[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/45 (2006.01)

[12] 发明专利说明书

专利号 ZL 200510108602.X

[45] 授权公告日 2009 年 4 月 8 日

[11] 授权公告号 CN 100476735C

[22] 申请日 2005.10.8
[21] 申请号 200510108602.X
[30] 优先权
[32] 2004.10.8 [33] JP [31] 296287/2004
[73] 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 小川一 坂田俊幸
[56] 参考文献
US2003/0140336A1 2003.7.24
CN1374585A 2002.10.16
US6064818A 2000.5.16
US6308322B1 2001.10.23
US5535392A 1996.7.9
US6412107B1 2002.6.25
CN1512334A 2004.7.14
审查员 李 强
[74] 专利代理机构 永新专利商标代理有限公司
代理人 胡建新

权利要求书 7 页 说明书 32 页 附图 39 页

[54] 发明名称

程序处理装置

[57] 摘要

一种程序处理装置，即使用户未向编译器提供附注所代表的提示信息，也能够通过自动地插入提示信息来修改源程序，以便能够进行良好的最优化，该装置是自动在程序中插入提供给编译器的提示信息的程序处理装置(102)，具有：语法分析部(104)，分析程序(100)的语法，并生成分析信息；提示信息插入部，根据所述分析信息，输出把提供给编译器的逻辑上不矛盾的提示信息附加到程序(100)中的程序(101)。

```
#pragma _software_pipelining
for (i = 0; i < 100; i++) {
    x[i] = y[i] + z[i];
}
```

1. 一种程序处理装置，其特征在于，接受第 1 程序，输出向所述第 1 程序附加了提供给编译器的逻辑上不矛盾的提示信息的第 2 程序，该程序处理装置具有：

语法分析单元，分析所述第 1 程序的语法并生成分析信息；

提示信息附加单元，根据所述分析信息，输出向所述第 1 程序附加了提供给编译器的逻辑上不矛盾的提示信息的所述第 2 程序。

2. 根据权利要求 1 所述的程序处理装置，其特征在于，

所述语法分析单元静态分析所述第 1 程序的语法，并生成所述分析信息，

所述提示信息附加单元根据所述分析信息，输出把可以通过程序的静态分析得到的静态信息作为所述提示信息附加到所述第 1 程序中的所述第 2 程序。

3. 根据权利要求 2 所述的程序处理装置，其特征在于，

所述语法分析单元分析所述第 1 程序中包含的循环处理的反复次数，并生成包括该反复次数的所述分析信息，

所述提示信息附加单元根据所述分析信息，输出把与循环处理的反复次数相关的提示信息附加到所述第 1 程序中的所述第 2 程序。

4. 根据权利要求 3 所述的程序处理装置，其特征在于，所述提示信息附加单元根据所述分析信息，输出把所述第 1 程序中包含的循环处理的反复次数的最小次数或最大次数作为提示信息附加的所述第 2 程序。

5. 根据权利要求 3 所述的程序处理装置，其特征在于，

所述提示信息附加单元具有：

偶数判断部，根据所述分析信息，判断所述第 1 程序中包含的循环处理的反复次数是否一定是偶数；

程序输出部，在所述偶数判断部的判断结果为肯定时，输出把该循环处理的反复次数一定是偶数作为提示信息附加到所述第 1 程序中的所述第 2 程序。

6. 根据权利要求 3 所述的程序处理装置，其特征在于，

所述提示信息附加单元具有：

奇数判断部，根据所述分析信息，判断所述第 1 程序中包含的循环处理的反复次数是否一定是奇数；

程序输出部，在所述奇数判断部的判断结果为肯定时，输出把该循环处理的反复次数一定是奇数作为提示信息附加到所述第 1 程序中的所述第 2 程序。

7. 根据权利要求 2 所述的程序处理装置，其特征在于，

所述语法分析单元分析所述第 1 程序中包含的指针变量所指的数据的配置位置，并生成包括该数据的配置位置的所述分析信息，

所述提示信息附加单元根据所述分析信息，输出把与数据的配置位置相关的提示信息附加到所述第 1 程序中的所述第 2 程序。

8. 根据权利要求 7 所述的程序处理装置，其特征在于，

所述提示信息附加单元具有：

定位判断部，根据所述分析信息，判断所述第 1 程序中包含的指针变量所指的数据是否按照规定值被定位；

程序输出部，在所述定位判断部的判断结果为肯定时，输出把该数据和该数据按照规定值被定位作为提示信息附加到所述第 1 程序中的所述第 2 程序。

9. 根据权利要求 2 所述的程序处理装置，其特征在于，

所述语法分析单元分析所述第 1 程序中包含的指针变量所存取的区域，并生成包括分析结果的所述分析信息，

所述提示信息附加单元根据所述分析信息，输出把与指针变量存取的区域相关的提示信息附加到所述第 1 程序中的所述第 2 程序。

10. 根据权利要求 9 所述的程序处理装置，其特征在于，

所述提示信息附加单元具有：

重复判断部，根据所述分析信息，判断所述第 1 程序中包含的指针变量所存取的区域是否具有与其他指针变量所存取的区域之间重复的区域；

程序输出部，在所述重复判断部的判断结果为否定时，输出把提示信息附加到所述第 1 程序中的所述第 2 程序，该提示信息是表示所述第 1 程序中包含的所述指针变量所存取的区域不具有与所述其他指针变量所存取的区域之间重复的区域的信息。

11. 根据权利要求 2 所述的程序处理装置，其特征在于，

所述语法分析单元对从所述第 1 程序中包含的变量读出数据或向该变量写入数据进行分析，并生成包括分析结果的所述分析信息，

所述提示信息附加单元根据所述分析结果，输出把与从变量读出数据或向变量写入数据相关的提示信息附加到所述第 1 程序中的所述第 2 程序。

12. 根据权利要求 11 所述的程序处理装置，其特征在于，

所述提示信息附加单元具有：

写入判断部，根据所述分析信息，判断向所述第 1 程序中包含的变量的存取是否是从数据的写入开始的；

程序输出部，在所述写入判断部的判断结果为肯定时，把该变量和向该变量的存取是从数据的写入开始作为提示信息，在所述第 1 程序中的、进行向所述变量写入数据的位置的前面插入所述提示信息，生成所述第 2 程序并输出。

13. 根据权利要求 2 所述的程序处理装置，其特征在于，

所述语法分析单元分析所述第 1 程序中包含的分支条件的静态成立频次，并生成包括分析结果的所述分析信息，

所述提示信息附加单元根据所述分析结果，输出把与分支条件的

成立频次相关的提示信息附加到所述第 1 程序中的所述第 2 程序。

14. 根据权利要求 13 所述的程序处理装置，其特征在于，

所述提示信息附加单元具有：

可能性判断部，根据所述分析信息，判断所述第 1 程序中包含的分支条件成立的可能性；

在可以判断为所述可能性判断部的判断结果为肯定的概率高的情况下，输出把表示所述分支条件成立的可能性大的提示信息附加到所述第 1 程序中的所述第 2 程序。

15. 根据权利要求 13 所述的程序处理装置，其特征在于，

所述提示信息附加单元具有：

可能性判断部，根据所述分析信息，判断所述第 1 程序中包含的分支条件成立的可能性；

在可以判断为所述可能性判断部的判断结果为否定的概率高的情况下，输出把表示所述分支条件不成立的可能性大的提示信息附加到所述第 1 程序中的所述第 2 程序。

16. 根据权利要求 1 所述的程序处理装置，其特征在于，所述语法分析单元静态分析所述第 1 程序中的函数间的调用关系，并生成所述分析信息。

17. 根据权利要求 1 所述的程序处理装置，其特征在于，所述提示信息附加单元输出编译选择作为所述提示信息，该编译选择是在起动编译器时用户对成为编译对象的所述第 1 程序所指定的。

18. 根据权利要求 17 所述的程序处理装置，其特征在于，所述编译选择是指示所述第 1 程序中包含的指针变量所指的数据的配置方法的编译选择。

19. 根据权利要求 1 所述的程序处理装置，其特征在于，所述第 1 程序包括第 1 提示信息，

所述提示信息附加单元根据所述分析信息，输出把第 1 程序中的

所述第 1 提示信息订正为第 2 提示信息的第 2 程序。

20. 根据权利要求 19 所述的程序处理装置，其特征在于，所述第 1 提示信息是可以通过所述第 1 程序的静态分析得到的静态信息，

所述第2提示信息是指示所述第2程序的最优化处理方法的最优化指示信息。

21. 根据权利要求 20 所述的程序处理装置，其特征在于，所述第 1 提示信息是与所述第 1 程序中的循环处理的反复次数相关的信息，

所述第 2 提示信息是对所述循环处理指示基于软件流水线操作的最优化的信息。

22. 根据权利要求 20 所述的程序处理装置，其特征在于，所述第1提示信息是与所述第1程序中包含的指针变量所指的数据的配置位置相关的信息，

所述第 2 提示信息是对该信息指示基于成对命令的生成的最优化的信息。

23. 根据权利要求 1 所述的程序处理装置，其特征在于，所述提示信息是指示编译器的程序的最优化处理方法的信息。

24. 根据权利要求 23 所述的程序处理装置，其特征在于，

所述语法分析单元分析所述第 1 程序中包含的循环处理的反复次数，并生成包括该反复次数的分析信息，

所述提示信息附加单元根据所述分析信息，输出把指示循环展开的循环处理的最优化的提示信息附加到所述第1程序中的所述第2程序。

25. 根据权利要求 24 所述的程序处理装置，其特征在于，

所述提示信息附加单元具有：

条件判断部，在循环展开时展开的最低反复次数为 2 时，根据所述分析信息，判断所述第 1 程序中包含的循环处理的反复次数是否满

足 2 次以上、并且只是奇数次数或偶数次数中之一这一条件，

程序输出部，输出对所述第 1 程序中包含的、满足所述条件的循环处理附加了指示循环展开的循环处理的最优化的提示信息的所述第 2 程序。

26. 根据权利要求 23 所述的程序处理装置，其特征在于，所述语法分析单元分析所述第 1 程序中包含的循环处理的反复次数，并生成包括该反复次数的分析信息，

所述提示信息附加单元根据所述分析信息，输出把指示软件流水线操作的循环处理的最优化的提示信息附加到所述第 1 程序中的所述第 2 程序。

27. 根据权利要求 26 所述的程序处理装置，其特征在于，

所述提示信息附加单元具有：

反复次数判断部，在同时执行的最低反复次数为 n 时，根据所述分析信息，判断对应所述提示信息的循环处理的反复次数是否为 n 次以上，n 为 2 以上的整数；

程序输出部，输出对所述第 1 程序中包含的、所述反复次数判断部的判断结果为肯定的循环处理附加了指示软件流水线操作的循环处理的最优化的提示信息的所述第 2 程序。

28. 根据权利要求 23 所述的程序处理装置，其特征在于，

所述语法分析单元分析所述第 1 程序中包含的数据的定位值，并生成包括分析结果的所述分析信息，

所述提示信息附加单元具有：

条件判断部，根据所述分析信息，判断第 1 程序中包含的数据的定位值是否满足该数据的类型的尺寸的 2 倍以上这一条件；

程序输出部，输出对所述第 1 程序中包含的满足所述条件的数据附加了指示基于成对命令的生成的数据存取的最优化的提示信息的所述第 2 程序。

29. 根据权利要求 23 所述的程序处理装置，其特征在于，所述提示信息是指示变量和高速缓存存储器的控制处理的最优化的信息。

30. 根据权利要求 29 所述的程序处理装置，其特征在于，

所述语法分析单元对从所述第 1 程序中包含的变量读出数据或向该变量写入数据进行分析，并生成包括分析结果的所述分析信息，

所述提示信息附加单元具有：

写入判断部，根据所述分析信息，判断向所述第 1 程序中包含的变量的存取是否是从数据的写入开始的；

程序输出部，在所述写入判断部的判断结果为肯定时，把该变量和在所述高速缓存存储器上确保用于存储该变量的值的区域作为提示信息，在所述第 1 程序中的、向所述变量写入数据的位置的前面插入所述提示信息，生成所述第 2 程序并输出。

31. 根据权利要求 1 所述的程序处理装置，其特征在于，所述第 1 和第 2 程序是利用 C 语言或 C＋＋语言记述的，

所述提示信息是在所述程序中可以记述的对编译器的指示即附注记述。

程序处理装置

技术领域

本发明涉及一种修改利用 C 语言等高级语言记述的源程序的程序处理装置，特别涉及通过向把源程序转换为机械语言程序的编译器插入所提供的提示信息，来修改源程序的程序处理装置。

背景技术

近年来，伴随媒介处理应用的增加、多样化，该应用的开发工时增加，在媒介处理领域也需要开发使用高级语言的应用。因此，进行了实现使用高级语言的媒介处理应用开发的尝试。此时，用户期望使用高级语言的开发也能够进行更精密的调整。为此，需要具体控制编译器进行的最优化战略。

该最优化战略的控制方法大致分为两种。

作为一种控制方法，可以列举直接对编译器进行与某种最优化相关的指示的控制方法。作为其他控制方法，可以列举对编译器表示程序整体的静态信息，由此进行编译器的最优化的支持的控制方法。

并且，作为具体实现这两种控制方法的方法之一，可以列举对于使用附注（pragma）的编译器的指示。所述“附注”是依赖于语言处理系统的记述，用于对编译器提供某种信息。

以下表示附注的示例。图 1 是表示包括直接进行与最优化相关的指示的附注的源程序的一例图。“#pragma_software_pipelining”是为了对此后记述的循环处理实施基于软件流水线操作的最优化而进行指示的附注，由用户进行记述。编译器根据该附注的指示，对该循环处理实施基于软件流水线操作的最优化。所述软件流水线操作是同

时执行几个不同的迭代（反复处理）的技术。

图 2 是表示包括对编译器表示程序整体的静态信息的附注的源程序的一例图。“#pragma_min_iteration＝5”是用户保证对此后记述的循环处理执行至少 5 次循环处理的反复的附注。编译器根据该附注，例如判断软件流水线操作的最优化是否可行，如果可行则进行这种最优化。

关于这种附注，在日本专利特开 2004－38597 号公报中有详细说明。

但是，在上述两种控制方法中，用户必须记述附注以便在逻辑上不与实际的程序产生矛盾。为了避免生成错误的机械语言程序，用户只能附加自己可以分析的范围内的消极的附注，存在不能进行良好的最优化的问题。

并且，用户必须自己分析程序中的多个模块，并在程序中记述附注，但由于模块调用的烦杂程度，用户有可能在程序中附加矛盾的附注。在这种情况下，编译器根据错误的附注指示，进行中间代码的最优化等。因此，存在编译器生成错误的机械语言程序的问题。

发明内容

本发明就是为了解决上述课题而提出的，其第 1 目的在于提供一种程序处理装置，即使用户未向编译器提供附注所代表的提示信息，也能够通过自动插入提示信息来修改源程序，以便能够进行良好的最优化。

并且，本发明的第 2 目的在于提供一种程序处理装置，即使用户积极地向编译器提供了附注所代表的提示信息，也能够进行提示信息的检查并修改源程序，以便能够进行良好的最优化。

另外，本发明的第 3 目的在于提供一种程序处理装置，可以检查附注所代表的提示信息并修改源程序，以使编译器不生成错误的机械

语言程序。

为了达到上述目的，本发明涉及的程序处理装置，接受第 1 程序，输出向所述第 1 程序附加了提供给编译器的逻辑上不矛盾的提示信息的第 2 程序，所述程序处理装置具有：语法分析单元，分析第 1 程序的语法并生成分析信息；提示信息附加单元，根据所述分析信息，输出向所述第 1 程序附加了提供给编译器的逻辑上不矛盾的提示信息的第 2 程序。

根据分析信息，提供给编译器的逻辑上不矛盾的提示信息被附加到第 1 程序中。因此，即使用户未向编译器提供附注所代表的提示信息，也能够自动地插入提示信息以能够进行良好的最优化，从而可以修改源程序。另外，这种提示信息包括以下两种：对编译器直接进行与某种最优化相关的指示的提示信息，和通过对编译器表示程序整体的静态信息来进行编译器的最优化的支持的提示信息。

优选所述语法分析单元静态分析所述第 1 程序的语法，并生成所述分析信息，优选所述提示信息附加单元根据所述分析信息，输出把通过程序的静态分析得到的静态信息作为所述提示信息附加到所述第 1 程序中的所述第 2 程序。并且，所述语法分析单元也可以分析所述第 1 程序中包含的循环处理的反复次数，并生成包括该反复次数的所述分析信息，所述提示信息附加单元根据所述分析信息，输出把与循环处理的反复次数相关的提示信息附加到所述第 1 程序中的所述第 2 程序。例如，所述提示信息附加单元根据所述分析信息，输出把所述第 1 程序中包含的循环处理的反复次数的最小次数或最大次数作为提示信息附加的所述第 2 程序。

可以在程序中自动插入与循环处理的反复次数相关的提示信息。例如，通过把循环处理的反复次数的最小次数作为提示信息进行附加，根据利用该提示信息指定的循环处理的最低反复次数，进行可否对该循环处理适用软件流水线操作的判定，如果可以，则进行对该循

环处理实施软件流水线操作的最优化。

另外，优选所述语法分析单元分析所述第 1 程序中包含的指针变量所指的数据的配置位置，并生成包括该数据的配置位置的所述分析信息，优选所述提示信息附加单元根据所述分析信息，输出把与数据的配置位置相关的提示信息附加到所述第 1 程序中的所述第 2 程序。并且，所述提示信息附加单元也可以具有：定位判断部，根据所述分析信息，判断所述第 1 程序中包含的指针变量所指的数据是否按照规定值被定位；程序输出部，在所述定位判断部的判断结果为肯定时，输出把该数据和该数据按照规定值被定位作为提示信息附加到所述第 1 程序中的所述第 2 程序。

可以在程序中自动插入与数据的配置相关的提示信息。通过附加上述的提示信息，编译器可以利用成对命令，进行从存储器中一次读出多个数据并写入的最优化。由此，可以减少存储器存取的次数，可以使处理高速化。

另外，优选所述语法分析单元分析所述第 1 程序中包含的指针变量存取的区域，并生成包括分析结果的所述分析信息，优选所述提示信息附加单元根据所述分析信息，输出把与指针变量存取的区域相关的提示信息附加到所述第 1 程序中的所述第 2 程序。并且，所述提示信息附加单元也可以具有：重复判断部，根据所述分析信息，判断所述第 1 程序中包含的指针变量存取的区域是否具有与其他指针变量存取的区域之间重复的区域；程序输出部，在所述重复判断部的判断结果为否定时，输出把提示信息附加到所述第 1 程序中的所述第 2 程序，该提示信息是表示所述第 1 程序中包含的所述指针变量存取的区域不具有与所述其他指针变量存取的区域之间重复的区域的信息。例如，所述第 1 和第 2 程序是利用基于 ISO/IEC 9899：1999－Programming Language C 标准的语言记述的，所述提示信息是所述指针变量和 restrict 记述的组合。

通过插入 restrict 记述得知，例如，指针变量 r1 所指的区域和指针变量 r2 所指的区域互不重叠。在这种情况下，编译器可以进行更换在前者区域写入数据的命令、和在后者区域写入数据的命令的执行顺序的最优化，能够使处理高速化。

另外，优选所述语法分析单元对源自所述第 1 程序中包含的变量读出数据或向该变量写入数据进行分析，并生成包括分析结果的所述分析信息，优选所述提示信息附加单元根据所述分析结果，输出把与从变量读出数据或向变量写入数据相关的提示信息附加到所述第 1 程序中的所述第 2 程序。例如，所述提示信息附加单元具有：写入判断部，根据所述分析信息，判断向所述第 1 程序中包含的变量的存取是否是从数据的写入开始的；程序输出部，在所述写入判断部的判断结果为肯定时，把该变量和向该变量的存取是从数据的写入开始的作为提示信息，在所述第 1 程序中的、进行向所述变量写入数据的位置的前面插入所述提示信息，生成所述第 2 程序并输出。

可以在程序中自动插入与从变量的数据读出或向变量的数据写入相关的提示信息。通过把该变量和向变量的存取是从数据的写入开始的作为提示信息插入，编译器把该提示信息作为线索，在具有高速缓存存储器的计算机中产生了向利用该提示信息指定的变量的存储器存取的情况下，可以进行只确保用于存储该变量的值的区域，而且不从主存储器向高速缓存存储器转发（预取）该变量的值的最优化。由此，可以减少执行机械语言程序时的存储器存取时间。

另外，优选所述语法分析单元分析所述第 1 程序中包含的分支条件的静态成立频次，并生成包括分析结果的所述分析信息，优选所述提示信息附加单元根据所述分析结果，输出把与分支条件的成立频次相关的提示信息附加到所述第 1 程序中的所述第 2 程序。例如，所述提示信息附加单元具有：可能性判断部，根据所述分析信息，判断所述第 1 程序中包含的分支条件成立的可能性；程序输出部，在可以判

断为所述可能性判断部的判断结果为肯定的概率高的情况下，输出把表示所述分支条件成立的可能性大的提示信息附加到所述第 1 程序中的所述第 2 程序。

可以在程序中自动插入与分支条件的成立频次相关的提示信息。并且，通过插入表示分支条件成立的可能性大的提示信息，编译器可以按照该提示信息进行下述的机械语言命令配置的最优化，即，相比 C 语言中的 if 语句不成立时执行的命令串即利用 else 语句指定的命令串，优先执行 if 语句的条件式成立时执行的命令串。由此，可以提高执行机械语言程序时的处理时间。

另外，优选所述提示信息是指示编译器的程序的最优化处理方法的信息。并且，所述语法分析单元也可以分析所述第 1 程序中包含的循环处理的反复次数，并生成包括该反复次数的分析信息，所述提示信息附加单元也可以根据所述分析信息，输出把指示基于循环展开的循环处理的最优化的提示信息附加到所述第 1 程序中的所述第 2 程序。

可以在程序中自动插入直接对编译器指示基于循环展开的最优化的提示信息。通过在第 1 程序中附加这种提示信息，编译器可以对所指定的循环处理实施基于循环展开的最优化。由此，在执行机械语言程序时，可以高速执行循环处理。

另外，优选所述语法分析单元分析所述第 1 程序中包含的循环处理的反复次数，并生成包括该反复次数的分析信息，优选所述提示信息附加单元根据所述分析信息，输出把指示基于软件流水线操作的循环处理的最优化的提示信息附加到所述第 1 程序中的所述第 2 程序。

可以在程序中自动插入直接对编译器指示基于软件流水线操作的最优化的提示信息。通过在第 1 程序中附加这种提示信息，编译器可以对所指定的循环处理实施基于软件流水线操作的最优化。由此，在执行机械语言程序时，可以高速执行循环处理。

另外，优选所述语法分析单元分析所述第 1 程序中包含的数据的定位值，并生成包括该分析结果的所述分析信息，优选所述提示信息附加单元具有：条件判断部，根据所述分析信息，判断第 1 程序中包含的数据的定位值是否满足该数据的类型尺寸的 2 倍以上这一条件；程序输出部，输出对所述第 1 程序中包含的满足所述条件的数据附加了指示基于成对命令的生成的数据存取的最优化的提示信息的所述第 2 程序。

可以在程序中自动插入直接指示编译器输出成对命令的提示信息。通过对上述数据附加这种提示信息，编译器可以发出从存储器中一次读出该数据并写入存储器的成对命令。由此，可以减少执行机械语言程序时的存储器存取的次数，可以使处理高速化。

另外，优选所述提示信息是指示变量和高速缓存存储器的控制处理的最优化的信息。例如，所述语法分析单元对从所述第 1 程序中包含的变量读出数据或向该变量写入数据进行分析，优选所述提示信息附加单元具有：写入判断部，根据所述分析信息，判断向所述第 1 程序中包含的变量的存取是否是从数据的写入开始的；程序输出部，在所述写入判断部的判断结果为肯定时，把该变量和在所述高速缓存存储器上确保用于存储该变量的值作为提示信息，在所述第 1 程序中的、向所述变量写入数据的位置的前面插入所述提示信息，生成所述第 2 程序并输出。

可以在程序中自动插入指示高速缓存存储器等的控制处理的提示信息。通过在第 1 程序中插入上述提示信息，编译器把该提示信息作为线索，可以在具有高速缓存存储器的计算机中进行下述最优化，即，在高速缓存存储器上确保用于存储利用该提示信息指定的变量的值的区域。由此，可以减少执行机械语言程序时的存储器存取时间。

另外，优选所述第 1 程序包括第 1 提示信息，优选所述提示信息附加单元根据所述分析信息，输出把提供给编译器的逻辑上不矛盾的

第 2 提示信息附加到所述第 1 程序中的第 2 程序。

在第 1 程序中，即使用户积极地向编译器提供了附注所代表的提示信息时，也能够进行提示信息的检查，并修改源程序（第 1 程序），以便能够进行良好的最优化。并且，可以检查附注所代表的提示信息，并修改源程序（第 1 程序），以使编译器不生成错误的机械语言程序。

并且，所述提示信息附加单元也可以附加订正了所述第 1 提示信息的逻辑错误的所述第 2 提示信息。

即使是只能使用第 2 提示信息的编译器，也能够编程包括第 1 提示信息的程序，可以实现资产的活用。

本发明另外涉及的程序处理装置，在程序中插入提供给编译器的提示信息，具有提示信息附加单元，把第 1 程序和与所述第 1 程序不同的程序的语法分析结果即分析信息作为输入，根据所述分析信息，输出把提供给编译器的逻辑上不矛盾的提示信息附加到第 1 程序中的第 2 程序。

根据从外部输入的分析信息，提供给编译器的逻辑上不矛盾的提示信息被附加到第 1 程序中。因此，即使用户未向编译器提供附注所代表的提示信息，也能够自动插入提示信息并修改源程序，以便能够进行良好的最优化。

另外，本发明不仅能够实现具有这种特征单元的程序处理装置，也能够实现把程序处理装置具有的特征单元作为步骤的程序处理方法。并且，能够实现使计算机发挥程序处理装置具有的特征单元的作用的程序。此外，这种程序当然可以通过 CD－ROM 等记录介质和因特网等通信网络流通。

根据本发明可以提供一种程序处理装置，即使用户未向编译器提供附注所代表的提示信息，也能够通过自动插入提示信息来修改源程序，以便能够进行良好的最优化。

并且，可以提供一种程序处理装置，即使用户积极地向编译器提供了附注所代表的提示信息，也能够进行提示信息的检查并修改源程

序，以便能够进行良好的最优化。

另外，可以提供一种程序处理装置，能够检查附注所代表的提示信息并修改源程序，以使编译器不生成错误的机械语言程序。

本发明的上述及其他目的、特征和优点，根据表示作为本发明示例的优选实施方式的附图及以下相关说明将更加明确。

附图说明

图 1 是表示包括直接进行与最优化相关的指示的附注的源程序的一例图。

图 2 是表示包括对编译器表示程序整体的静态信息的附注的源程序的一例图。

图 3 是表示程序处理装置的结构的功能方框图。

图 4A 是表示包括含有循环处理的函数 func1 的程序的一例图。

图 4B 是表示含有 main 函数、函数 func2 和函数 func3 的程序的一例图。

图 5 是语法分析部执行的处理的流程图。

图 6 是表示调用流图的一例图。

图 7 是表示分析信息的一例图。

图 8 是提示信息插入部执行的处理的流程图。

图 9 是表示提示信息插入部的输出结果即程序的一例图。

图 10A 是表示含有函数 func1、函数 func2 和函数 func3 的程序的一例图。

图 10B 是表示含有 main 函数的程序的一例图。

图 11 是根据图 10A 和图 10B 所示的程序，由语法分析部生成的调用流图的一例图。

图 12 是表示由语法分析部生成的分析信息的一例图。

图 13 是提示信息插入部执行的处理的流程图。

图 14 是表示提示信息插入部的输出结果即程序的一例图。

图 15 是表示输入程序处理装置的程序的一例图。

图 16 是根据图 15 所示的程序，由语法分析部生成的调用流图的一例图。

图 17 是表示由语法分析部生成的分析信息的一例图。

图 18 是提示信息插入部执行的处理的流程图。

图 19 是表示包括提示信息的程序的一例图。

图 20 是表示输入程序处理装置的程序的一例图。

图 21 是根据图 20 所示的程序，由语法分析部生成的调用流图的一例图。

图 22 是表示由语法分析部生成的分析信息的一例图。

图 23 是提示信息插入部执行的处理的流程图。

图 24 是表示包括提示信息的程序的一例图。

图 25 是表示输入程序处理装置的程序的一例图。

图 26 是表示图 5 的 S202 的处理结果生成的调用流图的图。

图 27 是表示图 5 的 S203 和 S204 的处理结果生成的分析信息的图。

图 28 是提示信息插入部执行的处理的流程图。

图 29 是表示提示信息插入部的输出结果即程序的一例图。

图 30 是表示输入程序处理装置的程序 100 的一例图。

图 31 是表示调用流图的一例图。

图 32 是表示分析信息的一例图。

图 33 是表示提示信息插入部的输出结果即程序的一例图。

图 34 是表示输入程序处理装置的程序的一例图。

图 35 是表示图 5 的 S202 的处理结果生成的调用流图的图。

图 36 是表示图 5 的 S203 和 S204 的处理结果生成的分析信息的图。

图 37 是提示信息插入部执行的处理的流程图。

图 38 是表示提示信息插入部的输出结果即程序的一例图。

图 39 是表示输入程序处理装置的程序的一例图。

图 40 是表示图 5 的 S202 的处理结果生成的调用流图的图。

图 41 是表示图 5 的 S203 和 S204 的处理结果生成的分析信息的图。

图 42 是提示信息插入部执行的处理的流程图。

图 43 是包括提示信息的程序的一例图。

图 44 是说明提示信息的修改和置换的图。

图 45 是说明提示信息的修改和置换的图。

图 46 是说明程序处理装置的编译选择的输出处理的图。

图 47 是表示程序处理装置的其他结构的功能方框图。

图 48 是表示输入程序处理装置的程序的一例图。

图 49 是表示输入程序处理装置的分析信息的一例图。

图 50 是包括提示信息的程序的一例图。

具体实施方式

以下，参照附图说明本发明的实施方式涉及的程序处理装置。

图 3 是表示程序处理装置的结构的功能方框图。程序处理装置 102 接受利用高级语言记述的程序 100，自动生成包括附注所代表的针对编译器的提示信息的程序 101，具有语法分析部 104 和提示信息插入部 108。其中，输入程序处理装置 102 的程序 100 是用户记述的普通程序，是不包括提示信息的程序。

语法分析部 104 是处理部，接受程序 100（在程序 100 由多个文件构成时，适当地称为程序 100a 和程序 100b），对程序 100 进行在编译器等中使用的普通的语法分析处理，把分析结果作为分析信息 106 输出。

提示信息插入部108是处理部，根据程序100和分析信息106，进行可以插入程序100的提示信息的插入，输出包括提示信息的程序101（在程序101由多个文件构成时，适当地称为程序101a和程序101b）。

另外，语法分析部104和提示信息插入部108执行的处理在后述的每个实施方式中是不同的。因此，在说明各个实施方式时进行详细说明。

（实施方式1.与反复次数相关的提示信息）

在本实施方式中，说明在程序中自动插入与循环处理（for、while、do）的反复次数相关的提示信息的程序处理装置。与循环处理的反复次数相关的提示信息例如包括：

（1）指定循环处理的反复的最大次数的信息，

（2）指定循环处理的反复的最小次数的信息，

（3）指定循环处理的反复次数一定是偶数的信息，

（4）指定循环处理的反复次数一定是奇数的信息。

说明自动生成包括这些提示信息的程序101的程序处理装置102执行的处理。

图4A和图4B是表示输入程序处理装置102的程序的一例图。图4A是表示含有包括循环处理（★多个★）的函数func1的程序的一例图，图4B是表示含有main函数、函数func2和函数func3的程序的一例图。

说明以这些程序100a和100b为输入的语法分析部104和提示信息插入部108执行的处理。

图5是语法分析部104执行的处理的流程图。语法分析部104分析整个程序100（程序100a和程序100b），生成各个函数的调用流图（S202）。这与利用普通编译器等生成的调用流图相同。

图6是表示调用流图的一例图。图6所示的调用流图107是利用

图 4A 和图 4B 分别示出的程序 100a 和程序 100b 生成的。根据该调用流图 107 得知，main 函数调用了函数 func2 和函数 func3。并且，得知函数 func2 调用了函数 func1，函数 func3 调用了函数 func1。这样，调用流图 107 利用箭头的指向表示函数之间的调用关系。

然后，如图 5 所示，语法分析部 104 根据所生成的调用流图 107，分析广域函数或各个函数被调用时的哑变元取什么值（S203）。在本实施方式中，语法分析部 104 分析广域函数或哑变元能够取得的值的最大值和最小值，并且分析是否能够取得偶数值以及是否能够取得奇数值。另外，也可以分析函数内的局部变量。

语法分析部 104 把分析的结果作为分析信息 106 输出（S204）。图 7 是表示分析信息的一例图。图 7 所示的分析信息 106 表示分析结果的一部分。例如，关于函数 func3 的哑变元 s，最大值和最小值都是“5”，表示不能取得偶数值，能够取得奇数值。这可以通过进行以下分析求出。即，从图 6 所示的调用流图 107 得知函数 func3 被 main 函数调用。从程序 101b 得知 main 函数中的函数 func3 的实变元是“5”，函数 func3 只被调用 1 次。因此，可以获得上述的分析结果。

并且，关于函数 func1 内的广域变量 y，最大值是“6”，最小值是“5”，分析信息 106 表示偶数值和奇数值都能取得。这可以通过进行以下分析求出。即，从调用流图 107 得知函数 func1 被函数 func2 和函数 func3 调用。在函数 func2 内，广域变量 y 被代入“6”。并且，在函数 func3 内，广域变量 y 被代入哑变元 s 的值。此处，从调用流图 107 得知函数 func1 被 main 函数调用。因此，调查 main 函数内的函数 func3 的实变元得知是“5”。所以，函数 func3 的哑变元 s 的值是“5”。即，函数 func3 内的广域变量 y 的值也是“5”。由此，函数 func3 内的广域变量能够取得的值是“5”或“6”。因此，可以获得上述的分析结果。同样，可以获得图 7 所示的分析信息 106。

下面，说明提示信息插入部 108 执行的处理。

图 8 是提示信息插入部 108 执行的处理的流程图。提示信息插入部 108 对程序 100a 和程序 100b 中包含的各个循环处理反复以下处理。如图 4A 所示，程序 100a 含有循环处理 A 和循环处理 B 这两个循环处理。因此，关于这两个循环处理分别执行以下处理。

提示信息插入部 108 一面检查分析信息 106，一面调查能够取得着手的循环处理的反复次数的可能性（S301）。提示信息插入部 108 检查能否算出着手的循环处理的最低反复次数（S302）。如果能够算出该最低反复次数（S302 为是），提示信息插入部 108 在把该最低反复次数设为 n 次时，在程序 100a 中插入指定着手的循环处理的最低反复次数的提示信息即附注“#pragma_min_iteration=n”（S303）。该附注是上述第（2）个提示信息的一种。

然后，提示信息插入部 108 调查算出着手的循环处理的反复次数只是偶数或只是奇数的可能性（S304）。如果能够算出（S304 为是），提示信息插入部 108 在着手的循环处理的反复次数只是偶数时，在程序 100a 中插入指定循环处理的反复次数一定是偶数次的提示信息即附注“#pragma_iteration_even”，在着手的循环处理的反复次数只是奇数时，在程序 100a 中插入指定循环处理的反复次数一定是奇数次的提示信息即附注“#pragma_iteration_odd”（S305）。提示信息插入部 108 通过在程序 100a 中插入这种附注，生成包括提示信息的程序 101a。

其中，附注“#pragma_iteration_even”和附注“#pragma_iteration_odd”分别是上述第（3）个提示信息和第（4）个提示信息的一种。

下面，说明提示信息插入部 108 插入附注的结果的一例。例如，如果着手图 4A 所示的程序 100a 的循环处理 A，则从分析信息 106 得知用于规定循环处理的反复次数的哑变元 x 的值的最小值是“2”。因此，可以求出循环处理 A 的最低反复次数是“2”。所以，在图 9 所示

的提示信息插入部 108 的输出结果即程序 101a 的第 3 行插入附注#pragma_min_iteration＝2”，作为针对循环处理 A 的提示信息。

并且，如果是循环处理 A，从分析信息 106 得知用于规定循环处理的反复次数的哑变元 x 的值只能取偶数，不能取奇数。因此，在程序 101a 的第 4 行插入附注#pragma_iteration_even”，作为针对循环处理 A 的提示信息。

同样，如果着手循环处理 B，则循环处理 B 的最低反复次数的是“5”。因此，在程序 101a 的第 14 行插入附注“#pragma_min_iteration＝5”，作为针对循环处理 B 的提示信息。

如上所述，根据本实施方式，能够在程序中自动插入与循环处理的反复次数相关的提示信息。

通过插入附注“#pragma_min_iteration”，可以根据利用附注“#pragma_min_iteration”指定的循环处理的最低反复次数，对该循环处理进行能否适用软件流水线操作的判定，在能够适用时，对该循环处理进行软件流水线操作的最优化。所述软件流水线操作指同时执行几个不同的迭代（interation）（反复处理）的技术。

并且，通过插入附注“#pragma_iteration_even”或附注“#pragma_iteration_odd”，编译器可以根据该附注进行循环展开的最优化。所述循环展开是循环处理高速化的一种方法，是通过同时执行多个（此处为 2 个）迭代（反复处理），使循环处理内的执行速度高速化的方法。循环展开在展开的反复次数是 2 次时，最优化的处理方法对于循环处理的反复次数为偶数和奇数时不同。如果是偶数，可以直接进行循环展开，如果是奇数，则需要在循环处理的外侧执行半端的一次。

另外，在上述实施方式中，使用第（2）到第（4）个提示信息的一种即附注进行了说明，但是，作为第（1）个提示信息的一种，也可以使用附注“#pragma_max_iteration”。该附注表示提示信息之后

的循环处理（for，while，do）的反复次数最大是几次。例如，在得知程序100中的循环处理的反复次数最大为10次时，也可以在程序100中插入附注“#pragma_max_iteration＝10”，生成包括提示信息的程序101。

并且，提示信息插入部108在不使用分析信息106即可插入提示信息的情况下，也可以进行这种处理。这是因为，例如，在循环处理的反复次数不是变量而是利用常数定义的情况下，不参照分析信息106，提示信息插入部108即可求出循环处理的最低反复次数等。

（实施方式2.与指针变量相关的提示信息）

在本实施方式中，说明在程序中自动插入与指针变量相关的提示信息的程序处理装置。与指针变量相关的提示信息例如包括：

（1）指定指针变量所示的数据的定位值的提示信息，

（2）表示指针变量所示的区域不重叠的提示信息。

说明自动生成包括这些提示信息的程序101的程序处理装置102执行的处理。

图10A和图10B是表示输入程序处理装置102的程序的一例图。图10A是表示包括函数func1、函数func2和函数func3的程序的一例图，图10B是表示包括main函数的程序的一例图。

图10A所示的程序101a和图10B所示的程序101b分别被编程后具有链接关系。

程序100a中包含的附注“#pragma_align_object”是用户指示，指示利用所指定的字节数定位（alignment）此后示出的数据。例如，函数func1内的附注“#pragma_align_object＝4a，b”这种表述，是以4字节单位在主存储器上定位变量a和b的指示。变量a和b是short型变量，通常short型变量是2字节的变量。因此，在本实施方式中，在没有附注“#pragma_align_object”的指定的情况下，short型变量以2字节单位被定位。

说明以这些程序100a和100b为输入时的语法分析部104和提示信息插入部108执行的处理。

语法分析部104执行和图5所示流程图相同的处理。但是，在S203的处理中分析的对象不同。即，语法分析部104分析函数的指针变量所指的数据的定位值，并且对指针变量所指区域进行分析，分析是否存在指向与其他指针相同的区域的可能性。

图11是表示根据图10A和图10B所示程序101a和101b由语法分析部104生成的调用流图的一例图。根据该调用流图107，main函数调用了函数func1和函数func2。并且，函数func1和函数func2在分别调用函数func3。

图12是表示由语法分析部104生成的分析信息106的一例图。在图5的S203的处理中，语法分析部104根据程序101a和101b，对指示各个函数调用时的指针变量和广域指针变量的定位及相同区域的可能性进行分析。

例如，对函数func3的哑变元q1进行分析。根据调用流图107得知，函数func3被函数func1和函数func2调用。对应函数func3的哑变元q1的函数func1内的函数func3的实变元“&a”所指的数据，根据附注“#pragma_align_object＝4a，b”被以4字节单位定位。并且，对应函数func3的哑变元q1的函数func2内的函数func3的实变元“&x”所指的数据，根据附注“#pragma_align_object＝8 x”被以8字节单位定位。因此，函数func3的哑变元q1所指向的数据的定位值是4或8。

并且，函数func3的哑变元q1和q2分别对应函数func1内的函数func3的实变元“&a”和p1。在函数func1内调用函数func3之前，p1被代入“&a”的值。因此，p1和“&a”表示相同的值。所以，作为有可能指定与哑变元q1所指区域相同的区域的指针，可以求出哑变元q2。

以下，同样对哑变元 q2 和 q3 进行分析，可以获得图 12 所示的分析信息 106。

下面，说明提示信息插入部 108 执行的处理。

图 13 是提示信息插入部 108 执行的处理的流程图。提示信息插入部 108 对程序 100a 和程序 100b 中包含的各个指针变量反复以下处理。如图 10A 所示，程序 100a 包括指针变量 q1、q2 和 q3 等。

提示信息插入部 108 一面检查分析信息 106 一面调查指针变量所指数据的配置和该指针变量的数据存取（S401）。提示信息插入部 108 检查能否算出与着手的指针变量所指数据的默认定位值不同的定位值（S402）。指针变量所指数据的默认的定位值根据指针变量的类型来确定，例如，指针变量为 short 型时为 2 字节。

如果能够算出与默认定位值不同的定位值（S402 为是），提示信息插入部 108 在程序 100a 插入指定着手的指针变量所指数据的定位值中的最小值的提示信息即附注“#pragma_align_pointer”（S403）。例如，如果针对指针变量 q 的定位值的最小值是 n，则在程序 100a 中插入附注“#pragma_align_pointer＝n q”。该附注是上述第（1）个提示信息的一种。

然后，提示信息插入部 108 调查可否判别为着手的指针变量所指的数据区域与其他指针变量所指的数据区域互不重叠（S404）。

在可以判别为不重叠时（S404 为可以），提示信息插入部 108 在着手的指针变量的前面插入 restrict 表述（S405）。restrict 表述是用 C99 语言（ISO/IEC 9899：1999－Programming Language C）导入的，表示在该范围内指定的所有指针变量所指的主存储器上的区域互不重叠。restrict 表述是上述第（2）个提示信息的一种。通过在程序 100a 中插入这种附注或 restrict 表述，可以生成包括提示信息的程序 101a。

下面，说明提示信息插入部 108 插入附注和 restrict 表述的结

果的一例。例如，如果着手图 10A 所示的程序 100a 的函数 func3 的哑变元（指针变量）q1，根据分析信息 106 得知哑变元 q1 所指数据的定位值是 4 或 8。因此，在图 14 所示的提示信息插入部 108 的输出结果即程序 101a 的函数 func3 的前面，插入把哑变元 q1 所指数据的定位值的最小值“4”指定为定位值的附注“#pragma_align_pointer＝4 q1 ”。同样，在函数 func3 的前面插入附注“#pragma_align_pointer＝4 q2”。另外，在图 14 中，把这两个附注一并表述为附注“#pragma_align_pointer＝4 q1、q2”。

并且，根据分析信息 106 得知，哑变元 q3 所指的数据区域与其他指针变量所指的数据区域互不重叠。因此，在程序 101a 中的哑变元 q3 前面插入 restrict 表述。

如上所述，根据本实施方式，可以在程序中自动插入与数据的配置相关的提示信息。

通过插入附注“#pragma_align_pointer”，编译器可以利用成对命令进行从存储器一次读出多个数据并写入的最优化。由此，可以减少存储器存取的次数，能够使处理高速化。

并且，通过插入 restrict 表述，例如，可以得知指针变量 r1 所指的区域与指针变量 r2 所指的区域互不重叠。在这种情况下，编译器可以进行更换在前者区域写入数据的命令、和在后者区域写入数据的命令的执行顺序的最优化，能够使处理高速化。

另外，在上述实施方式中，把函数的哑变元即指针变量作为具体示例进行了说明，但不限于哑变元，可以对广域变量即指针执行相同的处理，在程序中插入提示信息。

（实施方式 3.与变量的读写相关的提示信息）

在本实施方式中，说明在程序中自动插入与变量的读写相关的提示信息的程序处理装置。与变量的读写相关的提示信息例如包括：

（1）表示在提示信息的配置位置以后，向所指定变量的存取是

从数据的写入开始的提示信息。

说明自动生成包括该提示信息的程序 101 的程序处理装置 102 执行的处理。

图 15 是表示输入程序处理装置 102 的程序的一例图。程序 100 包括函数 func1、函数 func2 和函数 func3。

说明被输入了该程序100时的语法分析部104和提示信息插入部108 执行的处理。

语法分析部 104 执行与图 5 所示流程图相同的处理。但是，在 S203 的处理中分析的对象不同。即，语法分析部 104 对各个函数进行是否存在向广域变量的读出、写入的分析。

图 16 是表示根据图15所示的程序101由语法分析部104生成的调用流图的一例图。根据该调用流图 107，函数 func1 调用了函数 func2 和函数 func3。

图 17 是表示由语法分析部 104 生成的分析信息 106 的一例图。在图 5 的 S203 的处理中，语法分析部 104 对各个函数的广域变量检查有无读写。

例如，在函数 func2 中存在向广域变量 y 的数据写入、和从广域变量 x 的数据读出，在函数 func3 中存在向广域变量 z 的数据写入、和从广域变量 y 的数据读出，所以分析信息 106 为图 17 所示结构。另外，对函数 func1 也进行相同的处理，生成分析信息 106。

下面，说明提示信息插入部 108 执行的处理。

图 18 是提示信息插入部 108 执行的处理的流程图。提示信息插入部 108 对程序 100 中包含的各个广域变量反复以下处理。如图 15 所示，程序 100 包含 3 个广域变量 x、y、z。

提示信息插入部 108 一面检查分析信息 106，一面调查从着手的广域变量的数据读出及向着手的广域变量的数据写入（S501）。如果向着手的广域变量的存取是从数据的写入开始的（S501 为肯定），提

示信息插入部 108 在发生向该广域变量的写入之前插入附注“#pragma_start_from_write”(S503)。例如，如果广域变量是 a，则在发生向该广域变量 a 的写入之前插入附注“#pragma_start_from_write a”。该附注是上述第（1）个提示信息的一种。通过在程序 100 中插入这种附注，生成图 19 所示的包括提示信息的程序 101。

下面，说明提示信息插入部 108 插入附注的结果的一例。例如，着手图 15 所示程序 100 的广域变量 y。关于向广域变量 y 的数据写入，在调查分析信息 106 时，在函数 func2 只进行数据的写入。因此，在调用函数 func2 时，得知向广域变量 y 的存取是从数据的写入开始的。并且，根据调用流图 107 得知函数 func2 被函数 func1 调用。因此，提示信息插入部 108 在函数 func1 内调用函数 func2 的位置前面插入附注“#pragma_start_from_write y”。

同样，提示信息插入部 108 在函数 func1 内调用函数 func3 的位置前面插入附注“#pragma_start_from_write z”。由此，生成包括提示信息的程序 101。

如上所述，根据本实施方式，可以在程序中自动插入与从变量的数据读出或向变量的数据写入相关的提示信息。

通过插入附注“#pragma_start_from_write”，在具有高速缓存存储器的计算机中产生向利用该附注指定的变量的存储器存取的情况下，编译器以该附注为线索，可以进行只确保用于存储该变量的值的区域，而且不从主存储器向高速缓存存储器转发（预取处理）该变量的值的最优化。由此，可以减少执行机械语言程序时的存储器存取时间。

（实施方式 4. 与静态频次相关的提示信息）

在本实施方式中，说明在程序中自动插入与静态频次相关的提示信息的程序处理装置。与静态频次相关的提示信息例如包括：

（1）表示分支条件的成立频次高的提示信息，

（2）表示分支条件的不成立频次高的提示信息。

说明自动生成包括这些提示信息的程序101的程序处理装置102执行的处理。

图20是表示输入程序处理装置102的程序的一例图。程序100包括函数func1和函数func2。

说明被输入了该程序100时的语法分析部104和提示信息插入部108执行的处理。

语法分析部104执行与图5所示流程图相同的处理。但是，在S203的处理中分析的对象不同。即，语法分析部104对各个函数的哑变元和广域变量进行值的频次分析。

图21是表示根据图20所示的程序100由语法分析部104生成的调用流图的一例图。根据该调用流图107，函数func1调用了函数func2。

图22是表示由语法分析部104生成的分析信息106的一例图。在图5的S203的处理中，语法分析部104对各个函数的哑变元和广域变量进行值的频次分析。

例如，分析函数func2的哑变元i的频次，函数func2有可能被函数func1调用合计17次，把此时的哑变元i的值的频次表示为分析信息106。例如，哑变元i为0的次数为2次。因此，表示哑变元i为0的概率为2/17。

下面，说明提示信息插入部108执行的处理。

图23是提示信息插入部108执行的处理的流程图。提示信息插入部108对程序100中包含的各个if语句反复以下处理。如图20所示，程序100包含if语句“if（i%5==0)”。

提示信息插入部108一面检查分析信息106，一面调查着手的if语句的条件式成立或不成立的频次（S601)。在可以判断为该条件式

成立的概率高的情况下（S601 为肯定），在该 if 语句前面插入表示分支语句的成立频次高的提示信息的一种即附注“#pragma_likely_true”（S603）。该附注是上述第（1）个提示信息的一种。

在可以判断为该条件式不成立的概率高的情况下（S604 为肯定），提示信息插入部 108 在该 if 语句前面插入表示分支语句的不成立频次高的提示信息的一种即附注“#pragma_likely_false”（S605）。该附注是上述第（2）个提示信息的一种。提示信息插入部 108 在程序 100 中插入这种附注，由此生成包括提示信息的程序 101。

下面，说明提示信息插入部 108 插入附注的结果的一例。例如，着手图 20 所示程序 100 的 if 语句“if（i%5==0）”时，根据分析信息 106，该 if 语句的条件式“（i%5==0）”成立的概率为 4/17（哑变元 i 为 0 的概率 2/17 与哑变元 i 为 5 的概率 2/17 之和）。同样，根据分析信息 106，该 if 语句的条件式“（i%5==0）”不成立的概率为 13/17。因此，由于 if 语句的条件式“（i%5==0）”不成立的概率超过 1/2，所以提示信息插入部 108 进行该条件式不成立的概率高的判断。所以，如图 24 所示，在 if 语句“if（i%5==0）”的前面插入附注“#pragma_likely_false”。由此，生成包括提示信息的程序 101。

如上所述，根据本实施方式，可以在程序中自动插入与静态频次（分支条件的成立频次）相关的提示信息。

通过插入附注“#pragma_likely_true”，编译器可以按照该附注进行下述的机械语言命令配置的最优化，即，相比 if 语句的条件式不成立时执行的命令串即用 else 语句指定的命令串，优先执行 if 语句成立时执行的命令串。由此，可以提高执行机械语言程序时的处理时间。

并且，通过插入附注“#pragma_likely_false”，编译器可以按

照该附注进行下述的机械语言命令配置的最优化，即，相比 if 语句的条件式成立时执行的命令串，优先执行 if 语句不成立时执行的命令串即用 else 语句指定的命令串。由此，可以提高执行机械语言程序时的处理时间。

（实施方式 5.指定最优化方法的第 1 提示信息）

在本实施方式中，说明在程序中自动插入直接对编译器指定最优化方法的提示信息的程序处理装置。作为直接对编译器指定最优化方法的提示信息，例如有指定循环展开的提示信息或指定软件流水线操作的提示信息。

说明自动生成包括该提示信息的程序 101 的程序处理装置 102 执行的处理。

图 25 是表示输入程序处理装置 102 的程序的一例图。程序 100 包含函数 func1、函数 func2 和函数 func3。

说明被输入了该程序 100 时的语法分析部 104 和提示信息插入部 108 执行的处理。

语法分析部 104 执行与图 5 所示流程图相同的处理。图 26 是表示图 5 的 S202 的处理结果所生成的调用流图的一例图。根据调用流图 107 得知，函数 func2 和函数 func3 分别在调用函数 func1。

图 27 是图 5 的 S203 和 S204 的处理结果所生成的分析信息的图。根据分析信息 106，函数 func1 的哑变元 x 的最大值是“6”，最小值是“4”，表示不能取得奇数值而取偶数值。分析信息 106 的生成处理与实施方式 1 相同，所以在此不重复其详细说明。

下面，说明提示信息插入部 108 执行的处理。

图 28 是提示信息插入部 108 执行的处理的流程图。提示信息插入部 108 对程序 100 中包含的各个循环处理反复以下处理。

提示信息插入部 108 一面检查分析信息 106，一面调查着手的循环处理的反复次数（S701）。在着手的循环处理的反复次数为 2 次以

上，而且反复次数只能取偶数次数或只能取奇数次数的情况下（S702为是），提示信息插入部 108 在该循环处理的前面插入附注“#pragma_loop_unrolling”（S703）。该附注是对通过循环展开使着手的循环处理最优化的编译器的提示信息。此处，假定将要进行的反复次数为 2 次。

在着手循环处理的反复次数可以取偶数和奇数双方，而且着手的循环处理的反复次数为 2 次以上情况下（S704 为是），提示信息插入部 108 在该循环处理的前面插入附注“#pragma_software_pipelining”（S705）。该附注是对编译器的通过软件流水线操作使着手的循环处理最优化的提示信息。

下面，说明提示信息插入部 108 插入附注的结果的一例。例如，如果着手图 25 所示程序 100 中的 for 循环处理，则根据图 27 所示分析信息 106 得知，该 for 循环处理的反复次数一定在 2 次以上，而且是偶数次。因此，在图 29 所示的提示信息插入部 108 的输出结果即程序 101 中插入附注“#pragma_loop_unrolling”，作为针对该 for 循环处理的提示信息。

在图 30 所示程序 100 被输入程序处理装置 102 的情况下，语法分析部 104 生成图 31 所示的调用流图 107 和图 32 所示的分析信息 106。如果着手图 30 所示程序 100 的 for 循环处理，根据分析信息 106，该 for 循环处理的反复次数虽然可以取偶数和奇数双方，但最小值为 2 次以上。因此，在图 33 所示的提示信息插入部 108 的输出结果即程序 101 中插入附注“#pragma_software_pipelining”，作为针对该 for 循环处理的提示信息。

如上所述，根据本实施方式，可以在程序中自动插入直接指示编译器进行基于循环展开的最优化的提示信息。

并且，可以在程序中自动插入直接指示编译器进行基于软件流水线操作的最优化的提示信息。

通过插入附注“#pragma_loop_unrolling”，编译器可以对所指定的循环处理实施基于循环展开的最优化。由此，在执行机械语言程序时，可以高速执行循环处理。

并且，通过插入附注“#pragma_software_pipelining”，编译器可以对所指定的循环处理实施基于软件流水线操作的最优化。由此，在执行机械语言程序时，可以高速执行循环处理。

（实施方式 6.指定最优化方法的第 2 提示信息）

在本实施方式中，说明在程序中自动插入直接对编译器指定最优化方法的提示信息的其他程序处理装置。作为直接对编译器装置指定最优化方法的提示信息，例如有指定输出成对命令的提示信息。

说明自动生成包括该提示信息的程序 101 的程序处理装置 102 执行的处理。

图 34 是表示输入程序处理装置 102 的程序的一例图。程序 100 包含函数 func1 和函数 func2。

说明被输入了该程序 100 时的语法分析部 104 和提示信息插入部 108 执行的处理。

语法分析部 104 执行与图 5 所示流程图相同的处理。但是，不同之处是在 S203 的处理中分析函数的哑变元所指排列的要素的定位值。另外，定位值的分析方法与实施方式 2 所示方法相同。所以，在此不重复其详细说明。

图 35 是表示图 5 的 S202 的处理结果所生成的调用流图的一例图。根据调用流图 107 得知，函数 func1 在调用函数 func2。

图 36 是图 5 的 S203 和 S204 的处理结果所生成的分析信息的一例图。根据分析信息 106，函数 func2 的哑变元 a 所指区域的数据、即排列 a 的各个要素以 4 字节单位被定位。

下面，说明提示信息插入部 108 执行的处理。

图 37 是提示信息插入部 108 执行的处理的流程图。提示信息插

入部 108 对程序 100 中包含的各个指针变量反复以下处理。

提示信息插入部 108 一面检查分析信息 106，一面调查着手的指针变量所指数据的配置（S901）。提示信息插入部 108 调查着手的指针变量所指数据的定位值是否为默认定位值的 2 倍以上（S902）。如果是默认定位值的 2 倍以上（S902 为是），在程序 100 中的该指针变量被说明开始到被参照的期间，插入附注“#pragma_pair_inst”。在着手的指针变量为 a 时，实际上插入附注“#pragma_pair_inst a”。该附注是针对编译器的指示，使其发出从存储器一次读出用着手的指针变量为 a 指定的数据（例如排列 a 的要素）并向存储器写入的成对命令，并进行最优化。

下面，说明提示信息插入部 108 插入附注的结果的一例。例如，着手图 34 所示的由函数 func2 的指针变量构成的哑变元 a。根据图 36 所示的分析信息 106 得知，哑变元 a 的定位值是 4。哑变元 a 是 short 型，short 型的尺寸是 2 字节。因此，得知哑变元 a 的定位值是默认定位值的 2 倍。所以，在图 38 所示的提示信息插入部 108 的输出结果即程序 101 中插入附注“#pragma_pair_inst a”，作为针对该哑变元 a（指针变量 a）的提示信息。

如上所述，根据本实施方式，可以在程序中自动插入直接指示编译器输出成对命令的提示信息。

通过插入附注“#pragma_ pair_inst a”，编译器可以发出从存储器一次读出指针 a 所指数据并向从存储器写入的成对命令。由此，在执行机械语言程序时，可以减少存储器存取的次数，能够使处理高速化。

（实施方式 7. 指定最优化方法的第 3 提示信息）

在本实施方式中，说明在程序中自动插入直接对编译器指定最优化方法的提示信息的另外其他的程序处理装置。作为直接对编译器装置指定最优化方法的提示信息，例如有使用指示高速缓存存储器等的

控制处理的内部函数的提示信息。

说明自动生成包括该提示信息的程序 101 的程序处理装置 102 执行的处理。

图 39 是表示输入程序处理装置 102 的程序的一例图。程序 100 包含函数 func1 和函数 func2。

语法分析部 104 执行与图 5 所示流程图相同的处理。但是，在 S203 的处理中，与实施方式 3 相同，对各个函数进行是否存在从广域变量的数据读出和向广域变量的数据写入的分析。

图 40 是表示图 5 的 S202 的处理结果所生成的调用流图的一例图。根据调用流图 107 得知，函数 func1 在调用函数 func2。

图 41 是图 5 的 S203 和 S204 的处理结果所生成的分析信息的图。根据分析信息 106 得知，例如，在函数 func2 内，进行从广域变量 x 的数据读出和向广域变量 y 的数据写入。

下面，说明提示信息插入部 108 执行的处理。

图 42 是提示信息插入部 108 执行的处理的流程图。提示信息插入部 108 对程序 100 中包含的各个广域变量反复以下处理。

提示信息插入部 108 一面检查分析信息 106，一面调查从着手的广域变量的数据读出和向着手的广域变量的数据写入（S1001）。在向着手的广域变量的存取是从数据的写入开始的情况下（S1002 为是），在发生向该广域变量的数据写入之前插入内部函数 reserve_region（）（S1003）。例如，如果广域变量是 a，则插入内部函数 reserve_region（a）。内部函数 reserve_region（a）是指示高速缓存存储器等的控制处理的提示信息的一种。内部函数 reserve_region（a）是执行下述处理的内部函数，即，在高速缓存存储器上只确保用于存储用实变元 a 指定的变量的值的区域，而且不从主存储器向高速缓存存储器转发（预取）该变量的值。

通过在程序 100 中插入这种附注，生成图 43 所示的包括提示信

息的程序 101。

下面，说明提示信息插入部 108 插入附注的结果的一例。例如，着手图 39 所示的程序 100 的广域变量 y。关于向广域变量 y 的数据写入，调查分析信息 106，在函数 func2 中只进行数据的写入。因此，得知在调用函数 func2 时，向广域变量 y 的存取是从数据的写入开始的。并且，根据调用流图 107 得知，函数 func2 被函数 func1 调用。所以，在函数 func1 内调用函数 func2 的位置前面插入内部函数 reserve_region (y)。由此，生成包括提示信息的程序 101。

如上所述，根据本实施方式，可以在程序中自动插入使用指示高速缓存存储器等的控制处理的内部函数的提示信息。

通过插入内部函数 reserve_region ()，在具有高速缓存存储器的计算机中该内部函数被调用的情况下，编译器以该内部函数为线索，可以进行在高速缓存存储器上只确保用于存储利用该内部函数的实变元指定的变量的值的区域，而且不从主存储器向高速缓存存储器转发（预取）该变量的值的最优化。由此，可以减少执行机械语言程序时的存储器存取时间。

根据上述实施方式 1～7，即使用户未向编译器提供附注所代表的提示信息，也能够自动地插入提示信息来以修改源程序，以便能够进行良好的最优化。

以上根据实施方式说明了本发明涉及的程序处理装置，但本发明不限于该实施方式。

例如，程序处理装置也可以接受预先包括提示信息的程序，而修改提示信息的错误或置换为其他提示信息。

图 44 是说明提示信息的修改和置换的图。在提供了图 44（a）所示的包括附注“#pragma_min_iteration＝5”的程序 100 时，程序处理装置 102 通过进行与实施方式 1 相同的处理，可以求出附注“#pragma_min_iteration＝2”作为应该赋予给函数 func1 的 for 循

环的附注。因此，程序处理装置 102 也可以生成把图 44（a）的程序 100 中包含的附注“#pragma_min_iteration＝5”修改为图 44（b）所示的附注“#pragma_min_iteration＝2”的程序 101。

并且，程序处理装置 102 通过还进行与实施方式 5 相同的处理，可以求出附注“#pragma_software_pipelining”作为应该赋予给函数 func1 的 for 循环的附注。因此，程序处理装置 102 也可以生成把图 44（b）的程序 100 中包含的附注“#pragma_min_iteration＝2”修改为图 44（c）所示的附注“#pragma_software_pipelining”的程序 101。

通过进行以上处理，即使用户积极地向编译器提供了附注所代表的提示信息，也能够进行提示信息的检查并修改源程序，以便能够进行良好的最优化。

并且，能够检查附注所代表的提示信息并修改源程序，以使编译器不生成错误的机械语言程序。

另外，即使不分析附注，如果程序中含有附注“#pragma_min_iteration＝n”（n 为 2 以上的整数），也可以自动把该附注转换为附注“#pragma_software_pipelining”。通过进行这种处理，可以使用能够解释附注“#pragma_software_pipelining”、但不能解释附注“#pragma_min_iteration”的编译器，进行程序的编程。因此，可以提高程序的资产价值。

图 45 是说明提示信息的修改和置换的其他图。在提供了图 45(a)所示的包括附注“#pragma_align_pointer＝8 a”的程序 100 时，程序处理装置 102 通过进行与实施方式 2 相同的处理，可以求出附注“#pragma_ align_pointer＝4 a”作为应该赋予给函数 func2 的哑变元即指针变量 a 的附注。因此，程序处理装置 102 也可以生成把图 45（a）的程序 100 中包含的附注“#pragma_align_pointer＝8 a”修改为图 45（b）所示的附注“#pragma_align_pointer＝4 a”的程

序 101。

并且，程序处理装置 102 通过还进行与实施方式 6 相同的处理，可以求出附注“#pragma_pair_inst a”作为应该赋予给函数 func2 的哑变元即指针变量 a 的附注。因此，程序处理装置 102 也可以生成把图 45（b）的程序 100 中包含的附注“#pragma_align_pointer＝4 a”修改为图 45（c）所示的附注“#pragma_pair_inst a”的程序 101。

并且，程序处理装置 102 也可以在程序中插入附注并输出编译选择。图 46 是说明程序处理装置 102 进行的编译选择的输出处理的图。

例如，在输入了图 46（a）所示的程序 100 时，根据上述实施方式，程序处理装置 102 按照图 46（b）所示，例如输出插入了附注“#pragma_align_pointer＝8 a”的程序 101，但也可以按照图 46（c）所示，输出编程时需要的编译选择。另外，“编译选择”指起动编译器时可以进行成为编译对象的程序 100 的指定及用户可以任意指定的针对编译器的指示。例如，用户编程程序 100“foo.c”时，可以使用命令“cc”，在计算机的命令行上输入 cc－falign－all－array＝8 foo.c。

另外，代替图 3 所示的程序处理装置 102，也可以使用图 47 所示的程序处理装置 202。程序处理装置 202 接受利用高级语言记述的程序 100 和对该程序 100 的分析信息 106，自动生成包括附注所代表的针对编译器的提示信息的程序 101，具有语法分析部 204 和提示信息插入部 108。语法分析部 204 是接受程序 100（100a、100b），对程序 100（100a、100b）进行在编译器等中使用的普通的语法分析处理的处理部。另外，分析信息 106 是对与通过语法分析部 204 进行了语法分析的程序 100（100a、100b）不同的程序 100（100a、100b）的分析结果。例如，程序 100 由程序 100a 和程序 100b 构成时，分析信息 106 是针对程序 100a 的分析信息 106，输入语法分析部 204 的程序是程序 100b。

提示信息插入部 108 的结构与上述的实施方式相同。所以，在此不重复其详细说明。图 48 表示输入程序处理装置 202 的程序 100a 的一例，图 49 表示输入程序处理装置 202 的分析信息 106 的一例。结果，可以从程序处理装置 202 获得图 50a 所示的包括提示信息的程序 101a。

并且，进行了程序 100 和 101 作为用 C 语言记述的源程序的说明，但也可以是利用除此以外的 C＋＋语言等高级语言记述的源程序，还可以是目标程序、利用中间语言等记述的程序、利用汇编语言记述的程序。

另外，在上述实施方式中，作为提示信息主要以附注为例进行了说明，但是，提示信息不限于附注，也可以是内部函数、编译选择、编程语言的预约语句等。

并且，在循环展开中进行的反复次数不限于 2 次，可以是比其多的次数，通过进行基于相同宗旨的处理，可以生成包括提示信息的程序 101。

本发明可以用作自动生成提供给编译器的提示信息并追加到程序中的程序处理装置，特别可以用作向程序自动插入编程时的最优化处理用的提示信息的预处理器等。

```
#pragma _software_pipelining
for (i = 0; i < 100; i++) {
      x[i] = y[i] + z[i];
}
```

图1

```
#pragma _min_iteration=5
for (i = 0; i < n; i++) {
      x[i] = y[i] + z[i];
}
```

图2

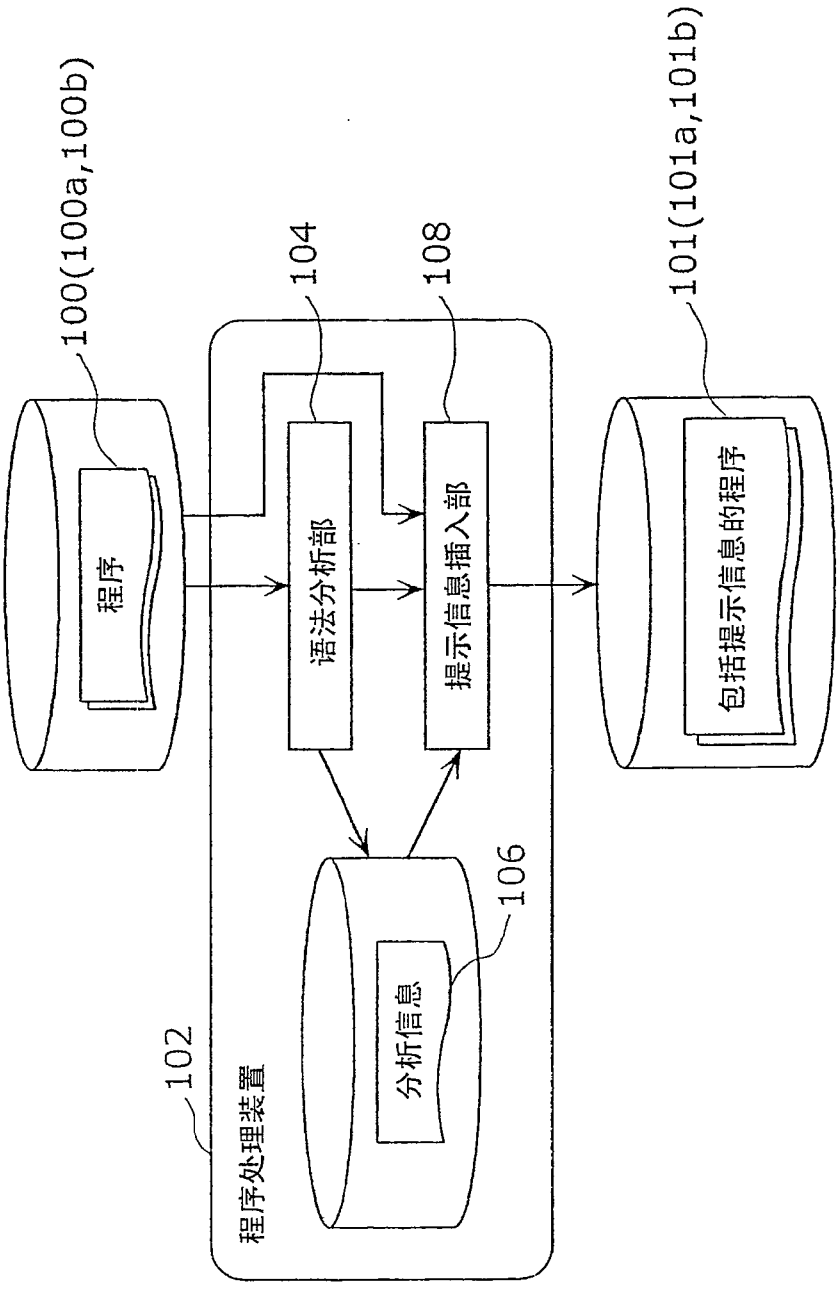
100(100a,100b)
程序
104
语法分析部
108
提示信息插入部
101(101a,101b)
包括提示信息的程序
106
分析信息
102
程序处理装置

图3

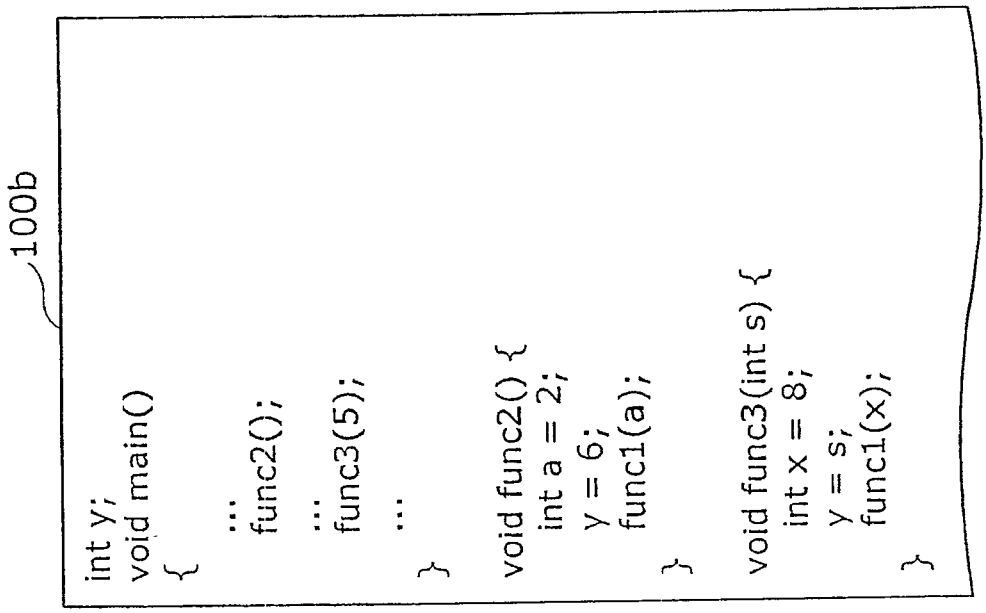
100b
int y;
void main()
{
...
func2();
...
func3(5);
...
}
void func2() {
int a = 2;
y = 6;
func1(a);
}
void func3(int s) {
int x = 8;
y = s;
func1(x);
}

图4B

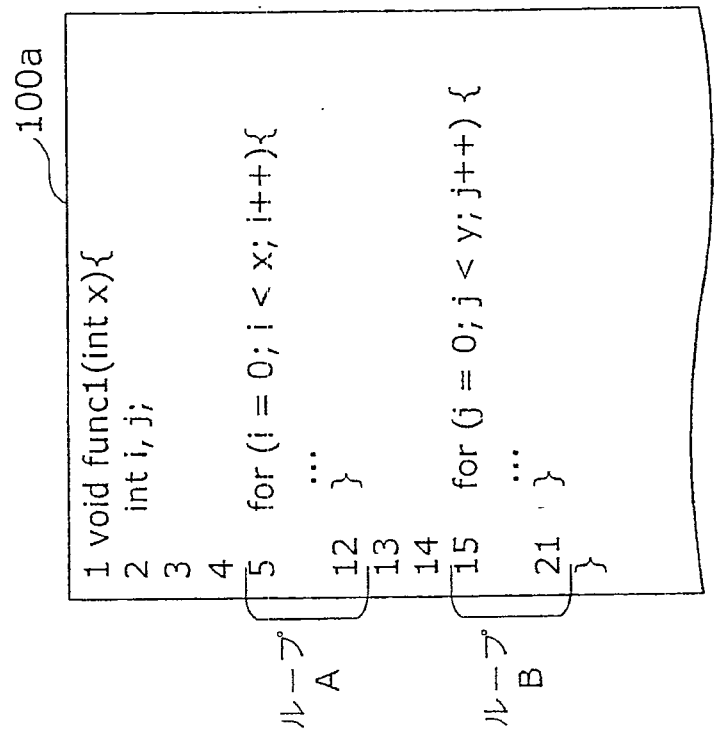
100a
1 void func1(int x){
2 int i, j;
3
4
5 for (i = 0; i < x; i++){
...
12 }
13
14
15 for (j = 0; j < y; j++) {
...
21 }
}
ループA
ループB

图4A

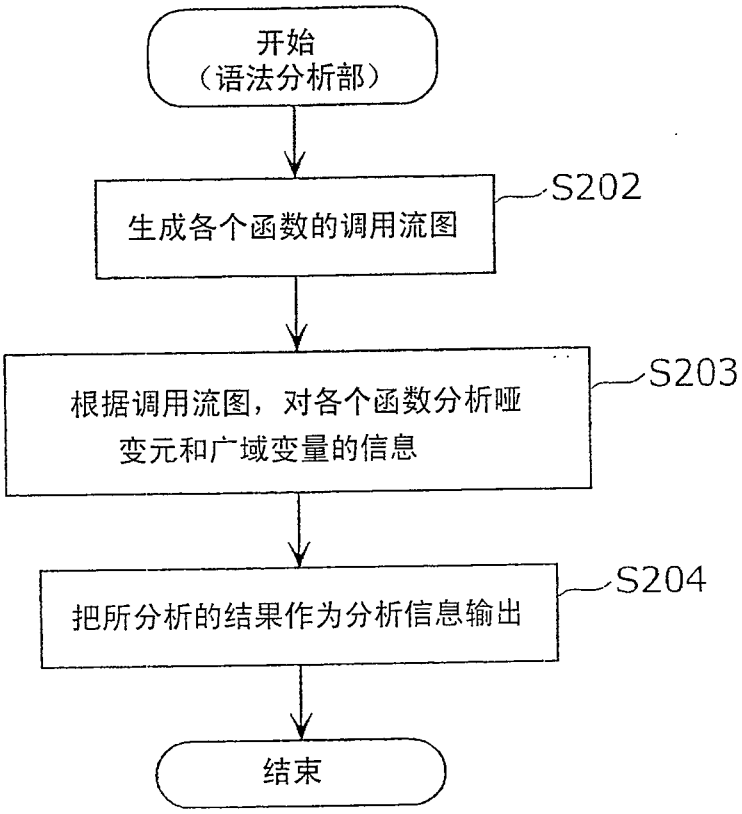
开始
（语法分析部）
生成各个函数的调用流图
S202
根据调用流图，对各个函数分析哑
变元和广域变量的信息
S203
把所分析的结果作为分析信息输出
S204
结束

图5

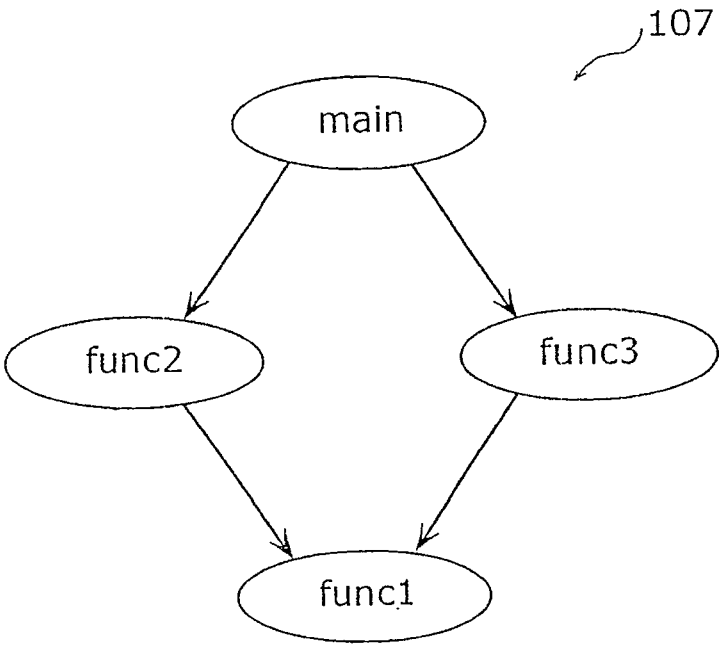
107
main
func2
func3
func1

图6

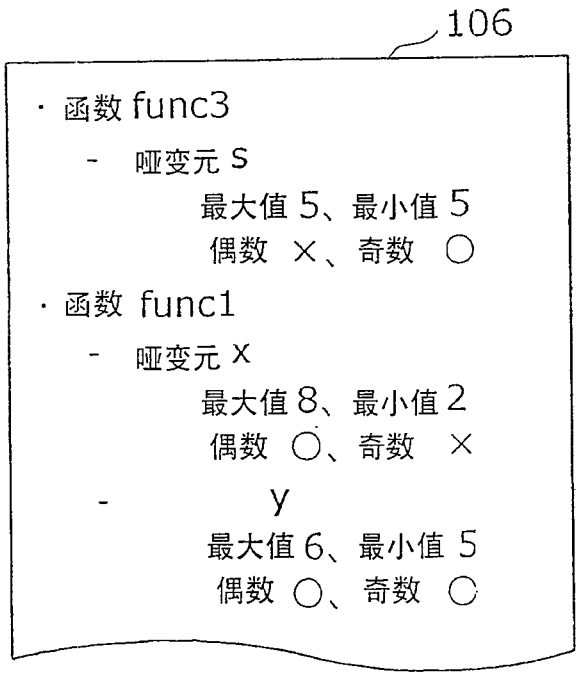
106
· 函数 func3
- 哑变元 s
最大值 5、最小值 5
偶数 ×、奇数 ○
· 函数 func1
- 哑变元 x
最大值 8、最小值 2
偶数 ○、奇数 ×
- y
最大值 6、最小值 5
偶数 ○、奇数 ○

图7

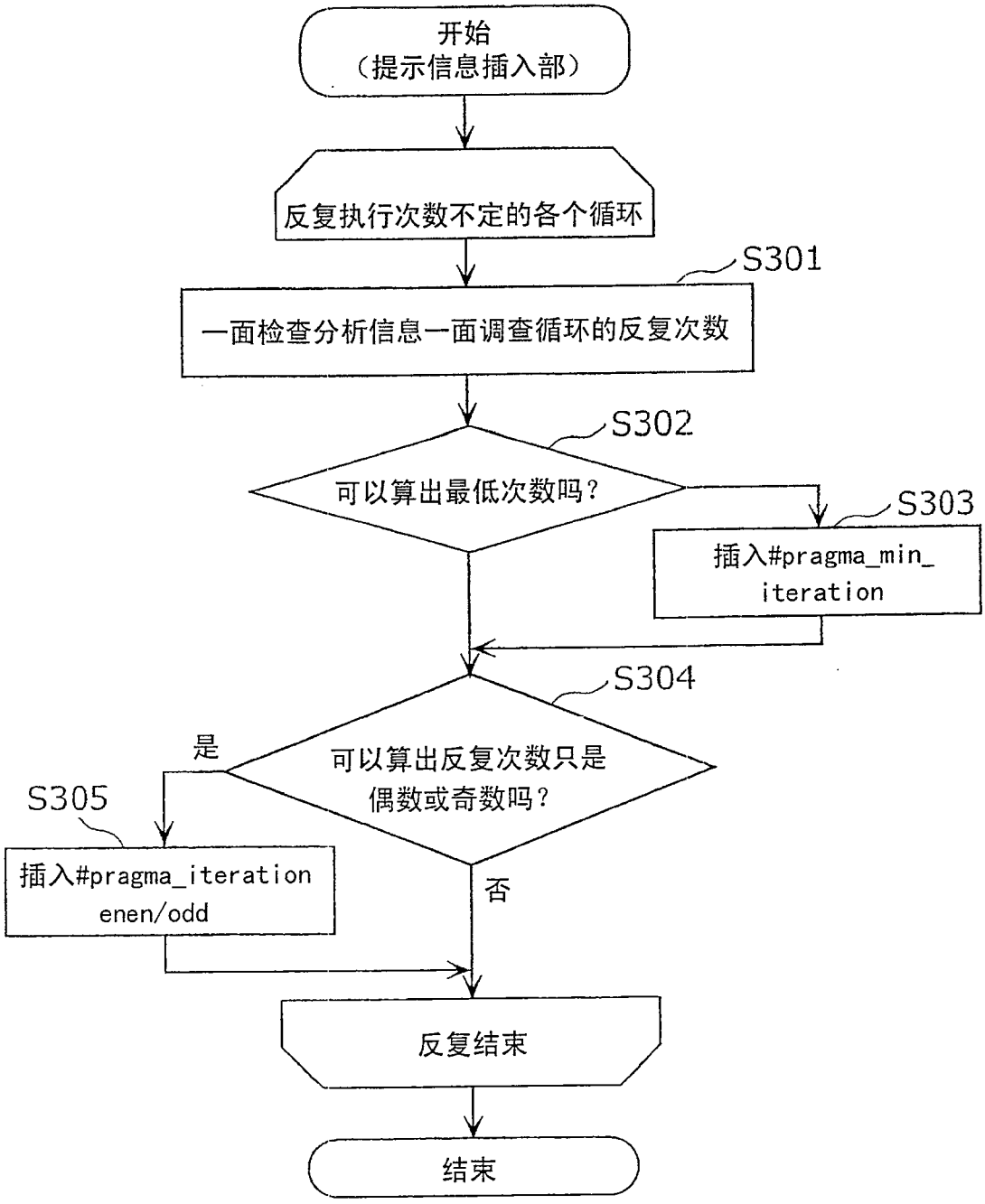
开始
（提示信息插入部）
反复执行次数不定的各个循环
S301
一面检查分析信息一面调查循环的反复次数
S302
可以算出最低次数吗？
S303
插入#pragma_min_
iteration
S304
可以算出反复次数只是
偶数或奇数吗？
是
S305
插入#pragma_iteration
enen/odd
否
反复结束
结束

图8

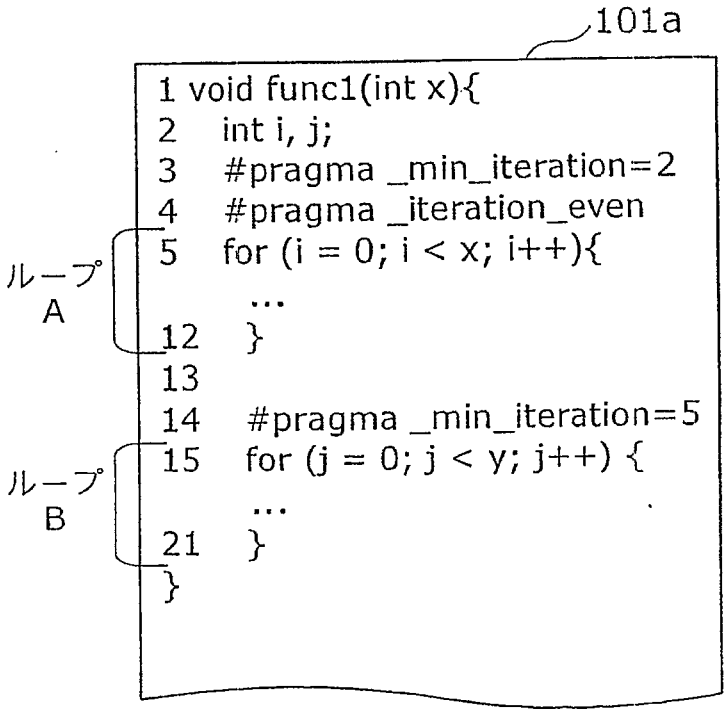
101a
1 void func1(int x){
2 int i, j;
3 #pragma _min_iteration=2
4 #pragma _iteration_even
5 for (i = 0; i < x; i++){
...
12 }
13
14 #pragma _min_iteration=5
15 for (j = 0; j < y; j++) {
...
21 }
}
ループA
ループB

图9

100a

```
void func1( ){
   #pragma _align_object=4 a,b
   short a, b;
   short *p1, *p2;
   .....
   p1 = &a;
   p2 = &b;
   func3(&a, p1, p2);
   .....
}
void func2( ) {
#pragma _align_object=8 x
#pragma _align_object=4 y
   short x, y, z;
   func3(&x, &y, &z);
}
void func3(short *q1,
           short *q2,
           short *q3){
   .....
}
```

图10A

100b

```
void main( ) {
   ...
   func1()
   ...
   func2()
   ...
}
```

图10B

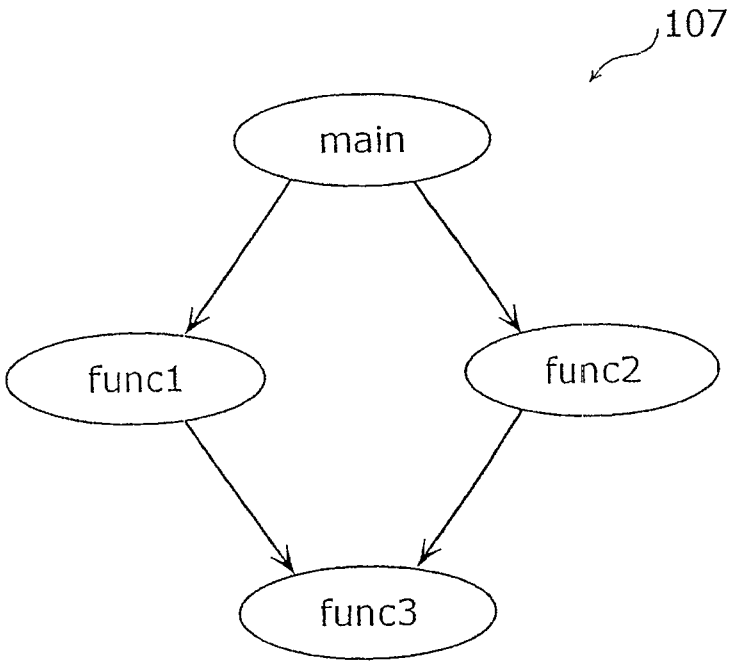
107
main
func1
func2
func3

图11

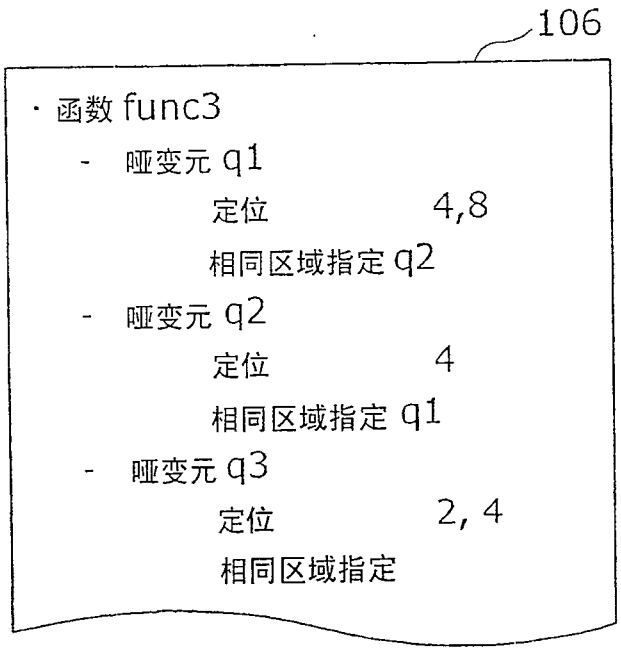
106
· 函数 func3
- 哑变元 q1
定位 4,8
相同区域指定 q2
- 哑变元 q2
定位 4
相同区域指定 q1
- 哑变元 q3
定位 2, 4
相同区域指定

图12

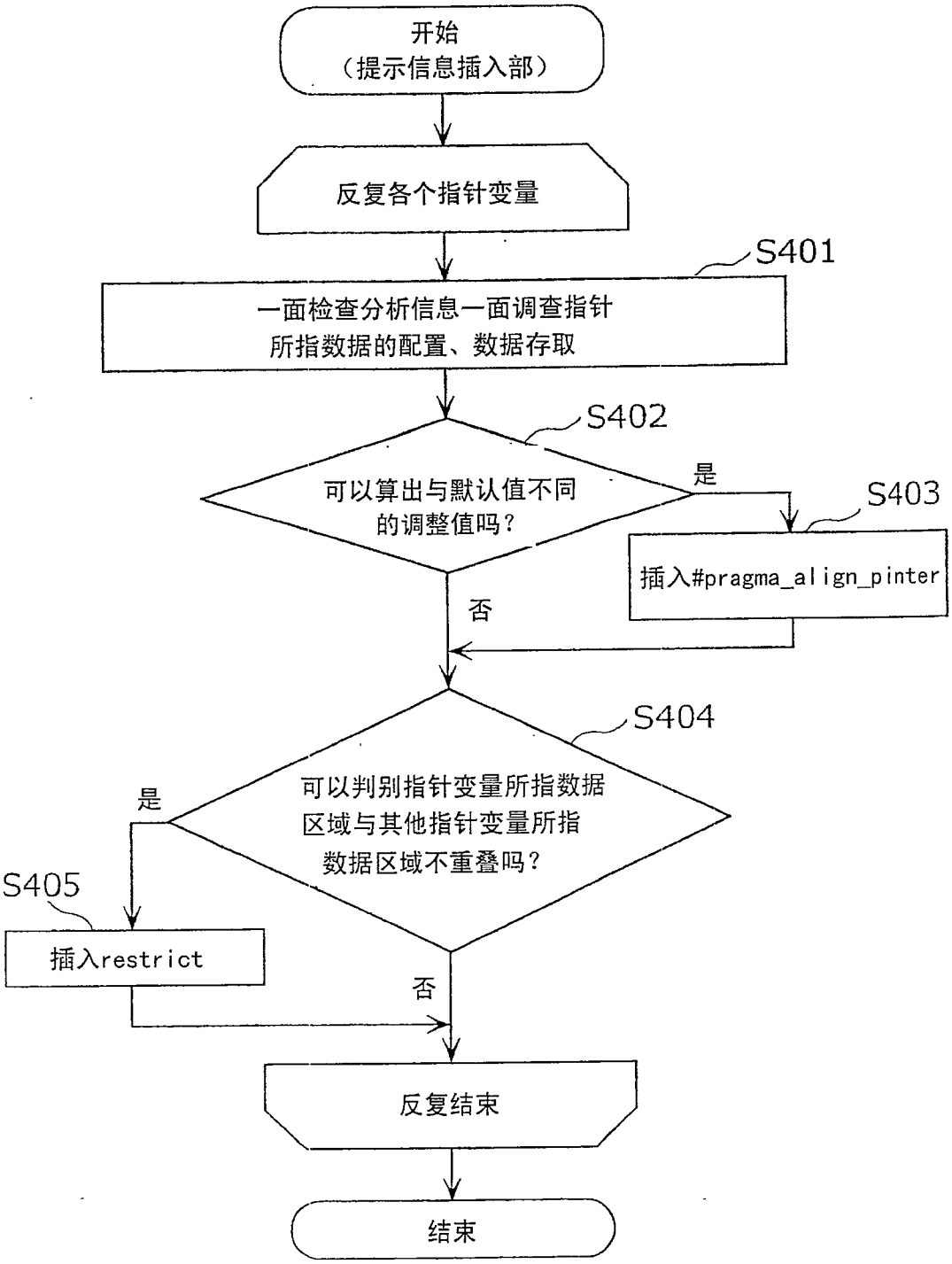
开始
（提示信息插入部）
反复各个指针变量
S401
一面检查分析信息一面调查指针
所指数据的配置、数据存取
S402
可以算出与默认值不同
的调整值吗？
是
S403
插入#pragma_align_pinter
否
S404
可以判别指针变量所指数据
区域与其他指针变量所指
数据区域不重叠吗？
是
S405
插入restrict
否
反复结束
结束

图13

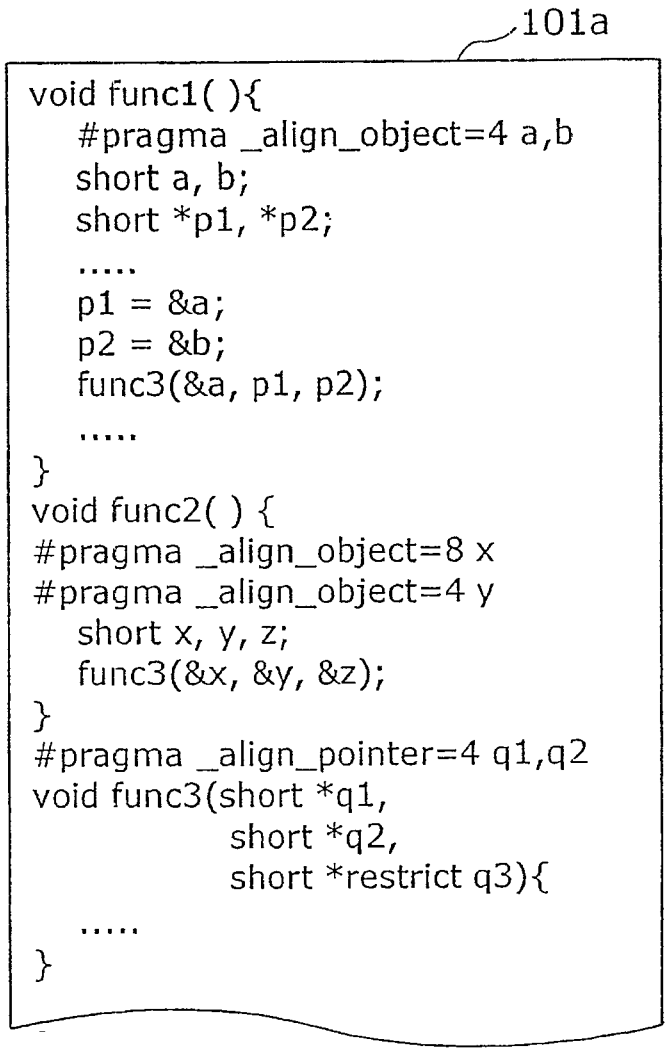
101a
void func1( ){
#pragma _align_object=4 a,b
short a, b;
short *p1, *p2;
.....
p1 = &a;
p2 = &b;
func3(&a, p1, p2);
.....
}
void func2( ) {
#pragma _align_object=8 x
#pragma _align_object=4 y
short x, y, z;
func3(&x, &y, &z);
}
#pragma _align_pointer=4 q1,q2
void func3(short *q1,
short *q2,
short *restrict q3){
.....
}

图14

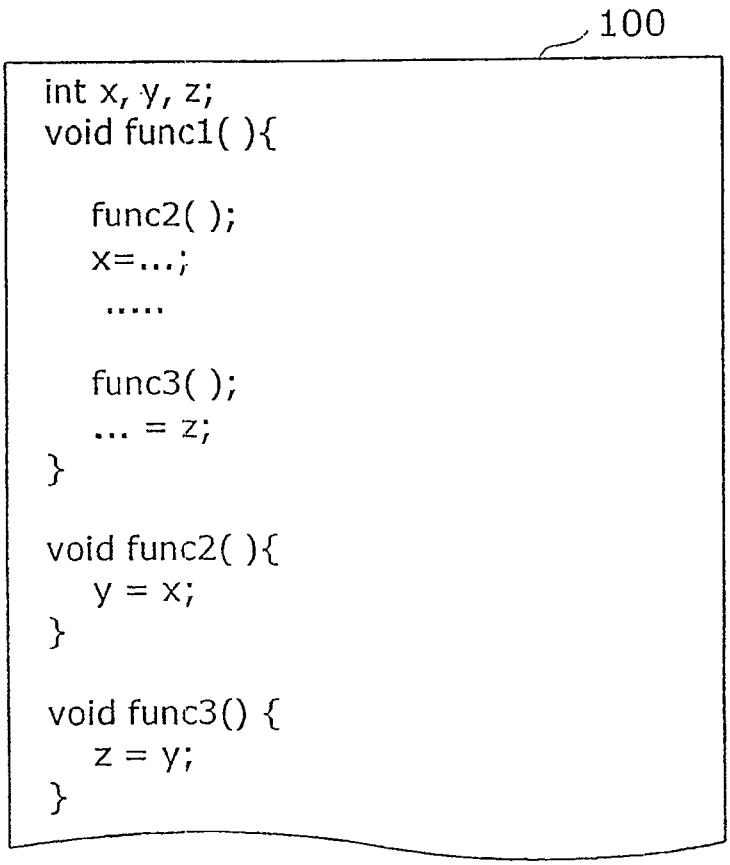
100
int x, y, z;
void func1( ){
func2( );
x=...;
.....
func3( );
... = z;
}
void func2( ){
y = x;
}
void func3() {
z = y;
}

图15

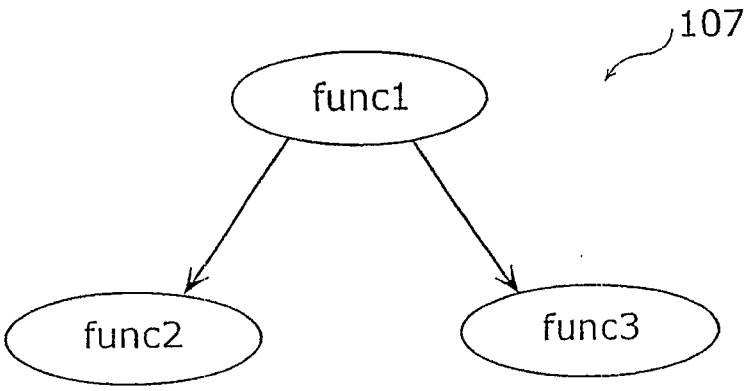
107
func1
func2
func3

图16

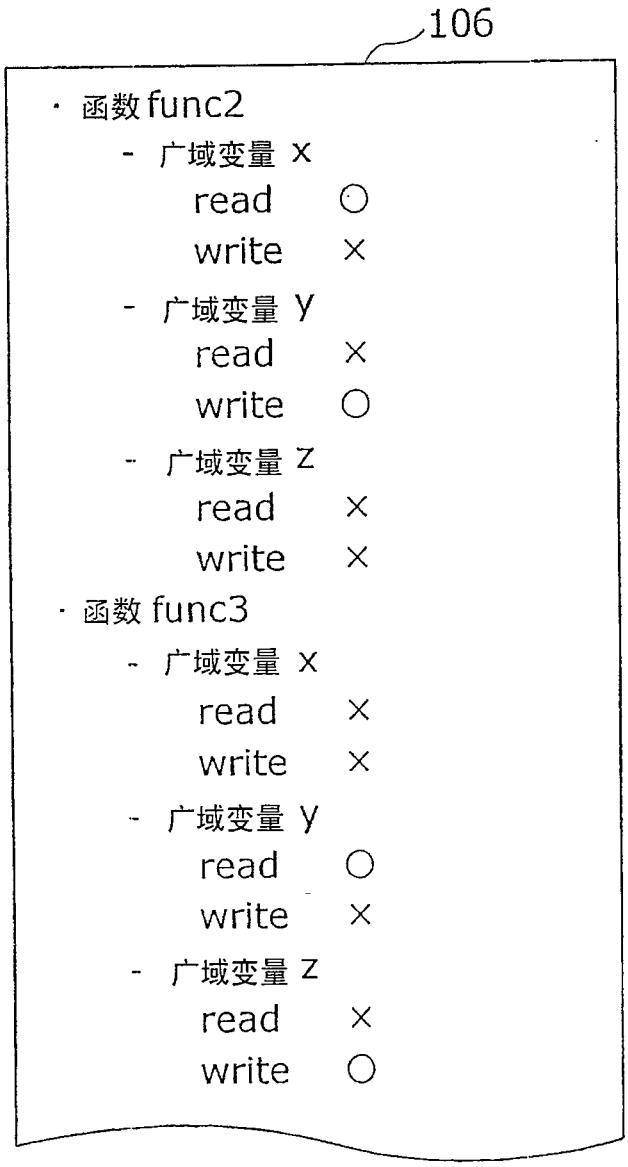
106
· 函数 func2
- 广域变量 x
read ○
write ×
- 广域变量 y
read ×
write ○
- 广域变量 z
read ×
write ×
· 函数 func3
- 广域变量 x
read ×
write ×
- 广域变量 y
read ○
write ×
- 广域变量 z
read ×
write ○

图17

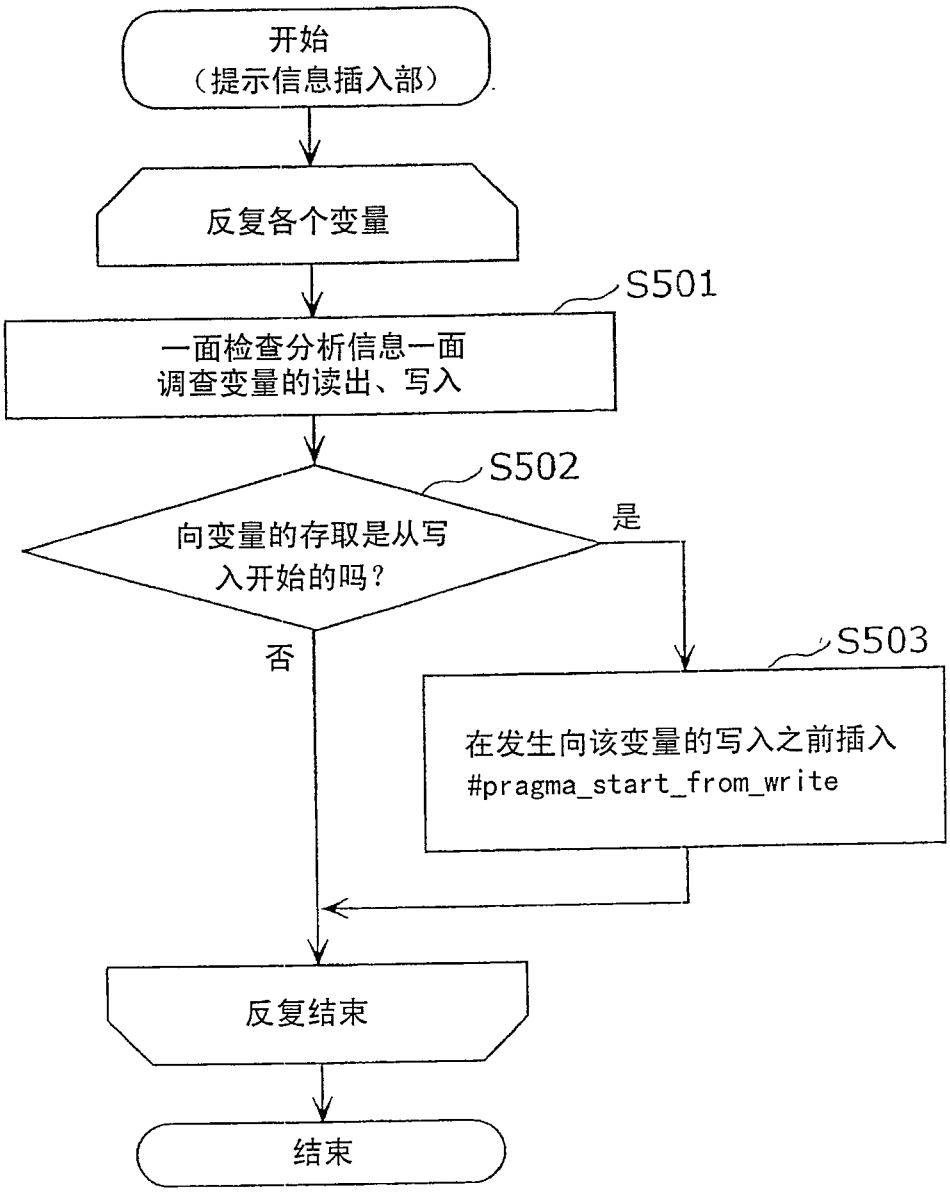
开始
（提示信息插入部）
反复各个变量
S501
一面检查分析信息一面
调查变量的读出、写入
S502
向变量的存取是从写
入开始的吗？
是
否
S503
在发生向该变量的写入之前插入
#pragma_start_from_write
反复结束
结束

图18

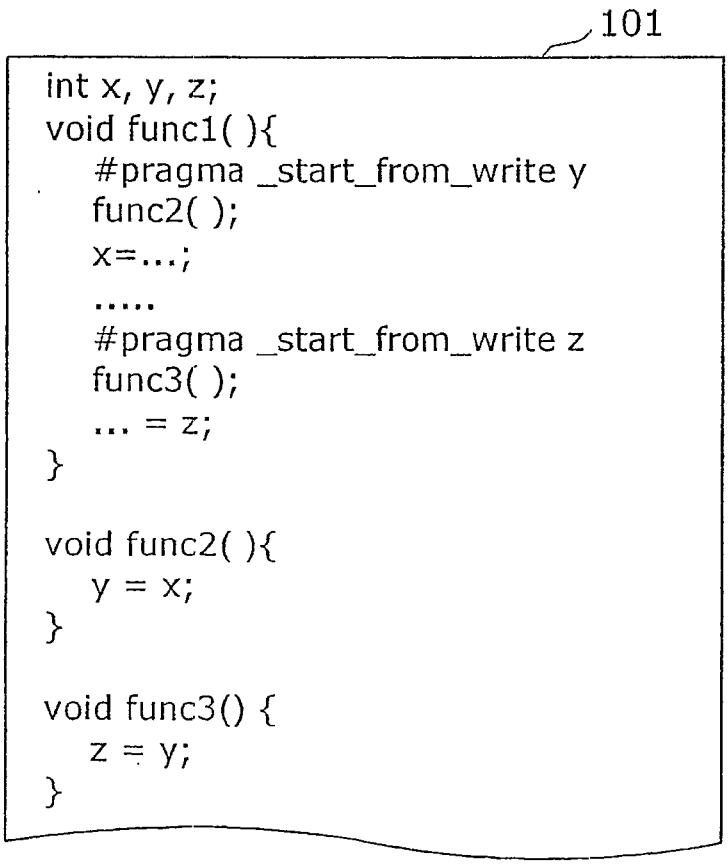
101
int x, y, z;
void func1( ){
#pragma _start_from_write y
func2( );
x=...;
.....
#pragma _start_from_write z
func3( );
... = z;
}
void func2( ){
y = x;
}
void func3() {
z = y;
}

图19

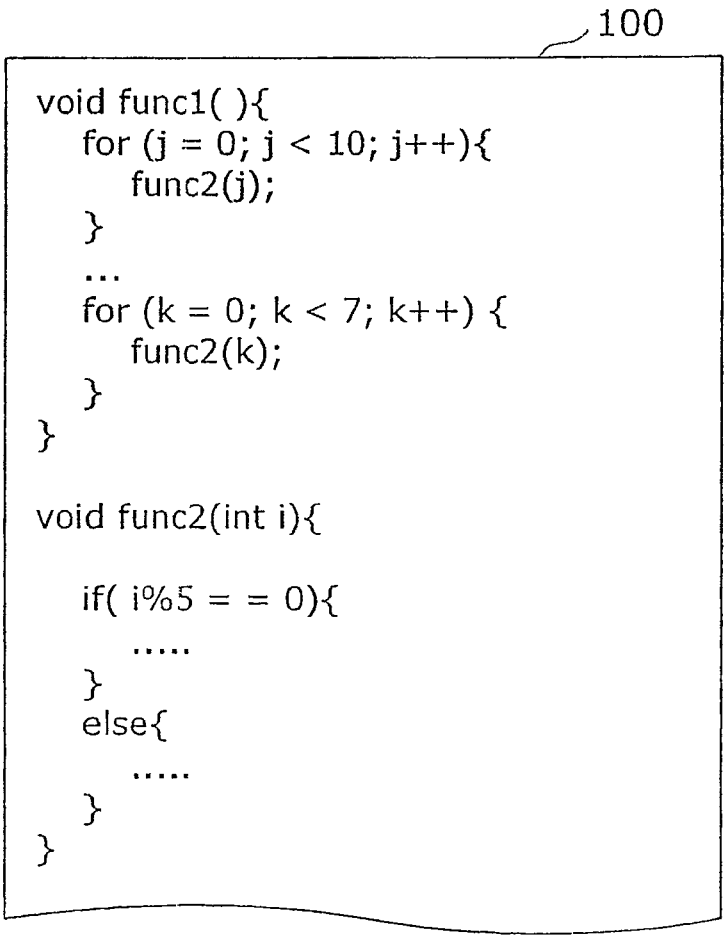
100
void func1( ){
for (j = 0; j < 10; j++){
func2(j);
}
...
for (k = 0; k < 7; k++) {
func2(k);
}
}
void func2(int i){
if(i%5 = = 0){
.....
}
else{
.....
}
}

图20

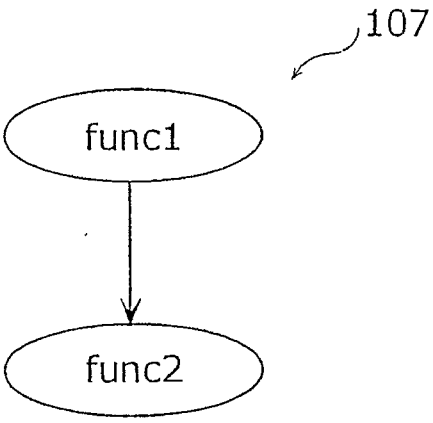
107
func1
func2

图21

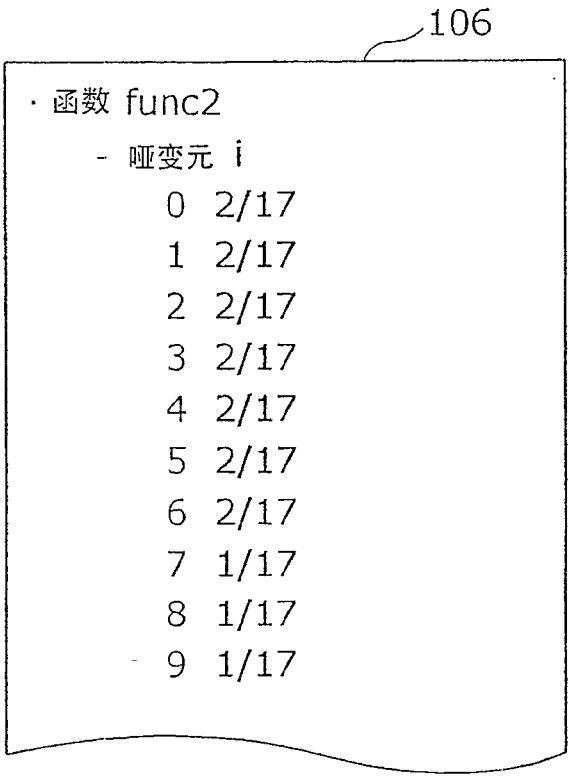
106
· 函数 func2
- 哑变元 i
0 2/17
1 2/17
2 2/17
3 2/17
4 2/17
5 2/17
6 2/17
7 1/17
8 1/17
9 1/17

图22

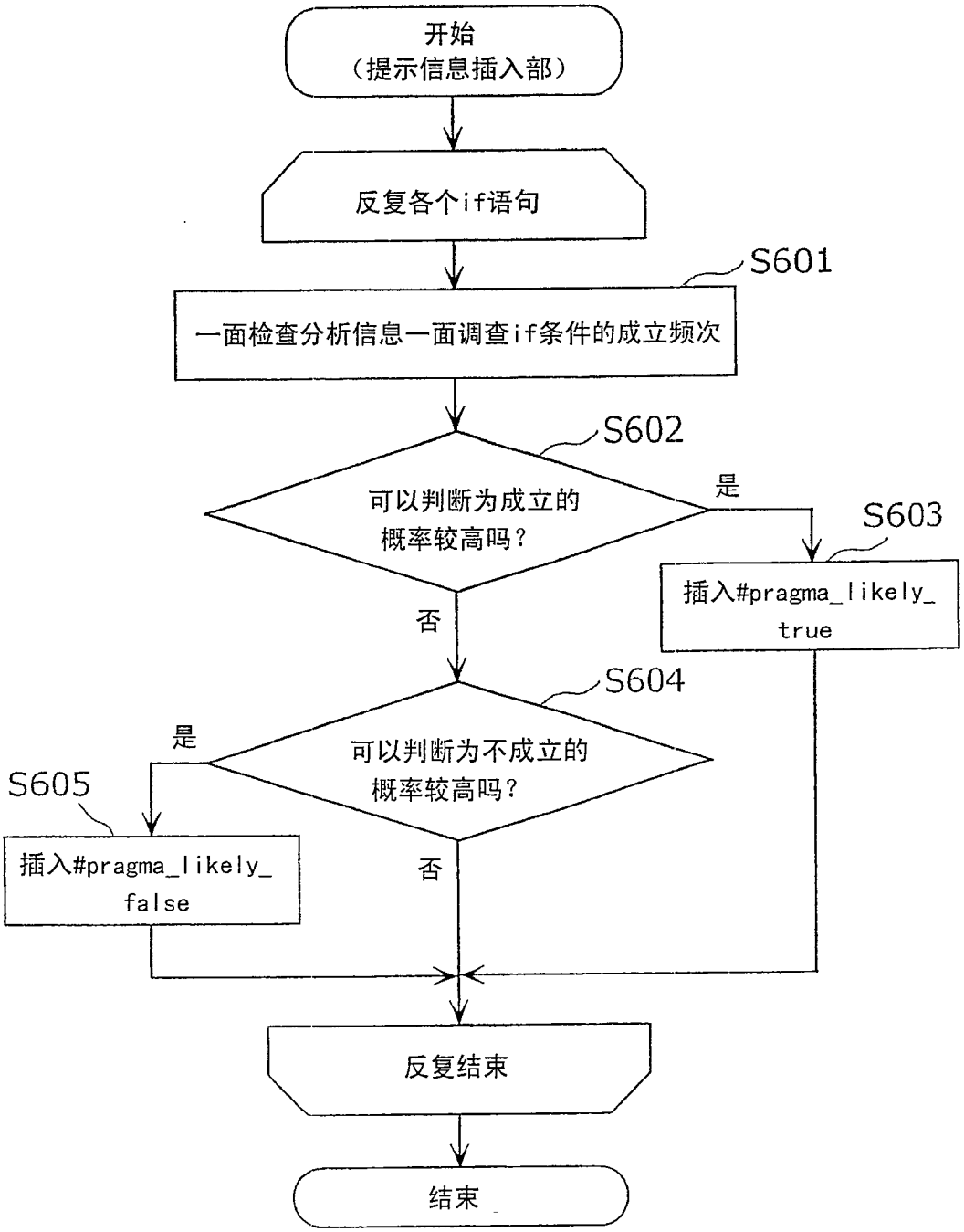
开始
（提示信息插入部）
反复各个if语句
S601
一面检查分析信息一面调查if条件的成立频次
S602
可以判断为成立的
概率较高吗？
是
S603
插入#pragma_likely_
true
否
S604
可以判断为不成立的
概率较高吗？
是
S605
插入#pragma_likely_
false
否
反复结束
结束

图23

101

```
void func1( ){
   for (j = 0; j < 10; j++){
      func2(j);
   }
   ...
   for (k = 0; k < 7; k++) {
      func2(k);
   }
}

void func2(int i){
   #pragma _likely_false
   if( i%5 = = 0){
      .....
   }
   else{
      .....
   }
}
```

图24

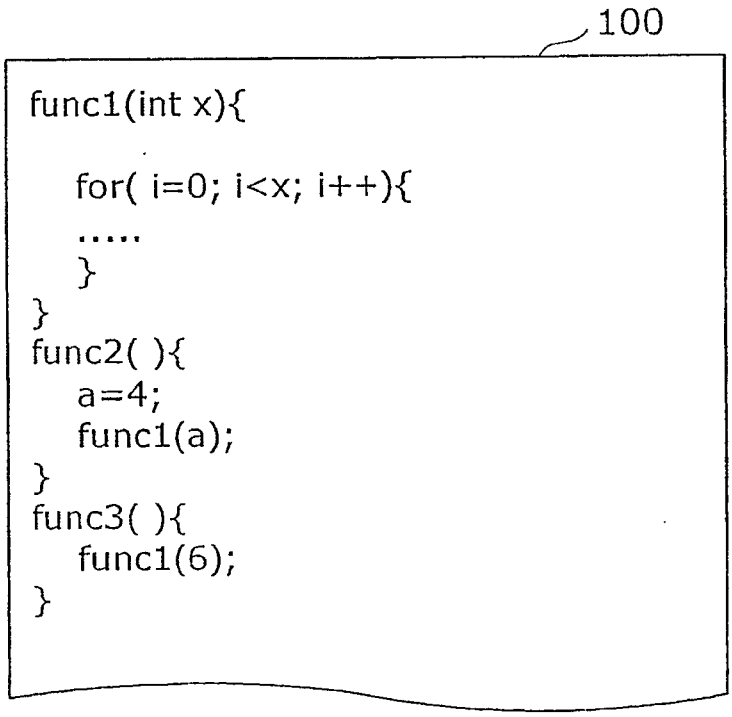

图25

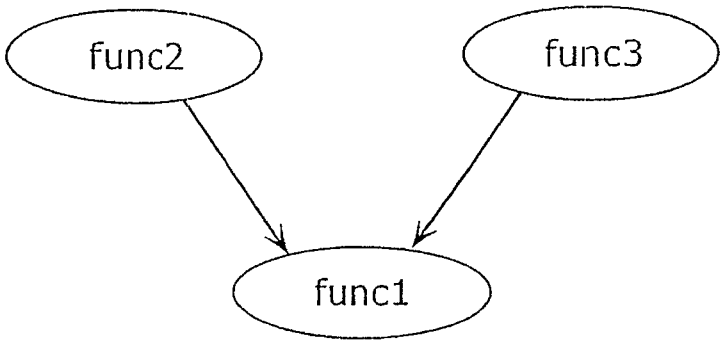

图26

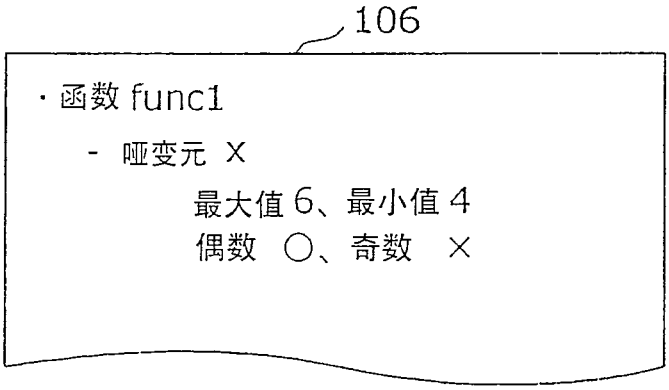

图27

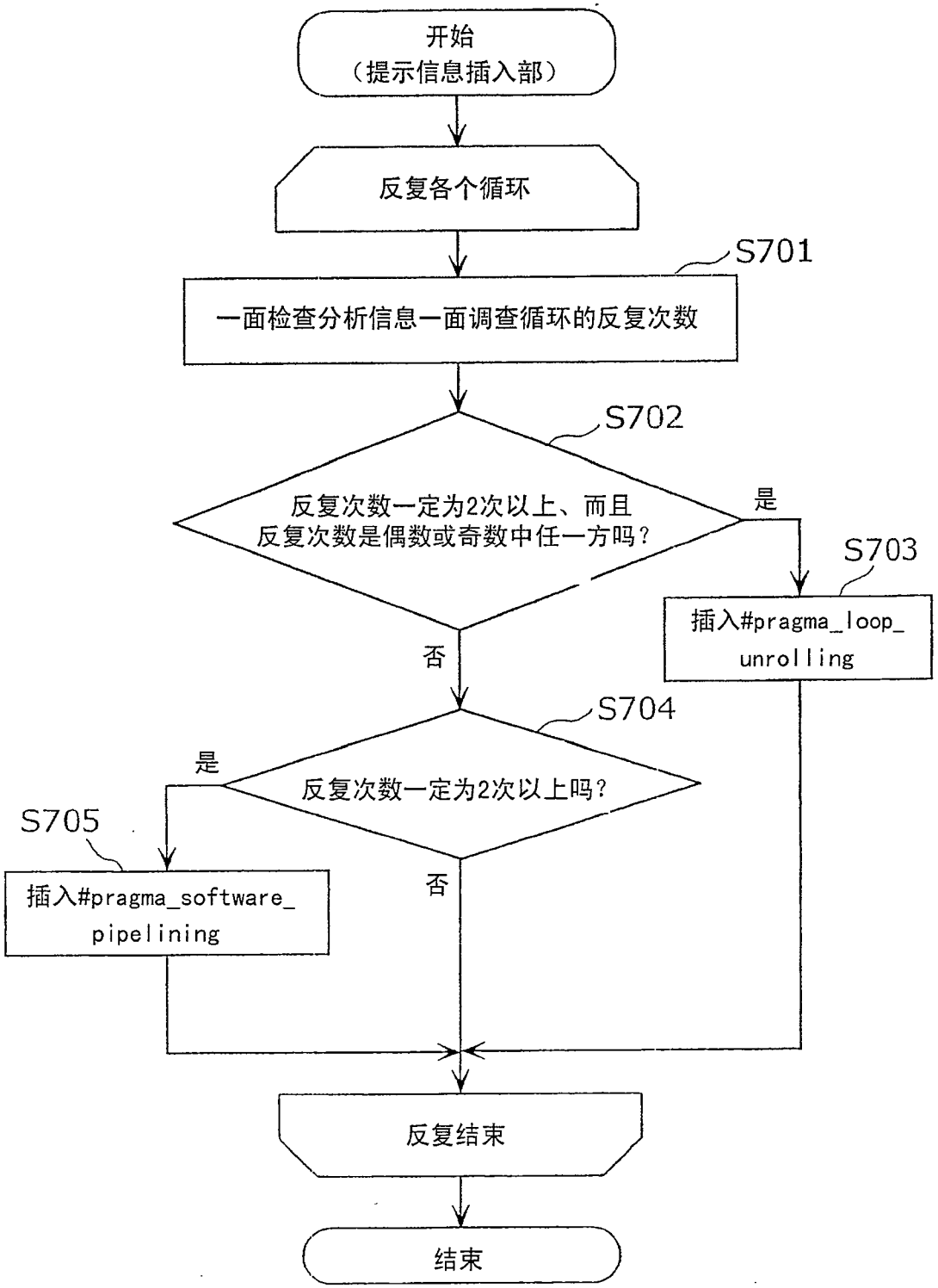
开始
（提示信息插入部）
反复各个循环
S701
一面检查分析信息一面调查循环的反复次数
S702
反复次数一定为2次以上、而且
反复次数是偶数或奇数中任一方吗？
是
S703
插入#pragma_loop_
unrolling
否
S704
是
反复次数一定为2次以上吗？
S705
插入#pragma_software_
pipelining
否
反复结束
结束

图28

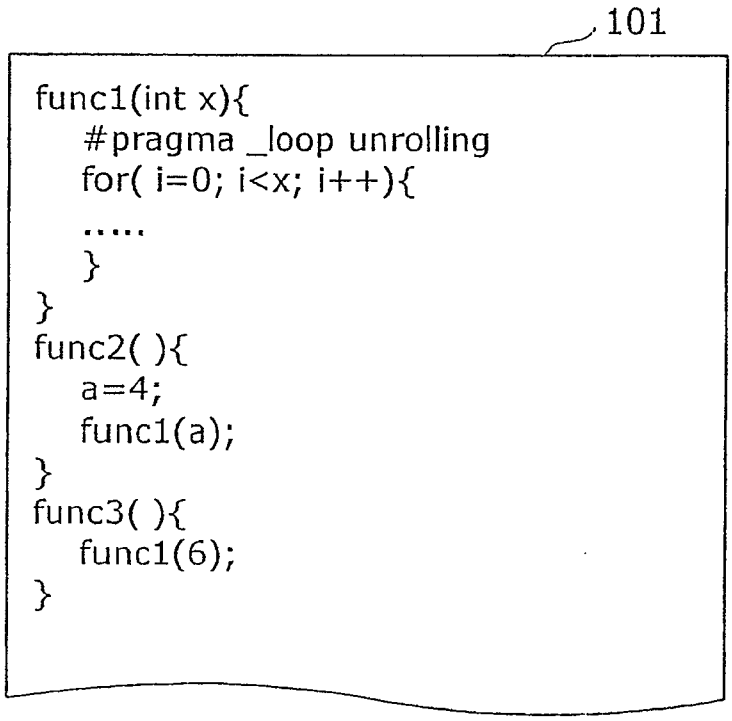
101
func1(int x){
#pragma _loop unrolling
for(i=0; i<x; i++){
.....
}
}
func2( ){
a=4;
func1(a);
}
func3( ){
func1(6);
}

图29

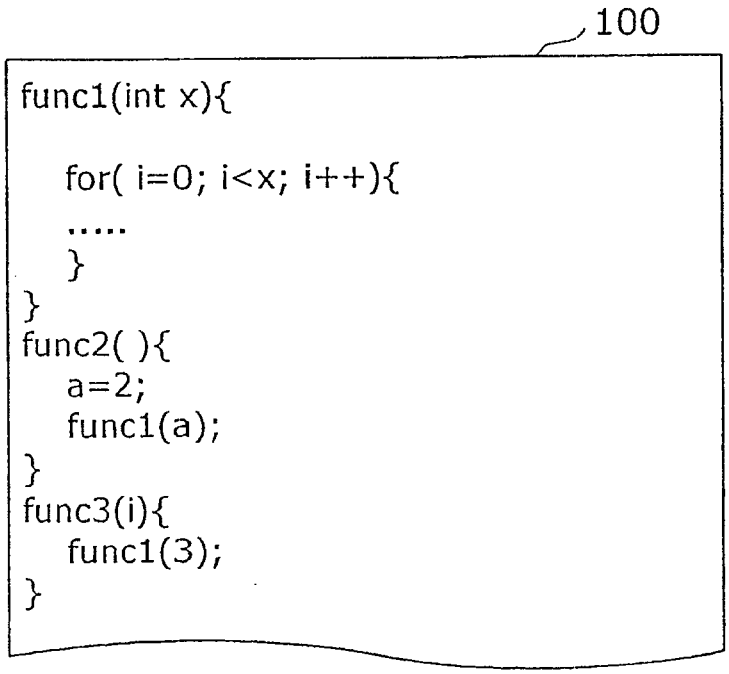

图30

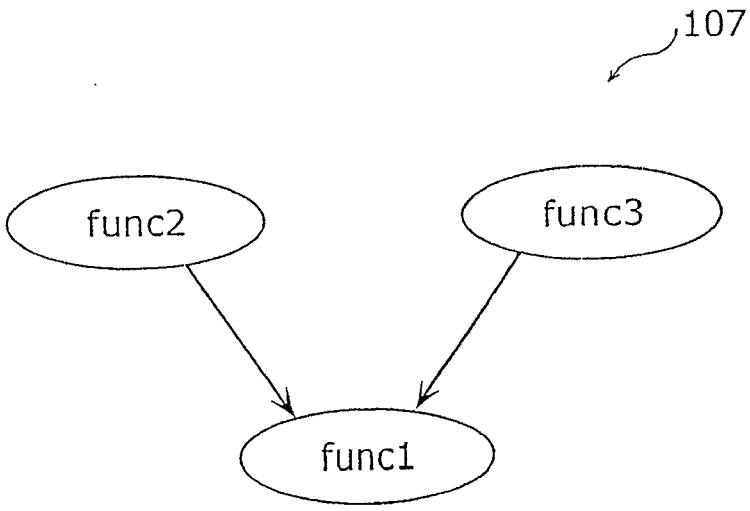

图31

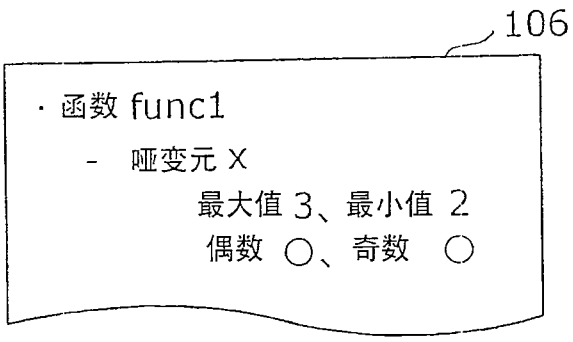

图32

101

```
func1(int x){
    #pragma _software_pipelining
    for( i=0; i<x; i++){
    .....
    }
}
func2( ){
    a=2;
    func1(a);
}
```

图33

100

```
void func1( ){
    #pragma _align_object=4 a
    short a[100]={...};
     .....
    func2(a);
     .....
}
void func2(short *a){
        a[0]=...;
        a[1]=...;
          .....
}
```

图34

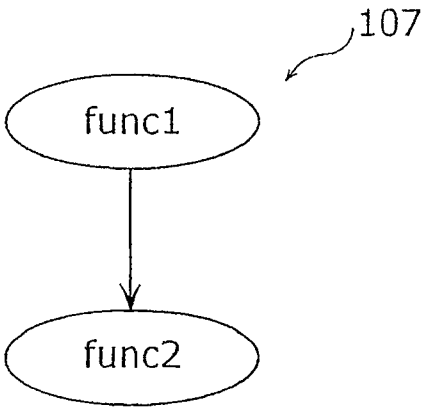

图35

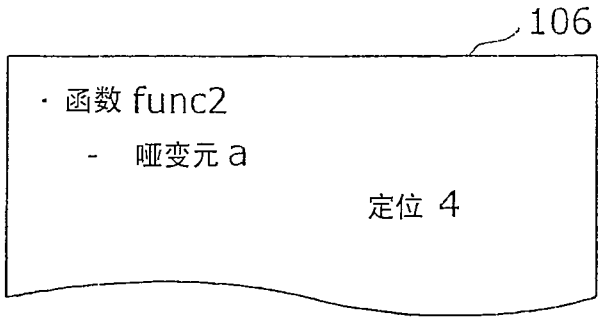

图36

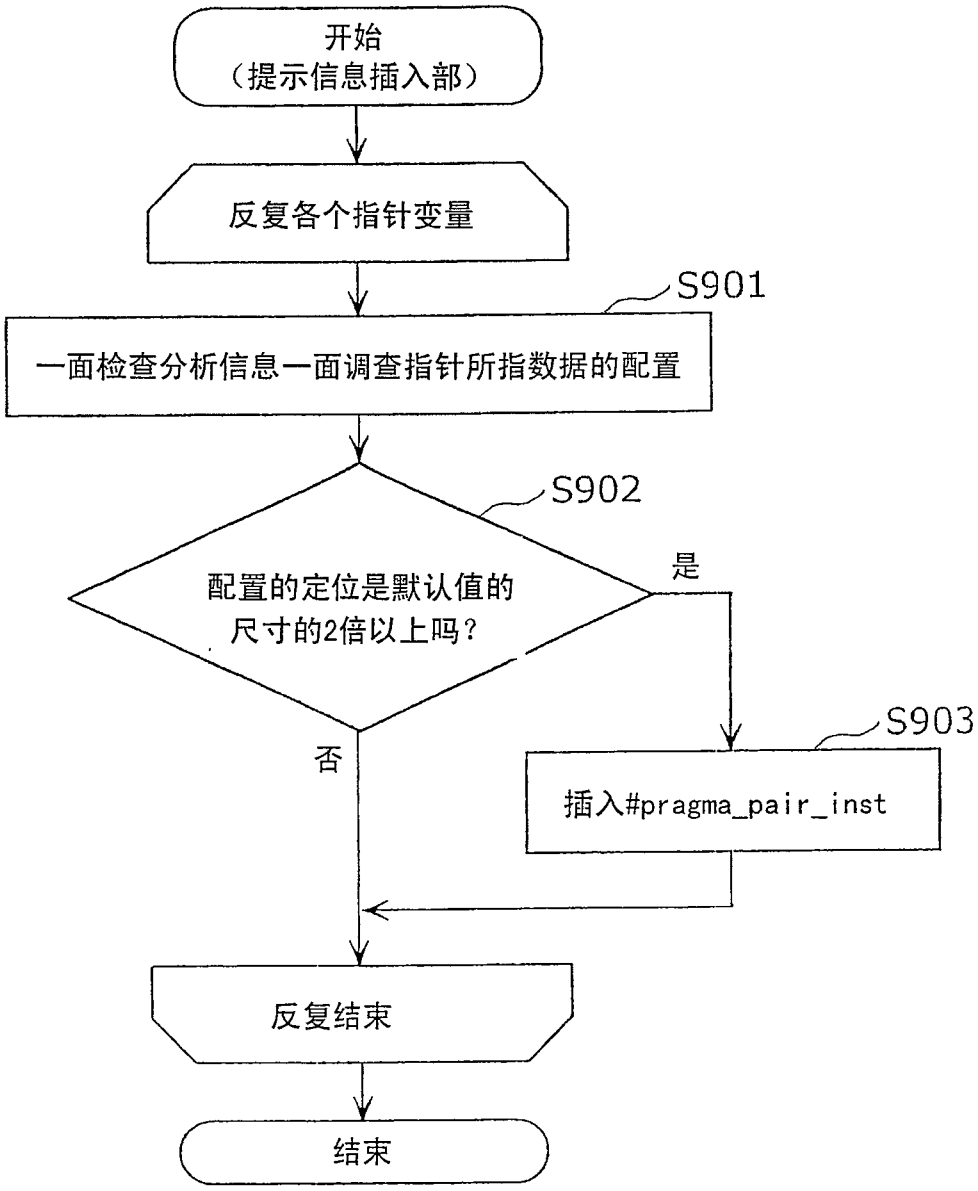
开始
（提示信息插入部）
反复各个指针变量
S901
一面检查分析信息一面调查指针所指数据的配置
S902
配置的定位是默认值的
尺寸的2倍以上吗？
是
否
S903
插入#pragma_pair_inst
反复结束
结束

图37

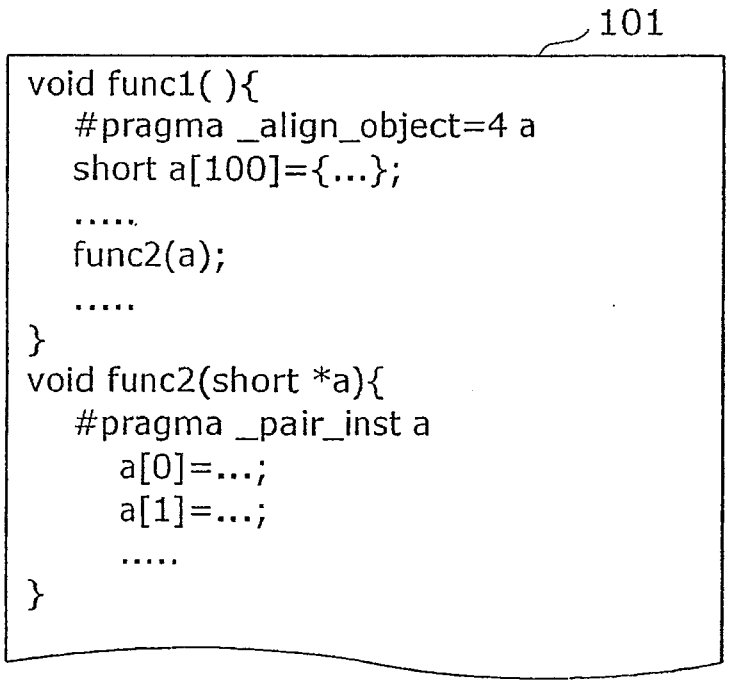
101
void func1( ){
#pragma _align_object=4 a
short a[100]={...};
.....
func2(a);
.....
}
void func2(short *a){
#pragma _pair_inst a
a[0]=...;
a[1]=...;
.....
}

图38

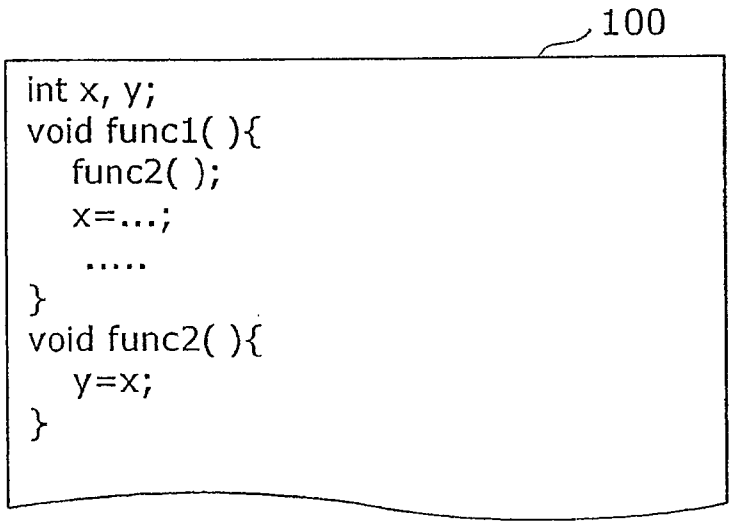

图39

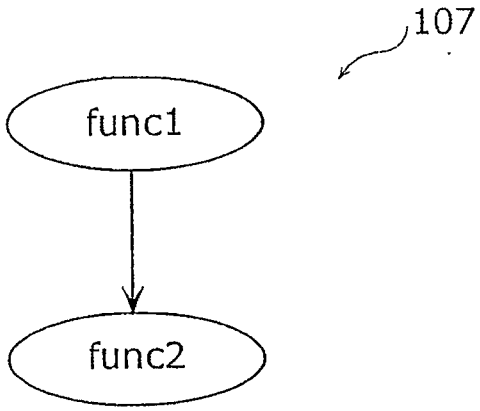
107
func1
func2

图40

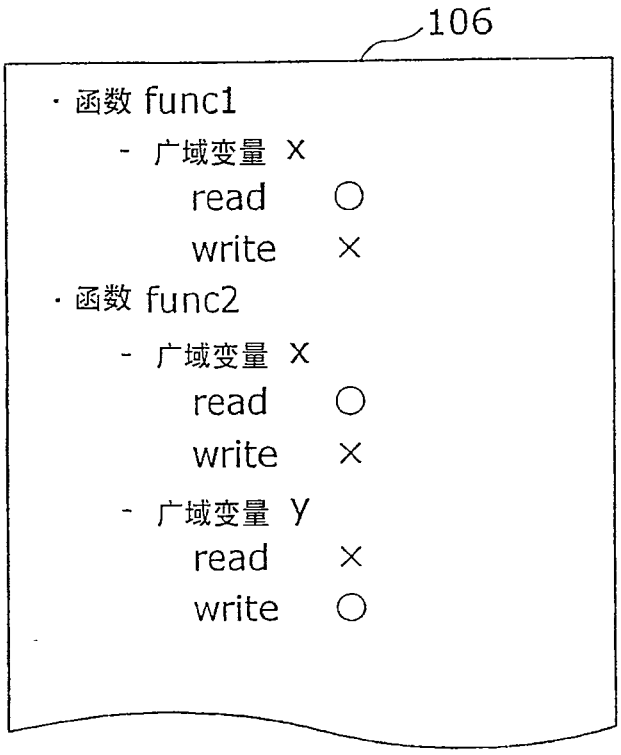
106
· 函数 func1
- 广域变量 x
read ○
write ×
· 函数 func2
- 广域变量 x
read ○
write ×
- 广域变量 y
read ×
write ○

图41

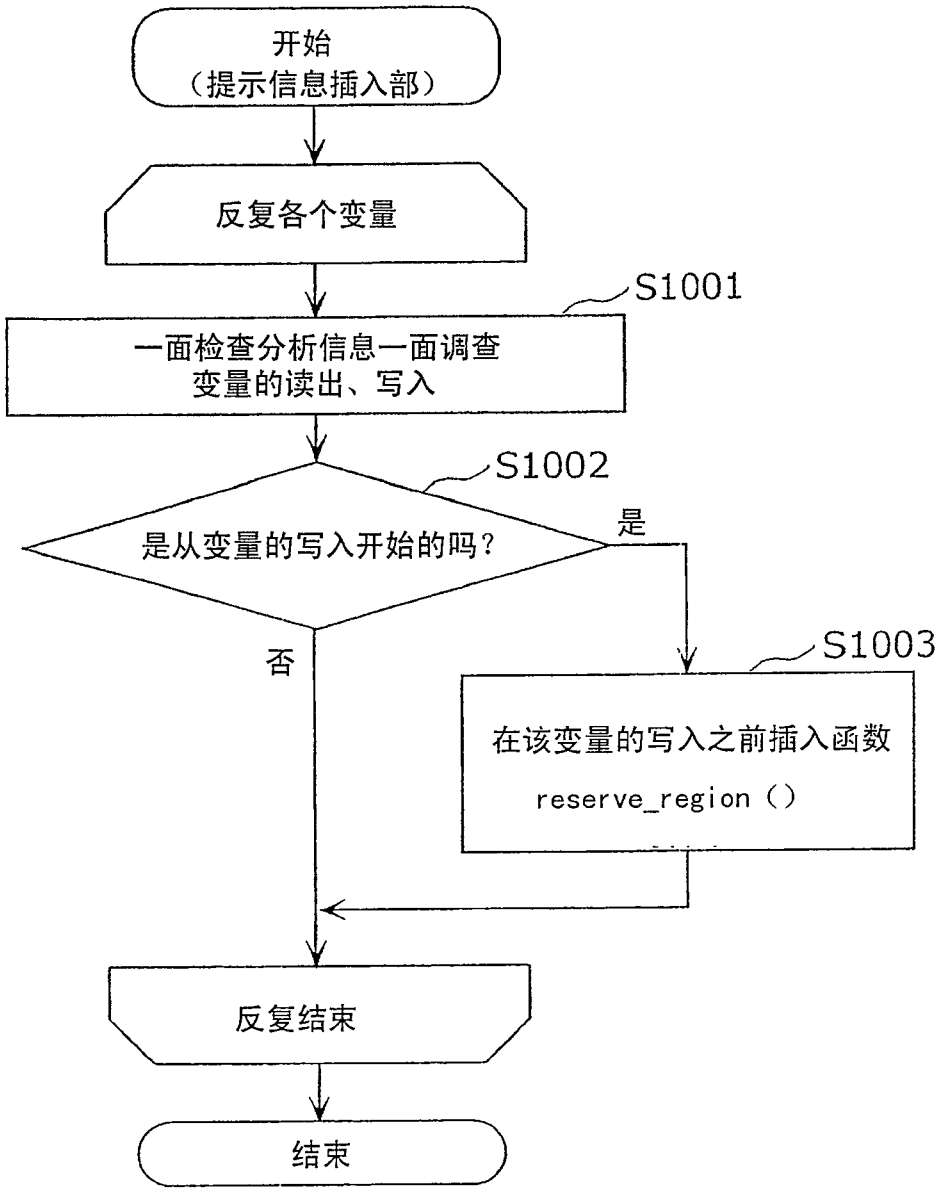
开始
（提示信息插入部）
反复各个变量
S1001
一面检查分析信息一面调查
变量的读出、写入
S1002
是从变量的写入开始的吗？
是
否
S1003
在该变量的写入之前插入函数
reserve_region（）
反复结束
结束

图42

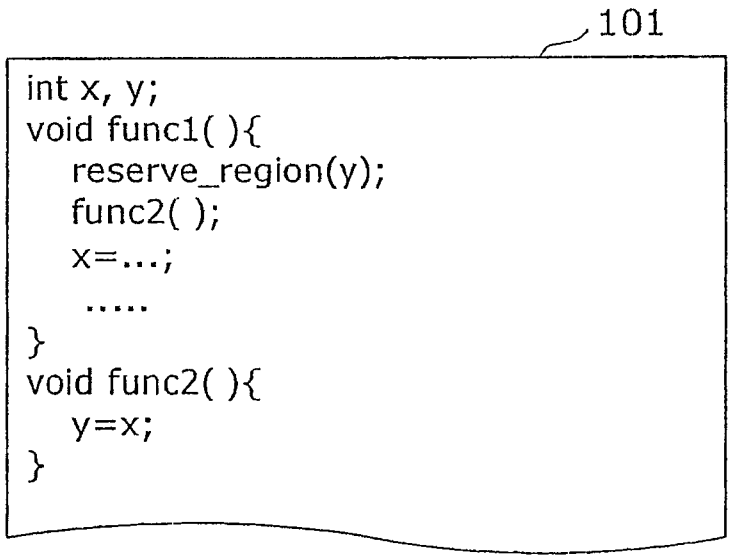

图43

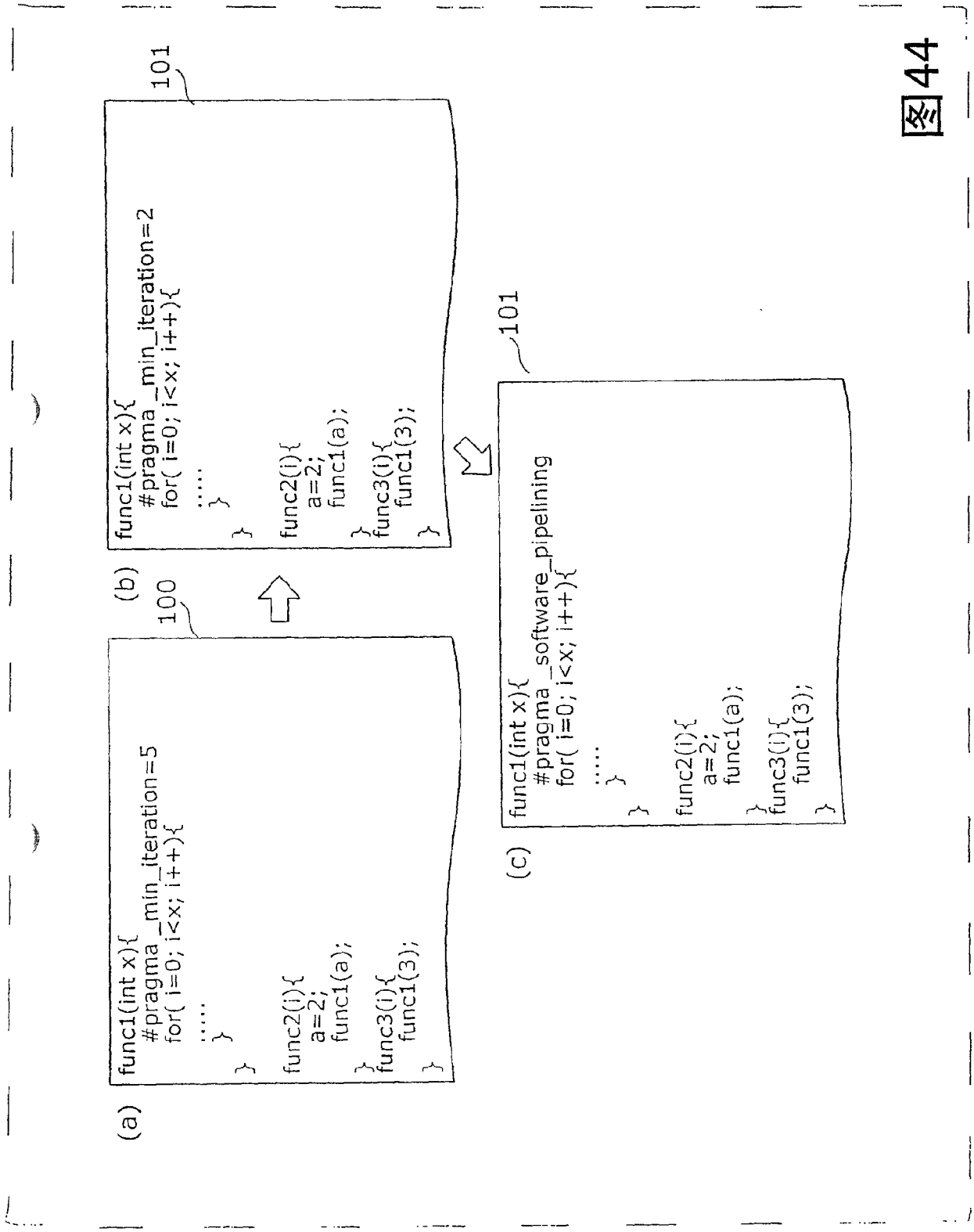
(a)
100
func1(int x){
#pragma _min_iteration=5
for(i=0; i<x; i++){
.....
}
}
func2(i){
a=2;
func1(a);
}
func3(i){
func1(3);
}
(b)
101
func1(int x){
#pragma _min_iteration=2
for(i=0; i<x; i++){
.....
}
}
func2(i){
a=2;
func1(a);
}
func3(i){
func1(3);
}
(c)
101
func1(int x){
#pragma _software_pipelining
for(i=0; i<x; i++){
.....
}
}
func2(i){
a=2;
func1(a);
}
func3(i){
func1(3);
}

图44

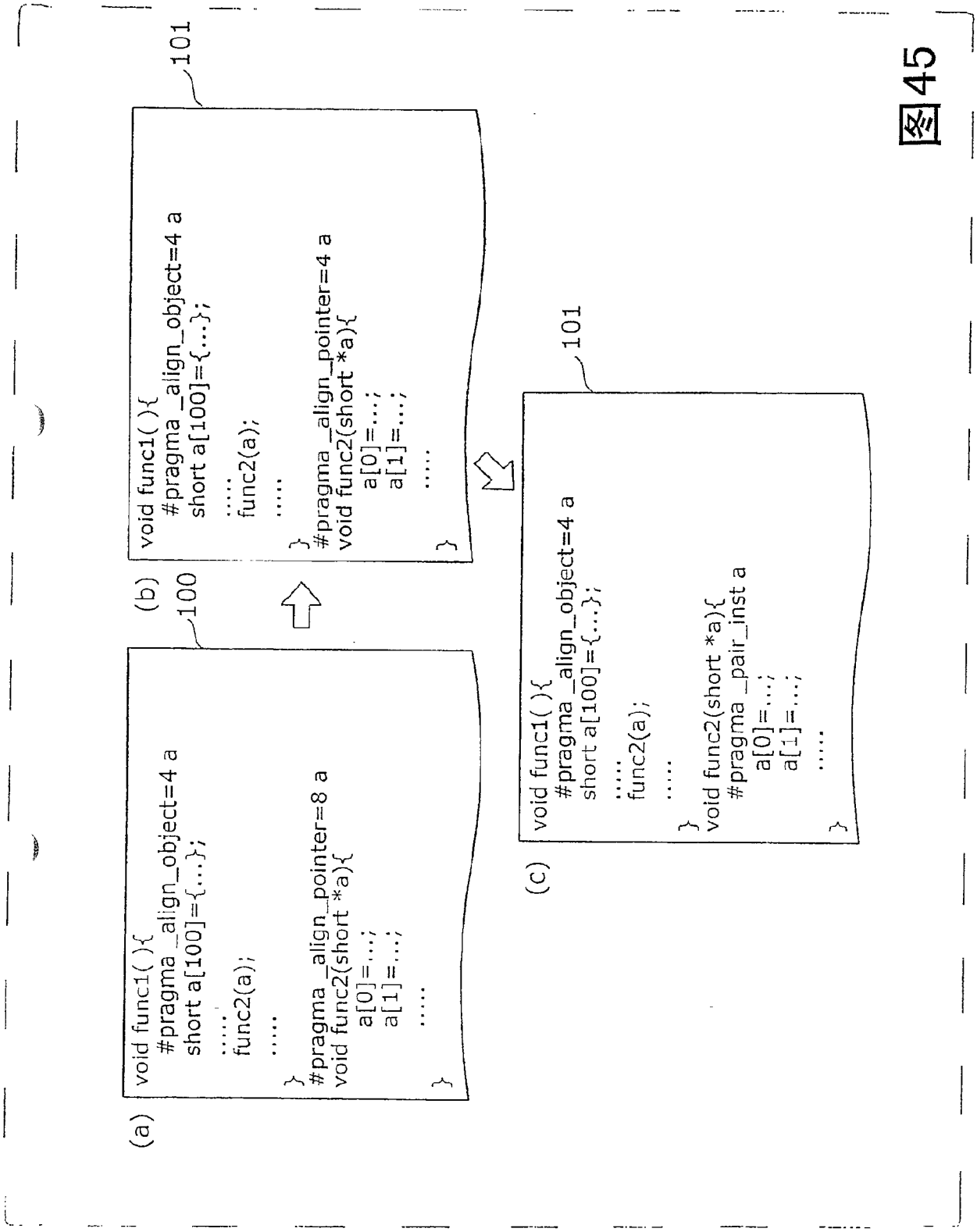
(a)
100
void func1( ){
#pragma _align_object=4 a
short a[100]={...};
.....
func2(a);
.....
}
#pragma _align_pointer=8 a
void func2(short *a){
a[0]=...;
a[1]=...;
.....
}
(b)
101
void func1( ){
#pragma _align_object=4 a
short a[100]={...};
.....
func2(a);
.....
}
#pragma _align_pointer=4 a
void func2(short *a){
a[0]=...;
a[1]=...;
.....
}
(c)
101
void func1( ){
#pragma _align_object=4 a
short a[100]={...};
.....
func2(a);
.....
}
void func2(short *a){
#pragma _pair_inst a
a[0]=...;
a[1]=...;
.....
}

图45

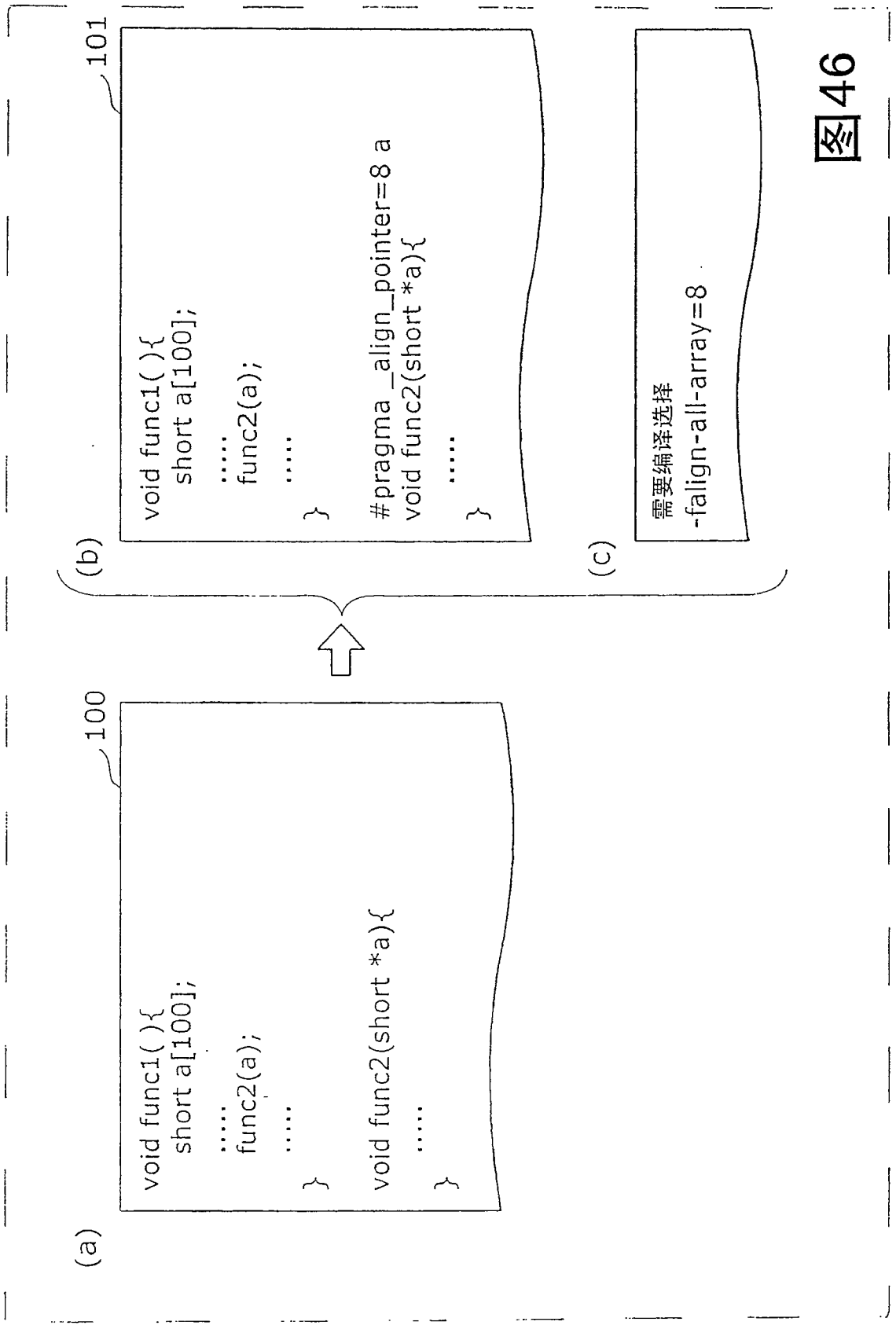
(a)
100
void func1( ){
short a[100];
.....
func2(a);
.....
}
void func2(short *a){
.....
}
(b)
101
void func1( ){
short a[100];
.....
func2(a);
.....
}
#pragma _align_pointer=8 a
void func2(short *a){
.....
}
(c)
需要编译选择
-falign-all-array=8

图46

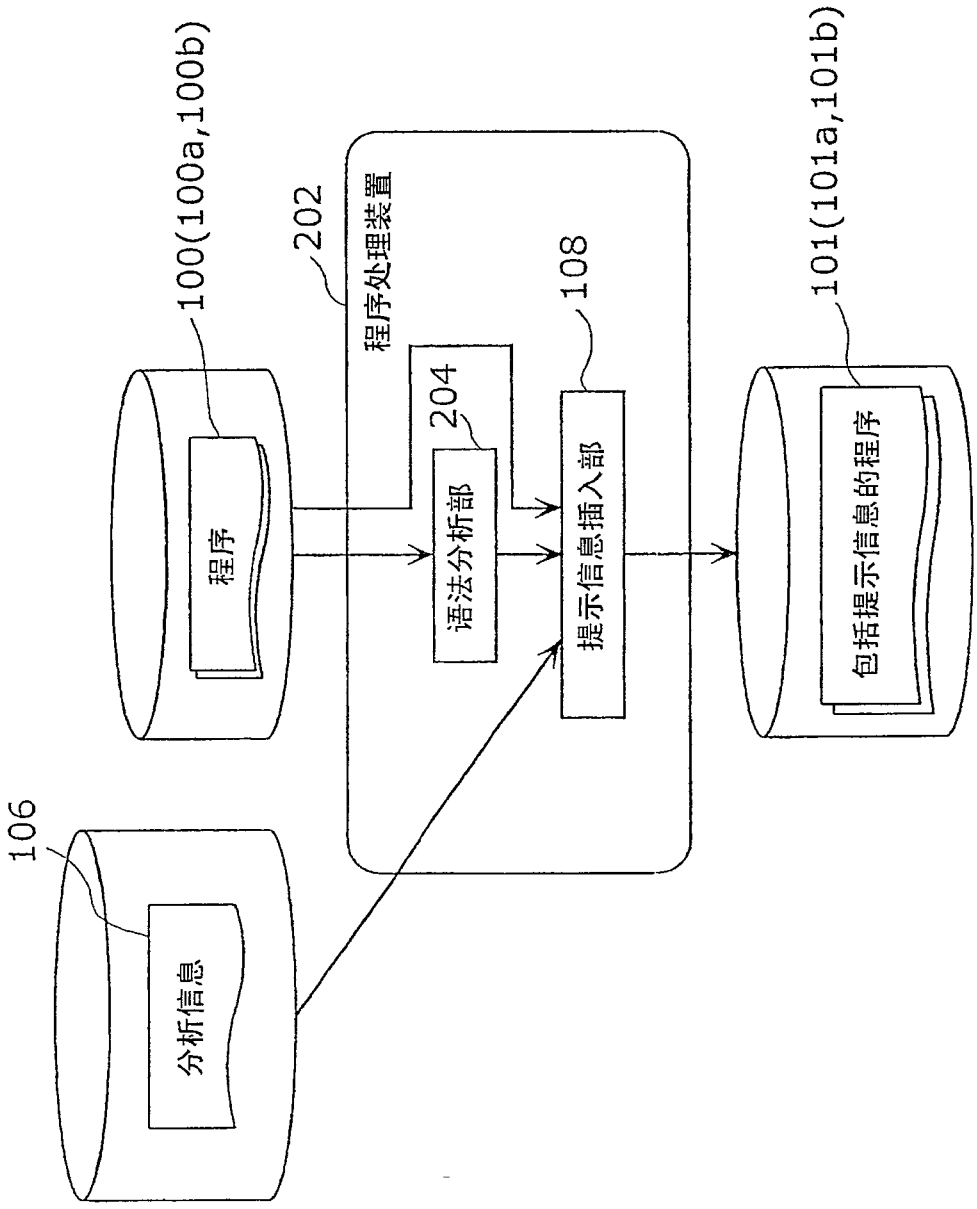
100(100a,100b)
程序
202
程序处理装置
204
语法分析部
108
提示信息插入部
101(101a,101b)
包括提示信息的程序
106
分析信息

图47

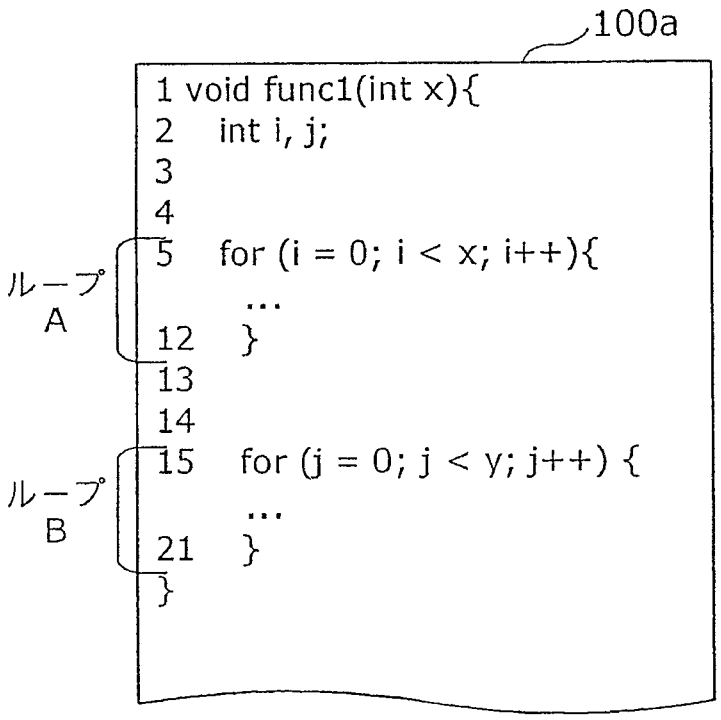
100a
1 void func1(int x){
2 int i, j;
3
4
5 for (i = 0; i < x; i++){
...
12 }
13
14
15 for (j = 0; j < y; j++) {
...
21 }
}
ループ A
ループ B

图48

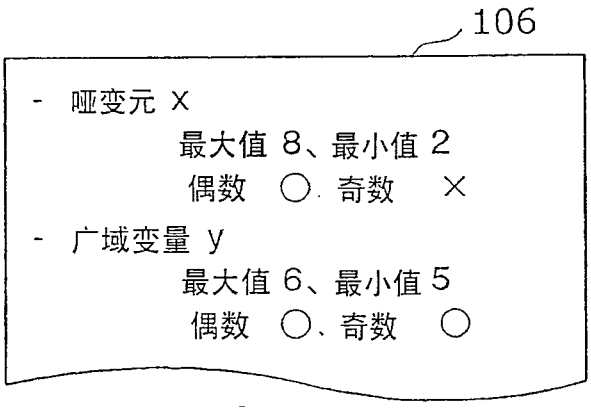
106
- 哑变元 x
最大值 8、最小值 2
偶数 ○ 奇数 ×
- 广域变量 y
最大值 6、最小值 5
偶数 ○、奇数 ○

图49

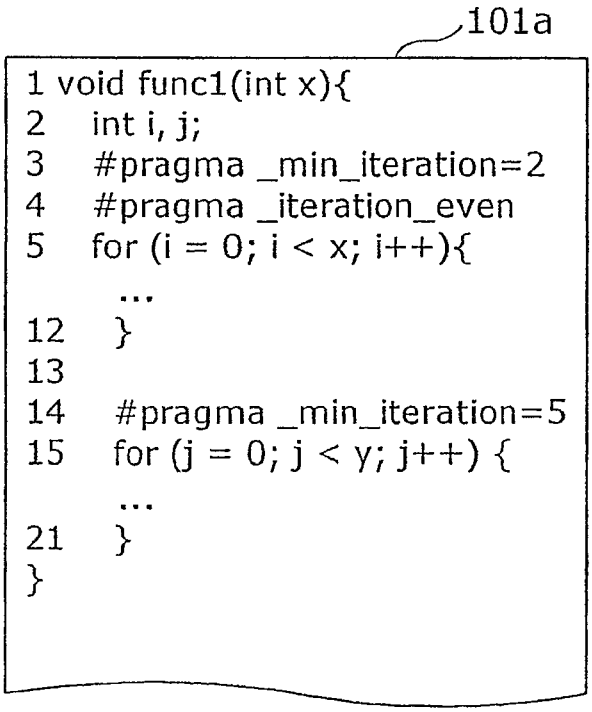
101a
1 void func1(int x){
2 int i, j;
3 #pragma _min_iteration=2
4 #pragma _iteration_even
5 for (i = 0; i < x; i++){
...
12 }
13
14 #pragma _min_iteration=5
15 for (j = 0; j < y; j++) {
...
21 }
}

图50