(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 115687131 A
(43)申请公布日 2023.02.03

(21)申请号 202211380471.0

(22)申请日 2022.11.04

(71)申请人 无锡先进技术研究院
地址 214122 江苏省无锡市滨湖区绣溪路50号2号楼

(72)发明人 谢汶兵 王俊 仲海梅 刘汉旭 尚博文 李佳梅

(74)专利代理机构 南京纵横知识产权代理有限公司 32224
专利代理师 董建林

(51)Int.Cl.
G06F 11/36(2006.01)

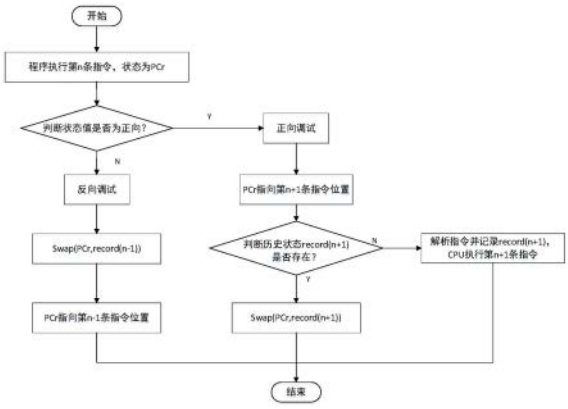
权利要求书1页 说明书5页 附图2页

(54)发明名称

一种程序调试方法

(57)摘要

本发明公开了一种程序调试方法,属于计算机技术领域,包括:运行程序至第n条指令时,记录当前指令n的程序状态为PCr;根据当前进程的执行方向状态值判断调试方向为正向调试或反向调试;若调试方向为反向调试,则将所要跳转目标指令n-1的历史状态record(n-1)与当前指令n的程序状态PCr进行交换;当前指令n的程序状态PCr指向所要跳转目标指令n-1的指令位置;本发明调试过程开启record记录之后,将指令关键信息保存在一个双向链表结构中,该链表始终维持着程序状态调试轨迹,把所要跳转的目标指令历史状态和程序当前状态下的寄存器和内存做内容交换,解决了传统调试器信息记录不精准、架构支持不足的问题。



CN 115687131 A

1.一种程序调试方法,其特征在于,包括:

运行程序至第n条指令时,记录当前指令n的程序状态为PCr;

根据当前进程的执行方向状态值判断调试方向为正向调试或反向调试;

若调试方向为反向调试,则将所要跳转目标指令n-1的历史状态record(n-1)与当前指令n的程序状态PCr进行交换;当前指令n的程序状态PCr指向所要跳转目标指令n-1的指令位置;

若调试方向为正向调试,则将当前指令n的程序状态PCr指向指令n+1的指令位置,并在双向链表中查找指令n+1的历史状态record(n+1)是否存在;

若指令n+1的历史状态record(n+1)已被记录,则将当前指令n的程序状态PCr与指令n+1的历史状态record(n+1)进行交换,执行第n+1条指令;

若指令n+1的历史状态record(n+1)未被记录,则解析指令n+1的程序状态并将其记录至历史状态record(n+1)中,执行第n+1条指令。

2.根据权利要求1所述的一种程序调试方法,其特征在于,所述程序状态包括程序计数器PC、状态控制寄存器、通用寄存器和内存中的数据。

3.根据权利要求1所述的一种程序调试方法,其特征在于,所述历史状态record(n+1)包括执行指令n+1的程序计数器PC和指令n+1所修改的寄存器或内存信息。

4.根据权利要求1所述的一种程序调试方法,其特征在于,所述将所要跳转目标指令n-1的历史状态record(n-1)与当前指令n的程序状态PCr进行交换具体包括:

检查要跳转的目标指令n-1的历史状态record(n-1)结束标记的前驱节点;如果该节点类型是寄存器,则和当前指令n的程序状态PCr交换寄存器的值,否则交换内存中的值;

循环遍历历史状态record(n-1)的每一个节点,直至遇到结束标记结束。

5.根据权利要求1所述的一种程序调试方法,其特征在于,所述当前指令n的程序状态PCr与历史状态record(n+1)进行交换具体包括:

检查指令n+1的历史状态record(n+1)结束标记的前驱节点;

如果该节点类型是寄存器,则和当前指令n的程序状态PCr交换寄存器的值,否则交换内存中的值;

循环遍历历史状态record(n+1)的每一个节点,直至遇到结束标记结束。

6.根据权利要求1所述的一种程序调试方法,其特征在于,所述解析指令n+1的程序状态并将其记录至历史状态record(n+1)中具体包括:

解析指令n+1所对应的程序计数器PC位置;

得到指令类型和目的操作数;

将目的操作数和程序计数器PC保存至双向链表对应的节点中。

7.一种程序调试装置,其特征在于,包括处理器及存储介质;所述存储介质用于存储指令;

所述处理器用于根据所述指令进行操作以执行根据权利要求1~6任一项所述方法的步骤。

8.计算机可读存储介质,其上存储有计算机程序,其特征在于,该程序被处理器执行时实现权利要求1~6任一项所述方法的步骤。

一种程序调试方法

技术领域

[0001] 本发明涉及一种程序调试方法,属于计算机技术领域。

背景技术

[0002] 当前国产化适配过程中,随着应用程序的规模和复杂度上升,其对调试器依赖度越来越高,这就要求开发与处理器相对应的配套高级调试工具来适应适配国产化进程,反向调试和架构紧密相关,工作量庞大且复杂,其包括反汇编、指令识别、状态记录、以及信息恢复等。

[0003] 目前反向调试对于传统调试器极其重要,但是大部分传统调试器均没有支持该功能,反向调试技术不支持共享内存、多线程和子进程,同时反向调试状态记录方法需要大量的内存空间,在执行过程中需要大量的时间负载。

[0004] 公开于该背景技术部分的信息仅仅旨在增加对本发明的总体背景的理解,而不应当被视为承认或以任何形式暗示该信息构成已为本领域普通技术人员所公知的现有技术。

发明内容

[0005] 本发明的目的在于克服现有技术中的不足,提供一种程序调试方法,解决了传统调试器信息记录不精准、架构支持不足的问题。

[0006] 为达到上述目的/为解决上述技术问题,本发明是采用下述技术方案实现的:

一种程序调试方法,包括:运行程序至第n条指令时,记录当前指令n的程序状态为PCr;

根据当前进程的执行方向状态值判断调试方向为正向调试或反向调试;

若调试方向为反向调试,则将所要跳转目标指令n-1的历史状态record(n-1)与当前指令n的程序状态PCr进行交换;当前指令n的程序状态PCr指向所要跳转目标指令n-1的指令位置;

若调试方向为正向调试,则将当前指令n的程序状态PCr指向指令n+1的指令位置,并在双向链表中查找指令n+1的历史状态record(n+1)是否存在;

若指令n+1的历史状态record(n+1)已被记录,则将当前指令n的程序状态PCr与指令n+1的历史状态record(n+1)进行交换,执行第n+1条指令;

若指令n+1的历史状态record(n+1)未被记录,则解析指令n+1的程序状态并将其记录至历史状态record(n+1)中,执行第n+1条指令。

[0007] 进一步地,所述程序状态包括程序计数器PC、状态控制寄存器、通用寄存器和内存中的数据。

[0008] 进一步地,所述历史状态record(n+1)包括执行指令n+1的程序计数器PC和指令n+1所修改的寄存器或内存信息。

[0009] 进一步地,所述将所要跳转目标指令n-1的历史状态record(n-1)与当前指令n的程序状态PCr进行交换具体包括:

检查要跳转的目标指令n-1的历史状态record(n-1)结束标记的前驱节点;如果该节点类型是寄存器,则和当前指令n的程序状态PCr交换寄存器的值,否则交换内存中的值;

循环遍历历史状态record(n-1)的每一个节点,直至遇到结束标记结束。

[0010] 进一步地,所述当前指令n的程序状态PCr与历史状态record(n+1)进行交换具体包括:

检查指令n+1的历史状态record(n+1)结束标记的前驱节点;

如果该节点类型是寄存器,则和当前指令n的程序状态PCr交换寄存器的值,否则交换内存中的值;

循环遍历历史状态record(n+1)的每一个节点,直至遇到结束标记结束。

[0011] 进一步地,所述解析指令n+1的程序状态并将其记录至历史状态record(n+1)中具体包括:

解析指令n+1所对应的程序计数器PC位置;

得到指令类型和目的操作数;

将目的操作数和程序计数器PC保存至双向链表对应的节点中。

[0012] 本发明的目的之二在于提供一种程序调试装置,包括处理器及存储介质;

所述存储介质用于存储指令;所述处理器用于根据所述指令进行操作以执行上述的方法的步骤。

[0013] 本发明的目的之三在于提供一种计算机可读存储介质,其上存储有计算机程序,该程序被处理器执行时实现上述方法的步骤。

[0014] 与现有技术相比,本发明所达到的有益效果:

本发明调试过程开启record记录之后,将指令关键信息保存在一个双向链表结构中,利用结束标记区分每一条指令的历史执行状态,该链表始终维持着程序状态调试轨迹,通过把所要跳转的目标指令历史状态和程序当前状态下的寄存器和内存做内容交换,实现正向和反向调试,解决了传统调试器信息记录不精准、架构支持不足的问题。

附图说明

[0015] 图1是本发明实施例提供的一种程序调试方法的流程图;

图2是本发明实施例提供的一种程序调试方法的双向链表结构的示意图;

图3是发明实施例提供的一种程序调试方法的双向链表的结构体示意图;

图4是发明实施例提供的一种程序调试方法的record结构体示意图。

具体实施方式

[0016] 下面结合附图对本发明作进一步描述。以下实施例仅用于更加清楚地说明本发明的技术方案,而不能以此来限制本发明的保护范围。

[0017] 实施例一

如图1所示,一种程序调试方法,包括:

运行程序至第n条指令时,记录当前指令n的程序状态为PCr,程序状态包括程序计数器PC、状态控制寄存器、通用寄存器和内存中的数据;

根据当前进程的执行方向状态值判断调试方向为正向调试(exec_forward)或反

向调试(exec_reverse)；

若调试方向为反向调试，则将所要跳转目标指令n-1的历史状态record(n-1)与当前指令n的程序状态PCr进行交换；当前指令n的程序状态PCr指向所要跳转目标指令n-1的指令位置，具体的：

检查要跳转的目标指令n-1的历史状态record(n-1)结束标记的前驱节点；如果该节点类型是寄存器，则和当前指令n的程序状态PCr交换寄存器的值，否则交换内存中的值；循环遍历历史状态record(n-1)的每一个节点，直至遇到结束标记结束；

若调试方向为正向调试，则将当前指令n的程序状态PCr指向指令n+1的指令位置，并在双向链表中查找指令n+1的历史状态record(n+1)是否存在；历史状态record(n+1)包括执行指令n+1的程序计数器PC和被指令n+1所修改的寄存器或内存信息；

如图4所示，record结构体记录每条指令的历史状态信息，每一个历史状态对应一条已执行的历史指令；程序每一个历史状态中的内容为寄存器、内存和结束标记等三类节点，由于程序计数器PC在动态变化，因此每个历史状态都包含程序计数器PC寄存器节点；结束标记表示一条指令记录内容的结束，实现对相邻的指令历史状态的区分；

其中，record(n-1)为其前一条指令反向执行状态双向链表存储情况，即record(n-1)=record(n)->prev；record(n+1)为其下一条待执行指令正向执行状态双向链表存储情况，即record(n+1) =record(n) ->next；

若指令n+1的历史状态record(n+1)未被记录，则解析指令n+1的程序状态并将其记录至历史状态record(n+1)中，执行第n+1条指令；具体的：

解析指令n+1所对应的程序计数器PC位置，得到指令类型和目的操作数；为了得到程序的历史调试状态，需基于国产平台反汇编其指令流得到汇编指令，根据指令操作码不同分类为五大类指令格式：系统调用类、转移控制类、访存类、简单运算类、复合运算类；针对指令译码后得到目的操作数以及类型，进而决定该指令目的操作数修改的是寄存器还是内存，例如：指令(ldl Ra,disp(Rb))，其指令格式为访存类，目的操作数是Ra；

打开记录record功能后，提取申威指令32位中的前6位得到操作码；

根据操作码解析指令类型，判断目的操作数在指令中的偏移位置；

将目的操作数和程序计数器PC保存至双向链表对应的节点中；

根据申威处理器指令特点，对于涉及浮点运算的指令还需要维护程序状态寄存器FPCR，包含浮点异常的状态、浮点舍入模式、浮点异常自陷控制以及特殊数据的控制等从而支持浮点操作的正确调试；

若指令n+1的历史状态record(n+1)已被记录，则将当前指令n的程序状态PCr与指令n+1的历史状态record(n+1)进行交换，执行第n+1条指令；具体的：检查指令n+1的历史状态record(n+1)结束标记的前驱节点；如果该节点类型是寄存器，则和当前指令n的程序状态PCr交换寄存器的值，否则交换内存中的值；循环遍历历史状态record(n+1)的每一个节点，直至遇到结束标记结束；

程序反向运行至第n条指令位置，反向调试支持单步反向、多步反向执行等；对于多步反向执行操作，支持指令计数和插入断点两种方法，在源码级别逆向调试时，首先得到源码行对应的指令总条数，在该总条数内循环回退每一条指令即可；在记录record功能下，可以在任何指令位置设置标签，回退时将会计算标签到程序当前状态的指令总条数，最后

回退所记录指令总条数即可到达标签位置;支持执行rc调试,可以插入断点实现逆向到达指定位置的目的;支持反向跳出函数体,通过在函数首条指令位置插入断点。

[0018] 如图2所示,由于每条指令目的操作数可能会修改内存,也可能会修改寄存器,也有可能不需要对操作数做任何修改,因此使用固定数组来存储历史状态势必会造成内存浪费,循环分配双向链表节点,只对执行过程中被修改的寄存器和内存记录信息并分配双向节点,极大地节省了内存,也保证了反向调试的效率;

如图3所示,定义双向链表节点结构体instruction_node,每个节点的记录内容可能是寄存器、内存和结束标记之一;

其中,reg表示寄存器,内容为寄存器编号number和寄存器值reg_buffer;

mem为内存,内容为地址addr和对应值mem_buffer;

end是链表结束标记,内容为信号signal和已记录指令的数量insn_num;

prev和next是链表节点的前驱和后继指针。

[0019] 实施例二

一种程序调试装置,包括处理器及存储介质;

存储介质用于存储指令;处理器用于根据指令进行操作以执行以下的方法的步骤:

运行程序至第n条指令时,记录当前指令n的程序状态为PCr;

根据当前进程的执行方向状态值判断调试方向为正向调试(exec_forward)或反向调试(exec_reverse);

若调试方向为反向调试,则将所要跳转目标指令n-1的历史状态record(n-1)与当前指令n的程序状态PCr进行交换;当前指令n的程序状态PCr指向所要跳转目标指令n-1的指令位置;

若调试方向为正向调试,则将当前指令n的程序状态PCr指向指令n+1的指令位置,并在双向链表中查找指令n+1的历史状态record(n+1)是否存在;

若指令n+1的历史状态record(n+1)已被记录,则将当前指令n的程序状态PCr与指令n+1的历史状态record(n+1)进行交换,执行第n+1条指令;

若指令n+1的历史状态record(n+1)未被记录,则解析指令n+1的程序状态并将其记录至历史状态record(n+1)中,执行第n+1条指令。

[0020] 实施例三

一种计算机可读存储介质,其上存储有计算机程序,该程序被处理器执行时实现以下方法的步骤:

运行程序至第n条指令时,记录当前指令n的程序状态为PCr;

根据当前进程的执行方向状态值判断调试方向为正向调试(exec_forward)或反向调试(exec_reverse);

若调试方向为反向调试,则将所要跳转目标指令n-1的历史状态record(n-1)与当前指令n的程序状态PCr进行交换;当前指令n的程序状态PCr指向所要跳转目标指令n-1的指令位置;

若调试方向为正向调试,则将当前指令n的程序状态PCr指向指令n+1的指令位置,并在双向链表中查找指令n+1的历史状态record(n+1)是否存在;

若指令n+1的历史状态record(n+1)已被记录，则将当前指令n的程序状态PCr与指令n+1的历史状态record(n+1)进行交换，执行第n+1条指令；

若指令n+1的历史状态record(n+1)未被记录，则解析指令n+1的程序状态并将其记录至历史状态record(n+1)中，执行第n+1条指令。

[0021] 以上所述仅是本发明的优选实施方式，应当指出，对于本技术领域的普通技术人员来说，在不脱离本发明技术原理的前提下，还可以做出若干改进和变形，这些改进和变形也应视为本发明的保护范围。

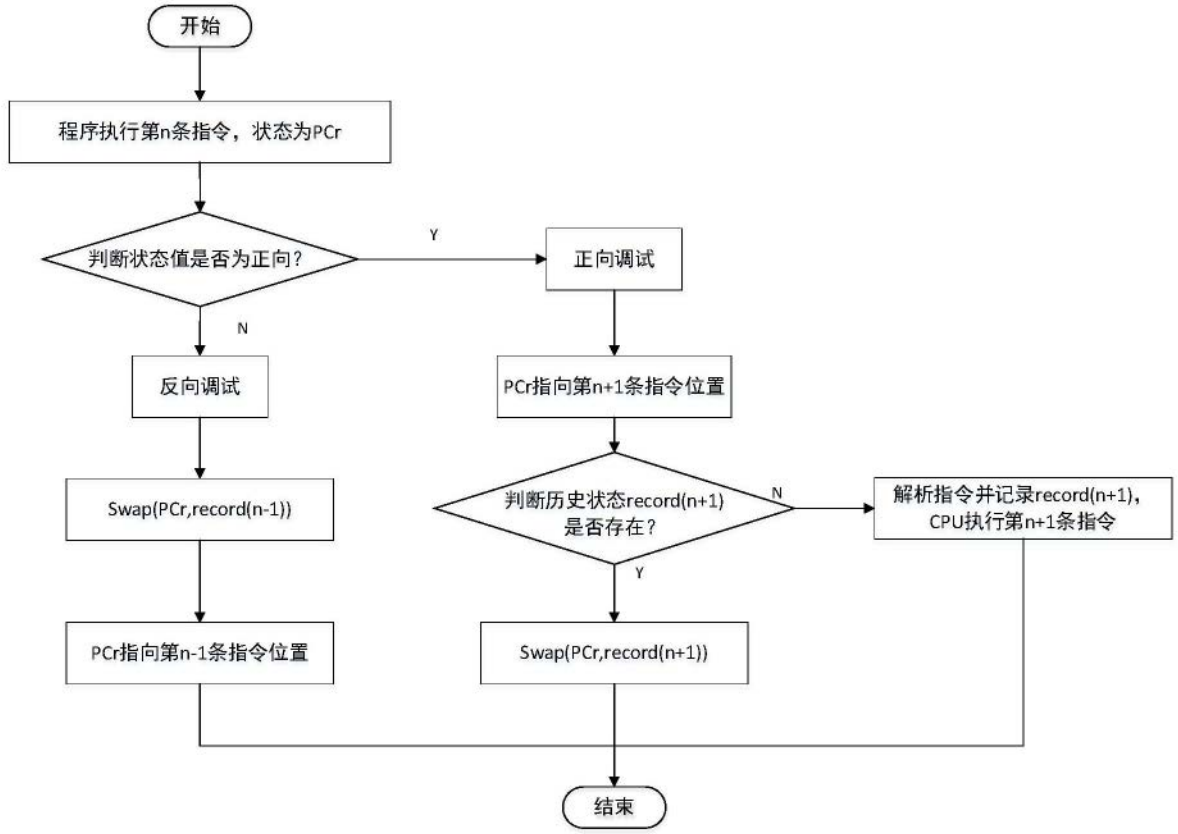

开始
程序执行第n条指令，状态为PCr
判断状态值是否为正向？
Y
正向调试
N
反向调试
PCr指向第n+1条指令位置
Swap(PCr,record(n-1))
判断历史状态record(n+1)
是否存在？
N
解析指令并记录record(n+1)，
CPU执行第n+1条指令
Y
PCr指向第n-1条指令位置
Swap(PCr,record(n+1))
结束

图1

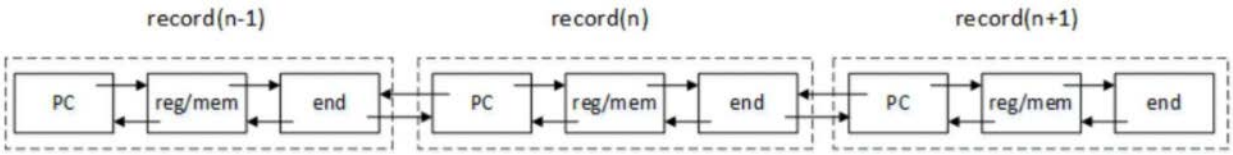

record(n-1)
record(n)
record(n+1)
PC
reg/mem
end
PC
reg/mem
end
PC
reg/mem
end

图2

```
1.  struct instruction_node {
2.      union{
3.          struct reg{
4.              number;
5.              reg_Buffer;};
6.          struct mem{
7.              addr;
8.              mem_buffer;};
9.          struct end{
10.             signal;
11.             insn_num;};
12.     }
13.   instruction_node *prev;
14.   instruction_node *next;
15. }node;
```

图3

```
1.  struct record{
2.  struct instruction_node reg;
3.  struct instruction_node mem;
4.  struct instruction_node PC;
5.  } record;
```

图4