



(12) 发明专利申请

(10) 申请公布号 CN 102411534 A

(43) 申请公布日 2012. 04. 11

(21) 申请号 201110209571. 2

(22) 申请日 2011. 07. 25

(71) 申请人 中国科学院声学研究所

地址 100190 北京市海淀区北四环西路 21 号

(72) 发明人 彭楚 王东辉 朱浩 洪纓 侯朝焕

(74) 专利代理机构 北京亿腾知识产权代理事务 所 11309

代理人 陈霁

(51) Int. Cl.

G06F 11/36 (2006. 01)

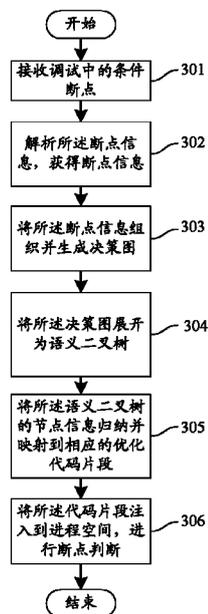
权利要求书 1 页 说明书 5 页 附图 3 页

(54) 发明名称

一种断点调试方法和调试器

(57) 摘要

本发明涉及一种断点调试方法和调试器。所述调试方法包括以下步骤:接收调试中的条件断点;解析所述条件断点,获得断点信息;将所述断点信息组织成决策图;分析所述决策图选择生成有序二叉决策图或展开为语义二叉树;将所述有序二叉决策图的节点展开形成语义二叉树;将所述语义二叉树的节点信息映射到相应的优化代码片段上;将所述代码片段注入到进程空间,进行断点信息判断。所述调试器包括如下功能单元:接收单元、解析单元、生成单元、展开单元、映射单元和注入单元。根据本发明的调试方法和调试器,可以大幅提升调试器的执行效率。



1. 一种断点调试方法,其特征在于,所述方法包括:
接收调试中的条件断点;
解析所述条件断点,获得断点信息;
将所述断点信息组织并生成决策图;
将所述决策图展开为语义二叉树;
将所述语义二叉树的节点信息归纳并映射到相应的优化代码片段;
将所述代码片段注入到进程空间,进行断点信息判断。
2. 根据权利要求1所述的方法,其特征在于:将所述决策图展开为语义二叉树步骤替代为:分析所述决策图并生成有序二叉决策图,将所述有序二叉决策图展开为语义二叉树。
3. 根据权利要求1或2所述的方法,其特征在于:所述将所述断点信息组织生成决策图步骤包括将所述断点信息由布尔函数来表示,根据布尔函数性质来生成决策图。
4. 根据权利要求2所述的方法,其特征在于:采用有序二叉决策图简化算法将所述决策图生成有序二叉决策图。
5. 根据权利要求1或2所述的方法,其特征在于:所述将所述语义二叉树的节点信息归纳并映射到相应的优化代码片段的步骤是根据所述语义二叉树的节点信息的特征码构成的模式将其映射到相应的优化代码片段。
6. 根据权利要求1或2所述的方法,其特征在于:所述将所述代码片段注入到进程空间包括:
预先加载一个有效的静态库,通过在所述静态库的原始C文件中嵌入空操作语句实现进程空间的划分;
将所述代码片段覆盖所述空操作语句。
7. 一种调试器,其特征在于,所述调试器包括:
接收单元,用于接收调试中的条件断点;
解析单元,用于解析所述条件断点,获得断点信息;
生成单元,用于将所述断点信息组织生成决策图;
展开单元,用于将所述决策图展开为语义二叉树,或者用于分析所述决策图并生成有序二叉决策图,将所述有序二叉决策图展开成语义二叉树;
映射单元,用于将所述语义二叉树的节点信息归纳并映射到相应的优化代码片段;
注入单元,用于将所述代码片段注入到进程空间。
8. 根据权利要求7所述的调试器,其特征在于:所述生成单元将所述断点信息由布尔函数来表示,根据布尔函数性质来生成决策图。
9. 根据权利要求7所述的调试器,其特征在于:所述展开单元采用有序二叉决策图简化算法将所述决策图生成有序二叉决策图。
10. 根据权利要求7所述的调试器,其特征在于:所述映射单元根据所述语义二叉树的节点信息的特征码构成的模式将其映射到相应的优化代码片段。

一种断点调试方法和调试器

技术领域

[0001] 本发明涉及计算机软件设计,尤其涉及一种断点调试方法和调试器。

背景技术

[0002] 调试器是处理器体系结构设计与软硬件协同验证中不可或缺的一部分。一方面,它在软件层次上对目标处理器的体系结构进行建模来模拟其执行行为,另一方面,通过提供调试接口,接收用户输入的各种控制信息,实现对程序执行过程中的数据流、控制流的监测及修改。

[0003] 调试器最基础、最核心的功能之一是断点功能,它可以让程序中断在需要的地方,从而方便其分析。断点可以分为行断点和条件断点两种,其中行断点是一种特殊的条件断点,而条件断点通常是以逻辑表达式的形式呈现。

[0004] 现有技术的调试器一般采用表达式计算来实现条件断点的解析,而表达式计算一般采用如下两种方案进行处理:

[0005] 第一种方案:其于栈结构的表达式计算,以后缀表达式 $a+b*c > (d*e+f)*g$ 为例,首先将后缀表达式利用栈操作生成中缀表达式 $abc*+de*f+g* >$,按 $a, b, c*, +, d, e, *, f, +, g, *, >$ 的顺序依次压栈,在压栈过程中,遇到有效操作符的时候,按照操作符所需要操作数的个数,从栈中弹出相应个数,计算后将结果再次压入栈中。例如,当压入操作符 $*$ 的时候,需要将操作数 c 和 b 依次弹出栈,并计算 $b*c$ 后将结果再压入到栈中。具体过程如图 1 所示。

[0006] 第二种方案:基于语义二叉树结构的表达式计算,按字符的优先级组织成二叉树结构,计算时应用深度优先搜索算法从二叉树的底层从左至右,从下至上依次进行计算。以第一种方案中的后缀表达式为例,处理的二叉树结构如图 2 所示。

[0007] 现有技术上述两种方案存在的缺点是只能处理有限的固定模式,应用面窄,当通过调试接口输入大量断点信息时,繁琐的表达式计算会严重影响调试器的执行效率,严重影响调试器性能。

发明内容

[0008] 本发明的目的是提供一种处理模式灵活的断点调试方法和调试器,来解决条件断点中繁琐的表达式计算问题,从而大大提高了调试器的执行效率。

[0009] 为实现上述目的,本发明提供了一种断点调试方法,其特征在于包括以下步骤:

[0010] 接收调试中的条件断点;解析所述条件断点,获得断点信息;将所述断点信息中条件断点生成决策图;选择将所述决策图分析生成有序二叉决策图或展开为语义二叉树;将所述有序二叉决策图展开为语义二叉树,并将节点信息映射到相应的优化代码片段;将所述代码片段注入到进程空间,进行断点信息判断。

[0011] 本发明还提供了一种调试器,其特征在于包括:

[0012] 接收单元,用于接收调试中的条件断点;解析单元,用于解析条件断点,获得断点

信息；生成单元，用于将断点信息组织生成决策图；展开单元，用于将决策图展开为语义二叉树，或者用于分析所述决策图并生成有序二叉决策图，将有序二叉决策图展开成语义二叉树；映射单元，用于将语义二叉树的节点信息归纳并映射到相应的优化代码片段；注入单元，用于将代码片段注入到进程空间。

[0013] 本发明实施例的断点调试方法和调试器，能够解决了在调试器处理繁琐的条件断点表达式时所遇到的执行效率低下问题，从而大幅提升调试器的执行效率。

附图说明

- [0014] 图 1 为现有技术的基于栈结构的表达式计算算法；
- [0015] 图 2 为现有技术的基于语义二叉树结构的表达式计算算法；
- [0016] 图 3 为本发明一实施例的断点调试方法流程图；
- [0017] 图 4 为本发明一实施例的调试器示意结构图；
- [0018] 图 5 为本发明一实施例的决策图；
- [0019] 图 6 为根据图 5 简化后的决策图；
- [0020] 图 7 为根据图 6 简化后最终的有序二叉决策图；
- [0021] 图 8 为根据图 7 形成的判断流程图；
- [0022] 图 9 为本发明一实施例的语义二叉树；
- [0023] 图 10 示意性示出了一种断点调试方法的处理系统。

具体实施方式

[0024] 下面通过附图和实施例，对本发明的技术方案做进一步的详细描述。

[0025] 图 3 为本发明一实施例的断点调试方法流程图。

[0026] 在步骤 301，接收调试中的条件断点。

[0027] 调试器在调试中接收来自用户设置的条件断点，包括用户设置的全局条件断点。

[0028] 在步骤 302，解析所述条件断点，获得其中的断点信息。

[0029] 调试器对在调试过程中接收到条件断点进行解析，获得条件断点中的断点信息包括：操作符、变量对应的进程地址和立即数。

[0030] 在步骤 303，将所述断点信息组织并生成决策图。

[0031] 在一个实施例中，现以条件断点表达式 $(\text{address_regfile}[0] \geq 10 \mid \mid \text{data_regfile}[1] \leq 13) \&\& \text{insncounter} \geq 10$ 为例来描述有序二叉决策图的生成过程。上述条件断点的断点信息可以由布尔函数来表示，并根据布尔函数性质来生成决策图。

[0032] 首先，将条件断点中的 $\text{address_regfile}[0] \geq 10$ 用 X_1 来替代， $\text{data_regfile}[1] \leq 13$ 用 X_2 来替代， $\text{insncounter} \geq 10$ 用 X_3 来替代，条件断点表达式等效为 $(X_1 \mid \mid X_2) \&\& X_3$ ，而布尔代数是而非空集合 B (B 中至少含有两个不同元素)，以及 B 上的二元运算“ \cdot ”、“ $+$ ”，一元运算“ $'$ ”组成的多元组，因此可以将条件断点中的“ $\&\&$ ”运算符映射为布尔代数运算符“ \cdot ”，将条件断点“ $\mid \mid$ ”运算符映射为“ $+$ ”运算符，因此该条件断点对应的在变量序 $\pi : x_1 < x_2 < x_3$ 下的布尔函数为： $f(x_1, x_2, x_3) = (x_1 + x_2) \cdot x_3$ ，它对应的布尔函数族 $\#f(x_1, x_2, x_3) = \{(x_1 + x_2) \cdot x_3, (x_2 \cdot x_3), x_3, 0, 1\}$ 。

[0033] 根据上述的布尔函数族，生成如图 5 所示的决策图，其中节点的 0-分支用虚线连

接,节点的 1-分支用实线连接,图 5 中描述出布尔函数族在 x_1, x_2, x_3 不同取值情况下各路径对应的结果。图中的 X3 节点存在着子节点完全相同的情况,也就是说明其对应相同的布尔函数,因此可以将这些冗余的节点进行合并形成如图 6 所示的决策图。

[0034] 在如图 6 所示的决策图中,在 X1 的 1-分支上无论 X2 结点取何值都不会影响该条路径的最终值,因此,X1 的 1-分支上的 X2 结点可被删除,同理在 X1 的 0-分支上的结点 X3 也被删除,最终形成如图 7 所示的最简有序二叉决策图。

[0035] 针对图 7 所示的最简有序二叉决策图可采用递归算法进行进一步的分析,例如,当图 7 中的 X3 节点取值为 0 时,该有序二叉决策图的布尔函数值始终为 0,则条件断点不成立;当 X3 节点取值为 1,且 X1 或 X2 取值为 1 时,该有序二叉决策图的布尔函数值为 1,则条件断点成立。有序二叉决策图的分析结果描述成如图 8 的模式。

[0036] 在步骤 304,将所述决策图展开为语义二叉树。

[0037] 在一个实施例中,可以通过分析决策图生成有序二叉决策图,再将有序二叉决策图展开为语义二叉树。上述生成有序二叉决策图的方法可以采用 Bryant 提出的有序二叉决策图简化算法。

[0038] 在步骤 305,将所述语义二叉树的节点信息归纳并映射到相应的优化代码片段。

[0039] 调试器根据最简有序二叉决策图展开为语义二叉树,并将语义二叉树树上的各类操作符节点信息的特征码构成的模式映射到不同的优化代码片段上。

[0040] 在一个实施例中,描述如何将语义二叉树的节点信息映射到相应的优化代码片段上。首先,将条件断点表达式 ($\text{address_regfile}[0] \geq 10 \mid \mid \text{data_regfile}[1] \leq 13$)&&\text{inscounter} \geq 10) 生成如图 9 所示的语义二叉树,图 9 中虚线指引的各个操作符对应于图 7 中的各个节点。然后通过采用深度优先算法遍历该语义二叉树,以 X1 节点为例,其左孩子的属性为 $\text{addr}32$,代表该操作数是 32 位,需要从相应的地址获取;而右孩子为 $\text{u_imm}32$,代表该操作数为 32 位的无符号立即数,而 X1 节点代表 ge ,表示“ \geq ”操作,因此 X1 节点的模式集合为: $\{\text{addr}32, \text{ge}, \text{u_imm}32\}$,通过查询代码片段表并获取相应的指令片段信息。

[0041] 而诸如跳转等需要目的地址的指令,在遍历语义二叉树生成代码片段的过程中只需要填入相应的指令保留位,而目的地址则根据目的地址所需要的字节数以 0 值的形式填入指令槽中,在后续工作中完成地址的分配。

[0042] 在如图 8 中,当 X3 的取值不为 1 时,条件断点判断失败,程序流需要跳转到下一个断点起始处,而当 X3 为 1 时,同时 X1 也为 1 则同样跳转到下一个全局断点的起始处,否则继续判断 X2。因此,X3 中 jump 指令的目的地址是下一断点的起始处,而 X1 中 jump 指令的目的地址则是该断点的 return 语句的起始处。

[0043] 在步骤 306 中,将所述代码片段注入到进程空间,进行断点信息判断。

[0044] 调试器将获取的指令片段信息注入到进程空间,为实现代码片段的注入,调试器需要开辟一段独立有效,并占据一定字节数的进程空间。在本发明实施例中,通过调试器预先加载一个有效的静态库实现,通过在静态库的原始 C 文件中嵌入大量的空操作 (nop) 语句实现进程空间的占用。在调试器执行时,只需将生成好的代码段覆盖原始 C 文件中的 nop 语句处即可实现。

[0045] 静态库如下所示:

[0046]

```
Int breakpoint_test(void)
{
    _asm{
        nop    /*空操作语句*/
        nop
        nop
        ...
        nop
    }
    return -1;
}
```

[0047] 在代码片段注入到进程空间后,调试器通过调用 breakpoint_test 函数即可以实现对断点的判断并执行。

[0048] 图 4 为本发明一实施例的调试器示意结构图。如图所示:40 表示调试器,41 表示接收单元,42 表示解析单元,43 表示生成单元,44 表示展开单元,45 表示映射单元,以及 46 表示注入单元。

[0049] 接收单元 41 用于接收调试中的条件断点信息;解析单元 42 用于解析条件断点信息,获得断点信息;生成单元 43 用于将条件断点中的断点信息组织生成决策图;展开单元 44 用于用于将决策图展开为语义二叉树,或者用于分析决策图并生成有序二叉决策图,将有序二叉决策图展开成语义二叉树。映射单元 45 用于用于将语义二叉树的节点信息归纳并映射到相应的优化代码片段;注入单元 46 用于将代码片段注入到进程空间。

[0050] 在本实施例中,首先由调试器 40 的接收单元 41 接收来自用户设置的断点信息,包括用户设置的全局断点信息。接收单元 41 在调试中可以接收来自用户设置的大规模全局条件断点信息,解析单元 42 对接收单元 41 在调试中接收的条件断点信息进行解析,解析出条件断点信息中的断点信息,再由生成单元 43 根据解析单元 42 解析出的条件断点信息生成决策图,优选地,生成单元 43 可以将断点信息中的条件断点采用布尔函数来表示,根据布尔函数来生成决策图,再由展开单元 44 对上述决策图展开为语义二叉树,或者用于分析上述决策图并生成有序二叉决策图,将有序二叉决策图展开成语义二叉树,最后由映射单元 45 根据语义二叉树的节点信息的特征码构成的模式将其映射到相应的优化代码片段。调试器 40 通过注入单元 46 将上述优化代码片段注入到进程空间,判断并执行。

[0051] 图 10 示意性示出了一种断点调试方法的处理系统。图 3 中所示的断点调试方法可以在该系统中实现。图 10 中所示的处理系统包括 CPU(中央处理器)1001, RAM(随机存取器)1002, ROM(只读存储器)1003, 系统总线 1004, 硬盘控制器 1005, 鼠标控制器 1006, 键盘控制器 1007, 显示器控制器 1008, 硬盘 1009, 鼠标 1010, 键盘 1012, 显示器 1013。在这些部件中,与系统总线 1004 相连的有 CPU1001、RAM1002、ROM1003、硬盘控制器 1005, 鼠标控制器 1006, 键盘控制器 1007 和显示器控制器 1008。硬盘 1009 与硬盘控制器 1005 相连,鼠标 1010 与鼠标控制器 1006 相连,键盘 1012 与键盘控制器 1007 相连,以及显示器 1013 与显示器控制器 1008 相连。

[0052] 图 10 中每个部件的功能在本技术领域内都是众所周知的,并且图 10 所示的结构也是常规的。通常作为软件存储在硬盘 1009 中的计算机可读指令控制。在图 3 中所示的流程图的基础上,对于一个技术领域内熟练的技术人员无需创造性的工作即可开发出一个或更多的软件,这样开发出的软件将执行图 3 所示的断点调试方法。

[0053] 专业人员应该还可以进一步意识到,结合本文中所公开的实施例描述的各示例的单元及算法步骤,能够以电子硬件、计算机软件或者二者的结合来实现,为了清楚地说明硬件和软件的可互换性,在上述说明中已经按照功能一般性地描述了各示例的组成及步骤。这些功能究竟以硬件还是软件方式来执行,取决于技术方案的特定应用和设计约束条件。专业技术人员可以对每个特定的应用来使用不同方法来实现所描述的功能,但是这种实现不应认为超出本发明的范围。

[0054] 以上所述的具体实施方式,对本发明的目的、技术方案和有益效果进行了进一步详细说明,所应理解的是,以上所述仅为本发明的具体实施方式而已,并不用于限定本发明的保护范围,凡在本发明的精神和原则之内,所做的任何修改、等同替换、改进等,均应包含在本发明的保护范围之内。

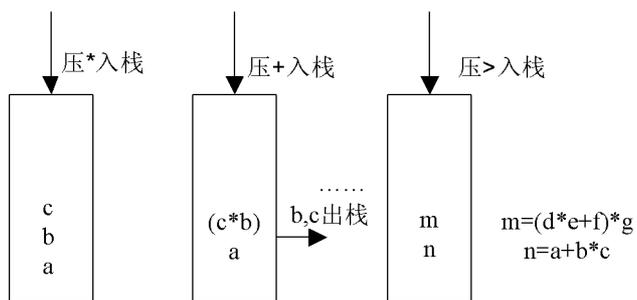


图 1

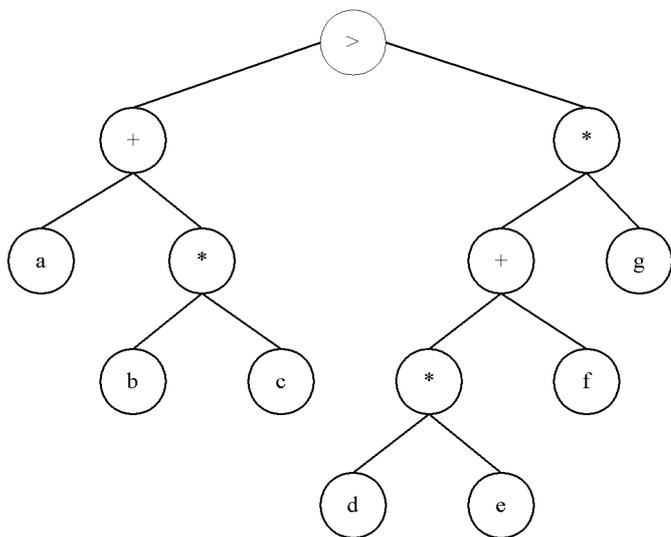


图 2

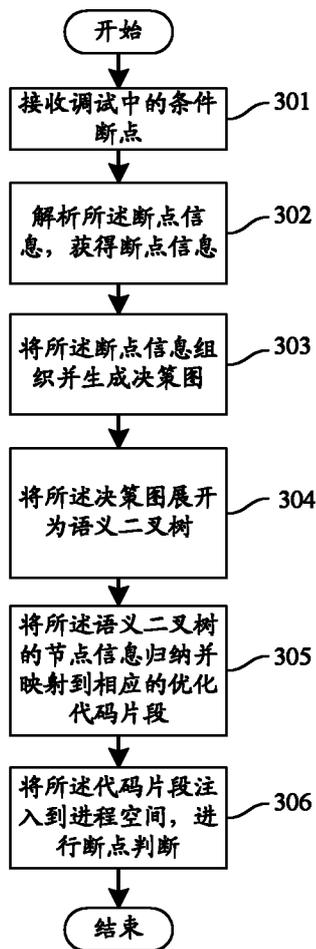


图 3

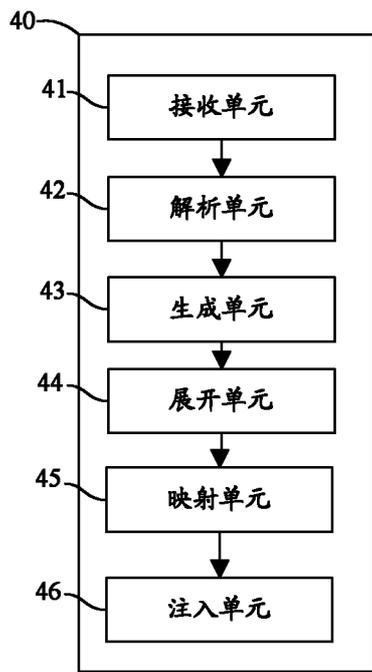


图 4

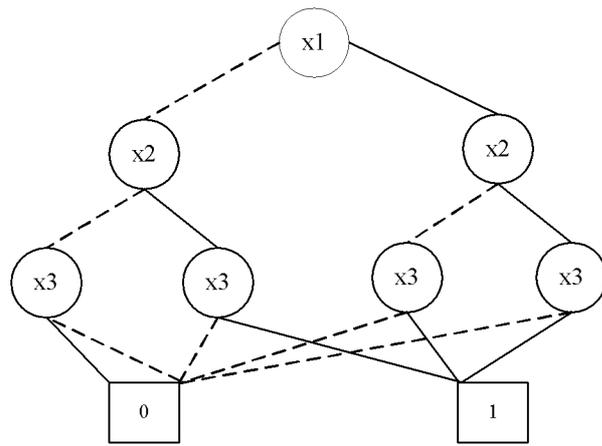


图 5

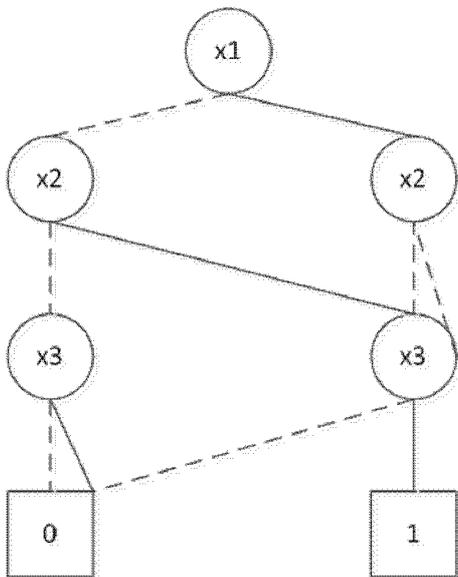


图 6

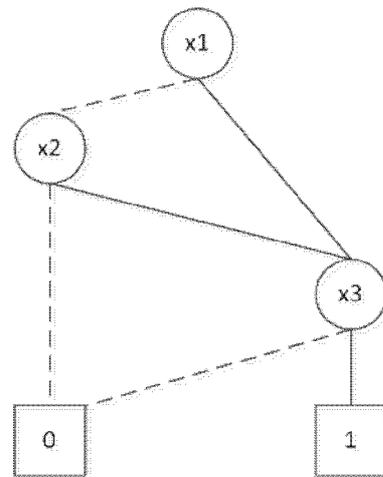


图 7

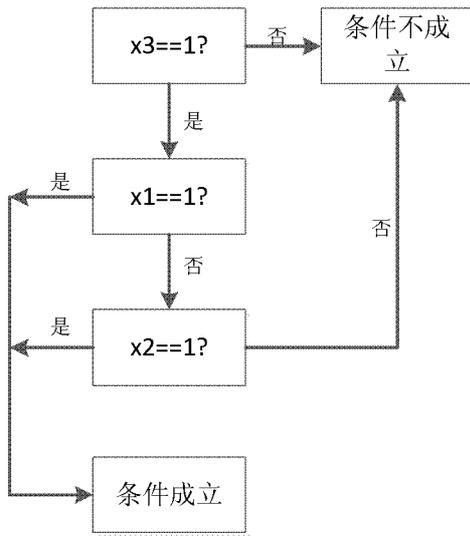


图 8

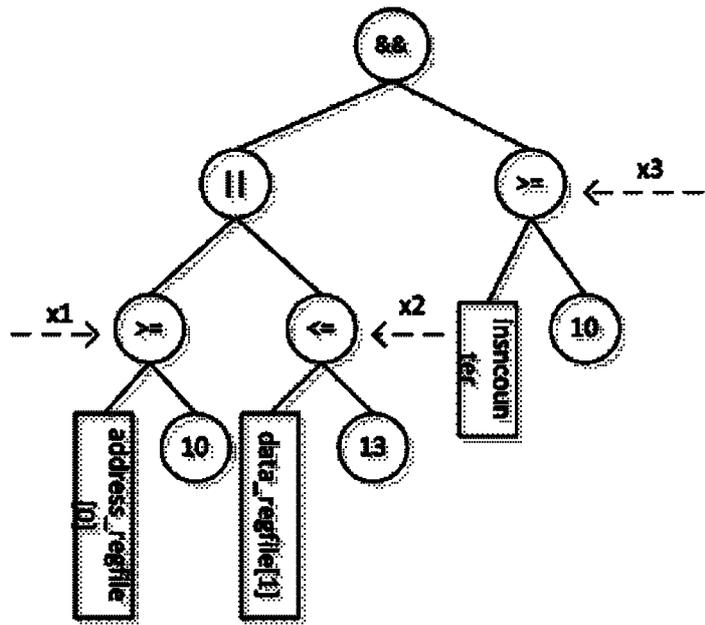


图 9

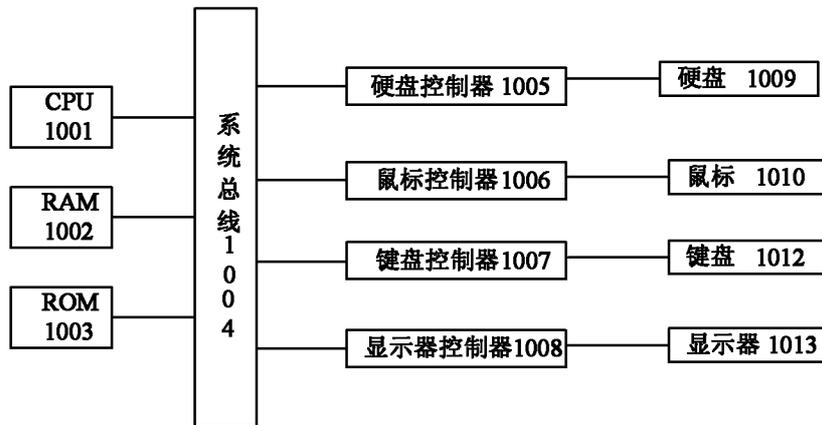


图 10