

(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国际局

(43) 国际公布日
2014 年 2 月 13 日 (13.02.2014)



(10) 国际公布号
WO 2014/023069 A1

- (51) 国际专利分类号:
G06F 9/44 (2006.01)
- (21) 国际申请号: PCT/CN2012/085154
- (22) 国际申请日: 2012 年 11 月 23 日 (23.11.2012)
- (25) 申请语言: 中文
- (26) 公布语言: 中文
- (30) 优先权:
201210283230.4 2012 年 8 月 10 日 (10.08.2012) CN
- (71) 申请人: 国网浙江省电力公司电力科学研究院
(ELECTRIC POWER RESEARCH INSTITUTE OF
STATE GRID ZHEJIANG ELECTRIC POWER
COMPANY) [CN/CN]; 中国浙江省杭州市朝晖八区
华电弄 1 号, Zhejiang 310014 (CN)。 国家电网公司
(STATE GRID CORPORATION OF CHINA)
[CN/CN]; 中国北京市西城区西长安街 86 号, Beijing
100031 (CN)。
- (72) 发明人: 朱炯 (ZHU, Jiong); 中国浙江省杭州市朝
晖八区华电弄 1 号, Zhejiang 310014 (CN)。 姚力
(YAO, Li); 中国浙江省杭州市朝晖八区华电弄 1 号,
Zhejiang 310014 (CN)。 李少腾 (LI, Shaoteng); 中国

浙江省杭州市朝晖八区华电弄 1 号, Zhejiang 310014
(CN)。 楼轶 (LOU, Yi); 中国浙江省杭州市朝晖八
区华电弄 1 号, Zhejiang 310014 (CN)。 胡琰俊 (HU,
Yingjun); 中国浙江省杭州市朝晖八区华电弄 1 号,
Zhejiang 310014 (CN)。 吴幸 (WU, Xing); 中国浙江
省杭州市朝晖八区华电弄 1 号, Zhejiang 310014
(CN)。

(74) 代理人: 北京集佳知识产权代理有限公司
(UNITALEN ATTORNEYS AT LAW); 中国北京市朝
阳区建国门外大街 22 号赛特广场 7 层, Beijing
100004 (CN)。

(81) 指定国 (除另有指明, 要求每一种可提供的国家保
护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG,
BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR,
CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB,
GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP,
KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT,
LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY,
MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT,
QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST,
SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ,
VC, VN, ZA, ZM, ZW。

[见续页]

(54) Title: METHOD AND SYSTEM FOR CLEARING REGISTER IN DECOMPILING DATA FLOW ANALYSIS

(54) 发明名称: 一种反编译数据流分析中的寄存器清除方法及系统

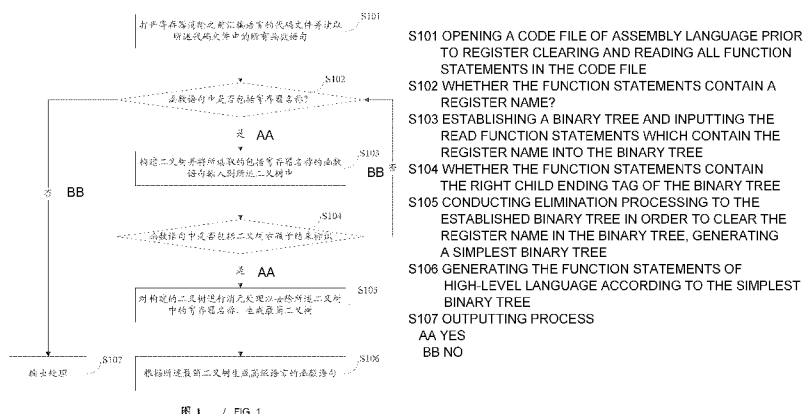


图 1 / FIG. 1

(57) Abstract: A method and a system for clearing a register in decompiling data flow analysis. The method comprises: reading all function statements in a code file and sequentially judging for the read function statements, and when a function statement contains a register name, establishing a binary tree and inputting the function statement into the binary tree; and further sequentially judging for the function statements containing the register name, and when the function statements contain the right child ending tag of the binary tree, conducting elimination processing to the established binary tree in order to clear the register name in the binary tree, generating a simplest binary tree, and generating the function statements of high-level language according to the simplest binary tree. With the invention, since all statements can be read at a time, it is not necessary to read and write the statements at many times. Further, since the binary tree is established according to the read function statements and the elimination processing is conducted to the binary tree, the function statements which do not contain the register name can be obtained very conveniently and rapidly, and the execution efficiency is high.

(57) 摘要:

[见续页]



WO 2014/023069 A1



(84) **指定国** (除另有指明, 要求每一种可提供的地区保护): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG,

CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)。

本国际公布:

— 包括国际检索报告(条约第 21 条(3))。

本发明公开了一种反编译数据流分析中的寄存器清除方法及系统, 可以读取代码文件中的所有函数语句并对所读取的函数语句依次进行判断, 当函数语句中包括寄存器名称时, 构建二叉树并将该函数语句输入到所述二叉树中; 进一步对包含有寄存器名称的函数语句依次进行判断, 当该函数语句中包括二叉树右孩子结束标识时, 对构建的二叉树进行消元处理以去除所述二叉树中的寄存器名称, 生成最简二叉树, 根据所述最简二叉树生成高级语言的函数语句。由于可以一次将所有语句进行读取, 因此本发明不用多次进行读写。同时, 由于根据读取的函数语句构建了二叉树并对二叉树进行消元处理, 因此可以很方便、快捷的得到不包含寄存器名称的函数语句, 执行效率高。

一种反编译数据流分析中的寄存器清除方法及系统

本申请要求于 2012 年 08 月 10 日提交中国专利局、申请号为 201210283230.4、发明名称为“一种反编译数据流分析中的寄存器清除方法及系统”的中国专利申请的优先权，其全部内容通过引用结合在本申请中。

5 技术领域

本发明涉及反编译技术领域，特别是涉及一种反编译数据流分析中的寄存器清除方法及系统。

背景技术

反编译技术作为计算机逆向技术中不可或缺的一部分，在分析系统漏洞、
10 软件安全以及病毒木马等领域得到了广泛应用。反编译技术可以将汇编语言)转换为高级语言(如 C 语言)。在反编译技术中，最重要、最困难的部分是数据流分析。数据流分析是一种用于收集计算机程序在不同点计算的值的的技术。具体的，可以使用程序控制流图(CFG, control flow graph)来确定对变量的一次赋值可能传播到程序中的哪些部分。

15 在数据流分析中，主要包括寄存器的清除和条件码的清除两个阶段。其中，现有的寄存器清除阶段一般采用如下方法：让两个语句相互映射，然后消去其中叠加的寄存器。

这种方法会造成磁盘的频繁读写，耗费了大量时间，执行效率低。

发明内容

20 为解决上述技术问题，本发明实施例提供一种反编译数据流分析中的寄存器清除方法及系统，以解决现有寄存器清除方法执行效率低的问题，技术方案如下：

一种反编译数据流分析中的寄存器清除方法，包括：

25 打开寄存器清除之前汇编语言的代码文件并读取所述代码文件中的所有函数语句；

对所读取的函数语句依次进行判断，判断函数语句中是否包括寄存器名

称,如果是,则构建二叉树并将该函数语句输入到所述二叉树中;

对包含有寄存器名称的函数语句依次进行判断,判断该函数语句中是否包括二叉树右孩子结束标识,如果是,则对构建的二叉树进行消元处理以去除所述二叉树中的寄存器名称,生成最简二叉树,根据所述最简二叉树生成高级语言
5 的函数语句;否则,继续判断后续函数语句中是否包括寄存器名称。

优选的,所述寄存器名称为:EAX、AX、BX、CX、DX、SP、SI、BP或DI。

优选的,所述打开寄存器清除之前汇编语言的代码文件并读取所述代码文件中的所有函数语句,包括:

- 10 打开寄存器清除之前汇编语言的代码文件;
根据所述代码文件中的所有函数语句构建单链表;
遍历所述单链表。

优选的,当所述寄存器名称为EAX时,所述二叉树右孩子结束标识为“eax;”。

- 15 优选的,所述构建二叉树并将该函数语句输入到所述二叉树中,包括:
构建二叉树;
将该函数语句中等号左边的代码输入到所述二叉树中的左孩子中;
将该函数语句中等号右边的代码输入到所述二叉树中的右孩子中。

- 20 优选的,所述进行消元处理以去除所述二叉树中的寄存器名称,生成最简二叉树,包括:

使用顶层根节点的右孩子中的代码替换顶层根节点下一层节点的右孩子中的寄存器名称并删除所述顶层根节点的下一层节点。

- 25 一种反编译数据流分析中的寄存器清除系统,包括:读取单元、寄存器名称判断单元、二叉树构建单元、结束标识判断单元、消元单元和高级语言生成单元,

所述读取单元,用于打开寄存器清除之前汇编语言的代码文件并读取所述代码文件中的所有函数语句;

所述寄存器名称判断单元,用于对所读取的函数语句依次进行判断,判断函数语句中是否包括寄存器名称,如果是,则触发所述二叉树构建单元;

所述二叉树构建单元,用于构建二叉树并将该函数语句输入到所述二叉树中;

- 5 所述结束标识判断单元,用于对包含有寄存器名称的函数语句依次进行判断,判断该函数语句中是否包括二叉树右孩子结束标识,如果是,则触发所述消元单元;否则发送一判断指令到所述寄存器名称判断单元,使所述寄存器名称判断单元继续判断后续函数语句中是否包括寄存器名称;

- 10 所述消元单元,用于对构建的二叉树进行消元处理以去除所述二叉树中的寄存器名称,生成最简二叉树;

所述高级语言生成单元,用于根据所述最简二叉树生成高级语言的函数语句。

优选的,所述读取单元包括:文件打开单元、单链表构建子单元和单链表遍历子单元,

- 15 所述文件打开子单元,用于打开寄存器清除之前汇编语言的代码文件;

所述单链表构建子单元,用于根据所述代码文件中的所有函数语句构建单链表;

所述单链表遍历子单元,用于遍历所述单链表。

- 20 优选的,所述二叉树构建单元包括:二叉树构建子单元、左孩子子单元和右孩子子单元,

所述二叉树构建子单元,用于构建二叉树;

所述左孩子子单元,用于将该函数语句中等号左边的代码输入到所述二叉树中的左孩子中;

- 25 所述右孩子子单元,用于将该函数语句中等号右边的代码输入到所述二叉树中的右孩子中。

优选的,所述消元单元具体设置为:

使用顶层根节点的右孩子中的代码替换顶层根节点下一层节点的右孩子

中的寄存器名称并删除所述顶层根节点的下一层节点。

通过应用以上技术方案,本发明提供的一种反编译数据流分析中的寄存器清除方法及系统,可以一次读取所有的函数语句,不用再多次进行读写。同时,由于根据读取的函数语句构建了二叉树并对二叉树进行消元处理,因此可以很方便、快捷的得到不包含寄存器名称的函数语句,执行效率高。

附图说明

为了更清除地说明本发明实施例或现有技术中的技术方案,下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍,显而易见地,下面描述中的附图仅仅是本发明中记载的一些实施例,对于本领域普通技术人员来讲,在不付出创造性劳动的前提下,还可以根据这些附图获得其他的附图。

图 1 为本发明实施例提供的一种反编译数据流分析中的寄存器清除方法的流程示意图;

图 2 为本发明实施例提供的一种汇编语言函数语句的示意图;

图 3 为本发明实施例提供的另一种反编译数据流分析中的寄存器清除方法的流程示意图;

图 4 为本发明实施例提供的一种反编译数据流分析中的寄存器清除方法中单链表的示意图;

图 5 为本发明实施例提供的另一种反编译数据流分析中的寄存器清除方法的流程示意图;

图 6 为本发明实施例提供的汇编语言语句的示意图;

图 7 为本发明实施例提供的二叉树的示意图;

图 8 为本发明实施例提供的二叉树的示意图;

图 9 为本发明实施例提供的二叉树的示意图;

图 10 为本发明实施例提供的二叉树的示意图;

图 11 为本发明实施例提供的二叉树的示意图;

图 12 为本发明实施例提供的二叉树的示意图;

图 13 为本发明实施例提供的一种反编译数据流分析中的寄存器清除系统的结构示意图;

图 14 为本发明实施例提供的另一种反编译数据流分析中的寄存器清除系统的结构示意图;

图 15 为本发明实施例提供的另一种反编译数据流分析中的寄存器清除系统的结构示意图。

10 具体实施方式

为了使本技术领域的人员更好地理解本发明中的技术方案,下面将结合本发明实施例中的附图,对本发明实施例中的技术方案进行清除、完整地描述,显然,所描述的实施例仅仅是本发明一部分实施例,而不是全部的实施例。基于本发明中的实施例,本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例,都应当属于本发明保护的范围。

如图 1 所示,本发明实施例提供的一种反编译数据流分析中的寄存器清除方法,可以包括:

S101、打开寄存器清除之前汇编语言的代码文件并读取所述代码文件中的所有函数语句;

其中,寄存器清除之前汇编语言的代码文件被打开后可以如图 2 所示。可以理解的是,汇编语言以“;”作为函数语句间的标识,每一个“;”都代表着一个函数语句的结束。本发明可以在一个读操作的前提下,将所述代码文件中的函数语句全部读取。

S102、对所读取的函数语句依次进行判断,判断函数语句中是否包括寄存

器名称, 如果是, 则执行步骤 S103; 否则执行步骤 S107;

所述寄存器名称可以为: EAX、AX、BX、CX、DX、SP、SI、BP 或 DI。
需要说明的一点是, 在汇编语言及高级语言中, EAX 可以为小写的 “eax”。

一般而言, 在约翰·冯·诺依曼计算机体系结构中, 计算机寄存器主要包
5 括 AX、BX、CX、DX、SP、SI、BP、DI 等。随着低 8 位、高 8 位、16 位、
32 位的不同, 对寄存器的读取以及名称的叫法也有不同。在汇编代码中, 使
用最多且具有迭代意义主要是 AX 中的 EAX。例如: 堆栈寄存器的用处主要
是恢复系统环境等。本发明在此并不限定寄存器的具体名称。

如图 2 所示, 在读取代码文件的函数语句后, 会发现第一句 (int a=1;))
10 中没有寄存器名称, 这时, 可以直接将该函数语句进行输出处理, 不用再进行
寄存器清除。

S103、构建二叉树并将该函数语句输入到所述二叉树中;

在计算机科学中, 二叉树是每个结点最多有两个子树的有序树。通常子树
的根被称作左子树 (left subtree) 和右子树 (right subtree)。每个节点的两个子
15 节点分别被称为左孩子和右孩子。

在实际应用中, 可以将包括寄存器名称的函数语句中等号左边的代码输入
到二叉树中的左孩子中, 将包括寄存器名称的函数语句中等号右边的代码输入
到二叉树中的右孩子中。

由于读取的函数语句为多个, 可以按照函数语句的顺序依次构建。如, 将
20 第一个包含寄存器名称的函数语句构造为高度为 2 的二叉树, 将等号左右边的
代码分别输入该二叉树的左孩子和右孩子中。将第二个包含寄存器名称的函数
语句中等号左边的代码输入上一左孩子的左孩子中, 将等号右边的代码输入上
一左孩子的右孩子中。按照以上规律, 依次对剩余的函数语句进行处理, 直到
函数语句中出现二叉树右孩子结束标识为止。

25 S104、对包含有寄存器名称的函数语句依次进行判断, 判断该函数语句中
是否包括二叉树右孩子结束标识, 如果是, 则执行步骤 S105, 否则, 继续执
行步骤 S102;

由于并不确定哪一个函数语句中存在二叉树右孩子结束标识,因此可以在步骤 S103 后,进一步依次判断包含有寄存器名称的函数语句中是否包括有二叉树右孩子结束标识。

其中,二叉树右孩子结束标识可以为“寄存器名称;”,如:“eax;”或“BX;”等。当寄存器名称为 eax 时,二叉树右孩子结束标识为“eax;”。

S105、对构建的二叉树进行消元处理以去除所述二叉树中的寄存器名称,生成最简二叉树;

具体的,可以使用顶层根节点的右孩子中的代码替换顶层根节点下一层节点的右孩子中的寄存器名称并删除所述顶层根节点的下一层节点。

10 S106、根据所述最简二叉树生成高级语言的函数语句;

在实际应用中,可以将最简二叉树左孩子中的代码放于等号左边,将最简二叉树右孩子中的代码放于等号右边,并和等号一起,形成高级语言的函数语句。

S107、进行输出处理。

15 本发明实施例提供的一种反编译数据流分析中的寄存器清除方法,可以一次读取所有的函数语句,不用再多次进行读写。同时,由于根据读取的函数语句构建了二叉树并对二叉树进行消元处理,因此可以很方便、快捷的得到不包含寄存器名称的函数语句,执行效率高。

20 如图 3 所示,在本发明实施例提供的另一种反编译数据流分析中的寄存器清除方法中,图 1 所示实施例中的步骤 S101 可以包括:

S101a、打开寄存器清除之前汇编语言的代码文件;

S101b、根据所述代码文件中的所有函数语句构建单链表;

25 单链表可以用一组地址任意的存储单元存放线性表中的数据元素。它以元素(数据元素的映象)加指针(指示后继元素存储位置)来表示结点。与直接读取代码文件中的函数语句相比,使用单链表可以随意改变读取顺序,更为方便。

S101c、遍历所述单链表。

对于图 2 中的第一个函数语句“int a=1;”，可以构建如图 4 所示的单链表，其中，单链表中每个字符占用一个节点，“i”为头结点，指针指向头结点“i”，该单链表中包括一首节点。指针可以遍历单链表，同时进行字符匹配操作。如果在指针遇到“;”之前，都未遇到 eax，则输出所遍历的代码。在图 4 中显示为遍历了“int a = 1;”，则将“int a = 1;”输出而不进行寄存器删除处理。

如图 5 所示，本发明实施例提供的另一种反编译数据流分析中的寄存器清除方法中，图 1 所示实施例中的步骤 S103 可以包括：

S103a、构建二叉树；

S103b、将该函数语句中等号左边的代码输入到所述二叉树中的左孩子中；

10 S103c、将该函数语句中等号右边的代码输入到所述二叉树中的右孩子中。

如图 6 所示，以图 2 所示的部分函数语句为例进行说明：

由于第一句函数语句“int a = 1;”中并不包括寄存器名称，因此直接进行输出处理。然后判断第二句，由于第二句中包含寄存器名称“eax”，因此将等号左边的代码“eax”输入二叉树的左孩子（节点 B）中，将右边的代码“int a”输入二叉树的右孩子（节点 C）中，其中，该二叉树的根节点为节点 A。形成如图 7 所示的二叉树，该二叉树的高度为 2。

按照顺序继续对下一函数语句进行判断，第三句中包含“eax”，同理，将“eax”放入 B 节点的左孩子（节点 D）中，将“eax+int a”放入 B 节点的右孩子（节点 E）中，形成如图 8 所示的二叉树，该二叉树的高度为 3。

20 按照上述方法继续对二叉树进行构建，并最终构建形成如图 9 所示的二叉树，该二叉树的高度为 5。由于图 6 所示的函数语句中的最后一句中包括有二叉树右孩子结束标识“eax;”，因此在处理完这句函数语句后，该二叉树就创建完毕。后续函数语句将形成新的二叉树。

在创建完二叉树后，需要对其进行消元处理，以去除二叉树中的寄存器名称。优选的，图 1 所示实施例中的步骤 S105 可以包括：

使用顶层根节点的右孩子中的代码替换顶层根节点下一层节点的右孩子中的寄存器名称并删除顶层根节点下一层节点。

下面以图 9 所示的已经创建好的二叉树为例进行说明:

首先,从图 9 所示二叉树的顶层根节点(节点 A)开始,将节点 A 的右孩子(节点 C)中的代码替换下一层的右孩子(节点 E)中的寄存器名称“eax”并删除该层的节点 B 和节点 C,形成如图 10 所示的二叉树。此时,顶层根节点的左孩子变为节点 D,右孩子变为节点 E。

然后,将图 10 所示二叉树中的顶层根节点的右孩子(节点 E)中的代码替换下一层右孩子(节点 G)中的寄存器名称,并删除节点 D 和节点 E,形成如图 11 所示的二叉树。此时,顶层根节点的左孩子变为节点 F,右孩子变为节点 G。

最后,将图 11 所示二叉树中的顶层根节点的右孩子(节点 G)中的代码替换下一层右孩子(节点 I)中的寄存器名称,并删除节点 F 和节点 G,形成如图 12 所示的二叉树。此时,顶层根节点的左孩子变为节点 H,右孩子变为节点 I。由于此时二叉树的高度已经为 2,已经是最简二叉树,因此不用再进行消元处理。

这时,可以执行步骤 S106,根据图 12 所示的最简二叉树,将最简二叉树的左孩子作为所要输出函数等号左边的代码,将最简二叉树的右孩子作为所要输出的函数等号右边的代码,并最终得到所要输出的高级语言函数语句为:“int a1 = int a + int a + int a;”。可以看到,该高级语言函数语句中并不包含寄存器名称,本发明已经成功的将其去除。

与本发明提供的方法实施例相对应,本发明还提供了一种反编译数据流分析中的寄存器清除系统。

如图 13 所示,本发明实施例提供的一种反编译数据流分析中的寄存器清除系统,可以包括:读取单元 100、寄存器名称判断单元 200、二叉树构建单元 300、结束标识判断单元 400、消元单元 500 和高级语言生成单元 600,

所述读取单元 100,用于打开寄存器清除之前汇编语言的代码文件并读取所述代码文件中的所有函数语句;

其中,寄存器清除之前汇编语言的代码文件被打开后可以如图 2 所示。可

以理解的是，汇编语言以“;”作为函数语句间的标识，每一个“;”都代表着一个函数语句的结束。本发明可以在一个读操作的前提下，将所述代码文件中的函数语句全部读取。

所述寄存器名称判断单元 200，用于对所读取的函数语句依次进行判断，
5 判断函数语句中是否包括寄存器名称，如果是，则触发所述二叉树构建单元 300;

所述寄存器名称可以为：EAX、AX、BX、CX、DX、SP、SI、BP 或 DI。
需要说明的一点是，在汇编语言及高级语言中，EAX 可以为小写的“eax”

一般而言，在约翰·冯·诺依曼计算机体系结构中，计算机寄存器主要包
10 括 AX、BX、CX、DX、SP、SI、BP、DI 等。随着低 8 位、高 8 位、16 位、
32 位的不同，对寄存器的读取以及名称的叫法也有不同。在汇编代码中，使用最多且具有迭代意义主要是 AX 中的 EAX。例如：堆栈寄存器的用处主要是恢复系统环境等。本发明在此并不限定寄存器的具体名称。

如图 2 所示，在读取代码文件的函数语句后，会发现第一句（int a=1;）
15 中没有寄存器名称，这时，可以直接将该函数语句进行输出处理，不用再进行寄存器清除。

所述二叉树构建单元 300，用于构建二叉树并将该函数语句输入到所述二叉树中；

在计算机科学中，二叉树是每个结点最多有两个子树的有序树。通常子树
20 的根被称作左子树（left subtree）和右子树（right subtree）。每个节点的两个子节点分别被称为左孩子和右孩子。

在实际应用中，可以将包括寄存器名称的函数语句中等号左边的代码输入到二叉树中的左孩子中，将包括寄存器名称的函数语句中等号右边的代码输入到二叉树中的右孩子中。

25 由于读取的函数语句为多个，可以按照函数语句的顺序依次构建。如，将第一个包含寄存器名称的函数语句构造为高度为 2 的二叉树，将等号左右边的代码分别输入该二叉树的左孩子和右孩子中。将第二个包含寄存器名称的函数

语句中等号左边的代码输入上一左孩子的左孩子中,将等号右边的代码输入上一左孩子的右孩子中。按照以上规律,依次对剩余的函数语句进行处理,直到函数语句中出现二叉树右孩子结束标识为止。

所述结束标识判断单元 400,用于对包含有寄存器名称的函数语句依次进行判断,判断该函数语句中是否包括二叉树右孩子结束标识,如果是,则触发
5 所述消元单元 500;否则发送一判断指令到所述寄存器名称判断单元 200,使所述寄存器名称判断单元 200 继续判断后续函数语句中是否包括寄存器名称;

其中,二叉树右孩子结束标识可以为“寄存器名称;”,如:“eax;”或“BX;”等。当寄存器名称为 eax 时,二叉树右孩子结束标识为“eax;”。

10 所述消元单元 500,用于对构建的二叉树进行消元处理以去除所述二叉树中的寄存器名称,生成最简二叉树;

具体的,可以使用顶层根节点的右孩子中的代码替换顶层根节点下一层节点的右孩子中的寄存器名称并删除所述顶层根节点的下一层节点。

15 所述高级语言生成单元 600,用于根据所述最简二叉树生成高级语言的函数语句。

在实际应用中,可以将最简二叉树左孩子中的代码放于等号左边,将最简二叉树右孩子中的代码放于等号右边,并和等号一起,形成高级语言的函数语句。

可以理解的是,在生成高级语言函数语句后就可以对其进行输出处理。

20 本发明实施例提供的一种反编译数据流分析中的寄存器清除系统,可以一次读取所有的函数语句,不用再多次进行读写。同时,由于根据读取的函数语句构建了二叉树并对二叉树进行消元处理,因此可以很方便、快捷的得到不包含寄存器名称的函数语句,执行效率高。

如图 14 所示,本发明实施例提供的另一种反编译数据流分析中的寄存器
25 清除系统中,所述读取单元 100 可以包括:文件打开单元 110、单链表构建子单元 120 和单链表遍历子单元 130,

所述文件打开子单元 110,用于打开寄存器清除之前汇编语言的代码文件;

所述单链表构建子单元 120, 用于根据所述代码文件中的所有函数语句构建单链表;

单链表可以用一组地址任意的存储单元存放线性表中的数据元素。它以元素(数据元素的映象) 加指针(指示后继元素存储位置) 来表示结点。与直接读取代码文件中的函数语句相比, 使用单链表可以随意改变读取顺序, 更为方便。

所述单链表遍历子单元 130, 用于遍历所述单链表。

对于图 2 中的第一个函数语句“int a=1;”, 可以构建如图 4 所示的单链表, 其中, 单链表中每个字符占用一个节点, “i” 为头结点, 指针指向头结点 “i”, 该单链表中包括一首节点。指针可以遍历单链表, 同时进行字符匹配操作。如果在指针遇到 “;” 之前, 都未遇到 eax, 则输出所遍历的代码。在图 4 中显示为遍历了 “int a = 1;”, 则将 “int a = 1;” 输出而不进行寄存器删除处理。

如图 15 所示, 本发明实施例提供的另一种反编译数据流分析中的寄存器清除系统中, 所述二叉树构建单元 300 可以包括: 二叉树构建子单元 310、左孩子子单元 320 和右孩子子单元 330,

所述二叉树构建子单元 310, 用于构建二叉树;

所述左孩子子单元 320, 用于将该函数语句中等号左边的代码输入到所述二叉树中的左孩子中;

所述右孩子子单元 330, 用于将该函数语句中等号右边的代码输入到所述二叉树中的右孩子中。

如图 6 所示, 以图 2 所示的部分函数语句为例进行说明:

由于第一句函数语句 “int a = 1;” 中并不包括寄存器名称, 因此直接进行输出处理。然后判断第二句, 由于第二句中包含寄存器名称 “eax”, 因此将等号左边的代码 “eax” 输入二叉树的左孩子(节点 B) 中, 将右边的代码 “int a” 输入二叉树的右孩子(节点 C) 中, 其中, 该二叉树的根节点为节点 A。形成如图 7 所示的二叉树, 该二叉树的高度为 2。

按照顺序继续对下一函数语句进行判断, 第三句中包含 “eax”, 同理, 将

“eax”放入B节点的左孩子(节点D)中,将“eax+int a”放入B节点的右孩子(节点E)中,形成如图8所示的二叉树,该二叉树的高度为3。

按照上述方法继续对二叉树进行构建,并最终构建形成如图9所示的二叉树,该二叉树的高度为5。由于图6所示的函数语句中的最后一句中包括有二叉树右孩子结束标识“eax;”,因此在处理完这句函数语句后,该二叉树就创建完毕。后续函数语句将形成新的二叉树。

在创建完二叉树后,需要对其进行消元处理,以去除二叉树中的寄存器名称。优选的,消元单元500可以具体设置为:

使用顶层根节点的右孩子中的代码替换顶层根节点下一层节点的右孩子中的寄存器名称并删除所述顶层根节点的下一层节点。

下面以图9所示的已经创建好的二叉树为例进行说明:

首先,从图9所示二叉树的顶层根节点(节点A)开始,将节点A的右孩子(节点C)中的代码替换下一层的右孩子(节点E)中的寄存器名称“eax”并删除该层的节点B和节点C,形成如图10所示的二叉树。此时,顶层根节点的左孩子变为节点D,右孩子变为节点E。

然后,将图10所示二叉树中的顶层根节点的右孩子(节点E)中的代码替换下一层右孩子(节点G)中的寄存器名称,并删除节点D和节点E,形成如图11所示的二叉树。此时,顶层根节点的左孩子变为节点F,右孩子变为节点G。

最后,将图11所示二叉树中的顶层根节点的右孩子(节点G)中的代码替换下一层右孩子(节点I)中的寄存器名称,并删除节点F和节点G,形成如图12所示的二叉树。此时,顶层根节点的左孩子变为节点H,右孩子变为节点I。由于此时二叉树的高度已经为2,已经是最简二叉树,因此不用再进行消元处理。

这时,可以触发高级语言生成单元600,根据图12所示的最简二叉树,将最简二叉树的左孩子作为所要输出函数等号左边的代码,将最简二叉树的右孩子作为所要输出的函数等号右边的代码,并最终得到所要输出的高级语言函

数语句为：“int a1 = int a + int a + int a;”。可以看到，该高级语言函数语句中并不包含寄存器名称，本发明已经成功的将其去除。

为了描述的方便，描述以上装置时以功能分为各种单元分别描述。当然，在实施本发明时可以把各单元的功能在同一个或多个软件和/或硬件中实现。

5 通过以上的实施方式的描述可知，本领域的技术人员可以清除地了解到本发明可借助软件加必需的通用硬件平台的方式来实现。基于这样的理解，本发明的技术方案本质上或者说对现有技术做出贡献的部分可以以软件产品的形式体现出来，该计算机软件产品可以存储在存储介质中，如 ROM/RAM、磁碟、光盘等，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，
10 或者网络设备等）执行本发明各个实施例或者实施例的某些部分所述的方法。

本说明书中的各个实施例均采用递进的方式描述，各个实施例之间相同相似的部分互相参见即可，每个实施例重点说明的都是与其他实施例的不同之处。尤其，对于系统实施例而言，由于其基本相似于方法实施例，所以描述得比较简单，相关之处参见方法实施例的部分说明即可。以上所描述的系统实施
15 例仅仅是示意性的，其中所述作为分离部件说明的单元可以是或者也可以不是物理上分开的，作为单元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。本领域普通技术人员在不付出创造性劳动的情况下，即可以理解并实施。

20 本发明可用于众多通用或专用的计算系统环境或配置中。例如：个人计算机、服务器计算机、手持设备或便携式设备、平板型设备、多处理器系统、基于微处理器的系统、置顶盒、可编程的消费电子设备、网络 PC、小型计算机、大型计算机、包括以上任何系统或设备的分布式计算环境等等。

本发明可以在由计算机执行的计算机可执行指令的一般上下文中描述，例
25 如程序模块。一般地，程序模块包括执行特定任务或实现特定抽象数据类型的例程、程序、对象、组件、数据结构等等。也可以在分布式计算环境中实践本发明，在这些分布式计算环境中，由通过通信网络而被连接的远程处理设备来

执行任务。在分布式计算环境中，程序模块可以位于包括存储设备在内的本地和远程计算机存储介质中。

需要说明的是，在本文中，诸如第一和第二等之类的关系术语仅仅用来将一个实体或者操作与另一个实体或操作区分开来，而不一定要求或者暗示这些

5 实体或操作之间存在任何这种实际的关系或者顺序。

以上所述仅是本发明的具体实施方式，应当指出，对于本技术领域的普通技术人员来说，在不脱离本发明原理的前提下，还可以做出若干改进和润饰，这些改进和润饰也应视为本发明的保护范围。

权 利 要 求

1、一种反编译数据流分析中的寄存器清除方法，其特征在于，包括：

打开寄存器清除之前汇编语言的代码文件并读取所述代码文件中的所有函数语句；

5 对所读取的函数语句依次进行判断，判断函数语句中是否包括寄存器名称，如果是，则构建二叉树并将该函数语句输入到所述二叉树中；

对包含有寄存器名称的函数语句依次进行判断，判断该函数语句中是否包括二叉树右孩子结束标识，如果是，则对构建的二叉树进行消元处理以去除所述二叉树中的寄存器名称，生成最简二叉树，根据所述最简二叉树生成高级语
10 言的函数语句；否则，继续判断后续函数语句中是否包括寄存器名称。

2、根据权利要求1所述的方法，其特征在于，所述寄存器名称为：EAX、AX、BX、CX、DX、SP、SI、BP或DI。

3、根据权利要求1或2所述的方法，其特征在于，所述打开寄存器清除之前汇编语言的代码文件并读取所述代码文件中的所有函数语句，包括：

15 打开寄存器清除之前汇编语言的代码文件；
根据所述代码文件中的所有函数语句构建单链表；
遍历所述单链表。

4、根据权利要求1所述的方法，其特征在于，当所述寄存器名称为EAX时，所述二叉树右孩子结束标识为“eax;”。

20 5、根据权利要求1所述的方法，其特征在于，所述构建二叉树并将该函数语句输入到所述二叉树中，包括：

构建二叉树；
将该函数语句中等号左边的代码输入到所述二叉树中的左孩子中；
将该函数语句中等号右边的代码输入到所述二叉树中的右孩子中。

25 6、根据权利要求1所述的方法，其特征在于，所述进行消元处理以去除所述二叉树中的寄存器名称，生成最简二叉树，包括：

使用顶层根节点的右孩子中的代码替换顶层根节点下一层节点的右孩子

中的寄存器名称并删除所述顶层根节点的下一层节点。

7、一种反编译数据流分析中的寄存器清除系统，其特征在于，包括：读取单元、寄存器名称判断单元、二叉树构建单元、结束标识判断单元、消元单元和高级语言生成单元，

5 所述读取单元，用于打开寄存器清除之前汇编语言的代码文件并读取所述代码文件中的所有函数语句；

所述寄存器名称判断单元，用于对所读取的函数语句依次进行判断，判断函数语句中是否包括寄存器名称，如果是，则触发所述二叉树构建单元；

10 所述二叉树构建单元，用于构建二叉树并将该函数语句输入到所述二叉树中；

所述结束标识判断单元，用于对包含有寄存器名称的函数语句依次进行判断，判断该函数语句中是否包括二叉树右孩子结束标识，如果是，则触发所述消元单元；否则发送一判断指令到所述寄存器名称判断单元，使所述寄存器名称判断单元继续判断后续函数语句中是否包括寄存器名称；

15 所述消元单元，用于对构建的二叉树进行消元处理以去除所述二叉树中的寄存器名称，生成最简二叉树；

所述高级语言生成单元，用于根据所述最简二叉树生成高级语言的函数语句。

20 8、根据权利要求7所述的系统，其特征在于，所述读取单元包括：文件打开单元、单链表构建子单元和单链表遍历子单元，

所述文件打开子单元，用于打开寄存器清除之前汇编语言的代码文件；

所述单链表构建子单元，用于根据所述代码文件中的所有函数语句构建单链表；

所述单链表遍历子单元，用于遍历所述单链表。

25 9、根据权利要求7所述的系统，其特征在于，所述二叉树构建单元包括：二叉树构建子单元、左孩子子单元和右孩子子单元，

所述二叉树构建子单元，用于构建二叉树；

所述左孩子子单元,用于将该函数语句中等号左边的代码输入到所述二叉树中的左孩子中;

所述右孩子子单元,用于将该函数语句中等号右边的代码输入到所述二叉树中的右孩子中。

- 5 10、根据权利要求7所述的系统,其特征在于,所述消元单元具体设置为:
使用顶层根节点的右孩子中的代码替换顶层根节点下一层节点的右孩子中的寄存器名称并删除所述顶层根节点的下一层节点。

— 1/8 —

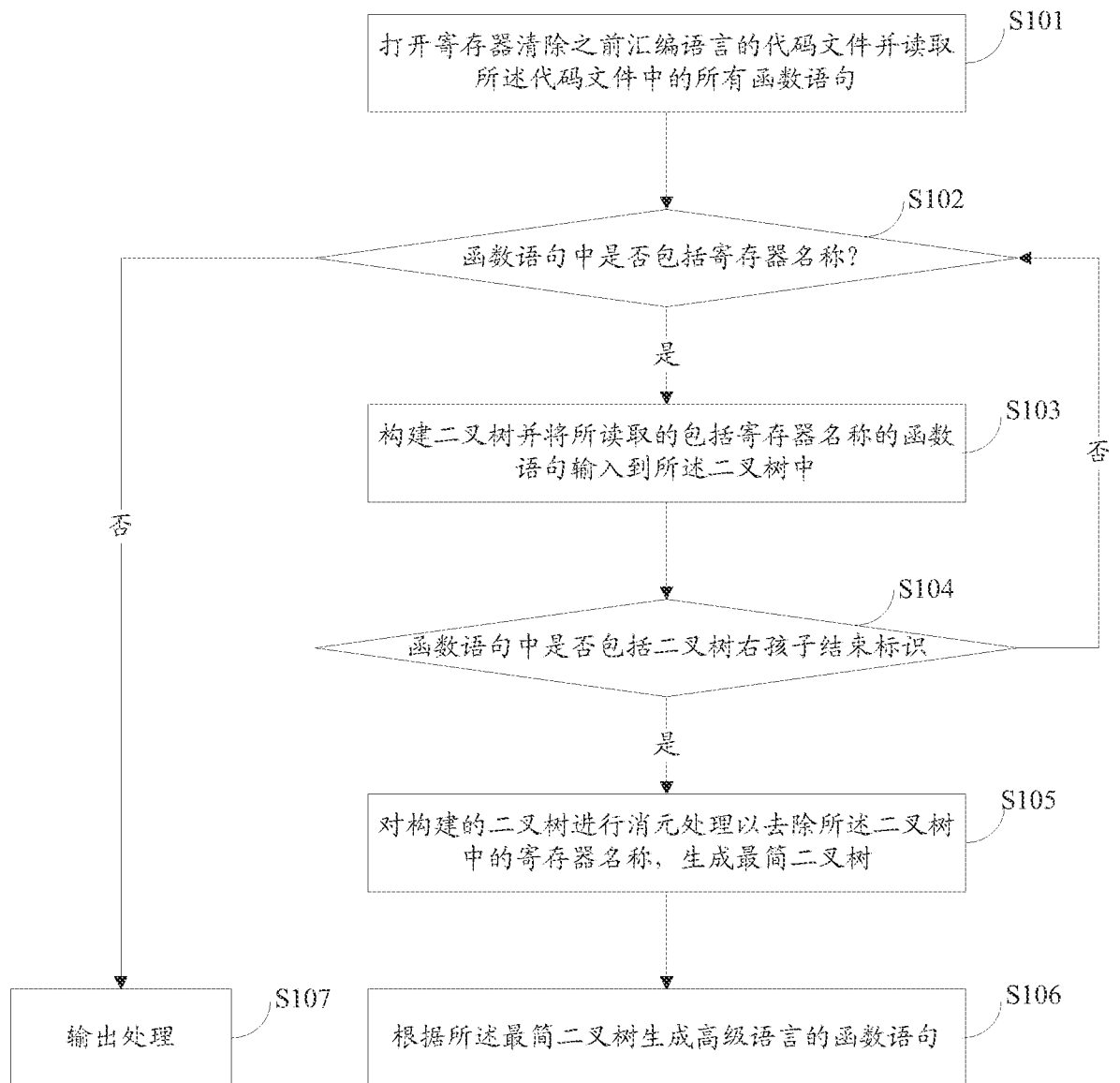


图 1

—2/8—

```
int a = 1 ;  
eax = int a ;  
eax = eax + int a ;  
eax = eax + int a ;  
int al = eax ;  
eax = int a ;  
eax = eax + int al ;  
eax = eax + int a ;  
int b = eax ;  
eax = int a ;  
eax = eax + int b ;  
eax = eax + int al ;  
eax = eax + int a ;  
int c = eax ;  
eax = int a ;  
eax = eax + int c ;  
eax = eax + int b ;  
eax = eax + int c ;  
int d = eax ;  
eax = int a ;  
eax = eax + int b ;  
eax = eax + int al ;  
int f = eax ;
```

图 2

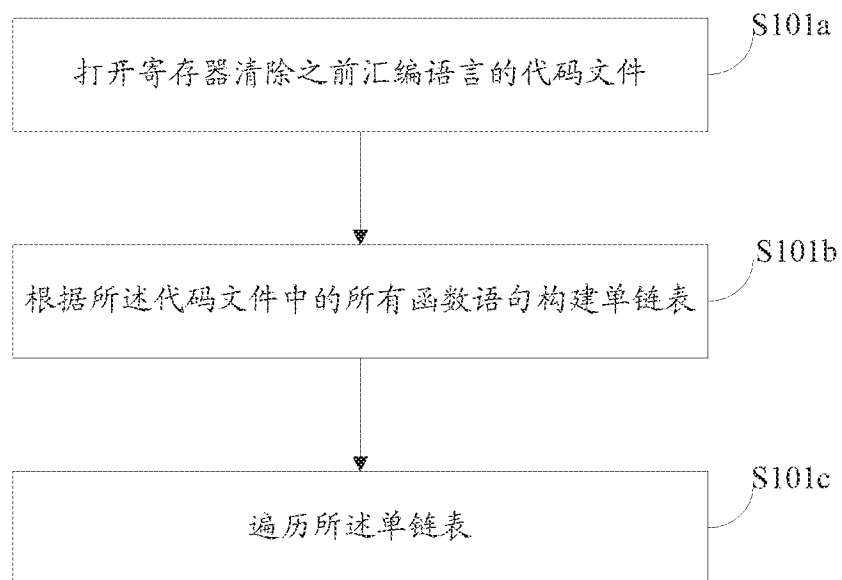


图 3

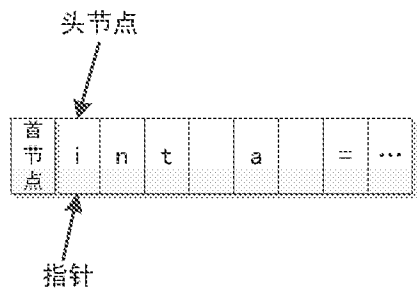


图 4

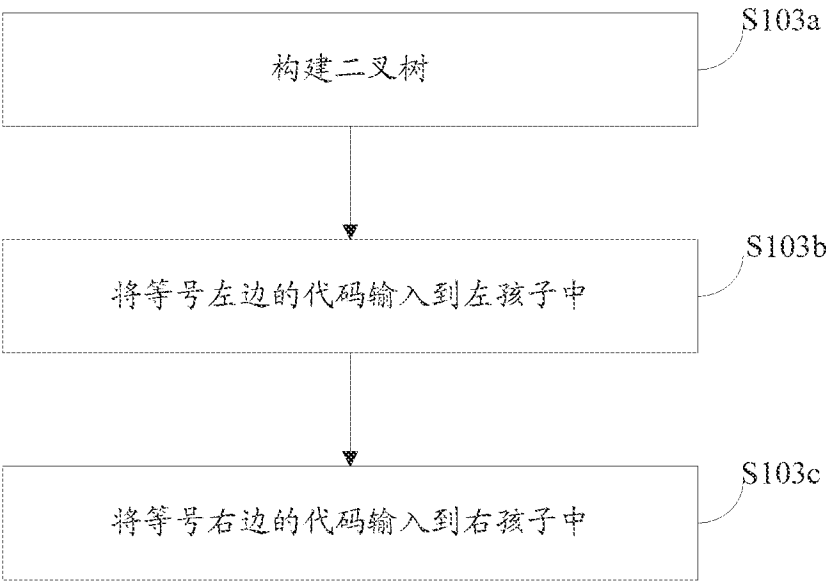


图 5

```
int a = 1 ;  
eax = int a ;  
eax = eax + int a ;  
eax = eax + int a ;  
int a1 = eax ;
```

图 6

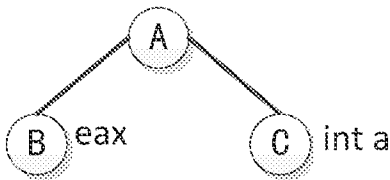


图 7

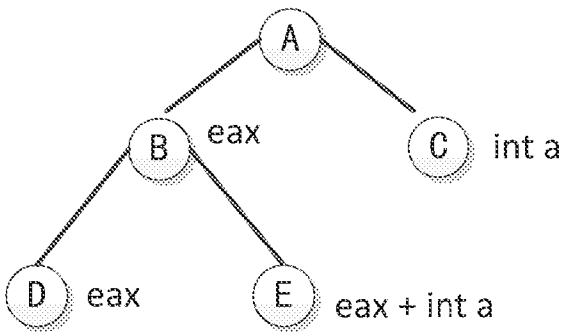


图 8

—5/8—

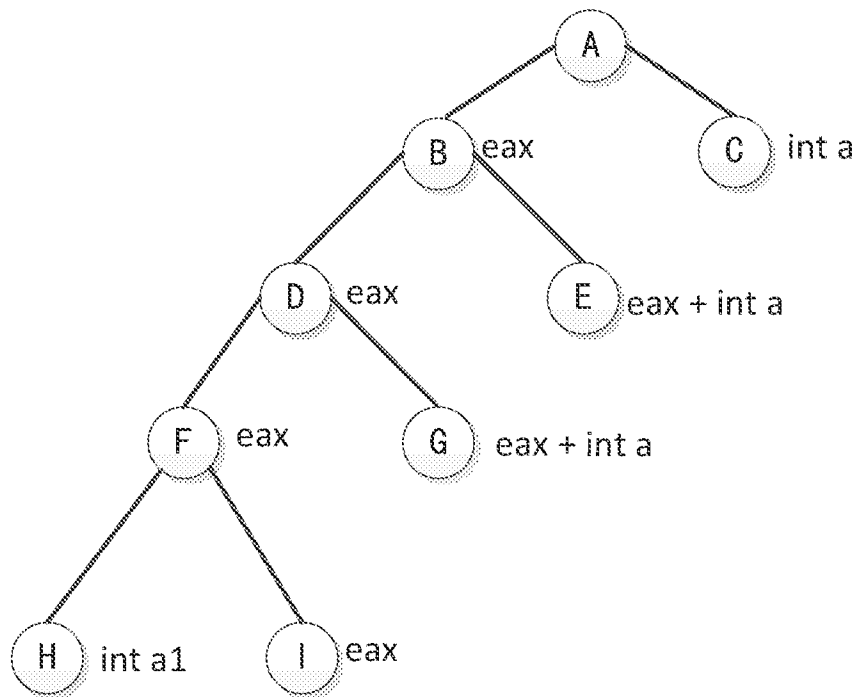


图 9

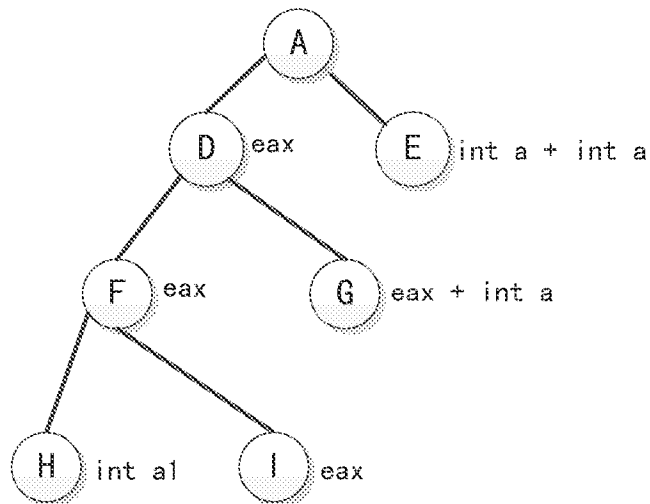


图 10

—6/8—

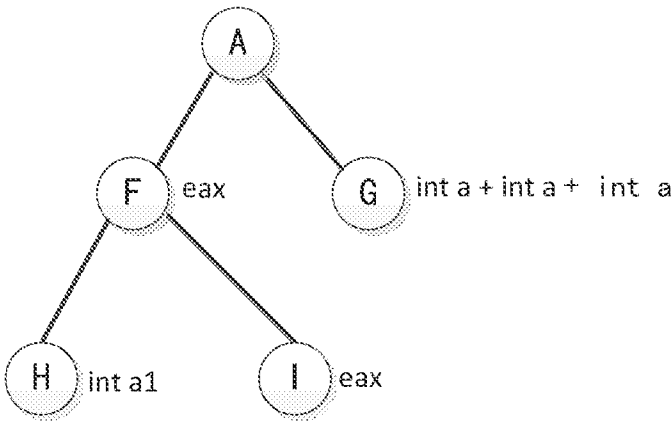


图 11

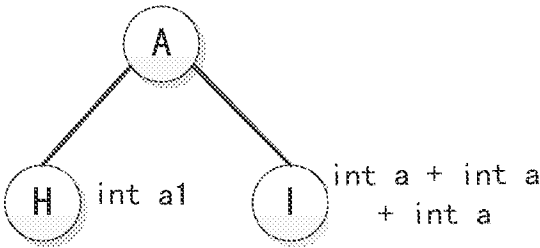


图 12

—7/8—

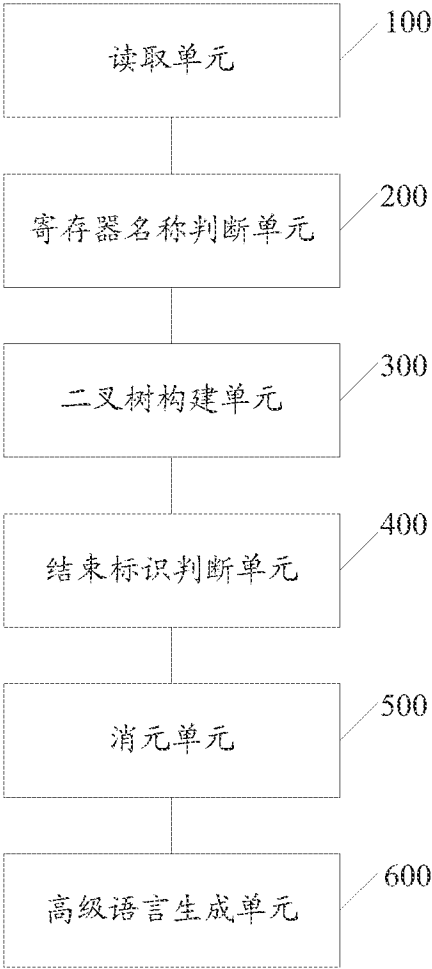


图 13



图 14

— 8/8 —

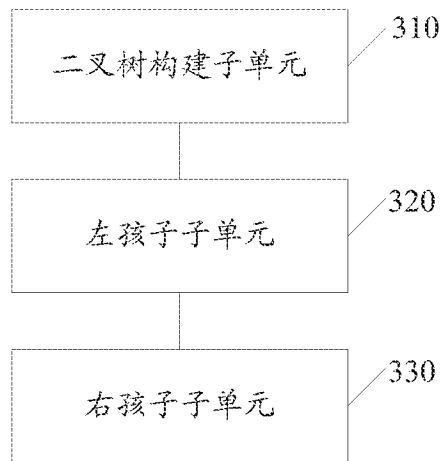


图 15

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2012/085154

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/44 (2006.01) i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

database: CNABS, DWPI, CNTXT, USTXT, CNKI

search term: compile, assemble, register, memory, binary tree, clear, child

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 101751334 A (UNIV CHINESE PEOPLES LIBERATION ARMY NAT) 23 June 2010 (23.06.2010) see claim 1, description, paragraphs [0005], [0091] to [0098] and figure 1	1-10
A	CN 1900910 A (CHINESE ACAD SCI COMPUTING TECHNOLOGY IN) 24 January 2007 (24.01.2007) see the whole document	1-10
A	CN 101561779 A (INT BUSINESS MACHINES CORP.) 21 October 2009 (21.10.2009) see the whole document	1-10
A	US 6516319 B1 (INT BUSINESS MACHINES CORP.) 04 February 2003 (04.02.2003) see the whole document	1-10

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" earlier application or patent but published on or after the international filing date	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&" document member of the same patent family
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search 03 May 2013 (03.05.2013)	Date of mailing of the international search report 23 May 2013 (23.05.2013)
Name and mailing address of the ISA State Intellectual Property Office of the P. R. China No. 6, Xitucheng Road, Jimenqiao Haidian District, Beijing 100088, China Facsimile No. (86-10) 62019451	Authorized officer XIE, Zhiyuan Telephone No. (86-10) 62411648

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2012/085154

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
CN 101751334 A	23.06.2010	CN 101751334 B	21.03.2012
CN 1900910 A	24.01.2007	CN 100377089 C	26.03.2008
CN 101561779 A	21.10.2009	None	
US 6516319 B1	04.02.2003	None	

国际检索报告

国际申请号
PCT/CN2012/085154

A. 主题的分类

G06F 9/44(2006.01)i

按照国际专利分类(IPC)或者同时按照国家分类和 IPC 两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

IPC: G06F

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称, 和使用的检索词(如使用))

数据库: CNABS, DWPI, CNTXT, USTXT, CNKI

检索词: 编译, 汇编, 寄存器, 存储器, 二叉树, 二乘树, 清除, 孩子

compile, assemble, register, memory, binary tree, clear, child

C. 相关文件

类 型*	引用文件, 必要时, 指明相关段落	相关的权利要求
X	CN 101751334 A (中国人民解放军国防科学技术大学) 23. 6 月 2010 (23.06.2010) 参见权利要求 1, 说明书第 0005 段, 第 0091-0098 段, 图 1	1-10
A	CN 1900910 A (中国科学院计算技术研究所) 24. 1 月 2007 (24.01.2007) 参见全文	1-10
A	CN 101561779 A (国际商业机器公司) 21. 10 月 2009 (21.10.2009) 参见全文	1-10
A	US 6516319 B1 (国际商业机器公司) 04. 2 月 2003 (04.02.2003) 参见全文	1-10

☐ 其余文件在 C 栏的续页中列出。

☒ 见同族专利附件。

* 引用文件的具体类型:

“A” 认为不特别相关的表示了现有技术一般状态的文件

“E” 在国际申请日的当天或之后公布的在先申请或专利

“L” 可能对优先权要求构成怀疑的文件, 或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件(如具体说明的)

“O” 涉及口头公开、使用、展览或其他方式公开的文件

“P” 公布日先于国际申请日但迟于所要求的优先权日的文件

“T” 在申请日或优先权日之后公布, 与申请不相抵触, 但为了理解发明之理论或原理的在后文件

“X” 特别相关的文件, 单独考虑该文件, 认定要求保护的发明不是新颖的或不具有创造性

“Y” 特别相关的文件, 当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时, 要求保护的发明不具有创造性

“&” 同族专利的文件

国际检索实际完成的日期
03. 5 月 2013 (03.05.2013)

国际检索报告邮寄日期
23.5 月 2013 (23.05.2013)

ISA/CN 的名称和邮寄地址:
中华人民共和国国家知识产权局
中国北京市海淀区蓟门桥西土城路 6 号 100088
传真号: (86-10)62019451

授权官员

谢志远
电话号码: (86-10) 010-62411648

国际检索报告
关于同族专利的信息

国际申请号
PCT/CN2012/085154

检索报告中引用的 专利文件	公布日期	同族专利	公布日期
CN 101751334 A	23.06.2010	CN 101751334 B	21.03.2012
CN 1900910 A	24.01.2007	CN 100377089 C	26.03.2008
CN 101561779 A	21.10.2009	无	
US 6516319 B1	04.02.2003	无	