[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 11/36

[12] 发明专利说明书

[21] ZL 专利号 02127806.7

[45] 授权公告日 2005 年 5 月 25 日

[11] 授权公告号 CN 1203409C

[22] 申请日 2002.8.6 [21] 申请号 02127806.7
[30] 优先权
[32] 2001. 8. 6 [33] JP [31] 238031/2001
[71] 专利权人 松下电器产业株式会社
地址 日本大阪府
[72] 发明人 津幡真太郎 隅田清彦
审查员 丁文勋

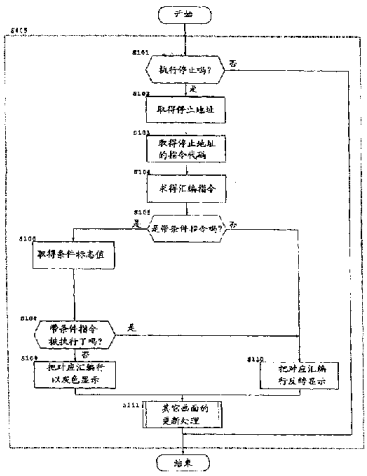
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 马铁良 叶恺东

权利要求书 3 页 说明书 21 页 附图 30 页

[54] 发明名称 软件调试方法和软件开发辅助方法

[57] 摘要

在对具备带条件指令的微电脑上的程序开发进行辅助的软件开发辅助系统中，以把带条件指令所实现的程序条件逻辑结构的执行过程向利用者易理解地进行显示为目的。 为此，在调试器中具有进行程序的步骤执行处理时判定停止地址的指令是否为带条件指令的功能；当指令为带条件指令的情况下取得指令的条件标志值的功能；根据条件标志值的值判定带条件指令是否被执行的功能；根据判定结果改变停止地址的指令显示方法向画面上显示的功能。 根据此结构，当带条件指令由于条件不成立而没有被执行的情况下，能够改变显示方法向利用者提示，并易于确认程序的执行经过。



ISSN1008-4274

1．一种在把调试对象的微电脑程序向画面上显示时，上述程序的执行停止时的停止地址可与其他地址区别显示的调试方法，具有：

判定在进行上述程序步骤执行处理时上述停止地址的指令是否是带条件指令的步骤；

在上述指令是带条件指令的情况下取得上述指令的条件标志值的步骤；

根据上述条件标志值的值判定带条件指令是否被执行的步骤；

根据上述判定结果来改变上述停止地址的指令显示方法向上述画面上显示的步骤。

2．权利要求 1 记载的调试方法，具有：

在根据带条件指令是否被执行的判定结果改变停止地址指令的显示方法向画面上显示的时候，把停止地址部分的颜色反转显示的步骤。

3．权利要求 1 记载的调试方法，具有：

在根据带条件指令是否被执行的判定结果改变停止地址指令的显示方法向画面上显示的时候，把停止地址部分和条件标志部分的颜色反转显示的步骤。

4．一种在进行调试对象的微电脑程序执行处理后，把上述程序的指令跟踪输出的调试方法，具有：

所执行的指令是标志更新指令的情况下把被更新的标志值作为跟踪数据输出的步骤；

判定跟踪输出对象的指令是否是带条件指令的步骤；

在上述判定结果是跟踪输出对象指令是带条件指令的情况下作为上述跟踪数据取得所输出的标志值的步骤；

根据上述标志值的值判定带条件指令是否被执行的步骤；

根据上述判定结果来改变跟踪输出的指令输出方法并输出的步骤。

5．一种在把调试对象的微电脑程序向画面上显示的时候，能够将上述程序的执行停止时的停止地址与其他地址区别显示的软件开发辅助方法，具有：

从上述调试对象程序检出全部带条件指令的条件标志的生存区间

的步骤;

判定显示对象地址的指令是否为带条件指令的步骤;

在上述判定结果为带条件指令的情况下判定上述显示对象地址和停止地址是否存在于在上述带条件指令被参照的条件标志的同一生存区间内的步骤;

在上述判定结果是带条件指令的情况下取得在上述带条件指令被参照的条件标志值的步骤;

根据上述条件标志值的值来判定上述带条件指令是否被执行的步骤;

根据上述显示对象地址和停止地址是否存在于在上述带条件指令被参照的条件标志的同一生存区间内的判定结果和由上述条件标志值的值上述带条件指令是否被执行的判定结果，来改变停止地址以及其前后地址的指令显示方法向上述画面上显示的步骤。

6. 一种微电脑的编译方法，具有:

检出指令代码串中包含的带条件指令的条件标志的生存区间的步骤;

把上述检出的条件标志的生存区间的信息输出的步骤。

7. 一种在把调试对象的微电脑程序向画面上显示的时候，能把上述程序执行停止时的停止地址与其他地址区别显示的调试方法，具有:

判定显示对象地址的指令是否为带条件指令的步骤;

在上述判定结果为带条件指令的情况下判定上述显示对象地址是否存在于在上述带条件指令被参照的条件标志的生存区间内的步骤;

在上述判定结果为带条件指令的情况下取得在上述带条件指令被参照的条件标志值的步骤;

根据上述条件标志值的值来判定上述带条件指令是否被执行的步骤;

根据上述显示对象地址是否存在于上述条件标志的生存区间内的判定结果和由上述条件标志值的值上述带条件指令是否被执行的判定结果，来改变停止地址以及其前后地址的指令显示方法向上述画面上显示的步骤。

8. 权利要求 7 记载的调试方法，具有:

输入通过权利要求 6 记载的编译方法输出的条件标志的生存区间信息的步骤。

软件调试方法和软件开发辅助方法

技术领域

本发明涉及进行微电脑上的执行程序调试的软件调试方法和软件开发辅助方法，特别涉及具有条件指令的微电脑程序执行过程的显示方法。

现有技术

随着近年电子技术的发展，在家电设备等组合设备上加入微电脑等，进行伴随程序开发的设备开发。

而且，随着家电设备的高性能化，具备为达到高速化的带条件指令的高性能微电脑已经被采用。

在程序开发上，软件开发者，利用调试器，通过确认开发的程序是否按照设计的意图动作，进行程序调试。这时候，利用被称为程序逐次执行一条指令的逐步执行和显示系列执行结果的跟踪功能的调试器功能来进行程序动作的确认。

图 26 中，表示现有例子的软件开发辅助系统的结构。201 是 C 源程序，软件开发者是实现开发对象的应用系统的程序员。202 是，编译器，C 源程序 201 作为输入，输出由在开发对象的应用系统里使用的微电脑用的执行代码和调试信息组成的目标文件 203。204 是调试器，输入目标文件 203，从用户界面输入来自软件开发者的指示，确认调试对象的程序动作，进行调试。

下面对现有的调试器 204 的动作予以说明。图 27 中，表示调试器 204 的功能方框图。

301 是，调试器处理部，进行作为利用者的软件开发者的调试指令的输入，向利用者的信息显示和信息输出等的输入输出处理，还进行与输入的调试指令相对应的处理。

302 是执行目标环境，是目标文件 203 程序的执行环境，比如在计算机上构筑的微电脑模拟器。还有，执行目标环境 302 也可以是，作为目标的装备着微电脑和内存的评价板。调试器处理部 301 是，通过控制这个执行目标环境 302，实现调试器功能。

303 是目标代码信息，是根据利用者的指示把作为调试对象程序的目标文件 203 读入调试器 204 在内部存储的信息。

304 是调试信息，包含调试中必要的各种信息。这个是记录在目标文件 203，被读入调试器中。它的内容是由，比如函数名和变量名等符号和地址的对应的信息和，进行 C 语言程序的源代码行和地址的对应的行号信息和，记载变量是分配了寄存器还是内存等，哪些资源的配置信息组成。

305 是指令形式信息，是指令代码和指令形式对应的信息，还包含汇编形式的助记字符串的信息。如果参照这个信息，从存在于内存上的指令代码的内存值，能够实现反汇编。

306 是，跟踪数据信息，存储为进行跟踪输出的跟踪帧数据。

图 28 中，表示以前调试器的处理流程，调试器 204 起动后，在步骤 S401，进行调试器的作业用内存的初始化和显示用窗体的绘图等初始处理。

接下来，在步骤 S402，从利用者，接受调试器指令的输入。在这里，利用者，输入调试对象程序目标代码的读入指令和，程序执行开始和断点设定，内存转储等调试必要的指令。

接下来在步骤 S403，解释输入的指令，在步骤 S404，进行与输入的指令对应的各种处理。处理内容为以前的技术所以不再详细说明了。

接下来在步骤 S405，进行调试器的显示更新处理。在显示更新处理中包括，在显示代码的窗体显示程序执行停止地址前后的代码，更新显示内容的处理和，在显示内存内容的内存窗体，伴随内存内容的变更来更新显示内容的处理等。

在步骤 S404，在输入结束指令的情况下，在步骤 S406，进行调试器的作业区域的释放以及与执行目标环境的连接的切断等结束处理。

在图 29，表示步骤 S405 的显示更新处理的处理流程。在显示更新处理，显示更新在必要的情况下进行对应处理。

在步骤 S2801，在目标执行环境上，调查调试对象程序的执行状态，在执行停止的情况下，进入到步骤 S2802。执行没有停止，程序正在执行的情况下，因为显示更新没有必要，所以结束处理。在执行停止的情况下，在后续的行，进行向开发者显示在调试对象程序的哪个

地方停止了的处理。

在步骤S2802，取得调试对象程序的停止地址。即，取得执行目标环境302上的微电脑程序计数器(PC)的寄存器的值。

接下来在步骤S2803，从在调试器内部存储的目标代码信息303，读出在步骤S2802取得的停止地址的指令代码。

接下来在步骤S2804，把指令代码反汇编，得到与指令代码对应的汇编行字符串。还有，反汇编是，通过参照调试器内部管理的指令形式信息305来实现的。

接下来在步骤S2805，把得到的汇编行字符串在画面上用反转的颜色显示，把先前反转显示的行返回通常的颜色。

接下来在步骤S2806，进行指令代码显示以外的内存内容等显示的更新处理。

用上面这样组成的调试器，程序开发者，比如通过把步骤执行指令连续地输入调试器，当前程序的执行指令被逐个的反转显示，便可以知道调试对象程序的执行经过，使调试成为可能。

还有，日本国专利第2892978号中公布的「软件·跟踪的生成装置及方法」是，在显示程序的执行过程的跟踪上，以把被执行的各种指令根据执行顺序分颜色显示为特征，由此把程序的执行过程易判别的显示出来。

不过，在具备带条件指令的微电脑，把在以前用多个指令实现的条件执行处理，用一个带条件的指令就能实现。为此，如果在以前，把执行过的指令和执行经过结合逐步跟踪的话能够确认程序的执行过程，但只通过跟踪被执行过的指令，便留下了确认困难的课题。

下面，采用带条件指令的程序的例子，把在现有例子里的调试器执行过程的显示予以说明。

图5中，表示调试对象的C语言程序的一部分。「01:」，「02:」等是为了说明方便的行号。是根据变量a的值，有条件执行02行和04行处理的程序。

在图6中，表示把前述C语言程序汇编成具备带条件指令的微电脑指令代码的结果。「0x400:」，「0x404:」等是，表示配置了指令的地址。在这里，C语言的if-else条件执行结构是，利用带条件指令来实现的。

在 0x400 地址段，寄存器 Ra 的值和 10 比较后的结果代入标志 c0。在 0x404 地址段，标志 c0 为真的情况下，向寄存器 Rb 传入寄存器 Ra 值的平方。标志 c0 为伪的情况下，在这个指令，不进行任何处理。

在 0x408 地址段，标志 c0 为伪的情况下，向寄存器 Rb 传送寄存器 Rc 的内容。标志 c0 为真的情况下，不进行任何处理。图 6 从 0x400 地址段到 0x408 地址段的指令组，和图 5 第 01 行到第 05 行的 if-else 语句记述的逻辑结构相对应。

下面，对于在以前的调试器上，利用者对前述 C 语言程序部分的动作从地址为 0x400 的地址段到 0x408 地址段的部分，针对以步骤执行来确认情况下的例子给予说明。

假定调试对象程序在 0x400 地址段停止执行，这种情况下，正如图 30(a)表示的那样，把停止地址 0x400 地址的汇编行反转后显示在代码显示画面上。

利用者，在步骤 S402，输入步骤执行指令。

在步骤 S403，进行输入指令的解释，取出指令类别为步骤执行等必要的信息。

在步骤 S404，根据在步骤 S403 取出的信息，进行步骤执行处理。步骤执行处理是，通过在作为执行目标环境 302 的微电脑模拟器上，控制指令执行使之执行一条指令来实现的。这样做的结果是，调试对象程序，在执行目标环境 302 上，从停止地址执行一条指令，然后停止执行。

在步骤 S405，为了进行显示更新处理，进入到步骤 S2801。

在步骤 S2801，判断执行是否停止。因为是执行停止所以进入到步骤 S2802。

在步骤 S2802，取得作为执行目标环境 302 的微电脑模拟器的程序计数器(PC)的值，取得停止地址。这种情况下，作为停止地址得到 0x404。

接下来，在步骤 S2803，从目标代码信息 303，读到停止地址 0x404 的指令代码，接着在步骤 S2804，参照指令形式信息 305，把指令代码反汇编，作为结果字符串取得汇编行『(c0) mul Ra，Ra，Rb』。

在步骤 S2805，把前述汇编行『(c0) mul Ra，Ra，Rb』的显示颜色更新为反转色。还有，把前面颜色反转的汇编行恢复到通常的颜色。

这个结果的显示例在图 30(b)中表示。

下面同样，重复步骤执行的情况下的显示例在图 30(c)，30(d)中表示。

通过确认反转显示的汇编行逐次变化下去，便能确认程序执行的过程。不过，以地址 0x404 的指令实现所得的处理『mul Ra，Ra，Rb』和以地址 0x408 的指令实现所得的处理『mov Rc，Rb』，难以确认实际上是否被执行。

即，有难以确认在 if-else 控制结构中记载的程序逻辑的课题。

还有，对于跟踪结果的显示也有同样的课题。

发明内容

本发明鉴于上述课题，其目的在于提供能够把用带条件指令来实现的程序条件逻辑结构的执行过程向利用者容易理解地显示出来的软件调试器和软件开发辅助系统。

第一发明的软件调试器是，在把调试对象的微电脑程序向画面上显示的时候，能够将程序执行停止时的停止地址与其他地址区别显示的软件调试器，具有判定在进行程序步骤执行处理时停止地址的指令是否是带条件指令的功能；在指令是带条件指令的情况下取得指令的条件标志值的功能；根据条件标志值的值判定带条件指令是否被执行的功能；根据判定结果来改变停止地址的指令显示方法向画面上显示的功能。

根据这个结构，带条件指令当条件不成立而没被执行的情况下，能够改变显示方法向利用者提示，利用者能够易于确认包含带条件指令的程序的执行过程。

在上述的第一发明结构中，理想的是具有在根据带条件指令是否被执行的判定结果改变停止地址指令的显示方法向画面上显示的时候，把停止地址部分反转显示的功能(第二发明)。或者，理想的是具有在根据带条件指令是否被执行的判定结果改变停止地址指令的显示方法向画面上显示的时候，把停止地址部分和条件标志部分反转显示的功能(第三发明)。

还有，第四发明的软件调试器是在进行调试对象的微电脑程序的执行处理后，把程序指令跟踪输出的软件调试器，具有所执行的指令

是标志更新指令的情况下把被更新的标志值作为跟踪数据输出的功能；判定跟踪输出对象的指令是否是带条件指令的功能；在判定结果是跟踪输出对象指令是带条件指令的情况下作为跟踪数据取得所输出的标志值的功能；根据标志值的值判定带条件指令是否被执行的功能；根据判定结果来改变跟踪输出的指令输出方法并输出的功能。

根据这个结构，在跟踪输出中，能够参照条件标志值，能够判定带条件指令的执行结果，在带条件指令因条件不成立没被执行的情况下，能够改变显示方法向利用者提示，利用者能够易于确认包含带条件指令的程序的执行过程。

还有，第五发明的软件开发辅助系统是，在把调试对象的微电脑程序向画面上显示的时候，能够将程序的执行停止时的停止地址与其他地址区别显示的软件开发辅助系统，具有从调试对象程序检出全部带条件指令的条件标志的生存区间的功能；判定显示对象地址的指令是否为带条件指令的功能；在判定结果为带条件指令的情况下判定显示对象地址和停止地址是否存在于在带条件指令被参照的条件标志的同一生存区间内的功能；在判定结果是带条件指令的情况下取得在带条件指令被参照的条件标志值的功能；根据条件标志值的值来判定带条件指令是否被执行的功能；根据显示对象地址和停止地址是否存在于在带条件指令被参照的条件标志的同一生存区间内的判定结果和由条件标志值的值带条件指令是否被执行的判定结果，来改变停止地址以及其前后地址的指令显示方法向画面上显示的功能。

根据这个结构，在进行程序执行停止时的显示的时候，如果在条件标志的生存区间内则判定带条件指令的执行，可改变显示方法向利用者提示，利用者能够易于确认包含带条件指令的程序的执行过程。

还有，第六发明的编译器是，微电脑的编译器，具有检出指令代码串中包含的带条件指令的条件标志的生存区间的功能；把检出的条件标志的生存区间的信息输出的功能。

还有，第七发明的软件调试器是，在把调试对象的微电脑程序向画面上显示的时候，能把程序执行停止时的停止地址与其他地址区别显示的软件调试器，具有判定显示对象地址的指令是否为带条件指令的功能；在判定结果为带条件指令的情况下判定显示对象地址是否存在于在带条件指令被参照的条件标志的生存区间内的功能；在判定结

果为带条件指令的情况下取得在带条件指令被参照的条件标志值的功能；根据条件标志值的值来判定带条件指令是否被执行的功能；根据显示对象地址是否存在于条件标志的生存区间内的判定结果和由条件标志值的值带条件指令是否被执行的判定结果，来改变停止地址以及其前后地址的指令显示方法向画面上显示的功能。

还有，第八发明的软件调试器是，在第七发明的软件调试器中，具有输入从第六发明的编译器输出的条件标志的生存区间信息的功能。

利用上述的第六发明的编译器，第七，第八发明的软件调试器便能够构成第五发明的软件开发辅助系统。

还有，第九发明的机器可读记录介质，记录了微电脑程序的机器语言代码以及其存储地址和带条件指令的条件标志的定义地址以及参照地址的信息。

附图说明

图 1 是本发明第一实施例的显示更新处理流程图。

图 2 是表示本发明第一实施例的软件开发辅助系统的结构图。

图 3 是本发明第一实施例的调试器功能方框图。

图 4 是本发明第一实施例的调试处理流程图。

图 5 是表示调试对象的 C 语言程序的一部分图。

图 6 是表示对应于调试对象的 C 语言程序的一部分的汇编例子图。

图 7 是表示本发明第一实施例的步骤执行画面的例子图。

图 8 是表示本发明第二实施例的软件开发辅助系统的结构图。

图 9 是本发明第二实施例的调试器功能方框图。

图 10 是本发明第二实施例的调试处理流程图。

图 11 是本发明第二实施例的模拟器的处理流程图。

图 12 是本发明第二实施例的模拟器的跟踪数据生成处理的处理流程图。

图 13(a)是本发明第二实施例的跟踪的帧数据说明图，图 13(b)是表示本发明第二实施例的跟踪的帧数据的例子图。

图 14 是本发明第二实施例的跟踪输出处理的处理流程图。

图 15 是表示本发明第二实施例的跟踪的输出例子图。

图 16 是表示本发明第三实施例的软件开发辅助系统结构图。

图 17 是本发明第三实施例的编译器的功能方框图。

图 18 是本发明第三实施例的标志生存区间信息输出部的处理流程图。

图 19 是本发明第三实施例的使用定义连锁信息的说明图。

图 20(a)是本发明第三实施例的标志生存区间信息的说明图，图 20(b)是表示本发明第三实施例的标志生存区间信息的例子图。

图 21 是本发明第三实施例的调试器的功能方框图。

图 22 是本发明第三实施例的调试器处理流程图。

图 23 是本发明第三实施例的显示更新处理流程图。

图 24(a) ~ (d)是表示本发明第三实施例的执行停止时的显示例子图。

图 25 是本发明实施例的指令形式信息的说明图。

图 26 是表示现有例的软件开发辅助系统的结构图。

图 27 是现有例的调试器的功能方框图。

图 28 是现有例的调试器的处理流程图。

图 29 是现有例的显示更新处理的流程图。

图 30 是表示现有例的步骤执行画面的例子图。

实施方式

对于本发明的实施例参照附图予以说明。

第一实施例

首先，对于本发明第一实施例予以说明。图 2 中，表示本发明第一实施例的软件开发辅助系统的结构。201 是 C 源程序，作为利用者的软件开发者是实现开发对象的应用系统的程序员。202 是，编译器，C 源程序 201 作为输入，输出由在开发对象的应用系统中使用的微电脑用的执行代码和调试信息构成的目标文件 203。204A 是，调试器，输入目标文件 203，把来自软件开发者的指示从用户接口输入，确认调试对象的程序动作等，进行调试。

下面，对于本实施例的调试器 204A 予以说明。图 3 中，表示调试器 204A 的功能方框图。

301 是，调试器处理部，进行输入来自作为利用者的软件开发者的调试器指令，和向利用者的信息显示和输出等的输入输出处理，还进行与输入的调试器指令相对应的处理。

302 是，执行目标环境，是目标代码 203 的程序执行环境，比如在计算机上构筑的微电脑模拟器。而且，执行目标环境 302 也可以是成为目标的微电脑或配置了内存的评价板。调试器处理部 301 是，通过控制这个执行目标环境 302 来实现调试的功能。

303 是，目标代码信息，根据利用者的指示把调试对象程序的目标文件 203 读入到调试器 204A 内在内部存储的信息。

304 是，调试信息，包含调试中必要的各种信息。在目标文件 203 中记录，被读入调试器。内容是由，比如函数名和变量名等符号和地址相对应的信息和，C 语言程序的源程序行和地址相对应的行号信息和，记载变量分配了寄存器或内存等，哪些资源的配置信息等组成。

305 是，指令形式信息，指令代码和指令形式相对应的信息，还包含汇编形式的助记字符串的信息。如果参照这个信息，从内存上存在的指令代码即内存值，能实现反汇编。

图 25 中，表示指令形式信息 305 的例子。这个指令形式信息是由，为判别指令代码的指令格式信息 3001，保持对应的助记字符串的助记字符串信息 3002，表示指令执行用去的时间的周期数信息 3003，表示是否为带条件指令的带条件指令信息 3004，表示是否是更新在带条件指令参照的条件标志的指令的标志更新信息 3005 构成，与各指令形式相对应，被管理着。在 3006，3007，3008 中，各表示一个例子。3006 是，add 指令的指令形式信息。指令格式是，以 2 进制表示，寄存器指定域是，以<R1>等表示。还有，表示不是带条件指令，不是标志更新指令。同样，在 3007，作为带条件指令的 add 指令的指令形式信息，3008 是，更新条件标志的 cmp 指令的指令形式信息。

306 是，跟踪数据信息，为了进行跟踪输出，存储跟踪帧数据。

图 4 中，表示调试处理部 301 的处理流程。调试器起动后，在步骤 S401，进行调试器作业用内存的初始化和显示用窗体的绘图等的初始处理。

接下来在步骤 S402，从利用者，接受调试器指令的输入。在这里，利用者输入调试对象程序目标代码的读入指令和程序执行开始或断点

的设定，内存的转储等调试所必需的指令。

接下来在步骤S403，解释输入的指令，在步骤S404，进行与输入的指令相对应的各种处理。处理内容因为是以前的技术不再详细说明。

接着在步骤S405，进行调试器的显示更新处理。在显示更新处理中有，在显示代码的代码窗体，更新显示程序执行停止地址前后的代码显示内容的处理，在显示内存内容的内存窗体，伴随内存内容的变更来更新显示内容的处理等。

在步骤S404，结束指令被输入的情况下，在步骤S406，进行调试器的作业领域的解放和与执行目标环境连接的切断等结束处理。

图1中，表示步骤S405的显示更新处理的处理流程

在步骤S101，执行目标环境302上，调查调试对象的程序执行状态，执行停止的情况下，进入步骤S102。执行没有停止，程序正在执行的情况下，因为显示的更新没有必要所以结束处理。执行停止的情况下，在后续的行，进行把调试对象的程序在哪个地方停止向程序开发者显示的处理。

在步骤S102，取得调试对象程序的停止地址。即，取得执行目标环境302上的微电脑程序计数器(PC)寄存器的值。

接下来在步骤S103，从在调试器内存储的目标代码信息303，读出在步骤S102取得的停止地址的指令代码。

接下来在步骤S104，把指令代码反汇编，得到与指令代码相对应的汇编行字符串。而且，反汇编是，通过参照在调试器内部管理的指令形式信息305实现的。

接下来在步骤S105，从指令代码参照在调试器内部管理的指令形式信息305，判定是否为带条件指令。是否为带条件指令，可以通过参照指令形式信息的带条件指令信息3004来判定。在是带条件指令的情况下，进入到步骤S106。在不是带条件指令的情况下，进入到步骤S110。

在步骤S106，参照指令形式信息305，求得条件标志的种类，从执行目标环境302取得前述条件标志的值。

在步骤S108，根据在步骤S106得到的条件标志值，判断带条件指令是否被执行。在带条件指令被执行的情况下，进入到步骤S110。

在带条件指令没有被执行的情况下，进入到步骤 S109。

在步骤 S109，带条件指令没有被执行的情况下，把没被执行的演算部分用浅灰色，把条件标志部用反转颜色显示，把前面反转显示的行恢复到通常的显示颜色。

在步骤 S110，把对应的汇编行字符串在画面上用反转的显示颜色显示，把前面反转显示的行恢复到通常的显示颜色。

在步骤 S111，进行指令代码显示以外的内存内容等的显示更新处理。

下面，具体地举个例子将动作予以说明。

图 5 中，表示调试对象的 C 语言程序的一部分。「01: 」，「02: 」等是，为了说明的行号。是根据变量 a 的值，02 行和 04 行的处理被有条件执行的程序。

图 6 中，表示将图 5 的 C 语言程序汇编成具有带条件指令的微电脑指令代码的结果。即，在编译器 202 汇编的结果的一部分。还有，「0x400: 」，「0x404: 」等是，表示配置了指令的地址。在这里，C 语言的 if-else 条件执行结构是，利用带条件指令来实现的。

在 0x400 地址段，寄存器 Ra 的值和 10 比较器结果带入标志 c0。在 0x404 地址段，标志 c0 为真的情况下，向寄存器 Rb 传送寄存器 Ra 的值的平方。标志 c0 为伪的情况下，在这个指令，不进行任何处理。

在 0x408 地址段，标志 c0 为伪的情况下，向寄存器 Rb 传送寄存器 Rc 的内容。标志 c0 为真的情况下，不进行任何处理。图 6 的从 0x400 地址段到 0x408 地址段的指令组是，与图 5 的第 01 行到第 05 行的 if-else 语句记述的逻辑结构相对应。

在图 6 所举的程序，在存器 Ra 的值为 10 的情况下，认为调试对象程序在 0x400 地址段停止执行。

这种情况下，如图 7(a)所表示的，把停止地址 0x400 地址段的汇编行颜色反转向代码显示画面上显示。这个显示画面是，与调试器 204A 连接的显示器(未图示)的画面。

利用者，在步骤 S402，输入步骤执行指令。

在步骤 S403，进行输入指令的解释，把指令的种类为步骤执行等必要的信息取出来。

在步骤 S404，根据在步骤 S403 取出的信息，进行步骤执行处理。

其结果，调试对象程序，在执行目标环境 302 上，从停止地址执行一条指令，而后停止执行。

在步骤 S405，为了进行显示更新处理，进入到步骤 S101。

在步骤 S101，判断执行是否停止。因为执行停止所以进入到步骤 S102。

在步骤 S102，从执行目标环境 302 取得微电脑的程序计数器(PC)的值，取得停止地址。这种情况下，作为停止地址取得 0x404。

接下来在步骤 S103，从目标代码信息 303，读到停止地址 0x404 的指令代码，接着在步骤 S104 参照指令形式信息 305，把指令代码反汇编，作为结果字符串取得汇编行『(c0) mul Ra，Ra，Rb』。

在步骤 S105，判定该指令为带条件指令，进入到步骤 S106。

在步骤 S106，取得该指令的条件标志 c0 的值。这种情况下，作为 c0 的值取得 1。

在步骤 S108，因为条件标志 c0 的值为 1，所以判定带条件指令被执行，进入到步骤 S110。

在步骤 S110，把汇编行『(c0) mul Ra，Ra，Rb』颜色反转显示，这个结果的显示例子在图 7(b)中表示。

接着，利用者输入了步骤执行指令的情况下，同样，进入到步骤 S101，S102，S103，S104，S105，S106，S108。

在步骤 S108，因为条件标志！c0 为 0，所以判定带条件指令没被执行，进入到步骤 S109。

在步骤 S109，把没被执行的演算部分『mov Rc，Rb』用浅灰色，把条件标志部『！c0』用反转显示颜色显示，把前面反转显示的行恢复到通常的显示颜色。其结果的显示例子在图 7(c)中表示。还有，在图 7(c)中，为了方便把『mov Rc，Rb』用四角括起来表示，实际上，认为没有四角括线，用浅灰色显示着。

同样，接着输入步骤执行指令的情况下，如图 7(d)表示的那样显示。

如上构成的调试器，利用者利用步骤执行逐步追踪程序执行的情况下，通过把没有执行的带条件指令改变其显示形式显示出来，能够将用带条件指令来实现的条件执行的逻辑易识别地显示出来，能够知道调试对象的程序执行经过，使调试成为可能

在本实施例，在步骤 S109，把在带条件指令没有执行情况下的显示方法作为，将没有执行的演算部分以浅灰色显示的，不过，覆盖取消线或把显示字体的大小变小，把显示字符串设为『未执行』等，其他的方法也可以

还有，在本实施例，在步骤 S109，把带条件指令没有执行情况下的显示方法作为，将停止地址部『0x408: 』和条件标志部『! c0』用反转显示颜色显示，将没有执行的演算部分作为浅灰色显示的，仅将停止地址部『0x408: 』用反转显示颜色显示，将条件标志部和没有执行的演算部分用浅灰色显示。或，覆盖取消线，或，将显示字体的大小变小，或，将显示字符串设为『未执行』也可以。

第二实施例

接着对于本发明第二实施例予以说明。图 8 是第二实施例的软件开发辅助系统的结构图，图 9 是，调试器 204B 的功能方框图，和第一实施例几乎相同的结构。第二实施例，和第一实施例相比，调试器 204B 内的处理，即在执行目标环境 302 进行的跟踪数据输出处理和，在步骤 S404 的跟踪输出处理的处理内容不一样。

在本发明第二实施例，执行目标环境 302 是在作为对象的微电脑的软件模拟器实现的。

图 10 是，调试器处理部 301 的处理流程图，是和本发明第一实施例一样的处理流程。

在调试器 204B，程序执行指令被输入的情况下，在步骤 S404 进行程序执行处理。程序执行处理是，对于执行目标环境 302 通过指示程序执行来进行。在本实施例，执行目标环境 302 是，由软件模拟器实现的，进行程序的执行是，通过进行指令模拟来实现的。

图 11 中，表示作为软件模拟器的执行目标环境 302 的指令模拟处理的处理流程。

在图 11，在步骤 S1101，读出模拟器的程序计数器寄存器(PC)的值，判定执行地址是否已到达既定的停止地址(比如 exit 函数)。如果到达结束处理，没有到达进入到步骤 S1102。

在步骤 S1102，从调试操作处理部把利用者指定的中断地址信息读出，判定执行地址是否到达该中断地址，如果到达结束处理，没有到达进入到步骤 S1103。还有，利用者，在通常调试对象程序的执行开始

前，把想使之中断的地址作为中断地址设定指令，传给步骤 S402。调试操作处理部，是在调试器处理部 301 中，在步骤 S402 得到把利用者输入的调试器指令在步骤 S403 解释的结果和中断地址的信息。

在步骤 S1103，进行一条指令的模拟。

在步骤 S1104，进行一条指令执行的跟踪数据的输出。图 12 中，表示跟踪数据输出处理 S1104 的处理流程。

跟踪结果是，把每条指令在调试器内的跟踪数据信息 306 中作为跟踪数据被记录。这个跟踪数据的结构在图 13(a)中表示。1301 是，帧号码域，记录着跟踪输出的信息号。1302 是，时刻域，记录着跟踪输出的时刻。1303 是，数据种类域，记录着跟踪数据的种类。1304 是，地址域，记录着与跟踪数据关联的地址。在跟踪数据记录了执行指令的情况下，记录前述指令的地址。1305 是，指令域，记录着执行的指令。1306 是，数据域，记录着与跟踪数据关联的数据。在跟踪数据记录了条件执行标志值的情况下，记录前述标志值

下面，参照图 12，对 1 条指令执行部分的跟踪数据输出处理 S1104 予以说明。

在步骤 S1201，记录执行指令的帧数据。记录帧号码，时刻，作为跟踪数据种类记录执行指令，记录执行地址，指令的汇编行。那之后，把帧号码增加 1。时刻是，取得模拟器管理的执行周期数，并记录。汇编行是，参照指令形式信息 305 求得。

在步骤 S1202，参照指令形式信息 305，判别现指令的种类。如果是标志更新指令，则进入到步骤 S1203，如果不是那样则结束。

在步骤 S1203，把条件标志的值新作为跟踪数据来记录。记录帧号码，时刻，作为跟踪数据种类记录条件标志的种类，记录执行地址，条件标志的值。那之后，把帧号码增加 1。条件标志的值是，从执行目标环境 302 的模拟器取得的。

下面，对从图 6 所示程序的 0x400 地址段执行到 0x40c 地址段的情况，具体予以说明。还有假定当初程序计数器的值为 0x400，寄存器 Ra 的值为 10。且，假定作为停止地址设为 0x410 地址段，中断地址没有被设定。

利用者，在步骤 S402 输入程序执行指令。在步骤 S403，解释输入的指令，在步骤 S404 进行程序执行处理。在步骤 S404，对作为软

件模拟器的执行目标环境 302，进行指令模拟器执行的控制。在软件模拟器，判定在步骤 S1101 执行地址 0x400 地址段是否为停止地址的 0x410 地址段。因为没有到达停止地址，所以进入到步骤 S1102。在步骤 S1102，进行中断地址的判定，因为中断地址没有被设定，所以进入到步骤 S1103。在步骤 S1103，进行 0x400 地址段的 cmp 指令的模拟。接着在步骤 S1104 为了进行跟踪数据的输出，而后进入到步骤 S1201。在步骤 S1201，将前述 cmp 指令的跟踪数据记录到跟踪数据信息 306 中。记录的跟踪数据信息在图 13(b)的 1307 中表示。接着在步骤 S1202 判定该指令是否为标志更新指令。前述 cmp 指令是更新标志 c0 的指令所以进入到步骤 S1203。在步骤 S1203，记录条件标志值跟踪数据。记录的跟踪数据信息在图 13(b)的 1308 中表示。下面同样，进行 0x404 地址段，0x408 地址段，0x40c 地址段的指令模拟，记录图 13(b)的 1309，1310，1311 中表示的跟踪数据。

在上面，程序执行处理的步骤 S404 结束，进入到步骤 S405，进行显示停止地址前后代码等的显示更新处理。那之后，等待下一条指令的输入(步骤 S402)。

接着，对利用者把跟踪数据信息 306 中蓄积的跟踪数据，输出的情况予以说明。

图 14 中，表示在从利用者输入了跟踪数据输出指令的情况下，在步骤 S404 进行跟踪数据输出处理时的处理流程。

跟踪数据输出，利用者为了确认调试对象程序的执行过程，程序执行后进行。

步骤 S1401 判定全部的跟踪数据输出是否完了。完了的情况下结束，没有完了的情况下进入到步骤 S1402。

在步骤 S1402，从跟踪数据信息 306 按顺序取出一条跟踪数据。

在步骤 S1403，判别跟踪数据的种类。在是条件标志值跟踪数据的情况下，进入到步骤 S1404，在是执行指令跟踪数据的情况下进入到步骤 S1405。

在步骤 S1404，从该跟踪数据读取条件标志的种类和其值而后存储。

在步骤 S1405，从该跟踪数据读取指令域，判定跟踪指令是否为带条件指令。带条件指令的情况下进入到步骤 S1406，不是带条件指令的

情况下，进入到步骤 S1409。

在步骤 S1406，条件指令的情况下，参照在步骤 S1404 存储的标志值。

在步骤 S1407，从条件标志值判定带条件指令是否被执行。被执行的情况下，进入到步骤 S1409。没有被执行的情况下，进入倒步骤 S1408。

在步骤 S1408，把没有被执行的演算部分用『< >』括起来然后输出指令，返回到步骤 S1401。

在步骤 S1409，输出执行的指令，返回到步骤 S1401，对全部跟踪数据，反复进行输出处理。

在上面，跟踪数据输出处理的步骤 S404 结束，进入到步骤 S405 的显示更新处理。这种情况下，因为不是停止地址和内存内容发生变化，所以不进行显示的更新。

下面，对从图 6 所示程序的 0x400 地址段执行到 0x40c 地址段的情况，具体予以说明。假定寄存器 Ra 的值为 10。这种情况下，在跟踪数据信息 306 中，存储着在图 13(b)的 1307，1308，1309，1310，1311 中表示的跟踪数据。

图 15 中，表示图 6 所示指令组被执行的情况下的，跟踪数据的输出例子。

利用者，在步骤 S402，输入跟踪数据输出指令。在步骤 S403，解释输入的指令，在步骤 S404 为了进行跟踪数据的输出处理，进入到步骤 S1401。在步骤 S1401，判定跟踪数据信息 306 中存储的全跟踪数据是否输出完毕。因为全跟踪数据没有输出完毕所以进入到步骤 S1402。在步骤 S1402，取出一条跟踪数据，得到 1307 中表示的跟踪数据。在步骤 S1403 判定跟踪数据的种类，1307 的种类是执行指令所以进入到在步骤 S1405。在步骤 S1405，判定是否为带条件指令。因为 1307 中记录的指令不是带条件指令所以进入到步骤 S1409。在步骤 S1409 以通常的形式输出『0x400: cmp Ra==10, c0』。

同样地顺次输出，对在带条件指令没有执行的地址 0x408 的指令，没有执行的演算部分输出『<mov Rc, Rb>』。

如上面一样，把图 13(b)中表示的跟踪数据作为输入，图 14 中表示处理结果，输出图 15 的数据。图 15 是根据跟踪数据输出指令输出

的跟踪数据文件的内容，不仅是文件，向画面上显示的情况也有。

在如上面这样构成的调试器，在利用者通过指令跟踪输出确认程序的执行经过的情况时，通过把没有执行的带条件指令改变显示形式显示，能够把由带条件指令实现的条件执行逻辑易判别的显示出来，能够知道调试对象的程序执行经过，使调试成为可能。

还有，在本实施例，在步骤S1408，把带条件指令没有执行情况的显示方法作为，把没有执行的演算部分用『< >』括起来输出，不过，覆盖取消线或把显示字体的大小变小，把显示字符串设为『未执行』等，其他的方法也可以。

还有，在本实施例，把执行目标环境 302 作为了软件模拟器，不过，作为评价板，采用把跟踪输出处理 S1104 在用硬件来实现的跟踪输出部进行的结构也可以。

第三实施例

接下来对本发明第三实施例，予以说明。

图 16 是本发明第三实施例的软件开发辅助系统的结构图，和第一实施例几乎一样的结构。和第一实施例相比，编译器 202C 及调试器 204C 的处理不一样。

图 17 中，表示本实施例的编译器 202C 的功能方框图。在图 17，1701 是源程序的输入部，输入源程序。

1702 是程序的变换部，把源程序向过渡记述水平表现变换。

1703 是程序最优化部，通过移动过渡记述水平表现的程序中的操作来改变执行顺序或把不要的操作删除来进行最优化并同时生成执行代码。

1704 是数据依存检出部，根据由在程序最优化部 1703 的数据解析得到的数据连锁信息，求得在对象微电脑上的条件标志的生存区间，登录到标志生存区间信息 1709 中。

1705 是标志生存区间信息输出部，把标志生存区间信息 1709 追记到目标代码 203 中然后输出。

1706 是执行代码输出部，把由程序最优化部 1703 生成的执行代码作为目标代码 203 输出。

1707 是调试信息生成部，生成调试需要的识别标记，行号及执行代码相互关联等的调试信息。

1708 是调试信息输出部，把由调试信息生成部 1707 生成的调试信息追记到目标文件 203 中作为调试信息输出。

图 18 中，表示数据依存检出部 1704 的处理流程。在步骤 S1801，从程序最优化部 1703 取得数据的使用定义连锁信息，对全部的使用定义连锁信息重复步骤 S1802，步骤 S1803 的处理。

在步骤 S1802，判定使用定义连锁信息是否与条件标志关联。条件成立的话进入到步骤 S1803，不成立的话进入到步骤 S1801。

在步骤 S1803，和条件标志的种类一起，把条件标志的定义地址以及参照地址作为生存区间，登录到标志生存区间信息 1709 中。

图 19 中，表示数据使用定义连锁信息的一例。1901 是，数据使用定义连锁信息号码。1902 是，该指令代码。1903 是，UD 列表，表示定义在该指令代码中使用的资源的指令代码的列表，列表的要素是数据使用定义连锁信息号码。寄存器 Ra，Rb，Rc，Rd，c0，c1 对各资源都有列表。1904 是，UD 列表，表示使用由该指令代码定义的资源的指令代码的列表，列表的要素是数据使用定义连锁信息号码。寄存器 Ra，Rb，Rc，Rd，c0，c1 按各资源都有列表。

比如，数据使用定义连锁信息号码 2 是，与「cmp Ra= =10，c0」指令代码相关联的，定义在这个指令代码中使用的资源 Ra 的指令是，在数据使用定义连锁信息号码 1 表示的指令代码，还有，使用定义的资源 c0 的指令是，表示着在数据使用定义连锁信息号码 3，4 表示的指令代码。数据使用定义连锁信息是，为了调查数据的依存关系在程序优化部 1703 生成的。

图 20(a)中，表示标志生存区间信息。标志生存区间是指，标志从被标志更新指令定义的时段，到被带条件指令和标志值传送指令参照的时段的区间，这期间，标志的值是一样的。2001 是，标志种类域。2002 是，开始地址域，2003 是结束地址域，表示标志的生存区间。

图 20(b)中，表示标志生存区间信息的例子。2004，2005，2006 各是一例。比如，2004 的例子表示，标志 c0 在地址 0x400 定义，从 0x400 到 0x408 之间被参照。

图 21 是，调试器 204C 的功能方框图，和第一实施例一样，301，302，303，304，305，306 各是，调试器处理部，执行目标环境，目标代码信息，调试信息，指令形式信息，跟踪数据信息。还有，210

是，标志生存区间信息，表示条件标志的生存地址区间。标志生存区间信息 2101，被以包含在由编译器 202C 生成的调试信息中的形式读入调试器。

图 22 是，调试器处理部 301 的处理流程图，是和本发明第一实施例一样的处理流程图。

图 23 中，表示本实施例的步骤 S405 的显示更新处理的处理流程。

在步骤 S2301，在执行目标环境 302，调查调试对象的程序执行状态，在执行停止的情况下，进入到步骤 S2302。在执行没有停止，程序正在执行的情况下，因为显示的更新不必要所以结束处理。在执行停止的情况下，在后续的行，进行把在调试对象程序的哪个地方停止了向程序开发者显示的处理。

在步骤 S2302，取得调试对象程序的停止地址。

接着在步骤 S2303，参照标志生存区间信息 2101，对全部的条件标志，取出包含现在执行停止地址的生存区间信息。

接着在步骤 S2304，把包含前述执行停止地址的生存期间存在的全部条件标志值从执行目标环境 302 取得。

接着在步骤 S2305，求得现在显示对象的地址区间，从那最初的地址到最后的地址重复以后的处理。对全部的显示对象地址的处理完了的话，结束。还有，现在显示对象的地址区间是，比如成为在显示源代码的代码窗体上把现在停止地址作为中心显示前后数行(由窗体的显示行数决定)的情况下的成为对象的区间，这是在进行调试器 204C 显示的地方（调试器处理部 301 内）管理着的信息。

接着在步骤 S2306，从前述显示对象的地址参照调试器内存储的目标代码信息 303，读出对应地址的指令代码。

接着在步骤 S2307，把前述指令代码反汇编，得到与指令代码对应的汇编行字符串。还有，反汇编是，通过参照在调试器内管理的指令形式信息 305 来实现的。

接着在步骤 S2308，从指令代码参照在调试器内管理的指令形式信息 305，判定是否为带条件指令。是带条件指令的情况下，进入到步骤 S2309。不是带条件指令的情况下，进入到步骤 S2313。

在步骤 S2309，对于该条件标志，参照在步骤 S2303 求得的生存区间信息，判断该条件标志对于现在的显示对象地址是否有效。即，

得到在步骤 S2304 取得的条件标志值是否有效。是生存区间内的话，进入到步骤 S2310，不是那样的话进入到步骤 S2314。

在 2310 行，参照在步骤 S2304 取得的条件标志值。

在步骤 S2311，参照前述条件标志值，判定带条件指令是否被执行，执行的情况下，进入到步骤 S2313，没有被执行的情况下，进入到步骤 S2312。

在步骤 S2312，带条件指令没有被执行的情况下，把没有执行的演算部分用浅灰色显示。进一步，显示对象地址是执行停止地址的话，把条件标志部用反转的显示颜色显示。

在步骤 S2313，显示对象地址是执行停止地址的情况下，把对应的汇编行字符串用反转显示颜色显示。显示对象地址不是执行停止地址的情况下，保持通常的颜色显示。

在步骤 S2314，在步骤 S2304 取得的条件标志值不是有效的，不能确定带条件指令是否执行，所以除了通常的显示颜色外，还加上下划线(以未确定的形式)显示。

接着，返回到步骤 S2305，对显示领域的地址区间，进行重复处理。

在图 24 中，表示显示例。还有，在图 24(a)到(d)，把应该用浅灰色显示没有执行的演算部分为了方便用四角括起来表示，实际上，认为不是四角的围线，而是用浅灰色显示的。

如上面那样根据本实施例，在断点，调试对象程序执行停止的情况下，取得那时的条件标志值，且通过使和条件标志的生存区间对应，通过不仅是执行停止地址对其前后的地址，也判定带条件指令的执行/未执行改变显示形式显示，由带条件指令能够把执行的条件执行的逻辑易判别的显示出来，能够知道调试对象程序的执行经过，使调试成为可能。

在本实施例，在步骤 S2312，把带条件指令没有执行情况下的显示方法作为，将没有执行的演算部分以浅灰色显示的，不过，覆盖取消线或把显示字体的大小变小，把显示字符串设为『未执行』等，其他的方法也可以。

还有，第九发明的机器可读记录介质是，这第三实施例的目标文件 203，和以前的目标文件相比追加了标志生存区间信息(1709)。

如上面说明的那样，根据本发明，通过取得带条件指令的执行条

件标志值，在带条件指令实际上能够判别执行的处理内容，能够向利用者把程序的执行过程易理解地显示出来。

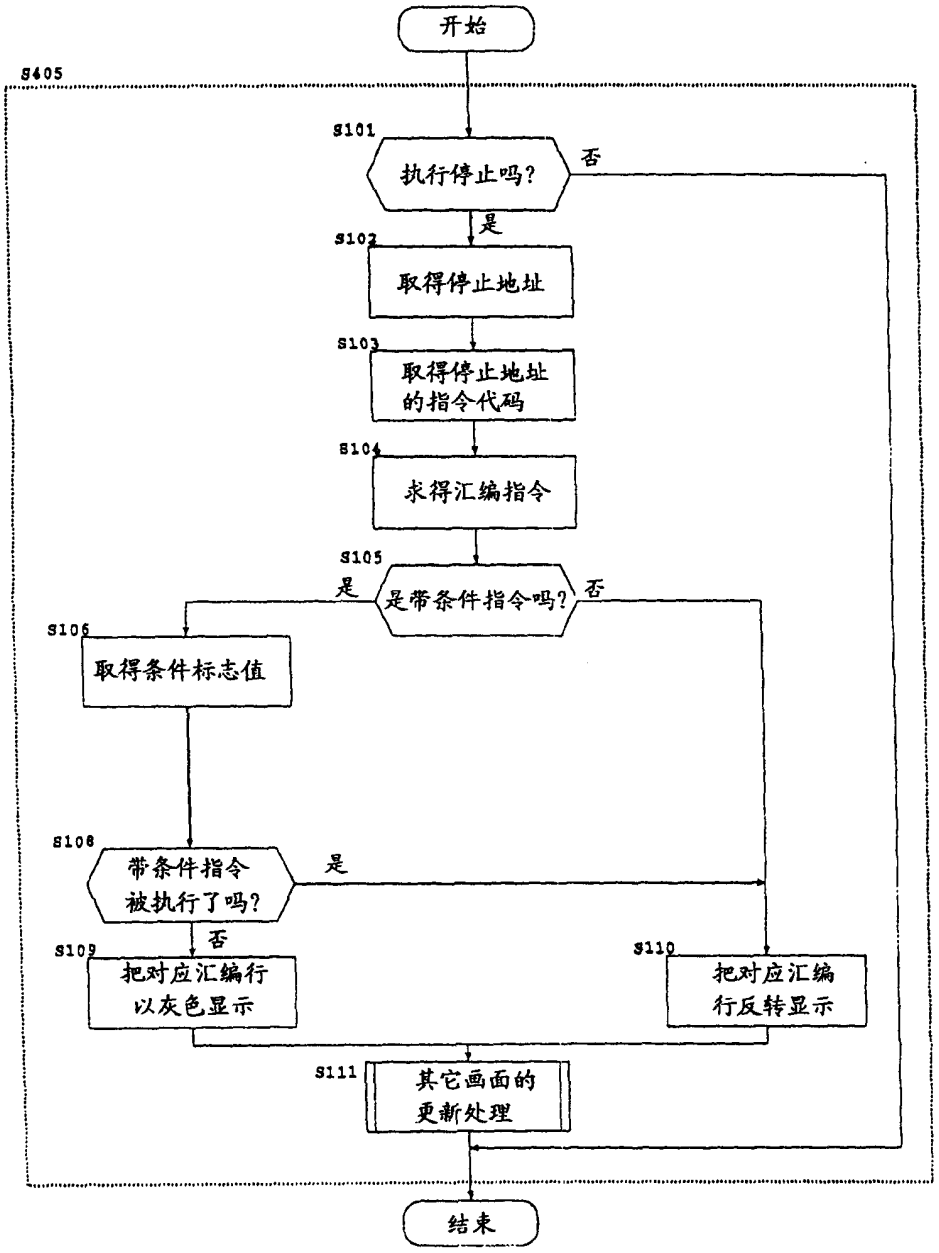

开始
S405
S101
执行停止吗?
否
是
S102
取得停止地址
S103
取得停止地址的指令代码
S104
求得汇编指令
S105
是
是带条件指令吗?
否
S106
取得条件标志值
S108
带条件指令被执行了吗?
是
否
S109
把对应汇编行以灰色显示
S110
把对应汇编行反转显示
S111
其它画面的更新处理
结束

图 1

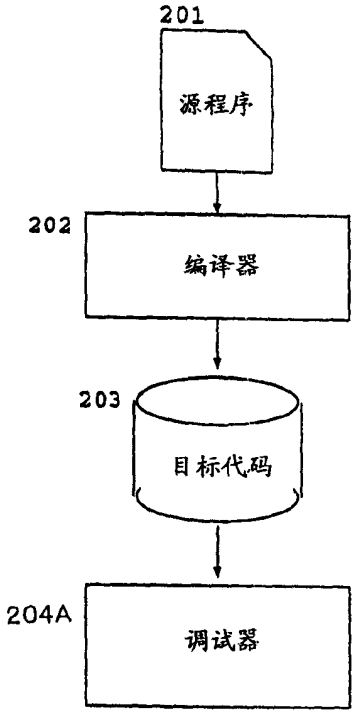


图 2

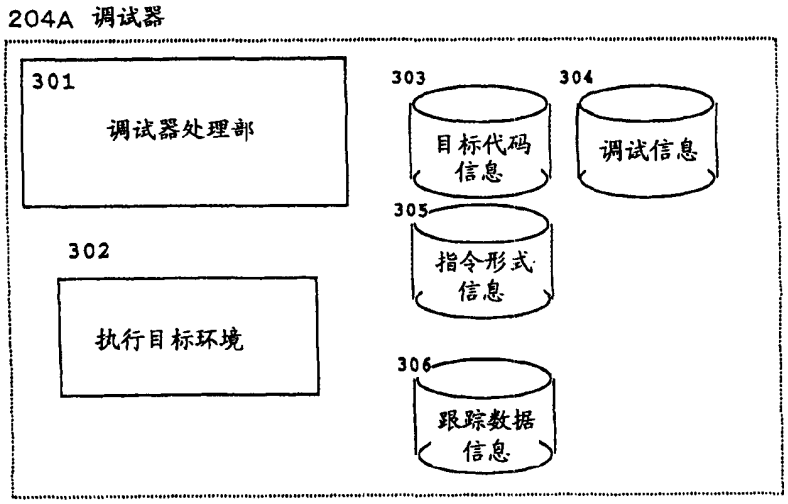

204A 调试器
301
调试器处理部
302
执行目标环境
303
目标代码信息
304
调试信息
305
指令形式信息
306
跟踪数据信息

图 3

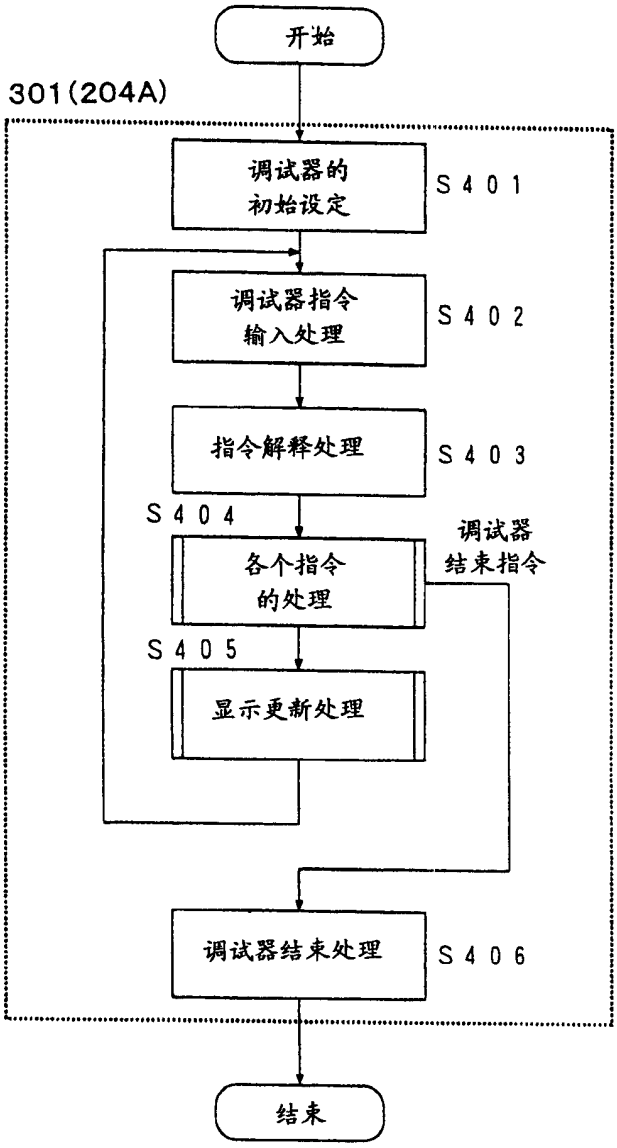

开始
301(204A)
调试器的
初始设定
S401
调试器指令
输入处理
S402
指令解释处理
S403
S404
各个指令
的处理
调试器
结束指令
S405
显示更新处理
调试器结束处理
S406
结束

图 4

```
01:     if (a == 10) {
02:         b = a * a;
03:     } else {
04:         b = c;
05:     }
06:     d = 1;
```

图 5

```
0x400:    cmp Ra==10, c0
0x404:    (c0)  mul Ra, Ra, Rb
0x408:    (!c0) mov Rc, Rb
0x40C:    mov 1, Rd
```

图 6

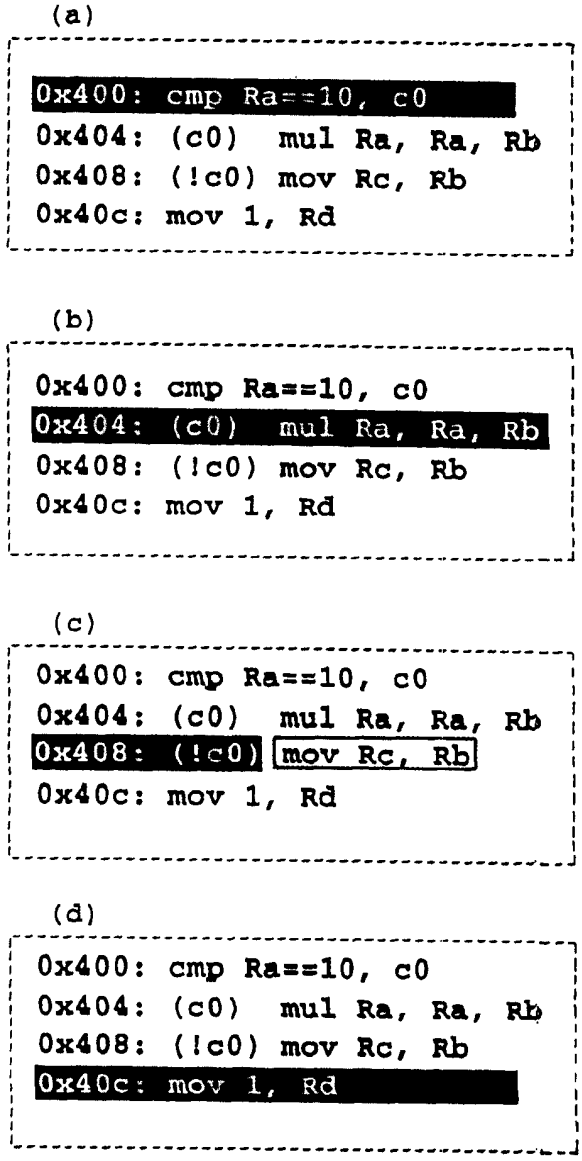

(a)
0x400: cmp Ra==10, c0
0x404: (c0) mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd
(b)
0x400: cmp Ra==10, c0
0x404: (c0) mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd
(c)
0x400: cmp Ra==10, c0
0x404: (c0) mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd
(d)
0x400: cmp Ra==10, c0
0x404: (c0) mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd

图 7

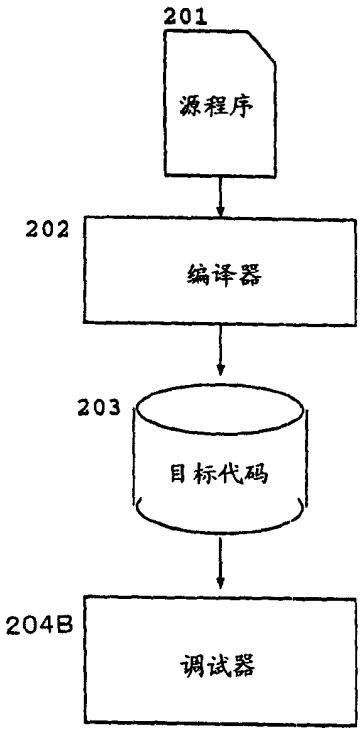


图 8

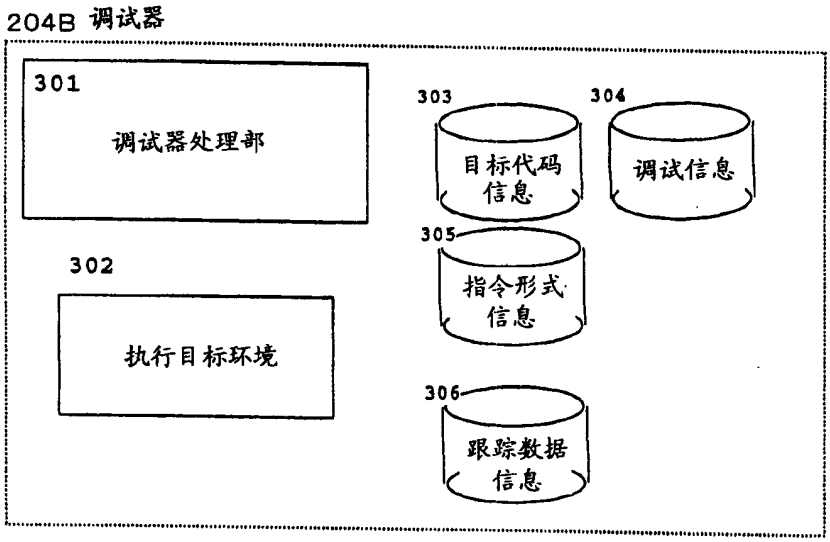

204B 调试器
301
调试器处理部
302
执行目标环境
303
目标代码信息
304
调试信息
305
指令形式信息
306
跟踪数据信息

图 9

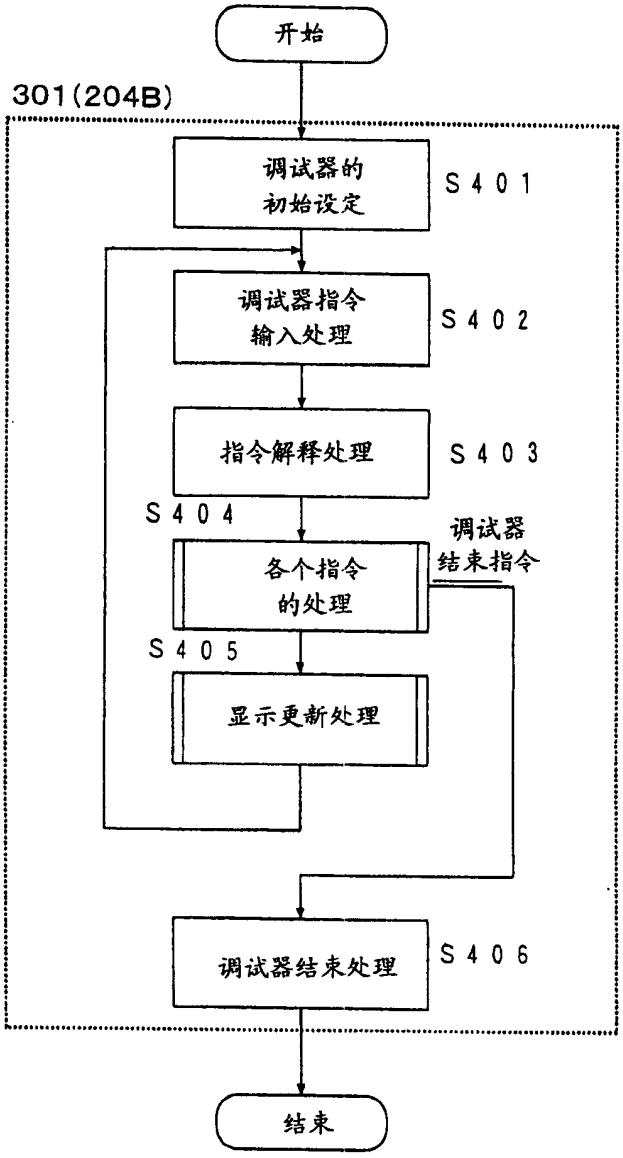

开始
301(204B)
调试器的
初始设定
S401
调试器指令
输入处理
S402
指令解释处理
S403
S404
调试器
结束指令
各个指令
的处理
S405
显示更新处理
调试器结束处理
S406
结束

图 10

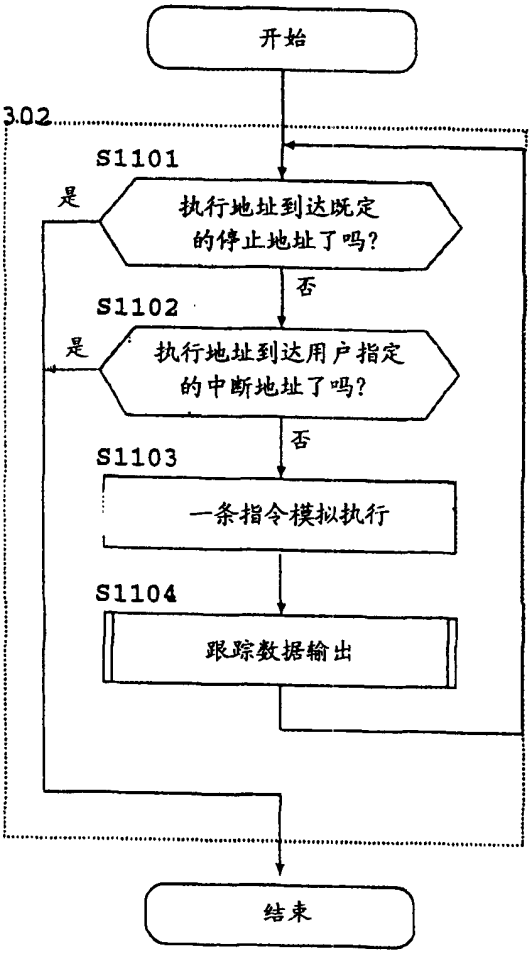

开始
302
S1101
是
执行地址到达既定的停止地址了吗?
否
S1102
是
执行地址到达用户指定的中断地址了吗?
否
S1103
一条指令模拟执行
S1104
跟踪数据输出
结束

图 11

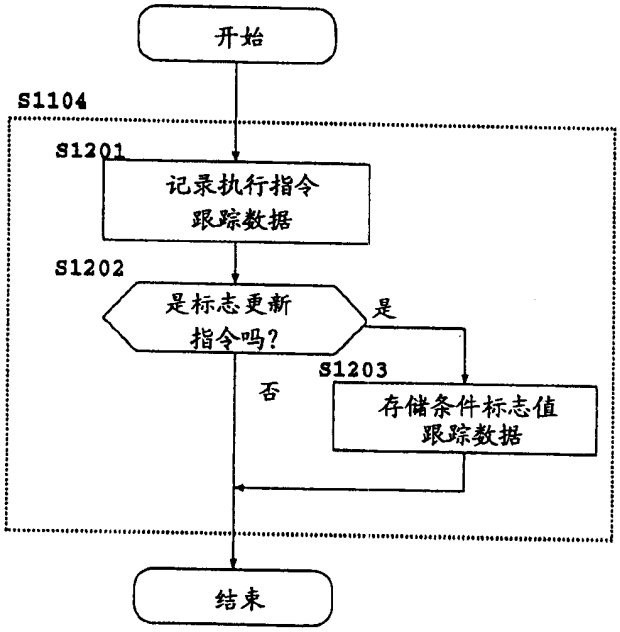

开始
S1104
S1201
记录执行指令
跟踪数据
S1202
是标志更新
指令吗?
是
否
S1203
存储条件标志值
跟踪数据
结束

图 12

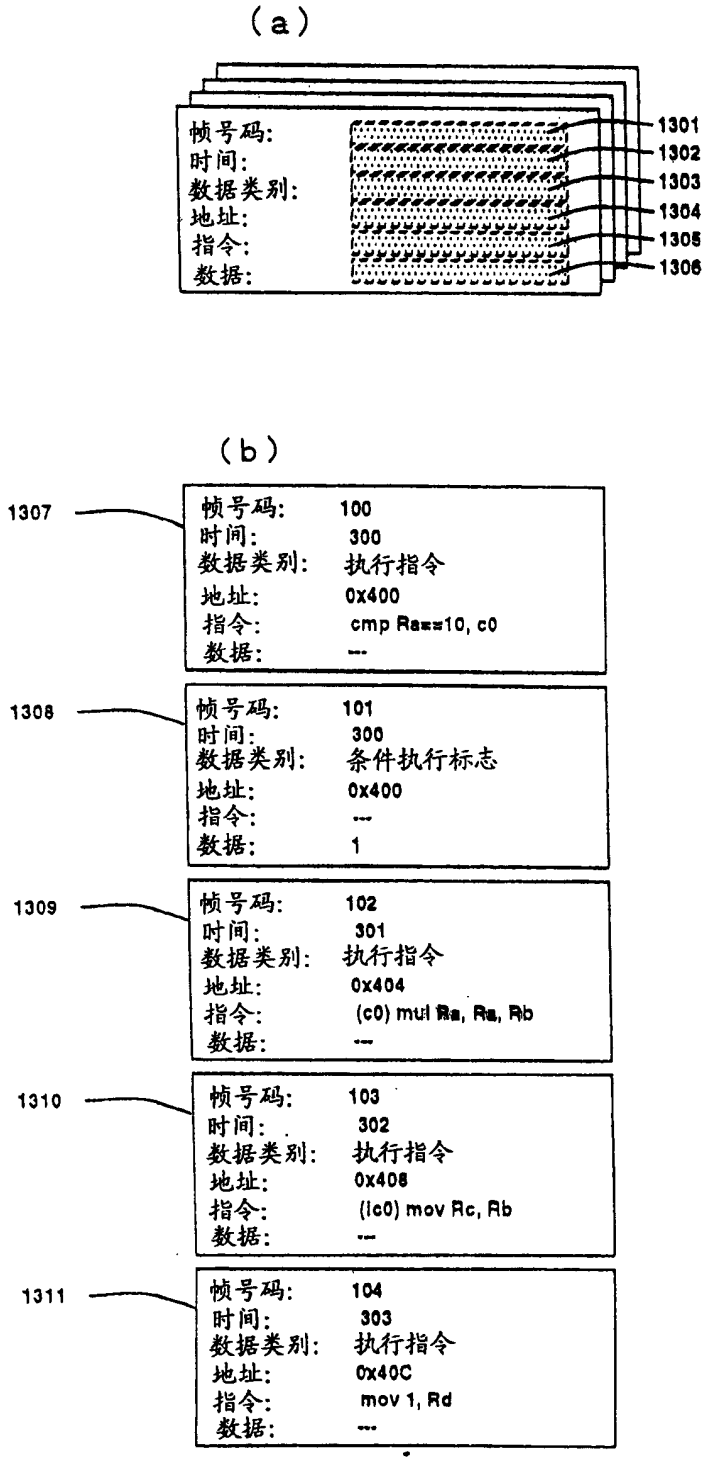

(a)
帧号码:
时间:
数据类别:
地址:
指令:
数据:
1301
1302
1303
1304
1305
1306
(b)
1307
帧号码: 100
时间: 300
数据类别: 执行指令
地址: 0x400
指令: cmp Ra==10, c0
数据: ---
1308
帧号码: 101
时间: 300
数据类别: 条件执行标志
地址: 0x400
指令: ---
数据: 1
1309
帧号码: 102
时间: 301
数据类别: 执行指令
地址: 0x404
指令: (c0) mul Ra, Rb, Rb
数据: ---
1310
帧号码: 103
时间: 302
数据类别: 执行指令
地址: 0x408
指令: (!c0) mov Rc, Rb
数据: ---
1311
帧号码: 104
时间: 303
数据类别: 执行指令
地址: 0x40C
指令: mov 1, Rd
数据: ---

图 13

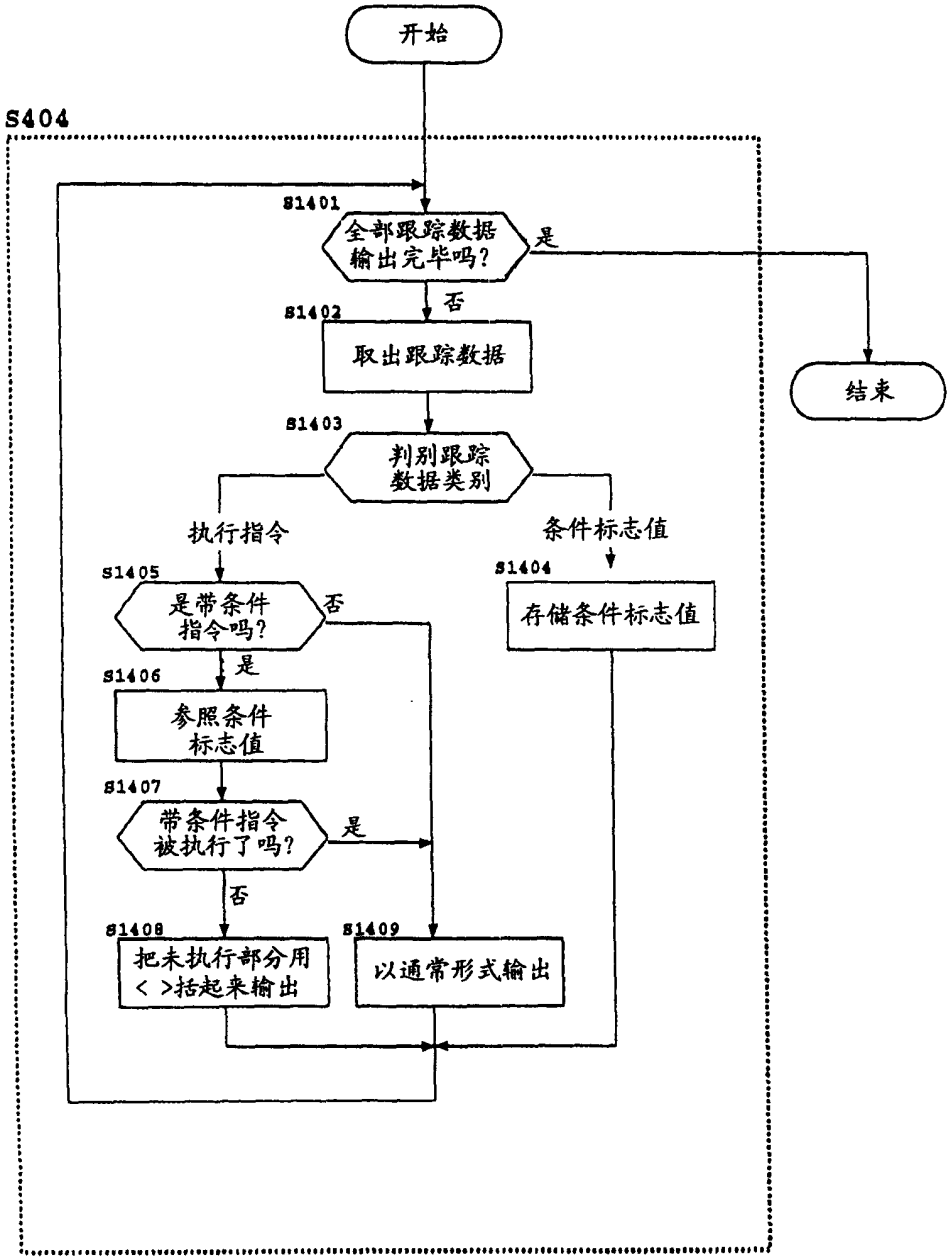

开始
S404
S1401
全部跟踪数据输出完毕吗?
是
否
结束
S1402
取出跟踪数据
S1403
判别跟踪数据类别
执行指令
条件标志值
S1405
是带条件指令吗?
否
是
S1404
存储条件标志值
S1406
参照条件标志值
S1407
带条件指令被执行了吗?
是
否
S1408
把未执行部分用< >括起来输出
S1409
以通常形式输出

图 14

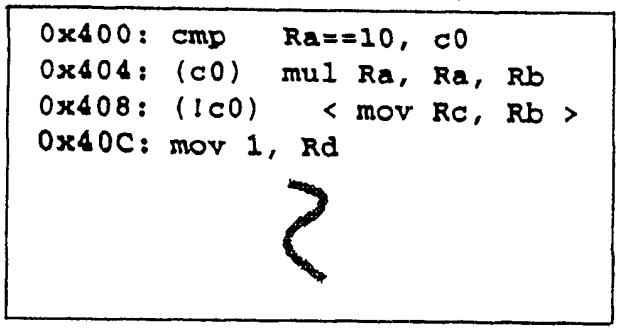

0x400: cmp Ra==10, c0
0x404: (c0) mul Ra, Ra, Rb
0x408: (!c0) < mov Rc, Rb >
0x40C: mov 1, Rd

图 15

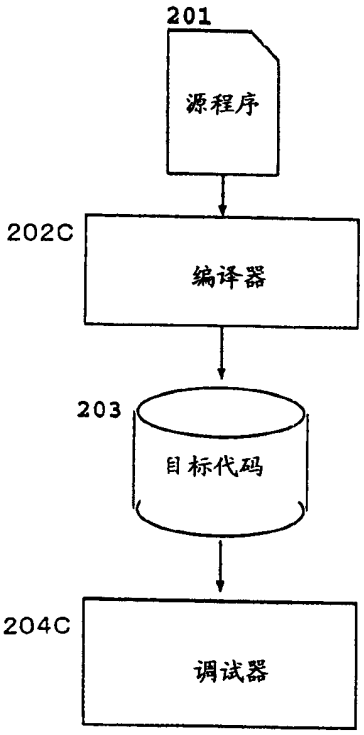


图 16

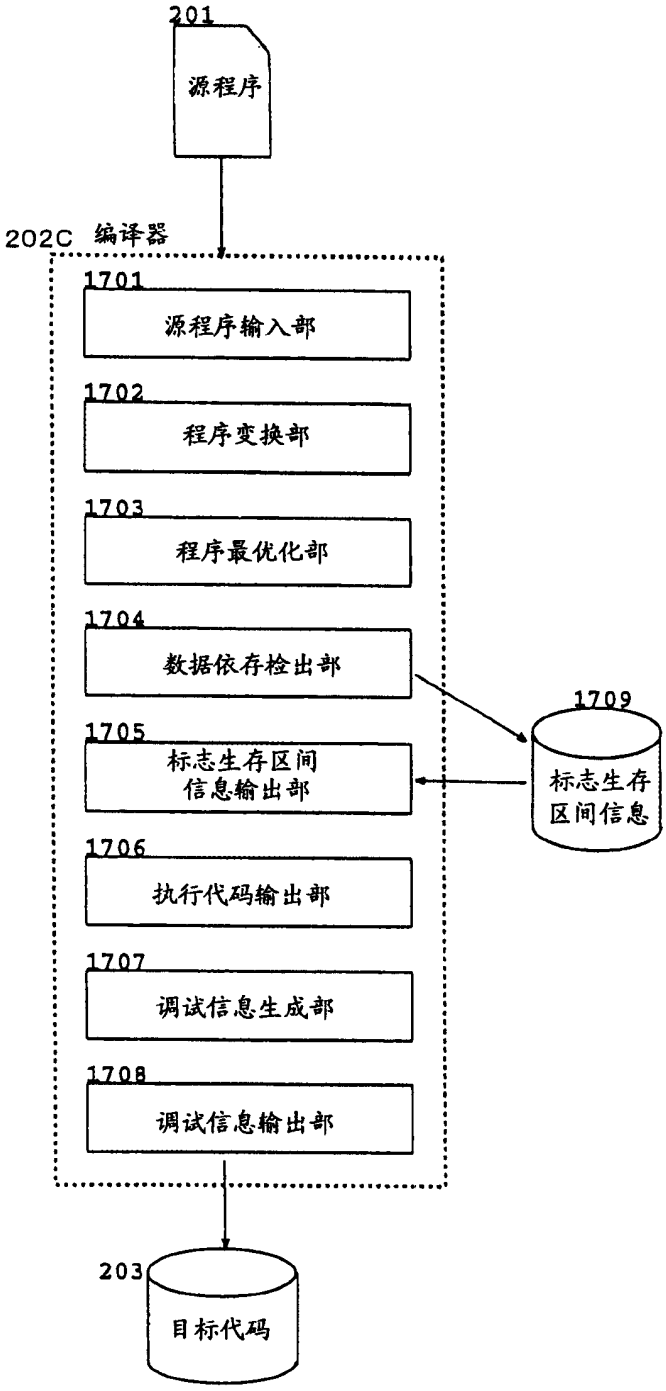

201
源程序
202C 编译器
1701
源程序输入部
1702
程序变换部
1703
程序最优化部
1704
数据依存检出部
1705
标志生存区间
信息输出部
1706
执行代码输出部
1707
调试信息生成部
1708
调试信息输出部
1709
标志生存
区间信息
203
目标代码

图 17

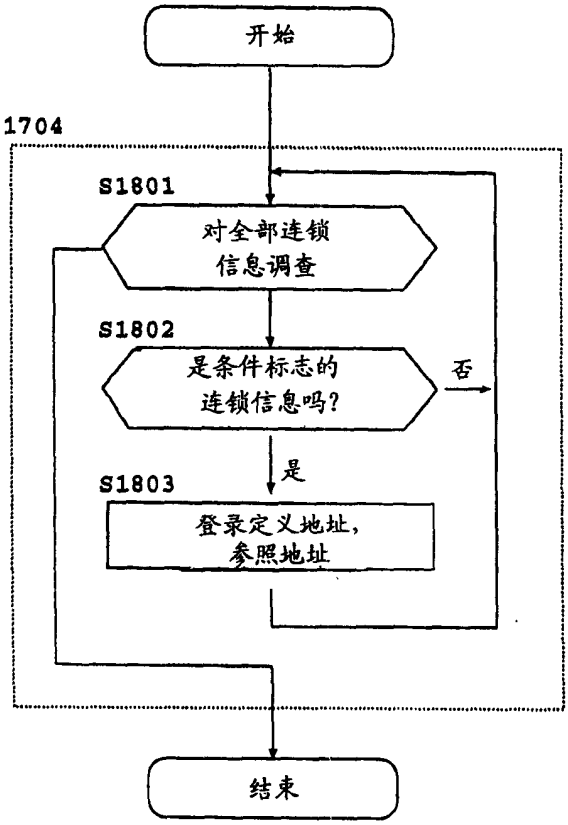
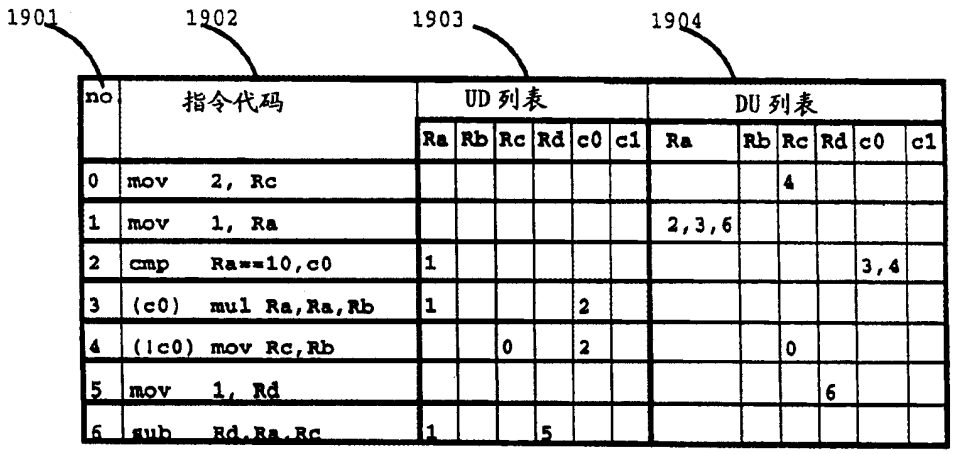

开始
1704
S1801
对全部连锁
信息调查
S1802
是条件标志的
连锁信息吗?
否
是
S1803
登录定义地址,
参照地址
结束

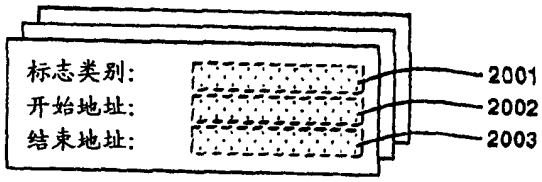
图 18



no	指令代码	UD 列表						DU 列表					
		Ra	Rb	Rc	Rd	c0	c1	Ra	Rb	Rc	Rd	c0	c1
0	mov 2, Rc									4			
1	mov 1, Ra							2,3,6					
2	cmp Ra==10,c0	1										3,4	
3	(c0) mul Ra,Ra,Rb	1				2							
4	(!c0) mov Rc,Rb			0		2				0			
5	mov 1, Rd										6		
6	sub Rd,Ra,Rc	1			5								

图 19

（a）


标志类别:
开始地址:
结束地址:
2001
2002
2003

（b）

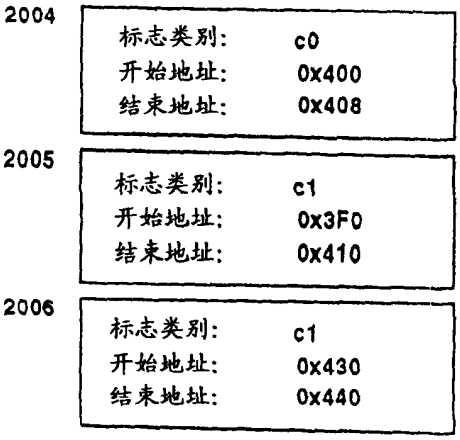

2004
标志类别: c0
开始地址: 0x400
结束地址: 0x408
2005
标志类别: c1
开始地址: 0x3F0
结束地址: 0x410
2006
标志类别: c1
开始地址: 0x430
结束地址: 0x440

图 20

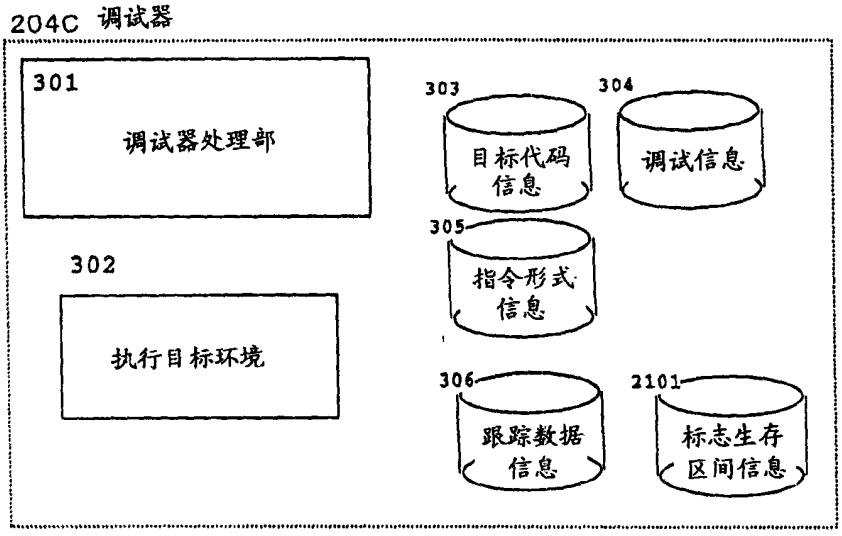

204C 调试器
301
调试器处理部
302
执行目标环境
303
目标代码信息
304
调试信息
305
指令形式信息
306
跟踪数据信息
2101
标志生存区间信息

图 21

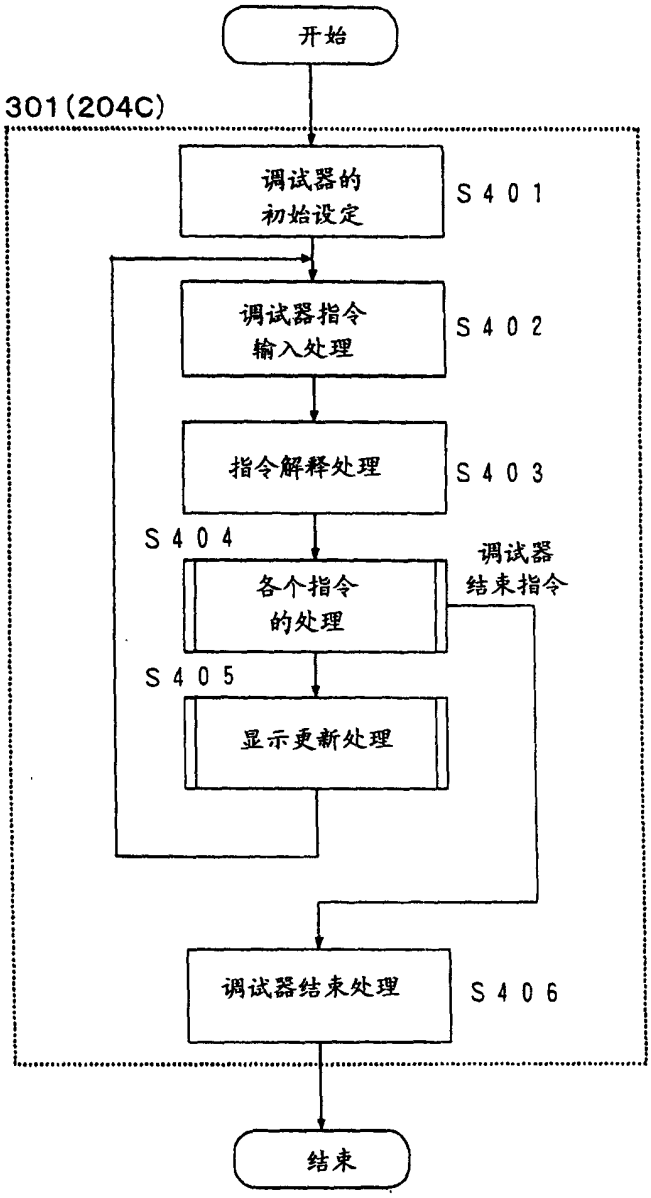

开始
301(204C)
调试器的
初始设定
S401
调试器指令
输入处理
S402
指令解释处理
S403
S404
各个指令
的处理
调试器
结束指令
S405
显示更新处理
调试器结束处理
S406
结束

图 22

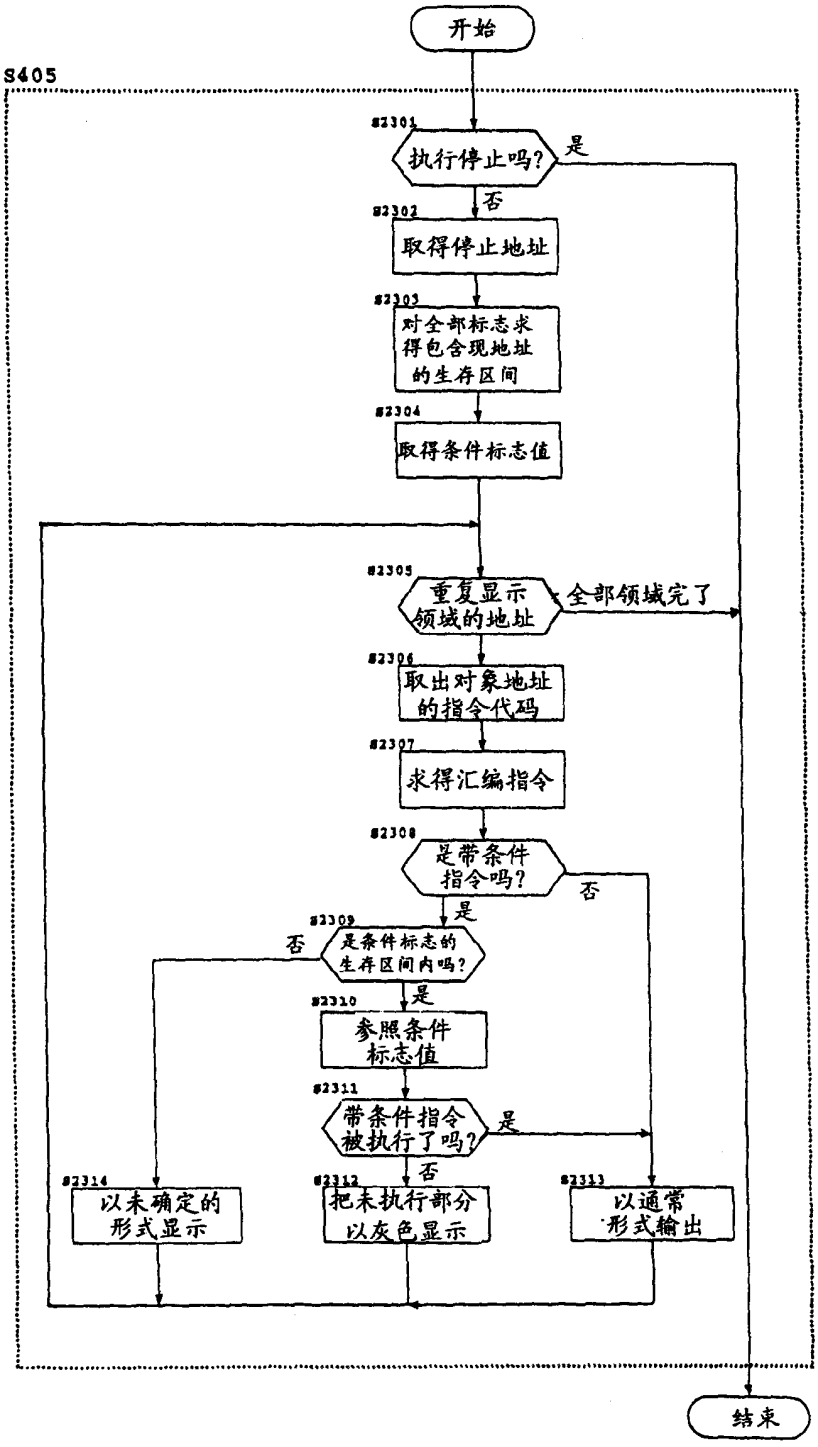

开始
S405
S2301
执行停止吗?
是
否
S2302
取得停止地址
S2303
对全部标志求得包含现地址的生存区间
S2304
取得条件标志值
S2305
重复显示领域的地址
全部领域完了
S2306
取出对象地址的指令代码
S2307
求得汇编指令
S2308
是带条件指令吗?
否
是
S2309
是条件标志的生存区间内吗?
否
是
S2310
参照条件标志值
S2311
带条件指令被执行了吗?
是
否
S2314
以未确定的形式显示
S2312
把未执行部分以灰色显示
S2313
以通常形式输出
结束

图 23

(a)c0=1，停止地址=0x404 的情况下

```
0x400: cmp Ra==10, c0
0x404: (c0)  mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd
```

(b)c0=1，停止地址=0x408 的情况下

```
0x400: cmp Ra==10, c0
0x404: (c0)  mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd
```

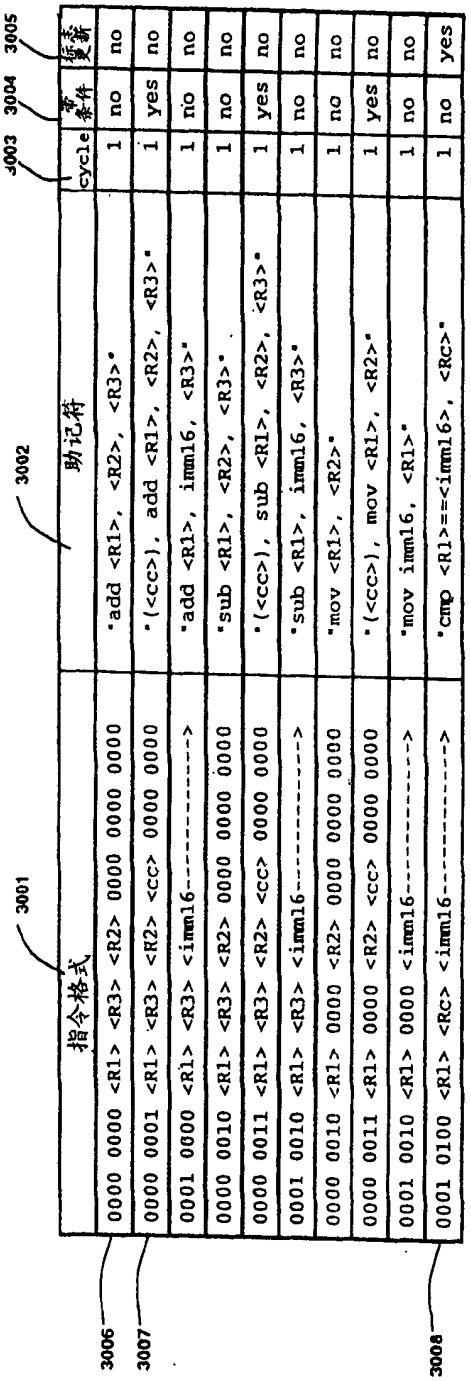
(c)c0=0，停止地址=0x404 的情况下

```
0x400: cmp Ra==10, c0
0x404: (c0)  mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd
```

(d)c0=0，停止地址=0x408 的情况下

```
0x400: cmp Ra==10, c0
0x404: (c0)  mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd
```

图 24



指令格式	助记符	cycle	条件	标志更新
0000 0000 <R1> <R3> <R2> 0000 0000 0000	"add <R1>, <R2>, <R3>"	1	no	no
0000 0001 <R1> <R3> <R2> <cc> 0000 0000	"(<cc>), add <R1>, <R2>, <R3>"	1	yes	no
0001 0000 <R1> <R3> <imm16>---------->	"add <R1>, imm16, <R3>"	1	no	no
0000 0010 <R1> <R3> <R2> 0000 0000 0000	"sub <R1>, <R2>, <R3>"	1	no	no
0000 0011 <R1> <R3> <R2> <cc> 0000 0000	"(<cc>), sub <R1>, <R2>, <R3>"	1	yes	no
0001 0010 <R1> <R3> <imm16>---------->	"sub <R1>, imm16, <R3>"	1	no	no
0000 0010 <R1> 0000 <R2> 0000 0000 0000	"mov <R1>, <R2>"	1	no	no
0000 0011 <R1> 0000 <R2> <cc> 0000 0000	"(<cc>), mov <R1>, <R2>"	1	yes	no
0001 0010 <R1> 0000 <imm16>---------->	"mov imm16, <R1>"	1	no	no
0001 0100 <R1> <Rc> <imm16>---------->	"cmp <R1>==<imm16>, <Rc>"	1	no	yes

图 25

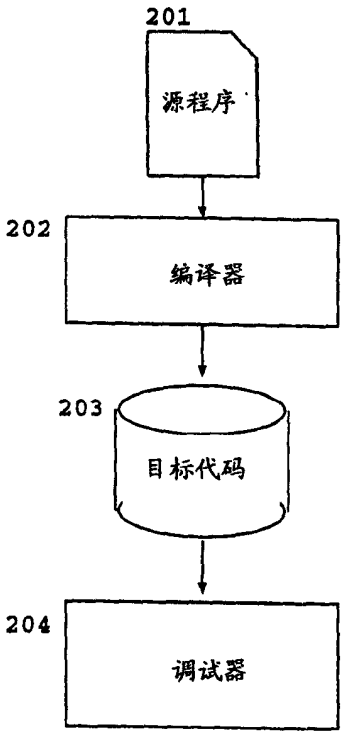


图 26

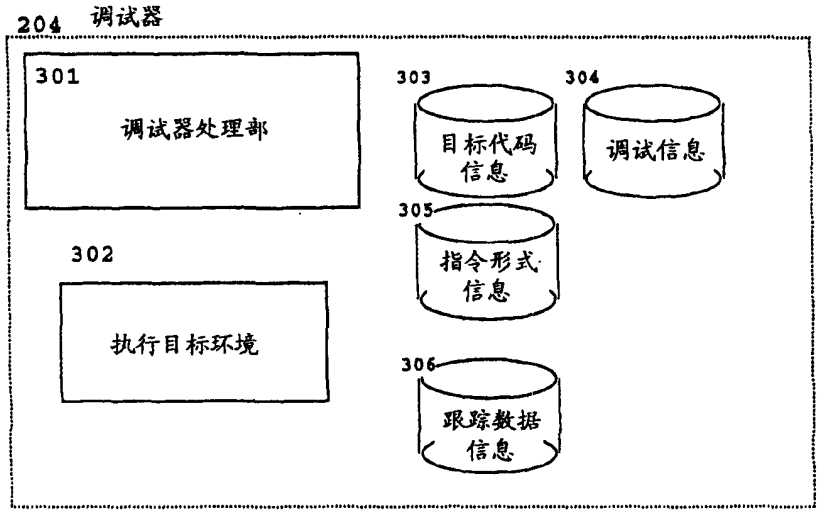

204 调试器
301
调试器处理部
302
执行目标环境
303
目标代码信息
304
调试信息
305
指令形式信息
306
跟踪数据信息

图 27

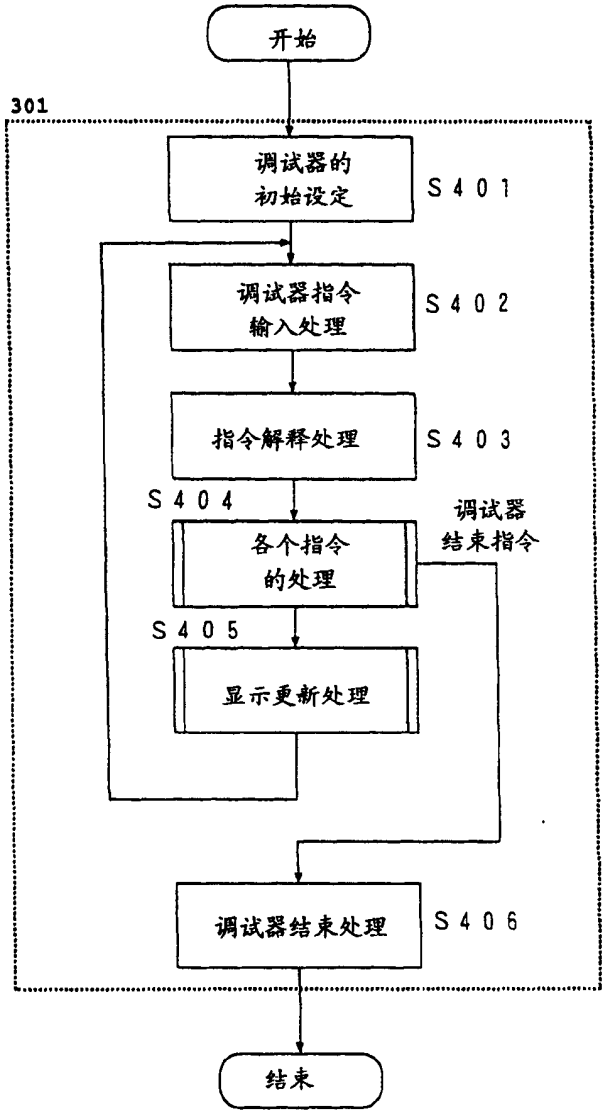

开始
301
调试器的
初始设定
S401
调试器指令
输入处理
S402
指令解释处理
S403
S404
各个指令
的处理
调试器
结束指令
S405
显示更新处理
调试器结束处理
S406
结束

图 28

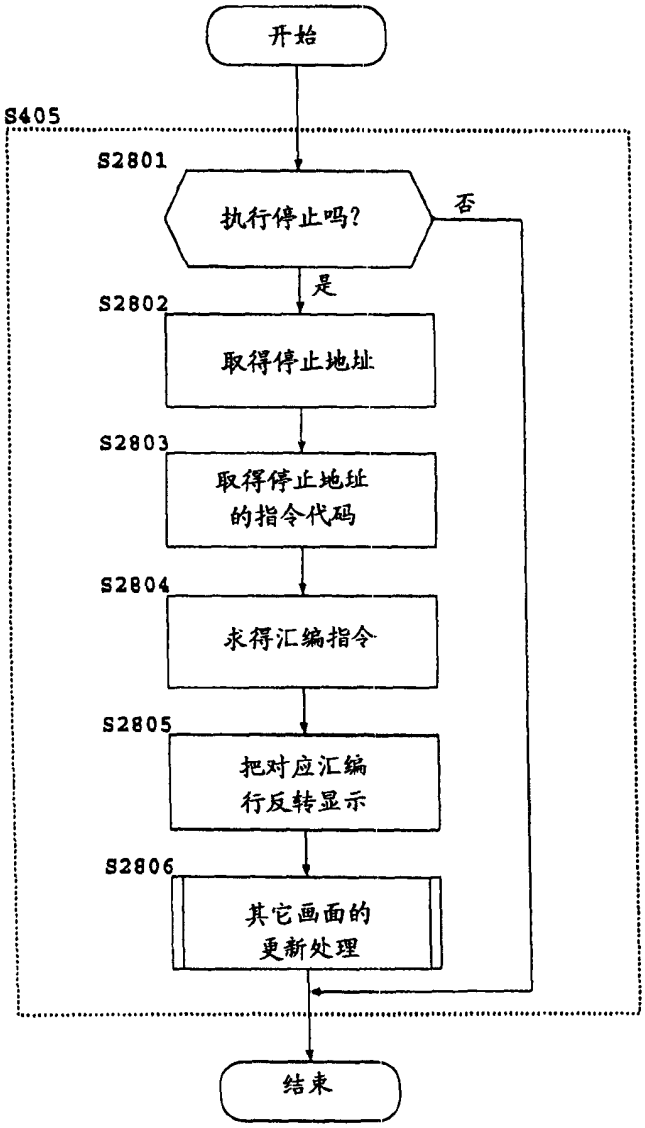

开始
S405
S2801
执行停止吗?
否
是
S2802
取得停止地址
S2803
取得停止地址
的指令代码
S2804
求得汇编指令
S2805
把对应汇编
行反转显示
S2806
其它画面的
更新处理
结束

图 29

(a)

```
0x400: cmp Ra==10, c0
0x404: (c0)  mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd
```

(b)

```
0x400: cmp Ra==10, c0
0x404: (c0)  mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd
```

(c)

```
0x400: cmp Ra==10, c0
0x404: (c0)  mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd
```

(d)

```
0x400: cmp Ra==10, c0
0x404: (c0)  mul Ra, Ra, Rb
0x408: (!c0) mov Rc, Rb
0x40c: mov 1, Rd
```

图 30