 示意性地示例说明关于安全域的处理模式的另一配置。在该配置中，识别安全和非安全域以及另外的域。该另外的域可以是以不需要与所示的安全域或非安全域交互作用的方式，独立于系统的其他部件，这样使得其发射与所属的这些部件无关。

如将意识到的，处理系统，诸如微型处理器通常具有寄存器组 88，其中，可以存储操作数值。图 6 示例说明在某些处理模式中，具有为某些寄存器数提供的专用寄存器的示例性寄存器组的程序员模式视图。更具体地说，图 6 的例子是具有用于每个处理模式的专用保存程序状态寄存器、专用栈指针寄存器和专用连接寄存器 R14 的已知 ARM 寄存器组（例如在 ARM Limited 的 ARM 处理器，Cambridge, England 中提供的）的扩展，但在这种情况下，通过提供监视模式来扩展。如图 6 所示，中断模式具有所提供的另外的专用寄存器以便在进入快速中断模式时，不需要保存，然后从其他模式恢复寄存器内容。用与快速中断模式相似的方式，在另外的实施例，监视模式可以具有另外的专用寄存器以便加速安全域转换的处理，以及降低与这些转换有关的系统等待时间。

图 7 示意性地示例说明另一实施例，其中以分别用在安全域和非安全域的两个完全和单独的寄存器组的形式，提供寄存器组 88。这是当切换到非安全域时，能防止存储在可在安全域中操作的寄存器内的安全数据被访问的一种方法。然而，这样配置妨碍数据从非安全域向安全域传送的可能性，而通过使用在寄存器中快速和有效机制对其进行替换，则可以允许并且是所期望的，该寄存器在非安全域和安全域中都是可以访问的。

具有安全寄存器组的重要优点是避免在从一个区域转换到另一个之前，刷新寄存器的内容的需要。如果等待时间不是主要问题，可以使用用于安全域区域，不具有重复寄存器的更简单的硬件系统，例如图 6。监视模式负责从一个域转换到另一个域。通过至少部分在监视模式中执行的监视程序，执行恢复上下文、保存先前上下文以及刷新寄存器。该系统由此表现为虚拟化模式。下面将进一步论述这种实施例。将参考例如在其上构造在此所述的安全特性的 ARM7 的程序员模

式。

处理器模式

代替安全模式中的复制模式，相同的模式支持安全和非安全域（见图 8）。监视模式知道内核的当前状态，不管是安全还是不安全（例如，如从所存储的 S 位所读出为协处理器配置寄存器）。

在图 8 中，只要出现 SMI（软件监视中断指令），内核进入监视模式以便适当地从一个区域切换到另一个。

参考图 9，其中从用户模式允许 SMIs：

1. 调度程序运行线程 1

2. 线程 1 需要执行安全函数=>SMI 安全调用，内核进入监视模式。

在硬件控制下，当前 PC 和 CPSR(当前处理器状态寄存器)存储在 R14_mon 和 SPSR_MON (为监视模式而保存的处理器状态寄存器) 并禁止 IRQ/FIQ 中断。

3. 监视程序执行下述任务：

- 设置 S 位（安全状态标志）

- 至少将 R14_mon 和 SPSR_mon 保存在堆栈中以便如果在运行安全应用程序时，出现异常，不会丢失非安全上下文。

- 校验是否运行新线程：安全线程 1。机制（在一些实施例中经线程 ID 表）表示线程 1 在安全区域中是有效的。

- 重新允许 IRQ/FIQ 中断。然后，在安全用户模式中，安全应用能开始。

4. 运行安全线程 1 直到它结束为止，然后转移（SIM）到监视程序模式的“从安全返回”函数上（当内核进入监视模式时，禁止 IRQ/FIQ 中断）。

5. “从安全返回”函数执行下述任务：

- 表明结束安全线程 1（例如，在线程 ID 表的情况下，从表移出线程 1）。

- 从栈非安全上下文恢复以及清除所需寄存器，以便只要返回到非安全域，就无法读取安全数据。

- 然后通过 SUBS 指令（这使程序计数器恢复到正确点以及更新状态标志），转移到非安全域，恢复 PC（从 R14_mon 恢复）和 CPSR（从 SPSR_mon）。因此，非安全域中的返回点是在线程 1 中的在前执行的

SMI 后的指令。

6. 线程 1 执行直到结束为止，然后将控制返回给调度程序。

根据具体的实施例，在监视程序和安全操作系统间可以分开一些上述功能性。

在其他实施例中，不期望在用户模式中出现允许 SMI。

安全区域进入

复位

当出现硬件复位时，禁止 MMU 以及通过所设置的 S 位，使 ARM 内核（处理器）转移到安全监督模式。只要终止安全引导，可以执行进入监视模式的 SMI，以及如果需要的话，监视能切换到非安全区域中的 OS（非安全 svc 模式）。如果期望使用传统的 OS，这能在安全监督模式中简单地引导以及忽略安全状态。

SMI 指令

能从非安全域中的任何非安全模式调用该指令（模式切换软件中断指令）（如前所述，可以期望将 SMI 限制到特许模式），但由相关矢量确定的目标进入点总是固定的并且在监视模式内。一直到 SMI 处理程序以便转移到必须运行的适当的安全函数（例如由通过指令传递的操作数控制）。

使用在图 6 类型寄存器组内的寄存器组的共享寄存器，执行将参数从非安全区域传递到安全区域。

当在非安全区域中出现 SMI 时，用硬件，ARM 内核可以执行下述动作：

- 使 SMI 矢量（由于现在将处于监视模式中，允许安全存储器访问）转移到监视模式中
- 将 PC 保存在 R14_mon 中以及使 CPSR 保存在 SPSR_mon 中
- 使用监视程序，设置 S 位
- 在监视模式中，开始执行安全异常处理程序（在多线程的情况下，恢复/保存上下文）
- 转移到安全用户模式（或另一模式，如 svc 模式）以便执行适当的函数
- 禁止 IRQ 和 FIQ，而内核处于监视模式中（增加等待时间）

安全区域退出

退出安全区域的两种可能：

- 结束安全功能以及返回到已经调用该函数的先前非安全模式。
- 通过非安全异常（例如 IRQ/FIQ/SMI）中断安全功能。

安全函数的正常结束

安全函数正常终止以及需要以正好在 SMI 后的指令恢复非安全区域中的应用程序。在安全用户模式中，执行“SMI”指令以便返回到通过对应于“从安全区域返回”例程的适当参数，返回到监视模式。在该阶段，刷新寄存器以避免在非安全和安全区域之间的数据泄漏，然后恢复非安全上下文通用寄存器以及用在非安全区域中已经具有的值更新非安全组寄存器。R14_mon 和 SPSR_mon 由此获得适当的值，通过执行“MOVS PC, R14”指令来恢复 SMI 后的非安全应用。

因非安全异常导致的安全函数退出

在这种情况下，未结束安全函数，以及在进入非安全异常处理程序后，必须保存安全上下文，不管是否需要处理这些中断。

安全中断

对安全中断，存在几种可能性。

根据下述，提出两种可能的解决方案，具体取决于：

- 是何种中断（安全还是非安全）
- 当出现 IRQ 时（不管是在安全还是非安全区域中），内核处于何种模式。

解决方案一

在该解决方案中，要求两个不同的管脚来支持安全和非安全中断。

当在非安全区域中，如果

- 出现 IRQ，内核进入 IRQ 模式以便处理该中断，如同在 ARM 内核中，诸如 ARM7。
- 出现 SIRQ，内核进入监视模式以便保存非安全上下文，然后进入安全 IRQ 处理程序以便处理安全中断。

当在安全区域中，如果

- 出现 SIRQ，内核进入安全 IRQ 处理程序。内核不离开安全区域
- 出现 IRQ，内核进入监视模式，其中保存安全上下文，然后进

入非安全 IRQ 处理程序以便处理该非安全中断。

换句话说，当出现不属于当前区域的中断时，内核直接进入监视模式，否则停留在当前区域（见图 10）。

在安全区域中出现 IRQ

见图 11A:

1. 调度程序运行线程 1。
2. 线程 1 需要执行安全函数=>SMI 安全调用，内核进入监视模式。

将当前 PC 和 CPSR 保存在 R14_mon 和 SPSR_MON 中，禁用 IRQ/FIQ。

3. 监视处理程序（程序）执行下述任务：

- 设置 S 位。

- 至少将 R14_mon 和 SPSR_mon 保存在堆栈中（以及还可能压入其他寄存器），以便如果在运行安全应用程序的同时，出现异常，也不至丢失非安全上下文。

- 校验运行新线程：安全线程 1：（经线程 ID 表）机制表明线程 1 在安全区域中有效。

- 然后在安全用户模式中开始安全应用程序。然后重新允许 IRQ/FIQ。

4. 在运行安全线程 1 时出现 IRQ。内核直接跳到监视模式（特定的矢量），以及在监视模式中，将当前 PC 存储在 R14_mon 中以及将 CPSR 存储在 SPSR_mon 中，（然后禁用 IRQ/FIQ）。

5. 必须保存安全上下文，恢复先前的非安全上下文。监视处理程序可以切换到 IRQ 模式来通过适当的值更新 R14_irq/SPSR_irq，然后将控制传递到非安全 IRQ 处理程序。

6. IRQ 处理程序提供 IRQ，然后将控制返回到非安全区域中的线程 1。通过使 SPSR_irq 和 R14_irq 再存入 CPSR 和 PC 中，现在，线程 1 指向已经中断的 SMI 指令。

7. 重新执行 SMI 指令（指令与 2 相同）。

8. 监视处理程序注意到先前已经中断该线程，以及恢复线程 1 上下文。然后转移到用户模式中的安全线程 1，指向已经中断指令上。

9. 运行安全线程 1 直到完成为止，然后转移到监视模式中的“从安全返回”函数上（专用 SMI）。

10. “从安全返回”函数执行下述任务：

- 表明完成安全线程 1（即，在线程 ID 表的情况下，从表移出线程 1）。
- 从栈非安全上下文恢复以及清除所需寄存器，以便只要返回到非安全区域，就无法读取安全数据。
- 通过 SUBS 指令，使转移回非安全区域，恢复 PC（从恢复的 R14_mon 和 CPSR（从 SPSR_mon）。因此，非安全区域中的返回点应当是线程 1 中先前执行的 SMI 后的指令。

11. 执行线程 1 直到结束为止，然后将控制返回调度程序。

在非安全区域中出现 SIRQ

见图 11B：

1. 调度程序运行线程 1
2. 当安全线程 1 正运行时，出现 SIRQ。内核直接跳转到监视模式（特定矢量）以及在监视模式中，将当前 PC 存储在 R14_mon 和将 CPSR 存储在 SPSR_mon 中，然后禁用 IRQ/FIQ。
3. 必须保存非安全上下文，然后内核进入安全 IRQ 处理程序。
4. IRQ 处理程序提供 SIRQ，然后使用具有适当参数的 SMI，将控制返回到监视模式处理程序。
5. 监视处理程序恢复非安全上下文以便 SUBS 指令使内核返回到非安全区域以及恢复中断线程 1。
6. 执行线程 1 直到结束为止，然后使控制返回到调度程序。

图 11A 的机制具有提供进入安全区域的确定方法的优点。然而，存在与中断优先级有关的一些问题：例如，当 SIRQ 正在安全中断处理程序中运行时，具有较高优先级的非安全 IRQ 会发生。只要完成非安全 IRQ，需要重建 SIRQ 事件以便内核能恢复安全中断。

解决方案二

在该机制中（见图 12），两个不同管脚，或仅一个可以支持安全和非安全中断。具有两个管脚降低中断等待时间。

当在非安全区域中时，如果

- 出现 IRQ，内核进入 IRQ 模式以便处理该中断，如在 ARM7 系统中。
- 出现 SIRQ，内核进入 IRQ 处理程序，其中，SMI 指令将使内核转移到监视模式以便保存非安全上下文，然后进入安全 IRQ 处理程序

以便处理安全中断。

当在安全区域中，如果

- 出现 SIRQ，内核进入安全 IRQ 处理程序。内核不离开安全区域
- 出现 IRQ，内核进入安全 IRQ 处理程序，其中，SMI 指令将使内核转移到监视模式（其中，保存安全上下文），然后进入非安全 IRQ 处理程序以便处理该非安全中断。

在安全区域中出现 IRQ

见图 13A：

1. 调度程序运行线程 1。

2. 线程 1 需要执行安全函数=>SMI 安全调用，内核进入监视模式。

将当前 PC 和 CPSR 保存在 R14_mon 和 SPSR_MON 中，禁用 IRQ/FIQ。

3. 监视处理程序执行下述任务：

- 设置 S 位。

- 至少将 R14_mon 和 SPSR_mon 保存在堆栈中（最后其他寄存器），以便如果在运行安全应用程序的同时，出现异常，也不至于丢失非安全上下文。

- 校验运行新线程：安全线程 1：（经线程 ID 表）机制表示在安全区域中，线程 1 有效。

- 然后在安全用户模式中开始安全应用程序。重新允许 IRQ/FIQ。

4. 在运行安全线程 1 时出现 IRQ。内核直接跳到安全 IRQ 模式。

5. 内核将当前 PC 存储在 R14_irq 中以及将 CPSR 存储在 SPSR_irq 中。IRQ 处理程序检测这是非安全中断以及执行 SMI 来通过适当参数进入监视模式。

6. 必须保存安全上下文，恢复先前的非安全上下文。监视处理程序通过读取 CPSR，知道 SMI 来自何处。还能进入 IRQ 模式来读取 R14_irq/SPSR_irq 以便适当地保存安全上下文。还能将非安全上下文保存在相同的寄存器中，而这些非安全上下文一旦 IRQ 事务处理结束就必须恢复。

7. IRQ 处理程序提供 IRQ，然后将控制返回到非安全区域中的线程 1。通过使 SPSR_irq 和 R14_irq 再存入 CPSR 和 PC 中，现在，内核指向已经中断的 SMI 指令。

8. 重新执行 SMI 指令（指令与 2 相同）。
9. 监视处理程序注意到先前已经中断该线程，以及恢复线程 1 上下文。然后转移到用户模式中的安全线程 1，指向已经中断指令上。
10. 运行安全线程 1 直到完成为止，然后转移到监视模式中的“从安全返回”函数上（专用 SMI）。
11. “从安全返回”函数执行下述任务：
 - 表示完成安全线程 1（即，在线程 ID 表的情况下，从表移出线程 1）。
 - 从栈非安全上下文恢复以及清除所需寄存器，以便只要返回到非安全区域，能读取安全数据。
 - 通过 SUBS 指令，使转移回非安全区域，恢复 PC（从恢复的 R14_mon 和 CPSR（从 SPSR_mon）。非安全区域中的返回点应当是线程 1 中先前执行的 SMI 后的指令。
12. 执行线程 1 直到结束为止，然后将控制返回调度程序。

在非安全区域中出现 SIRQ

见图 13B：

1. 调度程序运行线程 1
2. 当安全线程 1 正运行时，出现 SIRQ。
3. 内核直接跳转 irq 模式以及将当前 PC 存储在 R14_irq 和将 CPSR 存储在 SPSR_irq 中，然后禁用 IRQ。IRQ 处理程序检测这是 SIRQ 以及通过适当的参数，执行 SMI 指令。
4. 只要在监视模式中，必须保存非安全上下文，然后内核进入安全 IRQ 处理程序。
5. IRQ 处理程序提供 SIRQ 服务例程，然后通过具有适当参数的 SMI，将控制返回到监视模式处理程序。
6. 监视处理程序恢复非安全上下文以便 SUBS 指令使内核返回到非安全区域以及恢复中断 IRQ 处理程序。
7. 然后，通过执行 SUBS，IRQ 处理程序返回到非安全线程。
8. 执行线程 1 直到结束为止，然后将控制返回到调度程序。

通过图 12 的机制，在嵌套中断的情况下，不需要重建 SIRQ 事件，但不保证将执行安全中断。

异常矢量

保存至少两个物理矢量表（尽管从虚拟地址的观看点，它们看起来象单一矢量表），一个用于非安全存储器中的非安全区域，一个用于安全存储器（不能从非安全区域访问）中的安全区域。用在安全和非安全区域中的不同虚拟到物理存储器映射有效地允许相同的虚拟存储器地址访问存储在物理存储器中的不同矢量表。监视模式可以总是使用单调存储器映射以便在物理存储器中提供第三矢量表。

如果中断按照图 12 机制，将存在用于每个表的如图 14 所示的下述矢量。在安全和非安全存储器中复制该矢量集。

异常	矢量偏差	相应模式
复位	0x00	监督模式 (S 位设置)
未定义	0x04	监视模式/未定义模式
SWI	0x08	监督模式/监视模式
预取中止	0x0C	中止模式/监视模式
数据中止	0x10	中止模式/监视模式
IRQ/SIRQ	0x18	IRQ 模式
FIQ	0x1X	FIQ 模式
SMI	0x20	未定义模式/监视模式

NB. 复位项仅在安全矢量表中，当在非安全区域中执行复位时，内核硬件迫使进入监督模式以及设置 S 位以便能在安全存储器中访问复位矢量。

图 15 示例说明分别应用于安全模式、非安全模式和监视模式的三个异常矢量表。可以通过异常矢量编程这些异常矢量表以便满足安全和非安全操作系统的需求和特性。每一个异常矢量表都可以在存储指向存储器中的那个表的基地址的 CP15 内，具有相关矢量表基地址寄存器。当发生异常时，硬件将引用对应于该系统的当前状态的矢量表基地址寄存器来确定待使用的矢量表的基地址。另外，可以使用在不同模式中应用的不同虚拟到物理存储器映射来分开在不同物理存储器地址处存储的三个不同矢量表。如图 16 所示，在与处理器内核有关的系统（配置控制）协处理器（CP15）中提供异常中断屏蔽寄存器。该异常中断屏蔽寄存器提供与各个异常类型有关的标记。这些标记表明硬

件当前用来直接处理用于与其当前域有关的矢量还是应当强迫切换到监视模式（一种安全模式），然后按照监视模式矢量表中的矢量。仅能从监视模式写异常中断屏蔽寄存器（异常控制寄存器）。当在非安全模式中时，还能防止读取访问异常中断屏蔽寄存器。将看到图 16 的异常中断屏蔽寄存器不包括用于复位矢量的标记，因为该系统配置成总是使此跳转到如在安全矢量表中指定的安全监督模式中的复位矢量以便确保安全引导和向后兼容性。将看出在图 15 中，为完整起见，在除安全监督模式安全矢量表外，矢量表中示出了复位矢量。

图 16 还示例说明用于异常中断屏蔽寄存器内的不同异常类型的标记是可编程的，诸如在安全引导期间通过监视程序。另外，一些或全部标记在某些实现中可以通过物理输入信号来提供，例如可以硬布线安全中断标记 SIRQ 以便当接收安全中断信号时，总是使监视模式输入和执行相应的监视模式安全中断请求矢量。图 16 仅示例说明与非安全域异常有关的部分异常中断寄存器，对安全域异常，将提供类似的可编程位集。

虽然从上述可以理解，在一级，硬件要么迫使中断由当前域异常处理来控制管理来实现，要么由监视模式异常管理来实现，具体取决于异常控制寄存器标记，这仅是所应用的控制的第一级。例如，异常可以发生在安全模式中，安全模式异常矢量在安全模式异常处理程序后，但该安全模式异常处理程序确定该异常是非安全异常处理程序更好的属性，因此，利用 SMI 指令来切换到非安全模式以及调用非安全异常处理程序。逆过程也是可能的，其中硬件可以用来启动非安全处理程序，但是然后，执行安全异常处理程序或监视模式异常处理程序的直接处理的指令。

图 17 是示意性地示例说明系统的操作以便支持与新类型的异常有关的另外可能的切换请求类型的流程。在步骤 98，硬件检测正尝试改变监视模式的任何指令，如在当前程序状态寄存器（CPSR）中所表明的。当检测到该尝试时，那么触发新类型的异常，这称为 CPSR 冲突异常。在步骤 100，生成该 CPSR 冲突异常层到引用监视模式内的适当异常以及在步骤 102 运行监视程序以便处理该 CPSR 冲突异常。

将意识到除支持上述的 SMI 指令外，还可以提供与图 17 有关所述的用于启动安全域和非安全域间的切换的机制。当经 SMI 指令，进行

所有授权尝试时，可以提供该异常机制以便响应切换模式的未授权尝试。另外，该机制可以是合法方法来在安全域和非安全域间切换或可以提供以便提供与现有码的向后兼容性，例如，即使不是真正尝试执行非授权尝试来在安全域和非安全域间切换，作为其正常操作的一部分，试图清除处理状态寄存器。

如上所述，总的来说，当处理器正在监视模式中时，禁止中断。这样做的目的是增加系统的安全性。当中断发生时，将当时的处理器状态存储在中断异常寄存器中以便在完成中断功能时，在中断点能恢复中断函数的处理。如果在监视模式中允许该过程，则会减低监视模式的安全性，提供可能的安全数据泄漏路径。为此，在监视模式中，通常禁止中断。然而，在监视模式期间，禁止中断的一个后果是增加中断等待时间。

如果不存储执行该函数的处理器状态，则可以允许在监视模式中中断。这仅仅在中断之后不恢复该函数的情况下才能执行。因此，通过允许仅中断监视模式中的能安全重启的函数，可以解决监视模式中的中断等待时间的问题。在这种情况下，在监视模式中的中断后，涉及函数处理的数据不进行存储，而是将其丢弃，并在中断结束之后向处理器发出指令，从头对该函数进行处理。在上述例子中，这是当处理器简单地返回到切换到监视模式时的点时处理的简单的事件。应注意，对能重启的某些函数来说，只是对部分特定的函数来说是可能重启并产生可重复产生的结果的。如果函数已经改变处理器的状态以致如果重启它，将产生不同结果时，那么重启函数不是好的主意。为此，在监视模式中，仅中断能安全重启的那些函数，对于其他函数，则禁止中断。

图 18 示例说明根据本发明的实施例，处理在监视模式中出现的中断的方法。在非安全模式中处理任务 A 期间出现 SMI 以及这将处理器切换到监视模式。SMI 指令使内核通过专用的非安全 SMI 矢量进入监视模式。保存 PC 的当前状态，设置 s 位以及禁止中断。通常，使用 LR_mon 和 SPSR_mon 来保存非安全模式的 PC 和 CPSR。

然后在监视模式中启动函数，函数 C。函数 C 所做的第一件事是允许中断，然后处理函数 C。如果在处理函数 C 期间出现中断，不禁止中断以便接受和执行中断。然而，监视模式指示器指示处理器在该

中断后，不恢复该函数，而是重启。另外，这也可以通过单独的控制参数来指示处理器。因此，在中断后，用 LR_mon 和 SPSR_mon 的值来更新中断异常矢量以及不保存处理器的当前状态。

如图 18 所示，在完成中断任务，任务 B 后，处理器读取已经拷贝到中断寄存器的 SMI 指令的地址以及执行 SMI 和开始再次处理函数 C。

仅仅在函数 C 是可重的情况下，上述过程才能执行，即，如果重启，过程 C 将导致可重复的处理步骤。如果函数 C 改变处理器的状态的任何一个，诸如会影响其未来处理的栈指针的情形，情况就完全不一样了。用这种方式可重复的函数被认为具有幂等性。处理不具有幂等性的函数的问题的一种方法是重新排列定义函数的代码，用这种方法，代码的第一部分具有幂等性，以及只要不同可能排列码来具有幂等性，禁止中断。例如，如果代码 C 涉及写入栈，可以执行该操作，而至少不首先更新栈指针。只要确定不再可行地安全重启代码，那么用于函数 C 的代码能指示处理器禁止中断，然后它能将栈指针更新到正确的位置。这如图 18 所示，其中通过函数 C 的处理，用某种方式禁止中断。

图 19 示例说明稍微不同的例子。在该例子中，通过处理任务 C 的某些方法，设置另外的控制参数。这表明任务 C 的后续部分不是严格的幂等性，但假定首先运行修复例程，能安全地重启。该修复例程用来使处理器的状态恢复到在开始任务 C 时的情形，以便能安全地重启任务 C 以及在任务结束时，产生与没有被中断时相同的处理器状态。在一些实施例中，在设置另外的控制参数的点，可以短时间禁止中断，同时改正处理器的一些状态，诸如正更新的栈指针。这允许处理器稍后恢复到幂等性状态。

当在已经设置另外的控制参数后，出现中断时，那么有两种可能的方法来进行。要么立即执行修复例程（在 F1），然后能处理中断，或立即处理中断，以及在中断完全后，执行 SMI，然后在重启任务 C 前，执行修复例程（在 F2）。如所看到的，在这些实施例中，在监视模式中执行修复例程，因此，不影响非安全域中的执行，非安全域不知道安全域或监视模式。

如从图 19 所看到的，代码 C 的第一部分具有幂等性以及能在中断后重启。假定首先运行修复例程，可重启第二部分，这通过设置“另

外”控制参数来表明，以及不能重启代码的最后部分，因此，在处理该代码前，禁止中断。

图 20 示例说明另外的例子，在这种情况下，不同于其他实施例，在监视模式期间允许中断。然后，在监视模式中运行的函数，只要不再能被安全地重启，则禁止中断。如果重启在监视模式中中断的所有函数而不是恢复，这是可能的。

能几种方法来确保当中断时，重启而不是恢复在某一模式中运行的所有函数。一种方法是通过增加新处理器状态，其中中断保存指令序列的开始地址而不是中断指令的地址。在这种情况下，监视模式总是运行在该状态中。另一种方法是通过在开始每个函数时，将函数的起始地址预载到中断异常寄存器，在中断后，禁止处理器的状态的顺序写入以便中断异常寄存器。

在图 20 所示的实施例中，在终止中断函数后，立即执行函数的重启，或如果要求使函数安全重启，在修复例程后执行。

尽管已经根据具有安全和非安全域的系统以及监视模式描述了处理中断等待时间的方法，很显然，能应用于具有由于特定原因，不应当恢复的函数的任何系统。通常，这些函数通过禁止增加中断等待时间的中断来操作。将函数修改成可重启以及控制处理器在中断后重启它们允许对函数的处理的至少一部分，允许中断以及有助于降低中断等待时间。例如，操作系统的标准上下文切换。

访问安全和非安全存储器

如参考图 1 所述，数据处理装置具有存储器，尤其包括 TCM36、高速缓存 38、ROM44、从设备的存储器和外部存储器 56。如参考图 37 所示，例如，将存储器分区成安全和非安全存储器。将意识到在制作时，在存储器的安全存储器区和非安全存储器区间通常没有任何实质区别，但是当在安全域中操作时，这些区由数据处理装置的安全操作系统来定义。因此，可以将存储器设备的任何物理部分分配成安全存储器，以及可以将任何物理部分分配成非安全存储器。

如参考图 2 至 5 所述，处理系统具有安全域和非安全域。在安全域中，提供安全核心程序 80 以及在安全模式中执行。提供横跨于安全域和非安全域并至少部分在监视模式中执行的监视程序 72。在本发明的实施例中，监视程序部分在监视模式中执行以及部分在安全模式中

执行。如例如图 10 所示，有多个安全模式，尤其包括监督模式 SVC。

监视程序 72 负责管理在任一方向中，安全和非安全域间的所有改变。在节“处理器模式”中，参考图 8 和 9 描述其一些函数。监视程序响应在非安全模式中发出的模式转换请求 SMI 以便启动从所述非安全模式到所述安全模式的转换以及响应在安全模式中发出的模式切换请求 SMI，以便启动从所述安全模式到所述非安全模式的转换。如在节“区域间的切换”中所述，在监视模式中，切换发生至少一些寄存器从安全和非安全域的一个切换到另一个。涉及存在于一个域中的寄存器的状态的保存以及将新状态写入另一个域中的寄存器（或恢复寄存器中先前保存的状态）。如在此所述，当执行这种切换时，可以禁止访问一些寄存器。最好，在监视模式中，禁止所述中断。

因为监视程序执行的监视模式横跨于安全和非安全域，证明监视程序安全很重要：即，其仅实现那些意图实现的功能。因此，如果监视程序尽可能简单是有利的。安全模式允许仅在安全域中执行过程。在本发明的该实施例中，特许安全模式和监视模式允许访问相同的安全和非安全存储器。通过确保特许安全模式“看见”相同的安全和非安全存储器，将仅在监视模式中实现的函数传送到允许简化监视程序的安全模式。另外，这允许在特许安全模式中操作的过程直接切换到监视模式或反之亦然。允许从特许安全模式切换到监视模式以及在监视模式中，可以执行到非安全域的切换。非特许安全模式必须使用 SMI 来进入监视模式。该系统在复位后进入特许安全模式。执行监视模式和特许安全模式间的切换和返回以便于当在域间移动时的状态保存。

在其他实施例中，可以从安全特许模式以及监视模式内允许访问 S 标记。如果允许安全特许模式使处理器切换到监视模式中同时维持程序流的控制，那么，这些安全特许模式已经有效地具有改变 S 标记（位）的能力。因此，提供仅能在监视模式内改变 S 标志的另外的复杂度不合理。相反，S 标记能用与通过一个或多个安全特许模式改变的其他配置标记相同的方式存储。可以在多个安全特许模式的一个内改变 S 标记的这些实施例包括在当前技术中。

返回到先前描述的示例性实施例，该装置具有处理器内核 10，定义模式和定义模式的特许级，即，任何模式允许的函数集。因此，用公知的方式配置处理器内核 10 以便允许安全模式和监视模式访问安全

和非安全存储器和安全模式访问监视模式允许访问的所有存储器以及允许在特许安全模式中操作的过程直接切换到监视模式或反之亦然。处理器内核 10 最好配置成允许下述内容。

在该装置的一个例子中，将存储器分区成安全存储器和非安全存储器，以及仅在监视和安全模式中可访问安全和非安全存储器。最好，在监视模式、安全模式和非安全模式中可访问非安全存储器。

在该装置的另一例子中，在监视模式和安全模式的一个或多个中，安全模式拒绝访问非安全存储器，以及在非安全模式中，安全和监视模式拒绝访问非安全存储器。因此，仅在监视和安全模式中访问安全存储器，以及仅通过非安全模式访问非安全存储器，增加安全性。

在该装置的例子中，装置的复位和引导可以在监视模式中执行，该监视模式可以被认为是安全模式更有特权的模式。然而，在该装置的许多例子中，配置成在安全模式中提供重新设置或引导，这是可能的，因为允许安全模式和监视模式间的直接切换。

如参考图 2 所述，在安全域中，以及在安全模式中，安全内核 80（苛操作系统）函数，以及一个或多个安全应用程序 82、84 可以在安全内核 80 下运行。允许在安全模式中运行的安全核心和/或安全应用程序或任何其他程序代码访问安全和非安全存储器。

尽管已经参考具有处理器的装置描述了本发明的例子，本发明可以通过计算机程序来实现，当在适当的处理器上运行时，使处理器操作，如本节中所述。

下面，将根据图 21 至 23，描述从程序员的模型所考虑的本发明的另外的实施例如下：

在下述描述，使用在由 Cambridge, England 的 ARM Limited 设计的 ARM 处理器的情况下，必须理解的下述术语。

- S 位：安全状态位，包括在专用 CP15 寄存器中。
- “安全/非安全状态”。通过 S 位值来定义该状态。表示内核可以访问安全区域（当其处于安全状态，即 S=1 时）还是仅限制到非安全区域（S=0）。注意监视模式（另外见）覆盖 S 位状态。
- “非安全区域”集合可访问不需要安全性的非安全应用程序的所有硬件/软件。
- “安全区域”集合仅当执行安全代码时可访问的所有硬件/软

件（内核、存储器…）。

- 监视模式：负责在安全和非安全状态间切换内核的新模式。

概述

- 内核总是能访问非安全区域。
- 仅当处于安全状态或监视模式时，内核才能访问安全区域。
- SMI：软件监视中断：通过专用 SMI 异常矢量，使内核进入监视模式的新指令。“线程 ID”：与每个线程（受 OS 控制）有关的标识符。对一些类型的 OS，其中，OS 在非安全区域中运行，每次调用安全函数时，有必要将当前线程 ID 传递为参数以便将安全函数链接到其调用的非安全应用程序上。因此，安全区域能支持多线程。

- 安全中断定义由安全外设生成的中断。

程序员的模型

Carbon 内核概述

在此用于使用本技术的处理器的术语的 Carbon 的原理，体系结构包括单独的两个区域，一个安全区域和一个非安全区域。安全区域一定不能将任何数据泄漏到非安全区域。

在所提出的解决方案中，安全和非安全状态将共享相同的（现有）寄存器组。因此，存在于 ARM 内核中的所有当前模式（中止，未定义，Irq，用户，…）将存在于每个状态中。

由于在专用 CP15 寄存器中所示的新的状态位，S（安全）位，内核将知道它在安全或非安全状态中操作。

允许指令或事件来修改 S 位，即从一个状态改变成另一个状态的控制是该系统的安全性的主要特征。当前技术方案提出增加将“监督”两个状态间的切换的新模式，监视模式。通过写入适当的 CP15 寄存器，监视模式将是允许修改 S 位的唯一模式。

最后，建议将一些灵活性增加到异常处理。除复位外，所有异常将在它们出现的状态中处理，或指向监视模式。由于专用 CP15 寄存器，这将允许可配置。

在下述段落中，将论述该解决方案的详细情况。

处理器状态和模式

Carbon 新特征

安全或非安全状态（S 位）

Carbon 内核的一个主要特征是存在 S 位，表明内核处于安全 (S=1) 还是非安全 (S=0) 状态。当在安全状态中，内核将能访问安全或非安全区域中的任何数据。当在非安全状态中，内核将仅限于访问非安全区域。

该规则的唯一异常涉及监视模式，覆盖 S 位信息。即使当 S=0 时，当其位于监视模式中时，内核将执行安全特许访问。见下一段落，用于另外的信息的监视模式。

仅能在监视模式中，读取和写入 S 位。不管 S 值如何，如果任何其他模式尝试访问它，将被忽略或导致未定义异常。

除复位外，所有异常对安全状态位无影响。在复位时，将设置 S 位，以及内核将在监视模式中启动。参见用于详细信息的引导节。

安全/非安全状态是独立的，并且其操作独立于与 ARM/Thumb/Java 状态。

监视模式

Carbon 系统的一个其他重要特征是创建新模式，监视模式。这将用来控制安全和非安全状态间的内核切换。将总被视为安全模式，即不管 S 位的值如何，当它处于监视模式中时，内核将总是对外部区域执行安全特许访问。

仅仅通过写入 CPSR 模式位 (MSR, MOVS 或等效指令)，任何安全特许模式 (即当 S=1 时的特许模式) 将能切换到监视模式。然而，这在非安全模式或安全用户模式将被禁止。如果这经常发生，将忽略指令或导致异常。

将需要专用的 CPSR 冲突异常。通过直接从任何非安全模式或安全用户模式写入 CPSR，切换到监视模式的任何尝试，将产生该异常。

当监视模式有效时，实际上，将禁止除复位外的所有异常：

- 屏蔽所有中断
- 所有存储器异常要么可以忽略或者导致致命异常
- 未定义/SWI/SMI 要么可以忽略或者导致致命异常。

当进入监视模式时，自动禁止中断以及应当写入系统监视程序以便在系统监视程序正运行时，不发生任何其他类型的异常。

监视程序模式需要具有一些专用寄存器。该解决方案建议仅复制

最小寄存器集，即 R13 (sp_mon)、R14 (lr_mon) 和 SPSR (spsr_mon)。

在监视模式中，禁止 MMU (单调地址映射) 和 MPU 或分区校验器 (监视程序模式将总是执行安全特许外部访问)。然而，专用编程 MPU 区属性 (高速缓存能力，...) 将仍然有效。作为另一方案，不管安全域使用何种映射，监视模式可以使用。

新指令

该提议要求将一个新指令增加到现有的 ARM 指令集上。

将使用 SMI (软件监视中断) 来进入监视模式，在固定 SMI 异常矢量转移。该指令将主要用来向监视程序表示在非安全和安全状态间交换。

作为另一方案 (或另外)，将可以增加新指令以便允许监视模式将任何其他模式保存到监视栈/从监视栈恢复任何其他模式以便提高上下文切换性能。

处理器模式

如在前段落中所述，在内核中仅增加一个新模式，监视模式。所有现有的模式仍然可用，以及将存在于安全和非安全状态中。

实际上，Carbon 用户将看见如图 21 所示的结构。

处理器寄存器

本实施例提出安全和非安全区域共享相同的寄存器组。这意味着当通过监视模式，从一个区域切换到另一个区域时，系统监视程序将需要保存第一区域的上下文，以及在第二区域中创建 (或恢复) 上下文。

传递参数变为一项简单的任务：只要系统监视程序已经切换 S 位，包含在第一区域中的寄存器中的任何数据将用在第二区域中的相同寄存器中。

然而，除了需要严格控制的专用于传递参数的有限多个寄存器之外，当从安全传递到非安全状态时，所有其他寄存器需要被清除以便避免安全区域的任何泄漏。

当从安全切换到非安全状态时，实现硬件机制或新指令来直接刷新寄存器也是可能的。

所提出的另一解决方案涉及复制所有 (或大多数) 现有的寄存器组，从而在安全和非安全状态间具有两个物理分开的寄存器。该解决

方案具有明显地分开包含在寄存器中的安全和非安全数据的主要优点。还允许安全和非安全状态间的快速上下文切换。然而，缺点在于通过寄存器传递参数变得困难，除非创建一些专用指令来允许安全区域访问非安全寄存器。

图 22 示例说明根据处理器模式的可用寄存器。注意，处理器状态对该主题无影响。

异常

安全中断

当前解决方案

目前提出保持与当前内核中相同的中断管脚，即 IRQ 和 FIQ。与异常中断屏蔽寄存器（在本文献中稍后定义）有关，对任何系统来说，有足够的灵活性来实现和处理不同类型的中断。

VIC 增强

用下述方法增加 VIC（矢量中断控制器）：VIC 可以包含与每个矢量地址有关的一个安全信息位。仅可以通过监视或安全特许模式编程该位。表示所认为的中断是否应当被视为安全，从而应当在安全端处理。

还增加两个新的矢量地址寄存器，一个用于在非安全状态中出现的所有安全中断，另一个用于在安全状态中出现的所有非安全中断。

包含在 CP15 中的 S 位信息将可用于 VIC，作为新 VIC 输入。

根据输入中断的状态（安全或非安全，用与每个中断线有关的位表示）和内核的状态（CP15 中的 S 位=VIC 上的 S 输入信号），下述表概述不同可能的情形。

	安全状态中的内核 (CP15-S=1)	非安全状态中的内核 (CP15-S=0)
安全中断	不需要在区域间切换。VIC 直接向内核提供与中断线有关的安全地址。内核仅仅需要在该地址分支，其中应当查找相关的 ISR	在非安全域中，VIC 不具有与该中断有关的矢量。因此，向内核提供包含在专用于在非安全区域中出现的所有安全中断的矢量地址寄存器中的地址。仍然在非安全区域中，然后，内核分支到该地

		址，其中应当查找 SMI 指令来切换到安全区域。只要在安全区域中，将能访问正确的 ISR。
非安全中断	在安全域中，VIC 不具有与该中断有关的矢量。因此，向内核提供包含在专用于在安全区域中的所有非安全中断的矢量地址寄存器中的地址。仍然在安全区域中，然后内核分支到该地址，其中，应当查找 SMI 指令来切换到非安全区域。只要在非安全区域中，将能访问正确的 ISR	不需要在区域间切换。VIC 直接向内核提供与中断线有关的不安全地址。内核仅仅需要在该地址分支，其中应当查找相关的不安全 ISR

异常处理可配置性。

为提高 Carbon 灵活性，新寄存器，异常中断屏蔽将增加到 CP15 中。该寄存器将包含下述位：

- 位 0: 未定义异常 (非安全状态)
- 位 1: SWI 异常 (非安全状态)
- 位 2: 预取中止异常 (非安全状态)
- 位 3: 数据中止异常 (非安全状态)
- 位 4: IRQ 异常 (非安全状态)
- 位 5: FIQ 异常 (非安全状态)
- 位 6: SMI 异常 (非安全/安全状态)
- 位 16: 未定义异常 (安全状态)
- 位 17: SWI 异常 (安全状态)
- 位 18: 预取中止异常 (安全状态)
- 位 19: 数据中止异常 (安全状态)
- 位 20: IRQ 异常 (安全状态)
- 位 21: FIQ 异常 (安全状态)

复位异常在该寄存器中不具有任何相应位。复位将总是使内核通过其专用矢量进入安全监督模式。

如果设置位，相应的异常使内核进入监视模式。否则，将在其出现的区域中的相应的处理程序中处理该异常。

该寄存器将仅在监视模式中可见。将忽略在任何其他模式中尝试访问它的任何指令。

根据系统是否监视程序，应当将该寄存器初始化到系统专用值。能通过 VIC 控制该功能性。

异常矢量表

当有单独的安全和非安全区域时，还需要单独的安全和非安全异常矢量表。

此外，当监视程序还能设一些异常中断时，还需要专用于该监视程序的第三异常矢量表。

下述表概述那些三个不同异常矢量表：

在非安全存储器中：

地址	异常	模式	何时自动访问
0x00			
0x04	未定义	未定义	当内核在非安全状态和异常中断屏蔽寄存器中 [非安全未定义] = 0 时执行的未定义指令
0x08	SWI	监督	当内核在非安全状态和异常中断屏蔽寄存器中 [非安全 SWI] = 0 时的执行 SWI 指令
0x0C	预 取 中 止	中止	当内核在非安全状态和异常中断屏蔽寄存器中时 [非安全 PAbort=0 时，中止指令
0x10	数 据 中 止	中止	当内核在非安全状态和异常中断屏蔽寄存器中时 [非安全 DAbort=0 时，中止数据
0x14	预 留		
0x18	IRQ	IRQ	当内核在非安全状态和异常中断屏蔽寄存器中 [非安全 IRQ] = 0 时推断的 IRQ 管脚
0x1C	FIQ	FIQ	当内核在非安全状态和异常中断屏蔽寄存器中 [非安全 FIQ] = 0 时推断的 FIQ 管脚

在安全存储器中：

地址	异常	模式	何时自动访问
0x00	复位*	监督	推断的复位管脚
0x04	未定义	未定义	当内核在安全状态和异常中断屏蔽寄存器中 [安全未定义] = 0 时执行的未定义指令
0x08	SWI	监督	当内核在安全状态和异常中断屏蔽寄存器中 [安全 SWI] = 0 时的执行 SWI 指令
0x0C	预 取 中 止	中止	当内核在安全状态和异常中断屏蔽寄存器中 时 [安全 PAabort = 0 时, 中止指令
0x10	数 据 中 止	中止	当内核在安全状态和异常中断屏蔽寄存器中 时 [安全 DAabort = 0 时, 中止数据
0x14	预留		
0x18	IRQ	IRQ	当内核在安全状态和异常中断屏蔽寄存器中 [非安全 IRQ] = 0 时推断的 IRQ 管脚
0x1C	FIQ	FIQ	当内核在安全状态和异常中断屏蔽寄存器中 [非安全 FIQ] = 0 时推断的 FIQ 管脚

*参见进一步描述有关复位机制的“引导”节。

在监视存储器中（单调映射）

地址	异常	模式	何时自动访问
0x00			
0x04	未定义	监视	当内核在安全状态和异常中断屏蔽寄存器中 [安全未定义] = 1 以及内核在非安全状态 和异常中断屏蔽寄存器 [非安全未定义] = 1 时执行的未定义指令
0x08	SWI	监视	当内核在安全状态和异常中断屏蔽寄存器中 [安全 SWI] = 1 以及内核在非安全状态和 异常中断屏蔽寄存器 [非安全 SWI] = 1 时 执行的 SWI 指令
0x0C	预 取 中	监视	当内核在安全状态和异常中断屏蔽寄存器中

	止		[安全 IAbort] =1 以及内核在非安全状态和异常中断屏蔽寄存器 [非安全 IAbort] =1 时，中止指令
0x10	数 据 中 止	监视	当内核在安全状态和异常中断屏蔽寄存器中 [安全 PAbort] =1 以及内核在非安全状态和异常中断屏蔽寄存器 [非安全 PAbort] =1 时，中止数据
0x14	SMI	监视	
0x18	IRQ	监视	当内核在安全状态和异常中断屏蔽寄存器中 [安全 IRQ] =1 以及内核在非安全状态和异常中断屏蔽寄存器 [非安全 IRQ] =1 时推断的 IRQ 管脚
0x1C	FIQ	监视	当内核在安全状态和异常中断屏蔽寄存器中 [安全 FIQ] =1 以及内核在非安全状态和异常中断屏蔽寄存器 [非安全 FIQ] =1 时推断的 FIQ 管脚

在监视模式中，可以复制异常矢量，以便每个异常将具有两个不同的相关矢量：

- 一个用于在非安全状态中出现的异常
- 一个用于在安全状态中出现的异常

这对于降低异常等待时间是有用的，因为监视核心不再具有检测异常出现的原始状态的必要。

注意该特征可限于一些异常，SMI 是改进安全和非安全状态间的切换的最适当候选。

区域间的切换

当状态间切换时，监视模式必须将第一状态的上下文保存在其监视栈上，以及从监视栈恢复第二状态上下文。

因此，监视模式需要访问任何其他模式的任何寄存器，包括专用寄存器 (r14, SPSR, ...)。

为处理此，所提出的解决方案包括通过简单地写入 CPSR，向安全状态中的任何特许模式提供直接切换到监视模式的权利。

通过该系统，区域间的切换执行如下：

- 进入监视模式

- 设置 S 位

- 切换到监督模式-将监视寄存器保存在监视栈上（当然监督模式将需要访问监视栈指针，但这能容易实现，例如通过使用共用寄存器（R0 至 R8））

- 切换到系统模式-将寄存器（=与用户模式相同）保存在监视栈上

- 监视栈上的 IRQ 寄存器

- 等等… 用于所有模式

- 一旦保存所有模式的所有专用寄存器，通过简单的 MSR 指令（=简单地将监视值写在 CPSR 模式字段中），回复到监视模式

也考虑到其他解决方案：

- 增加允许监视程序将其他模式的专用寄存器保存在其自己的栈上的新指令。

- 将监视程序实现为新“状态”，即，能处于监视状态（具有适当的访问权）和 IRQ（或任何其他模式），以便查看 IRQ（或任何其他）专用寄存器。

基本情形（见图 23）

1. 线程 1 运行在非安全区域中（S 位=0）

2. 通过非安全 SMI 矢量，SMI 指令使内核进入监视模式。

使用 LR_mon 和 SPSR_mon 来保存非安全模式的 PC 和 CPSR。

在该阶段，S 位仍然不变，尽管该系统现在在安全状态中。

监视核心将非安全上下文保存在监视程序上。

还压入 LR_mon 和 SPSR_mon。

然后监视内核通过写入 CP15 寄存器来改变“S”位。

在该实施例中，监视内核记录将在安全区域中（例如通过更新线程 ID 表），启动“安全线程 1”。

最后，它退出监视模式以及切换到安全监督模式（在更新 LR_mon 和 SPSR_mon 后的 MOVS 指令？）。

3. 安全核心将应用程序调度到正确的安全存储单元，然后切换到用户模式（例如使用 MOVS）。

4. 在安全用户模式中执行安全函数。一旦完成，通过执行适当的 SWI，调用“退出”函数。

5. 通过反过来执行“退出”函数的专用 SWI 矢量，SWI 指令使内核进入安全 svc 模式。该“退出”函数以“SMI”结束以便切换回监视模式。

6. SMI 指令使内核通过专用的安全 SMI 矢量进入监视模式。

使用 LR_mon 和 SPSR_mon 来保存安全 svc 模式的 PC 和 CPSR。

S 位仍然不变（即安全状态）

监视核心记录结束安全线程 1 的事实（从线程 ID 表移出安全线程 1 ID? ）

然后通过写入 CP15 寄存器，返回到非安全状态来改变“S”位。

监视核心从监视栈恢复非安全上下文。

在步骤 2 中，还加载先前保存的 LR_mon 和 SPSR_mon。

最后，根据指令，它通过 SUBS 退出监视模式，使内核返回到非安全用户模式中。

7. 线程 1 能正常恢复。

参考图 6，在安全和非安全域间共享所有寄存器。在监视模式中，切换发生，寄存器从安全和非安全域的一个切换到另一个。包含对存在于一个域中的寄存器的状态进行保存以及将新状态写入另一个域中的寄存器（或恢复寄存器中的先前保存的状），如在上节“区域间的切换”中所述。

期望降低执行这一切换所需的时间。为降低执行切换所花的时间，当安全和非安全域间的切换保持不改变存储在其中的值时，禁止共享寄存器。例如，假定从非安全域切换到安全域。假定例如在安全区域中不需要图 6 所示的 FIQ 寄存器。因此，禁止那些寄存器以及不需要将它们切换到安全域以及不需要保存那些寄存器的内容。

可以采用多种方法来实现禁止寄存器。一种方法是封锁使用那些寄存器的模式。这是通过将控制位写入 CP15 寄存器中，表示禁止那个模式来实现的。

另外，在逐个指令的基础上，通过将控制位写入 CP15 寄存器，可以禁止访问寄存器。写入 CP15 中的位明确与该寄存器，而不是模式有关，以便不禁止工，而是禁止访问该模式中的寄存器。

FIQ 寄存器存储与快速中断有关的数据。如果禁止 FIQ 寄存器以及快速中断发生，处理器发出监视程序中的异常的信号。响应异常，监视模式能用来保存与一个域有关并存储在所述禁止寄存器中的任何数据值以及将与另一域有关的新数据值加载到那个寄存器中，然后重新允许 FIQ 模式寄存器。

可以将处理器配置成当在监视模式中时，当处理器切换域时，禁止所有成组寄存器。另外，禁止寄存器是可选择的，因为当切换域时，禁止一些预定的共享寄存器，以及根据程序员的选择，可以禁止其他寄存器。

可以将处理器配置成当在监视模式中切换域时，禁止一个或多个的共享寄存器，以及当存在一个域时，一个或多个其他共享寄存器保存它们的数据，以及将新数据加载在另一个域中。新数据可以是空数据。

图 24 示意性地示例说明将安全处理选项增加到传统的 ARM 内核的原理。该图示意性地表示如何通过将安全处理选项增加到现有的内核，能形成包含安全处理选项的处理器。如果该系统将向后与现有的传统操作系统兼容，直觉想到在处理器的传统非安全部分中操作的传统系统。然而，如该图的下半部分中示意所示以及下面的进一步详述，实际上，传统系统在该系统的安全部分中操作。

图 25 表示具有安全和非安全域以及示例说明复位并与图 2 类似的处理器。图 2 示例说明用来通过控制安全域中的处理的安全 OS 系统和控制非安全域中的处理的非安全 OS 系统，运行安全性敏感型操作的处理器。然而，该处理器与传统的操作系统向后兼容，因此，该处理器使用传统的操作系统，以安全性不敏感方式操作。

如图 25 所示，复位在安全域中，以及不管操作类型如何，通过所设置的 S 位或安全性状态标记，发生复位。在安全性不敏感的操作的情况下，在安全域中发生复位，然后处理在安全域内继续。然而，控制处理的传统操作系统不知道系统的安全性方面。

如图 25 所示，执行复位来设置地址，不管处理是安全敏感还是实际上安全不敏感，从该地址启动安全监督模式内的处理。一旦执行复位，那么执行存在于引导或重新引导机制中的另外的任务。下面描述引导机制。

引导机制

引导机制必须考虑下述特征：

- 保持与传统 OSes 的兼容性
- 在最特许模式中引导以便确保系统的安全性。

因此，Carbon 内核将在安全监督模式中引导。

然后不同系统将是：

- 对希望运行传统 OS 的系统，不考虑 S 位，以及内核将正好看其在监督模式中引导。
- 对希望使用 Carbon 特征的系统，内核在应当能在系统中配置所有安全保护（可以在切换到监视模式后）的安全特许模式中引导。

根据如上给出的引导机制的详细情况，本发明的实施例的处理器复位该处理器以便在所有情况下，在安全监督模式中开始处理。在安全性不敏感的操作的情况下，操作系统实际上在安全域中操作，尽管安全性在此不是问题，因为设置了 S 位（尽管操作系统不知道此）。其优点在于不能从非安全域访问的部分存储器在这种情况下是可以访问的。

在安全敏感系统中，在所有情况下，在安全监督模式中引导也有利，因为有助于确保系统的安全性。在安全敏感系统中，在安全监督模式中，在存储引导程序的引导点处提供地址，从而允许系统配置成安全系统以及切换到监视模式。通常允许从安全监督模式切换到监督模式以及在适当的时间，允许安全系统在监督模式中开始处理以便初始化监视模式配置。

图 26 示例说明在步骤 1，由非安全操作系统执行非安全线程 NSA。在步骤 2，非安全线程 NSA 经在步骤 3，运行监视模式程序的监视模式，调用安全域。在步骤 5，监视模式程序改变 S 位以便切换域以及在移动到安全操作系统前，执行任何所需的上下文保存和上下文恢复。然后，在步骤 6，在经受中断 irq 前，执行相应的安全线程 SA。在步骤 7，中断处理硬件触发返回到监视模式，其中确定有关中断将由安全操作系统还是非安全操作系统处理。在这种情况下，将由在步骤 9 开始的非安全操作系统处理中断。当由非安全操作系统处理该中断时，在步骤 11，在正常线程切换操作前，非安全线程 NSA 恢复成非安全操作系统中的当前任务。该线程切换可以是定时事件等待的结果。在步骤

12, 通过非安全操作系统中的非安全域, 执行不同线程 NSB, 然后在步骤 14, 经监视域/程序, 调用安全域。步骤 7 的监视程序已经存储标记, 在一些其他机制中, 用来表示根据中断的结果, 最后暂停安全操作系统, 而不是留下, 因为安全线程已经结束执行或由于正常请求而留下。因此, 由于通过中断暂停安全操作系统, 步骤 15 的监视程序使用指定返回线程 ID (例如, 根据非安全线程 NSB 的请求, 由安全操作系统启动的线程的标识符以及其他参数数据) 的软件伪中断, 重新进入安全操作系统。软件伪中断的这些参数可以作为寄存器值来传递。

在步骤 15, 软件伪中断触发安全操作系统的返回中断处理程序例程。该返回中断处理程序例程检查软件伪中断的返回线程 ID 以便确定这是否与在暂停前, 最后执行安全操作系统时中断的安全线程 SA 的线程 ID 匹配。在这种情况下, 没有匹配, 因此, 在步骤 16, 触发安全操作系统以便在已经保存安全线程 SA 的上下文后, 根据非安全线程 NSB 所指定的, 执行切换到返回线程的线程。然后, 根据请求, 从中断的点处重启安全线程 SA。

图 27 示意性地示例说明在图 26 中所示的行为的类型的另一例子。在该例子中, 尽管在非安全操作系统的控制下, 处理执行以便处理 irq, 但没有非安全切换, 因此当安全操作系统的返回中断处理程序接收到软件伪中断时, 确定不需要线程切换, 在步骤 15, 仅仅恢复安全线程 SA。

图 28 是示意性地示例说明由返回线程处理程序执行的处理的流程图。在步骤 4002, 开始返回线程处理程序。在步骤 4004, 当暂停安全操作系统时, 检查来自软件伪中断的返回线程标识符以及与当前执行的安全线程进行比较。如果这些匹配, 那么, 处理进入步骤 4006, 其中恢复安全线程。如果在步骤 4004, 比较得出不匹配, 那么处理进入步骤 4008, 其中, 在步骤 4010 执行切换到新安全线程前, 保存先前的安全线程的上下文 (用于后续恢复)。新线程可能已经进行, 因此步骤 4010 是恢复。

图 29 示意性地示例说明从安全操作系统遵循由主非安全操作系统执行的任务切换的处理。主非安全操作系统可以是传统的操作系统, 不具有用于与其他操作系统通信和协调其活动性的机制, 因此, 仅操作作为主程序。作为图 29 中的起始输入点, 非安全操作系统正在执行非

安全线程 NSA。使用软件中断，该非安全线程 NSA 调用将由安全操作系统执行的安全线程，SMI 调用。在步骤 2，SMI 调用进入在监视模式中执行的监视程序，由此，在步骤 4，监视程序在将调用传递到安全操作系统前，执行任何必要的上下文保存和切换。然后安全操作系统启动相应的安全线程 SA。诸如根据定时器事件等等的结果，该安全线程可以经监视模式，将控制返回到非安全操作系统。当在步骤 9，非安全线程 NSA 再次将控制传递给安全操作系统时，通过重新发出原始软件中断执行此。软件中断包括识别 NSA 的非安全线程 ID、将激活的目标安全线程的安全线程 ID，即识别安全线程 SA 的线程 ID 以及其他参数。

当通过监视程序传递在步骤 9 生成的调用并通过安全操作系统，在安全域中，在步骤 12 接收时，检查非安全线程 ID 以便通过非安全操作系统，确定是否具有上下文切换。也可以检查目标线程的安全线程 ID 以便查看安全操作系统下的正确线程重启或启动为新线程。在图 29 的例子中，通过安全操作系统，在安全域中不要求线程切换。

除在非安全操作系统的控制下，在非安全域中，在步骤 9 产生线程的切换外，图 30 与图 29 类似。因此，在步骤 11，它是使软件中断调用经过安全操作系统的不同的非安全线程 NSB。在步骤 14，安全操作系统识别非安全线程 NSB 的不同线程 ID，因此，执行包含保存安全线程 SA 的上下文以及开始安全线程 SB 的任务切换。

图 31 是示意性地示例说明当将软件中断接收为启动线程或恢复安全操作系统的线程的调用时，由安全操作系统执行的处理的流程图。在步骤 4012，接收调用。在步骤 4014，检查调用的参数以便确定它们是否与安全操作系统上的当前有效安全线程匹配。如果匹配发生，那么在步骤 4016，重启该安全线程。如果匹配未发生，那么处理进入步骤 4018，其中确定有关新请求的线程是否可用。由于诸如它是或要求已经由正在安全操作系统上执行的一些其他线程使用的互斥资源的原因，新请求的线程不可用。在这种情况下，通过返回给非安全操作系统的适当的消息，在步骤 4020 拒绝调用。如果在步骤 4018 确定新线程可用，那么处理进入步骤 4022，其中为稍后可能的恢复，保存先前的安全线程的上下文。在步骤 4024，根据在对安全操作系统所做的软件中断调用中所指定的，切换到新安全线程。

图 32 示意性地示例说明当通过由不同操作系统处理的不同中断，处理具有多个操作系统的系统内的中断时，发生优先级反转的操作。

处理从执行安全线程 SA 的安全操作系统开始。然后通过第一中断 Int1 中断。这触发监视模式中的监视程序确定该中断将在安全域还是非安全域中处理。在这种情况下，该中断将在安全域中处理以及处理返回到安全操作系统，启动用于中断 Int1 的中断处理例程。到执行用于 Int1 的中断处理例程的一半时，接收到具有更高优先级的另外的中断 Int2。因此，停止用于 Int1 的中断处理程序同，以及监视模式中的监视程序用来确定将处理中断 Int2。在这种情况下，将通过非安全操作系统处理中断 Int2，因此控制传递到非安全操作系统，以及用于 Int2 的中断处理程序启动。当用于中断 Int2 的中断处理程序结束时，非安全操作系统不具有表示在安全域中，有暂停维护的未决中断 Int2 的信息。因此，非安全操作系统会执行一些另外的处理，诸如任务切换或启动不同的非安全线程 NSB，同时最初中断 Int1 仍然未处理。

图 33 示例说明可以避免与图 32 的操作有关的问题的技术。当中断 Int1 产生时，监视程序将此传递到非安全域，其中启动插桩中断处理程序。该插桩中断处理程序相对较小，以及经由监视模式，快速将处理返回到安全域，以及触发安全域内用于中断 Int1 的中断处理程序。主要在安全域中处理中断 Int1 以及在非安全域中启动插桩中断处理程序能视为一种插桩符，向非安全域表示该中断正被挂在安全域中。

用于中断 Int1 的安全域中的中断处理程序再次遇到高优先级 Int2。如前所述，这触发执行非安全域中，用于中断 Int2 的中断处理程序。然而，在这种情况下，当用于 Int2 的那个中断处理程序结束时，非安全操作系统具有表示用于中断 Int1 的插桩中断处理程序仍然未决的数据，因此，将恢复该插桩中断处理程序。该插桩中断处理程序就象挂在使其调用回安全域的点处，因此重新执行该调用，从而切换到安全域。一旦回到安全域中，安全域本身能在暂停它的点处，重启用于中断 Int1 的中断处理程序。当在安全域内，用于中断 Int1 的中断处理程序结束时，调用返回到非安全域以便在恢复最初执行安全线程 SA 前，关闭非安全域中的插桩中断处理程序。

图 34 示意性地示例说明具有它们相关的优先级和不同类型的中断和如何处理它们。完全使用提供没有由非安全域处理的更高优先级中

断的安全域中断处理程序，处理高优先级中断。只要具有比后续中断更高优先级以及在非安全域中处理的中断，那么所有更低中断必须完全在非安全域中处理或利用图 33 所示的插桩中断处理程序技术，由此非安全域能了解这些中断，即使它们的大部分处理在安全域内发生。

如前所述，使用监视模式来执行安全域和非安全域间的切换。在两个不同域间共享寄存器的实施例中，这包含将那些寄存器内的状态保存在存储器中，然后将用于目标域的新状态从存储器加载到那些寄存器中。对不在两个域间共享的任何寄存器，不必保存状态，因为那些寄存器将不由另一域访问，而状态间的切换实现为安全和非安全域间的直接切换结果（即，存储在 CP15 寄存器的一个中的 S 位的值确定使用哪个非共享寄存器）。

当在监视模式中时需要切换的状态部分为控制由处理器访问存储器的处理器配置数据。由于每个域内，存在存储器的不同视图，例如，访问安全存储器的安全域用于存储安全数据，该安全存储器不能由非安全域访问，很显然，当在域间切换时，需要改变处理器配置数据。

如图 35 所示，该处理器配置数据存储在 CP15 寄存器 34 中，以及在一个实施例中，在域间这些寄存器是共享的。因此，当在安全域和非安全域间切换监视模式时，当前在 CP15 寄存器 34 中的处理器配置数据需要从 CP15 寄存器移入到存储器中，而与目标域有关的处理器配置数据需要加载到 CP15 寄存器 34 中。

由于 CP15 中的处理器配置数据通常对访问系统内的存储器具有迅速影响，因此，很显然当在监视模式中操作时，通过处理器更新它们，这些设定值迅速有效。然而，这不是所期望的，因为期望监视模式具有当在监视模式中时，控制访问存储器的静态处理器配置数据集。

因此，如图 35 所示，在本发明的一个实施例中，提供监视模式专用处理器配置数据 2000，能用来当处理器在监视模式中操作时，覆盖 CP15 寄存器 34 中的处理器配置数据。通过提供在其输入处接收存储在 CP15 寄存器中的处理器配置数据和监视模式专用处理器配置数据 2000 的复用器 2010，在图 35 所示的实施例中实现。此外，复用器 2010 在路径 2015 上接收表明处理器是否正在监视模式中操作的控制信号。如果处理器不在监视模式中操作，那么将 CP15 寄存器 34 中的处理器配置数据输出到系统，但在处理器在监视模式中操作的情况下，复用

器 2010 反而输出监视模式专用处理器配置数据 2000 以确保处理器正在监视模式中操作时，应用相符的处理器配置数据集。

能在系统内对监视模式专用处理器配置数据尽心硬编码，从而确保不能操作它。然而，假定当在安全特许模式中操作时，仅能修改监视模式专用处理器配置数据，则能编程监视模式专用处理器配置数据 2000，而不损害安全性。这提供有关监视模式专用处理器配置数据的设定值的一些灵活性。如果监视模式处理器配置数据配置成可编程，能将配置数据存储在系统内的适当地方，例如 CP15 寄存器 34 内的单独的寄存器集中。

典型地，将设置监视模式专用处理器配置数据以便提供用于在监视模式中操作处理器的非常安全的环境。因此，在上述实施例中，监视模式专用处理器配置数据可以指定当处理器正在监视模式中操作时，禁止存储器配置单元 30，从而禁止可以由存储器管理单元应用的任何虚拟到物理地址转换。在这种情况下，总是将处理器配置成当发出存储器访问请求时，直接发送物理地址，即，将采用单调映射。这确保处理器正在监视模式中操作时，能可靠地访问存储器，而与是否已经篡改任何虚拟到物理地址映射无关。

监视模式专用处理器配置数据通常还指定当处理器正在监视模式中操作时，允许处理器访问安全数据。这最好通过采用域状态位的存储器允许数据指定，该域状态位具有将为安全处理器配置数据内的相应的域状态位（“S”位）指定的相同值。因此，不管存储在 CP15 寄存器内的域状态位的实际值如何，通过由监视模式专用处理器配置数据指定的域状态位覆盖那个值，以确保监视模式访问安全数据。

监视模式专用处理器配置数据也指定用来控制访问部分存储器的其他数据。例如，监视模式专用处理器配置数据可以指定当处理器正在监视模式中操作时，不使用高速缓存 38 来访问数据。

在上述实施例中，已经假定包含处理器配置数据的所有 CP15 寄存器在域间共享。然而，在上述实施例中，“成组”多个 CP15 寄存器，以便例如具有用于存储处理器配置数据的特定项的两个寄存器，一个寄存器可在非安全域中访问以及包含用于非安全域的处理器配置数据的那个项的值，以及另一个寄存器可在安全域中访问以及包含用于安全域的处理器配置数据的那个项的值。

将不成组的一个 CP15 寄存器是包含“S”位，但原理上，如果需要的话，可以对任何其他 CP15 寄存器进行成组。在这些实施例中，通过监视模式切换处理器配置数据包含使当前在那些共享寄存器中的处理器配置数据从任何共享 CP15 寄存器移入存储器，以及将与目标域有关的处理器配置数据加载到那些共享 CP15 寄存器。对任何成组寄存器，处理器配置数据不需要存储在存储器外，相反根据改变存储在相关共享 CP15 寄存器中的 S 位值的结果，将自动发生切换。

如前所述，监视模式处理器配置数据将包括覆盖存储在相关 CP15 寄存器中的域状态位，而具有与用在安全域中的域状态位相同的值（即，在上述实施例中，S 位值为 1）。当多个 CP15 寄存器成组时，这表示能从存储在成组寄存器中的安全处理器配置数据导出图 35 中的监视模式专用处理器配置数据 2000 的至少一部分，因为在切换过程期间，那些寄存器内容不写到存储器外。

因此，例如，由于监视模式专用处理器配置数据将指定域状态位以便覆盖当不在监视模式中使用的域状态位，以及在优选实施例中，这具有与用在安全域中相同的值，这表示选择访问哪个成组 CP15 寄存器的逻辑将允许访问安全成组 CP15 寄存器。通过允许监视模式将该安全处理器配置数据用作监视模式专用处理器配置数据的相关部分，能实现节省资源，因为不再需要提供用于监视模式专用处理器配置数据的那些项的单独的寄存器集。

图 36 是示例说明当要求在一个域和另一个间转变时，切换处理器配置数据所执行的步骤的流程图。如前所述，发出 SMI 指令以便促使域间的转变。因此，在步骤 2020，等待发出 SMI 指令。当接收到 SMI 指令时，处理器进入步骤 2030，其中处理器开始在监视模式中运行监视程序，根据进入多路复用器 2010 中，导致多路复用器切换到监视模式专用处理器配置数据的路径 2015 上的控制信号，引发监视模式专用处理器配置数据的使用。如前所述，这能是独立的数据集，或能从存储在成组寄存器中的安全处理器配置数据导出的某些部分。

此后，在步骤 2040，从将 SMI 指令发出到存储器中的域，保存当前状态，这包括从任何共享 CP15 寄存器保存与那个域有关的处理器配置数据的状态。通常，存在除存储这些状态外设置的一部分存储器。然后，在步骤 2050，该状态指针切换到包含用于目标域的相应状态的

部分存储器。因此，通常，存在为存储状态信息而分配的两个存储器部分，一个分配用于存储用于非安全域的状态，以及一个分配用于存储用于安全域的状态。

一旦在步骤 2050 切换状态指针，在步骤 2060，现在将由状态指针指向的那个状态加载到相关共享 CP15 寄存器，这包括将用于目标域的配置数据加载在相关处理器中。此后，在步骤 2070，退出监视程序，如在监视模式中，然后，处理器切换到目标域中的所需模式。

图 37 更详细地示例说明本发明的一个实施例的存储器管理逻辑 30 的操作。存储器管理逻辑由存储器管理单元 (MMU) 200 和存储器保护单元 (MPU) 220 组成。在路径 234 上，由内核 10 发出、指定虚拟地址的任何存储器访问请求传递到 MMU200，MMU200 负责执行预定访问控制功能，更具体地说，用于确定对应于那个虚拟地址的物理地址，以及用于决定访问许可权和确定区域属性。

数据处理装置的存储器系统由安全存储器和非安全存储器组成，当内核或其他设备正在安全操作模式中操作，相应地，在安全域中操作时，安全存储器用来存储仅用来由内核 10，或一个或多个其他主设备访问的安全数据。

在图 37 所示的本发明的实施例中，通过 MPU220 内的分区校验器 222，执行通过在非安全模式中，在内核 10 上运行的应用程序，访问安全存储器中的安全数据的尝试控制，MPU220 受安全操作系统，在此也称为安全核心管理。

根据本发明的优选实施例，在非安全存储器内，例如外部存储器 56 的非安全存储器部分内提供非安全页表 58，以及用来存储用于在那个页表内定义的多个非安全存储器区的每一个的相应的描述符。

描述符包含信息，MMU200 能从该信息导出允许 MMU 执行预定访问控制功能所需的控制信息，因此，在参考图 37 所述的实施例中，将提供有关虚拟到物理地址映射、访问许可权以及任何区域属性的信息。

此外，根据本发明的优选实施例，在存储器系统的安全存储器内，例如外部存储器 56 的安全部分内提供至少一个安全页表 58，再次提供用于在表内定义的多个存储区的相关描述符。当处理器在非安全模式中操作时，引用非安全页表以便获得用在管理存储器访问中的相关描述符，同时当处理器正在安全模式中操作时，将使用来自安全页表

的描述符。

从 MMU 中的相关页表检索描述符如下。在内核 10 发出的存储器访问请求指定虚拟地址的情况下，在存储用于多个虚拟地址部分的一个的从相关页表获得的相应的物理地址部分的微 TLB206 中，执行查找。因此，微 TLB206 将虚拟地址的某些部分与存储在微 TLB 中的相应的虚拟地址进行比较以便确定是否有匹配。所比较的部分通常是一些预定多个虚拟地址的最高有效位，位数由页表 58 内的页的粒度而定。在微 TLB206 内执行的查找通常相对较快，因为微 TLB206 将仅包括相当少量条目，例如八个条目。

在微 TLB206 中没有发现匹配的情况下，那么在路径 242 上，将存储器访问请求传送到包含从页表获得的多个描述符的主 TLB208。如随后所作更详细的讨论，来自非安全页表和安全页表的描述符能共存于主 TLB208 中，以及主 TLB 内的每个条目具有可设置来表示已经从安全页表还是非安全页表获得那个输入中的相应的描述符的相应的标记（称为主标记）。在所有安全操作模式直接指定它们的存储器访问请求内的物理地址的任何实施例中，将意识到在主 TLB 内不需要这一标记，因为主 TLB 将仅存储非安全描述符。

在主 TLB208 内，执行类似的查找过程以便确定在存储器访问请求内发出的虚拟地址的相关部分是否对应于与操作的特定模式有关、与主 TLB208 中的描述符有关的虚拟地址部分的任何一个。因此，如果内核 10 正在非安全模式中操作，将仅校验已经从非安全页表获得的、主 TLB208 内的那些描述符，而如果内核 10 正在安全模式中操作，将仅校验已经从安全页表获得的主 TLB 内的描述符。

如果根据那个校验过程的结果，在主 TLB 中有命中，那么从相关描述符提取访问控制信息并在路径 242 上传回。特别地，在路径 242 上，将描述符的虚拟地址部分和相应的物理地址部分传送到微 TLB206，用于存储在微 TLB 的条目中，将访问许可权加载到访问许可逻辑 202 中，以及将区域属性加载到区域属性逻辑 204 中，使访问许可逻辑 202 和区域属性逻辑 204 与微 TLB 分开，或可以包含在微 TLB 中。

在这一点上，然后，MMU200 能处理存储请求，因为现在在微 TLB206 内有命中。因此，微 TLB206 将生成物理地址，然后，在路径 238 上输出到系统总线 40 上，用于传送到相关存储器，这将是片上存储器，诸

如 TCM36、高速缓存 38 等等，或可经外部总线接口 42 访问的外部存储单元的一个。此时，访问许可逻辑 202 将确定是否允许存储器访问，以及如果确定在当前操作模式中，不允许内核访问特定存储单元，在路径 230 上，将中止信号发送回内核 10。例如，当内核正在监督模式中操作时，某些存储器部分，不管是在安全存储器还是非安全存储器中，可以指定为仅可由那个内核访问，因此，如果当在例如用户模式中时，内核 10 正尝试访问该存储单元，访问许可逻辑 202 将检测内核 10 当前不具有适当的访问权，以及将在路径 230 上发出中止信号。这将导致中止存储器访问。最后，区域属性逻辑 204 将确定用于特定存储器访问的区域属性，诸如访问是否能高速缓存、可缓存等等，以及将在路径 232 上发出这些信号，然后，将使用它们来确定例如在高速缓存 38 内，是否能高速缓存存储器访问请求的主题的数据，在写入访问的情况下，是否能缓冲写入数据等等。

在主 TLB208 内没有命中的情况下，那么使用转换表行程逻辑(walk logic) 210 来访问相关页表 58 以便在路径 248 上检索所需描述符，然后在路径 246 上，将那个描述符传递到主 TLB208，将其存储在其中。用于非安全页表和安全页表的基地址将存储在 CP15 的寄存器 34 内，以及在 CP15 的寄存器内，还将设置处理器内核 10 正在操作的当前域，即安全域或非安全域，当在非安全域和安全域间发生转换，或反之亦然时，通过监视模式设置那个域状态寄存器。域状态寄存器的内容在此称为域位。因此，如果需要执行转换表行程过程，转换表行程逻辑 210 将知道内核 10 正在哪个域中操作，因此，其基地址用来访问相关表。然后将虚拟地址用作基地址的偏移以便访问适当页表内的适当项，从而获得所需描述符。

一旦通过转换表行程逻辑 210 检索到描述符，并放在主 TLB208 内，那么在主 TLB 内获得命中，以及调用先前描述的过程来检索访问控制信息，以及将其存储在微 TLB206、访问许可逻辑 202 和区域属性逻辑 204 内。然后通过 MMU200 能对存储器访问起作用。

如前所述，在优选实施例中，主 TLB208 能存储来自安全页表和非安全页表的描述符，但一旦相关信息存储在主 TLB206 内，仅由 MMU200 处理存储器访问请求。在优选实施例中，通过位于 MPU220 内的分区校验器 222 监视主 TLB208 和微 TLB206 间的信息传送，以确保在内核 10

正在非安全模式中操作的情况下，不会有访问控制信息从主 TLB208 内的描述符传送到微 TLB206，如果会导致在安全存储器内生成物理地址的话。

由能位于 CP15 的寄存器 34 内、划分定义安全和非安全存储器间的分区的信息的安全操作系统管理存储器保护单元。然后，分区校验器 222 能引用那个分区信息以便确定访问控制信息是否正传送到在非安全模式中，允许内核 10 访问安全存储器的微 TLB206。更具体地说，在优选实施例中，当内核 10 正在非安全操作模式中时，如用由 CP15 域状态寄存器内的监视模式设置的域位所示，分区校验器 222 能经路径 244 监视正试图从主 TLB208 恢复到微 TLB206 中的任何物理地址部分以及确定基于那个物理地址部分，对虚拟地址产生的物理地址是否在安全存储器内。在这种情况下，分区校验器 222 将在路径 230 上向内核 10 发出中止信号以防止发生存储器访问。

将意识到，另外，能将分区校验器 222 配置成真正防止那个物理地址部分存储在微 TLB206 中，或者，物理地址部分仍然位于微 TLB206 内，但中止过程部分将通过例如刷新微 TLB206，从微 TLB206 移出那个不正确的物理地址部分。

只要内核 10 经监视模式，在操作的非安全模式和安全模式间改变时，监视模式将改变 CP15 域状态寄存器内的域位值，以表示处理器的操作正在改变到该域中。作为域间传送过程的一部分，将刷新微 TLB206，因此，在安全域和非安全域间的转换后的第一存储器访问将在微 TLB206 中产生未命中，以及要求直接或经从相关页表检索相关描述符，从 TLB208 检索访问信息。

通过上述方法，将意识到分区校验器 222 将确保当内核正在非安全域中操作时，如果尝试使允许访问安全存储器的访问控制信息恢复到微 TLB206，将产生存储器访问中止。

如果在处理器内核 10 的任何操作模式中，将存储器访问请求配置成直接指定物理地址，那么，在那个操作模式中，将禁止 MMU200，以及在路径 236 上，将物理地址传送到 MPU220 中，在安全操作模式中，访问许可逻辑 224 和区域属性逻辑 226 将基于对 CP15 内的分区信息寄存器 34 内的相应区识别的访问许可权和区域属性，执行必要的访问许可和区域属性分析。如果正尝试访问的安全存储单元位于仅在某些操

作模式，例如安全特许模式中可访问的部分安全存储器内，那么，在不同操作模式，例如安全用户模式中，内核的访问尝试将使得访问许可逻辑 224 在路径 230 上，用与 MMU 的访问许可逻辑 202 在这些情况下产生中止相同的方式，向内核生成中止。类似地，区域属性逻辑 226 将用 MMU 的区域属性逻辑 204 以产生用于通过虚拟地址指定的存储器访问请求的那些信号相同的方式，生成可高速缓存和缓冲的信号。假定允许访问，然后，访问请求在路径 240 上进入到系统总线 40 上，从该总线传送到适当的存储单元。

对访问请求指定物理地址的非安全访问，经路径 236，使访问请求传送到分区校验器 222，分区校验器将引用 CP15 寄存器 34 内的分区信息，执行分区校验以便确定物理地址是否指定安全存储器内的单元，在这种情况下，在路径 230 上将再次产生中止信号。

现在，将参考图 39 和 40 的流程图，更详细地描述上述存储器管理逻辑的处理。图 39 示例说明在内核 10 上运行的程序产生虚拟地址的情形，如步骤 300 所示。由监视模式设置的 CP15 域状态寄存器 34 内的相关域位将表示内核当前正在安全域还是非安全域中运行。在内核正在安全域中运行的情况下，处理转移到步骤 302，其中在微 TLB206 内执行查找以便查看虚拟地址的相关部分是否与微 TLB206 内的虚拟地址部分相匹配。在步骤 302 命中的情况下，处理直接转移到步骤 312，其中访问许可逻辑 202 执行必要的访问分析。在步骤 314，然后确定是否有访问许可违反，以及如果有，过程进入步骤 316，其中访问许可逻辑 202 在路径 230 上发出中止。否则，在没有这种访问许可违反的情况下，过程从步骤 314 进入步骤 318，其中进入存储器访问。特别地，区域属性逻辑 204 将在路径 232 上输出必要的可高速缓存和缓冲的属性，以及微 TLB206 将在路径 238 上发出物理地址，如前所述。

如果在步骤 302，在微 TLB 中有未命中，那么在步骤 304，在主 TLB208 内执行查找过程以便确定所需安全描述符是否存在于主 TLB 中。如果没有，那么，在步骤 306 执行页表行程过程，由此转换表行程逻辑 210 从安全页表获得所需描述符，如参考图 37 所述。然后过程进入步骤 308，或在安全描述符已经存在于主 TLB208 中的情况下，直接从步骤 304 进入步骤 308。

在步骤 308，确定主 TLB 现在包含有效标记的安全描述符，因此，

过程进入步骤 310，通过包含物理地址部分的描述符的子部分，加载微 TLB。由于内核 10 当前正在安全模式中运行，不需要分区校验器 222 执行任何分区校验功能。

然后处理进入步骤 312，其中如前所述，执行存储器访问的剩余部分。

在非安全存储器访问的情况下，过程从步骤 300 转到步骤 320，其中，在微 TLB206 内执行查找过程以便确定来自非安全描述符的相应物理地址部分是否存在。如果是，那么处理直接转移到步骤 336，其中由访问许可逻辑 202 校验访问许可权。注意，在该点，如果相关物理地址部分在微 TLB 内，假定没有安全违反，由于分区校验器 222 在其存储在微 TLB 中前，有效地控制信息，以致如果信息位于微 TLB 内，假定它是适当的非安全信息很重要。只要在步骤 336 已经校验访问许可，处理进入步骤 338，其中确定是否有任何违反，在任一事件中，在步骤 316，发出访问许可故障中止。否则，处理进入步骤 318，其中执行存储器访问的剩余部分，如前所述。

在步骤 320，在微 TLB 中没有命中的情况下，处理进入步骤 322，其中在主 TLB208 中执行查找过程以便确定相关非安全描述符是否存在。如果没有，通过转换表行程逻辑 210，在步骤 324 执行页表行程过程以便使来自非安全页表的必要的非安全描述符恢复到主 TLB208 中。然后处理进入步骤 326，或在步骤 322 出现主 TLB208 内的命令的情况下，直接从步骤 322 进入步骤 326。在步骤 326，确定主 TLB 现在包含用于所述虚拟地址的有效标记非安全描述符，然后，在步骤 328，分区校验器 222 校验将从存储器访问请求的虚拟地址生成的物理地址（假定描述符内的物理地址部分）将指定非安全存储器中的单元。如果不，即，如果物理地址指向安全存储器中的单元，那么在步骤 330，确定有安全违反，以及处理进入步骤 331，其中由分区校验器 222 发出安全/非安全故障中止。

然而，如果分区校验器逻辑 222 确定没有安全违反，过程进入步骤 334，其中通过包含物理地址部分的相关描述符的子部分，加载微 TLB，然后，在步骤 336，用先前所述的方式，处理存储器访问。

现在，参考图 40 描述直接发出物理地址的存储器访问请求的处理。如前所述，在该情况下，将去激活 MMU200，这最好通过在 CP15

寄存器的相关寄存器内设置 MMU 允许位来实现，通过监视模式来实现该设置过程。其中，在步骤 350，内核 10 将生成在路径 236 传递到 MPU220 中的物理地址。然后，在步骤 352，MPU 校验许可以便校验正请求的存储器访问能处理当前操作模式，即用户、监督等等。此外，如果内核正在非安全模式中操作，在步骤 352，分区校验器 222 将校验物理地址是否在非安全存储器中。然后，在步骤 354，确定是否有违反，即，访问许可处理是否显示违反，或如果在非安全模式中，分区校验过程识别出违反。如果这些违反的任何一个发生，那么过程进入步骤 356，其中，通过 MPU220，生成访问许可故障中止。将意识到在某些实施例中，在两种中止间没有区别，而在另外的实施例中，中止信号能表示它与访问许可故障还是安全故障有关。

如果在步骤 354，未检测到违反，过程进入步骤 358，其中发生对由物理地址识别的单元的存储器访问。

在优选实施例中，仅将监视模式配置成直接生成物理地址，因此，在所有其他情况下，MMU200 将有效以及由存储器访问请求的虚拟地址生成物理地址将会发生，如前所述。

图 38 示例说明在所有存储器访问请求指定虚拟地址，因此，在任何操作模式中，不直接生成物理地址的情况下，存储器管理逻辑的另一实施例。在这种情况下，将意识到不需要单独的 MPU220，而是将分区校验器 222 包含在 MMU200 中。这改变除外，该处理正好用与参考图 37 和 39 所述的相同方式进行。

将意识到各种其他选择也是可能的。例如，假定通过指定虚拟地址的安全和非安全模式发出存储器访问请求，能提供两个 MMUs，一个用于安全访问请求以及一个用于非安全访问请求，即图 37 中的 MPU220 能用完整的 MMU 代替。在这些情况下，将不需要使用标记和每个 MMU 的主 TLB 来定义描述符是安全还是非安全，因为一个 MMU 将非安全描述符存储在其主 TLB 中，以及另一个 MMU 将安全描述符存储在其主 TLB 中。当然，仍然需要分区校验器来校验在内核处于非安全域中时，是否正尝试访问安全存储器。

另外，如果所有存储器访问请求直接指定物理地址，另外的实现可以使用两个 MPUs，一个用于安全访问请求，一个用于非安全访问请求。用于非安全访问请求的 MPU 将具有受分区校验器控制的其访问请

求以确保在非安全模式中，不允许访问安全存储器。

作为图 37 或图 38 配置所具有的另一特征，分区校验器 222 能配置成执行一些分区校验以便管理转换表行程逻辑 210 的活动性。特别地，如果内核当前正在非安全域中操作，那么分区校验器 222 能配置成只要转换表行程逻辑 210 正试图访问页表，校验它正在访问非安全页表而不是安全页表。如果检测到违反，最好生成中止信号。由于转换表行程逻辑 210 通常通过将页表基地址与通过存储器访问请求发出的虚拟地址的某些位结合来执行页表查找，该分区校验可以包含例如校验转换表行程逻辑 210 正使用非安全页表的基地址而不是安全页表的基地址。

图 41 示意性地示例说明当内核 10 正在非安全模式中操作时，由分区校验器 222 执行的过程。将意识到在正常的操作中，从非安全页表获得的描述符将仅描述在非安全存储器中映射的页。然而，在软件攻击的情况下，可以篡改描述符以便现在描述包含存储器的非安全和安全部分。因此，考虑图 41 中的例子，不可靠的非安全描述符可以覆盖包括非安全区 370、372、374 和安全区 376、378 和 380 的页。如果作为存储器访问请求的一部分发出的虚拟地址对应于安全存储区中的物理地址，例如，图 41 中所示的安全存储区 376，那么分区校验器 222 配置成生成中止以便防止访问发生。因此，即使在尝试访问安全存储器中损坏非安全描述符，分区校验器 222 可防止访问发生。相反，如果使用该描述符导出的物理地址对应于非安全存储区，例如图 41 中所示的区域 374，那么加载到微 TLB206 中的访问控制信息仅识别该非安全区 374。因此，非安全存储区 374 内的访问会发生，但不会发生访问安全区 376、378 或 380 的任何一个。因此，能看出即使主 TLB208 可以包含来自已经篡改的非安全页表的描述符，微 TLB 将仅包含将允许访问非安全存储区的物理地址部分。

如前所述，在非安全模式或安全模式可以生成指定虚拟地址的存储器访问请求的实施例，那么存储器最好包括非安全存储器内的非安全页表，以及安全存储器内的安全页表。当在非安全模式中，将通过转换表行程逻辑 210 引用非安全页表，而当在安全模式中，将通过转换表行程逻辑 210 引用安全页表。图 42 示例说明这两个页表。如图 42 所示，可以位于例如图 1 的外部存储器 56 中的非安全存储器 390 在其

其中包括通过引用基地址 397，在 CP15 寄存器 34 中指定的非安全页表 395。类似地，在同样位于图 1 的外部存储器 56 内的安全存储器 400 中，提供通过安全页表基地址 407，在专用 CP15 寄存器 34 内指定的相应安全页表 405。非安全页表 395 的每个描述符将指向非安全存储器 390 中的相应的非安全页，而安全页表 405 内的每个描述符将定义安全存储器 400 中的相应安全页。另外，如稍后更详细所述，对存储器的某些区域，可以共享可以由非安全模式和安全模式访问的存储区 410。

图 43 更详细地示例说明根据优选实施例，在主 TLB208 内执行的查找过程。如前所述，主 TLB208 包括识别相应的描述符是来自安全页表还是非安全页表的安全标记 425。这确保当执行查找过程时，将仅校验与内核 10 正在其中操作的特定域有关的描述符。图 43 示例说明内核正在安全域，也称为安全区域中运行的例子。如从图 43 所看出的，当执行主 TLB208 查找时，将导致忽略描述符 440，以及仅将描述符 445 识别为用于查找过程的候选。

根据优选实施例，提供在此也称为 ASID 标记的另外的过程 ID 标记 430，以便从过程特定页表识别描述符。因此，过程 P1、P2 和 P3 分别可以具有在存储器内提供的相应的页表，以及另外可以具有用于非安全操作和安全操作的不同页表。另外，将意识到安全域中的过程 P1、P2 和 P3 可以完全将过程与非安全域中的过程 P1、P2 和 P3 分开。因此，如图 43 所示，除当要求主 TLB 查找 208 时校验域外，还校验 ASID 标记。

因此，在图 43 的例子中，其中在安全域中，执行过程 P1，该查找过程正好识别主 TLB208 内的两个项 450，然后根据那两个描述符内的虚拟地址部分是否与由存储器访问请求发出的虚拟地址的相应部分匹配，生成命中或未命中。如果是，那么抽取相关访问控制信息并传递到微 TLB206、访问许可逻辑 202 和区域属性逻辑 204。否则，发生未命中，使用转换表行程逻辑 210 来将来自为安全过程 P1 提供的页表的所需描述符恢复到主 TLB208。如本领域的技术人员将意识到的，有许多用于管理 TLB 的内容的技术，因此，当检索存储在主 TLB208 中的新描述符时，主 TLB 已经满，可以使用多种已知技术的任何一个来确定从主 TLB 驱出哪个描述符来为新描述符腾出空位，例如最近使用

方法等等。

将意识到用在操作的安全模式中的安全核心可以开发成与非安全操作系统完全分开。然而，在某些情况下，安全核心和非安全操作系统的开发可以紧密结合在一起，以及在这些情况下，允许安全应用程序使用非安全描述符也是适当的。的确，这允许安全应用程序通过仅知道虚拟地址，直接访问非安全数据（用于共享）。当然，这假定对特定 ASID，安全虚拟映射和非安全虚拟映射是互斥的。在这些情况下，将不需要先前介绍的区分安全和非安全描述符的标记（即域标记）。然后，通过所有可用描述符，执行 TLB 中的查找。

在优选实施例中，通过在 CP15 控制寄存器内提供的特定位，能设置主 TLB 的配置和具有单独的安全和非安全描述符的先前描述的配置间的选择。在优选实施例中，该位仅通过安全核心设置。

在直接允许安全应用程序使用非安全虚拟地址的实施例中，将可以从安全域获得非安全栈指针。这能通过将识别非安全栈指针的非安全寄存器值拷贝到 CP15 寄存器 34 内的专用寄存器来完成。然后，根据由安全应用程序理解的方案，这将允许非安全应用程序经栈传递参数。

如前所述，可以将存储器划分成非安全和安全部分，以及使用专用于分区校验器 222 的 CP15 寄存器 34，控制该分区。基本分区方法基于如在典型的 MPU 设备中定义的区域访问许可。因此，存储器划分成区域，以及每个区域最好用其基地址、大小、存储器属性和访问许可来定义。另外，当覆盖区域被编程时，上一区域的属性获得最高优先级。另外，根据本发明的优选实施例，提供新区域属性以便定义相应区域位于安全存储器还是非安全存储器中。由安全核心使用该新区域属性以便定义将保护为安全存储器的存储器部分。

在引导阶段，如图 44 所示，执行第一分区。该初始分区将定义分配到非安全区域、非安全操作系统和非安全应用程序的存储器量 460。该量对应于在该分区中定义的非安全区。然后由非安全操作系统使用该信息，用于其存储器管理。非安全操作系统不知道定义为安全的存储器的其余部分 462、464。为保护非安全区域中的完整性，通过仅用于安全特许模式的访问许可，编程非安全存储器。因此，安全应用程序将不损坏非安全应用程序。如从图 44 所看到的，在该引导阶段分区

后，存储器 460 可用于由非安全操作系统使用，存储器 462 可用于由安全核心使用，以及存储器 464 可用于由安全应用程序使用。

一旦执行引导阶段分区，使用 MMU200，由非安全操作系统处理非安全存储器 460 的存储器映射，因此，能用常规方式定义一系列非安全页。这如图 45 所示。

如果安全应用程序需要与非安全应用程序共享存储器，安全内核能改变存储器部分的权利以便将数据从一个域人工传送到另一个。因此，如图 46 所示，在校验非安全页的完整性后，安全核心改变那个页的权利以便它变成共享存储器可访问的安全页 466。

当存储器的分区改变后，需要刷新微 TLB206。因此，在这种情况下，当顺序地发生非安全访问时，在微 TLB206 中将发生未命中，因此，将从主 TLB208 加载新描述符。当尝试将其恢复到微 TLB206 中时，由 MPU 的分区校验器 222 顺序地校验该新描述符，因此，将与存储器的新分区一致。

在优选实施例中，高速缓存 38 是虚拟索引和物理标记的。因此，当在高速缓存 38 中执行访问时，将首先在微 TLB206 中执行查找，并因此，对访问许可，特别是安全和非安全许可进行校验。因此，通过非安全应用程序，不能将安全数据存储在高速缓存 38 中。对高速缓存 38 的访问是在由分区校验器 222 执行的分区校验的控制之下，因此，在非安全模式中，不能执行访问安全数据。

然而，会发生的一个问题将是非安全域中的应用程序，能使用高速缓存操作寄存器来无效、清除或刷新该高速缓存。需要保证这些操作将不影响系统的安全性。例如，如果非安全操作系统想使高速缓存 38 无效，而不是清除它，任何安全恶劣数据数据在替换前，必须写到外部存储器。最好，在高速缓存中标记安全数据，因此，如果需要的话，能不同地处理。

在优选实施例中，如果通过非安全程序执行“通过地址使行无效”操作，由分区校验器 222 校验物理地址，以及如果高速缓存行是安全高速缓存行，操作变为“清除和无效”操作，从而维护确保系统的安全性。另外，在优选实施例中，由非安全程序执行的所有“通过索引使行无效”操作变为“通过索引清除和无效”操作。类似地，由非安全程序执行的所有“无效全部”操作变为“清除和无效全部”操作。

此外，参考图 1，由微 TLB206 控制通过 DMA32，对 TCM36 的任何访问。因此，当 DMA32 在 TLB 中执行查找以便将其虚拟地址翻译成物理地址时，已经增加在主 TLB 中的先前描述的标记允许执行所需安全性校验，正如已经由内核 10 发出访问请求一样。另外，如稍后所述，将复制分区校验器耦合到外部总线 70，最好位于判优器/解码器块 54 中，以便如果 DMA32 直接访问经外部总线接口 42 耦合到外部总线 70 的存储器时，连接到外部总线的复制分区校验器校验访问的有效性。此外，在某些优选实施例中，能通过将位增加到 CP15 寄存器 34 来定义 DMA 控制器 32 是否能用在非安全域中，当操作在特许模式中时，该位仅允许由安全核心设置。

考虑 TCM36，如果安全数据位于 TCM36 中，这必须小心处理。例如，能想像非安全操作系统编程用于 TCM 存储器 36 的物理地址范围以便它覆盖外部安全存储器部分的情形。如果操作模式改变成安全模式，安全核心会导致数据存储在那个覆盖部分，通常，将数据存储在 TCM36 中，因为 TCM36 通常具有比外部存储器更高的优先级。如果非安全操作系统改变用于 TCM36 的物理地址空间的设定值以便在存储器的非安全物理区中映射先前安全区，将意识到非安全操作系统能访问安全数据，因为分区校验器将该区域看作非安全并且将不断言中止。因此，为概述，如果将 TCM 配置成充当常规本地 RAM 以及不充当智能高速缓存，如果它能将 TCM 基寄存器移动到非安全物理地址，那么非安全操作系统可以读取安全区域数据。

为防止这种情况下，在优选实施例中，在 CP15 寄存器 34 内提供控制位，其仅可在安全特许操作模式中访问，以及提供两种可能的配置。在第一种配置中，将该控制位设置成“1”，在这种情况下，仅能由安全特许模式控制 TCM。因此，尝试的任何非安全访问 CP15 34 内的 TCM 控制寄存器将导致输入未定义的指令异常。因此，在该第一实施例中，安全模式和非安全模式能使用 TCM，但仅由安全特许模式控制 TCM。在该第二配置中，将控制位设置成“0”，在这种情况下，由非安全操作系统控制 TCM。在这种情况下，仅由非安全应用程序使用 TCM。任何安全数据都不能从 TCM 存储或加载。因此，当执行安全访问时，在 TCM 内不执行查看地址是否与 TCM 地址范围匹配的查找。

根据缺省情况，设想 TCM 仅由非安全操作系统使用，因为在这种

情况下，不需要改变非安全操作系统。

如前所述，除在 MPU220 内提供分区校验器 222 外，本发明的优选实施例还提供耦合到外部总线 70 的类似分区校验块，该另外的分区校验器用来管理由其他主设备，例如数字信号处理器（DSP）50、直接耦合到外部总线的 DMA 控制器 52、经外部总线接口 42 可连接到外部总线的 DMA 控制器 32 等等访问存储器。的确，在一些实施例中，如稍后所述，可以仅具有耦合到外部（或设备）总线上的分区校验块，以及不提供作为存储器管理逻辑 30 的一部分的分区校验器。在一些这种实施例中，可以可选地将分区校验器提供为存储器管理逻辑 30 的一部分，在这些实例中，该分区校验器能视为除耦合到设备总线的以外提供的另一分区校验器。

如前所述，整个存储器系统能由几个存储器单元组成，以及多个这些存储器单元可以存在于外部总线 70 上，例如外部存储器 56、引导 ROM44 或外设，诸如屏幕驱动器 46、I/O 接口 60、键存储单元 64 等等内的实际缓冲器或寄存器 48、62、66 上。此外，需要将存储器系统的不同部分定义为安全存储器，例如可以期望键存储单元 64 内的键缓冲器 66 处理为安全存储器。如果耦合到外部总线上的设备试图访问该安全存储器，那么很显然，在包含在内核 10 中的芯片内的先前所述的存储器管理逻辑 30 将不能管理该访问。

图 47 示例说明如何使用耦合到外部总线，在此也称为设备总线的另外的分区校验器 492。外部总线通常配置成只要通过设备，诸如设备 470、472 将存储器访问请求发送到那个外部总线上，那些存储器访问请求还包括定义操作模式，例如特许、用户等待的外部总线上的某些信号。根据本发明的优选实施例，存储器访问请求还包含将域信号发送到外部总线上以便识别该设备正在安全模式还是非安全模式中操作。最好在硬件级发出该域信号，以及在优选实施例中，能在安全或非安全域中操作的设备将包括用于在外部总线内的路径 490 上，输出域信号的预定管脚。为示例说明目的，该路径 490 与外部总线上的其他信号路径 488 分开表示。

在此也称为“S 位”的该域信号将识别发出存储器访问请求的设备正在安全域还是非安全域中操作，以及由耦合到外部总线的分区校验器 492 接收该信息。分区校验器 492 还访问识别存储区为安全还是

非安全的分区信息，因此，能配置成如果断定 S 位识别安全操作模式，仅允许设备访问存储器的安全部分。

根据缺省情况，设想不断定 S 位，因此，预先存在的非安全设备，诸如图 47 所示的设备 472 将不输出断言 S 位，因此，将永无保证分区校验器 492 访问存储器的任何安全部分，不管在屏幕驱动器 480 的寄存器或缓冲器 482、486、I/O 接口 484 还是外部存储器 474 内。

为方便说明，与用来确定提供存储器访问请求服务的适当的存储器设备的解码器 478 分开说明用来在由主设备，诸如设备 470、472 发出的存储器访问请求间判优的判优器块 476，并与分区校验器 492 分开。然而，将意识到如果需要的话，可以在相同单元中集成这些部件的一个或多个。

图 48 示例说明另外的实施例，其中不提供分区校验器。相反，将每个存储器设备 474、480、484 配置成根据 S 位值，管理其自身的存储器访问。因此，如果设备 470 想在非安全模式中向标记为安全存储器的屏幕驱动器 480 内的寄存器 482 提出存储器访问请求，那么屏幕驱动器 480 判定 S 位是不断言的，以及不处理存储器访问请求。因此，设想通过适当的各种存储器器件的设计，可以避免在外间总线上单独提供分区校验器 492 的需要。

在图 47 和 48 的上述描述中，所述“S 位”识别发出存储器访问请求的设备正在安全域还是非安全域中操作。看另一种方法，能将该 S 位看成表示存储器访问请求属于安全域还是非安全域。

在参考图 37 和 38 所述的实施例中，使用单个 MMU 以及单个页表集来执行虚拟地址到物理地址的转换。通过这种方法，通常用简单的方式在非安全存储器和安全存储器间分段物理地址空间，如图 49 所示。其中，物理地址空间包括从地址 0 开始并延伸到用于存储器系统，例如外部存储器 56 内的一个存储单元的地址 Y 的地址空间。对每个存储单元，可寻址存储器通常分成两部分，第一部分 2110 分配成非安全存储器以及第二部分 2120 分配成安全存储器。

通过这种方法，将意识到存在不能由特定域访问的某些物理地址，以及这些差异对用在那些域中的操作系统是显而易见的。同时用在安全域中的操作系统将了解非安全域，因此，对此将不关心，非安全域中的操作系统理论上将不需要了解存在安全域，但相反，应当就像安

全域不存在一样操作。

作为另外的问题，将意识到非安全操作系统将用于外部存储器的地址空间看成以地址 0 开始并延伸到地址 X，以及非安全操作系统不知道安全核心的任何信息，特别地存在从地址 X+1 延伸到地址 Y 的安全存储器。相反，安全核心将看不见从地址 0 开始的地址空间，这不是操作系统通常所期望的。

在图 51 中示意性地示例说明通过允许从其物理地址空间的非安全操作系统的视图安全隐藏安全存储区，以及通过允许安全域中的安全核心和非安全域中的非安全操作系统将用于外部存储器的它们的地址空间看成从地址 0 开始，避免上述问题的一个实施例。其中，物理地址空间 2200 能在页组分成安全或非安全段。在图 51 所示的例子中，用于外部存储器的地址空间表示为分成四个部分 2210、2220、2230 和 2240，由两个安全存储区和两个非安全存储区组成。

不是经单个页表转换的虚拟地址空间和物理地址空间间的转换，参考第一页表和第二页表执行两个单独地址层转换，从而允许引入根据处理器是在安全域还是非安全域中，不同配置的中间地址空间的概念。更具体地说，如图 51 所示，通过使用在页表集 2250 中的安全页表内提供的描述符，能将物理地址空间中的两个安全存储区 2210 和 2230 映射到用于安全域的中间地址空间中的单个区域 2265。关于所涉及的在处理器上运行的操作系统，将中间地址空间看成物理地址空间，以及使用 MMU 来将虚拟地址转换成中间地址空间内的中间地址。

类似地，对非安全域，能配置中间地址空间 2270，其中，经页表集 2250 内的非安全页表中的相应的描述符，将物理地址空间中的两个非安全存储区 2220 和 2240 映射到用于非安全域的中间地址空间中的非安全区 2275。

在一个实施例中，使用两个单独的 MMU，处理经中间地址，将虚拟地址转换成物理地址，如图 50A 所示，图 50A 中的 MMU2150 和 2170 的每一个能视为用与图 37 所示的 MMU200 类似的方式构成，但为方便说明，在图 50A 中省略某些细节。

第一 MMU2150 包括微 TLB2155、主 TLB2160 和转换表行程逻辑 2165，类似地，第二 MMU2170 包括微 TLB2175、主 TLB2180 和转换表行程逻辑 2185。当处理器正在非安全域中操作时，通过非安全操作系

统控制第一 MMU，或当处理器正在安全域中操作时，通过安全核心控制第一 MMU。然而，在优选实施例中，只能由安全核心或者监视程序控制第二 MMU。

当处理器内核 10 发出存储器访问请求时，在路径 2153 上，将虚拟地址发送到微 TLB2155。微 TLB2155 将存储用于多个虚拟地址部分的从存储在主 TLB2160 内的描述符、已经从与第一 MMU2150 有关的第一页表集检索的主 TLB2160 中的描述符检索的相应的中间地址部分。如果在微 TLB2155 中检测到命中，那么微 TLB2155 将在路径 2157 上发送对应于在路径 2153 上接收的虚拟地址对应的中间地址。如果在微 TLB2155 中无命中，那么将引用主 TLB2160 来查看是否在主 TLB 内检测到命中，以及如果是的话，将相关虚拟地址部分和相应的中间地址部分恢复到微 TLB2155 中，然后，在路径 2157 上发送中间地址。

如果在微 TLB2155 和主 TLB2160 中无命中，那么使用转换表行程逻辑 2165 来发送用于来自可由第一 MMU2150 访问的第一页表集中的预定页表的所需描述符的请求。典型地，具有与用于安全域或非安全域的单个处理器有关的页表，以及可由转换表行程逻辑 2165 访问用于那些页表的中间地址，例如从 CP15 寄存器 34 内的适当寄存器。因此，转换表行程逻辑 2165 能在路径 2167 上发送中间地址以便从适当页表请求描述符。

第二 MMU2170 经配置在路径 2157 上，由微 TLB2155，或在路径 2167 上，接收由转换表行程逻辑 2165 输出的任何中间地址，以及如果在微 TLB2175 内检测到命中，那么，在路径 2192 上，微 TLB 将所需物理地址发送到存储器以便在数据总线 2190 上检索所需数据。在路径 2157 上发送中间地址的情况下，这将导致所需数据返回到内核 10，而对在路径 2167 上发送的中间地址，将导致所需描述符返回到第一 MMU2150，用于存储在主 TLB2160 中。

在微 TLB2175 未命中的情况下，将引用主 TLB2180，以及如果在主 TLB 内有命中，将所需中间地址部分和相应的物理地址部分返回到微 TLB2175，然后允许微 TLB2175 在路径 2192 上发送所需物理地址。然而，在微 TLB2175 或主 TLB2170 中均缺少命中的情况下，那么，将转换表行程逻辑 2185 配置成在路径 2194 上输出从与第二 MMU2170 有关的第二页表集内的相关页表，请求所需描述符。该第二页表集包括

将中间地址部分与物理地址部分关联的描述符，通常至少存在用于安全域的一个页表和用于非安全域的一个页表。当在路径 2194 上发出请求时，将导致来自第二页表集的相关描述符返回到第二 MMU2170，用于存储在主 TLB2180 内。

现在，通过如下所述的特定的例子，进一步示例说明图 50A 中所示的实施例的操作，其中缩写 VA 表示虚拟地址，IA 表示中间地址，以及 PA 表示物理地址。

- 1) 内核发出 $VA=3000$ [$IA=5000, PA=7000$]
- 2) MMU1 的微 TLB 中的未命中
- 3) MMU1 的主 TLB 中的未命中
页表 1 基地址 = $8000IA$ [$PA=10000$]
- 4) MMU1 中的转换表行程逻辑执行页表查找-发出 $IA=8003$
- 5) MMU2 的微 TLB 中的未命中
- 6) MMU2 的主 TLB 中的未命中
页表 2 基地址 = $12000PA$
- 7) MMU2 中的转换表行程逻辑执行页表查找-发出 $PA=12008$
“ $8000IA=10000PA$ ” 返回为页表数据
- 8) - 存储在 MMU2 的主 TLB 中
- 9) - 存储在 MMU2 的微 TLB 中
- 10) MMU2 中的微 TLB 现在命中-发出 $PA=10003$ “ $3000VA=5000IA$ ”
返回为页表数据
- 11) - 存储在 MMU1 的主 TLB 中
- 12) - 存储在 MMU1 的微 TLB 中
- 13) MMU1 中的微 TLB 现在命中-发出 $IA=5000$ 来执行数据访问
- 14) MMU2 的微 TLB 中的未命中
- 15) MMU2 的主 TLB 中的未命中
- 16) MMU2 中的转换表行程逻辑执行页表查找-发出 $PA=12005$
“ $5000IA=7000PA$ ” 返回为页表数据
- 17) - 存储在 MMU2 的主 TLB 中
- 18) - 存储在 MMU2 的微 TLB 中
- 19) MMU2 中的微 TLB 现在命中-发出 $PA=7000$ 来执行数据访问
- 20) 物理地址 7000 处的数据返回到内核

下次内核发出存储器访问请求（假定 VA3001..）

- 1) 内核发出 VA=3001
- 2) 在 MMU1 的微 TLB 中命中，请求发送到 MMU2 的 IA5001
- 3) 在 MMU2 的微 TLB 中命令，请求发送到存储器的 PA7001
- 4) 返回到内核的 PA7001 的数据。

将意识到在上述例子中，在两个 MMU 的微 TLB 和主 TLB 中发生未命中，因此，该例子代表“最坏情况”情形。典型地，期望在微 TLB 或主 TLB 的至少一个中观察到命中，从而显著地降低检索数据所花的时间。

返回到图 51，通常在物理地址空间的某些区，在优选实施例中为安全区中提供第二页表集 2250。将第一页表集分成两种，命名为安全页表和非安全页表。最好，安全页表将连续地出现在中间地址空间 2265 内，就象非安全页表位于非安全中间地址空间 2275 中一样。然而，它们不需要连续地位于物理地址空间中，因此，例如，用于第一页表集的安全页表分布在整个安全区 2210、2230 上，用类似的方法，非安全页表可以分布在整个非安全存储区 2220 和 2240 上。

如前所述，使用两个页表集的二级方法的一个主要好处是对安全域的操作系统和非安全域的操作系统，能将物理地址空间配置成以 0 开始，这是通常操作系统所期望的。另外，从其“物理地址”空间的非安全操作系统看，能完全隐藏安全存储区，因为将中间地址空间看作其物理地址空间，其能配置具有连续的中间地址序列。

另外，使用这种方法大大地简化在非安全存储器和安全存储器间交换存储区的处理。这参考图 52 来示意说明。如图 52 所看到的，例如为单个存储页的存储区 2300 可以存在于非安全存储区 2220 中，以及类似地，存储区 2300 可以存在于安全存储区 2210 中。然而，能仅通过交换第二页表集内的相关描述符，容易交换这两个存储区 2300 和 2310，以便区域 2300 现在变成映射到安全域的中间地址空间中的区域 2305 的安全区，而区域 2310 能变成映射到非安全区的中间地址空间中的区域 2315 的非安全区。对安全域和非安全域中的操作系统来说，完全透明发生，因为它们的物理地址空间的视图实际上分别是安全域或非安全域的中间地址空间。因此，该方法避免了重新定义每个操作

系统内的物理地址空间。

现在，参考图 50B 描述使用两个 MMU 的本发明的另一实施例，与图 50A 的配置不同。从图 50B 与图 50A 的比较可以看出，配置几乎相同，但在该实施例中，将第一 MMU2150 配置成执行虚拟地址到物理地址转换以及第二 MMU 配置成执行中间地址到物理地址转换。因此，代替从第一 MMU2150 中的微 TLB2155 到用在图 50A 实施例中的第二 MMU2170 中的微 TLB2175 的路径 2157，将第一 MMU 中的微 TLB2155 配置成在路径 2192 上直接输出物理地址，如图 50B 所示。现在，通过在下文中示出的具体的例子来示例说明在图 50B 中所示的实施例的操作，其详述相同内核存储器访问请求的处理，如先前对图 50A 实施例所述。

- 1) 内核发出 $VA=3000$ [IA=5000, PA=7000]
- 2) MMU1 的微 TLB 和主 TLB 中的未命中
页表 1 基地址 = 8000IA [PA=10000]
- 3) MMU1 中的转换表行程逻辑执行页表查找-发出 IA=8003
- 4) MMU2 的微 TLB 和主 TLB 中的未命中
页表 2 基地址 = 12000PA
- 5) MMU2 中的转换表行程逻辑执行页表查找-发出 PA=12008
“8000IA=10000PA” 返回为页表数据
- 6) -存储在 MMU2 的主和微 TLB 中的 “8000IA=10000PA” 映射
- 7) MMU2 中的微 TLB 现在能将来自步骤 (3) 的请求转换成 PA1003 以及发出提取 “3000VA=5000IA” 返回为页表数据
- 注意：该转换通过 MMU1 保留在临时存储器中，但不直接存储在任何 TLB 中。
- 8) MMU1 中的转换表行程逻辑现在向 MMU2 发出 IA=5000 的请求
- 9) MMU2 的 uTLB 和主 TLB 中的 IA5000 未命中
- 10) MMU2 中的转换表行程逻辑执行页表查找-发出 PA=12005
“5000IA=7000PA” 返回为页表数据
- 11) MMU2 将 “5000IA=7000PA” 存储在 uTLB 和和主 TLB 中。该转换还传送到 MMU1。
- 12a) MMU2 发出 PA=7000 存储器访问
- 12b) MMU1 将 “3000VA=5000IA” 和 “5000IA=7000PA” 描述符

结合以给出“3000VA=7000PA”描述符，其存储在 MMU1 的主 TLB 和微 TLB 中。

13) 将 PA7000 处的数据返回到内核

下次内核发出存储器访问请求（假定 VA3001..）

1) 内核发出 VA=3001

2) 在 MMU1 的微 TLB 中命中，MMU1 发出 PA=7001 请求

3) PA7001 处的数据返回到内核。

如从上述例子与图 50A 所提供的比较可以看出，主要差别是在步骤 7 中，其中 MMU1 不直接存储第一表描述符，以及在步骤 12b (12a 和 12b 能同时发生)，其中 MMU1 还接收 IA->PA 转换以及进行结合并将其组合描述符存储在其 TLBs 中。

因此，能看出虽然该另一实施例仍然使用两个页表集来将虚拟地址转换成物理地址，微 TLB2155 和主 TLB2160 存储直接虚拟地址到物理地址转换的事实避免了当在微 TLB2155 或主 TLB2160 中产生命中时，在两个 MMU 中执行查找的需要。在这些情况下，第一 MMU 能直接处理来自内核的请求，而不参考第二 MMU。

将意识到第二 MMU2170 能配置成不包括微 TLB2175 和主 TLB2180，在任一情况下，页表行程逻辑 2185 将用于需要由第二 MMU 处理的每个请求。假定对第二 MMU 的需求不是很频繁，这可以节省第二 MMU 的复杂性和成本，达到可以接受的程度。由于第一 MMU 需要用于每个请求，通常有利地将微 TLB2155 和主 TLB2160 包括在第一 MMU2150 中以便提高第一 MMU 的操作速度。

应注意页表中的页可以改变大小，因此，用于转换的两半部分的描述符与不同大小的页有关。典型地，MMU1 页将小于 MMU2 页但这不一定必须如此。例如：

表 1 将在 0x40003000 的 4Kb 映射到 0x00081000 上

表 2 将在 0x00000000 的 1Mb 映射到 0x02000000 上

其中，两个大小的最小一个必须用于组合转换，因此，组合描述符是

将在 0x40003000 的 4Kb 映射到 0x02081000 上。

然而，其中，在区域间交换数据（如先前参考图 52 所述），反向成立是可能的，例如：

表 1 将在 0xc0003000 的 1Mb 映射到 0x00000000 上

表 2 将在 0x00042000 的 4Kb 映射到 0x02042000 上

现在，在地址 0xc0042010 从内核查找给出映射：

将在 0xc00420000 的 4Kb 映射到 0x02042000 上

即，两个大小的较小一个总是用于组合映射

注意在第二种情况下，处理效率较低，因为将重复查找和放弃表 1 中的（1Mb）描述符，因为访问不同 4Kb 区。然而，在典型的系统中，表 2 描述符将更大（如在第一例子中），大部分时间更有效（对指向适当 IA 空间部分的其他 4Kb 页，能再循环 1Mb 映射）。

作为采用两个单独的 MMU 的另一方案，如在图 50A 和 50B 中所示，能使用单个 MMU，如在图 53 中所示，其中在主 TLB2420 未命中时，由 MMU 产生异常，然后使软件在内核 10 中运行以便基于来自两个不同页表集的组合，产生虚拟到物理地址转换。更具体地说，如图 53 所示，内核 10 耦合到 MMU2400，其包括微 TLB2410 和主 TLB2420。当内核 10 发出存储器访问请求时，在路径 2430 上提供虚拟地址，以及如果在微 TLB 中发现命中，那么在路径 2440 上直接输出相应的物理地址，使得在路径 2450 上使数据返回到内核 10 中。然而，如果在微 TLB2410 中有未命中，引用主 TLB2420 以及如果相关描述符包含在主 TLB 中，使相关虚拟地址部分和相应的物理地址部分恢复到微 TLB2410 中，其后，在路径 2440 上发送物理地址。然而，如果主 TLB 也产生未命中，那么在路径 2422 上，向内核产生异常。现在，将进一步结合图 54 来描述从接收这一异常，在内核中执行的过程。

如图 54 所示，如果在步骤 2500，由内核检测到 TLB 未命中，那么在步骤 2510，以用于那个异常的预定矢量，内核进入监视模式。因此，这将导致运行页表合并码来执行在 54 所示的剩余步骤。

更具体地说，在步骤 2520，检索在路径 2430 上发送的，并引起微 TLB2410 和主 TLB2420 中的未命中的虚拟地址（在下文中称为故障虚拟地址），其后，在步骤 2530，根据用于第一表集内的适当表的中间基地址，确定用于所需第一描述符的中间地址。一旦已经确定中间地址（通常由虚拟地址和中间基地址的一些预定组合），那么参考第二表集内的相关表以便获得用于第一描述符的相应的物理地址。此后，在步骤 2550，能从存储器提取第一描述符以便允许确定用于故障虚拟

地址的中间地址。

然后，在步骤 2560，再次引用第二表来查找给出用于故障虚拟地址的中间地址的物理地址的第二描述符。此后，在步骤 2570，提取第二描述符以便获得用于故障虚拟地址的物理地址。

一旦已经获得上述信息，那么程序合并第一和第二描述符以便生成向物理地址转换提供所需的虚拟地址的新描述符，在步骤 2580 执行该步骤。用先前参考图 50B 所述的类似的方式，通过软件执行合并再次使用用于联合组合的最小页表大小。此后，在步骤 2590，将新描述符存储在主 TLB2420 内，此后，在步骤 2595，该过程由于异常返回。

此后，内核 10 将配置成在路径 2430 上重新发出用于存储器访问请求的虚拟地址，这仍将导致主 TLB2410 中的未命中，但现在，导致主 TLB2420 中命中。因此，能使虚拟地址部分和相应的物理地址部分恢复到微 TLB2410 中，此后，微 TLB2410 能在路径 2440 上发送物理地址，产生在路径 2450 上返回到内核 10 的所需数据。

将意识到，作为先前参考图 50A 和 50B 所述的替代实施例，使用参考图 53 和 54 所述的原理，能通过软件管理那些实施例中的一个或两个 MMU。

不管是否使用两个 MMU，如图 50A 或 50B 所示，或使用一个 MMU，如图 53 所示，当操作在监视模式中时（或者在特许安全模式中），由处理器管理第二页表集的事实确保了那些页表安全。因此，当处理器位于非安全域中时，仅能看见非安全存储器，因为当位于非安全域中时，它仅是通过第二页表集，为非安全域产生的、处理器能看见的中间地址。因此，不需要将分区校验器提供为如图 1 所示的存储器管理逻辑 30 的一部分。然而，仍然在外部总线上提供分区校验器以便监视由系统中的其他总线主机进行的访问。

在参考图 37 和 38 先前的实施例中，提供与 MMU200 有关的分区校验器 222，因此，当在高速缓存 38 中执行访问时，将已经首先在微 TLB206 中执行查找，因此，校验访问许可，特别是安全和非安全许可，可能已经完成。因此，在这些实施例中，通过非安全应用程序，安全数据不能存储在高速缓存 38 中。访问高速缓存 38 是在由分区校验器 222 执行分区校验的控制下，因此，在非安全模式中，不能执行访问安全数据。

然而，在本发明的替代实施例中，不提供用于监视在系统总线 40 上进行访问的分区校验器 222，相反数据处理装置仅具有耦合到外部总线 70，用于监视访问连接到那个外部总线的存储器单元的单个分区校验器。在这些实施例中，这意味着处理器内核 10 能访问直接耦合到系统 40 上的任何存储器单元，例如 TCM36 和高速缓存 38，而那些访问不受外部分区校验器控制，因此，需要一些机制来确保处理器内核 10 当在非安全模式中操作时，不访问高速缓存 38 或 TCM36 内的安全数据。

图 55 示例说明根据本发明的一个实施例的数据处理装置，提供机制来允许高速缓存 38 和/或 TCM36 来控制对它们的访问，而不需要提供与 MMU200 有关的任何分区校验逻辑。如图 55 所示，经 MMU200 将内核 10 耦合到系统总线 40、也耦合到系统总线 40 上的高速缓存 38 和 TCM36。经外部总线接口 42，使内核 10、高速缓存 38 和 TCM36 耦合到如图 55 所示、由地址总线 2620、控制总线 2630 和数据总线 2640 组成的外部总线 70。

能将内核 10、MMU200、高速缓存 38、TCM36 和外部总线接口 42 看作构成连接到外部总线 70，也称为设备总线上的单个设备，以及其他设备也可以耦合到那个设备总线，例如安全外围设备 470 或非安全外围设备 472。在设备总线 70 上还连接一个或多个存储器单元，例如外部存储器 56。另外，总线控制单元 2650 也连接到设备总线 70，以及通常包括判优器 2652、解码器 2654 和分区校验器 2656。对连接到设备总线的部件的操作的一般论述，应当参考先前所述的图 47。在先前所述的图 47 中，判优器、解码器和分区校验器以单独的模块图示，但当位于单个控制块 2650 中时，这些元件用相同方式工作。

在图 56 中，更详细地示例说明图 55 的 MMU200。通过图 56 和图 37 的比较，能看出以完全与图 37 的 MMU 相同的方式构成 MMU200，唯一区别在于不提供用于监视在主 TLB208 和微 TLB206 间，在路径 242 上发送的数据的分区校验器 222。如果处理器内核 10 发出指定虚拟地址的存储器访问请求，那么经 MMU200，路由存储器请求，如参考图 37 所述处理，产生在路径 238 上，从微 TLB206，在系统总线 40 上输出的物理地址。相反，如果存储器访问请求直接指定物理地址，则回避 MMU200，经路径 236，直接路径到系统总线 40 上。在一个实施例中，

仅当处理器正在监视模式中操作时，产生直接指定物理地址的存储器访问请求。

如从 MMU200 的先前描述可以想到，以及特别地，从图 43 的描述，主 TLB208 将包含多个描述符，以及对每个描述符，将提供域标记 425 以便识别相应的描述是来自安全页表还是非安全页表。在图 55 的 MMU200 中，示意性地示例说明这些描述符 435 和相关的域标记 425。

当内核 10 发出存储器访问请求时，这将产生用于在系统总线 40 上输出的那个存储器访问请求的物理地址，以及通常高速缓存 38 将执行查找处理来确定由那个地址所指定的数据项是否存储在高速缓存中。只要在高速缓存中出现未命中，即，确定经受访问请求的数据项未存储在高速缓存中，通过高速缓存启动行填充过程以便从外部存储器 56 检索包括存储器访问请求的主题的数据项的数据行。特别地，高速缓存经 EBI42，将行填充请求到设备总线 70 的控制总线 2630 上，以及将起始地址输出到地址总线 2620 上。另外，在路径 2632 上，将 HPROT 信号输出到控制总线 2630 上，将包括指定在发出存储器访问请求时，内核的操作模式的域信号。因此，行填充过程能看作通过高速缓存 38 将原始存储器访问请求传播到外部总线上。

由分区校验器 2656 接收该 HPROT 信号，因此，分区校验器将识别当发出存储器访问请求时，从外部存储器 56 请求指定数据的设备（在这种情况下，包含内核 10 和高速缓存 38 的设备）正在安全域还是非安全域中操作。分区校验器 2656 还将访问识别存储器的那些区安全或不安全的分区信息，因此，能确定是否允许设备访问它正请求的数据。因此，能将分区校验器配置成如果断言 HPROT 信号内的域信号（在此也称为 S 位）表示由正在安全操作模式中操作的设备请求访问该数据，仅允许设备访问存储器的安全部分。

如果分区校验器确定不允许内核 10 访问所请求的数据，例如，因为 HPROT 信号表示内核正在不安全操作模式中操作，但行填充请求正试图从位于存储器的安全区内的外部存储器检索数据，那么分区校验器 2656 将中止信号发送到控制总线 2630 上，在路径 2636 上，将其传递回 EBI42，以及从此回到高速缓存 38，产生在路径 2670 上发送到内核 10 的中止信号。然而，如果分区校验器 2656 确定允许访问，那么将输出表示从外部存储器检索的数据是安全数据还是非安全数据的 S

标记信号，以及经路径 2634，使该 S 标记信号传递回 EBI42，以及从此返回到高速缓存 38 以允许设置与行填充过程的主题，高速缓存行 2600 有关的标记 2602。

同时，控制逻辑 2650 将授权外部存储器 56 来输出所请求的行填充数据，在路径 2680 上，经 EBI42 将该数据传回到高速缓存 38，用来存储在相关高速缓存行 2600 中。因此，作为该过程的结果，用来自外部存储器 56 的数据项填充高速缓存 38 内的选定的高速缓存行，那些数据项包括为来自内核 10 的原始存储器访问请求的主题的数据项。然后，使来自内核的存储器访问请求的主题的数据项从高速缓存 38 返回到内核，或者在路径 2660 上，直接从 EBI 提供到内核 10。

在优选实施例中由于根据上述行填充过程，将产生在高速缓存中原始存储数据，将基于由分区校验器 2656 提供的值，设置与那个高速缓存行有关的标记 2602，然后由高速缓存 38 使用该标记以便直接控制对那个高速缓存行 2600 中的数据项的任何后续访问。因此，如果内核 10 顺序地发出在高速缓存 38 的特定高速缓存行 2600 中的命中的存储器访问请求时，高速缓存 38 将预览相关标记 2602 的值，并将该值与内核 10 的操作的当前模式进行比较。在优选实施例中，用由 CP15 域状态寄存器内的监视模式设置的域位表示内核 10 的当前操作模式。因此，能将高速缓存 38 配置成当处理器内核 10 正在安全操作模式中操作时，仅允许相应标记 2602 表示的高速缓存行中的数据项是由处理器内核 10 访问的安全数据。当内核正在非安全模式中时，内核访问高速缓存 38 内的安全数据的任何尝试将导致高速缓存 38 在路径 2670 上产生中止信号。

能用各种方式设置 TCM36。在一个实施例中，能将其设置成象高速缓存一样操作，以及在那个实施例中，用与高速缓存 38 相同的方式，将配置成包括多个行 2610，每个行具有与之有关的标记 2612。然后，用与参考高速缓存 38 所述的完全相同的方式，管理对 TCM36 的访问，以及任何 TCM 未命中产生正执行的行填充过程，作为其结果，使数据恢复到特定行 2610，以及分区校验器 2656 将生成所需的 S 标记值，用于存储在与那个行 2610 有关的标记 2612 中。

在一可选的实施例中，能将 TCM36 设置成外部存储器 56 的扩展，以及用来存储通常由处理器使用的数据，因为经系统总线访问 TCM 通

常快于访问外部存储器。在该实施例中，TCM36 将不使用标记 2612，相反，将使用不同机制来控制对 TCM 的访问。特别地，如前所述，在这些实施例中，提供当正在特许安全模式中操作时可设置的控制标记，表示仅当正在特许安全模式中执行时，可由处理器控制还是当正在至少一个非安全模式中执行时，可由处理器控制紧密耦合存储器。通过安全操作系统设置控制标记，以及实际上，定义 TCM 受特许安全模式还是受非安全模式控制。因此，能定义的一种结构是仅当处理器正在特许安全操作模式中操作时，控制 TCM。在这些实施例中，对 TCM 控制寄存器尝试的任何非安全访问将导致进入未定义指令异常。

在一可选的结构中，当正在非安全操作模式中操作时，能由处理器控制 TCM。在这些实施例中，仅由非安全应用程序使用 TCM。不将安全数据存储到 TCM 或从 TCM 加载安全数据。因此，当执行安全访问时，在 TCM 内不执行查看该地址是否与 TCM 地址范围匹配的查找。

图 57 是示例说明当在处理器内核 10 上操作的非安全程序生成虚拟地址时，由图 55 的装置执行的处理的流程图（步骤 2700）。首先，在步骤 2705，在微 TLB206 内执行查找，以及如果该结果命中，那么微 TLB 在步骤 2730 校验访问许可。参考图 56，该过程能看作由访问许可逻辑 202 执行。

如果在步骤 2705，在微 TLB 查找中出现未命中，那么在其中存储的非安全描述符中，在主 TLB208 中执行查找（步骤 2710）。如果这导致未命中，那么在步骤 2715 执行页表行程过程（已经参考图 37 描述过）在步骤 2720 后，确定主 TLB 包含有效标记的非安全描述符。如果在步骤 2710 查找产生命中，那么过程直接进入步骤 2720。

此后，在步骤 2725，通过包含物理地址的描述符部分，加载微 TLB，然后在步骤 2730，微 TLB 校验访问许可。

如果在步骤 2730，确定有违反访问许可，那么过程进入步骤 2740，其中在路径 230 上向处理器内核发出中止信号（与图 55 中所示的路径 2670 类似）。然而，假定未检测到违反，那么在步骤 2745，确定访问是否与可高速缓存的数据项有关。如果不，那么在步骤 2790 启动外部访问以便尝试从外部存储器 56 检索数据项。在步骤 2795，分区校验器 25656 将确定是否有安全分区违反，即，如果处理器内核 10 正在非安全模式中操作时，尝试访问安全存储器中的数据项，以及如果检测

到违反，那么分区校验器 5656 将在步骤 5775 生成中止信号。然而，假定没有安全分区违反，那么处理器进入步骤 2785，在步骤 2785 发生数据访问。

如果在步骤 2745，确定正请求的数据项能高速缓存，那么在步骤 2750，在高速缓存内执行高速缓存查找，以及如果检测到命中，那么在步骤 2755，高速缓存确定是否有安全行标记违反。因此，在该阶段，高速缓存将浏览与包含数据项的高速缓存行有关的标记 2602 的值，以及将那个标记的值与内核 10 的操作模式进行比较以便确定内核是否有权访问所请求的数据项。如果检测到安全行标记违反，那么过程进入步骤 2760，其中由高速缓存 38 生成安全违反故障中止信号并在路径 2670 上发送到内核 10。然而，假定在步骤 2755 没有检测到安全行标记违反，那么在步骤 2785 执行数据访问。

如果当在步骤 2750 执行高速缓存查找时，高速缓存未命中存在，那么在步骤 2765 启动高速缓存行填充。在步骤 2770，分区校验器 2656 检测是否有安全分区违反，以及如果是，在步骤 2775 发出中止信号。然而，假定未检测到安全分区违反，那么在步骤 2780，高速缓存行填充处理，导致在步骤 2785 结束数据访问。

如图 57 所示，在 MMU 内执行步骤 2705、2710、2715、2720、2725、2730 和 2735，以及由高速缓存执行步骤 2745、2750、2755、2765、2780 和 2790，以及由分区校验器执行步骤 2770 和步骤 2795。

图 58 是表示在内核上执行的安全程序生成虚拟地址的情况下执行的类似过程的流程图（步骤 2800）。通过将图 58 和图 57 比较，将意识到在 MMU 内执行的步骤 2805 至 2835 与先前参考图 57 所述的步骤 2705 至 2735 类似。唯一区别在步骤 2710，其中根据存储在主 TLB 内的任何安全描述符，执行在主 TLB 内执行的查找，作为结果，在步骤 2820，主 TLB 包含有效标记的安全描述符。

在高速缓存内，高速缓存不再需要查找任何安全行标记违反，因为在参考图 58 所示的实施例中，假定安全程序能访问安全数据和非安全数据。因此，如果在步骤 2850，在高速缓存查找期间生成命中，那么处理直接进入数据访问步骤 2885。

类似地，在要求外部存储器的外部访问的情况下（即，在步骤 3865 或 2890），分区校验器不需要执行分区校验，同样因为假定安全程序

能访问安全数据或非安全数据。

在高速缓存内执行的步骤 2845、2850、2865、2880 和 2890 与参考图 57 所述的步骤 2745、2750、2765、2780 和 2790 类似。)

图 59 图示了在处理器上运行的不同模式和应用程序。虚线表示根据本发明的实施例，在监视处理器期间，如何能将不同模式和/或应用程序与另一个分开和隔离。

监视处理器来定位可能的故障以及发现为何应用程序不按预期的那样执行，极为有用，而许多处理器提供这些功能。能用各种方法，包括调试和跟踪功能执行监视。

在根据当前技术的处理器中，调试能在包括暂停调试模式和监视调试模式的几种模式中操作。这些模式是插入模式并且导致此时运行的程序中止。在暂停调试模式中，当断点或观察点出现时，停止内核以及与系统的其余部分隔离以及内核进入调试状态。在进入时，停止内核，刷新流水线以及不预提取指令。冻结 PC 以及忽略任何中断（IRQ 和 FIQ）。然后可以检查内核内部状态（经 JTAG 串行接口）和存储器系统状态。这种状态插入执行，因为可以修改当前模式，修改寄存器内容等等。一旦终止调试，通过用重启指令，通过调试 TAP（测试访问端口）扫描，内核从调试状态退出。然后程序恢复执行。

在监视调试模式中，断点或观察点引发内核进入中止模式，分别进行预提取或数据中止矢量。在这种情况下，当在暂停调试模式中时，内核仍然在处于功能模式，并且不停止。中止处理程序与调试程序应用程序通信以便访问处理器和处理器状态或转储存储器。调试监视程序连接在调试硬件和软件调试程序之间。如果设置调试状态和控制寄存器 DSCR 的位 11（见稍后），能禁止中断（FIQ 和 IRQ）。在监视调试模式中，在数据中止和预提取中止时，禁止矢量俘获以避免根据为监视调试模式生成的中止结果，使处理器进入不可恢复状态。应注意到监视调试模式是一种调试模式以及不与为监督安全区域和非安全区域间的切换的模式的处理器的监视模式关联。

调试能提供在某一时刻处理器的状态的抽点打印。通过注意在接收调试启动请求时，不同寄存器中的值来完成。这些值记录在扫描链（图 67 的 541、544）上，然后，使用 JTAG 控制器（图 1 的 18），顺序地输出它们。

监视内核的一种可选方法是通过跟踪。跟踪不是插入以及当内核继续操作时，记录续状态。跟踪运行在图 1 的内嵌式跟踪宏单元 (ETM), 22、26。ETM 具有跟踪端口，通过该端口，输出跟踪信息，然后通过外部跟踪端口分析器分析。

本技术的实施例的处理器在两个单独的域中操作，在所述的实施例中，这些域包括安全域和非安全域。然而，为监视功能目的，对技术人员来说，这些域可以是数据不应当泄漏的任何两个域。本发明的实施例涉及防止在两个域间泄漏数据以及通常允许访问整个系统的监视功能，诸如调试和跟踪是域间数据泄漏的潜在源。

在上述给出的安全和非安全域或区域的例子中，安全数据一定不能用于非安全区域。另外，如果允许调试，在安全区域中，限制或隐藏安全区域内的一些数据是有利的。图 59 中的虚线表示分段数据访问和提供不同粒度级的可能方法的一些例子。在图 59 中，用方框 500 表示监视模式以及是所有模式的最安全模式，以及控制安全和非安全区域间的切换。在监视模式 500 下，有监督模式，这包括安全监督模式 510 和非安全监督模式 520。然后，有具有应用程序 522 和 524 的非安全用户模式和具有应用程序 512、514 和 516 的安全用户模式。能将监视模式（调试和跟踪）控制到仅监视非安全模式（虚拟 501 的左边）。另外，可以允许监视非安全域或区域以及安全用户模式（501 的左边以及位于 502 下的 501 的右边）。在另一实施例中，可以允许在安全用户域中运行的非安全区域和某些应用程序，在这种情况下，通过虚线 503 的另外的分段发生。这些分隔有助于防止在运行不同应用程序的不同用户间泄漏安全数据。在一些受控的情况下，可以允许监视整个系统。根据所要求的粒度，内核的下述部分需要在监视功能期间控制它们的访问。

在调试事件上，能设置四个寄存器：指令故障状态寄存器 (IFSR)、数据故障状态寄存器 (DFSR)、故障地址寄存器 (FAR) 和指令故障地址寄存器 (IFAR)。在一些实施例中，当从安全区域进入非安全区域时，应当刷新这些寄存器以避免数据的任何泄漏。

PC 样本寄存器： 调试 TAP 能通过扫描链 7 访问 PC。当在安全区域中调试时，根据在安全区域中选择的调试粒度，能屏蔽那个值。当内核正在安全区域中操作时，非安全区域，或非安全区域加上安全用户

应用程序不能获得 PC 的任何值很重要。

TLB 项：使用 CP15，可以读取微 TLB 项以及读取和写入主 TLB 项。还能控制主 TLB 和微 LTB 加载和匹配。必须严格控制这种操作，特别是如果安全线程识别调试需要 MMU/MPU 的帮助时。

性能监视控制寄存器：性能控制寄存器提供有关高速缓存未命中、微 TLB 未命中、外部存储器请求、所执行的转移指令等等的信息。非安全区域不应当访问该数据，即使在调试状态中。计数器应当在安全区域中操作，即使在安全区域中禁止调试。

在高速缓存系统中调试：在高速缓存系统中，调试必须是非插入模式。保持高速缓存和外部存储器间的一致很重要。使用 CP15，能使高速缓存无效，或能使高速缓存在所有区中直写。在任何情况下，在调试中允许修改高速缓存行为能是安全性弱点以及应当控制。

字节顺序（endianness）：不应当允许能访问调试的非安全区域或安全用户应用程序改变字节顺序。改变字节顺序能使安全核心误操作。根据粒度，在调试中，能禁止字节顺序访问。

在启动监视功能时，能控制内核部分的监视功能的访问。用各种方法启动调试和跟踪。仅允许在某些条件下初始化，本技术的实施例将监视功能的访问控制到内核的某些安全部分。

本技术的实施例试图通过下述粒度，将项限制到监视功能：

通过单独控制插入和可观察（跟踪）调试；

通过仅在安全用户模式或在整个安全区域中，允许调试项；

通过仅在安全用户模式中允许调试，此外，考虑线程 ID（运行的应用程序）。

为控制启动监视功能，知道如何能启动功能是很重要的。图 60 表示示例说明启动监视功能的可能方式、启动的监视功能的类型以及能编程启动指令的方式的表格。

通常，经软件或经硬件，即经 JTAG 控制器，输入这些监视指令。为了控制启动监视功能，使用控制值。这些包括根据条件而定的允许位，因此，如果存在特定条件，如果设置允许位，仅允许启动监视。这些位存储在位于 ICE530（见图 67）中的安全寄存器 CP14（调试和状态控制寄存器，DSCR）上。

在优选实施例中，存在允许/禁止插入和可观察调试的四个位，这

些包括安全调试允许位、安全跟踪允许位、安全用户模式允许位和安全线程识别允许位。这些控制值用来提供用于监视功能的可控制粒度，同样地，能帮助防止从特定域泄漏数据。图 61 提供这些位的概述以及如何访问它们。

这些控制位保存在安全域中的寄存器中，以及访问该寄存器限定到三种可能性。经 ARM 协处理器 MRC/MCR 指令，提供软件访问，以及仅从安全监督模式允许这些。另外，通过使用验证码，从任何其他模式提供软件访问。另一替代方案更多地涉及硬件访问以及调用经 JTAG 上的输入端口所写的指令。除用来输入与监视功能的可用性有关的控制值外，还能使用该输入端口来输入与处理器的其他功能有关的控制值。

下面提供与扫描链和 JTAG 有关的进一步描述。

寄存器逻辑单元

每个集成电路 (IC) 由两种逻辑组成：

- 组合逻辑单元，象 AND、OR、INV 门。这些门或这些门的组合用来根据一个或多个输入信号，计算布尔表达式。
- 寄存器逻辑单元：象锁存器、触发器。这些单元用来存储任何信号值。图 62 表示正沿触发的触发器视图。

当在时钟信号 (CK) 上产生正沿事件时，输出 (Q) 接收输入 (D) 的值，否则输出 (Q) 将其值保存在存储器中。

扫描链单元

为测试或调试目的，期望忽略寄存器逻辑单元的功能访问以及直接访问寄存器逻辑单元的内容。寄存器单元集成在图 63 所示的扫描链单元中。

在功能模式中，清除 (SE)（扫描允许）以及寄存器单元工作为单一寄存器单元。在测试或调试模式中，设置 SE 以及输入数据能来自 SI 输入（扫描输入），而不是 D 输入。

扫描链

使所有扫描链单元束缚在扫描链中，如图 64 所示。

在功能模式中，清除 SE 以及能正常访问所有寄存器单元并与电路的其他逻辑交互作用。在测试或调试模式中，设置 SE 以及所有寄存器束缚在扫描链的相互之间。根据每个时钟周期的步调信号，数据能来

自第一扫描链单元以及能移过任何其他扫描链单元。能移出数据以便查看寄存器的内容。

TAP 控制器

调试 TAP 控制器用来处理多个扫描链。TAP 控制器能选择特定的扫描链：它将“扫描入”和“扫描出”信号连接到那个特定的扫描链。然后，能将数据扫描到链中、移出或扫描出。外部由 JTAG 端口拾音器外部控制 TAP 控制器。图 65 示意性地示例说明 TAP 控制器。

JTAG 有选择禁止扫描链单元

出于安全原因，一些寄存器不能由扫描链访问，即使是在调试或测试模式中。新输入所谓的 JADI（JTAG 访问禁止）能允许从整个扫描链动态和静止地移出扫描链单元，而不修改集成电路的扫描链结构。

图 66A 和 66B 示意性地表示该输入。

如果 JADI 无效 ($JADI=0$)，不管是在功能还是测试或调试模式中，扫描链正常工作。如果 JADI 有效 ($JADI=1$)，以及如果正在测试或调试模式中，可以从扫描链结构移出一些扫描链单元（由设计者选择）。为保持相同扫描链单元号，JTAG 有选择允许扫描链单元使用旁路寄存器。注意扫描出 (S0) 以及扫描链单元输出 (Q) 现在不同。

图 67 示意性地图示了包括 JTAG 部分的处理器。在正常操作中，指令存储器 550 与内核通信以及在某些环境下，还能与寄存器 CP14 通信，以及复位控制值。这通常只在安全监督模式下才是允许的。

当启动调试时，经调试 TAP580 输入指令以及它是控制内核的那些指令。调试中的内核在逐个步骤模式中运行。调试 TAP 经内核访问 CP14（由在示为 JADI 管脚的 JSDAEN 管脚上输入，图 45 中的 JTAG 访问禁止输入的访问控制信号而定）并以这种方式复位控制值。

通过访问控制信号 JSDAEN 控制经调试 TAP580 访问 CP14 寄存器。这配置成为访问以及特别是允许写入访问，JSDAEN 必须设置成高。在板级阶段，当正验证整个处理器时，将 JSDAEN 设置成高以及在整个系统上允许调试。一旦校验该系统，能将 JSDAEN 管脚连接到地，这意味着经调试 TAP580，访问允许在安全模式中调试的控制值现在不可用。通常，生产模式中的处理器具有连接到地的 JSDAEN。经指令存储器 550，经软件路由，访问控制值仅可用。经这一路由访问限定到安全监督模式或假定给出验证码的另一模式（见图 68）。

应注意到根据缺省情况，调试（插入和可观察-跟踪）仅可用在非安全区域中。为允许它们用在安全区域中，需要设置控制值允许位。

其优点在于总是能由用户启动调试以便在非安全区域中运行。因此，尽管在调试中，访问安全区域不总是可用于用户，在许多情况下，这不是问题，因为限制访问该区域以及在使得可用之前，在板级完全验证安全区域。因此，预见在许多情况下，调试安全区域是不必要的。如果必要的话，安全监督员仍然经写入 CP14 的软件路由启动调试。

图 68 示意性图示了调试启动的控制。在该图中，内核部分 600 包括存储表示系统是否是安全区域的安全状态位 S 的存储元件 601（可以是如前所述的 CP15 寄存器。内核 600 还包括由表示处理器正在运行的模式，例如用户模式的位组成的寄存器 602，以及提供识别当前正在内核上运行的应用程序或线程的上下文标识符的寄存器 603。

当断点到达将存储在寄存器 611 上的断点与存储在寄存器 612 上的内核的址进行比较的比较器 610 时，将信号发送到控制逻辑 620。控制逻辑 620 查看安全状态 S、模式 602 和线程（上下文标识符）603 以及将它与存储在寄存器 CP14 上的控制值和条件指示符进行比较。如果系统不在安全区域中操作，那么将在 630 输出“进入调试”信号。然而，如果系统正在安全区域中操作，控制逻辑 620 将查看模式 602，以及如果在用户模式中，将校验以查看是否设置用户模式允许和调试允许位。如果它们是，那么假定还没有初始化线程识别位，将初始化调试。上文示例说明控制值的分层属性。

在图 68 中还示意地图示了监视控制的线程识别部分以及如何仅能从安全监督模式（在该实施例中，处理器位于产品级和 JSDAEN 连接到地），改变存储在寄存器 CP14 中的控制值。从安全用户模式，使用验证码，能进入安全监督模式，然后，能在 CP14 中设置控制值。

假定线程比较器 640 表示调试可用于那个线程，当地址比较器 610 表示已经到达断点时，控制逻辑 620 输出“进入调试”信号。这假定能在 CP14 中设置线程识别初始化位。如果在断点后，设置线程识别初始化位，如果地址和上下文标识符与在断点和可允许线程指示器中表示的那些匹配，能仅进入调试或跟踪。在启动监视功能后，将仅继续俘获诊断数据，同时由比较器 640 将上下文标识符标识为允许线程。当上下文标识符表示正在运行的应用程序不是允许的应用程序，那么

抑制俘获诊断数据。

应注意到在优选实施例中，在粒度内有一些等级。实际上，安全调试或跟踪允许位位于顶部，在安全用户模式允许位后以及最后来到安全线程识别允许位。这在图 69A 和 69B 中示例说明（见下文）。

保持在“调试和状态控制”寄存器（CP14）中的控制值根据域、模式和执行线程，控制安全调试粒度。其在安全监督模式的上部分。一旦构成“调试和状态控制”寄存器 CP14，它直到安全监督模式来编程相应的断点、观察点等等来使内核进入调试状态。

图 69A 图示了用于插入调试的安全调试粒度的概述。用灰色表示复位时的缺省值。

对有关可观察测试的调试粒度也是一样的。图 69B 表示在这种情况下，安全调试粒度的概述，其中也用灰色表示复位时的缺省值。

注意安全用户模式调试允许位和安全线程识别调试允许位通常用于插入和可观察调试。

线程识别初始化位存储在寄存器 CP14 中，以及表示是否需要由应用程序的粒度。如果已经初始化线程识别位，控制逻辑将进一步校验应用标识符或线程 603 是在线程识别控制位中表示的，如果是，那么初始化调试。如果未设置用户模式或调试允许位的任何一个或设置线程识别位以及应用程序不是在线程识别控制位中表示的一个，那么将忽略断点以及内核将继续它所执行的处理以及不初始化调试。

除控制监视功能的初始化外，用类似的方式，能控制监视模式期间俘获诊断数据。为执行此操作，内核必须继续考虑控制值，即存储在 CP14 中的允许位以及在监视功能期间，与它们有关的条件。

图 70 示意性地表示运行时的监视功能的粒度。在这种情况下，区域 A 与允许俘获诊断数据的区域有关，以及区域 B 与存储在 CP14 中的控制值表示不可能俘获诊断数据的区域有关。

因此，当调试运行以及程序正在区域 A 中操作时，在调试期间，以逐步方式输出诊断数据。当操作切换到区域 B 时，其中，不允许俘获诊断数据，调试不再以逐步方式进行，相反，其自动处理以及不俘获数据。这继续直到程序的操作再次进入区域 A 为止，然后，再次开始俘获诊断数据以及调试继续以逐步方式运行。

在上述实施例中，如果安全域未被授权，则总是将 SMI 指令看作

原子事件以及抑制俘获诊断数据。

此外，如果设置线程识别初始化位，那么相对于应用程序，在操作期间的监视功能的粒度也发生。

关于可观察调试或跟踪，这是通过 ETM 来实现的，而与调试完全无关。当允许跟踪时，ETM 正常工作以及当禁止它时，ETM 在安全区域，或由所选择的粒度而定的安全区域部分中隐藏跟踪。当不允许时，避免 ETM 俘获和跟踪安全域中的诊断数据的一种方法是当 S 位为高时，停止 ETM。这能通过将 S 位与 ETMPWRDOWN 信号结合来实现，以便当内核进入安全区域时，ETM 值保持在它们的最后值。因此，ETM 应当跟踪 SMI 指令，然后直到内核返回到非安全区域才停止。因此，ETM 仅看见非安全活动性。

下面给出不同监视功能和它们的粒度的一些的概述。

板级的插入调试

在板级，当不连接 JSDAEN 管脚时，在开始任何引导圣诞前，有能力在任何地方允许调试。类似地，如果在安全监督模式中，具有类似的权利。

如果在中止调试模式中初始化调试，能访问所有寄存器（非安全和安全寄存器组），以及能转储整个存储器，除专用于控制调试的位外。

能从任何模式和从任何域进入调试中止模式。能在安全或非安全存储器中设置断点和观察点。在调试状态中，可以经 MCR 指令，通过简单地改变 S 位来进入安全区域。

当安全异常发生时，能进入调试模式时，通过如下新位扩展矢量中断寄存器：

SMI 矢量中断允许

安全数据中止矢量中断允许

安全预提取中止矢量中断允许

安全非定义矢量中断允许

在监视调试模式中，如果任何地方允许调试，即使当在非安全区域中调试 SMI 时，可以在逐步调试中进入安全区域。当在安全域中出现断点时，安全中止处理程序能用来转储安全寄存器组和安全存储器。

安全和非安全区域中的两个中止处理程序向调试应用程序提供它

们的信息以便调试程序窗口（在相关调试控制 PC 上）能表示安全和非安全区域中的寄存器状态。

图 71A 表示当在监视调试模式中构成内核以及在安全区域中允许调试时所发生的事件。图 71B 表示当在监视调试模式中构成内核以及在安全区域中禁止调试时所发生的事件。下面将描述该后一过程。

在产品级的插入调试

在产品级，当连接 JSDAEN 以及使调试限制到非安全区域时，除非安全监督程序确定，那么图 71B 所示的表表示所发生的事件。在这种情况下，SMI 应当总是视为原子指令，以便在进入调试状态前，总是完成安全功能。

进入调试中止模式经受下述限制：

仅在非安全区域中考虑外部调试请求或内部调试请求。如果断言 EDBGREQ (外部调试请求) 同时在安全区域中，一旦终止安全功能和内核返回到非安全区域中，内核进入调试中止模式。

在安全存储器上编程断点或观察点没有影响以及当编程地址匹配时，不停止内核。

矢量中断寄存器（下面给出该详细情况）仅涉及非安全区域异常。先前所述的所有扩展中断允许位没有影响。

只要在中止调试模式中，应用下述限制：

不改变 S 位来强制安全区域输入，除非允许安全调试

如果仅在安全监督模式中允许调试，不改变模式位。

不改变控制安全调试的专用位。

如果加载和执行 SMI（通过系统速度访问），仅当完全执行安全功能时，内核重新进入调试状态。

在监视调试模式中，因为在安全区域中不发生监视，安全中止处理程序不需要支持调试监视程序。在非安全区域中，逐步是可能的，但只要执行 SMI，整个执行安全功能，换句话说，当在所有其他指令上，“步进”和“失步”是可能的，则允许 XWSI 单步。因此，XWSI 视为原子指令。

一旦禁止安全调试，具有下述限制：

在进入监视模式前：

仅在非安全区域中考虑断点和观察点。如果设置 S 位，忽略断点/

观察点。注意还能通过 MCR/MRC (CP14) 访问观察点单元，因为在安全存储器中，断点/观察点无影响，因此不是安全问题。

BKPT 通常用来替换在其上设置断点的指令。这假定用 BKPT 指令覆盖存储器中的这一指令，这种情况仅在非安全模式中是可能的。

矢量中断寄存器仅涉及非安全异常。如前所述的所有扩展的中断允许位没有影响。应当禁止数据中止和预提取中止允许位以避免处理器迫使进入不可恢复状态。

经 JTAG，具有与用于中止模式的相同限制（不能修改 S 位等等）。

一旦在监视模式中（非安全中止模式）

非安全中止处理程序能转储非安全区域以及在安全成组寄存器和安全存储器上没有可见度。

通过原子 SMI 指令执行安全功能

能改变 S 位以强制安全区域项

不能改变模式位，因为仅在安全监督模式中允许调试

注意如果发生外部调试请求 (EDBGRQ)，

在非安全区域中，内核中止当前指令，然后立即进入调试状态（在中止模式中）

在安全区域中，内核中止当前功能以及当已经返回到非安全区域中时，进入调试状态。

新调试需求意味着内核硬件中的改进。必须仔细地控制 S 位，以及出于安全原因，安全位必须不能插入扫描链中。

总的来说，在调试中，能修改模式位，只要在安全监督模式中允许调试。将防止任何人进入安全域中调试以便通过修改系统（修改 TBL 项等待）进入所有安全区域。用那种方式，每个线程能调试其自己的代码，以及仅其自己的代码。必须保持安全核心安全。因此，当内核正在非安全区域中运行时进入调试时，如以前，仅改变模式位。

该技术的实施例使用新矢量中断寄存器。如果该寄存器中的一个位设置成高以及相应的矢量触发，处理器进入调试状态，就象已经在从相关异常矢量提取的指令上设置断点一样。这些位的行为可以根据在调试控制寄存器中的安全区域允许位中的调试值而不同。

新矢量中断寄存器包括下述位：D_s_abort、P_s_abort、S_undef、SMI、FIQ、IRQ、Unaligned、D_abort、P_abort、SWI 和 Undef。

- D_s_abort 位：仅当在安全区域中允许调试时，以及当在中止调试模式中配置调试才设置。在监视调试模式中，该位应当是永不设置位。如果禁止在安全区域中调试，不管其值如何，该位没有影响。
- P_s_abort 位：与 D_s_abort 位相同。
- S_undef 位：仅当在安全区域中允许调试时才设置。如果禁止安全区域中的调试，不管其值如何，该位没有影响。
- SMI 位：应当仅当在安全区域中允许调试时才设置。如果禁止安全区域中的调试，不管其值如何，该位没有影响。
- FIQ、IRQ、Unaligned、D_abort、P_abort、SWI 和 Undef 位：对应于非安全异常，因此，即使禁止安全区域中的调试，它们也是有效的。注意在监视模式中，不应当断言 D_abort 和 P_abort 为高。
- Reset 位：当在复位发生时进入安全区域时，仅当允许安全区域中的调试时，该位才有效，否则没有影响。

尽管在此已经描述了本发明的具体实施例，很显然本发明不限于此，以及可以在本发明的范围内进行许多改进和增加。例如，根据独立权利要求的特征，能进行下述从属权利要求的特征的各种组合而不背离本发明的范围。

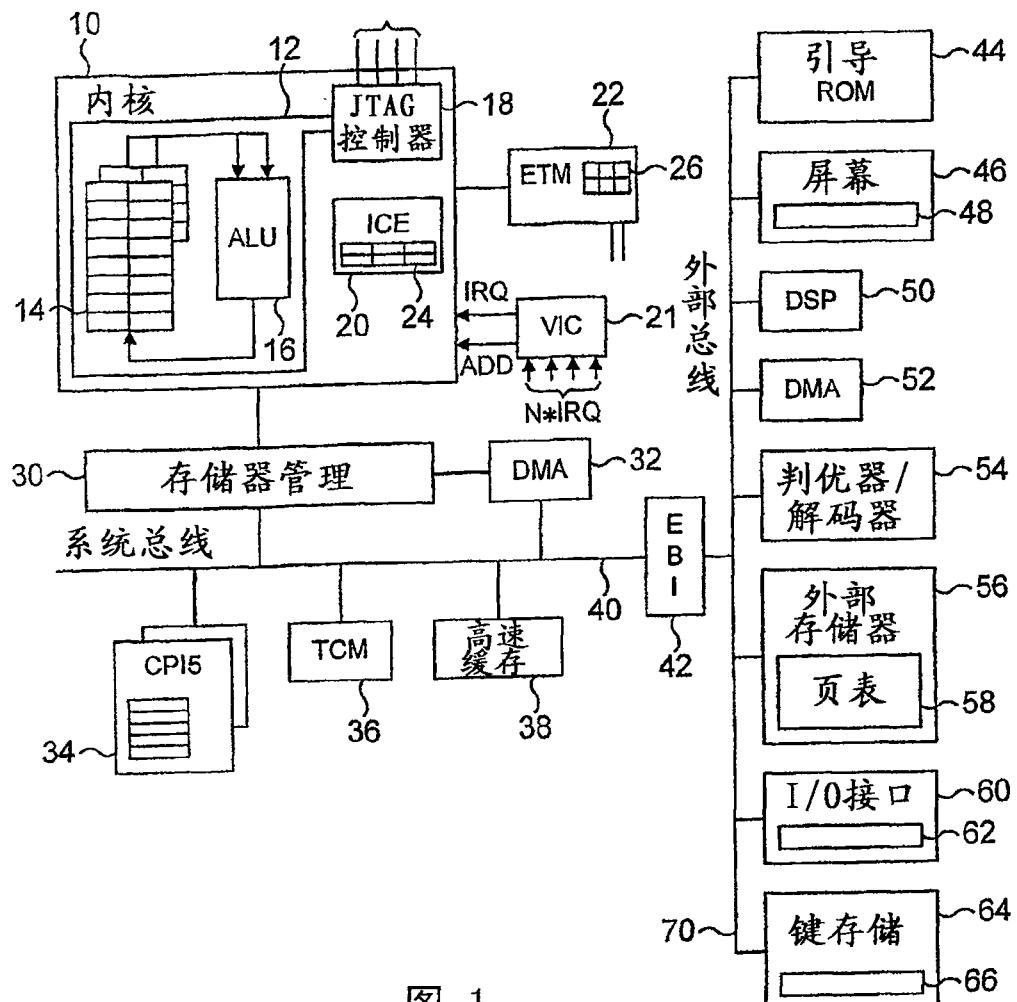


图 1

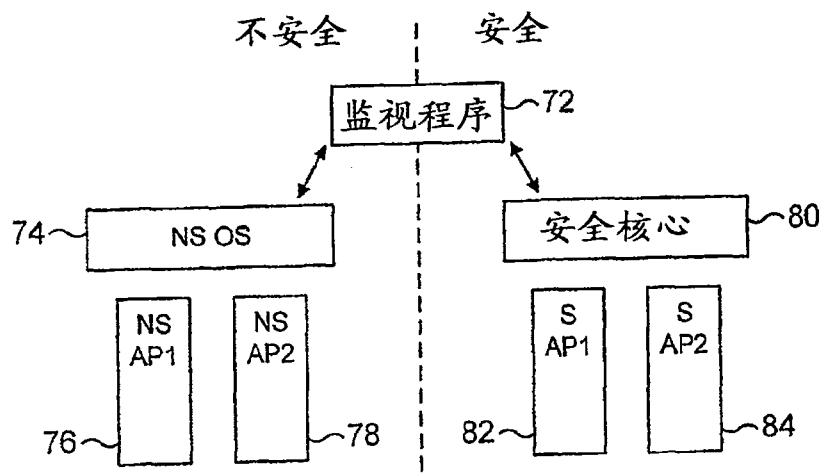


图 2

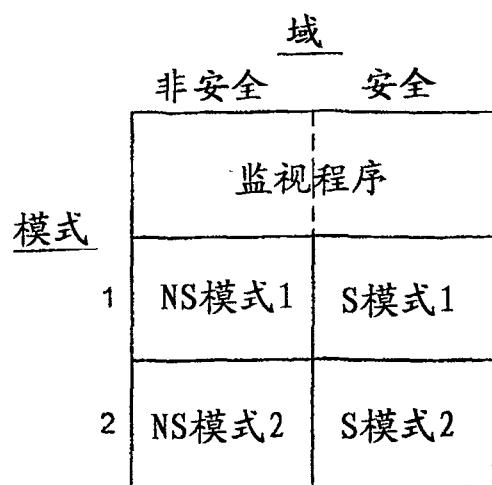


图 3

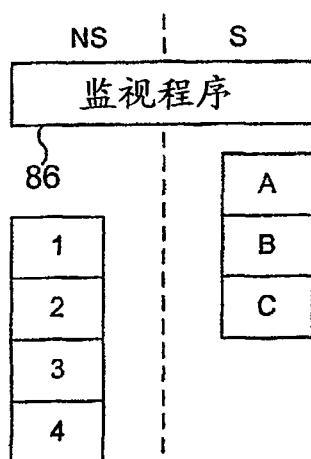


图 4

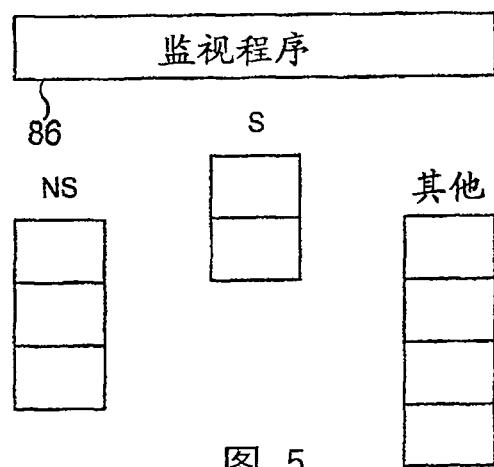


图 5

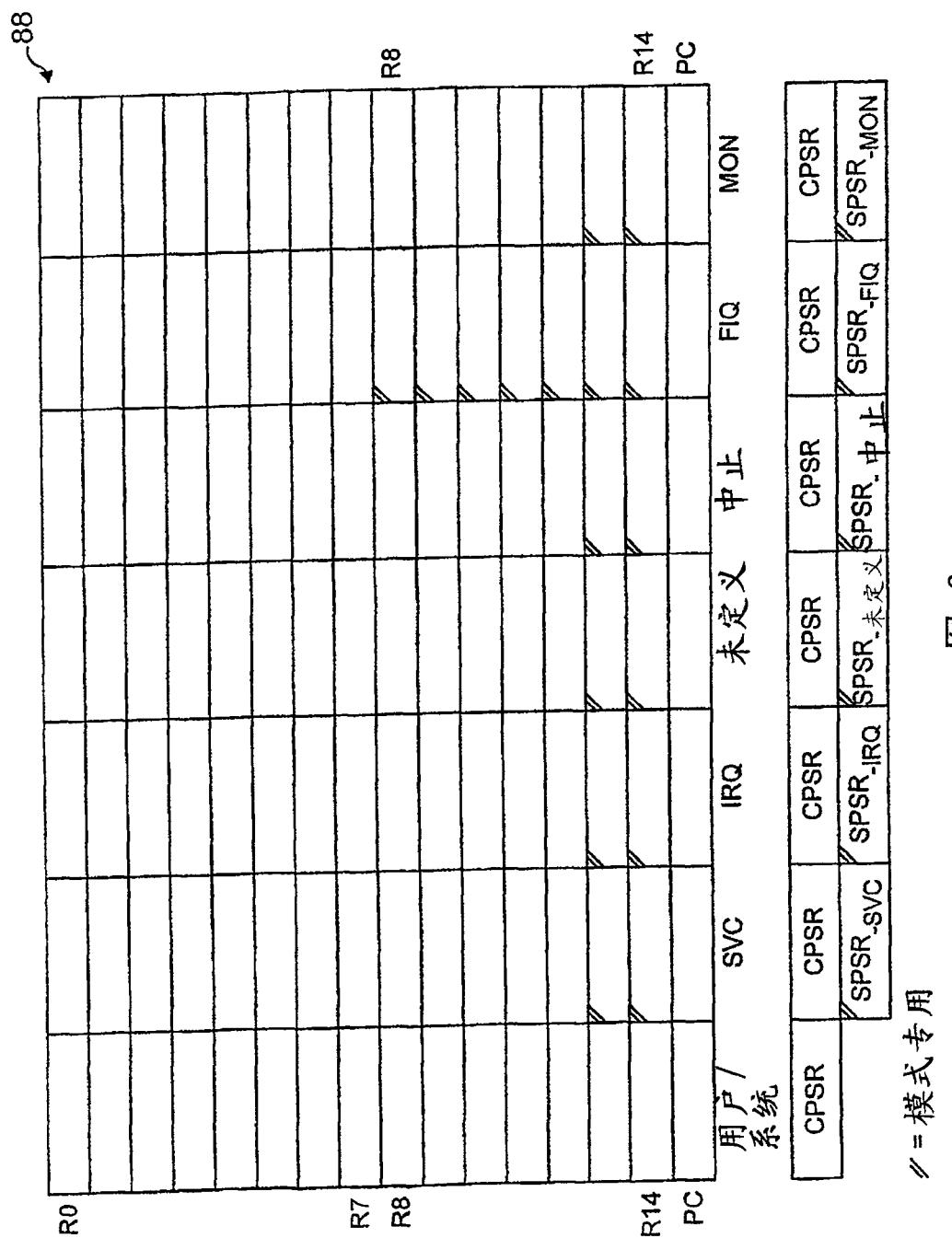


图 6

模式专用

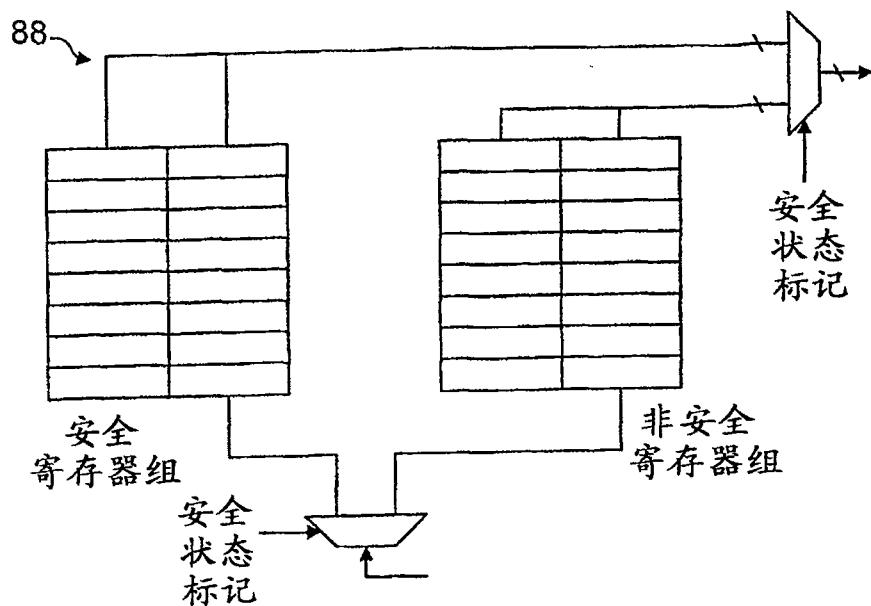


图 7

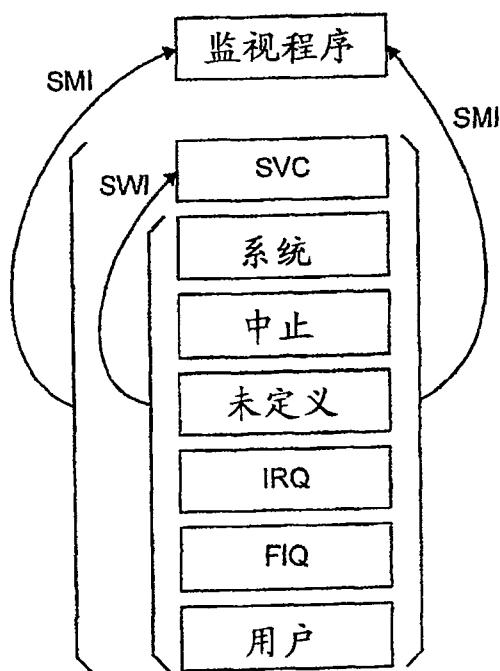


图 8

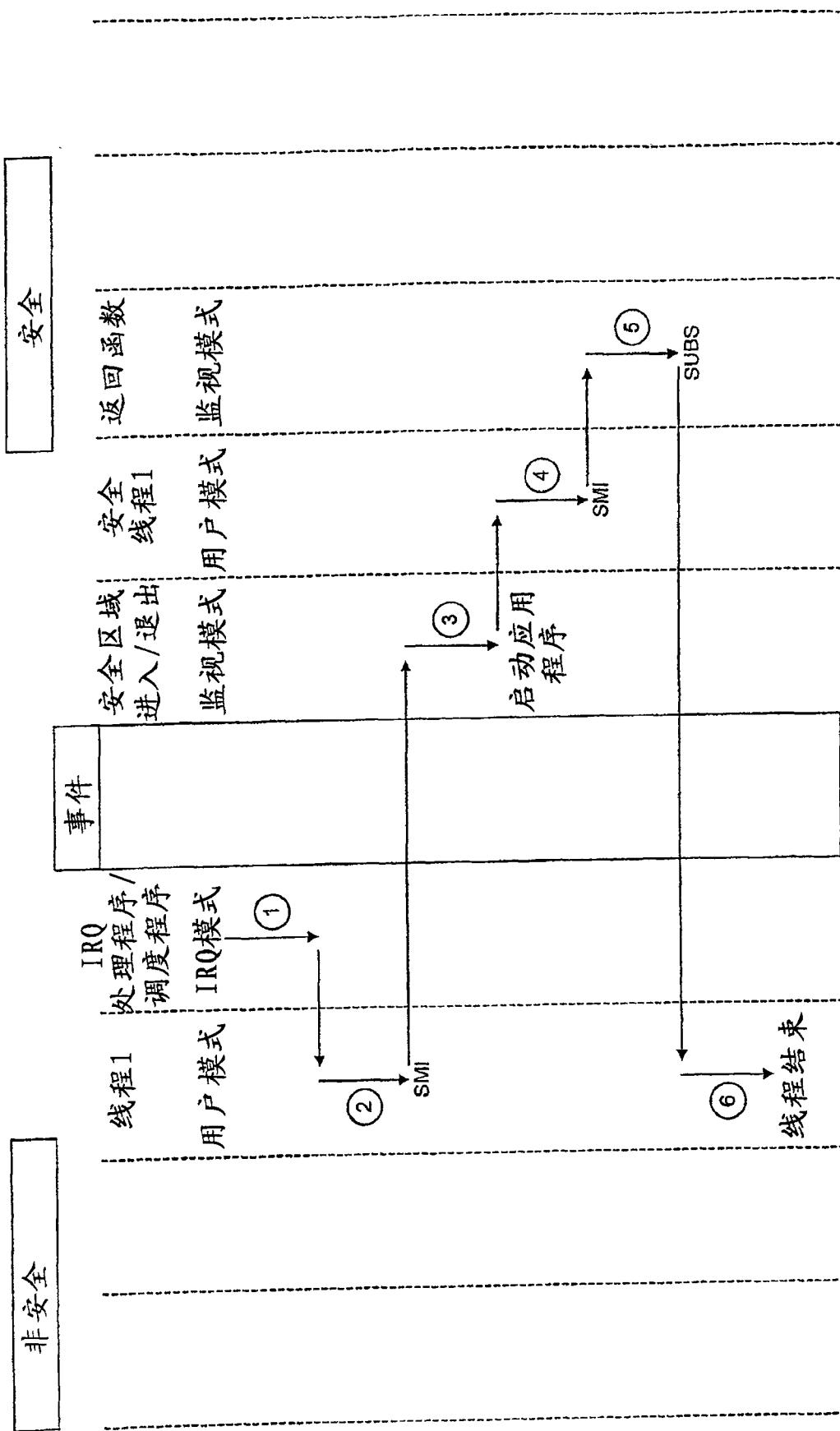


图 9

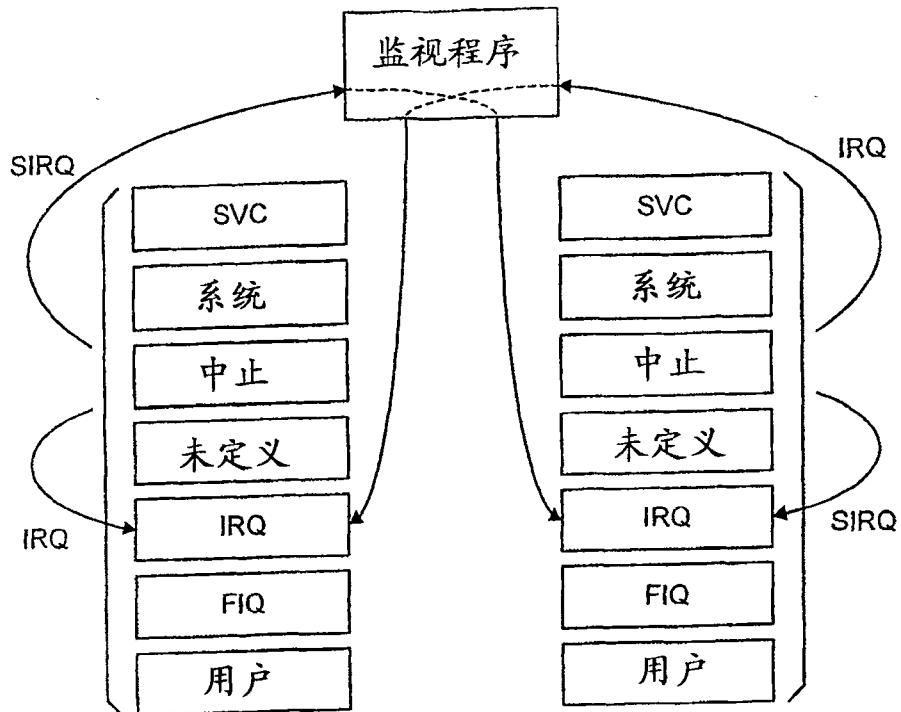


图 10

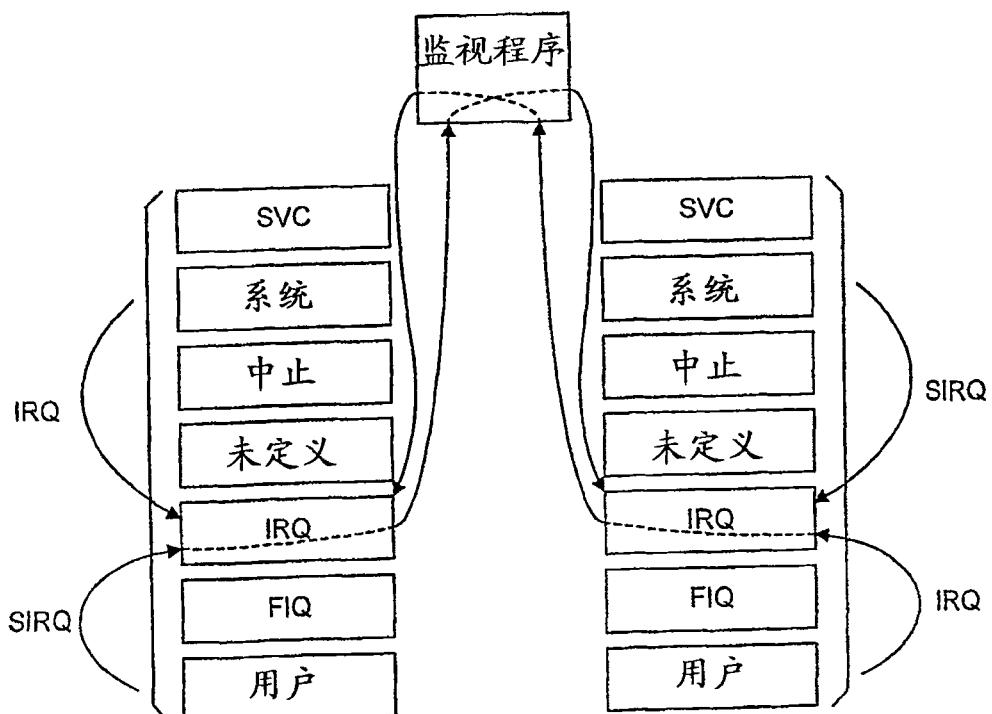


图 12

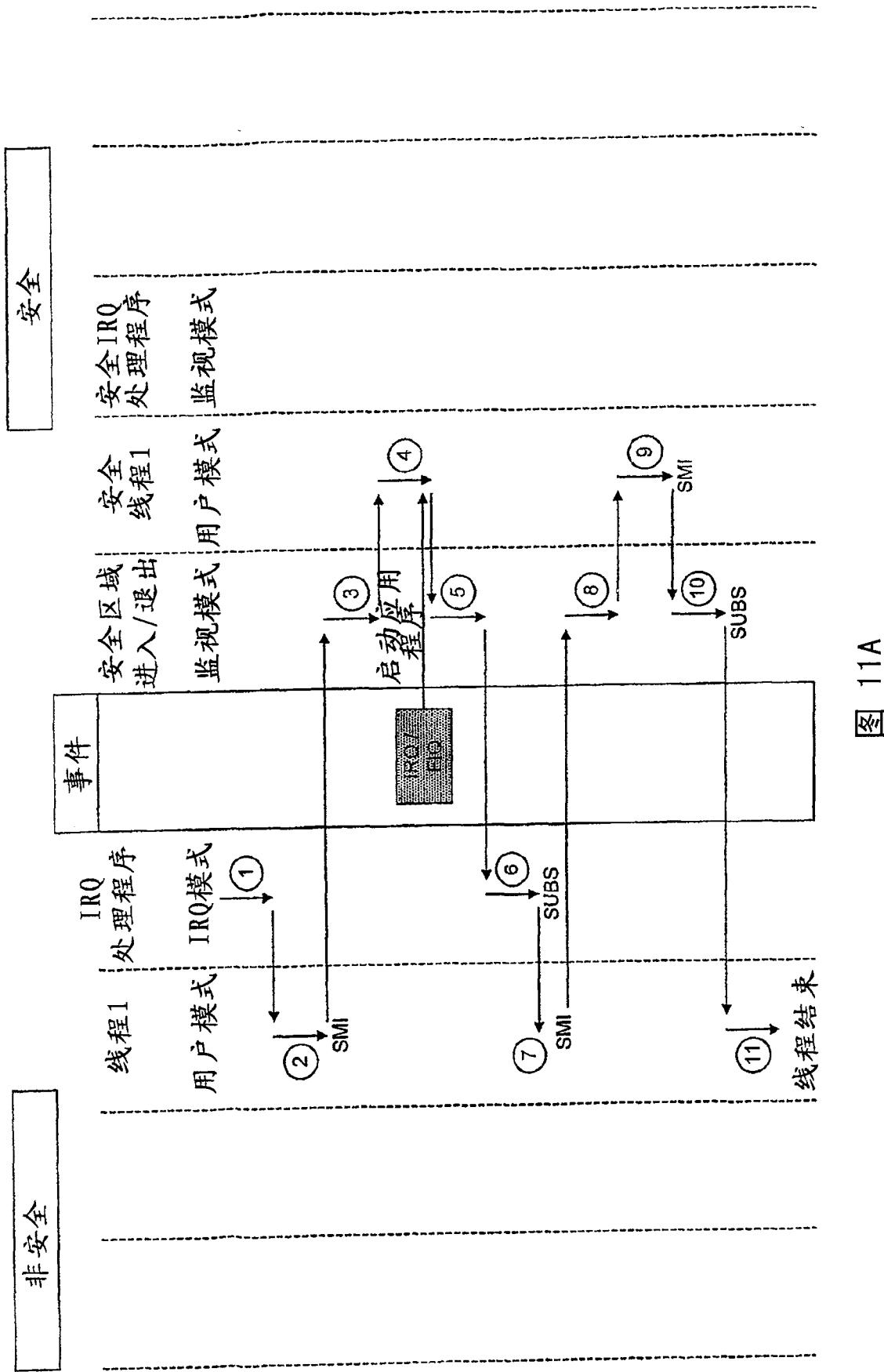


图 11A

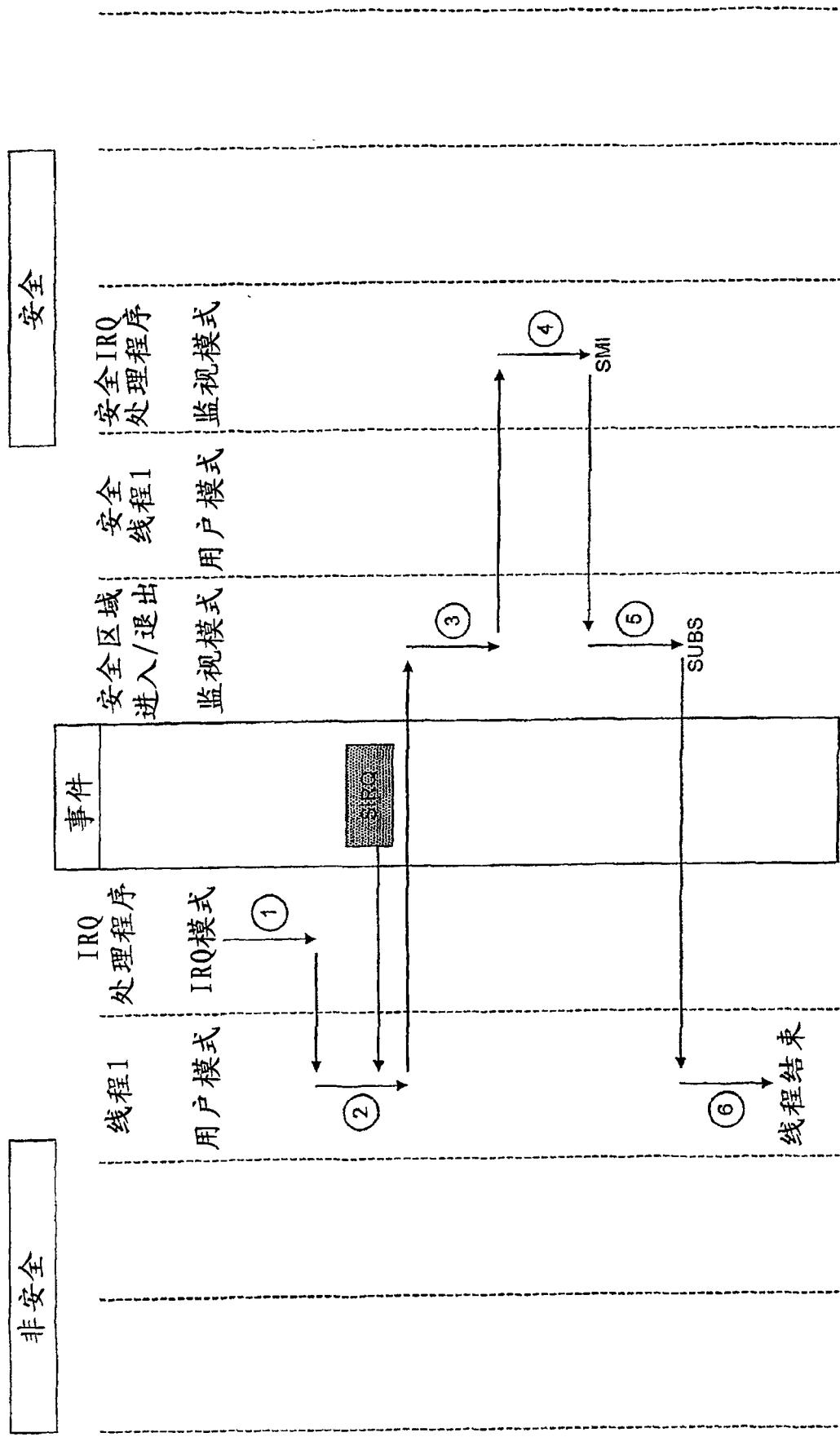


图 11B

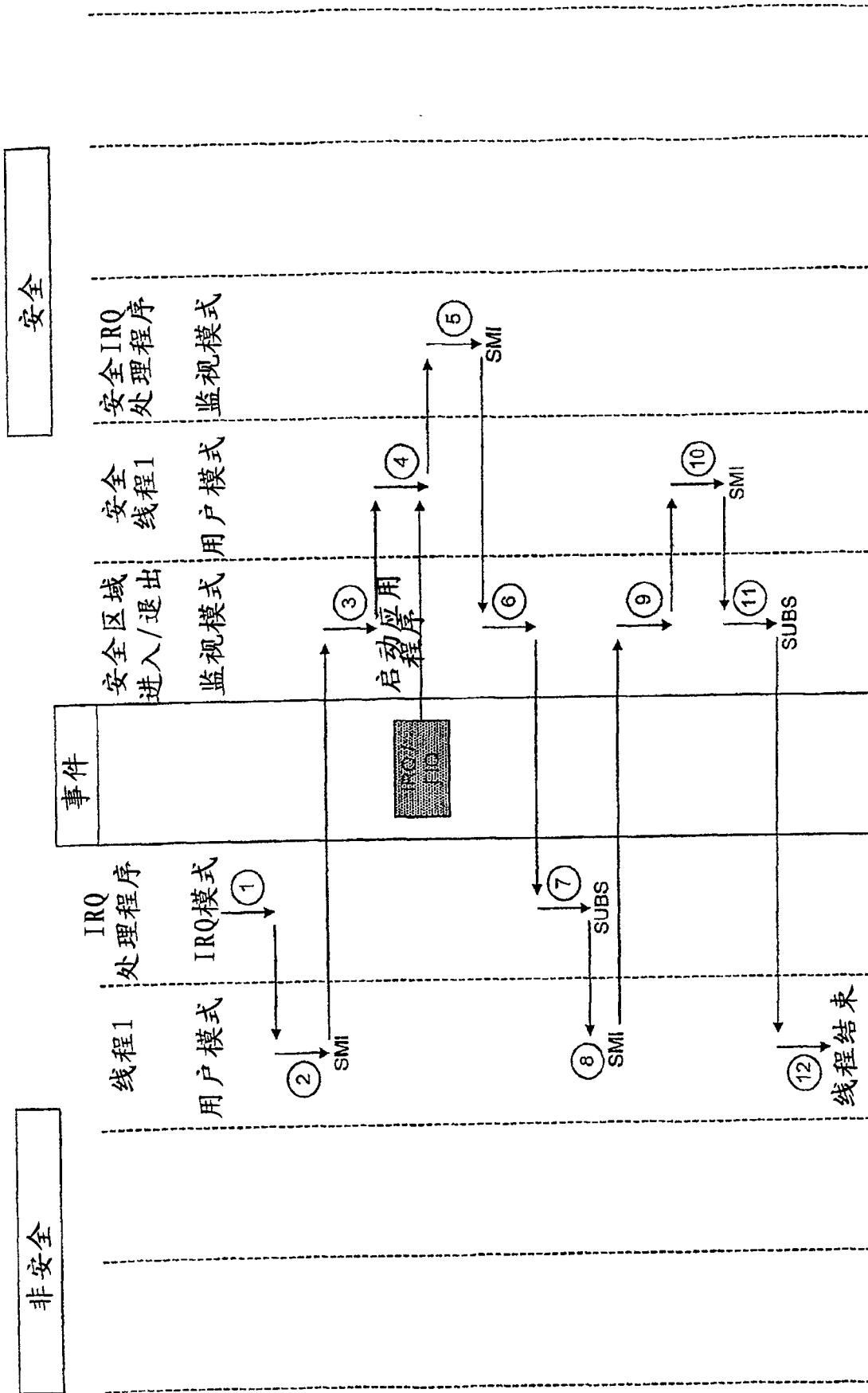


图 13A

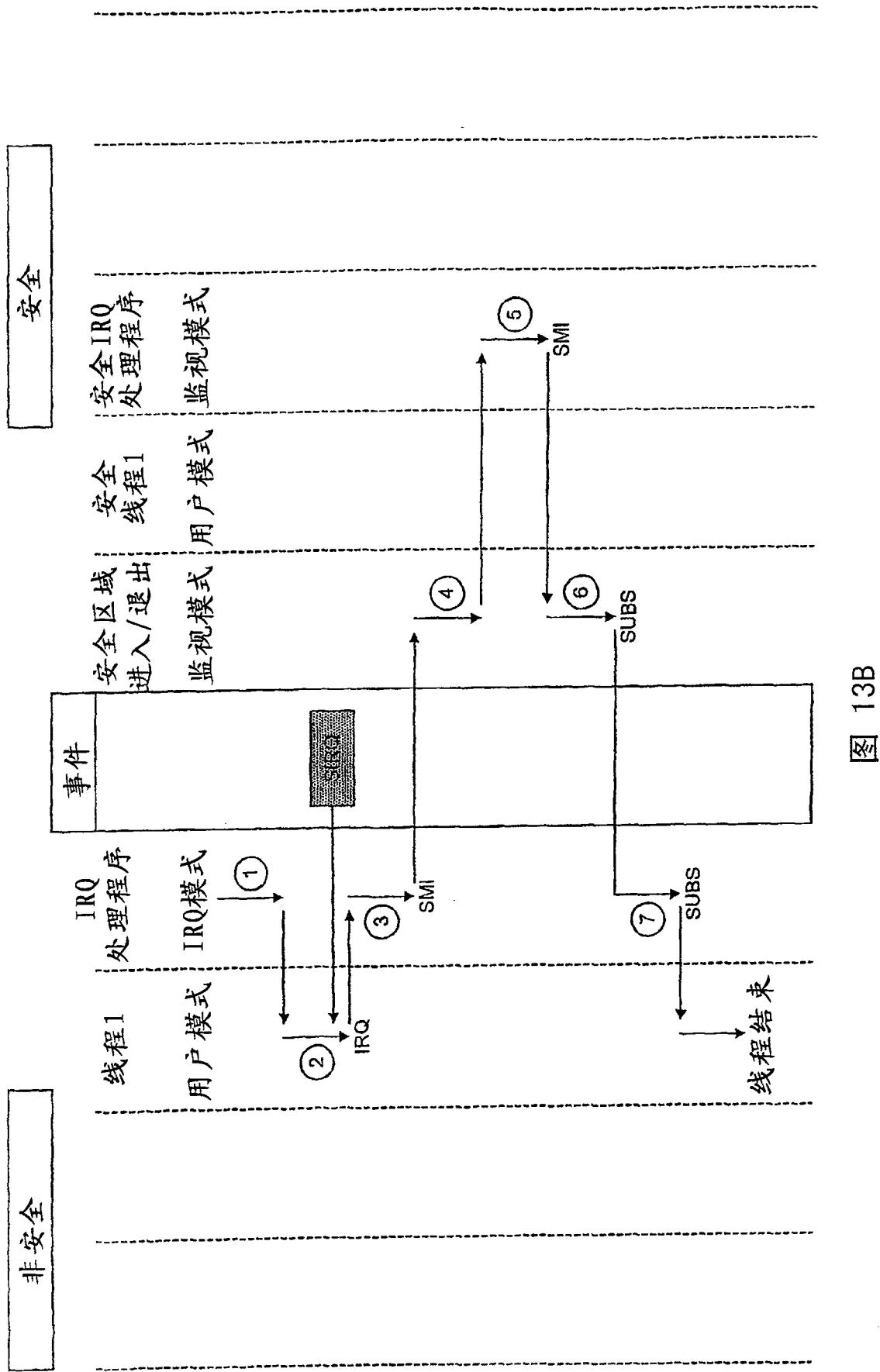


图 13B

中断向量	中断向量值	响应优先级
复位	0x00	监督模式
未定义	0x04	监视模式/未定义模式
SWI	0x08	监督模式/监视模式
预提取中止	0x0C	中止模式/监视模式
数据中止	0x10	中止模式/监视模式
IRQ / SIRQ	0x18	IRQ模式/监视模式
FIQ	0x1C	IRQ模式/监视模式
SMI	0x20	未定义模式/监视模式

图 14

监视程序

安全	非安全
复位	VMO
未定义	VM1
SWI	VM2
预提取中止	VM3
数据中止	VM4
IRQ / SIRQ	VM5
FIQ	VM6
SMI	VM7

安全	非安全
复位	VSO
未定义	VS1
SWI	VS2
预提取中止	VS3
数据中止	VS4
IRQ / SIRQ	VS5
FIQ	VS6
SMI	VS7

安全	非安全
复位	VNS0
未定义	VNS1
SWI	VNS2
预提取中止	VNS3
数据中止	VNS4
IRQ / SIRQ	VNS5
FIQ	VNS6
SMI	VNS7

图 15

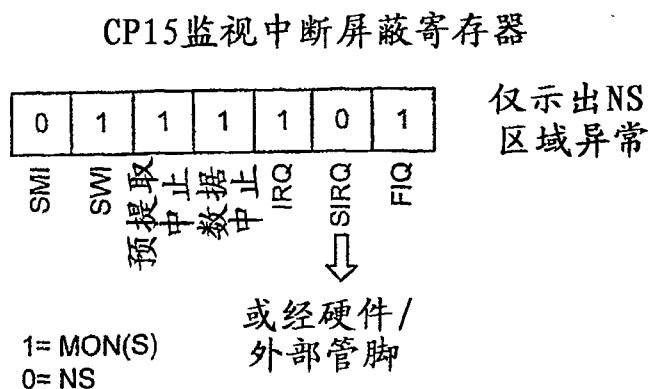


图 16

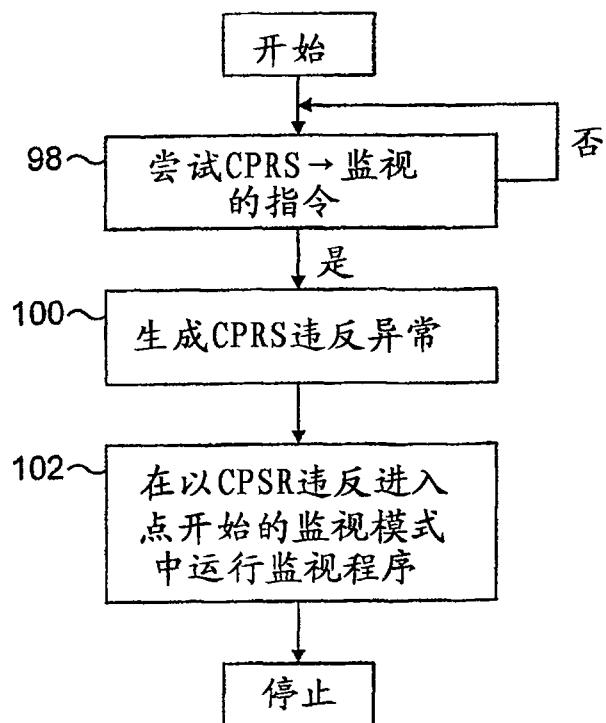


图 17

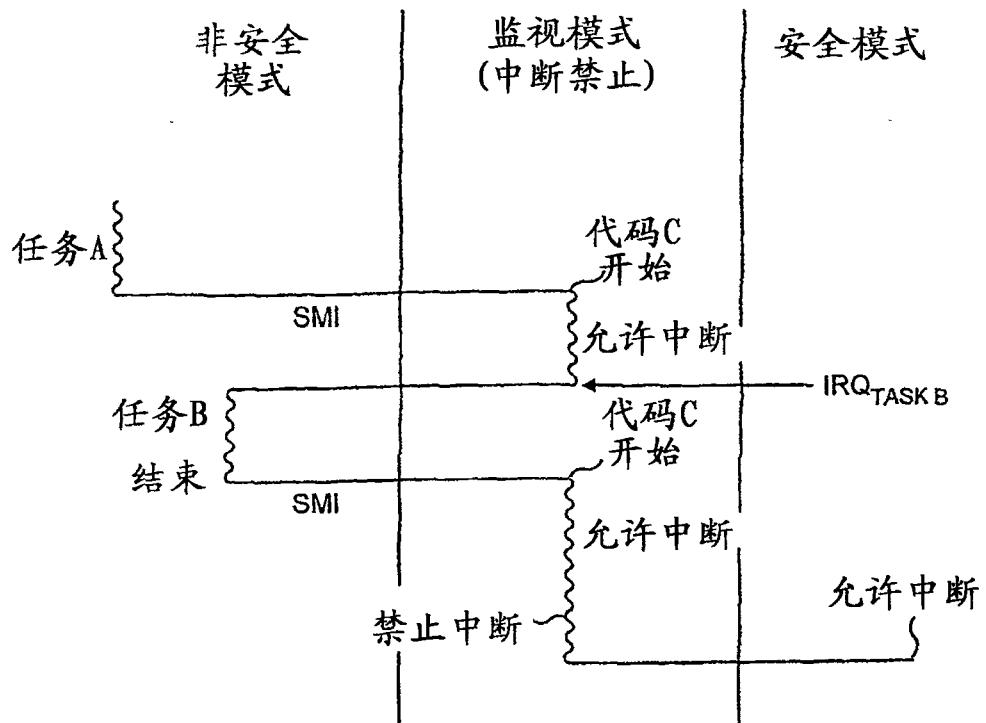


图 18

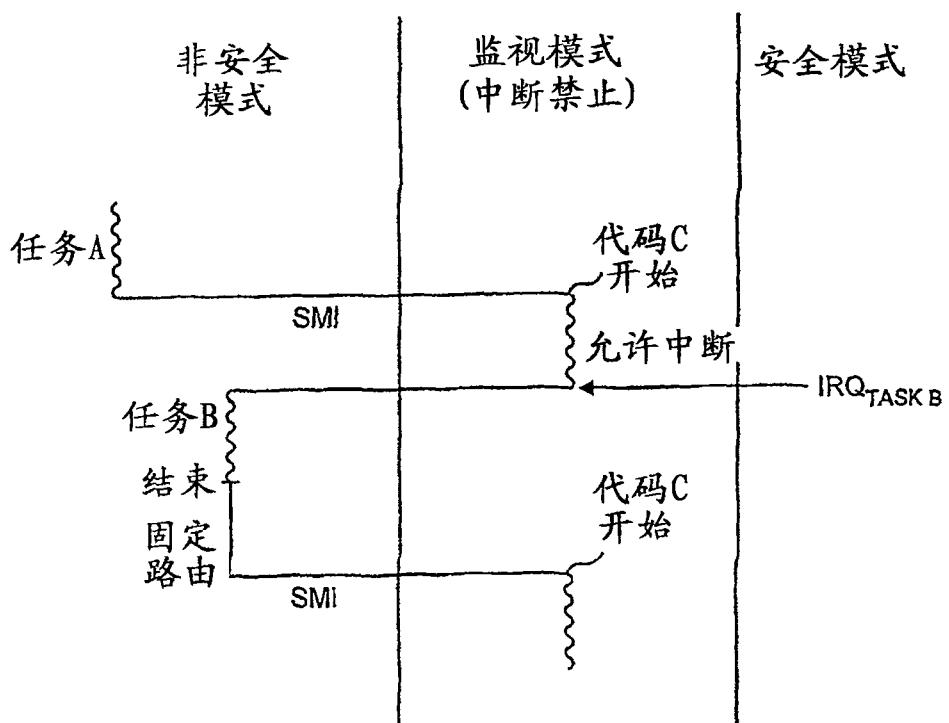


图 19

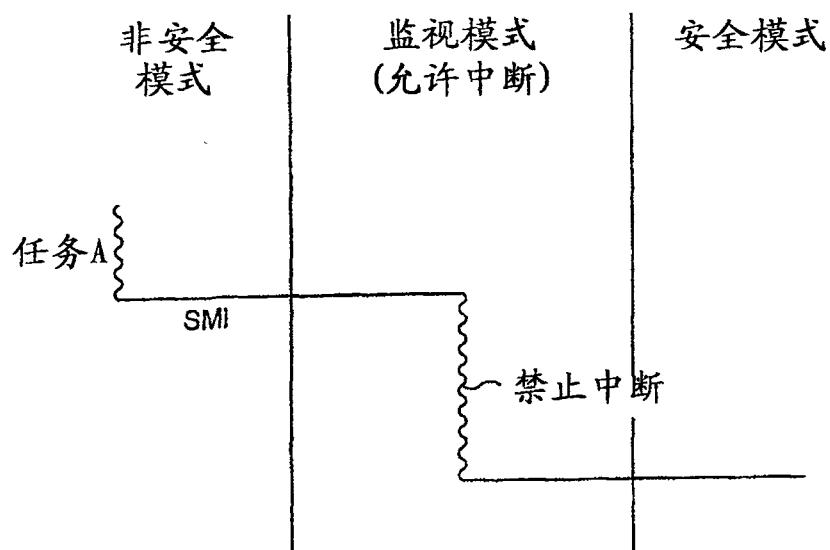


图 20

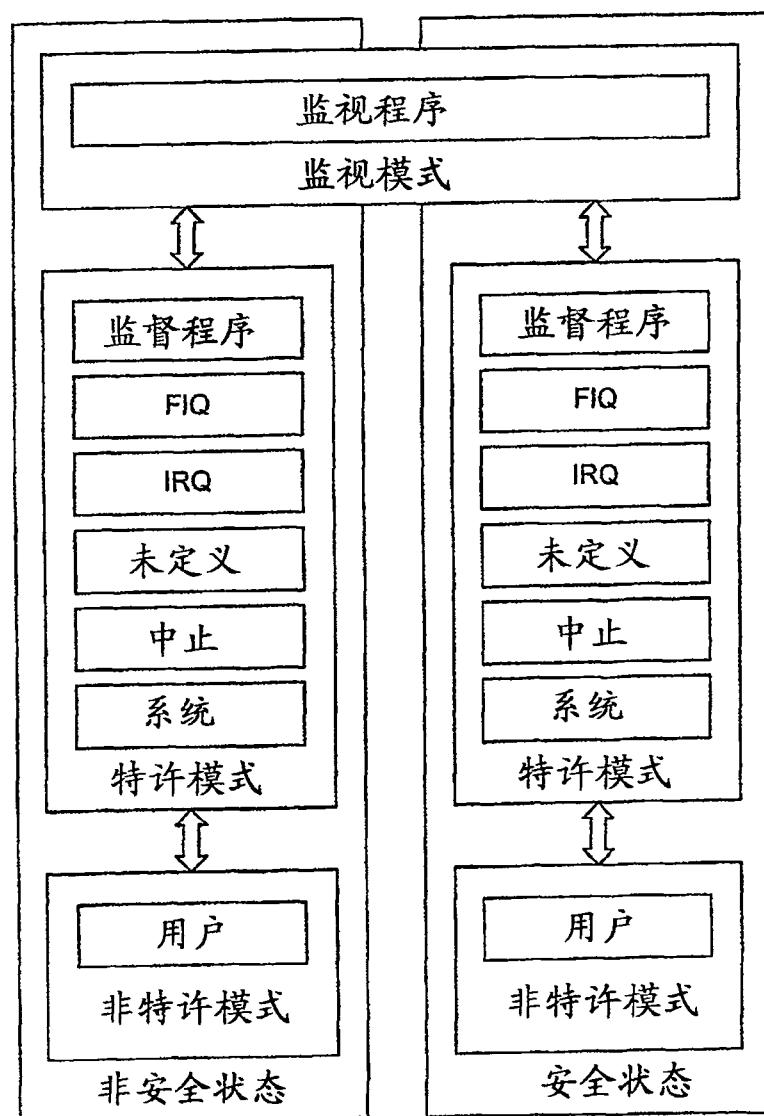


图 21

用户	系统	监督程序	中止	未定义	中断	快速中断	监视程序
R0	R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8	R8
R9	R9	R9	R9	R9	R9	R9	R9
R10	R10	R10	R10	R10	R10	R10	R10
R11	R11	R11	R11	R11	R11	R11	R11
R12	R12	R12	R12	R12	R12	R12	R12
R13	R13	R13_SVC	R13_SVC	R13_UND	R13_IRQ	R13_FIQ	R13_MON
R14	R14	R14_SVC	R14_SVC	R14_UND	R14_IRQ	R14_FIQ	R14_MON
PC	PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_SVC	SPSR_ABST	SPSR_UND	SPSR_IRQ	SPSR_FIQ	SPSR_MON	

图 22

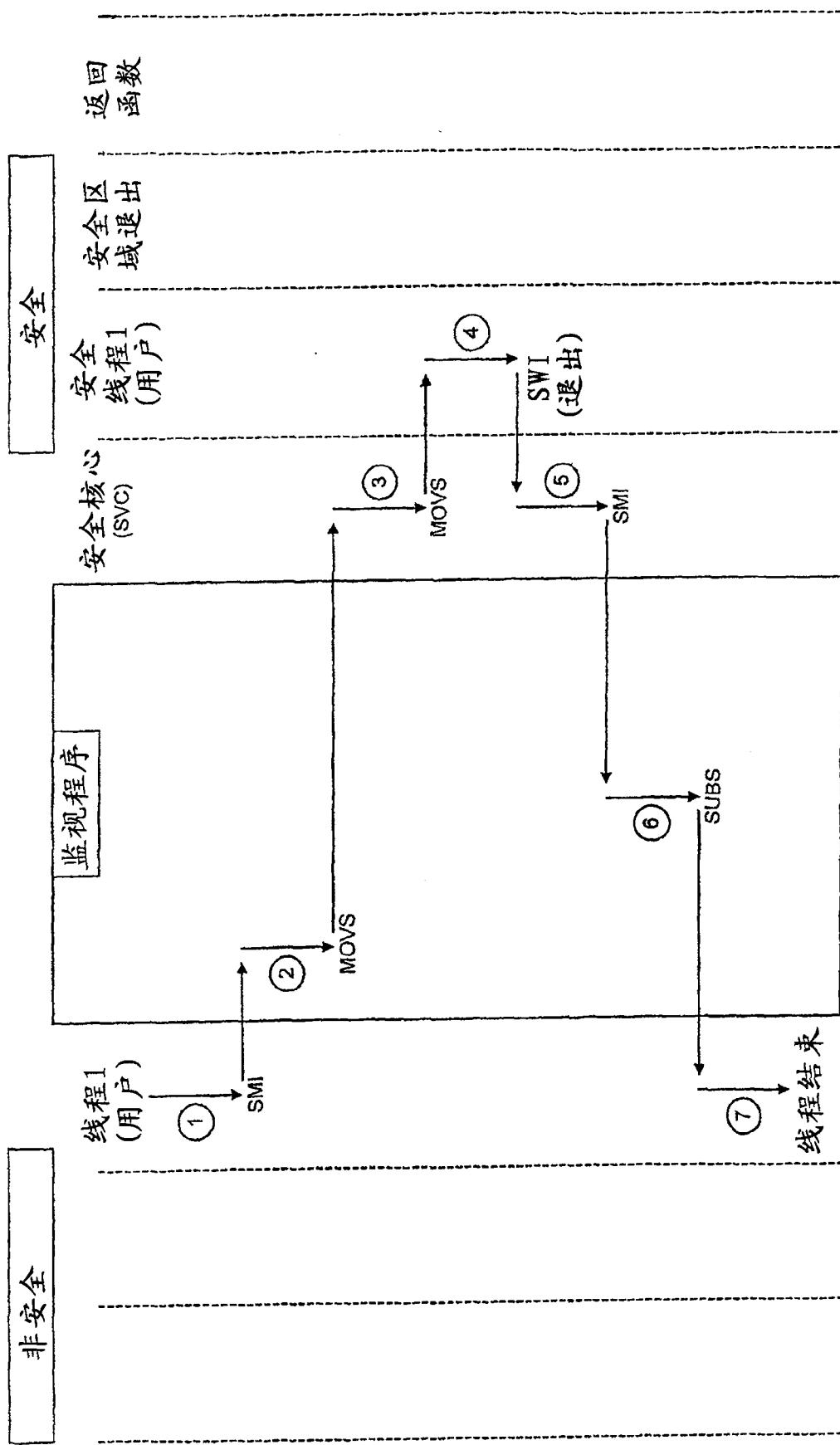


图 23

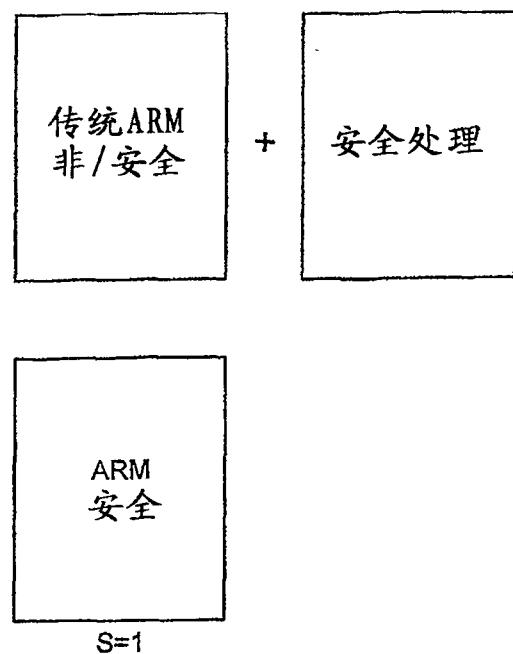


图 24

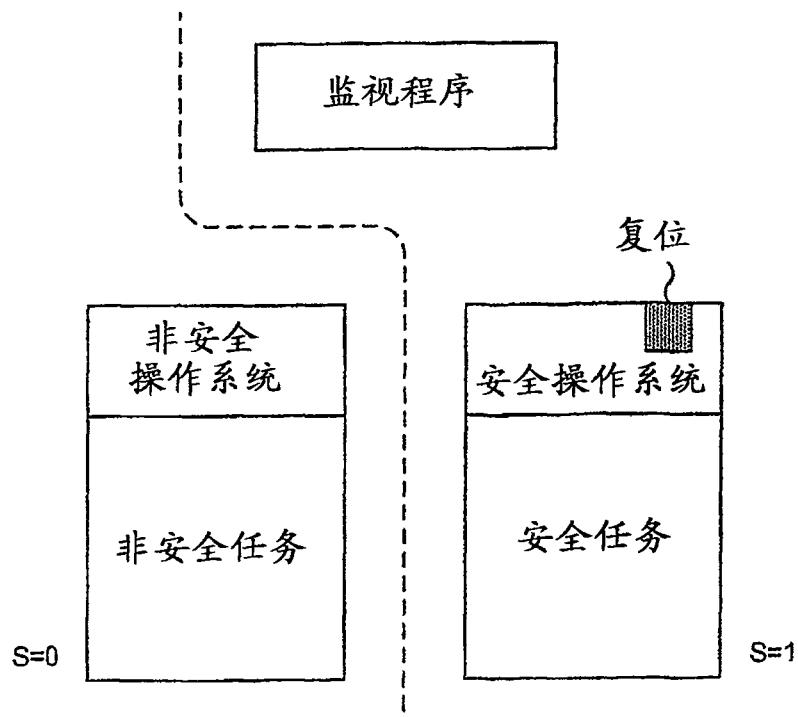


图 25

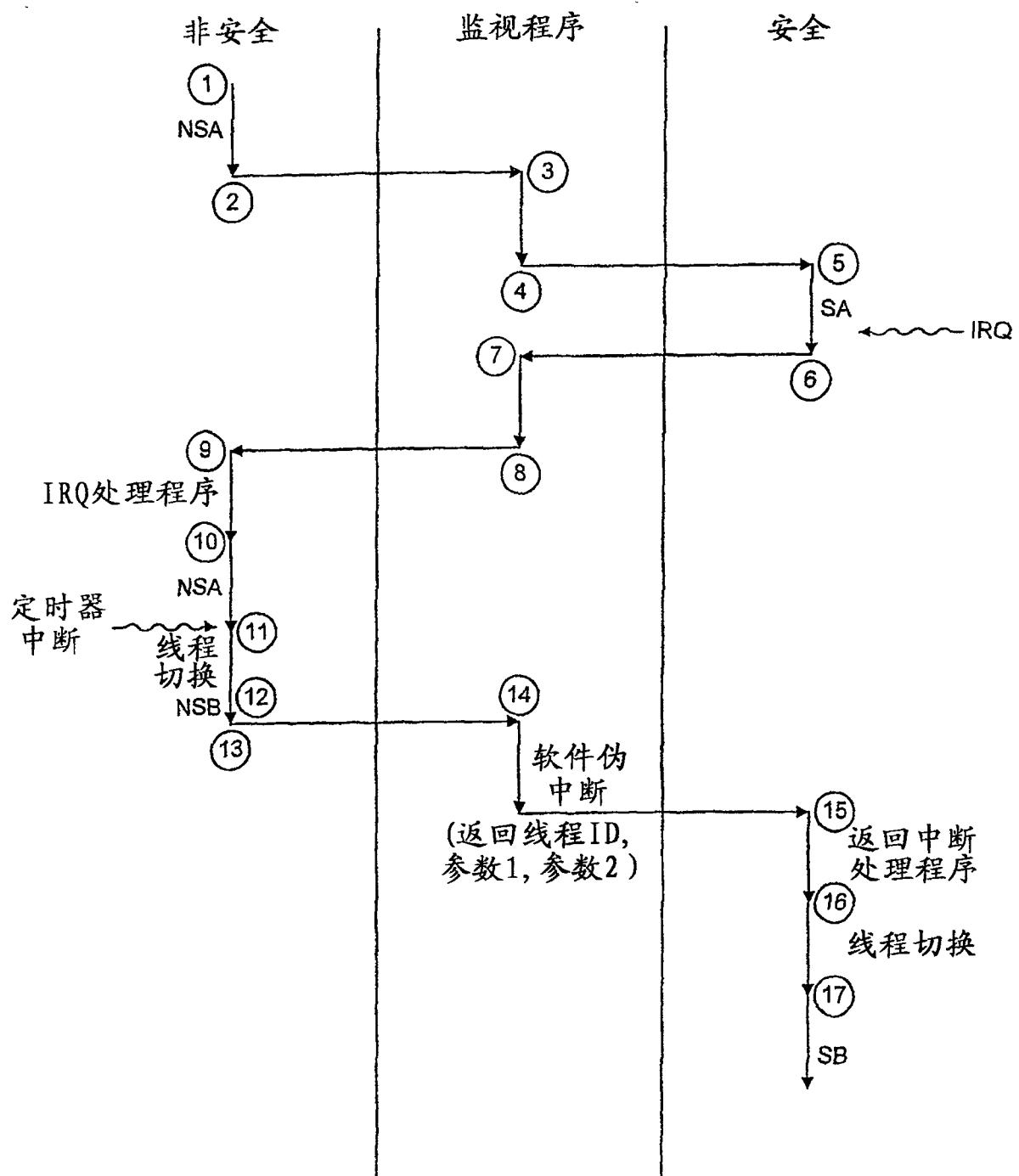


图 26

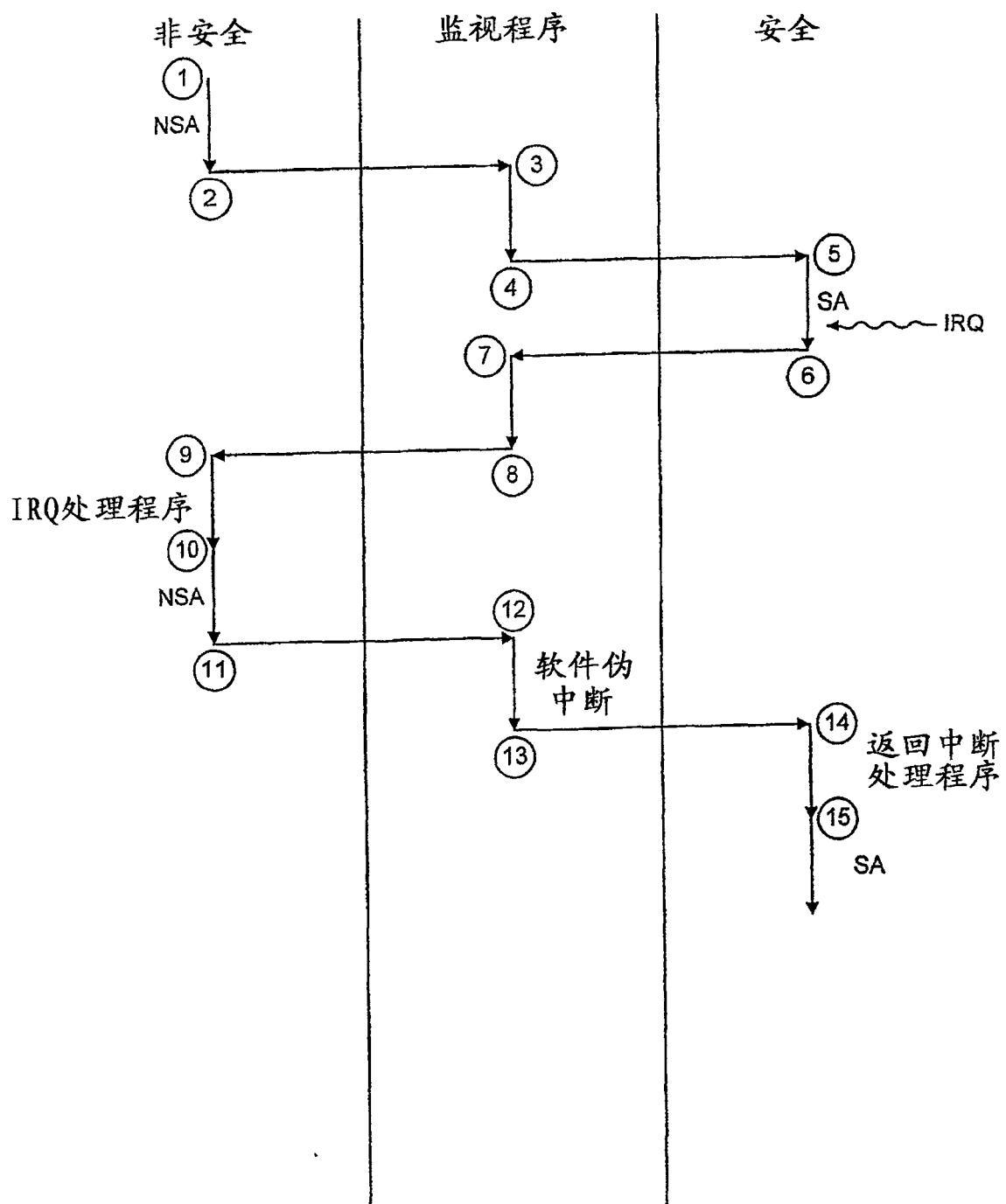


图 27

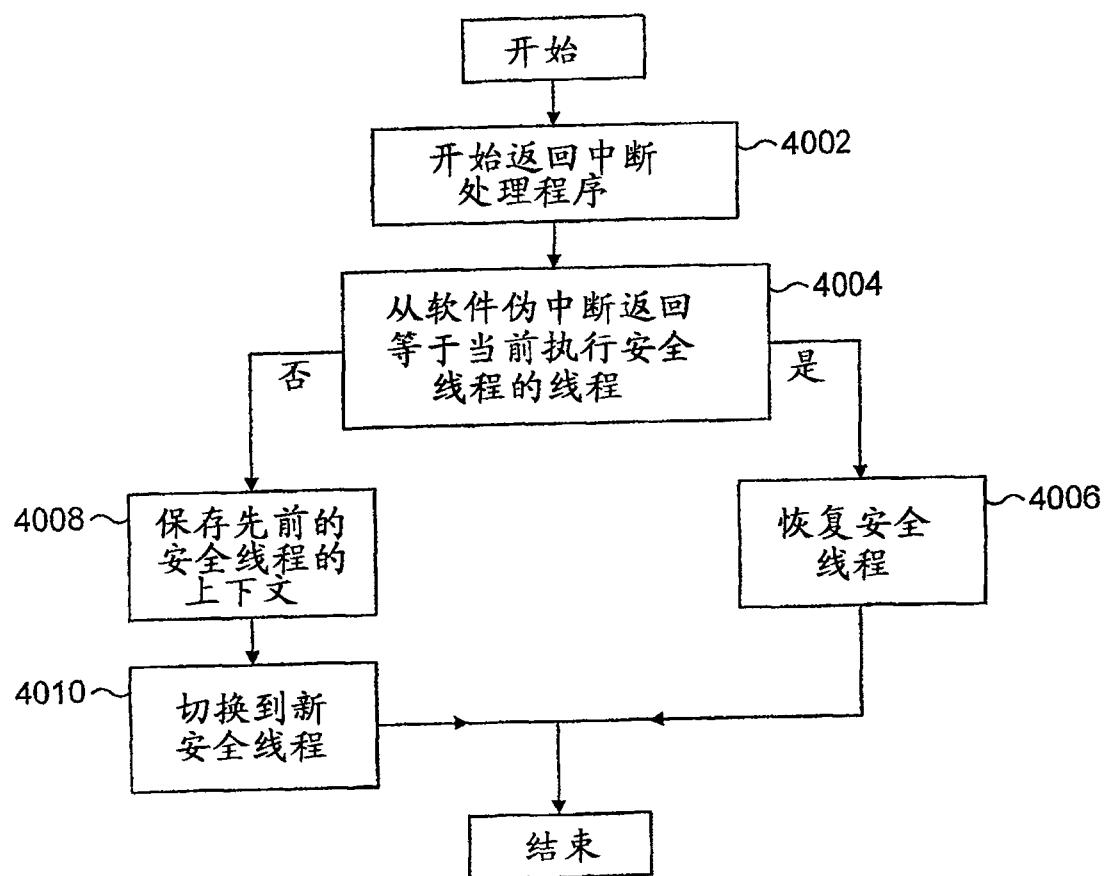


图 28

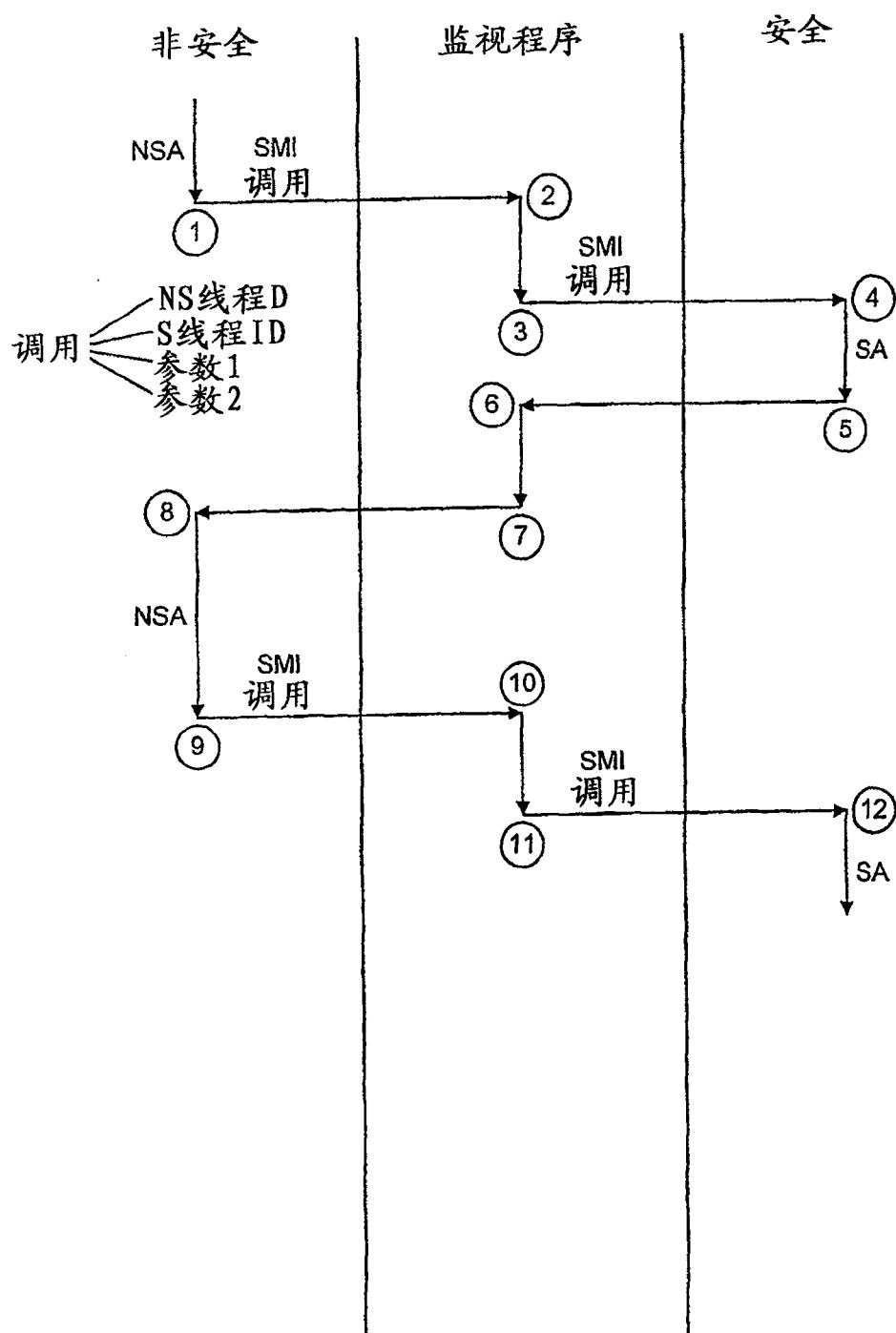


图 29

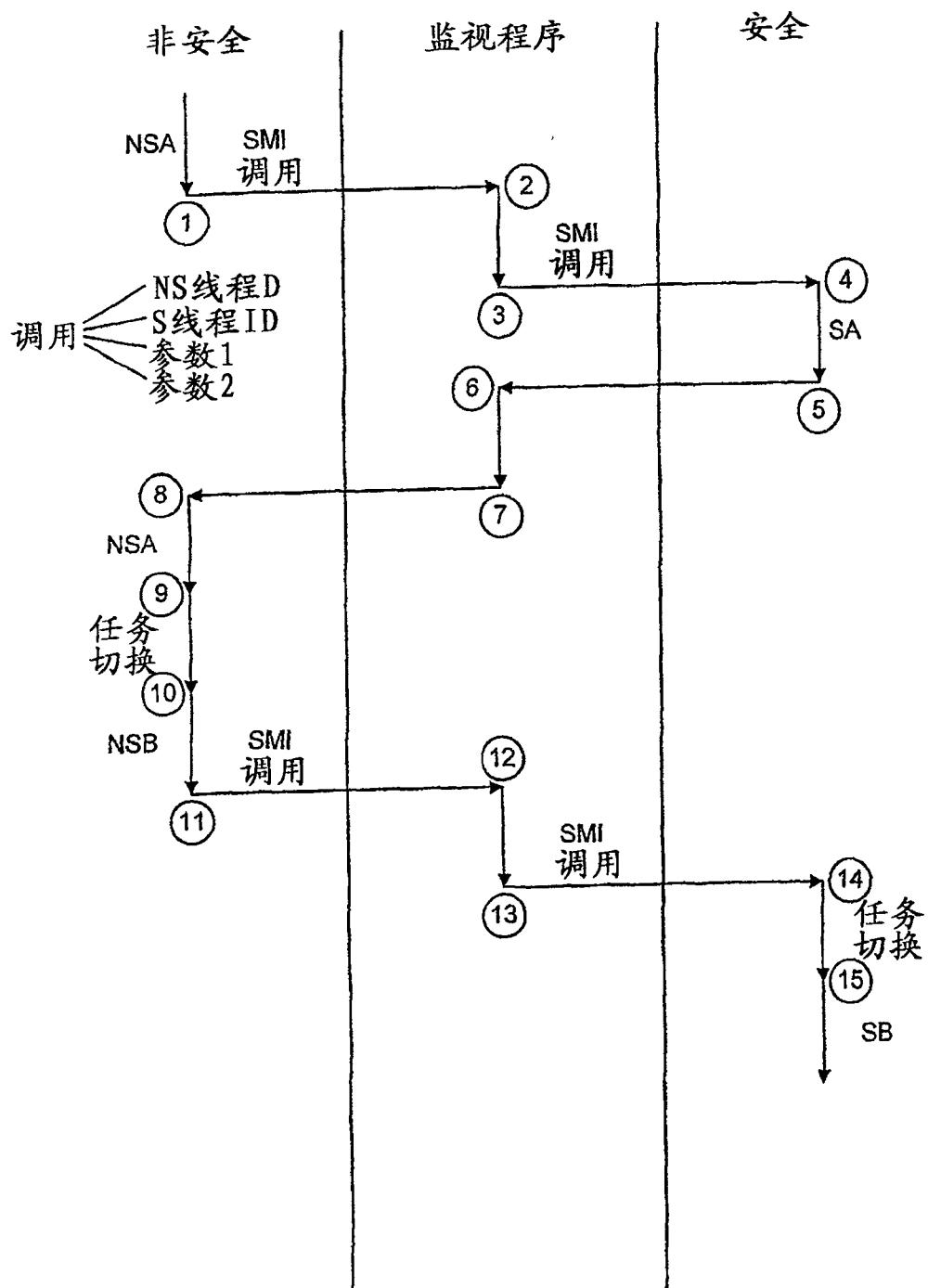


图 30

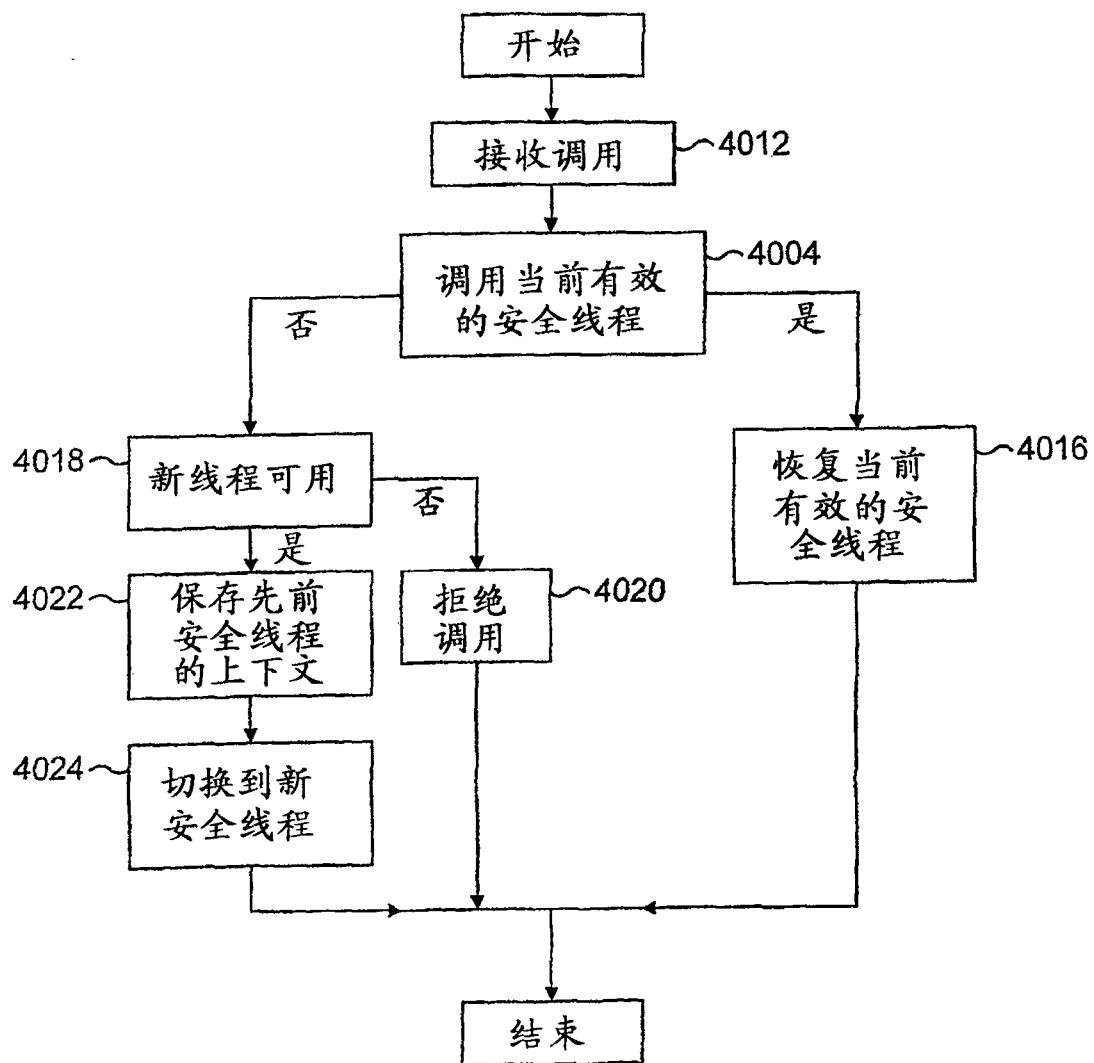


图 31

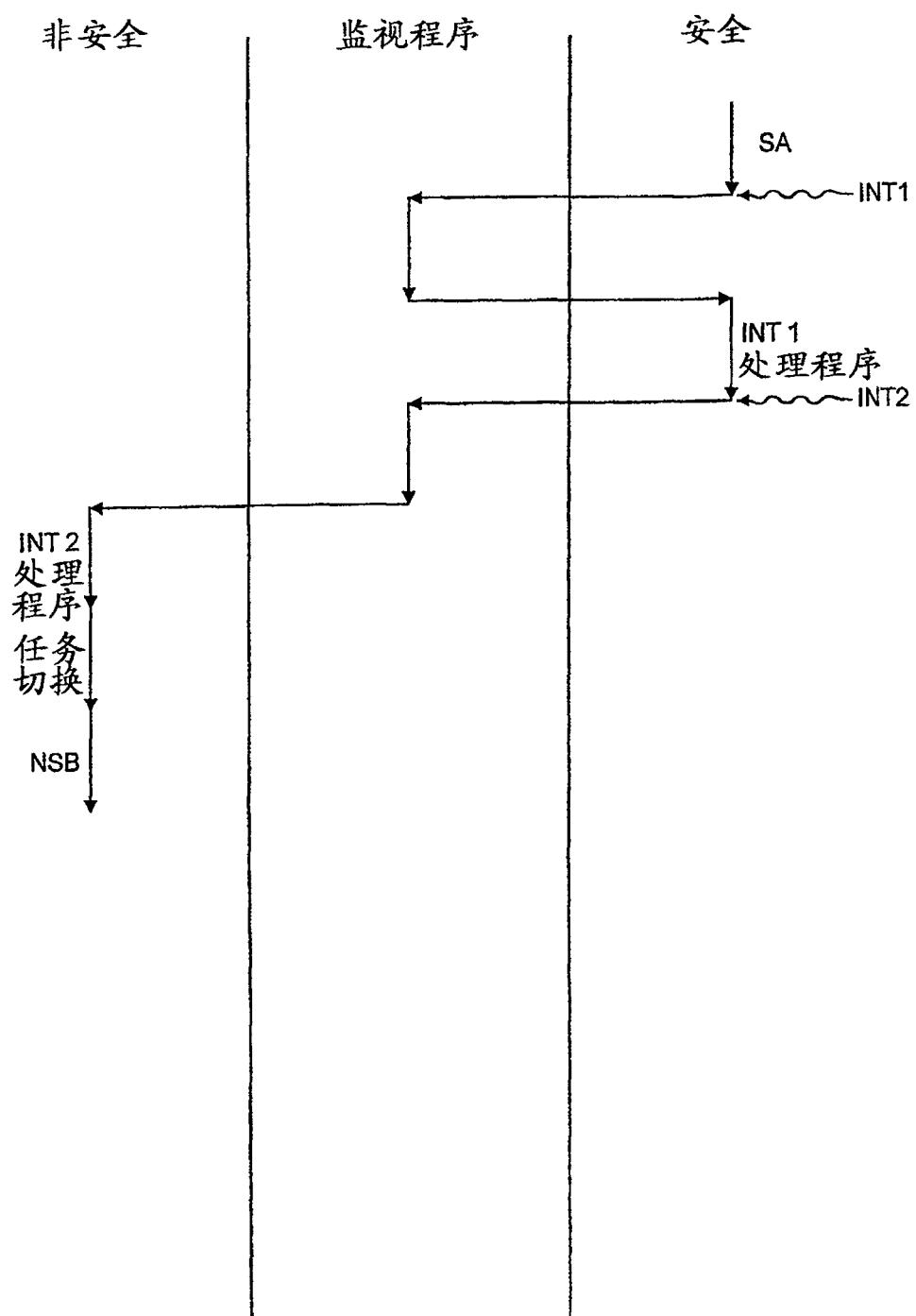


图 32

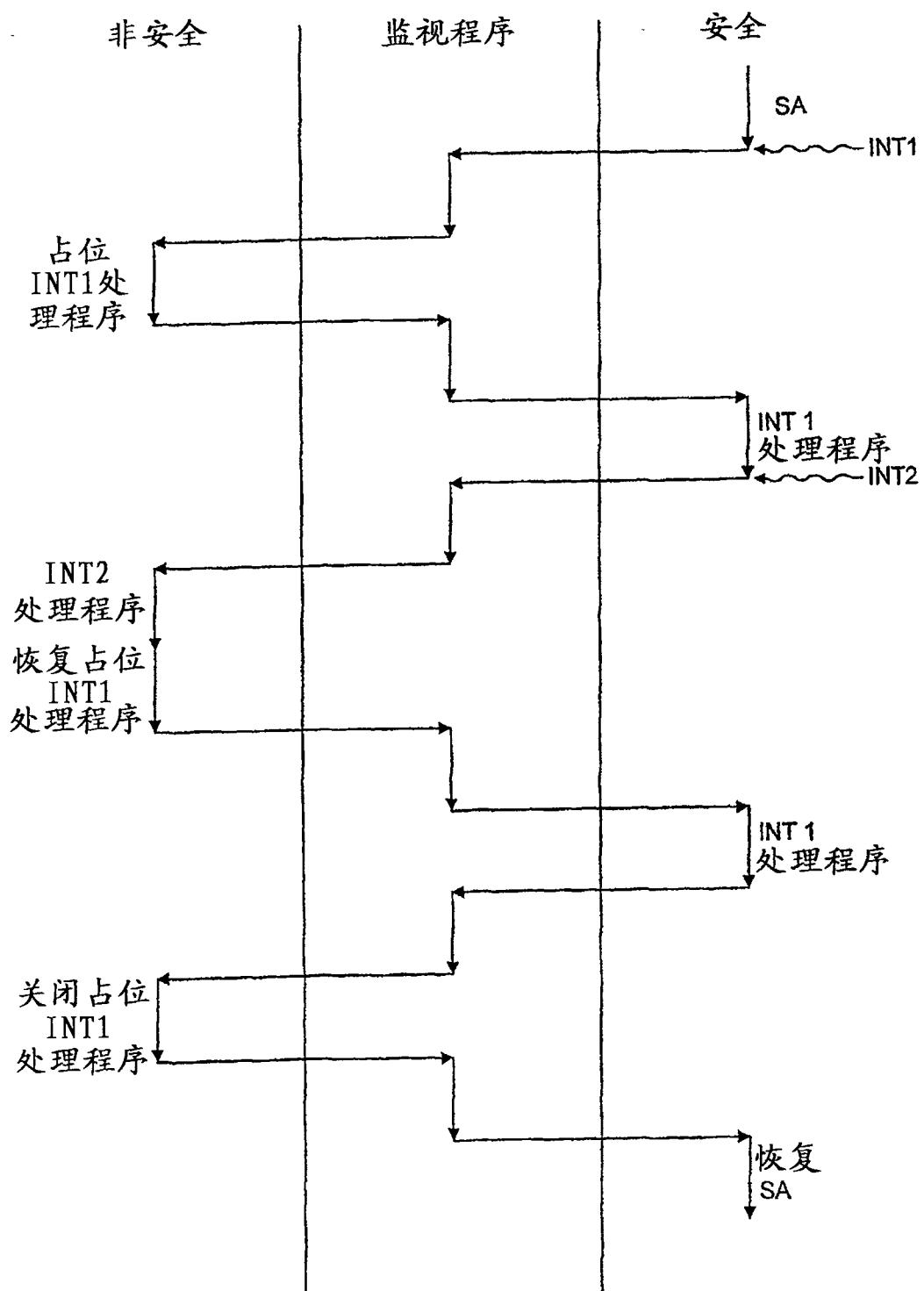


图 33

<u>中断类型 / 优先级</u>	<u>如何处理</u>
1	S
2	S
3	NS
4	NS/S
5	NS
6	NS/S
7	NS
·	·
·	·

仅处理
低于最
高NS处
理程序

图 34

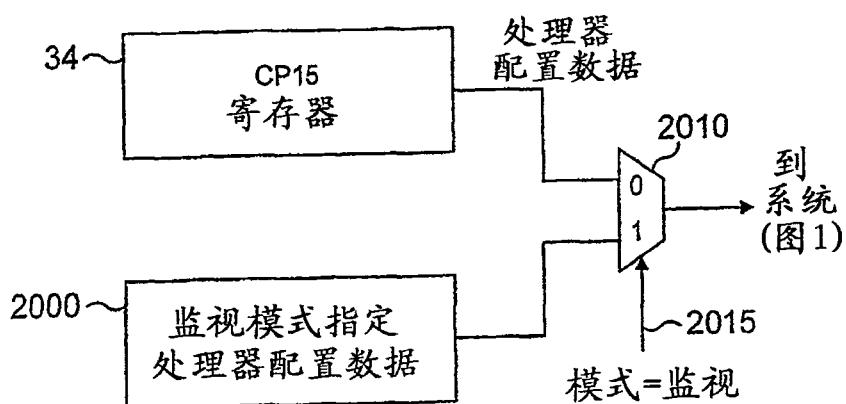


图 35

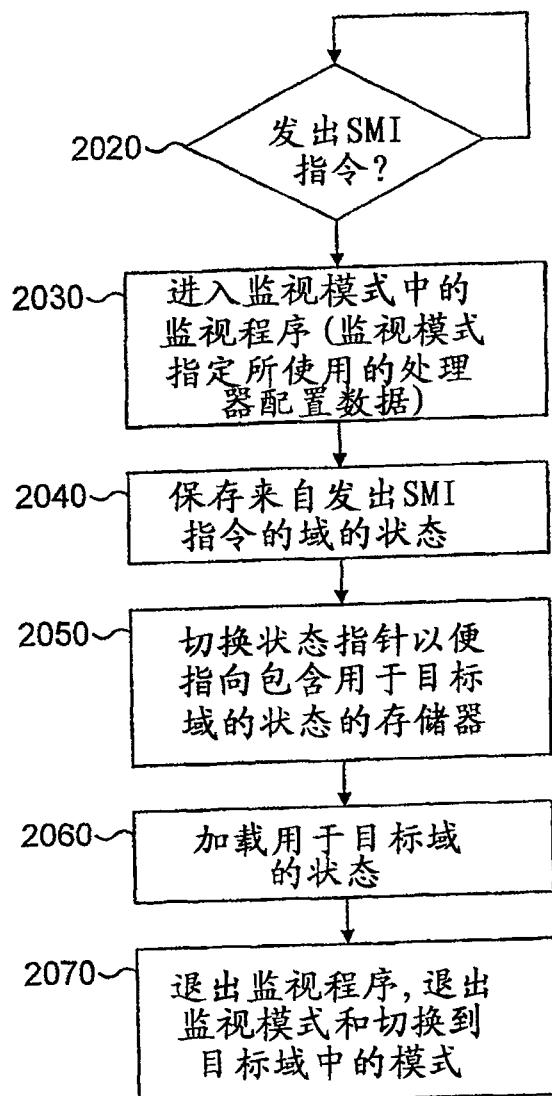


图 36

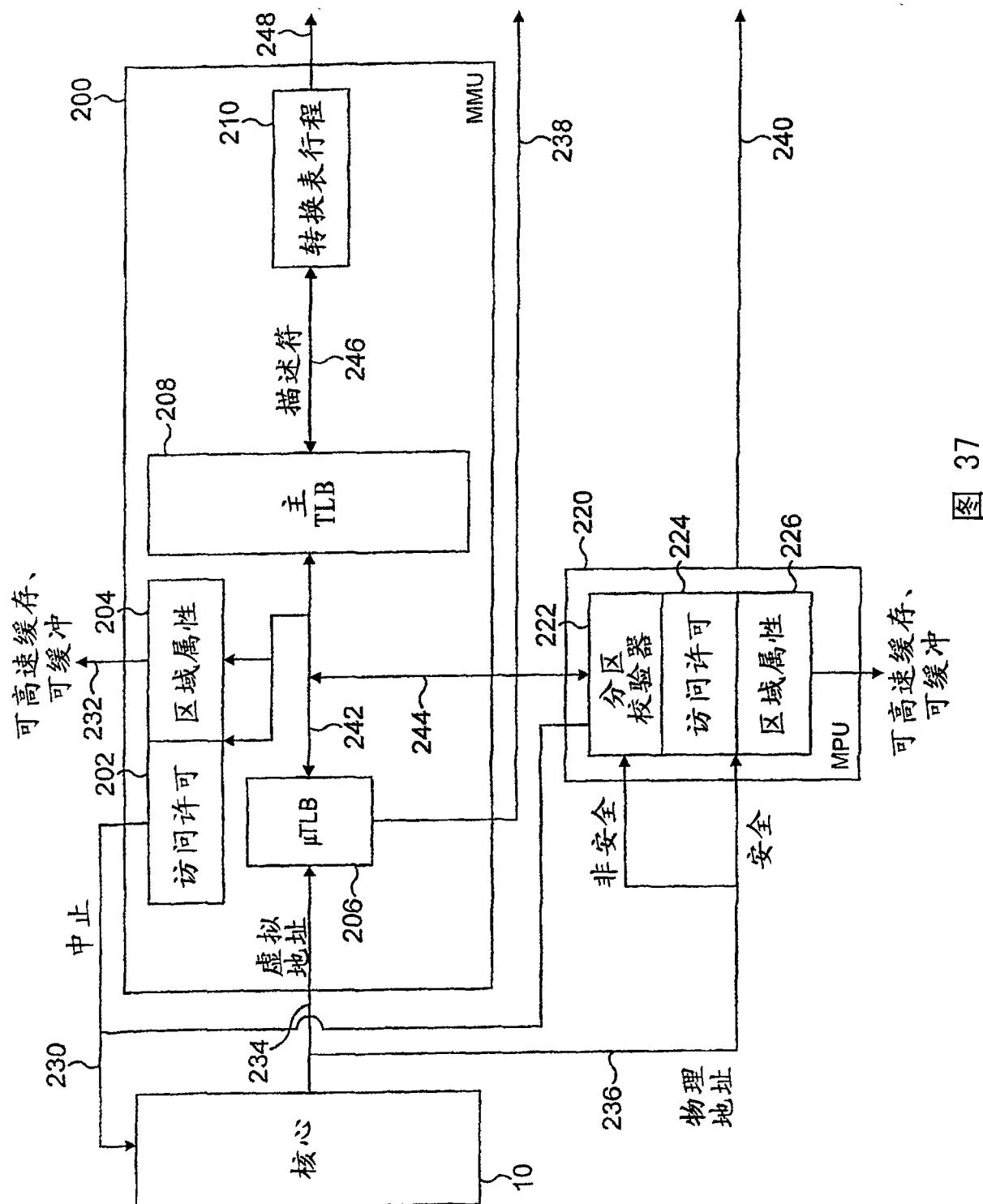


图 37

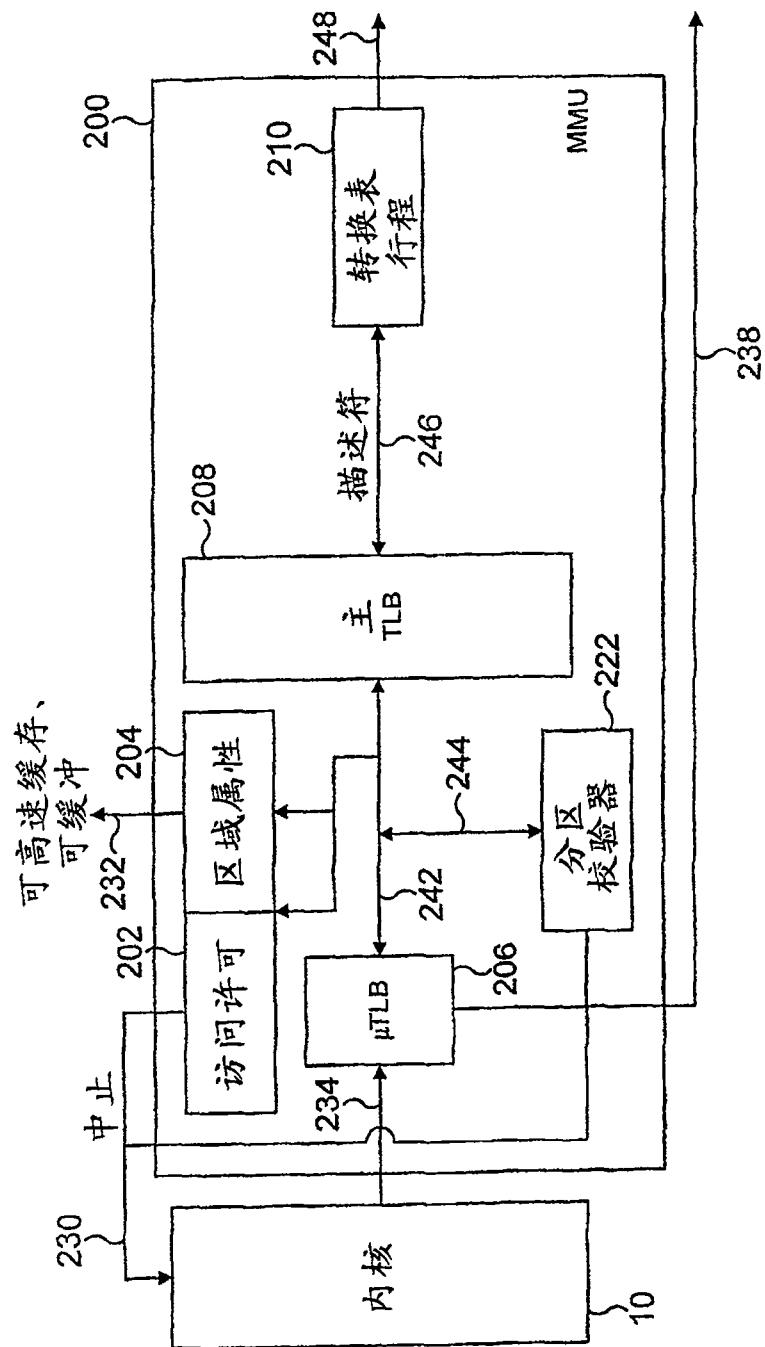


图 38

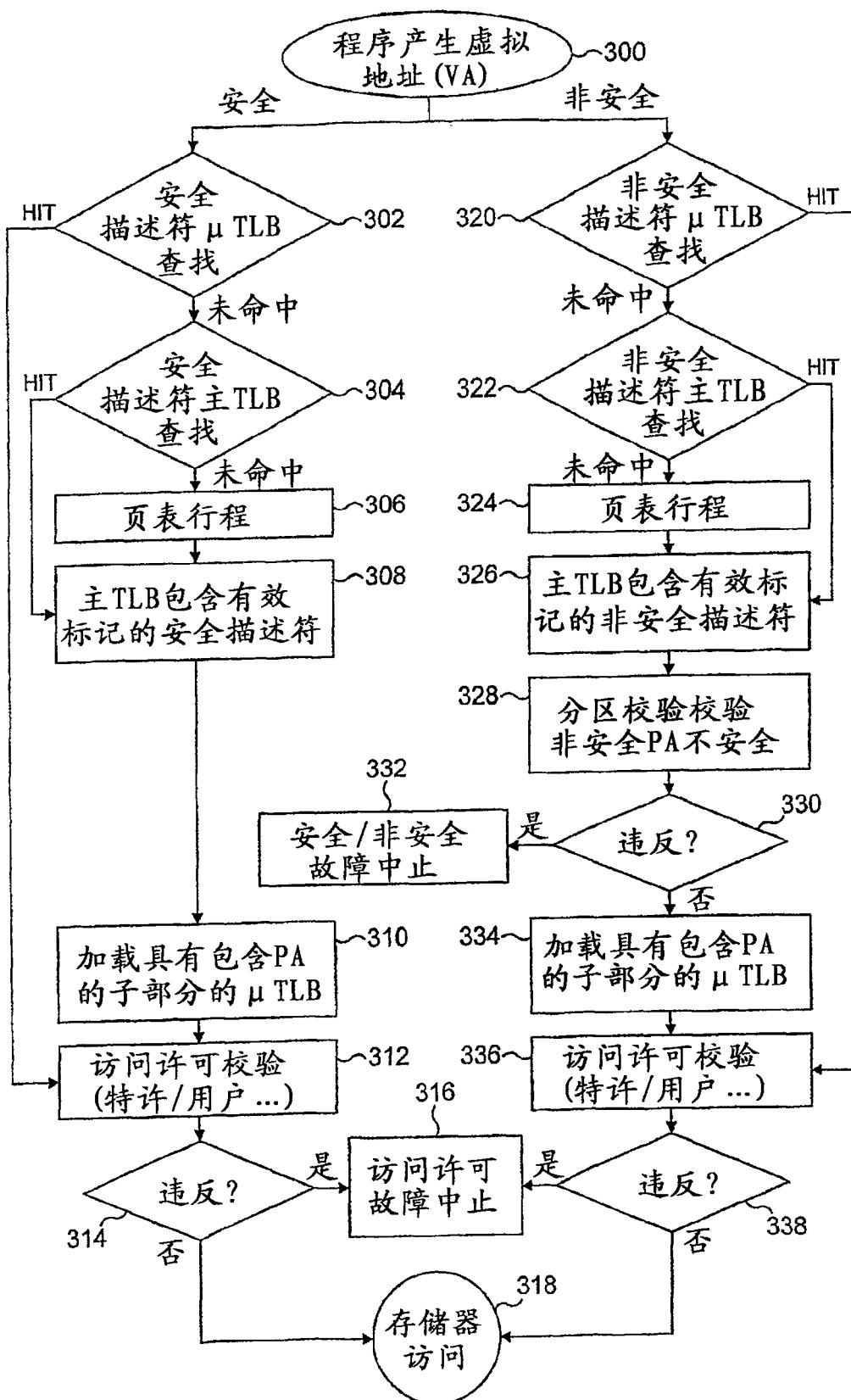


图 39

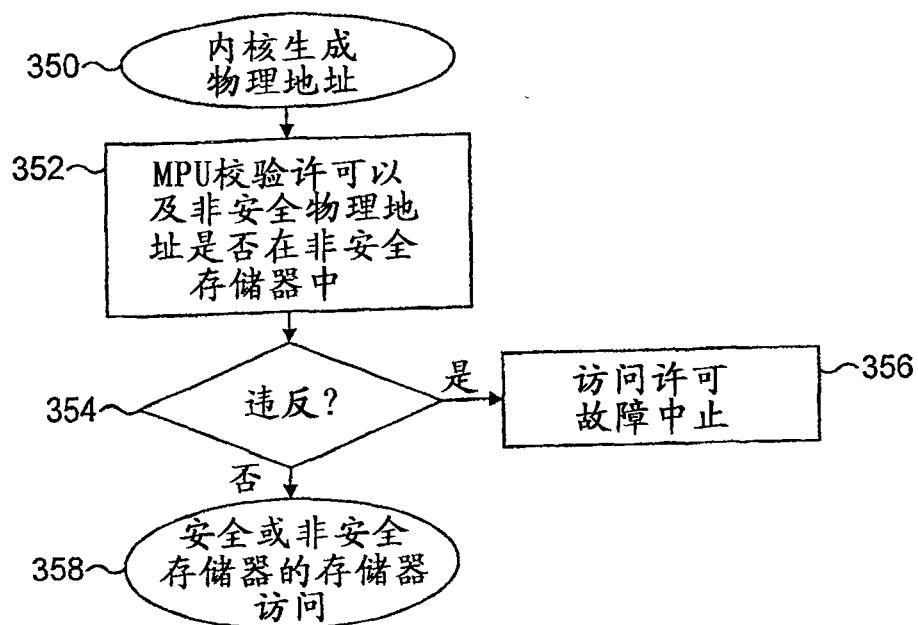


图 40

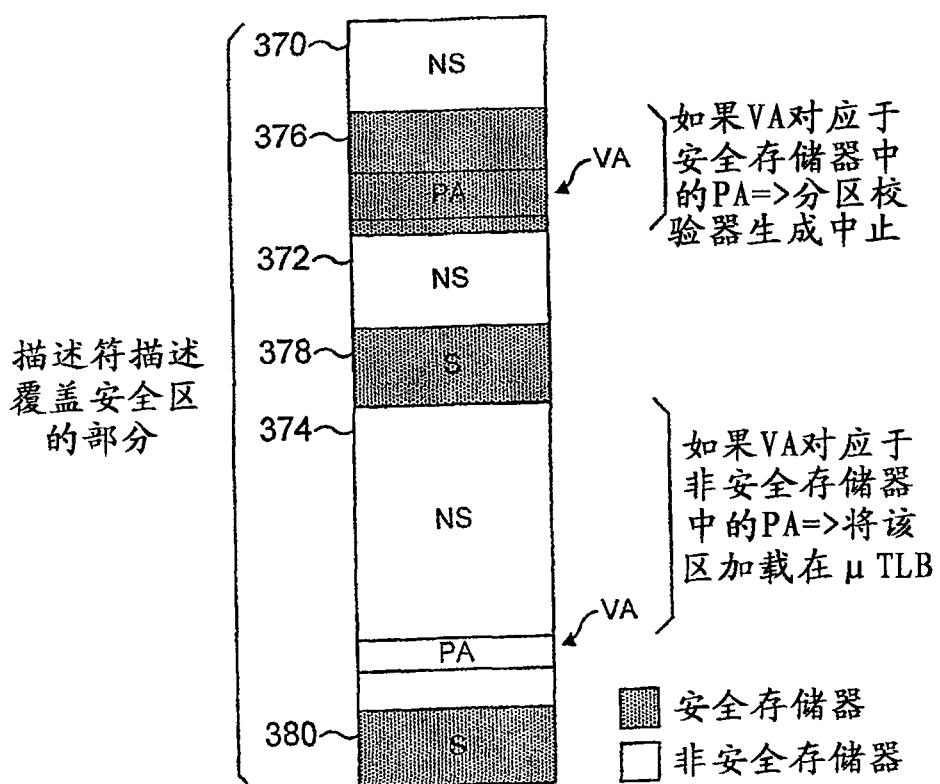


图 41

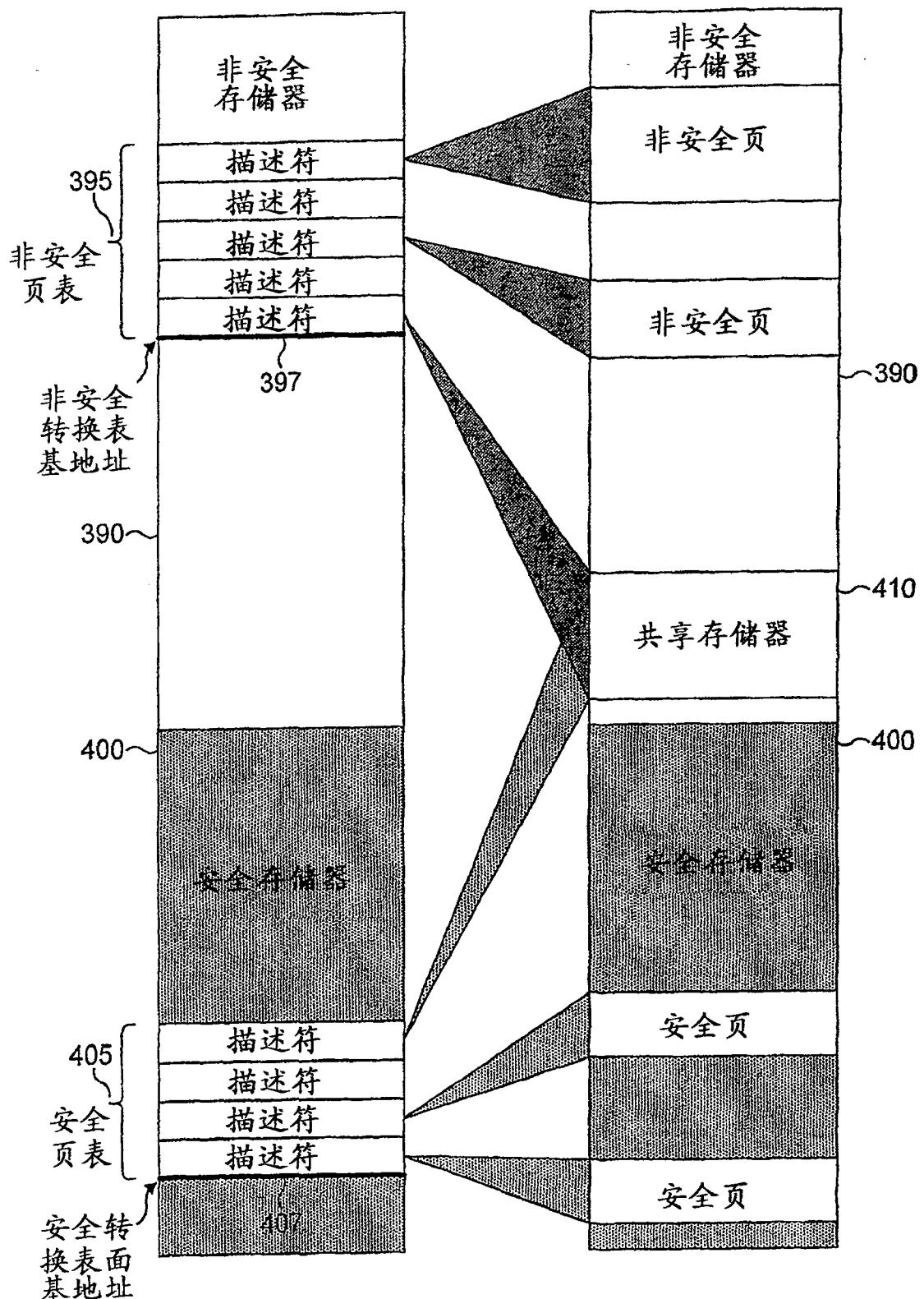
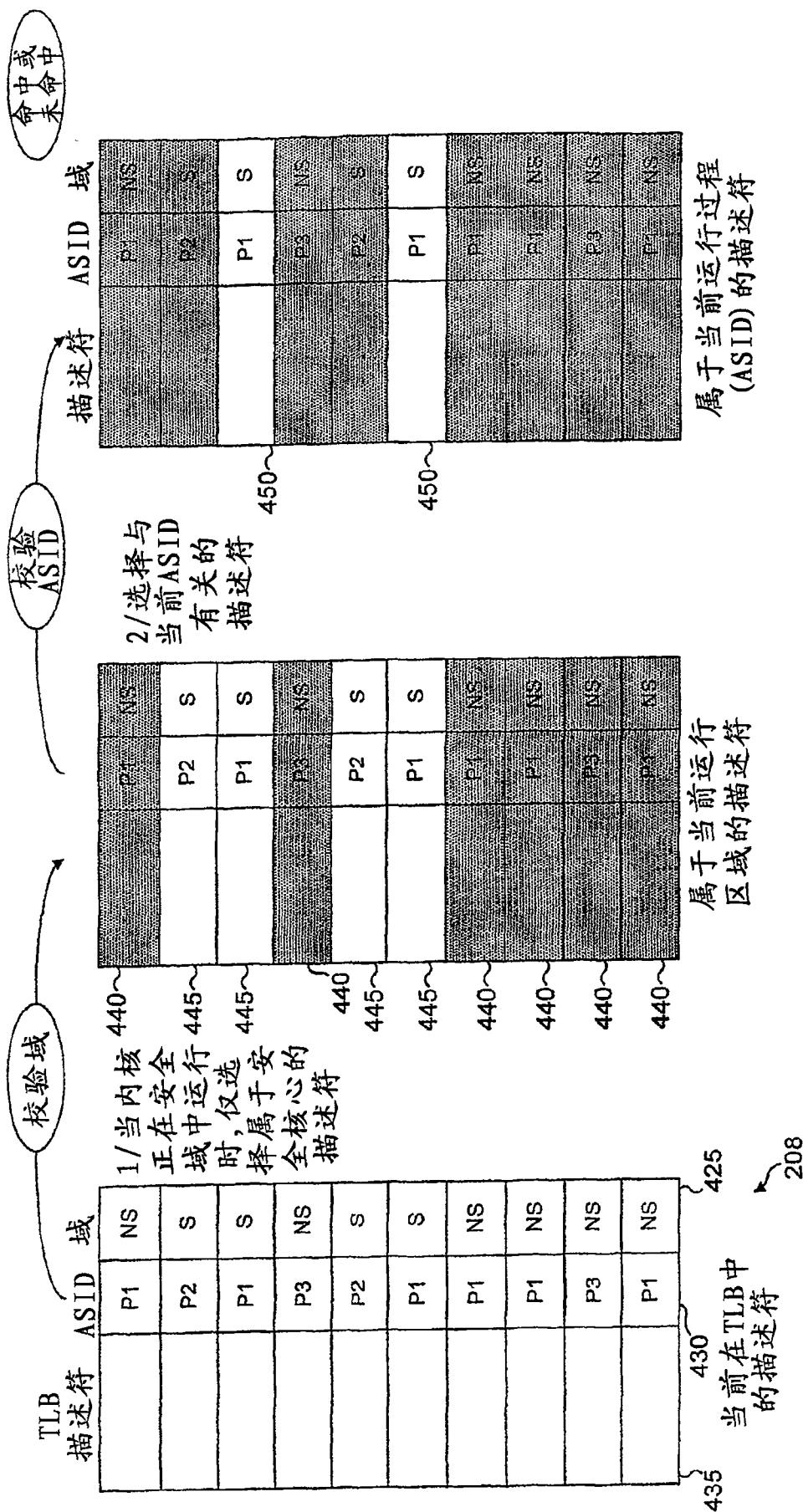


图 42



43

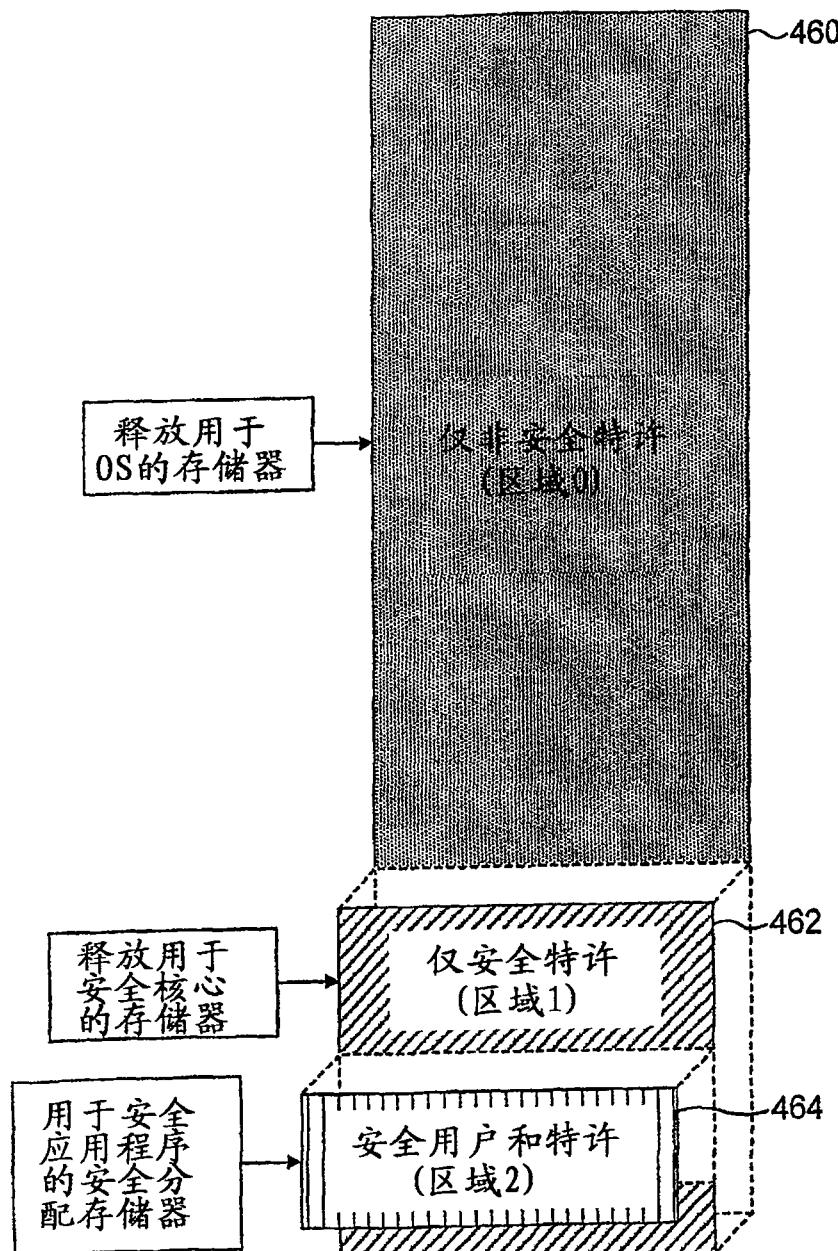


图 44

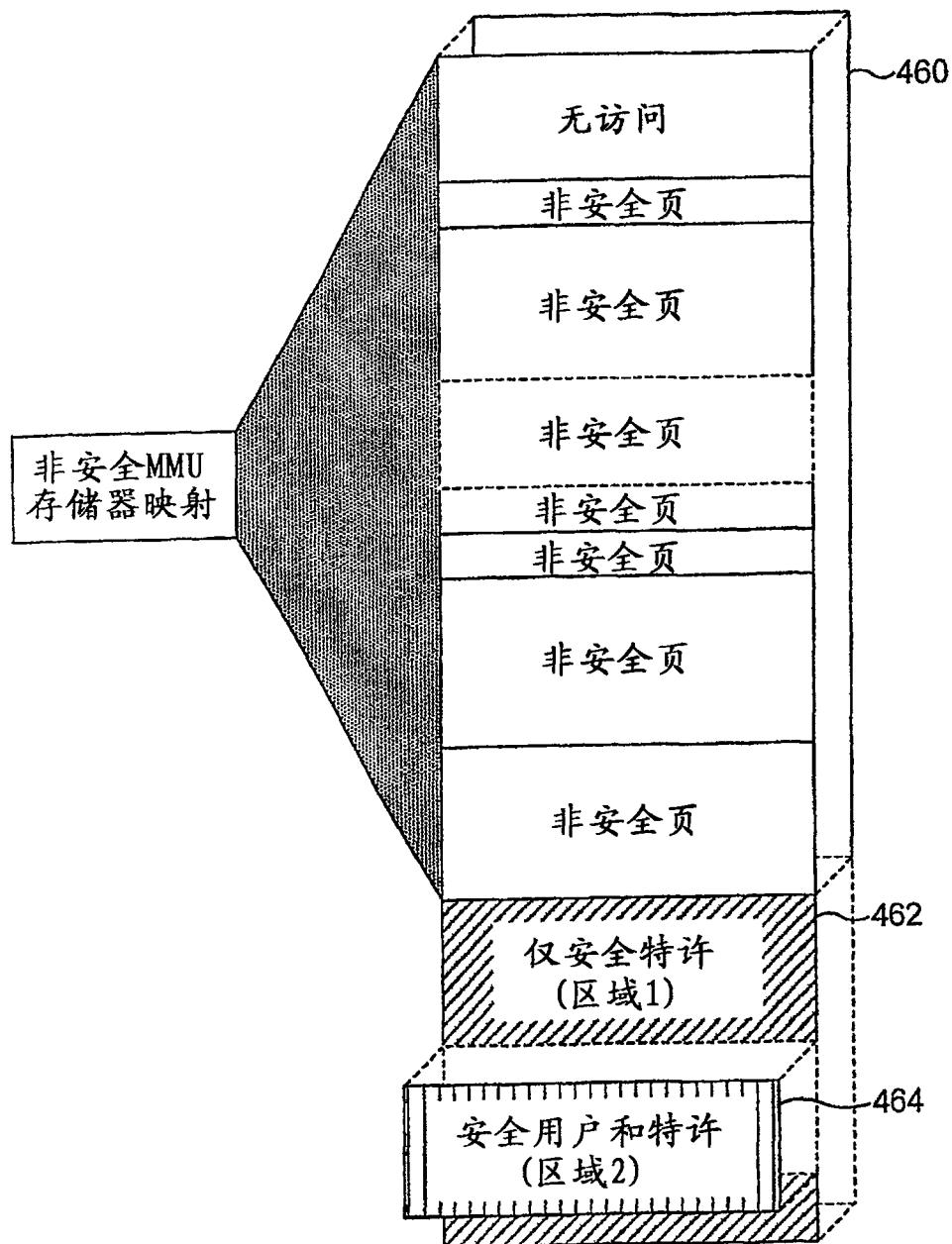


图 45

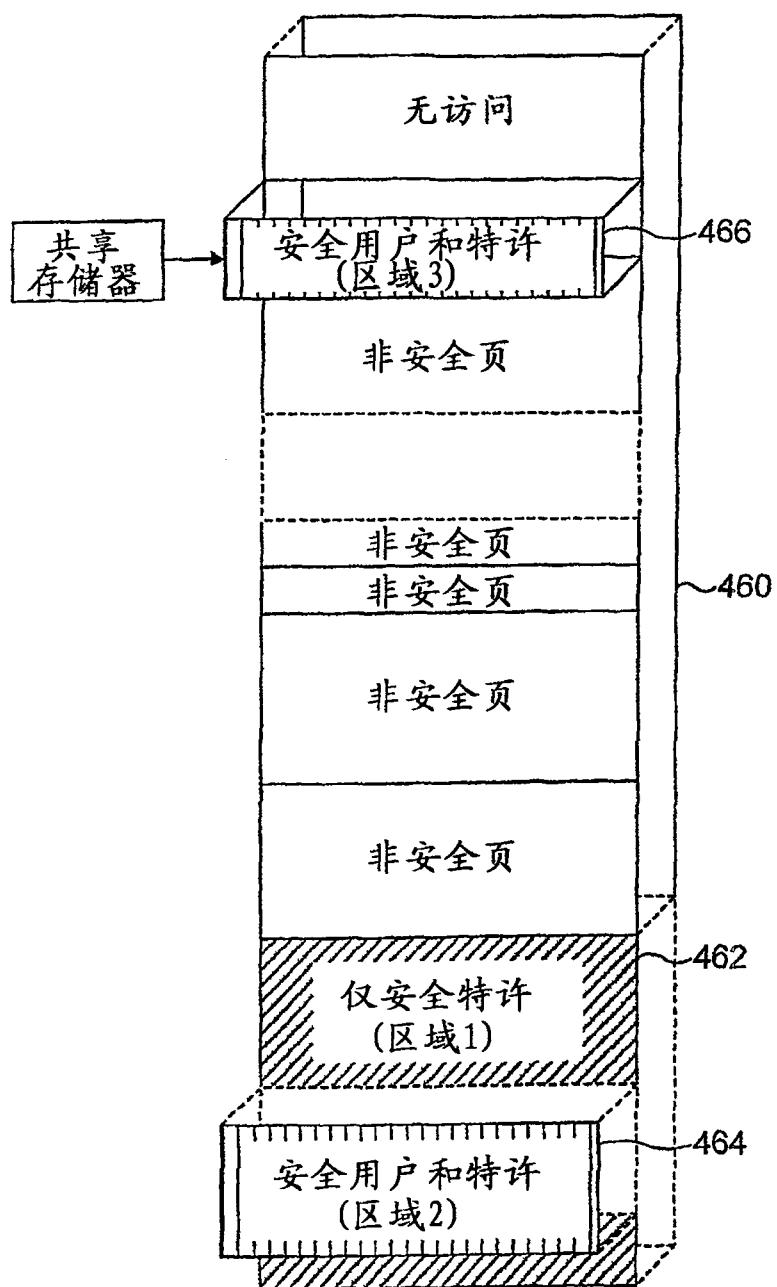


图 46

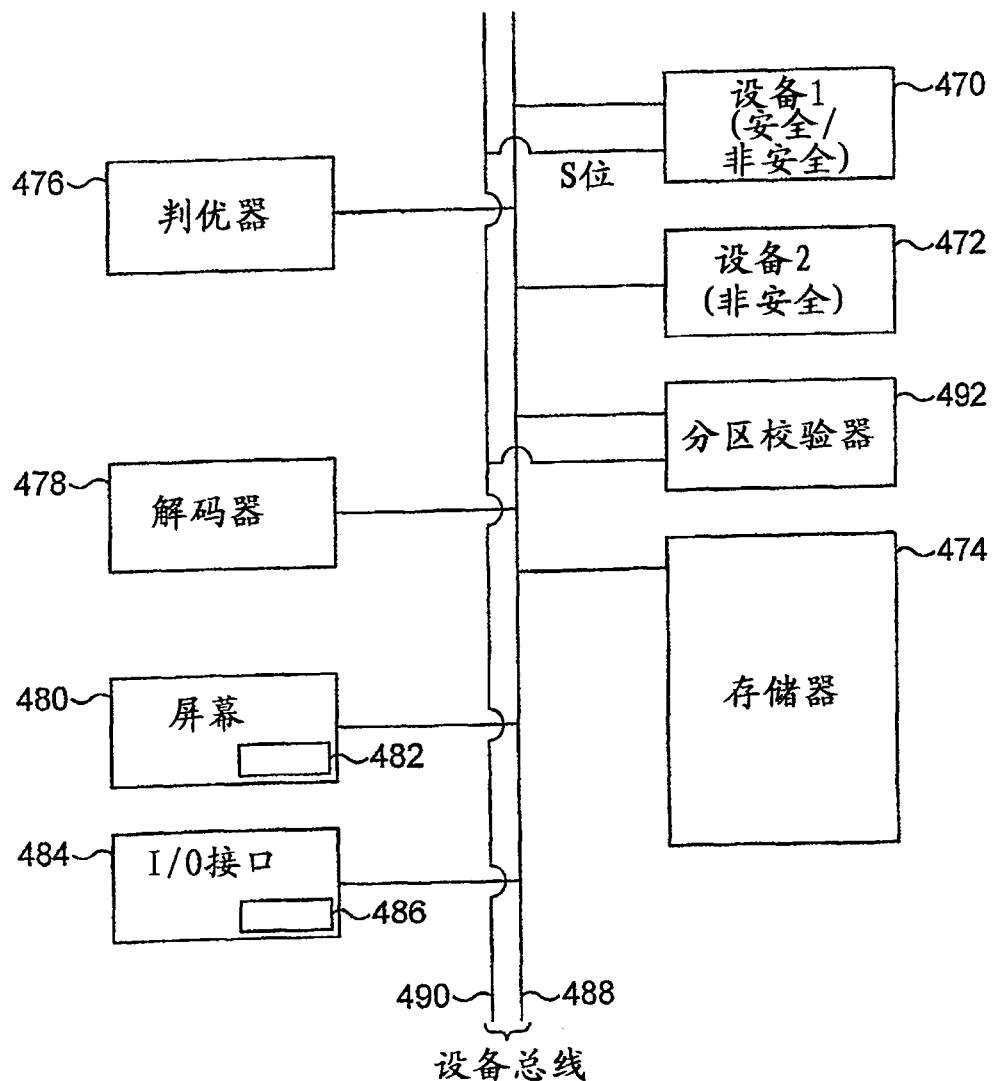
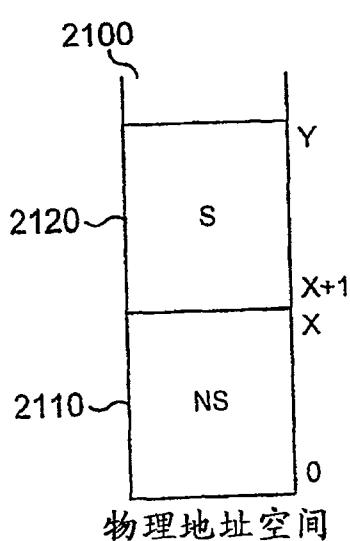
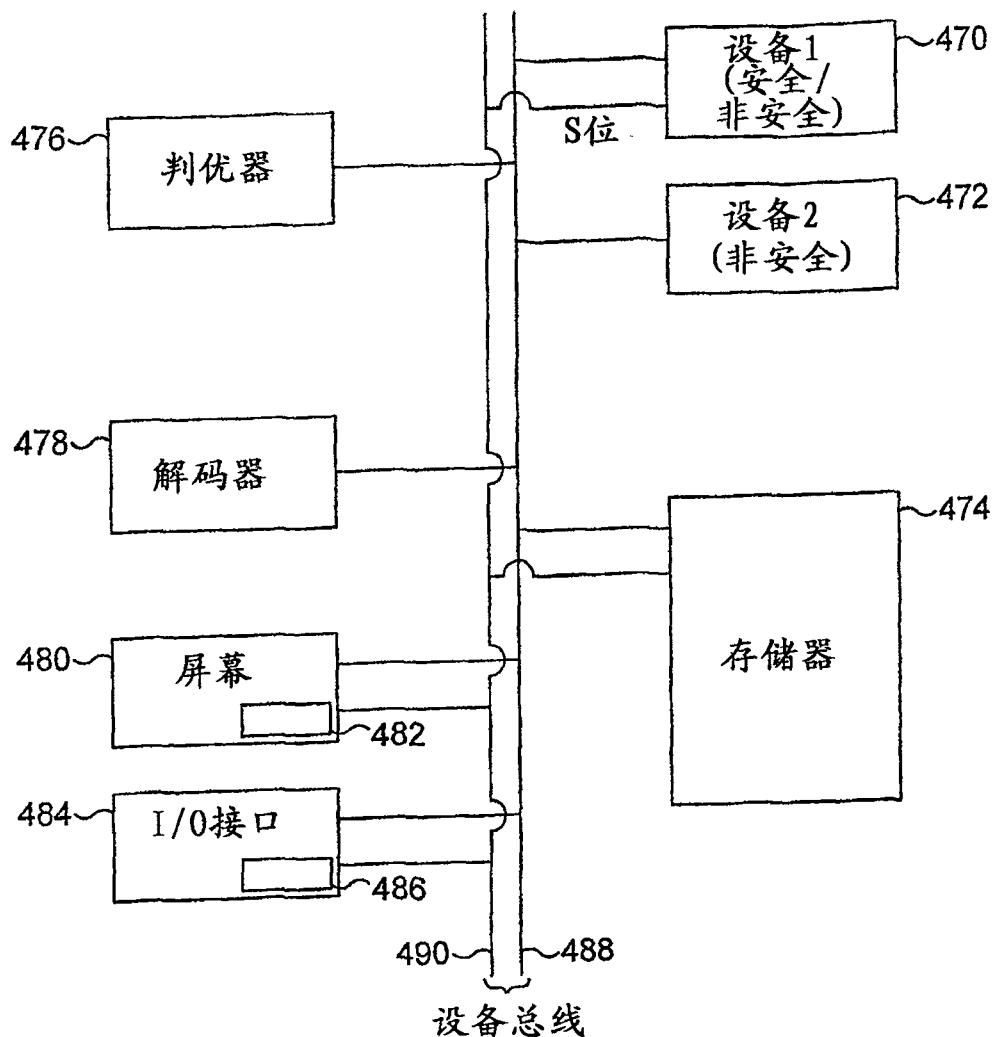


图 47



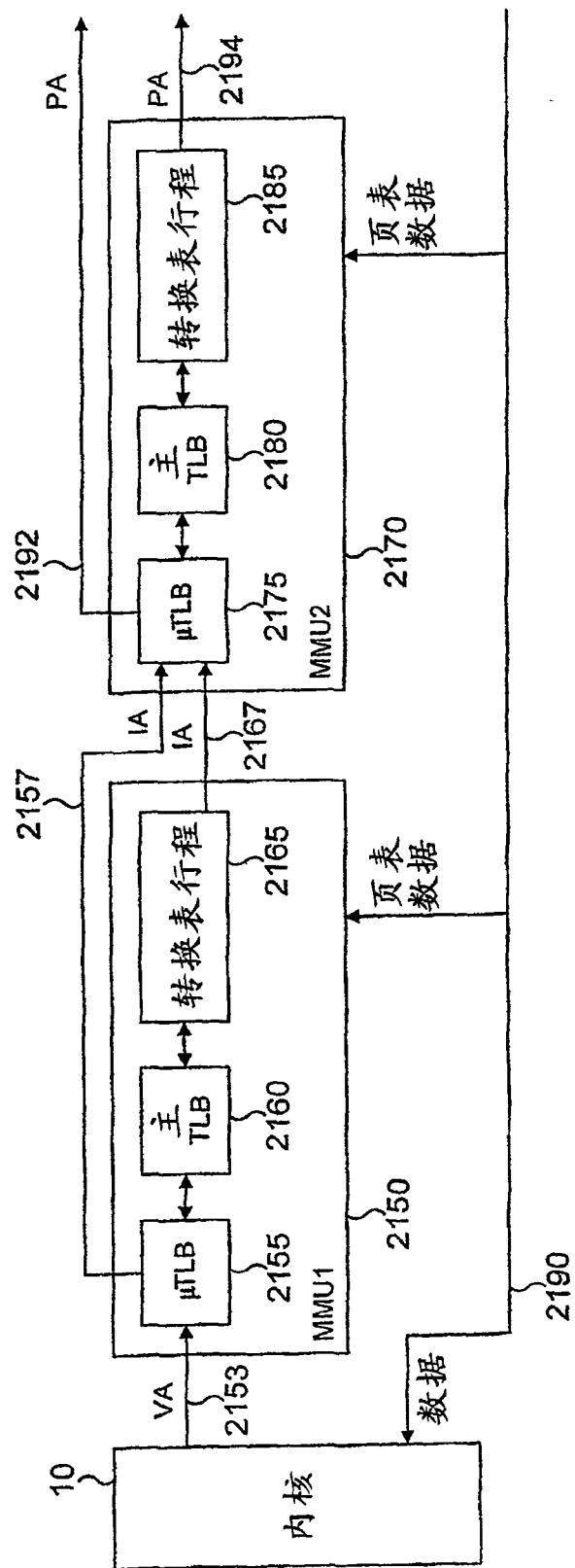


图 50A

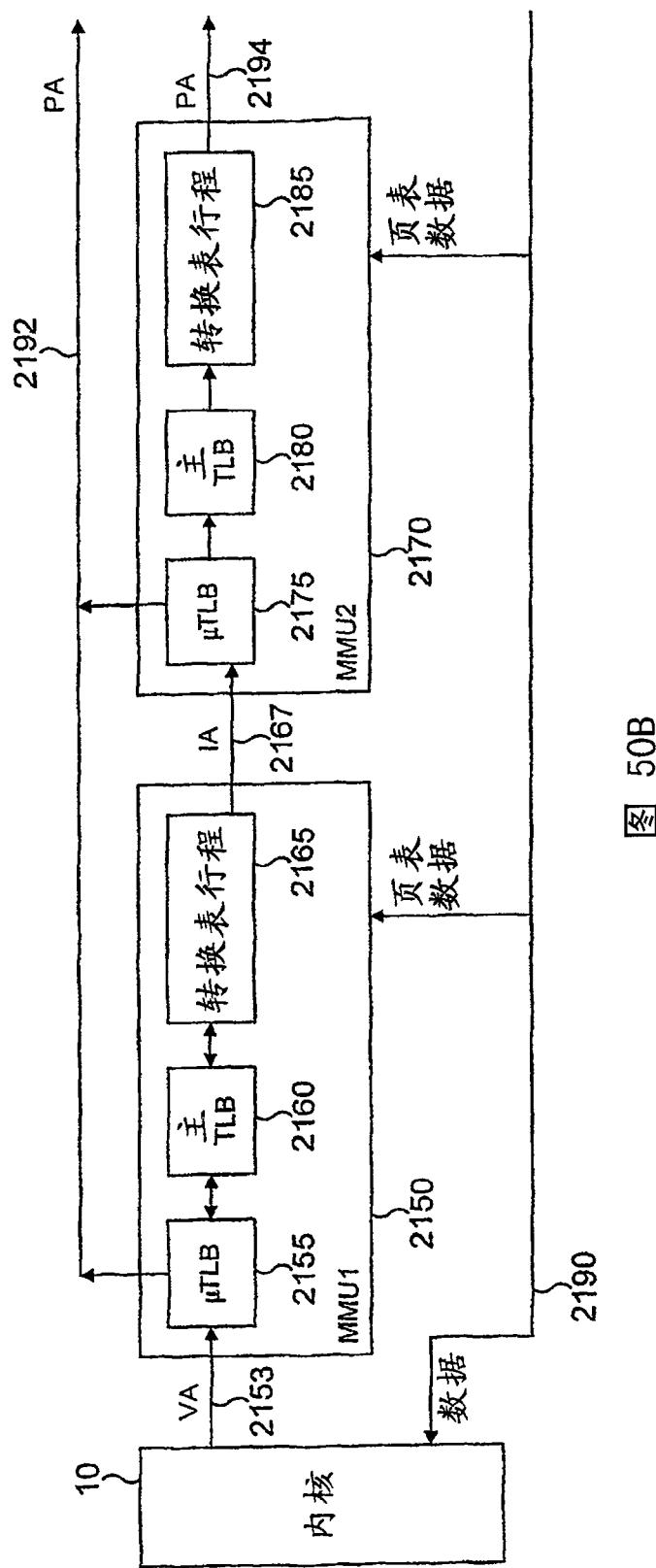
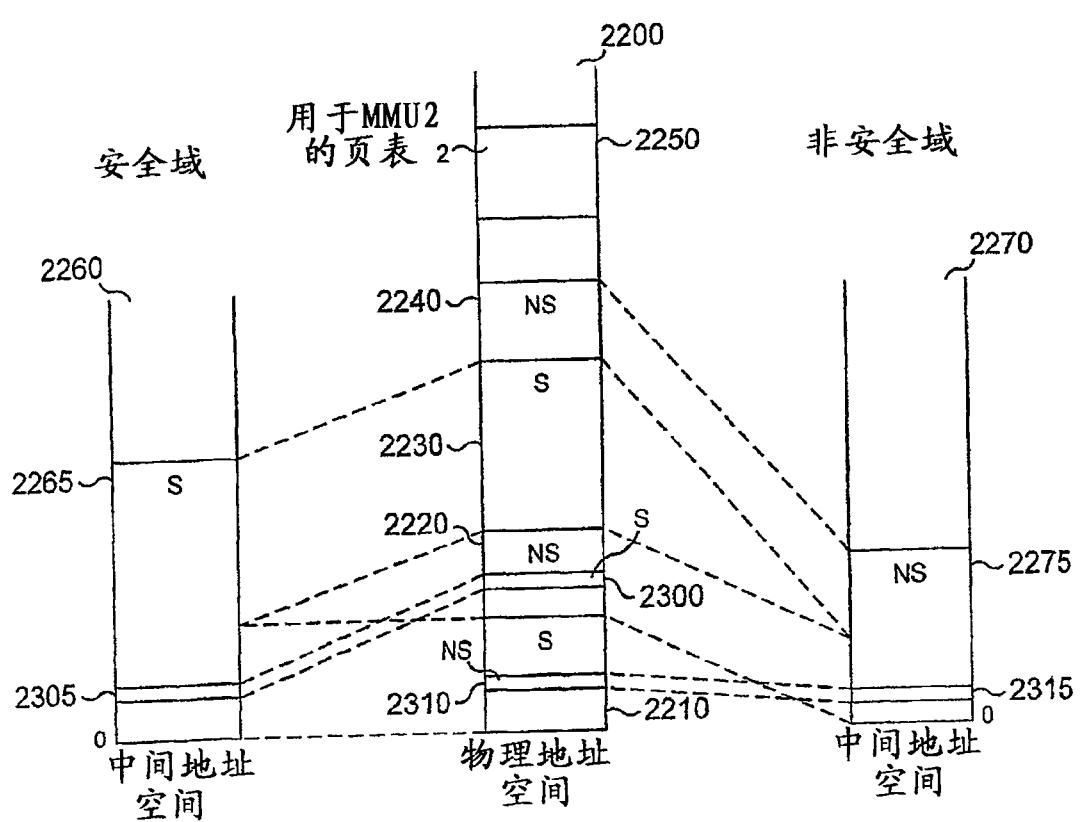
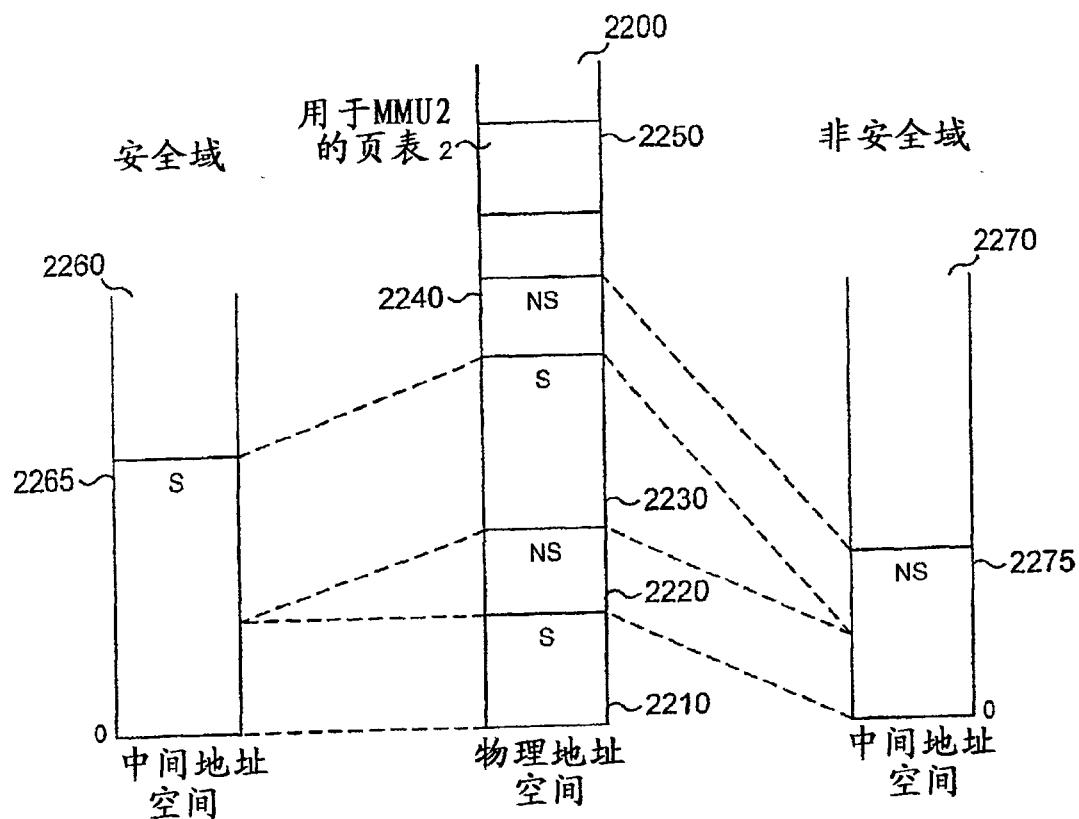


图 50B



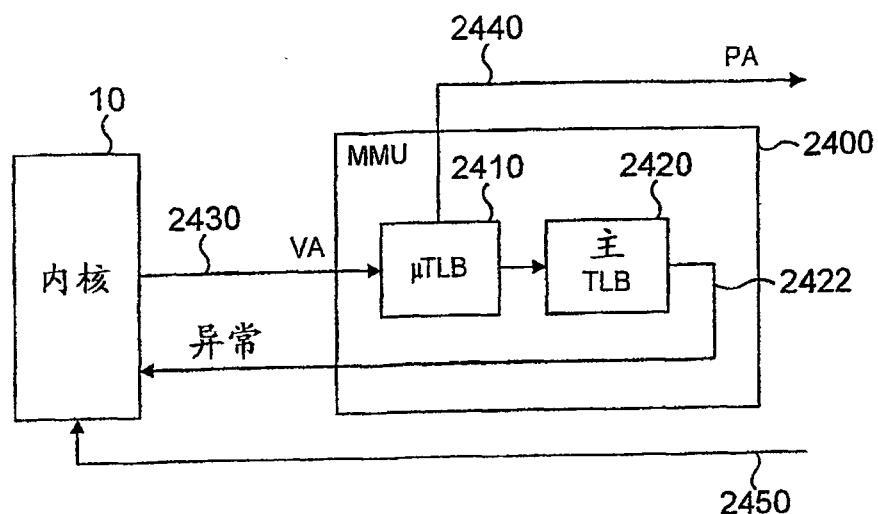


图 53

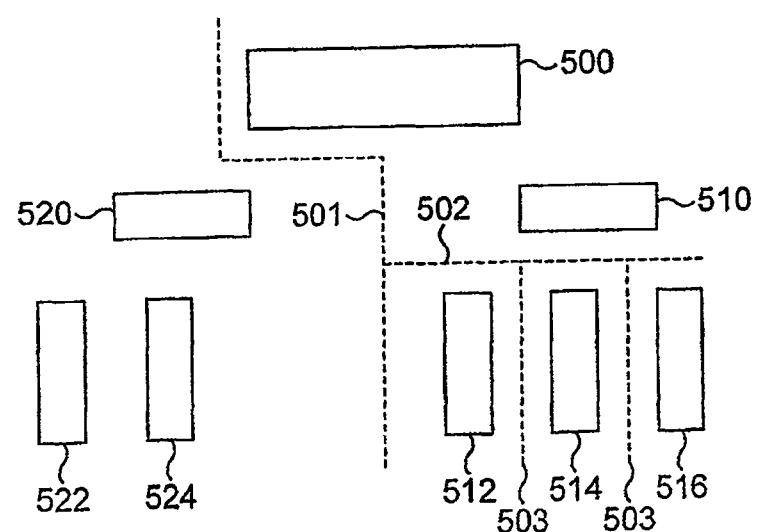


图 59

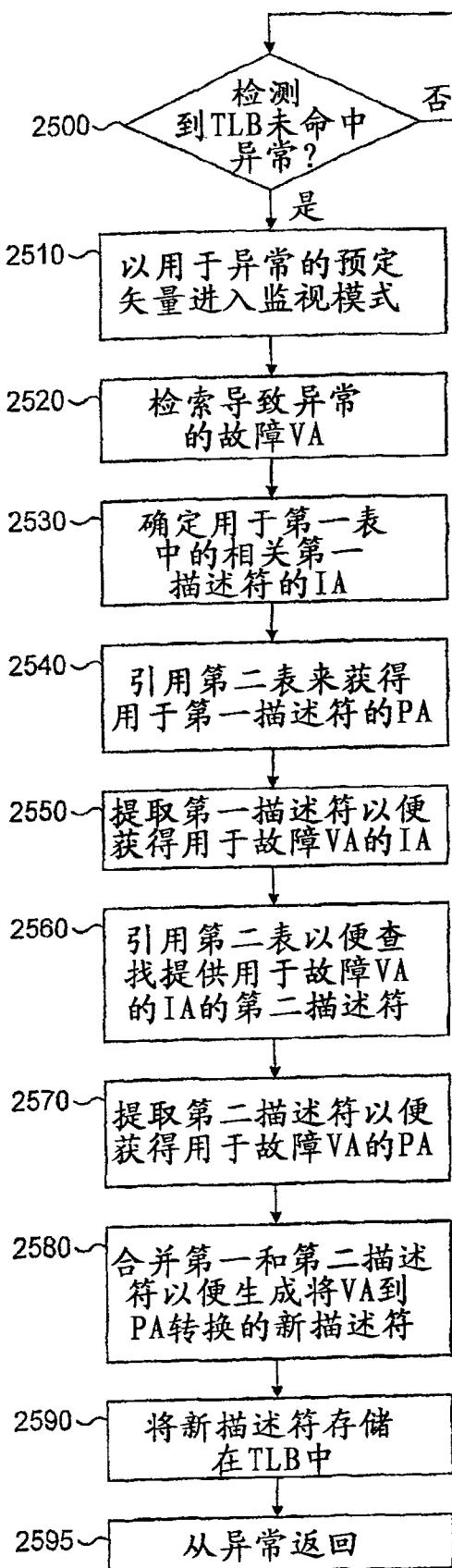


图 54

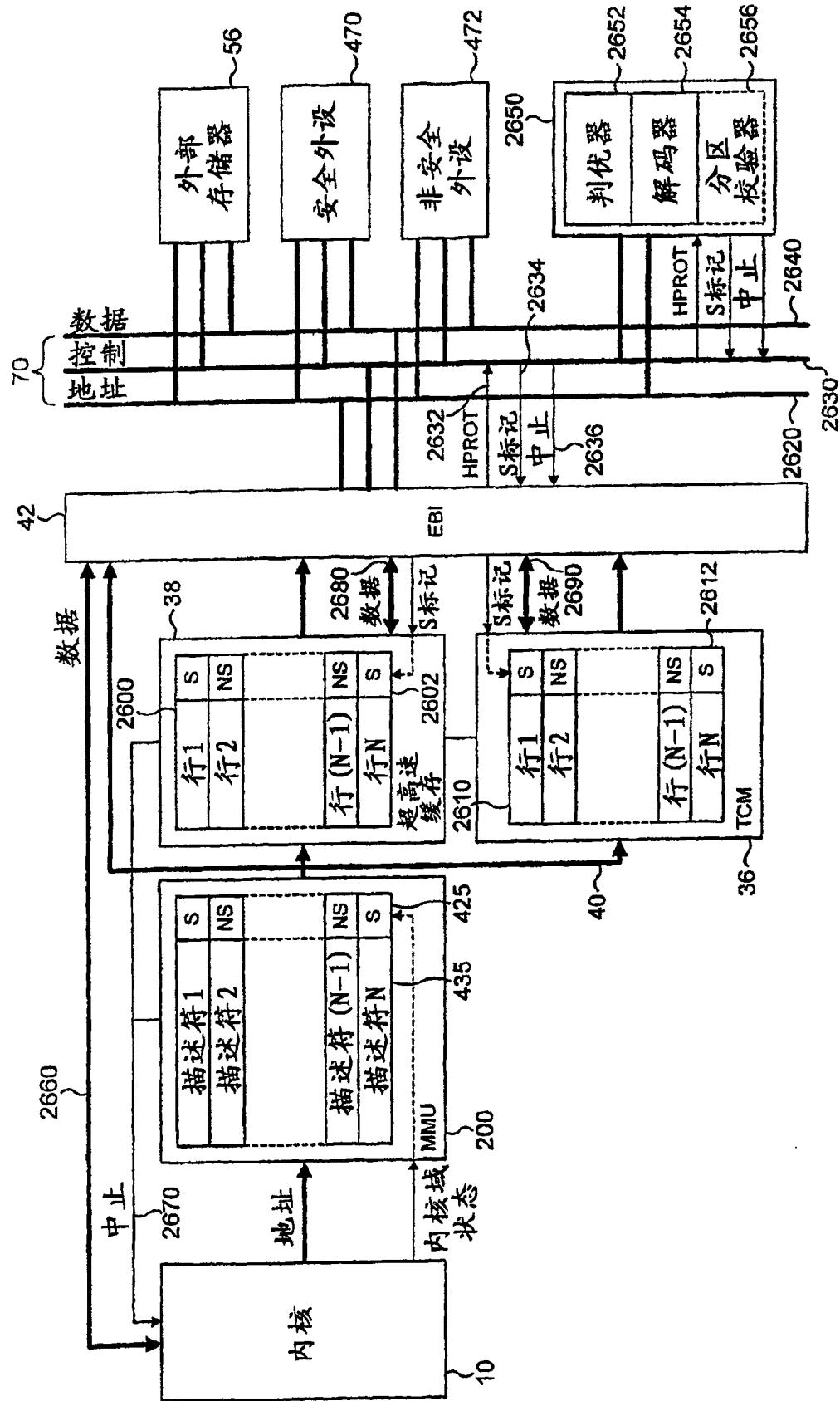


图 55

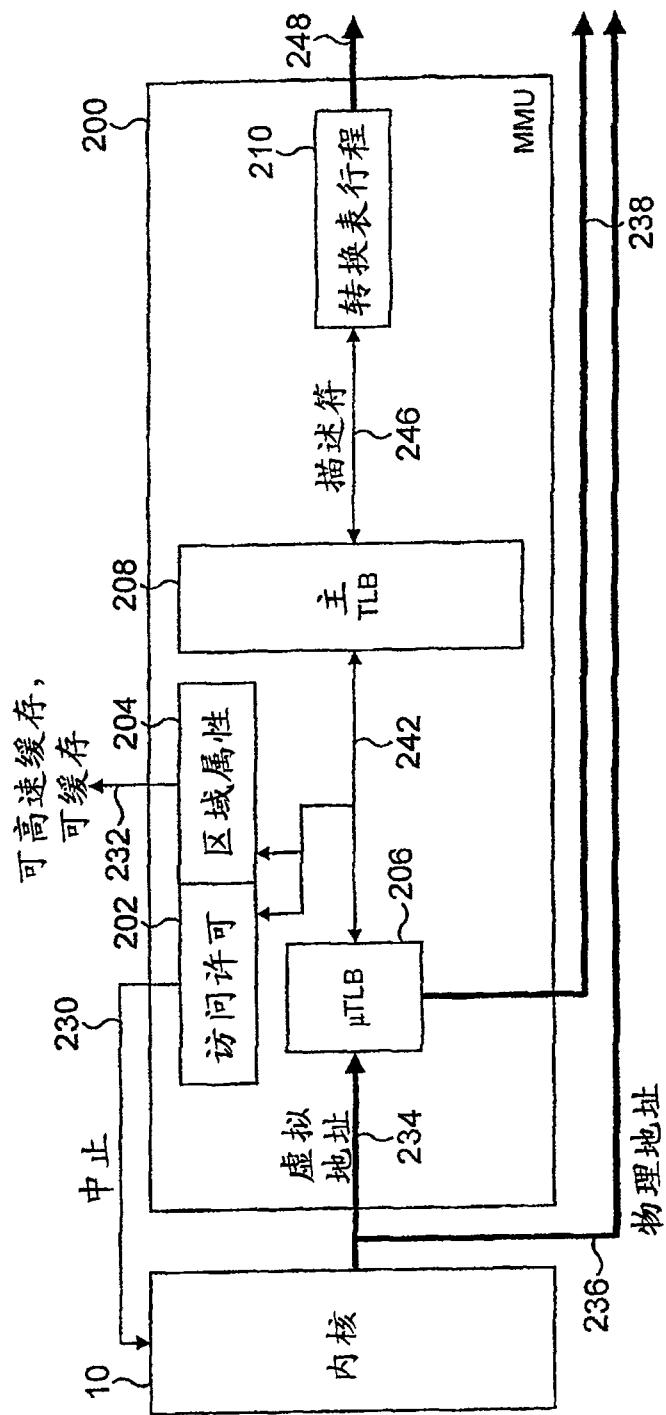
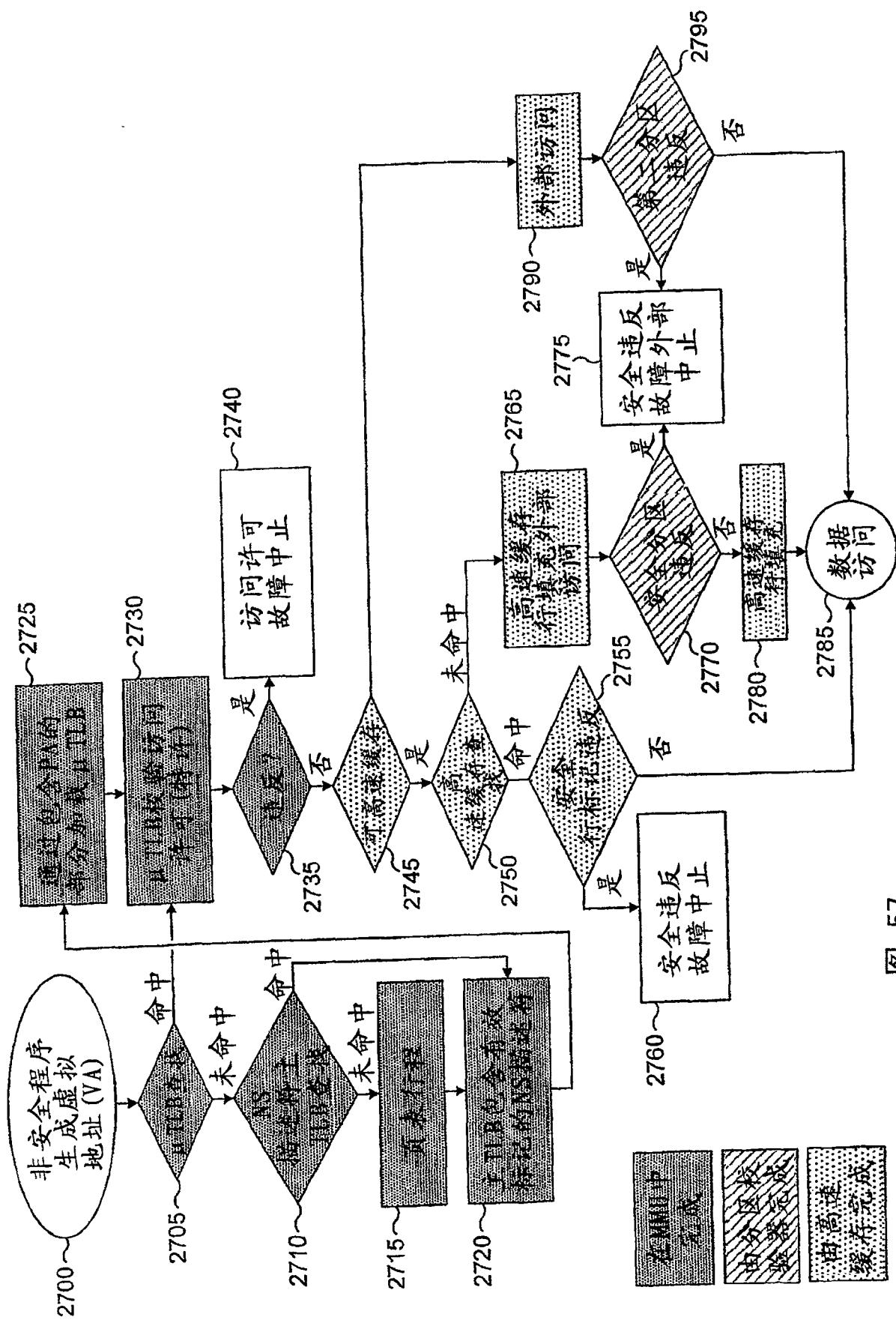
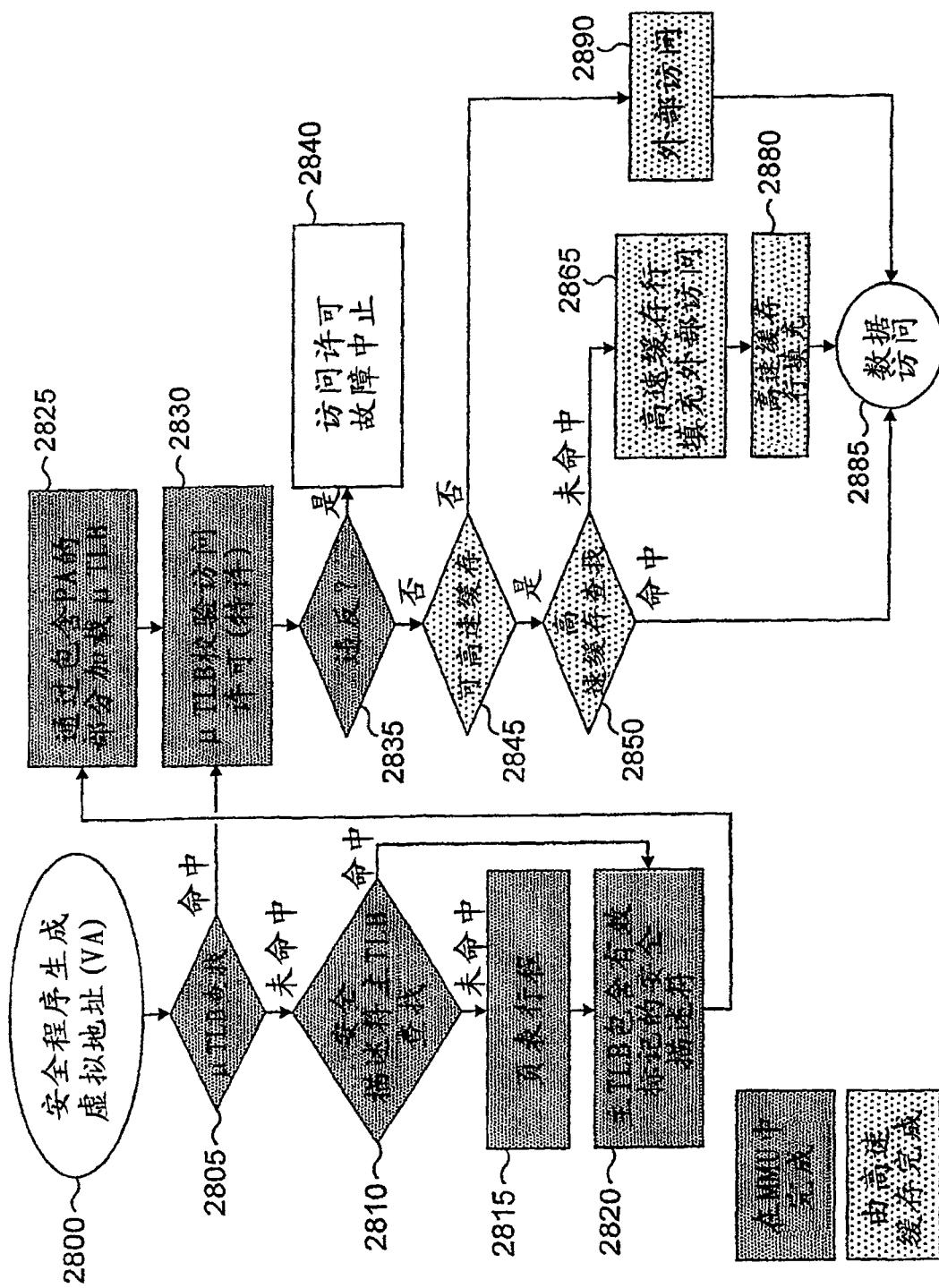


图 56



57



58

输入方法	如何编程	如何进入	进入模式
断点命中	调试TAP或软件 (CP14)	程序断点寄存器和/或上下文 ID寄存器以及通过指令地址和 /或CP上下文ID进行比较 ⁽²⁾	中止/监视 ⁽¹⁾
软件断点指令	通过调试TAP或直接 使用代码中的断点 指令将断点指令放 在扫描链4中(指令 在传送寄存器)	断点指令必须达到执行级	中止/监视
矢量中断断点	调试TAP	程序矢量中断寄存器和 地址匹配	中止/监视
观察点命中	调试TAP或软件 (CP14)	程序断点寄存器和/或上下文 ID寄存器以及通过指令地址和 /或CP上下文ID进行比较 ⁽²⁾	中止/监视 ⁽¹⁾
内部调试请求	调试TAP	已经扫入中止指令	中止
外部调试请求	未写说明	断言EDBGRQ输入管脚	中止

(1): 在监视模式中，断点和观察点不能由数据而定
 (2): 内核已经支持线程识别断点和观察点以便能允许
 一些特定线程上的安全调试

图 60

名称	含义	复位值	访问	插入扫描链, 用于测试
监视模式允许位	0: 中止模式 1: 监视模式	1	通过JTAG编程ICE来R/W(扫描1) 通过使用MRC/MCR指令来R/W(CP14)	是
安全调试允许位	0: 仅在非安全区域中调试 1: 在安全区域和非安全区域中调试	0	在功能模式或调试监视模式: 通过使用MRC/MCR指令(CP14)来R/W(仅在安全监督模式中) 在调试中止模式中: 不访问MCR/MRC指令没有影响(如果JTSDAEN = 1, 通过JTAG编程ICE来R/W)	否
安全跟踪允许位	0: 仅在非安全区域中允许ETM 1: 在安全区域和非安全区域中允许ETM	0	在功能模式或调试监视模式: 通过使用MRC/MCR指令(CP14)来R/W(仅在安全监督模式中) 在调试中止模式中: 不访问MCR/MRC指令没有影响(如果JTSDAEN = 1, 通过JTAG编程ICE来R/W)	否
安全用户模式允许位	0: 在安全模式中, 调试是不可能的 1: 在安全用户模式中调试是可能的	1	在功能模式或调试监视模式: 通过使用MRC/MCR指令(CP14)来R/W(仅在安全监督模式中) 在调试中止模式中: 不访问MCR/MRC指令没有影响(如果JTSDAEN = 1, 通过JTAG编程ICE来R/W)	否
安全线程识别允许位	0: 对特定线程来说, 调试是不可能的 1: 对特定线程来说, 调试是可能的	0	在功能模式或调试监视模式: 通过使用MRC/MCR指令(CP14)来R/W(仅在安全监督模式中) 在调试中止模式中: 不访问MCR/MRC指令没有影响(如果JTSDAEN = 1, 通过JTAG编程ICE来R/W)	否

图 61

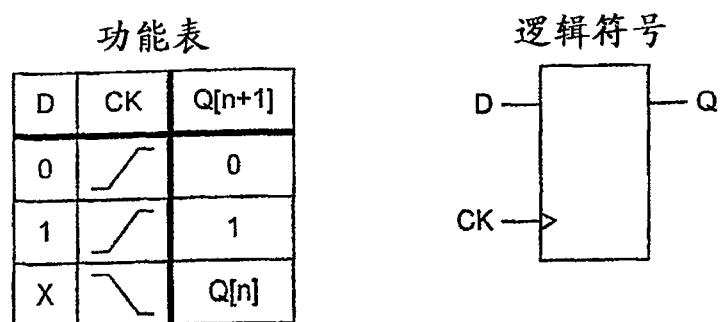


图 62

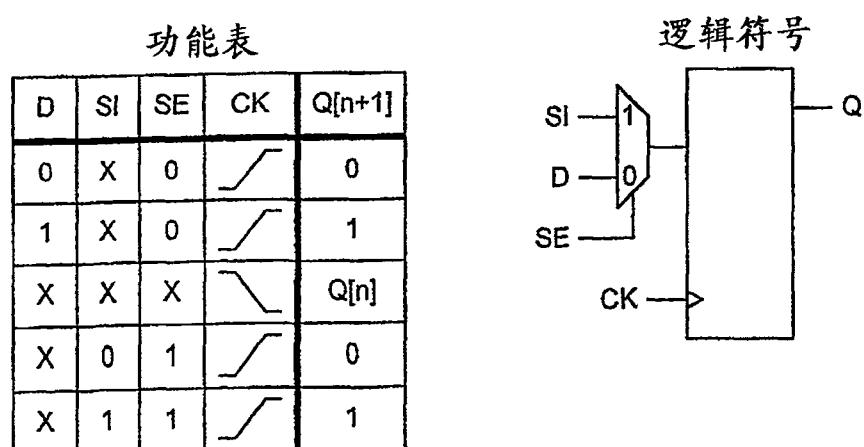


图 63

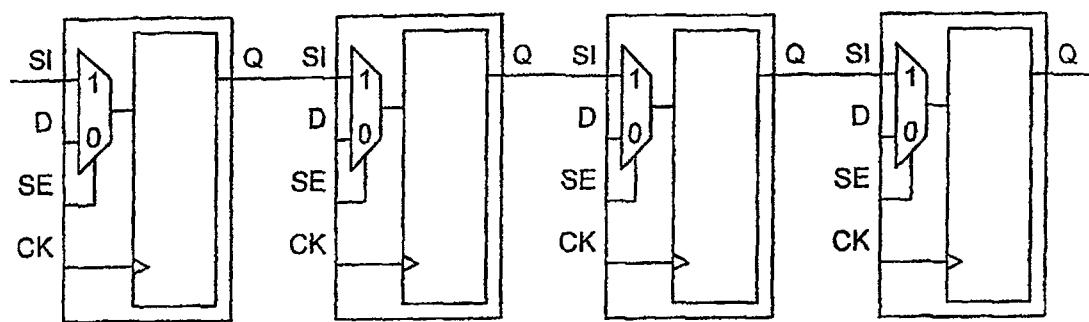


图 64

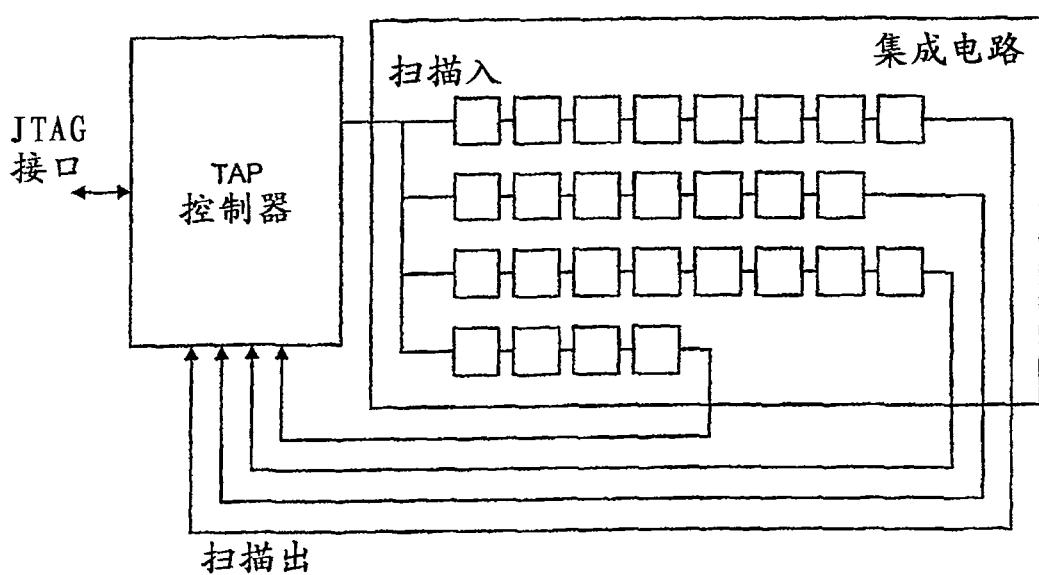


图 65

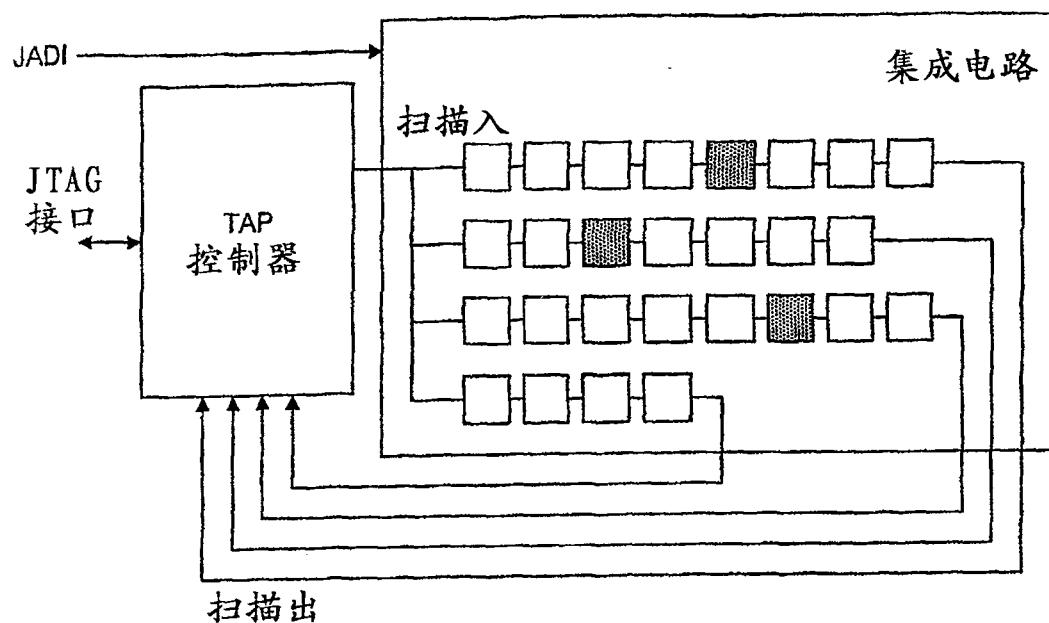


图 66A

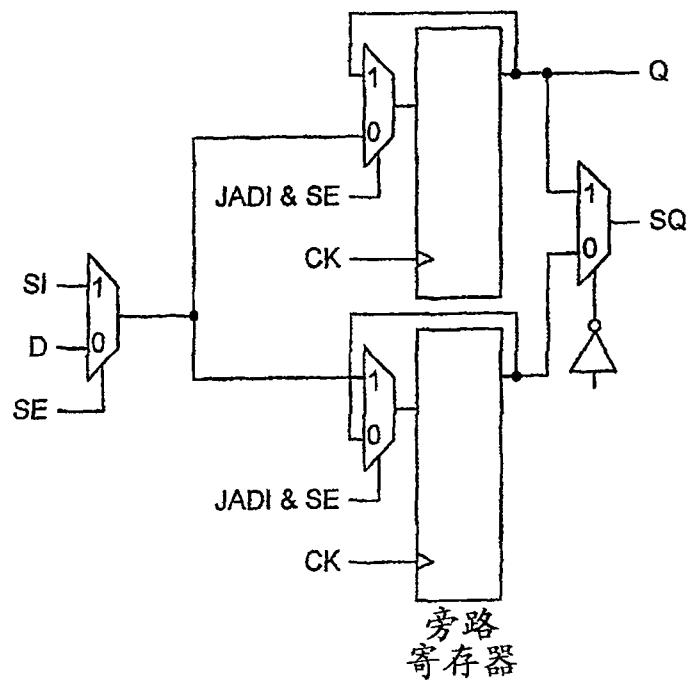


图 66B

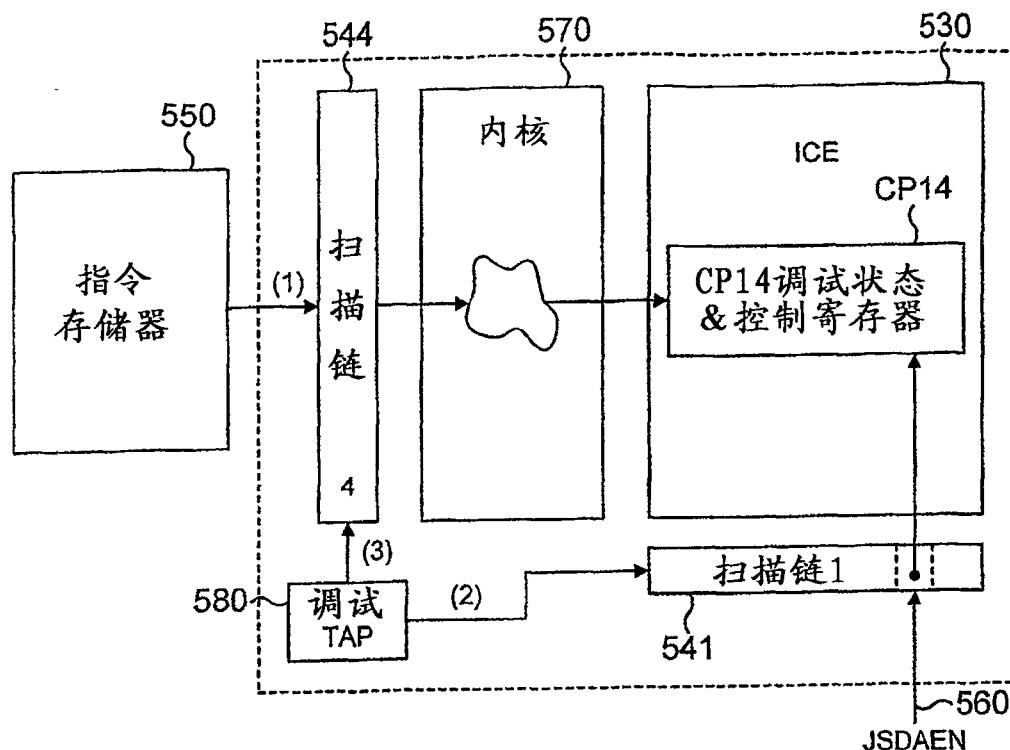


图 67

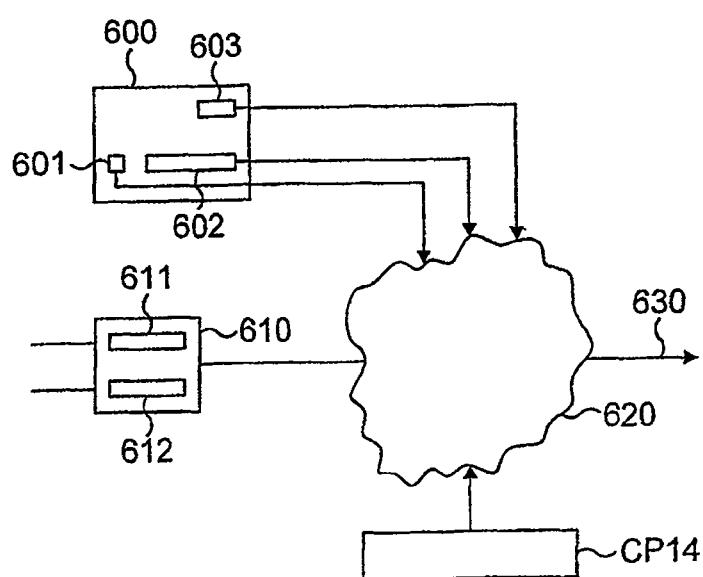


图 68

调试和状态控制寄存器中的CP14位				含义
安全调试 允许位	安全用户模式 调试允许位	安全线程识别 调试允许位		
0	X	X		在整个区域中无插入调试是可能的。在整个安全区域中忽略进入调试状态的任何调试请求、断点、观察点和其他机制
1	0	X		在整个安全区域中调试是可能的
1	1	0		仅在安全用户模式中调试。仅在用户模式中考虑进入调试状态的任何调试请求、断点、观察点和其他机制。(考虑链接或不链接到线程ID的断点和观察点)。
1	1	1		仅在一些特定线程中调试是可能的。在那种情况下，仅考虑链接到线程ID的线程识别状态。此外，调度它自己的代码，以及仅其自己的代码每个线程能

图 69A

调试和状态控制寄存器中的CP14位			含义
安全调试 允许位	安全用户模式 调试允许位	安全线程识别 调试允许位	
0	X	X	在整个区域中无可观察 调试是可能的。跟踪模 块(ETM)不必跟踪内部 内核活动性
1	0	X	在整个安全区域中跟踪 是可能的
1	1	0	仅当内核在安全用户模 式中时, 跟踪是可能的
1	1	1	在安全用户模式中, 仅当 内核正在执行一些特定的 线程时, 跟踪是可能的。 特定的硬件必须专用于此, 或重新使用断点寄存器对: 上下文ID匹配必须允许跟 踪, 而不是进入调试状态

图 69B

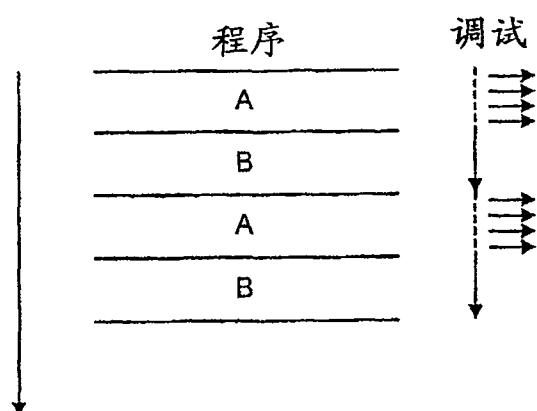


图 70

进入方法	当在非安全区域 中时进入	当在安全区域 中时输入
断点命中	非安全预提取中止 处理程序	安全预提取中止 处理程序
软件断点指令	非安全预提取中止 处理程序	安全预提取中止 处理程序
矢量中断断点	禁止非安全数据 中止和非安全预提 取中止中断。对其 他非安全异常， 预提取中止	禁止安全数据中止 和安全预提取中止 异常。对其他异常， 安全预提取中止
观察点命中	非安全数据中止 处理程序	安全数据中止 处理程序
内部调试请求	暂停模式中的调试状态	暂停模式中的调试状态
外部调试请求	暂停模式中的调试状态	暂停模式中的调试状态

- (1) 查看有关矢量中断寄存器的信息
(2) 注意当断言外部或内部调试请求时，
内核进入中止模式而不是监视模式

图 71A

进入方法	进入非安全区域	进入安全区域
断点命中	非安全预提取中止 处理程序	忽略中断
软件断点指令	非安全预提取中止 处理程序	忽略指令
矢量中断断点	禁止非安全数据中止 和非安全预提取中止 中断。对其他非安全 异常，预提取中止	忽略断点
观察点命中	非安全数据中止 处理程序	忽略观察点
内部调试请求	暂停模式中的调试状态	忽略请求
外部调试请求	暂停模式中的调试状态	忽略请求
从系统速度访问 调试重输入	不可应用	不可应用

- (1) 当从非安全区域替代安全区域中的BKPT指令时，
非安全中止必须处理违反

图 71B