(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102341806 B
(45)授权公告日 2014.09.24

(21)申请号 201080010097.4
(22)申请日 2010.03.01
(30)优先权数据
09290148.7 2009.03.02 EP
(85)PCT国际申请进入国家阶段日
2011.09.01
(86)PCT国际申请的申请数据
PCT/IB2010/050875 2010.03.01
(87)PCT国际申请的公布数据
WO2010/100598 EN 2010.09.10
(73)专利权人 NXP股份有限公司
地址 荷兰艾恩德霍芬
(72)发明人 胡格斯·德普提斯
(74)专利代理机构 中科专利商标代理有限责任公司 11021
代理人 王波波
(51)Int.Cl.
G06F 21/12(2013.01)
G06F 9/318(2006.01)

(56)对比文件
US 20060277530 A1,2006.12.07,
US 20060277530 A1,2006.12.07,
CN 101350055 A,2009.01.21, 全文.
US 7024663 B2,2006.04.04,
US 20040162989 A1,2004.08.19, 全文.
LARIN S Y 等.Compiler-driven cached code compression schemes for embedded ILP processors.《PROCEEDINGS OF THE 32ND. ANNUAL ACM/IEE INTERNATIONAL SYMPOSIUM ON MICROARCHITECTURE 》.1999,83、84.

审查员 张桂华

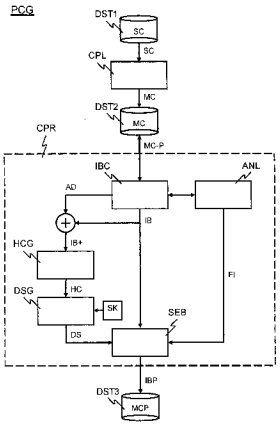
权利要求书2页 说明书14页 附图5页

(54)发明名称

软件保护

(57)摘要

通过以下方式保护可以由可编程电路执行的程序(MC)。基于程序的至少一部分(MC-P)来提供指令块(IB)。产生与指令块(IB)具有预定关系的保护性代码(DS)。分析(ANL)指令块(IB),以标识指令块内的空闲范围(FI),该空闲范围关于指令块的执行是中立的。值范围包括以下至少一种类型:比特范围和值范围。使用已被标识的空闲范围来将保护性代码(DS)嵌入到指令块(IB)中。



CN 102341806 B

1. 一种保护能够通过可编程电路(DPS)执行的程序(MC)的方法,所述方法包括:

-指令块编辑步骤(IBC),在该步骤中,基于所述程序的至少一部分(MC-P)来提供指令块(IB);

-保护性代码产生步骤(HCG、DSG),在该步骤中,产生与所述指令块具有预定关系的保护性代码(DS);

-分析步骤(ANL),在该步骤中,对所述指令块进行分析,以标识所述指令块内的空闲范围(FI),所述空闲范围关于所述指令块的执行是中立的,所述空闲范围包括以下类型中至少一项:比特范围和值范围;以及

-嵌入步骤(SEB),在该步骤中,使用已被标识的空闲范围来将所述保护性代码嵌入到所述指令块中,

所述保护性代码产生步骤(HCG、DSG)包括:

-散列码产生步骤(HCG),在该步骤中,从包括所述指令块(IB)的数据块(IB+)产生散列码(HC);以及

-签名产生步骤(DSG),在该步骤中,从所述散列码(HC)和密钥(SK)产生数字签名(DS),

其中,在所述散列码产生步骤(HCG)中,从其产生散列码(HC)的数据块(IB+)包括表示地址(AD)的数据单元,从所述地址(AD)执行所述指令块中的给定指令。

2. 根据权利要求 1 所述的保护程序的方法,其中,所述指令块编辑步骤(IBC)包括:

-重新格式化步骤(S6),在该步骤中,向要保护的程序(MC)中包括的指令应用熵编码,以创建所述指令块(IB)内的空闲范围。

3. 根据权利要求 1 所述的保护程序的方法,其中,所述指令块编辑步骤(IBC)包括:

-重新格式化步骤(S6),在该步骤中,将值指示包括在经过所述指令块编辑步骤的程序的一部分(MC-P)中可用的空闲范围中,所述值指示对所述部分中的指令的相应的给定比特具有相应的给定值进行指示,由此在所述分析步骤(ANL)中将所述相应的给定比特标识为空闲范围。

4. 根据权利要求 3 所述的保护程序的方法,其中,所述值指示对所述指令包括每个比特都具有给定值的比特组进行指示,所述比特组被标识为空闲范围。

5. 根据权利要求 1 所述的保护程序的方法,其中,所述指令块编辑步骤(IBC)包括:

-伪指令插入步骤(S4),在该步骤中,至少一个伪指令被包括在所述指令块(IB)中。

6. 根据权利要求 1 所述的保护程序的方法,所述指令块(IB)在大小上与高速缓存存储器(CHM)的存取单位相对应,所述高速缓存存储器(CHM)与能够执行所述程序的所述可编程电路(DSP)相关联。

7. 根据权利要求 1 所述的保护程序的方法,所述指令块(IB)包括将要连续执行的指令序列。

8. 一种处理器(PCG),用于保护能够通过可编程电路执行的程序(MC),所述处理器包括:

-指令块编辑器(IBC),被布置为基于所述程序的至少一部分(MC-P)来提供指令块(IB);

-保护性代码产生模块(HCG、DSG),被布置为产生与所述指令块具有预定关系的保护

性代码(DS)；

- 分析模块(ANL)，被布置为对所述指令块进行分析，以标识所述指令块内的空闲范围(FI)，所述空闲范围关于所述指令块的执行是中立的，所述空闲范围包括以下类型中至少一项：比特范围和值范围；以及

- 嵌入模块(SEB)，被布置为使用已被标识的空闲范围来将所述保护性代码嵌入到所述指令块中，

所述保护性代码产生模块(HCG、DSG)还被布置为从包括所述指令块(IB)的数据块(IB+)产生散列码(HC)，以及从所述散列码(HC)和密钥(SK)产生数字签名(DS)，

其中，从其产生散列码(HC)的数据块(IB+)包括表示地址(AD)的数据单元，从所述地址(AD)执行所述指令块中的给定指令。

9. 一种用于处理受保护的程序的方法，所述受保护的程序已通过执行根据权利要求 1 所述的方法而获得，所述处理方法包括：

- 数据分离步骤(SEP)，在该步骤中，提取已经被嵌入到所述指令块(IB)中的保护性代码(DS)；

- 检验步骤(CMP)，在该步骤中，检验所述保护性代码是否与所述指令块具有预定关系；以及

- 指令执行步骤(CPU)，在该步骤中，从所述地址(AD)执行所述指令块中的给定指令。

10. 一种被布置为处理受保护的程序的处理器(DPS)，所述受保护的程序已通过执行根据权利要求 1 所述的方法而获得，所述处理器包括：

- 数据分离器(SEP)，被布置为提取已经被嵌入到所述指令块(IB)中的保护性代码(DS)；

- 检验器(CMP)，被布置为检验所述保护性代码是否与所述指令块具有预定关系；以及

- 指令执行电路(CPU)，被布置为从所述地址(AD)执行所述指令块中的给定指令。

软件保护

技术领域

[0001] 本发明的一方面涉及保护可以由可编程电路执行的程序的方法。例如，可以应用该方法来防止对软件的未授权使用或未授权修改，或者防止对软件的未授权使用以及未授权修改。本发明的其他方面涉及用于保护程序的处理器、包括受保护的程序的数据流、处理受保护的程序的方法、用于处理受保护的程序的处理器以及计算机程序产品。

背景技术

[0002] 典型地，以描述性的高级编程语言来开发软件程序。这种开发（写软件）提供了源代码版本的软件程序。编译器对该源代码版本进行编译，以使得可以获得机器代码版本的软件程序。典型地，机器代码版本是可以由可编程电路执行的指令序列。典型地，机器代码指令是符合具体语法的比特串的形式，例如，32 个比特。

[0003] 一般地，对机器代码版本而不是源代码版本的软件程序进行商业化并相应地公开，将源代码版本保持机密。例如，可以将机器代码与载入了该机器代码的处理器一起商业化。还可以通过例如存储了该机器代码的存储介质的形式来对机器代码单独进行商业化，或者可以通过可从其下载机器代码的服务器的形式来对机器代码单独进行商业化。

[0004] 然而，机器代码容易受到所谓的入侵 (hack)。例如，为了进行未授权的克隆或者未授权的软件升级，进行欺诈的人可以禁止或者绕开在机器代码中包括的安全功能。入侵典型地涉及修改机器代码，以使得可以使执行机器代码的处理器执行不想要的动作。这些动作可以包括泄露与机器代码及机器代码中包括的任何安全功能有关的信息，修改或者绕开安全功能，以及以其他方式修改处理器行为等。通常将这种对机器代码的故意修改称为“攻击”，攻击可以具有物理或者逻辑的特性。物理攻击的示例是将执行机器代码的处理器暴露在强光或者其他类型的辐射下。逻辑攻击的示例是通过处理器内的可寻址缓冲存储器来插入代码。其他逻辑攻击可以涉及使缓冲器溢出或者使用软件缺陷，或者两者都使用。

[0005] 以编号 WO 2005/052795 公布的国际专利申请描述了保护或检验可以在数据处理单元中执行的程序的方法。针对程序中的每个命令产生错误代码或校验码。将校验码加入到每个命令。在数据处理单元中，在执行命令之前，立即执行对校验码的检验。一旦已经确认了命令的真实性，就在数据处理单元中执行该命令。

发明内容

[0006] 存在着对以相对适中的成本可以获得的改进的软件保护的需要。

[0007] 依照本发明的一方面，通过以下方式对可编程电路可以执行的程序进行保护。基于程序的至少一部分来提供指令块。产生与指令块具有预定关系的保护性代码。分析指令块，以使得能够标识指令块内的空闲范围，该空闲范围关于指令块的执行是中立的。值范围包括以下至少一种类型：比特范围和值范围。使用已被标识的空闲范围来将保护性代码嵌入到指令块中。

[0008] 例如，指令可以包括比特串，该比特串对于所述指令块的执行是中立的

(neutral)。亦即,有关比特串对执行指令时发生什么没有任何影响。因此,可以将这种比特串指定为空闲范围。比特串可以构建指令字段。单一比特也可以构建空闲范围。作为另一示例,指令可以包括可容纳值范围的字段。对于指令的执行,该值范围的一部分可以是中立的。亦即,有关值没有特定意义。可以将这种值的集合指定为空闲范围。

[0009] 根据本发明,使用空闲范围来将保护性代码嵌入指令块中,而不是将保护性代码附加到指令块。这使得在数据量上没有实质增加的情况下,或者甚至没有任何增加的情况下,能够保护程序。其消除了对可编程电路中的附加存储器容量的需要,该可编程电路被布置为执行已根据本发明进行保护的机器代码。具有充足容量用于存储给定的常规未保护机器代码程序的给定的程序存储器典型地也具有充足容量来用于存储该程序的受保护版本,该程序的受保护版本是根据本发明获得的。此外,通常不复杂的相对少的修改足以对常规的可编程电路进行适配,允许该电路处理已根据本发明进行保护的机器代码。从而,可以通过相对适中的成本获得软件保护。

[0010] 本发明的实现有利地包括以下的一个或多个附加特征,在与单独的从属权利要求相对应的单独的段落中对这些附加特征进行描述。

[0011] 优选地,将熵编码应用于要保护的程序中包括的指令,以使得可以在指令块内创建空闲范围,在该指令块中嵌入了保护性代码。相应地,指令块包括要保护的程序中出现的指令的熵编码的版本。创建空闲范围的熵编码可以提供充足的空间以用于在不增加任何数据量或者仅相对适中地增加数据量的情况下嵌入保护性代码。这有助于成本效率。

[0012] 此外,可以对指令重新格式化,以创建空闲范围。可以将值指示包括在程序的一部分中可用的空闲范围中,指令块由该程序的该部分组成。值指示对该部分中的指令的相应的给定比特具有相应的给定值进行指示。然后,在分析步骤中,将这些相应的给定比特标识为空闲范围。相应地,可以说,为了获得较大的空闲范围,可以放弃所布置的相对小的空闲。

[0013] 值指示可以对指令包括每个比特都具有给定值的比特组进行指示。然后,将该比特组标识为空闲范围。

[0014] 可以将至少一个伪指令(dummy instruction)包括在指令块中。可以说,这使得可以确保指令块中有充足的空间用于嵌入保护性代码。然而,一般优选地对指令重新格式化,以获得充足的空间。

[0015] 可以如下方式产生保护性代码:从包括指令块的数据集合产生散列码。从散列码和密钥产生数字签名。所产生的保护性代码可以检验完整性和真实性。

[0016] 从其产生散列码的数据集合优选地包括表示地址的数据单元,应该从该地址执行指令块中的给定指令。这防止入侵者(hacker)移动有效指令,增强了保护。

[0017] 指令块可以在大小上与高速缓存存储器的存取单位相对应,该高速缓存存储器与能够执行程序的可编程电路相关联。

[0018] 指令块可以包括要连续执行的指令序列。这意味着指令块基于程序中的一部分,程序中的该部分免于在该有关部分结束之前出现的任何跳转(jump)。

[0019] 参考附图的详细描述示出了之前所总结的本发明以及附加特征。

附图说明

[0020] 图 1 是示出保护代码产生器的方框图。

[0021] 图 2A、2B 和 2C 是示出不同类型的机器代码指令的比特图，可以在保护代码产生器中对机器代码指令进行保护。

[0022] 图 3 是示出为了形成将被保护的指令块而在保护代码产生器中执行的一系列步骤的流程图。

[0023] 图 4 是示出布置为执行保护代码产生器已经产生的机器代码的数据处理系统的方框图。

[0024] 图 5 是示出形成数据处理系统的一部分的安全模块的方框图。

[0025] 图 6 是示出数据处理系统的备选处理器的方框图。

具体实施方式

[0026] 图 1 示出了保护代码产生器 PCG。保护代码产生器 PCG 包括编译器 CPL、代码保护器 CPR 和各种数据存储空间 DST1、DST2、DST3。编译器 CPL 和代码保护器 CPR 可以通过例如适当编程的处理器的方式来各自实现。数据存储空间 DST1、DST2、DST3 可以形成单个物理存储介质（例如，如硬盘或者固态存储器电路）的一部分。作为另一示例，可以将每一个数据存储空间包括在单独的物理存储介质中。

[0027] 代码保护器 CPR 包括各种功能模块：指令块编辑器 IBC、分析器 ANL、散列码产生器 HCG、数字签名产生器 DSG 和签名嵌入模块 SEB。可以通过例如已经载入到可编程处理器中的指令集的方式来实现这些功能模块中的每一个。在这种基于软件的实现中，指令集定义了相关功能模块所执行的操作，稍后将对其进行描述。在这方面，可以将图 1 视为表示一种方法，从而编译器 CPL 表示编译步骤，指令块编辑器 IBC 表示指令块编辑步骤，分析器 ANL 表示分析步骤，散列码产生器 HCG 表示散列码产生步骤，数字签名产生器 DSG 表示数字签名产生步骤，以及签名嵌入模块 SEB 表示签名嵌入步骤。

[0028] 保护代码产生器 PCG 基本上如下操作。数据存储空间 DST1 包括软件程序的源代码版本 SC。编译器 CPL 对该源代码版本 SC 进行编译，以获得软件程序的机器代码版本 MC。数据存储空间 DST2 至少临时地存储该机器代码版本 MC。此后，为了简明，将软件程序的源代码版本 SC 和机器代码版本 MC 分别称为源代码 SC 和机器代码 MC。

[0029] 典型地，对机器代码 MC 而不是源代码 SC 进行商业化并相应地公开，可以将源代码 SC 保持机密。例如，可以将机器代码 MC 与已经载入该机器代码 MC 的处理器一起进行商业化。还可以通过例如存储有该机器代码 MC 的存储介质的形式来对机器代码 MC 单独进行商业化，或者可以通过可从其下载机器代码 MC 的服务器的形式来对机器代码 MC 单独进行商业化。

[0030] 然而，机器代码 MC 容易受到所谓的入侵。例如，为了进行未授权的克隆或者未授权的软件升级，进行欺诈的人可以禁止或者绕开在机器代码 MC 中包括的安全功能。入侵典型地涉及修改机器代码 MC 以使得可以使执行机器代码 MC 的处理器执行不想要的动作。这些动作可以包括泄露与机器代码 MC 以及机器代码 MC 中包括的任何安全功能有关的信息，修改或者绕开安全功能，以及修改处理器行为等。通常将这种对机器代码的故意修改称为“攻击”，攻击可以具有物理或者逻辑的特性。物理攻击的示例是将执行机器代码 MC 的处理器暴露在强光或者其他类型的辐射下。逻辑攻击的示例是通过处理器内的可寻址缓冲存储器来插入代码。其他逻辑攻击可以涉及使缓冲器溢出或者使用软件缺陷，或者两者都使用。

图 1 中示出的保护代码产生器 PCG 提供了以有效的方式对抗这种入侵的保护。

[0031] 指令块编辑器 IBC 基于机器代码 MC 的连续部分 MC-P 提供连续的指令块 IB，该连续部分 MC-P 出现在数据存储空间 DST2 中。机器代码部分 MC-P 自身是指令序列。指令块 IB 可以准确地对应于机器代码部分 MC-P。亦即，指令块 IB 与机器代码部分 MC-P 可以相同。指令块 IB 还可以对应于已经添加有一个或多个伪指令 (dummy instruction) 的机器代码部分 MC-P。典型地，伪指令是所谓的“nop”指令，“nop”是针对无操作的助记符号。指令块 IB 还可以包括这样的指令 :指令块 IB 所基于的机器代码部分 MC-P 中包括的指令的修改版本。亦即，指令块编辑器 IBC 可以对形成机器代码 MC 的一部分的一个或多个指令进行重新格式化。该重新格式化可以涉及熵编码，具体地，涉及指令中所谓的操作码 (opcode) 的熵编码。此后将对其进行更详细的描述。

[0032] 指令块 IB 可以包括固定数目的指令，例如，8 个指令。在该情况下，指令块 IB 的大小优选地与高速缓存存储器的存取单位相对应。典型地，这种存取单位是所谓的高速缓存行 (cache line)，其可以容纳给定数目的指令，然而，这种存取单位也可以是例如多个高速缓冲行。备选地，指令块 IB 可以包括可变数目的指令。例如，指令块 IB 可以包括在有关的软件程序的两个分支之间包括的指令序列。该软件程序在两个分支之间典型地有 7 至 10 个指令。

[0033] 分析器 ANL 控制指令块编辑器 IBC，以确保指令块 IB 可以容纳给定的最小数量的附加数据。这意味着分析器 ANL 决定指令块编辑器 IBC 是否应该将一个或多个伪指令添加到机器代码部分 MC-P，以提供指令块 IB。总的来说，保持指令块 IB 中包括的伪指令的数目越多，指令块 IB 可以容纳的附加数据的数量越大。

[0034] 然而，增加伪指令必然带来缺陷，并因此应该尽可能地避免。首先，将获得机器代码 MC 的扩展版本，该机器代码 MC 的扩展版本在需要执行机器代码（确切地说，机器代码的扩展版本）的处理器中需要更大的存储空间。其次，在增加的伪指令之后的任何跳转指令都应该被修改。跳转地址应该被改动。这是相当麻烦的过程，其必然带来附加的成本，并且可能成为错误的来源。

[0035] 分析器 ANL 提供向指令块 IB 应用的空闲范围指示 FI，指令块 IB 是指令块编辑器 IBC 所提供的。空闲范围指示 FI 指示指令块 IB 内可以容纳附加数据的所谓的空闲范围。例如，空闲范围可以是指令中其相应值对有关指令的执行没有影响的比特的范围。亦即，对于有关的相应比特的相应值，指令的执行是无关紧要的。因此，分析器 ANL 可以将构建这种“不关心”值的比特标注为空闲范围。在指令中的连续比特构建了“不关心”值的情况下，分析器 ANL 可以将这些连续的比特共同标注为空闲范围。

[0036] 例如，空闲范围还可以是值范围，可以将值范围分配给指令中的连续比特的组，然而其对有关指令的执行没有影响。亦即，在来自该范围的值没有具体含义这一方面来说，该值范围是不使用的。从而，对于来自有关值范围的任何具体的值来说，指令的执行是无关紧要的。因此，分析器 ANL 可以将这种不使用的值的范围标注为空闲范围。例如，假设指令包括 6 比特字段，针对该 6 比特字段已经定义了 29 个不同的值。由于 6 比特对应于 64 个不同的值，这意味着有 35 个值没有得到有效使用。这 35 个未使用的值构建了可以容纳价值稍多于 1 比特的信息的空闲范围。此后，将更详细地描述分析器 ANL 执行的用于标识空闲范围的操作。

[0037] 散列码产生器 HCG 将散列函数应用到互补指令块 IB+。通过将表示地址 AD 增加到指令块 IB 来获得互补指令块 IB+，这是由指令块编辑器 IBC 提供的。表示地址 AD 表示指令块 IB 内的指令的地址，应该从该地址执行有关指令。例如，表示地址 AD 可以是应该从其执行指令块 IB 中的第一指令的地址，或者仅是该地址的一部分。

[0038] 指令块编辑器 IBC 可以将表示地址 AD 确定为与机器代码 MC 的开始地址相关的地址。例如，指令块 IB 的表示地址 AD 可以与指令块编辑器 IBC 目前为止基于机器代码 MC 已提供的所有指令块 IB 中包括的指令的数目相对应。表示地址 AD 还可以是这种相关地址的一部分，例如，相关地址的 20 个最低有效位 (LSB)。优选地，表示地址具有这样的粒度 (granularity)：大到足以防止入侵，同时允许用于在存储器内部移动指令的一定余量。

[0039] 散列码产生器 HCG 提供通过向互补指令块 IB+ 应用散列函数而产生的散列码 HC。散列码 HC 包括与互补指令块 IB+ 中包括的比特的数目无关的固定数目的比特，互补指令块 IB+ 中包括的比特的数目可以是固定的或者可变的。例如，假设散列函数是已知使用的“SHA-2”的散列函数 (SHA 是安全散列算法的首字母缩写)。在这种情况下，散列码 HC 包括与互补指令块 IB+ 中包括的指令的数目无关的 256 个比特，互补指令块 IB+ 中包括的指令的数目可以是例如 8 个或 9 个。

[0040] 优选地，散列函数具有加密类型，这意味着散列函数是单向并且无冲突的。单向意味着在计算上找到输入“x”使得 H(x) 等于散列码 HC 是不可行的，H 表示散列函数。无冲突意味着在计算上找到不同于互补指令块 IB+ 的代码“y”使得 H(y) 等于散列码 HC 是不可行的。从而，将 H 称为弱的无冲突散列函数。强的无冲突散列函数 H 是这样的函数：对于该函数，在计算上找到任意的两个消息 x 和 y 使得 H(x) ＝ H(y) 是不可行的。因此，可以将散列码 HC 视为互补指令块 IB+ 的数字指纹。

[0041] 数字签名产生器 DSG 将签名算法应用于散列码 HC，以基于密钥 SK 产生数字签名 DS。密钥 SK 是安全地存储在保护代码产生器 PCG 中的私密的数字代码。在没有密钥 SK 的情况下，产生数字签名 DS 在计算上是不可行的。从而，数字签名 DS 使得可以验证互补指令块 IB+ 的真实性和完整性。亦即，数字签名 DS 可以保证互补指令块 IB+ 的来源 (authorship) 并保证其中包括的数据未受到任何修改。优选地，数字信号产生器应用非对称加密方案，这意味着存在公 - 私钥对。在使用这种方案的情况下，密钥 SK 是私钥，其具有对应的公钥。公钥允许从数字签名 DS 产生散列码 HC，而不是其他的方式。

[0042] 签名嵌入模块 SEB 将数字签名 DS 嵌入到指令块 IB 中，数字签名 DS 从该指令块 IB 中产生。为此，签名嵌入模块 SEB 使用分析器 ABL 提供的空闲范围指示 FI，空闲范围指示 FI 已标识出了指令块 IB 中存在的空闲范围。如上文所述，分析器 ANL 对指令块编辑器 IBC 进行控制，以使得指令块 IB 可以容纳数字签名 DS。亦即，可以说，分析器 ABL 确保了指令块 IB 中的空闲范围提供了充足的空间以用于成功嵌入数字签名 DS。下文将对其进行更详细的解释。

[0043] 从而，图 1 中示出的代码保护器 CPR 从机器代码 MC 的连续部分 MC-P 产生连续的受保护的指令块 IBP。受保护的指令块 IBP 是已经如前所述在其中嵌入了数字签名的指令块 IB。将代码保护器 CPR 产生的连续的受保护的指令块 IBP 写入数据存储空间 DST3 中。连续的受保护的指令块 IBP 共同构建了机器代码 MC 的受保护版本，亦即受保护的机器代码 MCP。

[0044] 图 2A、2B 和 2C 示出了将机器代码 MC 用于 MIPS 处理器 (MIPS 是无互锁流水线级微处理器的首字母缩写，并且是美国的 MIPS Technologies，Inc. 的注册商标）的情况下，机器代码 MC 可以包括的三种不同类型的指令。图 2A 示出了 R 类型的指令；图 2B 示出了 I 类型的指令；以及图 2C 示出了 J 类型的指令。每种类型的指令包括 32 个比特，由在图 2A、2B 和 2C 中水平方向上布置的 0 到 31 之间包括的范围中的数字来表示。数字 31 表示指令的最高有效位；数字 0 表示最低有效位。

[0045] R 类型、I 类型和 J 类型的指令每个都包括操作码 OP，其范围在比特 26 至比特 31（最高有效位）之间。以及，操作码 OP 构建了 6 比特字段。指令的操作码 OP 包括对有关指令进行标识的 6 比特值。字节 26-31 是指令标识比特。

[0046] 除了操作码 OP 之外，图 2A 中示出的 R 类型指令包括各种字段：范围从比特 21 至 25 的 RS 字段，范围从比特 16 至 20 的 RT 字段，范围从比特 11 至 16 的 RD 字段，范围从比特 6 至 10 的 SA 字段，以及范围从比特 0 至 5 的 FU 字段。R 类型指令的操作码 OP 是 000000，这意味着比特 31 至 26 每个都等于 0。FU 字段对功能进行了定义。有 29 种不同功能。由于 FU 字段包括 6 个比特，6 个比特对应于 64 个不同的值，这意味着有 35 个值没有有效得以使用。相应地，FU 字段包括可以容纳稍多于 1 比特的信息量的空闲范围。以下提供了针对 FU 字段可以指定的各种功能而能够发现的空闲范围的一些示例。

[0047] 在 FU 字段指定 RS、RT 和 RD 字段借以构建操作数的“and”功能的情况下，不使用 SA 字段。因此，在该情况下，SA 字段构建了可以容纳 5 比特信息的空闲范围。

[0048] 在 FU 字段指定“break”功能（“break”是中断点 (Breakpoint) 的助记符号）的情况下，不使用字段 RS、RT、RD 和 SA。因此，在该情况下，上述字段构建了可以容纳 20 比特信息的空闲范围。

[0049] 在 FU 字段指定 RS 和 RT 字段借以构建操作数的“div”功能（“div”是划分字 (Divide Word) 的助记符号）的情况下，不使用 RD 和 SA 字段。因此，在该情况下，上述字段构建了可以容纳 10 比特信息的空闲范围。

[0050] 除了操作码 OP 之外，图 2B 中示出的 I 类型指令包括各种字段：范围从比特 21 至 25 的 RS 字段，范围从比特 16 至 20 的 RT 字段，范围从比特 15 至 0 的 IM 字段。除了以下值之外，I 类型的操作码 OP 可以具有任意的 6 比特值：000000、00001* 和 0100**，* 表示任意比特值，其可以是 0 或者 1。以下提供了针对操作码 OP 可以指定的各种操作而能够发现的空闲范围的一些示例。

[0051] 在操作码 OP 指定 RS、RT 和 IM 字段借以构建操作数的“andi”操作（“andi”是直接和 (And Immediate) 的助记符号）时，不存在任何未使用的比特或值。因此，在该情况下，不存在任何空闲范围。

[0052] 在操作码 OP 指定 RS 和 IM 字段借以构建操作数的“bgez”操作（“bgez”是大于或等于零的分支的助记符号）时，RT 字段未使用。因此，在该情况下，RT 字段构建了可以容纳 5 比特信息的空闲范围。

[0053] 类似地，在操作码 OP 指定 RS 和 IM 字段借以构建操作数的“lui”操作（“lui”是直接向上载入 (Load Upper Immediate) 的助记符号）时，RT 字段未使用。因此，在该情况下，RT 字段构建了可以容纳 5 比特信息的空闲范围。

[0054] 除了操作码 OP 之外，图 2C 中示出的 J 类型指令包括 TG 字段。TG 字段范围从比特

25 至 0。比特 1 和 0 总是等于 1，因为 J 类型指令是字对齐的。因此，J 类型指令包括至少一个空闲范围，其可以容纳 2 比特的信息。

[0055] 图 3 示出了指令块编辑器 IBC 和分析器 ANL 出于以下两个目的能够执行的一系列步骤 S1-S6。首先，提供了指令块 IB，在指令块 IB 中有足够的空间容纳将针对该指令块 IB 而产生的数字签名 DS。第二，与指令块 IB 一起，提供了空闲范围指示 FI，空闲范围指示 FI 指示了构建上述空间的空闲范围。如上文所指出，可以通过可编程处理器的方式实现指令块编辑器 IBC 和分析器 ANL。从而，可以将图 3 视为软件程序的流程图表示，亦即，使得可编程处理器能够执行下文参考图 3 描述的各种操作的指令集。

[0056] 在步骤 S1 中，指令块编辑器 IBC 从数据存储空间 DST2 中取出机器代码部分 MC-P(MC-P → IBC)。典型地，所取出的机器代码部分 MC-P 在基于其提供最近的指令块的机器代码部分之后。机器代码部分 MC-P 构建了临时指令块 IB_{TMP} 的初始版本，该临时指令块 IB_{TMP} 的初始版本将经历此后描述的步骤 S1-S5(MC-P ＝ IB_{TMP})。

[0057] 在步骤 S2 中，分析器 ANL 确定可以在临时指令块 IB_{TMP} 中容纳的附加数据的量（$ANL：SP \subset IB_{TMP}$）。此后，出于方便考虑，将可以容纳的附加数据的量称为可用空间。分析器 ANL 可以通过以下方式确定可用空间。分析器 ANL 可以基于指令的操作码 OP 或者指令中包括的另一代码来识别指令。针对上文参考图 2A、2B 和 2C 描述的每种类型的指令，可以预先确定可用空间，该可用空间可通过比特数的形式表达。然后，在已经识别出指令之后，分析器 ANL 可以立即确定指令中的可用空间。

[0058] 例如，分析器 ANL 可以包括对各自指令的各自空闲范围进行指定的表。可以针对具体的指令集事先产生该表，例如，之前参考图 2A、2B 和 2C 描述的 MIPS 指令集。亦即，通过如前所述对该指令集的空闲范围分析而产生该表。从而，分析器 ANL 可以使用该表来确定空闲范围针对临时指令块 IB_{TMP} 中包括的每个指令提供的可用比特数。然后，分析器 ANL 可以计算已经确定的这些可用比特数的和。该和表示临时指令块 IB_{TMP} 中的可用空间。

[0059] 优选地，分析器 ANL 将临时指令块中 IB_{TMP} 中的一个或多个指令可能经历的任何重新格式化纳入考虑中。该重新格式化允许增加可用空间，随后将对其进行描述。亦即，通过对机器代码 MC 中的指令进行重新格式化所获得的已重新格式化的指令典型地包括与机器代码 MC 中的指令相比更多数目的可用比特。然后，分析器 ANL 优选地基于该更多数目的可用比特来计算临时指令块中 IB_{TMP} 中的可用空间。可以预先确定该更多数目的可用比特，因为重新格式化是确定性的操作。

[0060] 在步骤 S3 中，分析器 ANL 检验临时指令块中 IB_{TMP} 中的可用空间是否至少等于最小量 (SP ≥ MIN ？)。在可用空间小于最小量时，分析器 ANL 随后执行步骤 S4。在可用空间大于或等于最小量时，分析器 ANL 随后执行步骤 S5。该最小量与将要产生的数字签名 DS 中包括的数据量相对应。优选地，数字签名 DS 中包括的数据量是固定的，可以通过比特数的形式对其进行表达。可以确保这种固定量，因为散列码 HC 产生器 HCG 产生包括给定的固定数量的比特的散列码 HC。优选地，数字签名产生器 DSG 不修改该比特数 :数字签名 DS 与散列码 HC 具有相同的大小。

[0061] 在步骤 S4 中，分析器 ANL 将临时指令块 IB_{TMP} 中的指令替换为伪指令，例如“nop”指令（ΔIB_{TMP} :+NOP)。优选地，被替换的指令是非伪指令本身的指令，并且在序列的最后。实际上，从步骤 S1 中已经取出的机器代码 MC 的一部分删除该最后的非伪指令。这个被删除

的指令将形成机器代码 MC 的后续部分的一部分，其将会在重新执行该一系列的步骤 S1-S6 以提供后续指令块 IB 时被取出。亦即，将在后续的指令块 IB 中表示这个被删除的指令。

[0062] 相应地，步骤 S4 通过以伪指令来替换最后的非伪指令，提供了新的临时指令块 IB_{TMP}。然后，分析器 ANL 执行其后的步骤 S2 和步骤 S3，以检验新的临时指令块 IB_{TMP} 中的可用空间是否足以容纳数字签名 DS。如果不足以容纳，重复步骤 S4，这意味着以伪指令来替换其他的非伪指令。实际上，每个这种替换都是将伪指令插入到机器代码 MC 中。分析器 ANL 跟踪已被插入的伪指令的总数。

[0063] 在步骤 S5 中，分析器 ANL 对临时指令块 IB_{TMP} 中可能存在的任何跳转指令进行改动 ($MOD\_I_{JMP} \in IB_{TMP}$)。一个或多个伪指令的插入需要跳转地址的改动。该改动基于已经插入的伪指令的总数。例如，假设已经提供了连续的指令块 IB，并且总共已经插入 N 个伪指令，N 是整数。在这种情况下，存在着等于 N 的地址偏移，在临时指令块 IB_{TMP} 中存在的跳转指令中应该考虑到这点。这可以通过将跳转指令中的跳转地址增大 N 来进行。相应地，步骤 S5 从而提供了改动后的临时指令块 IB^*_{TMP}，其考虑到了伪指令的插入。

[0064] 在步骤 S6 中，指令块编辑器 IBC 使改动后的临时指令块 IB^*_{TMP} 经历重新格式化操作，以获得在其中将嵌入数字签名 DS 的指令块 IB（$FMT\_IB_{TMP}^* \Rightarrow IB$）。亦即，指令块编辑器 IBC 可以重新格式化一个或多个指令，以创建用于嵌入数字签名 DS 的附加空间。如之前提到的，重新格式化的指令典型地包括比其原始形式的对应指令更多数目的可用比特。

[0065] 例如，步骤 S6 中的重新格式化操作可以涉及操作码的熵编码。将相应的操作码映射到相应的可变长度代码。将机器代码 MC 中出现相对频繁的指令的操作码映射到相对短的可变长度代码。相反，将机器代码 MC 中很少出现的指令的操作码映射到相对长的可变长度代码。可以将可变长度代码放置在最初包括该操作码的指令字段中，例如，图 2A、2B 和 2C 中示出的 6 比特字段 OP。如果需要，可以将指令字段扩展一个比特或多个比特，以容纳可变长度代码。应该注意到，为了创建附加的空闲范围，除了操作码之外的字段也可以经历熵编码。

[0066] 典型地，由对操作码和可变长度代码之间的相关性进行定义的具体方案（熵编码方案）来表现熵编码的特征。典型地，不同的方案将在可以创建的附加空间（空闲范围）方面为给定的机器代码提供不同的改善。这是因为，在一个机器代码中相对频繁地出现的给定指令在另一机器指令中可能较不频繁地出现。因此，优选地确定为给定的机器代码提供最佳改善的熵编码方案，并在步骤 S6 中应用该最优的熵编码方案。在没有预先确定熵编码方案的情况下，需要使用某种方式向将要执行受保护的有关机器代码的处理器传送已经应用的熵编码方案。例如，可以在图 1 中示出的受保护的机器代码 MCP 中包括适当的指示。

[0067] 另一种重新格式化的形式如下：假定特定的指令在机器代码 MC 中相对频繁地出现。再假定每个指令包括以给定位置处未使用比特的形式出现的类似空闲范围，这是在该比特具有针对每一个指令的相同值的意义上来说的。未使用的比特是无意义的。可以使用这种未使用的比特来标识频繁出现的指令，而不是为此使用整个操作码。在这种情况下，频繁出现的指令的操作码变得可用作空闲范围。可以说，替换前述的操作码，放弃未使用比特而作为空闲范围，这构建了更大的空闲范围。可以将这种替换认为是熵编码。

[0068] 作为另一个示例，假定机器代码 MC 包括至少一种类型的、包括具体比特集（该比特集构建了所谓的字段）的指令，以指定值范围中的具体值。有关字段可以包括例如 16 个

比特。假定在有关字段中，8 比特值频繁出现。在该字段指定 8 比特值的情况下，实际上，存在着未使用的 8 个比特。典型地，这 8 个未使用的比特将是其值为零的字段的 8 个最高有效位。假定该有关指令或者与其相关联的指令包括以一个或多个未使用比特的形式表现的空闲范围。可以使用空闲范围中的单个比特来指示该有关字段包括 8 比特值。相应地，该字段中实际上未使用的 8 个比特变得可用作空闲范围。

[0069] 从而，对机器代码指令进行重新格式化可以创建用于数字签名 DS 嵌入的附加空间。这是有利的，因为这种重新格式化可以去除对包括伪指令（如前所述，这是有缺陷的）的需要。然而，典型地，对机器代码 MC 进行重新格式化将要求在需要执行机器代码 MC 的处理器中进行逆重新格式化。可能需要将已重新格式化的指令转变为其原始格式。

[0070] 将首先针对构建机器代码 MC 的开始的机器代码部分 MC-P 执行图 3 中示出的一系列步骤 S1-S6。然后，在分析器 ANL 的控制之下，将针对机器代码 MC 的后续部分 MC-P 重复执行该一系列的步骤 S1-S6，使得指令块编辑器 IBC 提供连续的指令块 IB。如参考图 1 所描述的，针对每个指令块 IB 产生数字签名 DS，并基于空闲范围指示 FI 将数字签名 DS 嵌入指令块 IB 中，空闲范围指示 FI 由分析器 ANL 提供。相应地，获得连续的受保护的指令块 IBP，其共同构建了受保护的机器代码 MC。

[0071] 图 4 示出了被布置来执行受保护的机器代码 MCP 的数据处理系统 DPS，该机器代码 MCP 已经如前参考图 1 和 3 所述地产生。数据处理系统 DPS 包括非易失性存储器 ROM、处理器 PRC 和数据处理路径 DHP。非易失性存储器 ROM 已经加载有受保护的机器代码 MCP。可以通过例如集成电路的形式实现数据处理系统 DPS。作为另一示例，可以通过包括集成电路的设备的形式实现数据处理系统 DPS，在该集成电路上已经实现了非易失性存储器 ROM 和处理器 PRC。

[0072] 处理器 PRC 包括两个接口 IF1、IF2，一个接口 IF1 用于非易失性存储器 ROM，另一接口 IF2 用于数据处理路径 DHP。处理器 PRC 还包括安全模块 SEM、高速缓存存储器 CHM 和指令执行电路 CPU。数据处理路径 DHP 可以包括用于临时存储数据的易失性存储器，以及可选地包括一个或多个专用数据处理电路。

[0073] 基本上，数据处理系统 DPS 如下操作。数据处理系统 DPS 向输入数据 DI 应用至少一个数据处理操作，以获得输出数据 DO。输入数据 DI 和输出数据 DO 可以具有例如数据流的形式。输入数据 DI 可以是例如输入信号的数字表示。

[0074] 处理器 PRC 可以直接执行数据处理操作，该数据处理操作由受保护的机器代码 MCP 所定义。备选地，处理器 PRC 可以控制数据处理路径 DHP 中的专用数据处理电路所执行的数据处理操作。处理器 PRC 还可以通过时间复用的方式控制若干数据处理操作。在任何一种情况下，处理器 PRC 执行受保护的机器代码 MCP，这使得处理器 PRC 执行数据处理操作或者控制操作，或者这些操作的组合。

[0075] 更详细地，接口 IF1 从非易失性存储器 ROM 取回受保护的指令块 IBP，该受保护的指令块 IBP 形成了受保护的机器代码 MCP 的一部分。接口 IF1 响应于存储器读取请求 MQ，取回受保护的指令块 IBP，存储器读取请求 MQ 由高速缓存存储器 CHM 所发出。接口 IF1 将受保护的指令块 IBP 传递到安全模块 SEM。如上文所解释，受保护的指令块 IBP 包括嵌入其中的数字签名。

[0076] 安全模块 SEM 基于受保护的指令块 IBP 中嵌入的数据签名，检查受保护的指令块

IBP 的真实性和完整性。在该检查具有负面结果的情况下，安全模块 SEM 提供告警指示 AL。在该情况下，防止处理器 PRC 对其他指令进行任何执行，亦即，对处理器 PRC 进行阻断或者重置，或者两者都进行。例如，可以响应于告警指示 AL，阻断指令执行电路 CPU。作为另一示例，安全模块 SEM 可以产生中断，该中断应用到能够阻断或者停止处理器 PRC 的一个或多个实体。

[0077] 在已经如上文参考图 1 和 3 所述应用重新格式化操作以产生受保护的指令块 IBP 的情况下，安全模块 SEM 还可以执行逆重新格式化操作。在任何情况下，安全模块 SEM 提供了重新获得的机器代码部分 MC-P*，该重新获得的机器代码部分 MC-P* 包括可能已经向其增加了一个或多个伪指令的原始机器代码部分 MC-P。在没有增加伪指令的情况下，图 4 中示出的重新获得的机器代码部分 MC-P* 与图 1 中示出的向代码保护器 CPR 输入的机器代码部分 MC-P 相对应。

[0078] 高速缓存存储器 CHM 临时存储重新获得的机器代码部分 MC-P*。指令执行电路 CPU 发出高速缓存读取请求 RQ，以从高速缓存存储器 CHM 获取要执行的机器代码指令 MC-I。高速缓存读取请求 RQ 可以包括指定具体的机器代码指令 MC-I 的地址。高速缓存存储器 CHM 可以包括控制器，该控制器将该地址映射到高速缓存存储器 CHM 中包括有关机器代码指令 MC-I 的具体存储器单元的地址，或者在高速缓存存储器 CHM 中不存在所请求的机器代码指令 MC-I 的情况下发出存储器读取请求 MQ。

[0079] 指令执行电路 CPU 连续执行从高速缓存存储器 CHM 中读取的机器代码指令 MC-I。机器代码指令 MC-I 的执行可以涉及经由接口 IF2 向数据处理路径 DHP 发出数据读取地址 RA 或者数据写入地址 WA。在数据读取地址 RA 的情况下，数据处理路径 DHP 可以提供要处理的输入数据单元 IE，其经由接口 IF2 到达指令执行电路 CPU。在数据写入地址 WA 的情况下，指令执行电路 CPU 可以提供输出数据单元 OE，其经由 IF2 传递至数据处理路径 DHP。

[0080] 总而言之，高速缓存存储器 CHM 和指令执行电路 CPU 以与这些实体在常规数据处理系统中的操作方式实质上相类似的方式而操作。在常规数据处理系统中，可以没有安全模块，而且高速缓存存储器可以直接从非易失性存储器获取普通的、未受保护的机器代码部分，在该非易失性存储器中可以储存图 1 中示出的机器代码 MC。

[0081] 图 5 示出来安全模块 SEM 的细节。安全模块 SEM 包括以下功能实体：数据分离器 SEP、逆重新格式化器 IFMT、签名解码器 DEC、散列码产生器 HCG 和比较器 CMP。可以通过例如已经加载到可编程处理器中的指令集的方式来实现这些功能实体中的每一个。在这种基于软件的实现中，指令集定义了相关功能实体执行的操作，稍后将对其进行描述。在这方面，可以将图 5 视为表示一种方法，由此数据分离器 SEP 表示数据分离步骤，逆重新格式化器 IFMT 表示逆重新格式化步骤，签名解码器 DEC 表示签名解码步骤，散列码产生器 HCG 表示散列码产生步骤，以及比较器 CMP 表示比较步骤。

[0082] 图 5 中示出的安全模块 SEM 如下操作。假定安全模块 SEM 接收到已经如上文参考图 1 和图 3 所述而产生的受保护的指令块 IBP。还假定受保护的指令块 IBP 没有经过任何修改。数据分离器 SEP 从应用到安全模块 SEM 的受保护的指令块 IBP 中提取并移除数字签名 DS。事实上，数据分离器 SEP 将受保护的指令块 IBP 分割为两部分：第一部分包括数字签名 DS，第二部分包括已经从其剥离了数字签名 DS 的指令块 IB。

[0083] 逆重新格式化器 IFMT 向指令块 IB 应用逆重新格式化操作。逆重新格式化操作把

由图 3 中示出的步骤 S6 中的格式化操作（该操作由图 1 中示出的指令块编辑器 IBC 执行）所引入的修改进行还原。重新格式化的指令返回其原始格式，即，出现在图 1 中示出的机器代码 MC 中的有关指令的格式。相应地，逆重新格式化器 IFMT 提供了获取的机器代码部分 MC-P^*，该获取的机器代码部分 MC-P^* 包括原始机器代码部分 MC-P 和可能已经插入的一个或多个伪指令。原则上，不需要移除这些伪指令，因为其不影响处理器 PRC 所执行的操作。

[0084] 签名解码器 DEC 基于从受保护的指令块 IBP 提取出的数字签名 DS 和密钥 KY 产生目标散列码 HC_T。该密钥 KY 与图 1 中示出的已被用于产生数字签名 DS 的密钥 SK 具有给定的预定关系。在如上文所述已应用非对称加密方案的情况下，密钥 KY 可以是公钥。所产生的目标散列码 HC_T 与图 1 中示出的散列码产生器 HCG 已产生的散列码 HC 相对应。这是因为假定数字签名 DS 没有经过任何修改。

[0085] 图 5 中示出的散列码产生器 HCG 应用与图 1 中示出的散列码产生器 HCG 相同的散列函数。散列码产生器 HCG 向互补指令块 IB+ 应用该散列函数。通过向指令块 IB 增加表示地址 AD 来获得互补指令块 IB+。表示地址 AD 是将从其执行有关指令的指令块 IB 内的指令地址。例如，处理器 PRC 内的地址计数器可以提供表示地址 AD。

[0086] 散列码产生器 HCG 通过向互补指令块 IB+ 应用散列函数而提供获取的散列码 HC_R。如果图 5 中示出的数据处理系统 DPS 内的表示地址 AD 与图 1 中示出的保护代码产生器 PCG 内的表示地址 AD 相同，获取的散列码 HC_R 与图 1 中示出的散列码产生器 HCG 已提供的散列码 HC 相对应。此外，如上文所述，已假定指令块 IB 未经过任何修改。

[0087] 比较器 CMP 将获取的散列码 HC_R（基于指令块 IB 和表示地址 AD）与目标散列码 HC_T（基于数字签名 DS）相比较。已假定受保护的指令块 IBP 没有经过任何修改，这意味着数字签名 DS、指令块 IB 或者表示地址 AD 都没有经过任何修改。结果，获取的散列码 HC_R 将与目标散列码 HC_T 相同。上述散列码 HC 的这种一致性提供了对指令块 IB 的真实性和完整性的保证。

[0088] 相反，如果受保护的指令块 IBP 已经被修改或者如果已经引起地址错误，则获取的散列码 HC_R 和目标散列码将不会彼此对应。亦即，如果以下单元中的任意一个经过了修改，将存在着不一致性：数字签名 DS、指令块 IB 和表示地址 AD。在该情况下，比较器 CMP 提供上文所述的告警指示 AL，这将阻断处理器 PRC 或者将防止执行其他指令。因为真实性或完整性或者两者均未得以保证，因此暂停处理。

[0089] 图 6 示出了可替换图 5 中示出的数据处理系统 DPS 中的处理器 PRC 的备选处理器 PRC_A。备选处理器 PRC_A 包括类似的实体：两个接口 IF1、IF2，一个接口 IF1 用于非易失性存储器 ROM，另一接口 IF2 用于数据处理路径 DHP。备选处理器 PRC_A 还包括安全模块 SEM、高速缓存存储器 CHM 和指令执行电路 CPU。在备选处理器 PRC_A 中，将安全模块 SEM 布置在高速缓存存储器 CHM 和指令执行电路 CPU 之间。这是与图 5 中示出的处理器 PRC 的主要区别，在图 5 中示出的处理器 PRC 中，将安全 SEM 布置在接口 IF1 和高速缓存存储器 CHM 之间。

[0090] 图 6 中示出的备选处理器 PRC_A 中的实体以与图 5 中示出的处理器 PRC 中的实体实质上相类似的方式操作。高速缓存存储器 CHM 临时存储从非易失性存储器 ROM 获取的受保护的指令块 IBP。安全模块 SEM 对受保护的指令块 IBP 执行真实性检查和完整性检查。在该检查提供了负面结果的情况下，阻断备选处理器 PRC_A。可以说，安全模块 SEM 还将受保护的指令块 IBP 转换为机器代码指令 MC-I 的序列。在执行机器代码指令 MC-I 中，为了分别

传递输入数据单元 IE 和输出数据单元 OE，指令执行电路 CPU 可以经由 IF2 将数据读取和写入地址 RA、WA 应用到数据处理路径 DHP。

[0091] 总而言之，在图 6 中示出的备选处理器 PRC_A 中，在高速缓存存储器 CHM 和指令执行电路 CPU 之间执行真实性和完整性检查，而在图 5 中示出的处理器 PRC 中，在非易失性存储器 ROM 和高速缓存存储器 CHM 之间的较早级中执行该检查。两种方案都有优势和缺陷。

[0092] 图 5 中示出的方案的优势是方案相对易于实现，可以将其指定为“高速缓存再填充时的检查”。在受保护的指令块 IBP 具有与高速缓存存储器 CHM 中的行大小相对应的给定的固定大小的情况下，这是非常符合的。例如，在行大小是 8 个指令的情况下，受保护的指令块 IBP 优选地包括 8 个指令。可以基于常规处理器，通过相对有成本效率的方式来实现图 5 中示出的处理器 PRC，该常规处理器典型地包括高速缓存存储器和指令执行电路。这种实现主要涉及在高速缓存存储器和指令执行电路之间增加安全模块。

[0093] 图 6 中示出的方案的优势是该方案潜在地提供更高的保护度，可以将其指定为“刚好在执行之前检查”。入侵者可以成功修改高速缓存存储器 CHM 中临时存储的一个或多个指令。例如，入侵者可以将处理器 PRC 暴露在相对强的光或者另一种类型的辐射下以修改程序计数器的计数器值。这将引起不想要的跳转。图 6 中示出的备选处理器 PRC_A 中的安全模块 SEM 将检测到这种不想要的跳转。

[0094] 相反，在图 5 中示出的“高速缓冲再填充时的检查”方案中，对临时存储在高速缓存存储器 CHM 中的指令几乎没有提供对抗修改的保护。这是因为，在将指令写入到高速缓存存储器 CHM 时、而不是在从高速缓存存储器 CHM 读取指令时执行真实性和完整性检查，以便仅在此之后立即执行。这是图 5 中示出的“高速缓冲再填充时的检查”的固有缺陷。

[0095] 图 6 中示出的“刚好在执行之前检查”方案的缺陷是，该方案的实现可能相对复杂。一般地，将不得不对指令执行电路 CPU 进行特别改动，以在该方案下操作。亦即，一般地，不可能使用从常规处理器精确拷贝的指令执行电路 CPU 来实现“刚好在执行之前检查”方案。可能需要改动。

[0096] 此外，典型地，“刚好在执行之前检查”方案可能要求相对复杂地形成的受保护的指令块 IBP。典型地，针对该方案的受保护的指令块 IBP 应该包括一个接一个连续执行的指令。除了在序列中最后出现的分支指令外，在受保护的指令块 IBP 中不应该出现分支指令。这种限制可能使受保护代码的产生和受保护代码的执行的实现变得复杂。此外，典型地，受保护的指令块 IBP 的大小将根据机器代码 MC 中在何处出现分支指令而发生变化。总而言之，一般要在复杂度和保护度之间进行折衷。

[0097] 结论性的评述

[0098] 上文参考附图的详细描述仅是对本发明以及在权利要求中定义的附加特征的示出。可以通过众多不同方式来实现本发明。为了对此进行示出，对一些备选进行简要指出。

[0099] 可以应用本发明以在涉及软件保护的众多类型的产品或方法中获得优势。例如，可以将本发明应用到通信设备（例如蜂窝电话）中，以保护对属于支付的特征的访问进行管理的软件。作为另一示例，可以将本发明应用到内容呈现设备（例如，所谓的蓝光设备）中，以保护实现数字版权管理的软件。可以将已经根据本发明进行保护的软件存储在任何类型的介质中。图 4 中示出的非易失性存储器 ROM 仅是示例。例如，可以将受保护的软件存储在易失性存储器或者可以光记录或可以磁记录的介质中，等等。

[0100] 存在着根据本发明的众多保护软件的方式。可以仅针对程序的具体部分产生保护性代码，由此仅将该保护性代码嵌入到该部分中。例如，可以仅将图 3 中示出的一系列步骤 S1-S6 应用到机器代码 MC 的具体部分，其被存储到图 1 中示出的数据存储空间 DST2 中。

[0101] 在不修改机器代码指令并且不插入任何伪指令的情况下，将保护性代码直接嵌入到机器代码指令组中是可能的。例如，参考图 1，可以将指令块 IBC 布置为提供作为相应的机器代码部分 MC-P 的精确拷贝的相应指令块 IB。亦即，可以联合地或者单独地去除图 3 中示出的步骤 S4 和步骤 S5，其中在步骤 S4 中插入了伪指令，而在步骤 S5 中对机器代码指令进行了重新格式化。可以在编译期间增加伪指令以确保指令块中具有充足的空闲空间。参考图 1，编译器 CMP 可以增加伪指令，使得指令块编辑器 IBC 不需要增加任何的伪指令。这种方法的优势是可以避免重新计算地址，这降低了复杂性和出错的风险。

[0102] 可以采用其他各种措施来确保指令块中有充足的空闲空间用于嵌入保护性代码。例如，在指令的数目方面，指令块可以具有动态定义的大小。在初始指令块中不具有充足的空闲空间的情况下，可以增加指令直到具有充足的空闲空间为止。作为另一示例，可以将保护性代码在大小方面进行改动，以使得该保护性代码可以说适于给定的指令块。例如，在给定的指令块可以容纳 32 个附加比特的情况下，针对该给定的指令块产生 32 比特的保护性代码。为此，例如，可以对图 1 中示出的代码保护器 CPR 进行修改，使得分析器 ANL 通过基于可用空闲空间选择适当的散列码函数来控制散列码产生器 HCG。

[0103] 存在着众多产生保护性代码的方式。保护性代码不需要一定具有数字签名的形式，数字签名意味着使用具有公 - 私密钥对的非对称加密方案。例如，可以通过其他加密技术产生保护性代码，例如，使用对称密钥对而不是私 - 公密钥对的加密技术。在仅想要对抗介质错误的情况下，保护性代码不需要一定涉及加密。例如，保护性代码可以是散列码。此外，保护性代码不需要一定从包括地址信息的数据集合产生。例如，可以对图 1 中示出的代码保护器 CPR 进行修改，使得可以仅从指令块 IB 产生散列码 HC，并因此产生数字签名 DS。亦即，不需要考虑表示地址 AD。

[0104] 应该广义地理解术语“高速缓存存储器”。该术语包含了用于存储已经依照本发明进行保护的指令块的任何类型的存储器。应该广义地理解术语“数字签名”。该术语包含了允许检验完整性和真实性的任何类型的代码。

[0105] 虽然附图中将不同的功能实体示出为不同的块，然而绝不应该排除单一实体执行若干功能或者若干实体执行单一功能的实现。在这方面，附图仅仅是图示性的。例如，参考图 1，可以通过单一处理器 PRC 的形式实现编译器 CPL 和代码保护器 CPR。

[0106] 存在着通过硬件或软件或者硬件和软件的组合的方式来实现功能实体的众多方式。虽然已经提到了这些功能实体的基于软件的实现，然而绝不应该排除基于硬件的实现。典型地，基于硬件的实现涉及专用电路，每个专用电路具有对该有关专用电路执行的操作进行定义的具体拓扑。在系统（或者系统中包括的功能实体）包括一个或多个专用电路以及一个或多个适当编程的处理器 PRC 方面，混合实现也是可能的。

[0107] 存在着众多存储和分发指令块（亦即，软件）的方式，其允许根据本发明来保护软件。例如，可以将软件存储在适当的介质中，例如，光盘或者存储器电路。可以将存储有软件的介质作为单独的产品提供，或者与可以执行软件的另一产品一起提供。这种介质还可以是能够执行软件的产品的一部分。还可以经由通信网络分发软件，通信网络可以是有线

的、无线的或者混合型的。例如，可以经由互联网分发软件。通过服务器的方式，可以使软件可被下载。可以经过支付来进行下载。

[0108] 在参考附图论证该详细描述之前在此进行的评述对本发明进行了示出而不是限制。存在落入所附权利要求范围内的众多备选。权利要求中的任何附图标记不应被解释为对该权利要求的限制。用语“包括”并不排除除了在权利要求中列出的元件或者步骤之外，存在着其他的元件或步骤。在元件和步骤之前的用词“一”或“一个”并不排除存在着多个这样的元件或步骤。相应的从属权利要求定义了相应的附加特征这一起码的事实不排除附加特征的组合，这对应于从属权利要求的组合。

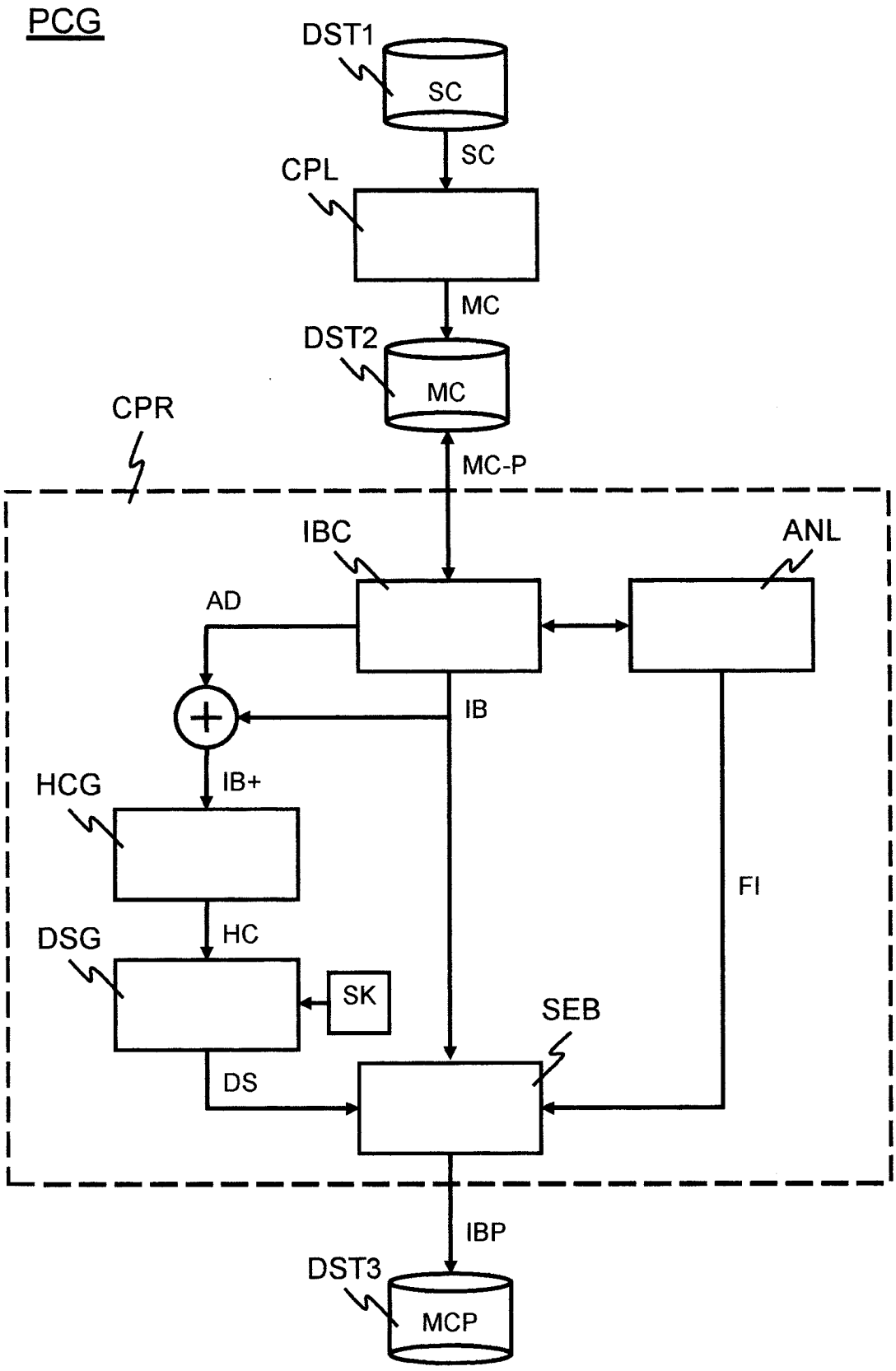
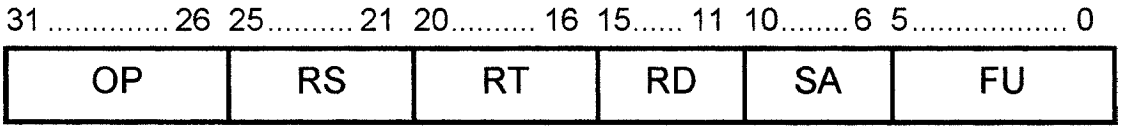
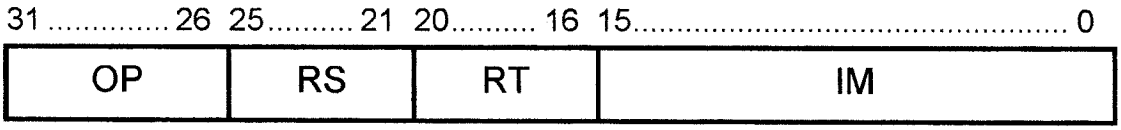

PCG
DST1
SC
SC
CPL
MC
DST2
MC
CPR
MC-P
IBC
ANL
AD
IB
IB+
HCG
HC
DSG
SK
SEB
DS
FI
IBP
DST3
MCP

图 1



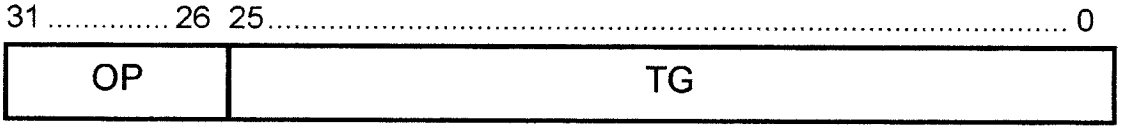
R

图 2A



I

图 2B



J

图 2C

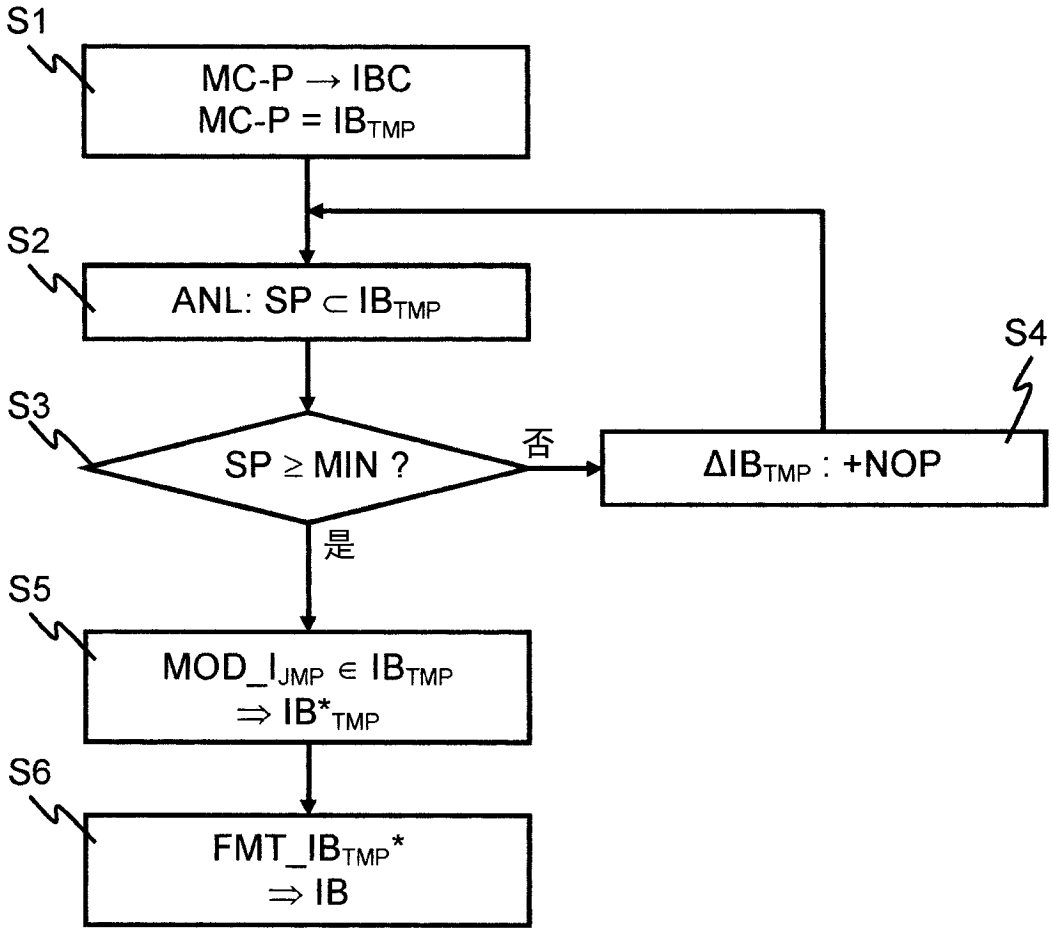

S1
MC-P → IBC
MC-P = IB_TMP
S2
ANL: SP ⊂ IB_TMP
S3
SP ≥ MIN ?
否
S4
ΔIB_TMP : +NOP
是
S5
MOD_I_JMP ∈ IB_TMP
⇒ IB*_TMP
S6
FMT_IB_TMP*
⇒ IB

图 3

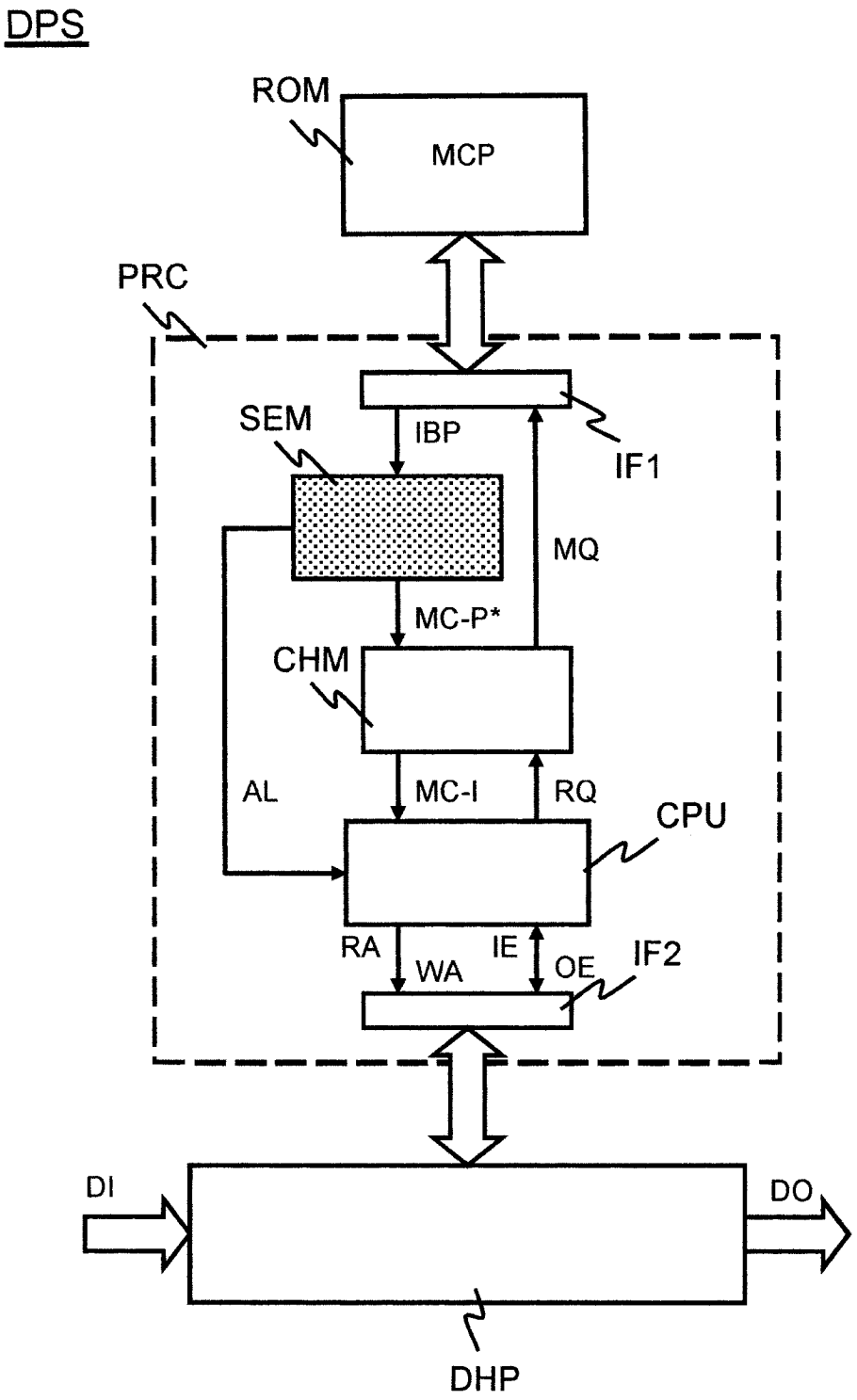

DPS
ROM
MCP
PRC
SEM
IBP
IF1
MQ
MC-P*
CHM
AL
MC-I
RQ
CPU
RA
WA
IE
OE
IF2
DI
DO
DHP

图 4

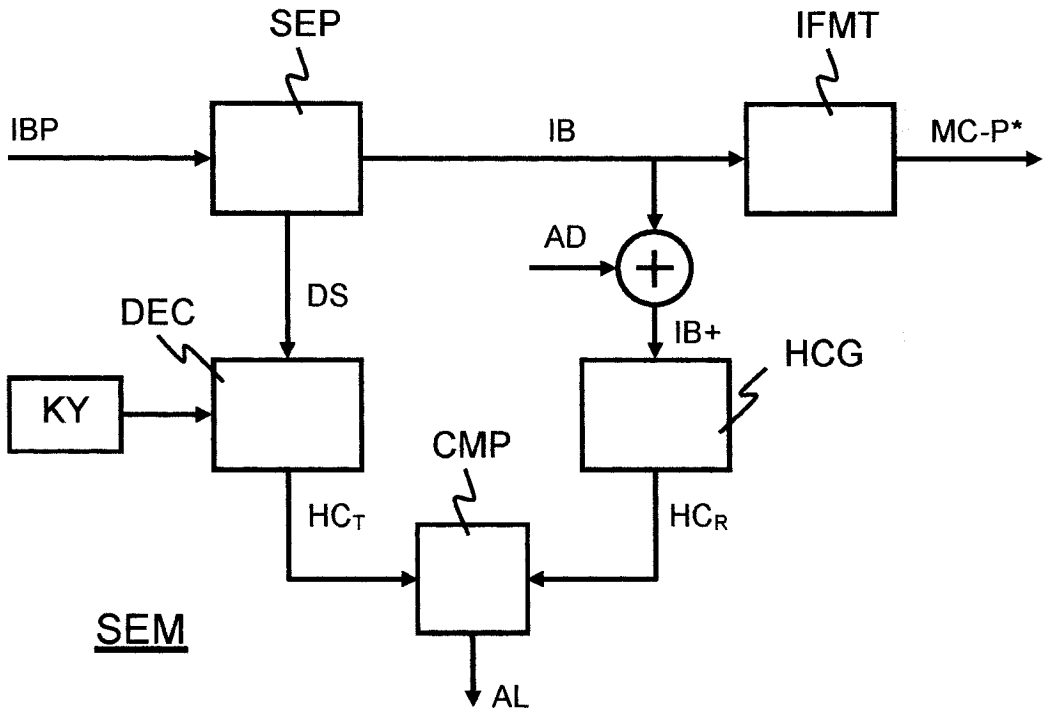

SEP
IFMT
IBP
IB
MC-P*
AD
DS
DEC
IB+
HCG
KY
CMP
HC_T
HC_R
SEM
AL

图 5

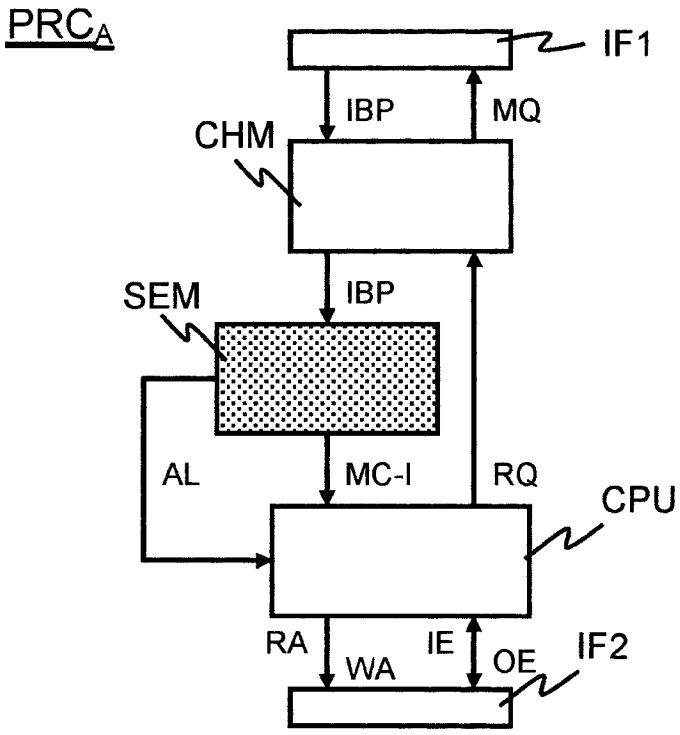

PRC_A
IF1
IBP
MQ
CHM
SEM
IBP
AL
MC-I
RQ
CPU
RA
WA
IE
OE
IF2

图 6