(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 112181491 A
(43)申请公布日 2021.01.05

(21)申请号 201910586325.5

(22)申请日 2019.07.01

(71)申请人 华为技术有限公司
地址 518129 广东省深圳市龙岗区坂田华为总部办公楼

(72)发明人 钱雅超 章庆隆 汤倩莹

(74)专利代理机构 北京同立钧成知识产权代理有限公司 11205
代理人 祝乐芳 刘芳

(51)Int.Cl.
*G06F 9/30*(2006.01)
*G06F 9/46*(2006.01)

权利要求书3页 说明书18页 附图8页

(54)发明名称

处理器及返回地址的处理方法

(57)摘要

本申请实施例提供一种处理器及返回地址的处理方法，在处理器中设置硬件的转换电路，当需要保存返回地址时，利用转换电路对返回地址进行转换，将得到的转换返回地址输出至内存中；当需要使用返回地址时，利用转换电路对内存中的转换返回地址进行转换，得到该返回地址。由于攻击者无法知道转换电路中所作的转换操作，使得攻击者无法将内存中的转换返回地址修改为恶意指令对应的转换返回地址，从而，能够防止攻击者对程序控制流的恶意更改。并且，由于上述转换过程是在程序运行过程中通过硬件的转换电路实现，无需在编译阶段对调用指令和返回指令进行识别，也无需插入额外的加密指令和解密指令，避免了对处理器的运行性能造成影响。


CN 112181491 A

1.一种处理器,其特征在于,包括:处理核和转换电路;

所述处理核用于输出返回地址;

所述转换电路用于对所述处理核输出的返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至内存中的栈;

所述转换电路还用于在所述处理核需要使用所述返回地址时,对所述栈中的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述处理核。

2.根据权利要求1所述的处理器,其特征在于,还包括寄存器;

所述转换电路具体用于对所述处理核输出的返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至所述寄存器,以使所述转换返回地址经由所述寄存器输出至内存中的栈;

所述转换电路还具体用于在所述处理核需要使用所述返回地址时,对所述栈中的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述寄存器,以使所述返回地址经由所述寄存器输出至所述处理核。

3.根据权利要求1所述的处理器,其特征在于,还包括寄存器;

在所述处理核输出所述返回地址时,所述寄存器用于寄存所述处理核输出的返回地址;

所述转换电路具体用于对所述寄存器输出的所述返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至内存中的栈;

在所述处理核需要使用所述返回地址时,所述寄存器还用于寄存所述栈输出的所述转换返回地址;

所述转换电路还具体用于对所述寄存器输出的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述处理核。

4.根据权利要求2或3所述的处理器,其特征在于,所述转换满足如下条件:

B＝IP(A),A＝IP(B)

其中,A为所述返回地址,B为所述转换返回地址,IP()为所述转换所采用的转换模型。

5.根据权利要求4所述的处理器,其特征在于,所述转换电路具体用于:

采用至少一种转换模型对所述返回地址的至少一个比特进行转换,得到所述转换返回地址。

6.根据权利要求5所述的处理器,其特征在于,所述转换电路具体用于:

对所述返回地址的至少一个比特进行分组,得到多个比特组;

采用所述转换模型对每个所述比特组中的比特进行转换,得到每个所述比特组对应的转换结果,其中,所述多个比特组中至少存在两个比特组采用的转换模型不同,或者,所述多个比特组采用的转换模型相同;

根据各所述比特组对应的转换结果,得到所述转换返回地址。

7.根据权利要求6所述的处理器,其特征在于,所述比特组的数量为两个,其中一个比特组中包括所述返回地址的奇数比特位对应的比特,另一个比特组中包括所述返回地址的偶数比特位对应的比特。

8.根据权利要求4至7任一项所述的处理器,其特征在于,所述转换模型的种类包括:模乘法转换模型、模加法转换模型。

9.根据权利要求2至8任一项所述的处理器,其特征在于,所述寄存器为用于存储返回地址的寄存器。

10.根据权利要求9所述的处理器,其特征在于,所述处理器为基于ARM指令集的处理器,所述寄存器为LR寄存器。

11.根据权利要求9所述的处理器,其特征在于,所述处理器为基于RISC V指令集的处理器,所述寄存器为RA寄存器。

12.一种返回地址的处理方法,其特征在于,应用于处理器,所述处理器包括:处理核和转换电路,所述方法包括:

在所述处理核输出所述返回地址时,通过所述转换电路对所述返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至内存中的栈;

在需要使用所述返回地址时,通过所述转换电路对所述栈中的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述处理核。

13.根据权利要求12所述的方法,其特征在于,所述处理器还包括寄存器,所述通过所述转换电路对所述返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至内存中的栈,包括:

通过所述转换电路对所述返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至所述寄存器,以使所述转换返回地址经由所述寄存器输出至内存中的栈;

所述通过所述转换电路对所述栈中的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述处理核,包括:

通过所述转换电路对所述栈中的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述寄存器,以使所述返回地址经由所述寄存器输出至所述处理核。

14.根据权利要求12所述的方法,其特征在于,所述处理器还包括寄存器,所述返回地址被所述处理核输出至所述寄存器,所述通过所述转换电路对所述返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至内存中的栈,包括:

通过所述转换电路对所述寄存器输出的所述返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至内存中的栈;

所述通过所述转换电路对所述栈中的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述处理核,包括:

通过所述转换电路对所述寄存器输出的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述处理核。

15.根据权利要求13或14所述的方法,其特征在于,所述转换满足如下条件:

$B=IP(A)$,$A=IP(B)$

其中,A为所述返回地址,B为所述转换返回地址,IP()为所述转换所采用的转换模型。

16.根据权利要求15所述的方法,其特征在于,所述对所述返回地址进行转换以得到转换返回地址,包括:

采用至少一种转换模型对所述返回地址的至少一个比特进行转换,得到所述转换返回地址。

17.根据权利要求16所述的方法,其特征在于,所述采用至少一种转换模型对所述返回

地址的至少一个比特进行转换，得到所述转换返回地址，包括：

对所述返回地址的至少一个比特进行分组，得到多个比特组；

采用所述转换模型对每个所述比特组中的比特进行转换，得到每个所述比特组对应的转换结果，其中，所述多个比特组中至少存在两个比特组采用的转换模型不同，或者，所述多个比特组采用的转换模型相同；

根据各所述比特组对应的转换结果，得到所述转换返回地址。

18.根据权利要求17所述的方法，其特征在于，所述比特组的数量为两个，其中一个比特组中包括所述返回地址的奇数比特位对应的比特，另一个比特组中包括所述返回地址的偶数比特位对应的比特。

19.根据权利要求15至18任一项所述的方法，其特征在于，所述转换模型的种类包括：模乘法转换模型、模加法转换模型。

20.根据权利要求13至19任一项所述的方法，其特征在于，所述寄存器为用于存储返回地址的寄存器。

21.根据权利要求20所述的方法，其特征在于，所述处理器为基于ARM指令集的处理器，所述寄存器为LR寄存器。

22.根据权利要求20所述的方法，其特征在于，所述处理器为基于RISC V指令集的处理器，所述寄存器为RA寄存器。

23.一种电子设备，其特征在于，包括如权利要求1至11任一项所述的处理器。

# 处理器及返回地址的处理方法

## 技术领域

[0001] 本申请实施例涉及计算机技术领域，尤其涉及一种处理器及返回地址的处理方法。

## 背景技术

[0002] 处理器运行程序的过程中，可能出现攻击者恶意劫持程序控制流的事件。具体的，攻击者通过修改子程序的返回地址，将正常的返回地址修改为恶意返回地址，使得处理器执行完子程序代码后跳转到恶意返回地址所指向的代码段，达到改变程序控制流的目的，从而破坏程序的控制流完整性(Control Flow Integrity，CFI)。

[0003] 目前，为了防御程序控制流被恶意改变，通常需要对程序运行时的控制流进行监控，若程序控制流被改变，则发出警报。一种相关技术中，在程序编译阶段，识别出子程序对应的调用指令和返回指令，在调用指令之前插入一个加密指令，并在返回指令之前插入一个解密指令。进而，在程序运行阶段，在处理器调用子程序之前，先利用加密指令对子程序的返回地址进行加密，将得到的加密地址入栈。在子程序执行完毕后，利用解密指令对出栈的加密地址进行解密，得到原始的返回地址，从而处理器能够从返回地址处继续执行。

[0004] 采用上述防御技术后，即使攻击者从栈中劫持到加密地址，由于攻击者不知道加密指令采用的密钥，无法将加密地址篡改为加密的恶意地址。也就是说，攻击者劫持程序控制流后仍无法控制程序的跳转位置，因此，能够阻止攻击者对程序控制流进行恶意改变，实现对程序的控制流完整性的保护。

[0005] 然而，上述技术中，需要在程序中插入多个额外指令，使得程序的运行性能降低。

## 发明内容

[0006] 本申请实施例提供一种处理器及返回地址的处理方法，在不降低程序运行性能的基础上，对程序的控制流完整性进行保护。

[0007] 第一方面，本申请实施例提供一种处理器，包括：处理核和转换电路；

[0008] 所述处理核用于输出返回地址；

[0009] 所述转换电路用于对所述处理核输出的返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至内存中的栈；

[0010] 所述转换电路还用于在所述处理核需要使用所述返回地址时，对所述栈中的所述转换返回地址进行所述转换，得到所述返回地址，并将所述返回地址输出至所述处理核。

[0011] 本实施例中，在返回地址存入内存之前，转换电路对返回地址进行一次转换，因此，内存中存储的是转换后的返回地址。由于攻击者无法知道转换电路中所作的转换操作，使得攻击者无法将内存中的转换返回地址修改为恶意指令对应的转换返回地址，从而，能够防止攻击者对程序控制流的恶意更改。在转换后的返回地址从内存出栈之后，转换电路对转换后返回地址再进行一次转换，得到原始的返回地址，使得处理核可以根据原始的返回地址进行后续指令的执行，保证了程序控制流的完整性。由于上述转换过程是在程序运

行过程中通过硬件的转换电路实现，无需在编译阶段对调用指令和返回指令进行识别，也无需插入额外的加密指令和解密指令，避免了对处理器的运行性能造成影响。同时，还规避了软件窃取风险。

[0012] 可选的，所述处理器还包括寄存器；

[0013] 所述转换电路具体用于对所述处理核输出的返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至所述寄存器，以使所述转换返回地址经由所述寄存器输出至内存中的栈；

[0014] 所述转换电路还具体用于在所述处理核需要使用所述返回地址时，对所述栈中的所述转换返回地址进行所述转换，得到所述返回地址，并将所述返回地址输出至所述寄存器，以使所述返回地址经由所述寄存器输出至所述处理核。

[0015] 本实施例中，在寄存器的写入通路上设置硬件的转换电路，该转换电路用于对进入寄存器的返回地址进行转换，即，所有需要进入寄存器的输入都会先经过该转换电路的转换，然后将转换结果输入至寄存器中。通过该方式，能够自动识别返回地址，无需改变处理器的控制流程，易于实施。

[0016] 可选的，所述处理器还包括寄存器；

[0017] 在所述处理核输出所述返回地址时，所述寄存器用于寄存所述处理核输出的返回地址；

[0018] 所述转换电路具体用于对所述寄存器输出的所述返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至内存中的栈；

[0019] 在所述处理核需要使用所述返回地址时，所述寄存器还用于寄存所述栈输出的所述转换返回地址；

[0020] 所述转换电路还具体用于对所述寄存器输出的所述转换返回地址进行所述转换，得到所述返回地址，并将所述返回地址输出至所述处理核。

[0021] 本实施例中，在寄存器的读出通路上设置硬件的转换电路，该转换电路用于对寄存器输出的返回地址进行转换，即，所有从寄存器输出的值都会先经过该转换电路的转换。通过该方式，能够自动识别返回地址，无需改变处理器的控制流程，易于实施。

[0022] 可选的，所述转换满足如下条件：

[0023] B＝IP(A)，A＝IP(B)

[0024] 其中，A为所述返回地址，B为所述转换返回地址，IP()为所述转换所采用的转换模型。

[0025] 可选的，所述转换电路具体用于：

[0026] 采用至少一种转换模型对所述返回地址的至少一个比特进行转换，得到所述转换返回地址。

[0027] 其中，所述至少一个比特包括程序的代码地址的变化区段对应的比特和不变区段对应的比特；所述程序的代码地址包括多个指令的指令地址，所述不变区段为多个指令地址中比特相同的比特位，所述变化区段为多个指令地址中比特不同的比特位。

[0028] 通过将转换模型及转换参数存在硬件电路中，使得攻击者无法获取这些敏感信息，提高了程序控制流的防御可靠性；转换电路对返回地址的至少一个比特进行转换时，对变化区段和不变区段同时进行转换，能够提高攻击者的暴力破解难度，保证程序控制流的

安全性。

[0029] 可选的,所述转换电路具体用于:

[0030] 对所述返回地址的至少一个比特进行分组,得到多个比特组;

[0031] 采用所述转换模型对每个所述比特组中的比特进行转换,得到每个所述比特组对应的转换结果,其中,所述多个比特组中至少存在两个比特组采用的转换模型不同,或者,所述多个比特组采用的转换模型相同;

[0032] 根据各所述比特组对应的转换结果,得到所述转换返回地址。

[0033] 可选的,所述比特组的数量为两个,其中一个比特组中包括所述返回地址的奇数比特位对应的比特,另一个比特组中包括所述返回地址的偶数比特位对应的比特。

[0034] 可选的,所述转换模型的种类包括:模乘法转换模型、模加法转换模型。

[0035] 可选的,所述寄存器为用于存储返回地址的寄存器。

[0036] 可选的,所述处理器为基于ARM指令集的处理器,所述寄存器为LR寄存器。

[0037] 可选的,所述处理器为基于RISC V指令集的处理器,所述寄存器为RA寄存器。

[0038] 第二方面,本申请实施例提供一种返回地址的处理方法,应用于处理器,所述处理器包括:处理核和转换电路,所述方法包括:

[0039] 在所述处理核输出所述返回地址时,通过所述转换电路对所述返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至内存中的栈;

[0040] 在需要使用所述返回地址时,通过所述转换电路对所述栈中的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述处理核。

[0041] 可选的,所述处理器还包括寄存器,所述通过所述转换电路对所述返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至内存中的栈,包括:

[0042] 通过所述转换电路对所述返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至所述寄存器,以使所述转换返回地址经由所述寄存器输出至内存中的栈;

[0043] 所述通过所述转换电路对所述栈中的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述处理核,包括:

[0044] 通过所述转换电路对所述栈中的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述寄存器,以使所述返回地址经由所述寄存器输出至所述处理核。

[0045] 可选的,所述处理器还包括寄存器,所述返回地址被所述处理核输出至所述寄存器,所述通过所述转换电路对所述返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至内存中的栈,包括:

[0046] 通过所述转换电路对所述寄存器输出的所述返回地址进行转换以得到转换返回地址,并将所述转换返回地址输出至内存中的栈;

[0047] 所述通过所述转换电路对所述栈中的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述处理核,包括:

[0048] 通过所述转换电路对所述寄存器输出的所述转换返回地址进行所述转换,得到所述返回地址,并将所述返回地址输出至所述处理核。

[0049] 可选的,所述转换满足如下条件:

[0050] B=IP(A),A=IP(B)

[0051] 其中，A为所述返回地址，B为所述转换返回地址，IP()为所述转换所采用的转换模型。

[0052] 可选的，所述对所述返回地址进行转换以得到转换返回地址，包括：

[0053] 采用至少一种转换模型对所述返回地址的至少一个比特进行转换，得到所述转换返回地址。

[0054] 可选的，所述采用至少一种转换模型对所述返回地址的至少一个比特进行转换，得到所述转换返回地址，包括：

[0055] 对所述返回地址的至少一个比特进行分组，得到多个比特组；

[0056] 采用所述转换模型对每个所述比特组中的比特进行转换，得到每个所述比特组对应的转换结果，其中，所述多个比特组中至少存在两个比特组采用的转换模型不同，或者，所述多个比特组采用的转换模型相同；

[0057] 根据各所述比特组对应的转换结果，得到所述转换返回地址。

[0058] 可选的，所述比特组的数量为两个，其中一个比特组中包括所述返回地址的奇数比特位对应的比特，另一个比特组中包括所述返回地址的偶数比特位对应的比特。

[0059] 可选的，所述转换模型的种类包括：模乘法转换模型、模加法转换模型。

[0060] 可选的，所述寄存器为用于存储返回地址的寄存器。

[0061] 可选的，所述处理器为基于ARM指令集的处理器，所述寄存器为LR寄存器。

[0062] 可选的，所述处理器为基于RISC V指令集的处理器，所述寄存器为RA寄存器。

[0063] 第三方面，本申请实施例提供一种电子设备，包括：如第一方面任一项所述的处理器。

[0064] 第四方面，本申请实施例提供一种芯片，包括：如第一方面任一项所述的处理器。

[0065] 本申请实施例提供的处理器及返回地址的处理方法，在处理器中设置硬件的转换电路，当需要保存返回地址时，利用转换电路对返回地址进行转换，将得到的转换返回地址输出至内存中；当需要使用返回地址时，利用转换电路对内存中的转换返回地址进行转换，得到该返回地址。本申请实施例中，由于攻击者无法知道转换电路中所作的转换操作，使得攻击者无法将内存中的转换返回地址修改为恶意指令对应的转换返回地址，从而，能够防止攻击者对程序控制流的恶意更改。并且，由于上述转换过程是在程序运行过程中通过硬件的转换电路实现，与上述相关技术相比，无需在编译阶段对调用指令和返回指令进行识别，也无需插入额外的加密指令和解密指令，避免了对处理器的运行性能造成影响。

## 附图说明

[0066] 图1为本申请实施例提供的电子设备的结构示意图；

[0067] 图2为本申请实施例提供的处理器的结构示意图；

[0068] 图3为本申请实施例提供的返回地址的处理过程示意图；

[0069] 图4A和图4B为现有的返回地址的处理过程示意图；

[0070] 图5A和图5B为本申请实施例提供的返回地址的处理过程示意图；

[0071] 图6为本申请实施例提供的程序运行过程的示意图；

[0072] 图7A和图7B为本申请实施例提供的返回地址的处理过程示意图；

[0073] 图8为本申请实施例提供的返回地址的处理方法的流程示意图。

## 具体实施方式

[0074] 为了便于对本申请的理解,首先结合图1,对本申请的处理器所适用的电子设备的结构进行说明。

[0075] 图1为本申请实施例提供的电子设备的结构示意图。如图1所示,电子设备10中包括处理器100和存储器(memory)200。其中,存储器200用于存储计算机程序和数据。存储器的种类很多,按其用途可以分为主存储器和辅助存储器。主存储器又称为“内存储器”,简称“内存”,内存用于在处理器运行过程中暂时存储计算机程序以及数据。辅助存储器又称为“外存储器”,简称“外存”,外存用于存储处理器运行过程中暂时不用的计算机程序和数据。处理器100用于执行存储器200中存储的计算机程序。

[0076] 其中,处理器是电子设备的核心器件。处理器通常包括至少一个处理核、缓存以及与电子设备的其他器件通信的输入输出接口。其中,处理核是指处理器中用于执行数据处理任务的处理单元。处理核是处理器中负责运算的主要器件。

[0077] 本申请中的处理器可以是中央处理器(central processing unit,CPU)、图形处理器(graphics processing unit,GPU)等。

[0078] 处理器可用于执行电子设备中的计算机程序。处理器能够识别并执行计算机程序中的指令,使得电子设备完成某种功能或得到某种结果。

[0079] 计算机程序是由多个指令组成的指令序列。计算机程序中还可以调用子程序。子程序也可以称为“子过程”,或者“子函数”。子程序是由一个或者多个指令组成,负责完成某项特定任务,具有相对的独立性。通常,将包含调用子程序的程序称为主程序。主程序和子程序是相对的,例如:程序A中调用了程序B,程序B中又调用了程序C。那么,相对于程序A而言,程序B为子程序,相对于程序C而言,程序B为主程序。

[0080] 计算机程序的正常执行流程,称为程序控制流。处理器在执行计算机程序的过程中,按照程序控制流对各个指令进行执行。然而,处理器在执行计算机程序的过程中,可能出现攻击者恶意劫持程序控制流的事件。这些恶意攻击事件的目的通常是改变程序的控制流,从而破坏程序的控制流完整性(Control Flow Integrity,CFI)。示例性的,一种常见的破坏程序的CFI的恶意攻击事件为面向返回地址的编程(return oriented programming,ROP)攻击。

[0081] 下面结合一个具体的示例描述正常情况下处理器执行计算机程序的过程,以及ROP攻击情况下处理器执行计算机程序的过程。

[0082] 假设计算机程序包括:指令A、指令B和指令C。其中,指令B为子程序调用指令。其中,指令A的地址为0x0000,指令B的地址为0x0004,指令C的地址为0x0008。正常的程序执行流程为依次执行指令A、指令B和指令C。处理器执行该程序的正常流程如下。

[0083] 1)处理器执行指令A。

[0084] 由于指令B为子程序调用指令,处理器执行指令B时会跳转到指令B对应的子程序的地址。为了保证处理器在执行完指令B对应的子程序后,能够正确返回到指令C的地址,处理器在执行指令B之前,会对指令C的地址进行保存。即,

[0085] 2)处理器将指令C的地址写入到内存中。本申请实施例中,将指令C的地址称为返回地址。

[0086] 示例性的,处理器可以将返回地址写入内存中的栈。

[0087] 3)处理器执行指令B,跳转到指令B对应的子程序的地址,对子程序进行执行。

[0088] 4)当子程序执行结束后,从内存中读取返回地址(0x0008),跳转到地址0x0008执行指令C。

[0089] 当存在ROP攻击时,攻击者对内存中存储的返回地址进行篡改。示例性的,攻击者通过远程软件将内存中存储的指令C的地址(0x0008)修改为恶意地址。使得处理器在执行完指令B对应的子程序后,从内存中读取该恶意地址,并跳转到该恶意地址处执行。从而,达到破坏程序控制流完整性的目的。

[0090] 目前,为了防止程序控制流被恶意改变,通常需要对程序运行时的控制流进行监控,若程序控制流被改变,则发出警报。

[0091] 一种相关技术中,采用软件方式对程序控制流的恶意改变进行防御。具体的,在程序编译阶段,首先识别出子程序对应的调用指令和返回指令。示例性的,调用指令为call指令,返回指令为ret指令。然后,在调用指令之前插入一个加密指令。示例性的,加密指令是采用预设的密钥对返回地址进行加密的指令。同时,在返回指令之前插入一个解密指令。示例性的,解密指令是采用相同的密钥对加密后的返回地址进行解密的指令。

[0092] 如此编译之后,在程序运行阶段,处理器调用子程序之前,先执行加密指令对子程序的返回地址进行加密,将得到的加密地址存入内存。在子程序执行完毕后,再利用解密指令对从内存中读取的加密地址进行解密,得到原始的返回地址,从而处理器能够从返回地址处继续执行。

[0093] 具体的加解密方式可以为,定义处理器中的特殊寄存器,专门用于存放密钥,而不能用作其他用途。在加密时,将原始的返回地址与特殊寄存器中的密钥进行异或运算,得到加密的返回地址,并存入内存中。在解密时,将从内存中读取的加密的返回地址与特殊寄存器中的密钥再次进行异或运算,得到原始的返回地址。即,加密指令和解密指令均需进行一次异或运算。

[0094] 采用上述防御技术后,即使攻击者从内存中劫持到加密的返回地址,由于攻击者不知道加密指令采用的密钥,无法将加密的返回地址篡改为加密的恶意地址。也就是说,攻击者劫持程序控制流后仍无法控制程序的跳转位置,因此,能够阻止攻击者对程序控制流进行恶意改变,实现对程序的控制流完整性的保护。

[0095] 然而,上述相关技术中,至少存在如下问题:1)在程序编译阶段需要识别出调用指令和返回指令,并在程序中插入多个额外的加密指令和解密指令,使得程序的运行性能降低。2)由于密钥存在寄存器中,且采用软件方式进行加密和解密,存在软件窃取风险,防护安全性差。3)使用异或运算对每个比特单独操作,使得暴力破解难度低,每次暴力破解都可使程序跳转到代码区,存在安全隐患。4)由于需要特殊寄存器来存储密钥,该特殊寄存器不能被其他功能使用,使得应用场景受限。

[0096] 为了解决上述问题中的至少一个,本申请实施例提供一种处理器,在处理器中设置硬件的转换电路,当需要保存返回地址时,利用转换电路对返回地址进行转换,将得到的转换返回地址输出至内存中;当需要使用返回地址时,利用转换电路对内存中的转换返回地址进行转换,得到该返回地址。本申请实施例中,由于攻击者无法知道转换电路中所作的转换操作,使得攻击者无法将内存中的转换返回地址修改为恶意指令对应的转换返回地址,从而,能够防止攻击者对程序控制流的恶意更改。并且,由于上述转换过程是在程序运

行过程中通过硬件的转换电路实现，与上述相关技术相比，无需在编译阶段对调用指令和返回指令进行识别，也无需插入额外的加密指令和解密指令，避免了对处理器的运行性能造成影响。

[0097] 下面，通过具体实施例，对本申请所示的技术方案进行详细说明。需要说明的是，下面几个实施例可以单独存在，也可以相互结合，对于相同或相似的内容，在不同的实施例中不再重复说明。

[0098] 图2为本申请实施例提供的处理器的结构示意图。如图2所示，本实施例的处理器100，包括：处理核110和转换电路120。

[0099] 其中，处理核110用于输出返回地址。

[0100] 转换电路120用于对处理核110输出的返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至内存中的栈。

[0101] 转换电路120还用于在处理核110需要使用所述返回地址时，对所述栈中的所述转换返回地址进行所述转换，得到所述返回地址，并将所述返回地址输出至处理核110。

[0102] 本申请实施例对于处理器的类型不作具体限定。本申请实施例中的处理器可以为但不限于下述类型：基于ARM指令集的处理器、基于RISC-V指令集的处理器。

[0103] 本申请实施例对于处理器的位宽不作具体限定。处理器可以为32位处理器，也可以为64位处理器，当然，还可以为其他位宽的处理器。

[0104] 其中，处理器中的处理核的数量可以为一个或者多个。处理核是指处理器中用于执行数据处理任务的处理单元。当处理核的数量为一个时，处理器为单核处理器。当处理核的数量为多个时，处理器为多核处理器。处理核用于输出返回地址。返回地址为处理核下一条待执行指令的地址。

[0105] 本申请中，处理核是需要将返回地址存入内存、以及需要从内存获取返回地址的任一电路。示例性的，处理核为程序计数器(Program Counter，PC)。PC也可称为指令计数器。PC用于存放处理器下一条待执行指令的地址。在程序开始执行前，处理器将程序的起始地址，即程序的第一条指令的地址送入PC。当执行指令时，处理器将自动修改PC中的值，即每执行一条指令，将PC中的值增加一个量，使其PC中的值始终指向待执行的下一条指令的地址。

[0106] 本申请中，转换电路设置在处理核和内存之间的数据通路上。下面结合图3描述转换电路对返回地址的处理过程。

[0107] 图3为本申请实施例提供的返回地址的处理过程示意图。如图3所示，当处理核需要将返回地址存入内存时，处理核输出的返回地址经过转换电路，转换电路对该返回地址进行转换得到转换返回地址，然后将该转换返回地址输出至内存中的栈。当处理核需要使用该返回地址时，从内存中出栈的所述转换返回地址经过转换电路，转换电路对该转换地址进行相同的转换，得到原始的返回地址，然后，将该原始的返回地址输出至处理核。

[0108] 其中，处理核需要将返回地址存入内存，可以是指处理核执行调用指令之前，需要将该调用指令对应的返回地址存入内存。

[0109] 相应的，处理核需要使用返回地址，可以是指处理核从调用指令对应的子程序返回时，需要从内存中获取该调用指令对应的返回地址。

[0110] 可以理解的，本申请实施例中，转换电路可以采用一种或者多种转换模型对返回

地址进行转换。本实施例对此不作具体限定，几种可能的转换方式可以参见后续实施例的详细描述。

[0111] 本申请中，在返回地址存入内存之前，转换电路对返回地址进行一次转换，因此，内存中存储的是转换后的返回地址。由于攻击者无法知道转换电路中所作的转换操作，使得攻击者无法将内存中的转换返回地址修改为恶意指令对应的转换返回地址，从而，能够防止攻击者对程序控制流的恶意更改。

[0112] 在转换后的返回地址从内存出栈之后，转换电路对转换后返回地址再进行一次转换，得到原始的返回地址，使得处理核可以根据原始的返回地址进行后续指令的执行，保证了程序控制流的完整性。

[0113] 本申请中，由于上述转换过程是在程序运行过程中通过硬件的转换电路实现，与上述相关技术相比，无需在编译阶段对调用指令和返回指令进行识别，也无需插入额外的加密指令和解密指令，避免了对处理器的运行性能造成影响。同时，与上述相关技术相比，无需专门的特殊寄存器来存储密钥，规避了软件窃取风险。

[0114] 一种可能的实施方式中，处理器中还包括控制电路，控制电路能够识别处理核输出的地址是否为返回地址，当控制电路识别出处理核输出的地址为返回地址时，则控制该返回地址输入转换电路中。转换电路对返回地址进行转换后得到转换返回地址，并将转换返回地址输出至内存中的栈。当控制电路识别出处理核需要使用该返回地址时，在控制电路的控制下，从内存中出栈的转换返回地址被输入转换电路。转换电路对转换返回地址进行转换，得到原始的返回地址，并将原始的返回地址输出至处理核。

[0115] 另一种可能的实施方式中，处理器并不感知转换电路的存在。转换电路设置处理核至内存之间的数据通路上，即，当返回地址从处理核传输至内存、以及从内存传输至处理核的过程中，均会经过转换电路。该实施方式仅需要在处理核与内存之间的数据通路上设置转换电路，而无需改变处理器的现有控制流程，易于实施。

[0116] 可选的，处理器中还包括寄存器，该寄存器为专门用于存储返回地址的寄存器。

[0117] 对于该类处理器，上述寄存器是返回地址从处理核至内存之间的必经之路。示例性的，在需要对子程序的返回地址输出至内存时，处理核首先将返回地址输出至上述寄存器中，然后寄存器将该返回地址输出至内存中。当需要使用该返回地址时，从内存中出栈的返回地址也先输出至上述寄存器中，然后，寄存器再将该返回地址输出至处理核。

[0118] 可以理解的，针对不同指令集的处理器，用于存储返回地址的寄存器可能有所不同。

[0119] 示例性的，所述处理器为基于ARM指令集的处理器，所述寄存器为连接寄存器(link register,LR)。

[0120] 示例性的，所述处理器为基于RISC V指令集的处理器，所述寄存器为返回地址(return address,RA)寄存器。

[0121] 下面以ARM指令集的处理器为例，描述处理器的结构以及返回地址的处理过程。图4A和图4B为现有的返回地址的处理过程示意图。其中，图4A示例的是返回地址入栈的过程示意图，图4B示例的是返回地址出栈的过程示意图。

[0122] 处理核执行调用指令之前，需要将该调用指令对应的返回地址存入内存。示例性的，如图4A所示，在处理核的控制下，PC输出的返回地址进入LR寄存器。然后，LR寄存器输出

的返回地址被存入内存中的栈。

[0123] 当处理核从调用指令对应的子程序返回时，需要从内存中读取返回地址。示例性的，如图4B所示，从内存出栈的返回地址进入LR寄存器。然后，LR寄存器将返回地址输出至处理核。

[0124] 本实施例中的转换电路可以设置在上述寄存器之前，也可以设置在上述寄存器之后。

[0125] 需要说明的是，本实施例中，转换电路设置在寄存器之前，是指转换电路设置在寄存器的写入通路上，即，设置在寄存器的输入端。转换电路设置在寄存器之前时，意味着进入该寄存器的所有返回地址会先经过转换电路，然后再进入寄存器。本实施例中，转换电路设置在寄存器之后，是指转换电路设置在寄存器的读出通路上，即，设置在寄存器的输出端。转换电路设置在寄存器之后时，意味着从该寄存器输出的所有返回地址均会经过转换电路。

[0126] 下面对上述的两种实施方式分别进行描述。

[0127] 图5A和图5B为本申请实施例提供的返回地址的处理过程示意图。本实施例中，将转换电路设置在LR寄存器的写入通路上。其中，图5A示例的是返回地址入栈的过程，图5B示例的是返回地址出栈的过程。

[0128] 所述转换电路具体用于对所述处理核输出的返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至所述寄存器，以使所述转换返回地址经由所述寄存器输出至内存中的栈。

[0129] 所述转换电路还具体用于在所述处理核需要使用所述返回地址时，对所述栈中的所述转换返回地址进行所述转换，得到所述返回地址，并将所述返回地址输出至所述寄存器，以使所述返回地址经由所述寄存器输出至所述处理核。

[0130] 如图5A所示，由于转换电路设置在寄存器的写入通路上，处理核输出的返回地址在输出至LR寄存器的过程中，会先经过该转换电路。转换电路对处理核输出的返回地址进行转换，得到转换返回地址，使得实际存入LR寄存器的是转换返回地址。然后，LR寄存器将转换返回地址输出至内存中的栈，即实际入栈的是转换返回地址。由于攻击者无法知道转换电路中所作的转换操作，使得攻击者无法将内存中的转换返回地址修改为恶意指令对应的转换返回地址，从而，能够防止攻击者对程序控制流的恶意更改。

[0131] 如图5B所示，当处理核需要使用该返回地址时，由于转换电路设置在LR寄存器的写入通路上，从内存出栈的转换返回地址在进入LR寄存器之前，会先经过转换电路。转换电路对该转换返回地址进行转换，得到原始的返回地址，使得实际存入LR寄存器的是原始的返回地址。然后，LR寄存器将该原始的返回地址输出至处理核。从而保证程序的正常控制流。

[0132] 下面以基于ARM指令集的处理器为例，结合一段实际程序的运行过程，进行举例说明。

[0133] 图6为本申请实施例提供的程序运行过程的示意图。如图6所示，该程序包括如下ARM指令：SUB、STP、ADD、LDP、ADD、以及RET。

[0134] 控制流准备进入子程序，即处理核执行STP指令时，将返回地址输出至LR(0x30)寄存器。在返回地址进入LR寄存器之前，返回地址先经过转换电路，转换电路对该返回地址进

行转换，得到转换返回地址，使得实际进入LR寄存器的是转换返回地址。之后，LR寄存器将转换返回地址输出至内存中的栈。

[0135] 控制流从子程序的结尾返回时，即处理核执行LDP指令时，从内存的栈中读取转换返回地址。该转换返回地址在进入LR(0x30)寄存器之前，会先经过转换电路，转换电路对该转换返回地址再次进行转换，得到原始的返回地址。该原始的返回地址被存入LR寄存器中，并在处理核执行RET指令时使用该返回地址。

[0136] 本实施例中，在LR寄存器的写入通路上设置硬件的转换电路，该转换电路用于对进入LR寄存器的返回地址进行转换，即，所有需要进入LR寄存器的输入都会先经过该转换电路的转换，然后将转换结果输入至LR寄存器中。通过该方式，能够自动识别返回地址，无需改变处理器的控制流程，易于实施。

[0137] 图7A和图7B为本申请实施例提供的返回地址的处理过程示意图。本实施例中，将转换电路设置在LR寄存器的读出通路上。其中，图7A示例的是返回地址入栈的过程，图7B示例的是返回地址出栈的过程。

[0138] 在所述处理核输出所述返回地址时，所述寄存器用于寄存所述处理核输出的返回地址。所述转换电路具体用于对所述寄存器输出的所述返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至内存中的栈。

[0139] 在所述处理核需要使用所述返回地址时，所述寄存器还用于寄存所述栈输出的所述转换返回地址。所述转换电路还具体用于对所述寄存器输出的所述转换返回地址进行所述转换，得到所述返回地址，并将所述返回地址输出至所述处理核。

[0140] 如图7A所示，处理核输出的返回地址输出至LR寄存器。由于转换电路设置在LR寄存器的读出通路上，在LR寄存器将该返回地址输出至内存中的栈的过程中，会经过该转换电路。转换电路对LR寄存器输出的返回地址进行转换，得到转换返回地址，使得实际入栈的转换返回地址。由于攻击者无法知道转换电路中所作的转换操作，使得攻击者无法将内存中的转换返回地址修改为恶意指令对应的转换返回地址，从而，能够防止攻击者对程序控制流的恶意更改。

[0141] 如图7B所示，当处理核需要使用该返回地址时，从内存出栈的转换返回地址进入LR寄存器。由于转换电路设置在LR寄存器的读出通路上，在LR寄存器将转换返回地址输出至处理核的过程中，会经过转换电路。转换电路对该转换返回地址进行转换，得到原始的返回地址，使得实际输入至处理核的是原始的返回地址。从而保证程序的正常控制流。

[0142] 本实施例中，在LR寄存器的读出通路上设置硬件的转换电路，该转换电路用于对LR寄存器输出的返回地址进行转换，即，所有从LR寄存器输出的值都会先经过该转换电路的转换。通过该方式，能够自动识别返回地址，无需改变处理器的控制流程，易于实施。

[0143] 需要说明的是，上述实施例中是以基于ARM指令集的处理器为例进行描述。但是本申请实施例所适应的处理器并不限于此。本申请实施例对于其他指令集的处理器同样适用，只要处理器中存在专门用于存储返回地址的寄存器即可。

[0144] 需要说明的是，上述各实施例中，对于基于ARM指令集的处理器和基于RISC-V指令集的处理器，如果编译器支持末端分支(leaf subroutine)优化选项(即在leaf subroutine中不将返回地址从LR/RA寄存器保存至内存中的栈，而是将其一直保存在LR/RA寄存器中)，则需要关闭此优化功能，保证LR/RA在leaf subroutine中的使用方式与一般分

支(subroutine)中没有区别。否则当返回地址从LR/RA寄存器读出时未作解密,会影响子程序的正常返回。

[0145] 下面对上述各实施例中的转换电路的具体转换过程进行介绍。

[0146] 本申请实施例中,转换电路对返回地址所进行的转换,满足如下条件:

[0147] B＝IP(A),A＝IP(B)

[0148] 其中,A为所述返回地址,B为所述转换返回地址,IP()为所述转换所采用的转换模型。

[0149] 可以理解的,满足上述条件的转换模型可以有多种,包括但不限于:异或转换模型、模乘法转换模型、模加法转换模型。

[0150] 一种可能的实施方式中,转换电路中存储有多种转换模型,不同的转换模型对应一组或者多组可选的转换参数。通过将转换模型及转换参数存在硬件电路中,使得攻击者无法获取这些敏感信息,提高了程序控制流的防御可靠性。

[0151] 转换电路对返回地址进行转换时,采用至少一种转换模型对返回地址的至少一个比特进行转换,得到所述转换返回地址。

[0152] 其中,所述至少一个比特包括程序的代码地址的变化区段对应的比特和不变区段对应的比特;所述程序的代码地址包括多个指令的指令地址,所述不变区段为多个指令地址中比特相同的比特位,所述变化区段为多个指令地址中比特不同的比特位。

[0153] 示例性的,以libc的代码区为例,由于程序中的指令数量有限,比较程序的代码地址的起始地址和结束地址可以发现,代码地址的几个高位比特、甚至十几或者几十个高位比特是不会变化的。本申请中将这些不会变化的比特位称为代码地址的不变区段。相应的,代码地址会变化的比特位,称为代码地址的变化区段。

[0154] 本申请实施例中,转换电路对返回地址的至少一个比特进行转换时,对变化区段和不变区段同时进行转换,能够提高攻击者的暴力破解难度,保证程序控制流的安全性。

[0155] 示例性的,在运行程序之前,转换电路随机选择一种转换模型,并随机选择一组转换参数,对程序运行过程中的返回地址进行转换。若程序运行发生错误,即出现ROP攻击的情况下,则在重新运行程序时,更换转换模型和/或转换参数,使得攻击者无法针对同一转换模型及转换参数进行多次攻击,进一步提高程序控制流的安全性。

[0156] 下面结合几种具体的实施方式,描述处理器对返回地址的处理过程。

[0157] 一种可能的实施方式中,假设处理器为32位的基于ARM指令集的处理器,转换电路设置在LR寄存器的写入通路上,转换电路中采用模乘法转换模型。处理器对返回地址的处理过程如下:

[0158] 1)在处理核执行call指令(例如调用func函数)之前,会将返回地址存入LR寄存器中。由于转换电路设置在寄存器的写入通路上,因此,返回地址会首先经过转换电路的转换,得到转换返回地址,使得实际进入LR寄存器的为转换返回地址。

[0159] 为了表述方便,本实施例中将处理核输出的返回地址(即原始的返回地址)记为a[31:0]。将转换电路转换得到的转换返回地址(即实际进入LR寄存器的返回地址)记为b[31:0]。转换电路进行如下所示的模乘法操作:

[0160] $a \times q \equiv b \bmod p$

[0161] 其中,p和q为模乘法转换模型对应的转换参数。由上式可知,返回地址a和转换电

路中的参数q进行模乘法，模为p，输出为转换返回地址b。其中，q和p满足如下关系：

[0162] $q^2 \equiv 1 \bmod p$

[0163] 可以理解的，满足上述关系的p和q可以有多种组合。例如：如果p＝2^{32}，那么q可选择为下述中的任一：

[0164] 4294967295,2147483649,2147483647

[0165] 实际应用中，p应大于最大的代码地址但不宜过大，以节约转换电路的芯片面积。另外，当q＝p-1或者q＝1时，总是满足上述要求，但为了安全性，q不选择1。

[0166] 本实施例中，由于模乘法转换模型是将32位的返回地址a[31:0]做整体运算，即代码地址的变化区段和不变区段同时进行运算，使得攻击者无法仅针对变化区段进行特定攻击，攻击者无法保证每次暴力破解都能跳转到代码区，提高了程序控制流的安全性。

[0167] 2)将转换返回地址从LR寄存器读出，并存入内存的栈中。

[0168] 3)执行函数func。

[0169] 4)在从函数func返回之前，从内存中读取转换返回地址，将出栈的转换返回地址存入LR寄存器中。由于LR寄存器的写入通路上设置了转换电路，出栈的转换返回地址会先经过转换电路。转换电路对转换返回地址再进行一次转换，得到原始的返回地址，然后原始的返回地址被存入LR寄存器中。转换电路进行同样的模乘法操作，如下：

[0170] $b \times q \equiv a \bmod p$

[0171] 其中，p和q为模乘法转换模型对应的转换参数，与上述的第一次转换过程采用的参数相同。转换返回地址b为转换模型的输入，返回地址a为转换模型的输出。

[0172] 由上述两次转换过程可知，返回地址经过两次同样的转换后变回原值。

[0173] 5)执行返回操作。

[0174] 若转换返回地址b在内存的栈中未被恶意篡改，则经过转换电路的转换后，得到的返回地址a是正确的返回地址，程序正常执行。若转换返回地址b在内存的栈中被恶意篡改，由于攻击者无法知晓敏感信息p和q，无法构造出合法的恶意转换返回地址，因此，经过转换电路的转换后，得到的返回地址a是乱码，执行返回操作时，程序将执行乱码地址处的代码，会大概率产生内存错误。从而，攻击者无法达到更改程序控制流的目的。

[0175] 另一种可能的实施方式中，假设处理器为32位的基于ARM指令集的处理器，转换电路设置在LR寄存器的读出通路上，转换电路中采用模加法转换模型。处理器对返回地址的处理过程如下：

[0176] 1)在处理核执行call指令(例如调用func函数)之前，将返回地址存入LR寄存器中。

[0177] 2)将返回地址从LR寄存器读出，并存入内存的栈中。

[0178] 由于转换电路设置在LR寄存器的读出通路上，因此，LR寄存器输出的返回地址会先经过转换电路的转换，得到转换返回地址，使得实际入栈的为转换返回地址。

[0179] 为了表述方便，本实施例中将LR寄存器输出的返回地址(即原始的返回地址)记为a[31:0]。将转换电路转换得到的转换返回地址(即实际入栈的返回地址)记为b[31:0]。转换电路进行如下式所示的模加法操作:

[0180] $a+q \equiv b \bmod p$

[0181] 其中，p和q为模加法转换模型对应的转换参数。由上式可知，返回地址a和转换电

路中的参数q进行模加法，模为p，输出为转换返回地址b。其中，q和p满足如下关系：

[0182] $q \equiv p/2$

[0183] 可以理解的，满足上述关系的p和q可以有多种组合。例如：如果p＝2^{32}+2，那么q＝2^{31}+1。

[0184] 实际应用中，p应大于最大的代码地址但不宜过大，以节约转换电路的芯片面积。

[0185] 本实施例中，由于模加法转换模型是将32位的返回地址a[31:0]做整体运算，即代码地址的变化区段和不变区段同时进行运算，使得攻击者无法仅针对变化区段进行特定攻击，攻击者无法保证每次暴力破解都能跳转到代码区，提高了程序控制流的安全性。

[0186] 3)执行函数func。

[0187] 4)在从函数func返回之前，从内存中读取转换返回地址，将出栈的转换返回地址存入LR寄存器中。

[0188] 5)将转换返回地址从LR寄存器中读出，并输出至处理核，执行返回操作。

[0189] 由于LR寄存器的读出通路上设置了转换电路，出栈的转换返回地址会先经过转换电路。转换电路对转换返回地址再进行一次转换，得到原始的返回地址，然后原始的返回地址被输出至处理核。转换电路进行同样的模加法操作，如下：

[0190] $b+q \equiv a \bmod p$

[0191] 其中，p和q为模加法转换模型对应的转换参数，与上述的第一次转换过程采用的参数相同。转换返回地址b为转换模型的输入，返回地址a为转换模型的输出。

[0192] 由上述两次转换过程可知，返回地址经过两次同样的转换后变回原值。

[0193] 若转换返回地址b在内存的栈中未被恶意篡改，则经过转换电路的转换后，得到的返回地址a是正确的返回地址，程序正常执行。若转换返回地址b在内存的栈中被恶意篡改，由于攻击者无法知晓敏感信息p和q，无法构造出合法的恶意转换返回地址，因此，经过转换电路的转换后，得到的返回地址a是乱码，执行返回操作时，程序将执行乱码地址处的代码，会大概率产生内存错误。从而，攻击者无法达到更改程序控制流的目的。

[0194] 再一种可能的实施方式中，转换电路对所述返回地址的至少一个比特进行分组，得到多个比特组；采用所述转换模型对每个所述比特组中的比特进行转换，得到每个所述比特组对应的转换结果，其中，所述多个比特组中至少存在两个比特组采用的转换模型不同，或者，所述多个比特组采用的转换模型相同；然后，根据各所述比特组对应的转换结果，得到所述转换返回地址。

[0195] 可以理解的，对返回地址的至少一个比特进行分组的方式可以有多种，本实施例对此不作具体限定。以32位处理器系统为例，可以将返回地址的32个比特分为两个比特组，还可以分为三个比特组，当然，还可以分为更多个比特组。

[0196] 示例性的，当分为两个比特组时，可以将前16个比特作为一组，将后16个比特作为一组；也可以将前8位比特作为一组，将后24位比特作为一组；还可以将奇数比特位作为一组，将偶数比特位作为一组。可以理解的，还存在其他的分组方式，此处不一一列举。当分为两个比特组时，这两个比特组采用的转换模型可以相同，也可以不同。例如：两个比特组均采用模乘法转换模型，或者，两个比特组均采用模加法转换模型，或者，一个比特组采用模乘法转换模型，一个比特组采用模加法转换模型。

[0197] 示例性的，当分为三个比特组时，可以将前8个比特作为一组，将中间16个比特作

为一组，将后8个比特作为一组；也可以将前10个比特作为一组，将中间8个比特作为一组，将后14个比特作为一组；还可以将第1、4、7、10、13、16、19、22、25、28、31个比特作为一组，将第2、5、8、11、14、17、20、23、26、29个比特作为一组，将第0、3、6、9、12、15、18、21、24、27、30个比特作为一组。可以理解的，还存在其他的分组方式，此处不一一列举。当分为三个比特组时，三个比特组所采用的转换模型可以相同，也可以不同。

[0198] 下面结合举例进行说明。

[0199] 假设处理器为32位的基于ARM指令集的处理器，转换电路设置在LR寄存器的写入通路上。转换电路将返回地址的32个比特划分为两个比特组，一个比特组中包括所述返回地址的奇数比特位对应的比特，另一个比特组中包括所述返回地址的偶数比特位对应的比特。其中一个比特组采用模乘法转换模型，另一个比特组采用模加法转换模型。则处理器对返回地址的处理过程如下：

[0200] 1)在处理核执行call指令(例如调用func函数)之前，会将返回地址存入LR寄存器中。由于转换电路设置在寄存器的写入通路上，因此，返回地址会首先经过转换电路的转换，得到转换返回地址，使得实际进入LR寄存器的为转换返回地址。

[0201] 为了表述方便，本实施例中将处理核输出的返回地址(即原始的返回地址)记为a[31:0]。将a[31:0]划分为两个比特组，分别为$a_1$[15:0]和$a_2$[15:0]，其中，

[0202] $a_1[15:0]\equiv\{a[31],a[29],a[27],\cdots,a[1]\}$

[0203] $a_2[15:0]\equiv\{a[30],a[28],a[26],\cdots,a[0]\}$

[0204] 相应的，将转换电路转换得到的转换返回地址(即实际进入LR寄存器的返回地址)记为b[31:0]。将b[31:0]划分为两个比特组，分别为$b_1$[15:0]和$b_2$[15:0]，其中，

[0205] $b_1[15:0]\equiv\{b[31],b[29],b[27],\cdots,b[1]\}$

[0206] $b_2[15:0]\equiv\{b[30],b[28],b[26],\cdots,b[0]\}$

[0207] 转换电路在对返回地址进行转换时，对$a_1$[15:0]采用模乘法转换模型，对$a_2$[15:0]采用模加法转换模型，如下所示：

[0208] $a_1\times q_1\equiv b_1 \bmod p_1$

[0209] $a_2+q_2\equiv b_2 \bmod p_2$

[0210] 其中，$p_1$和$q_1$为模乘法转换模型对应的转换参数，$p_2$和$q_2$为模加法转换模型对应的转换参数。由上式可知，$a_1$和转换电路中的参数$q_1$进行模乘法，模为$p_1$，输出为$b_1$。$a_2$和转换电路中的参数$q_2$进行模加法，模为$p_2$，输出为$b_2$。

[0211] 其中，$p_1$和$q_1$满足如下关系：

[0212] $q_1^2\equiv 1 \bmod p_1$

[0213] 可以理解的，满足上述关系的$p_1$和$q_1$可以有多种组合。例如：如果$p_1=2^{16}+8$，那么$q_1$可选择为下述中的任一：

[0214] 65543,60083,54619,49159,…,5461

[0215] $p_2$和$q_2$满足如下关系：

[0216] $q_2\equiv p_2/2$

[0217] 可以理解的，满足上述关系的$p_2$和$q_2$可以有多种组合。例如：如果$p_2=2^{32}+6$，那么$q_2=2^{31}+3$。

[0218] 实际应用中，$p_1$和$p_2$应大于最大的代码地址但不宜过大，以节约转换电路的芯片面

积。

[0219] 本实施例中，针对32位的返回地址a，将32个比特按照奇数比特位和偶数比特位划分为两组，其中一组采用模乘法转换模型，另一组采用模加法转换模型。可见，对代码地址的变化区段和不变区段同时进行运算，使得攻击者无法仅针对变化区段进行特定攻击，攻击者无法保证每次暴力破解都能跳转到代码区，提高了程序控制流的安全性。

[0220] 2)将转换返回地址从LR寄存器读出，并存入内存的栈中。

[0221] 3)执行函数func。

[0222] 4)在从函数func返回之前，从内存中读取转换返回地址，将出栈的转换返回地址存入LR寄存器中。由于LR寄存器的写入通路上设置了转换电路，出栈的转换返回地址会先经过转换电路。转换电路对转换返回地址再进行一次转换，得到原始的返回地址，然后原始的返回地址被存入LR寄存器中。

[0223] 转换电路在对转换返回地址进行转换时，对$b_1$[15:0]采用模乘法转换模型，对$b_2$[15:0]采用模加法转换模型，如下所示:

[0224] $b_1 \times q_1 \equiv a_1 \bmod p_1$

[0225] $b_2 + q_2 \equiv a_2 \bmod p_2$

[0226] 其中，$p_1$和$q_1$为模乘法转换模型对应的转换参数，与上述的第一次转换过程采用的参数相同。$p_2$和$q_2$为模加法转换模型对应的转换参数，与上述的第一次转换过程采用的参数相同。

[0227] 由上述两次转换过程可知，返回地址经过两次同样的转换后变回原值。

[0228] 5)执行返回操作。

[0229] 若转换返回地址b在内存的栈中未被恶意篡改，则经过转换电路的转换后，得到的返回地址a是正确的返回地址，程序正常执行。若转换返回地址b在内存的栈中被恶意篡改，由于攻击者无法知晓敏感信息$p_1$、$q_1$、$p_2$和$q_2$，无法构造出合法的恶意转换返回地址，因此，经过转换电路的转换后，得到的返回地址a是乱码，执行返回操作时，程序将执行乱码地址处的代码，会大概率产生内存错误。从而，攻击者无法达到更改程序控制流的目的。

[0230] 图8为本申请实施例提供的返回地址的处理方法的流程示意图。本实施例的方法由处理器执行，其中，处理器包括:处理核和转换电路。如图8所示，本实施例的方法，包括:

[0231] S801:在所述处理核输出所述返回地址时，通过所述转换电路对所述返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至内存中的栈。

[0232] S802:在需要使用所述返回地址时，通过所述转换电路对所述栈中的所述转换返回地址进行所述转换，得到所述返回地址，并将所述返回地址输出至所述处理核。

[0233] 一种可能的实施方式中，所述处理器还包括寄存器，所述通过所述转换电路对所述返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至内存中的栈，包括:

[0234] 通过所述转换电路对所述返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至所述寄存器，以使所述转换返回地址经由所述寄存器输出至内存中的栈;

[0235] 所述通过所述转换电路对所述栈中的所述转换返回地址进行所述转换，得到所述返回地址，并将所述返回地址输出至所述处理核，包括:

[0236] 通过所述转换电路对所述栈中的所述转换返回地址进行所述转换，得到所述返回

地址，并将所述返回地址输出至所述寄存器，以使所述返回地址经由所述寄存器输出至所述处理核。

[0237] 一种可能的实施方式中，所述处理器还包括寄存器，所述返回地址被所述处理核输出至所述寄存器，所述通过所述转换电路对所述返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至内存中的栈，包括：

[0238] 通过所述转换电路对所述寄存器输出的所述返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至内存中的栈；

[0239] 所述通过所述转换电路对所述栈中的所述转换返回地址进行所述转换，得到所述返回地址，并将所述返回地址输出至所述处理核，包括：

[0240] 通过所述转换电路对所述寄存器输出的所述转换返回地址进行所述转换，得到所述返回地址，并将所述返回地址输出至所述处理核。

[0241] 一种可能的实施方式中，所述转换满足如下条件：

[0242] B＝IP(A)，A＝IP(B)

[0243] 其中，A为所述返回地址，B为所述转换返回地址，IP()为所述转换所采用的转换模型。

[0244] 一种可能的实施方式中，所述对所述返回地址进行转换以得到转换返回地址，包括：

[0245] 采用至少一种转换模型对所述返回地址的至少一个比特进行转换，得到所述转换返回地址。

[0246] 一种可能的实施方式中，所述采用至少一种转换模型对所述返回地址的至少一个比特进行转换，得到所述转换返回地址，包括：

[0247] 对所述返回地址的至少一个比特进行分组，得到多个比特组；

[0248] 采用所述转换模型对每个所述比特组中的比特进行转换，得到每个所述比特组对应的转换结果，其中，所述多个比特组中至少存在两个比特组采用的转换模型不同，或者，所述多个比特组采用的转换模型相同；

[0249] 根据各所述比特组对应的转换结果，得到所述转换返回地址。

[0250] 一种可能的实施方式中，所述比特组的数量为两个，其中一个比特组中包括所述返回地址的奇数比特位对应的比特，另一个比特组中包括所述返回地址的偶数比特位对应的比特。

[0251] 一种可能的实施方式中，所述转换模型的种类包括：模乘法转换模型、模加法转换模型。

[0252] 一种可能的实施方式中，所述寄存器为用于存储返回地址的寄存器。

[0253] 一种可能的实施方式中，所述处理器为基于ARM指令集的处理器，所述寄存器为LR寄存器。

[0254] 一种可能的实施方式中，所述处理器为基于RISC V指令集的处理器，所述寄存器为RA寄存器。

[0255] 本实施例提供的返回地址的处理方法，可应用于上述任一实施例所述的处理器，其实现原理和技术效果类似，此处不再赘述。

[0256] 本申请实施例还提供一种电子设备，包括：处理器，其中，处理器可以采用上述任

一实施例中的处理器的结构,其实现原理和技术效果类似,本实施例此处不再赘述。

[0257] 本申请实施例还提供一种芯片,包括:处理器,所述处理器可以采用上述任一实施例中的处理器的结构,其实现原理和技术效果类似,此处不再赘述。

[0258] 在本申请所提供的几个实施例中,应该理解到,所揭露的设备和方法,可以通过其它的方式实现。例如,以上所描述的设备实施例仅仅是示意性的,例如,所述模块的划分,仅仅为一种逻辑功能划分,实际实现时可以有另外的划分方式,例如多个模块可以结合或者可以集成到另一个系统,或一些特征可以忽略,或不执行。另一点,所显示或讨论的相互之间的耦合或直接耦合或通信连接可以是通过一些接口,装置或模块的间接耦合或通信连接,可以是电性,机械或其它的形式。

[0259] 所述作为分离部件说明的模块可以是或者也可以不是物理上分开的,作为模块显示的部件可以是或者也可以不是物理单元,即可以位于一个地方,或者也可以分布到多个网络单元上。可以根据实际的需要选择其中的部分或者全部模块来实现本实施例方案的目的。

[0260] 另外,在本申请各个实施例中的各功能模块可以集成在一个处理单元中,也可以是各个模块单独物理存在,也可以两个或两个以上模块集成在一个单元中。上述模块成的单元既可以采用硬件的形式实现,也可以采用硬件加软件功能单元的形式实现。

[0261] 上述以软件功能模块的形式实现的集成的模块,可以存储在一个计算机可读取存储介质中。上述软件功能模块存储在一个存储介质中,包括若干指令用以使得一台计算机设备(可以是个人计算机,服务器,或者网络设备等)或处理器(英文:processor)执行本申请各个实施例所述方法的部分步骤。

[0262] 应理解,上述处理器可以是中央处理单元(英文:Central Processing Unit,简称:CPU),还可以是其他通用处理器、数字信号处理器(英文:Digital Signal Processor,简称:DSP)、专用集成电路(英文:Application Specific Integrated Circuit,简称:ASIC)等。通用处理器可以是微处理器或者该处理器也可以是任何常规的处理器等。结合申请所公开的方法的步骤可以直接体现为硬件处理器执行完成,或者用处理器中的硬件及软件模块组合执行完成。

[0263] 存储器可能包含高速RAM存储器,也可能还包括非易失性存储NVM,例如至少一个磁盘存储器,还可以为U盘、移动硬盘、只读存储器、磁盘或光盘等。

[0264] 总线可以是工业标准体系结构(Industry Standard Architecture,ISA)总线、外部设备互连(Peripheral Component,PCI)总线或扩展工业标准体系结构(Extended Industry Standard Architecture,EISA)总线等。总线可以分为地址总线、数据总线、控制总线等。为便于表示,本申请附图中的总线并不限定仅有一根总线或一种类型的总线。

[0265] 上述存储介质可以是由任何类型的易失性或非易失性存储设备或者它们的组合实现,如静态随机存取存储器(SRAM),电可擦除可编程只读存储器(EEPROM),可擦除可编程只读存储器(EPROM),可编程只读存储器(PROM),只读存储器(ROM),磁存储器,快闪存储器,磁盘或光盘。存储介质可以是通用或专用计算机能够存取的任何可用介质。

[0266] 一种示例性的存储介质耦合至处理器,从而使处理器能够从该存储介质读取信息,且可向该存储介质写入信息。当然,存储介质也可以是处理器的组成部分。处理器和存储介质可以位于专用集成电路(Application Specific Integrated Circuits,简称:

ASIC)中。当然,处理器和存储介质也可以作为分立组件存在于电子设备或主控设备中。

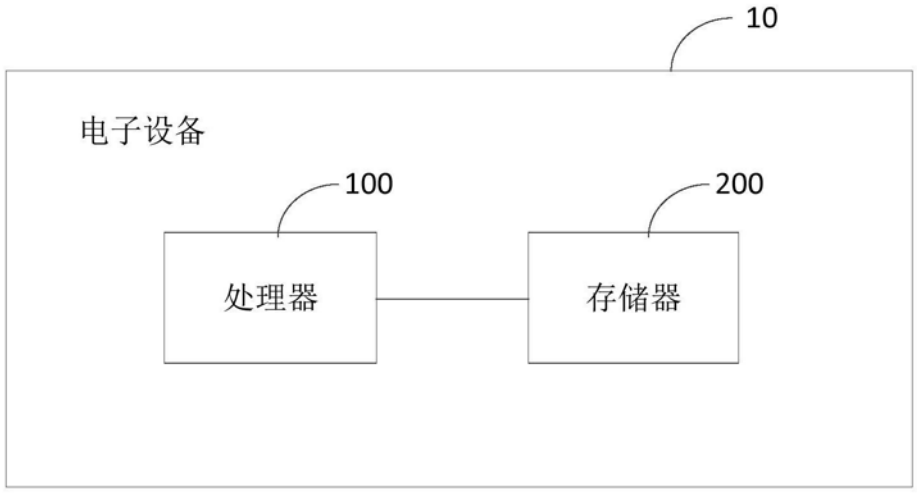
10
电子设备
100
200
处理器
存储器

图1

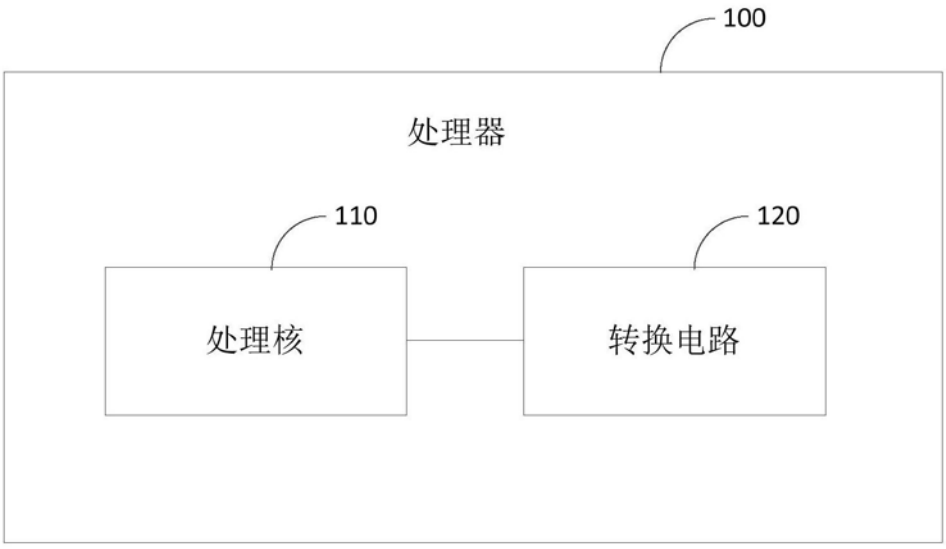
100
处理器
110
120
处理核
转换电路

图2


处理核
返回地址
返回地址
转换电路
转换返回地址
转换返回地址
内存

图3

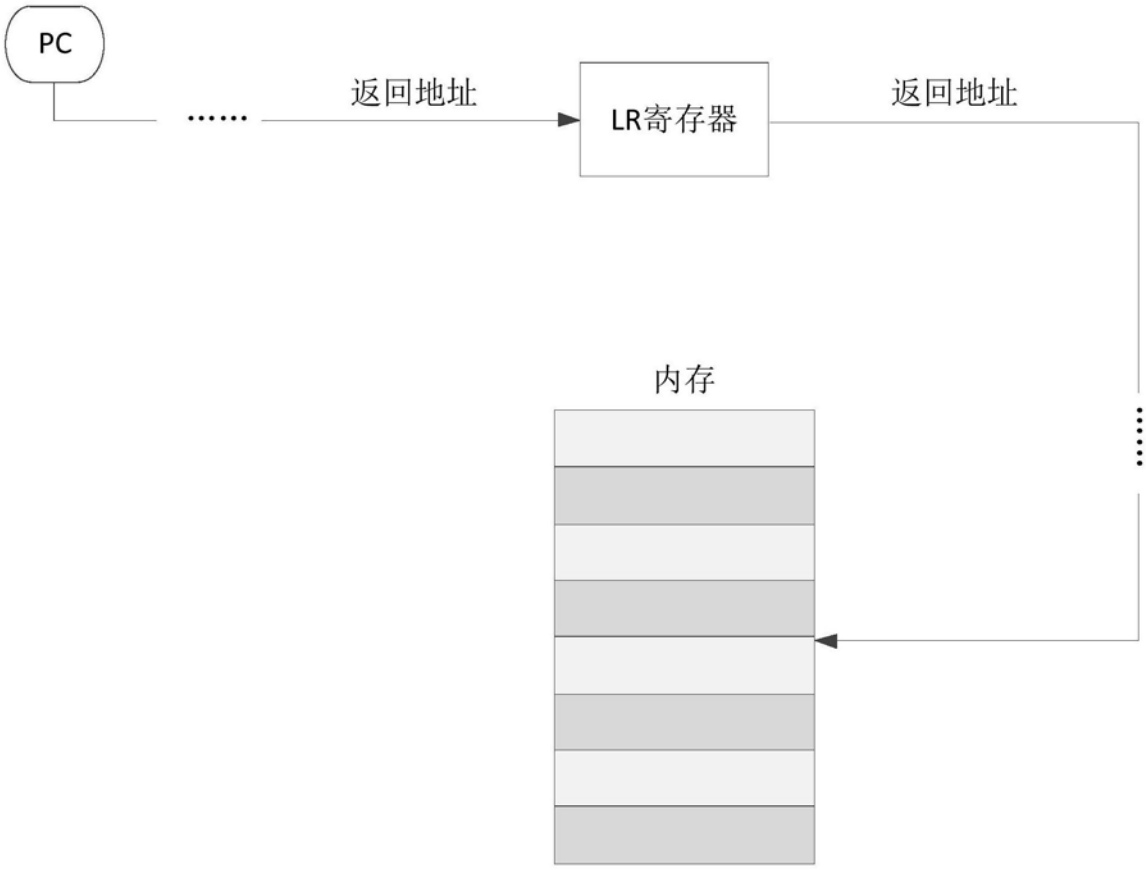


图4A

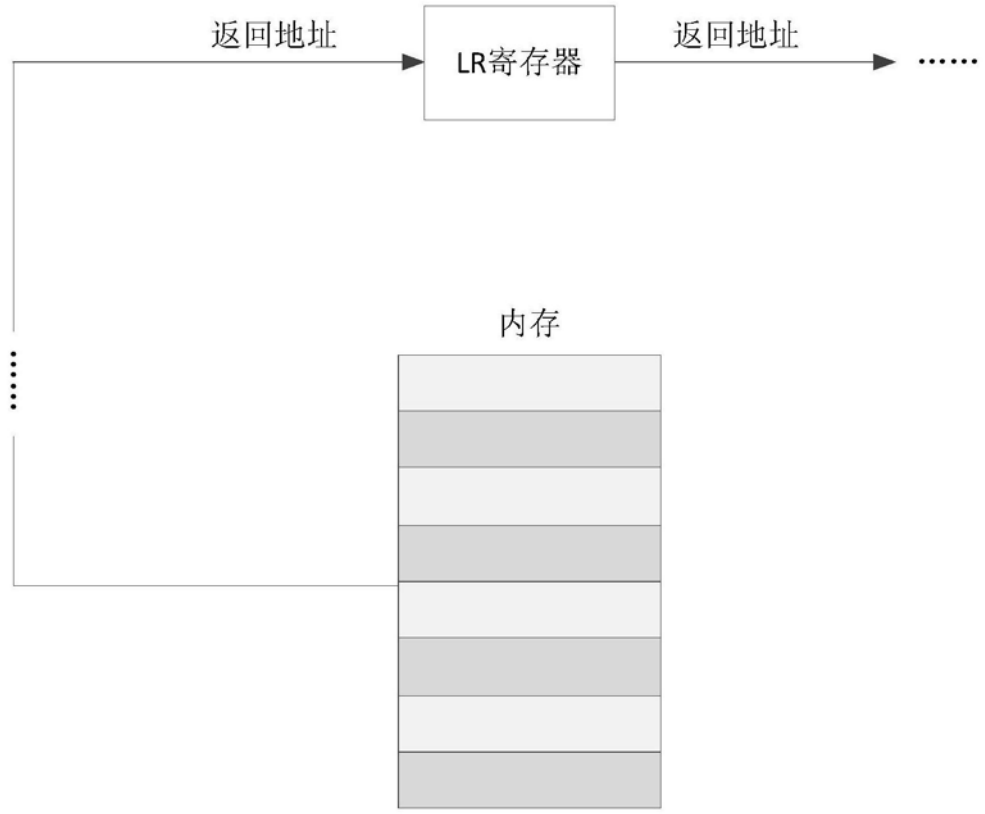


图4B

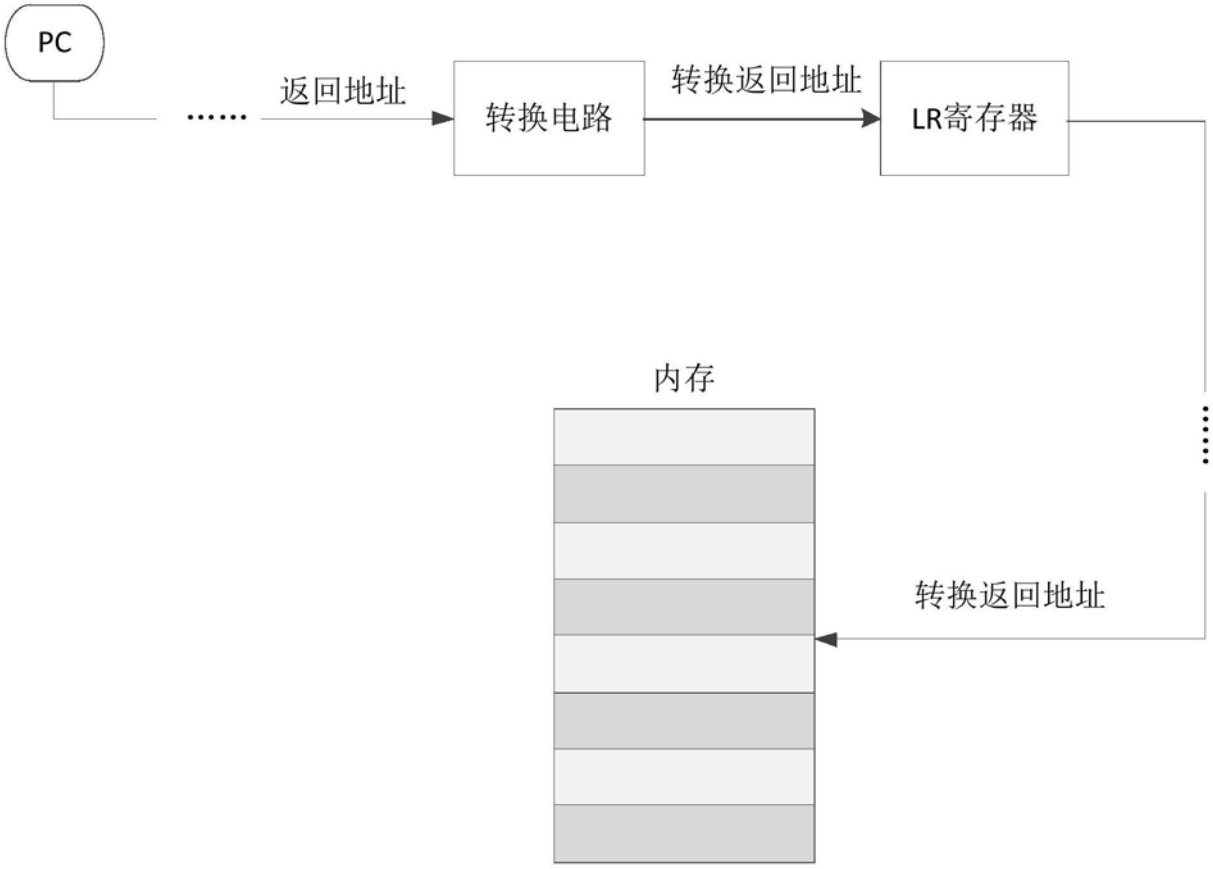


图5A

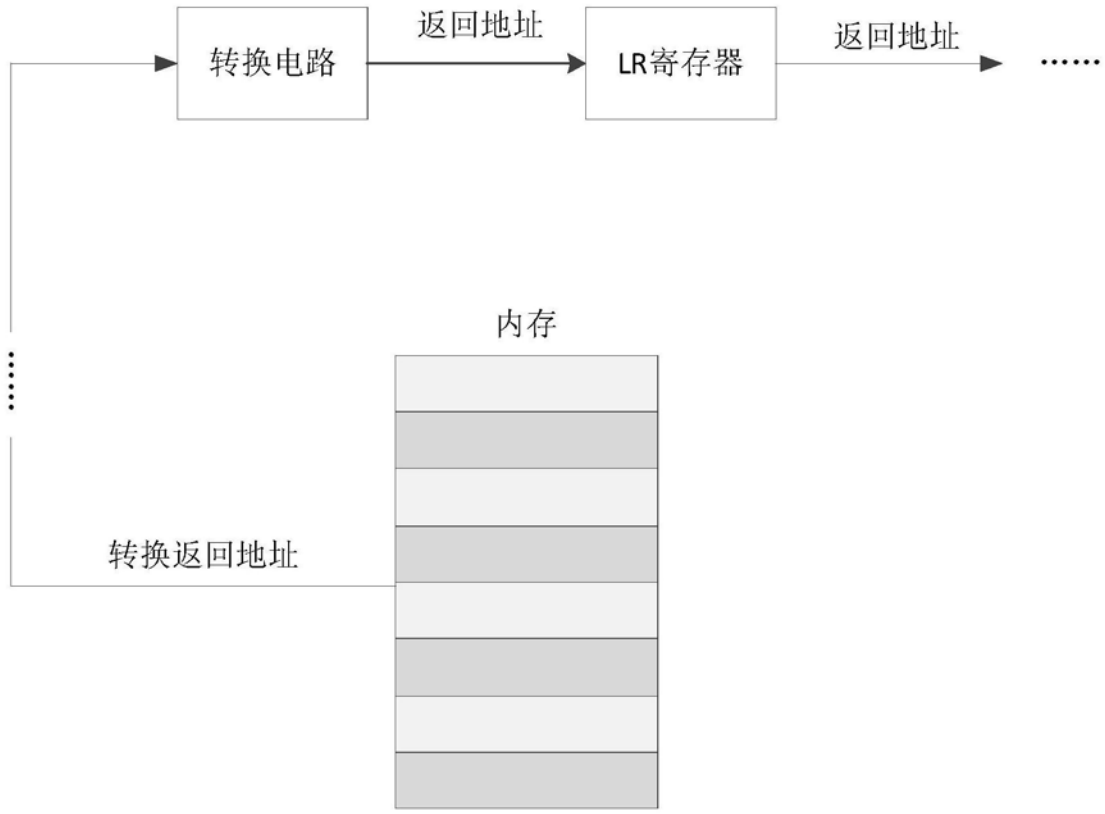


图5B

| ARM指令 | | 流程说明 |
| --- | --- | --- |
| 子程序开始 | SUB sp, sp, #0x40 | |
| | STP x29, x30, [sp, #0x30] | 将返回地址输出至LR（0x30）寄存器。在返回地址进入LR寄存器之前，返回地址先经过转换电路，转换电路对该返回地址进行转换，得到转换返回地址，使得实际进入LR寄存器的是转换返回地址。之后，LR寄存器将转换返回地址输出至内存中的栈 |
| | ADD x29, sp, #0x30<br>…… | |
| 子程序结束 | …… | |
| | LDP x29, x30, [sp, #0x30] | 处理核执行LDP指令时，从内存的栈中读取转换返回地址。该转换返回地址在进入LR（0x30）寄存器之前，会先经过转换电路，转换电路对该转换返回地址再次进行转换，得到原始的返回地址。该原始的返回地址被存入LR寄存器中，并在处理核执行RET指令时使用该返回地址 |
| | ADD sp, sp, #0x40<br>RET | |

图6

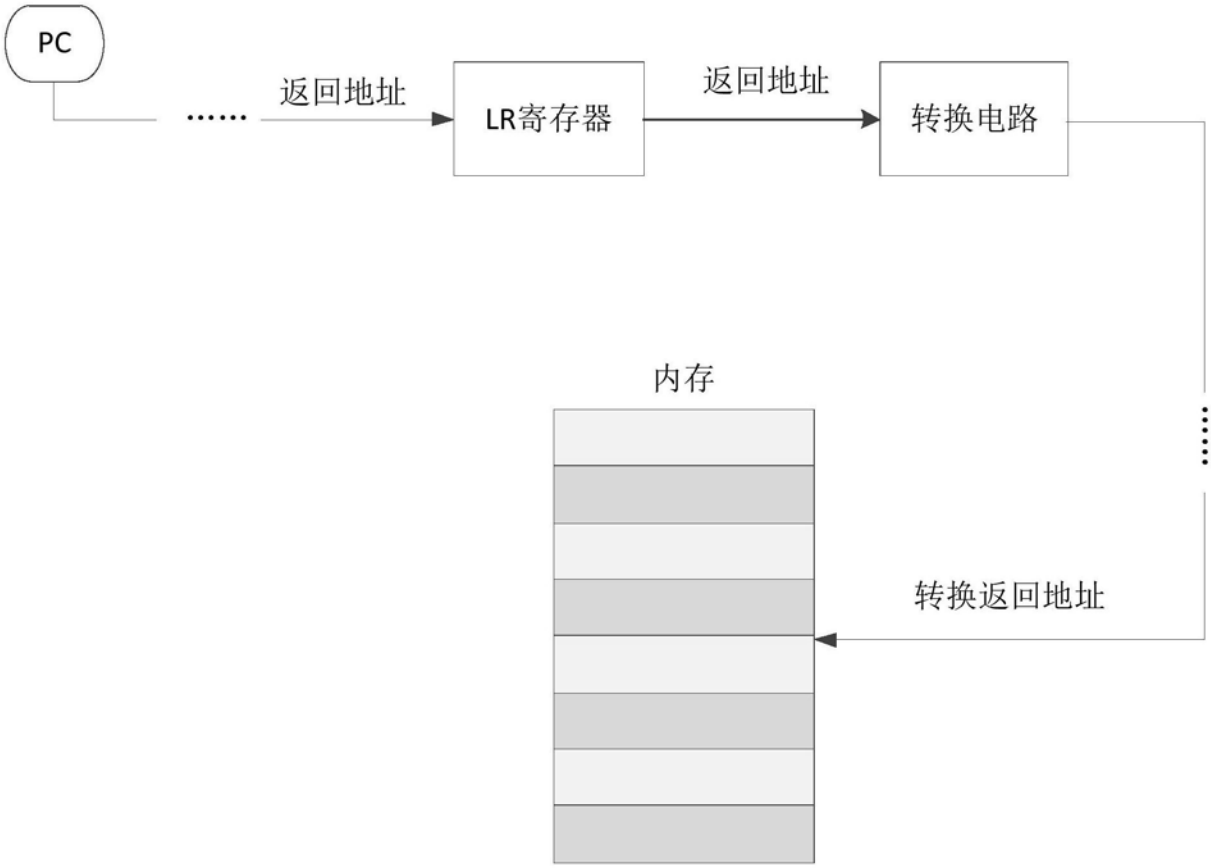


图7A

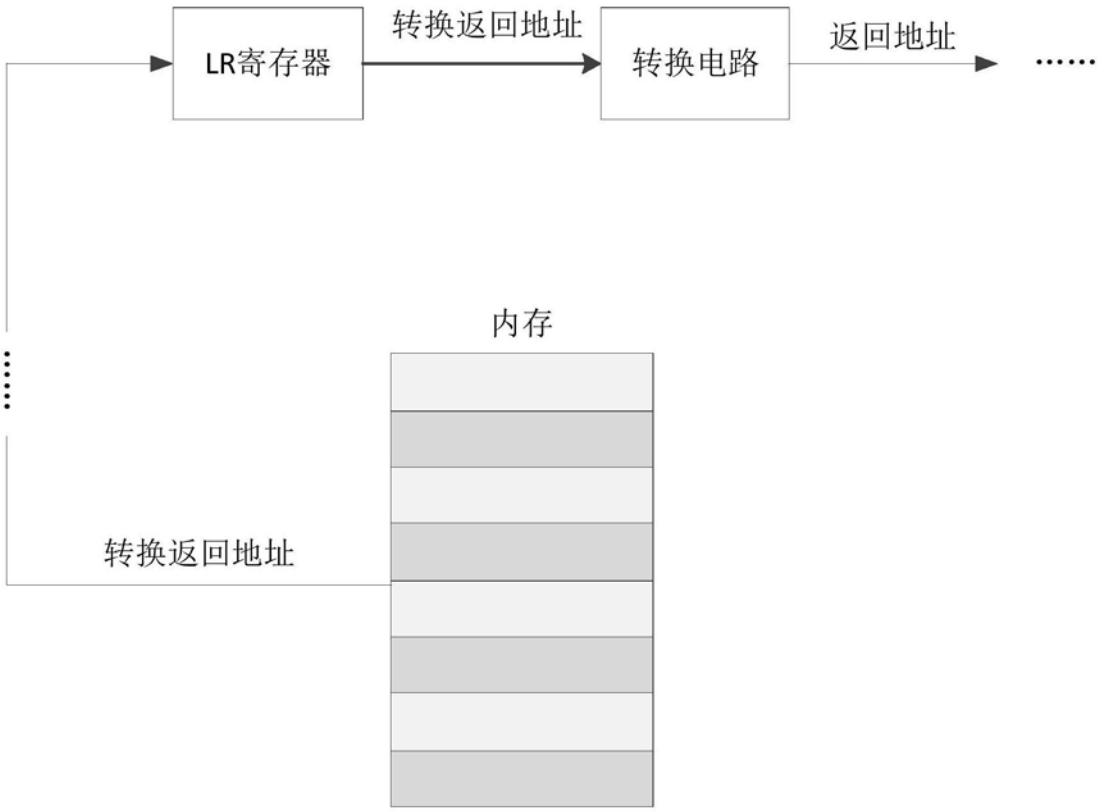
LR寄存器
转换返回地址
转换电路
返回地址
……
内存
转换返回地址

图7B

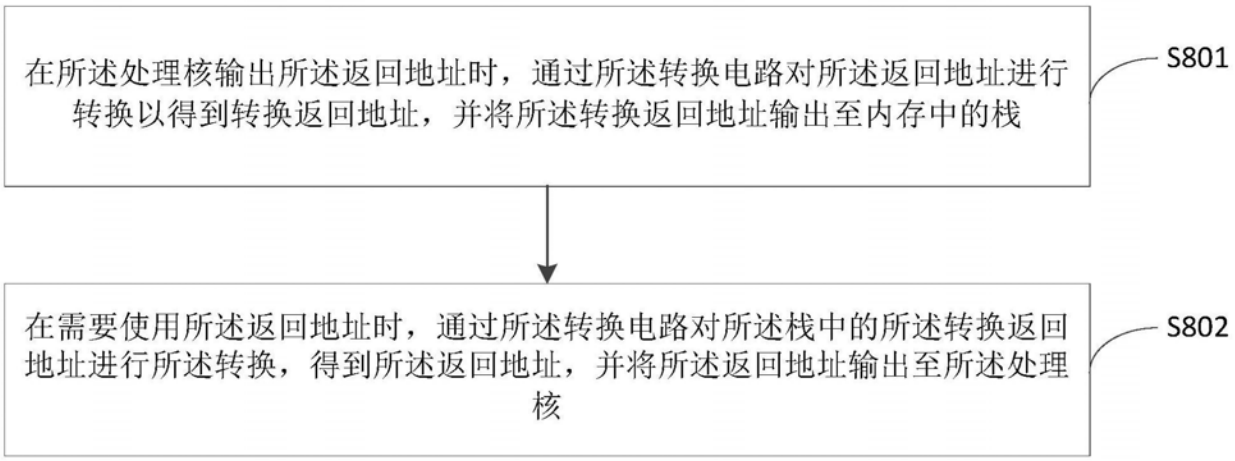
在所述处理核输出所述返回地址时，通过所述转换电路对所述返回地址进行转换以得到转换返回地址，并将所述转换返回地址输出至内存中的栈
S801
在需要使用所述返回地址时，通过所述转换电路对所述栈中的所述转换返回地址进行所述转换，得到所述返回地址，并将所述返回地址输出至所述处理核
S802

图8