

(12) 发明专利

(10) 授权公告号 CN 101276278 B

(45) 授权公告日 2013.02.06

(21) 申请号 200810087281.3

(56) 对比文件

(22) 申请日 2008.03.26

US 5832205 A, 1998.11.03, 全文.

(30) 优先权数据

CN 1609805 A, 2005.04.27, 全文.

082194/2007 2007.03.27 JP

CN 1521623 A, 2004.08.18, 全文.

009763/2008 2008.01.18 JP

CN 1482540 A, 2004.03.17, 全文.

(73) 专利权人 松下电器产业株式会社

审查员 张涛

地址 日本大阪府

(72) 发明人 瓶子岳人 道本昌平 饭村幸男

山本康博

(74) 专利代理机构 永新专利商标代理有限公司

72002

代理人 胡建新

(51) Int. Cl.

G06F 9/44 (2006.01)

G06F 9/45 (2006.01)

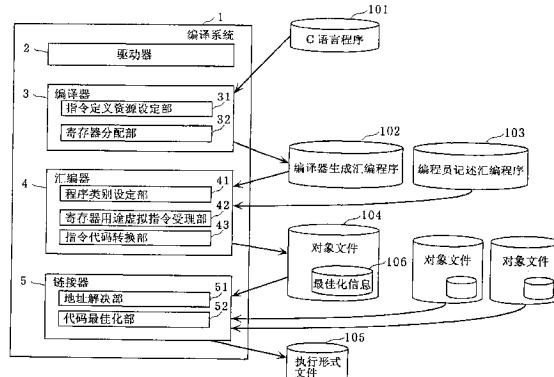
权利要求书 5 页 说明书 12 页 附图 13 页

(54) 发明名称

代码转换装置

(57) 摘要

提供一种编译系统，能够实现指令分解用的预备的寄存器的有效利用，并且，能够确保将预备的寄存器共享于多种用途时的检错性。指令定义资源设定部，作为由寄存器分配部处理的预处理，按每个指令在中间代码设定依据该指令来定义以及参照的寄存器等资源。指令定义资源设定部，检测各个指令被分解为多个指令的可能性，而对于有可能被分解的指令，视为存在用于分解的寄存器的定义以及参照，在中间代码设定该寄存器。寄存器分配部，在不跨越用于所述分解的寄存器的生存区间范围内，将该寄存器在通用用途上使用。



1. 一种代码转换装置,将以高级语言或汇编语言记述的程序转换为以目标处理器工作的机械语言代码,包括:

编译器,将以高级语言记述的程序转换为汇编程序;

汇编器,将汇编程序转换为包含机械语言代码的对象文件;以及

链接器,通过连接一个或多个对象文件,从而将所述一个或多个对象文件转换为执行形式文件,

所述编译器,包含第一用途指令判断单元以及第二用途指令生成单元,

所述第一用途指令判断单元,对于所述程序中的各个指令,判断是否将所述目标处理器的资源用于第一用途;以及

所述第二用途指令生成单元,对于由所述第一用途指令判断单元判断为不将所述资源用于第一用途的指令,生成将所述资源用于与所述第一用途不同的第二用途的机械语言代码,

所述第一用途指令判断单元,将在所述编译器后进行处理的所述汇编器以及所述链接器的处理中、有可能将所述资源用于所述第一用途的指令,判断为将所述资源用于所述第一用途的指令,

所述资源是所述目标处理器所具备的寄存器,

所述链接器包括地址解决单元,连接所输入的多个对象文件,从而确定内存上的配置地址,

所述地址解决单元,在所解决的数值操作数的值溢出了指令的操作数宽度的情况下检测错误。

2. 如权利要求1所述的代码转换装置,

所述第一用途或所述第二用途的用途是,对在将所述程序中的指令转换为机械语言代码时、数值操作数溢出指令的操作数宽度的情况下的指令进行补充。

3. 如权利要求2所述的代码转换装置,

所述补充是指,使用所述资源将数值操作数溢出指令的操作数宽度的所述指令,分割为多个指令。

4. 如权利要求1所述的代码转换装置,

所述第一用途或所述第二用途的用途是,将一个或多个指令转换为与该指令不同的、且与该指令等价的一个或多个指令。

5. 如权利要求1所述的代码转换装置,

所述第一用途或所述第二用途的用途是,控制所述目标处理器的特定的功能。

6. 如权利要求1所述的代码转换装置,

所述第一用途指令判断单元,依据指令的种类来判断是否有可能将所述资源用于所述第一用途。

7. 如权利要求1所述的代码转换装置,

所述第一用途指令判断单元,依据操作数可能的取值范围来判断是否有可能将所述资源用于所述第一用途。

8. 如权利要求1所述的代码转换装置,

所述编译器,对于所述目标处理器所具备的多个寄存器,将所述程序中的各个指令中

的、用于所述第二用途的寄存器的使用状况的信息传达给所述汇编器以及所述链接器，

所述汇编器以及所述链接器，根据由所述编译器传达的信息，选择按照所输入的各个指令的、用于所述第一用途的寄存器。

9. 如权利要求 1 所述的代码转换装置，

所述汇编器，将在所输入的汇编程序中所述资源所使用的范围判断为将所述资源用于第一用途的区间，将剩余的范围判断为将所述资源用于第二用途的区间。

10. 如权利要求 9 所述的代码转换装置，

所述第二用途是，对在将汇编程序转换为机械语言代码时、数值操作数溢出指令的操作数宽度的情况下的指令进行补充，上述补充基于所述汇编器或所述链接器来进行。

11. 一种代码转换装置，将以高级语言或汇编语言记述的程序转换为以目标处理器工作的机械语言代码，包括：

汇编器，将汇编程序转换为包含机械语言代码的对象文件；以及

链接器，通过连接一个或多个对象文件，从而将所述一个或多个对象文件转换为执行形式文件，

所述汇编器包括指示受理单元和代码生成单元，

所述指示受理单元，受理与所述目标处理器的资源的用途有关的程序员的指示；以及

所述代码生成单元，生成在所述指示所指定的用途上使用所述资源的机械语言代码，

所述资源是所述目标处理器所具备的寄存器，

所述代码生成单元，具有：

用途判断单元，根据所述指示受理单元所受理的指示，判断在所述汇编程序的各个指令记述中的所述资源的用途；以及

转换单元，将所述汇编程序的指令记述转换为在被判断的所述用途上使用所述资源的机械语言代码，

所述用途判断单元，根据所述指示受理单元所受理的指示，判断将相当于所述资源的目标处理器的寄存器是用于通用寄存器用途的区间、还是用于数值操作数溢出时的补充用途的区间，

所述转换单元，对于由所述用途判断单元判断为将所述寄存器用于所述通用寄存器用途的区间内包含的指令记述，将该指令记述转换为基于该指令记述的机械语言代码；对于由所述用途判断单元判断为将所述寄存器用于所述补充用途的区间内包含的指令记述，将该指令记述转换为，对该指令记述的数值操作数的溢出进行了补充的状态下的机械语言代码，

所述链接器包括地址解决单元，连接所输入的多个对象文件，从而确定内存上的配置地址，

所述地址解决单元，在所解决的数值操作数的值溢出了指令的操作数宽度的情况下检测错误。

12. 如权利要求 11 所述的代码转换装置，

所述程序员的指示是，在作为所述资源的多种用途的第一用途以及第二用途中、将所述资源在所述程序中用于第一用途的指令的指示。

13. 如权利要求 12 所述的代码转换装置，

所述第一用途或所述第二用途是,对在将所述程序中的指令转换为机械语言代码时、数值操作数溢出指令的操作数宽度的情况下的指令进行补充。

14. 如权利要求 13 所述的代码转换装置,

所述补充是指,使用所述资源将数值操作数溢出指令的操作数宽度的所述指令,分割为多个指令。

15. 如权利要求 12 所述的代码转换装置,

所述第一用途或所述第二用途的用途是,将一个或多个指令转换为与该指令不同的、且与该指令等价的一个或多个指令。

16. 如权利要求 12 所述的代码转换装置,

所述第一用途或所述第二用途的用途是,控制所述目标处理器的特定的功能。

17. 如权利要求 12 所述的代码转换装置,

所述程序员的指示是,示出将所述资源用于所述第一用途或所述第二用途的区间的起点和终点的指示。

18. 如权利要求 12 所述的代码转换装置,

所述程序员的指示是将信息汇总并记述在所述程序的一部分的指示,所述信息是将所述资源用于所述第一用途或所述第二用途的区间的信息。

19. 如权利要求 12 所述的代码转换装置,

所述程序员的指示是,以所述程序的文件为单位示出将所述资源用于所述第一用途以及所述第二用途中的哪种用途的指示。

20. 如权利要求 12 所述的代码转换装置,

所述程序员的指示是,以所述程序中的指令为单位示出将所述资源用于所述第一用途以及所述第二用途中的哪种用途的指示。

21. 如权利要求 12 所述的代码转换装置,

所述程序员的指示是,通过对所述程序中的所述资源的表示方法,在所述第一用途和所述第二用途采用不同的表示方法而被进行的。

22. 如权利要求 11 所述的代码转换装置,

所述程序员的指示是,对所述代码转换装置的、依据虚拟指令的指示。

23. 如权利要求 11 所述的代码转换装置,

所述程序员的指示是,依据所述代码转换装置的工作选项的指示。

24. 如权利要求 11 所述的代码转换装置,

所述程序员的指示是,对所述代码转换装置的、依据编译指示指令的指示。

25. 如权利要求 11 所述的代码转换装置,还包括,

错误检测单元,根据与所述资源的用途有关的所述程序员的指示,检测程序员的记述的错误。

26. 如权利要求 25 所述的代码转换装置,还包括,

程序类别判断单元,判断所述程序是否是由程序员记述的程序,

所述错误检测单元,只在所述程序类别判断单元判断为所述程序是由程序员记述的程序的情况下进行检错。

27. 如权利要求 26 所述的代码转换装置,

所述程序类别判断单元,依据对所述代码转换装置的虚拟指令进行程序的类别的判断。

28. 如权利要求 26 所述的代码转换装置,

所述程序类别判断单元,依据所述代码转换装置的工作选项进行程序的类别的判断。

29. 如权利要求 26 所述的代码转换装置,

所述程序类别判断单元,依据对所述代码转换装置的编译指示指令进行类别的判断。

30. 如权利要求 11 所述的代码转换装置,还包括,

警告消息显示单元,根据与所述资源的用途有关的编程员的指示,显示对于将所述资源在所述用途上使用了的程序中的部分的警告消息。

31. 如权利要求 30 所述的代码转换装置,

所述警告消息显示单元,依据与所述资源的用途有关的所述编程员的指示,选择显示或不显示警告消息。

32. 如权利要求 31 所述的代码转换装置,

所述编程员的指示是,对所述代码转换装置的、依据虚拟指令的指示。

33. 如权利要求 31 所述的代码转换装置,

所述编程员的指示是,依据所述代码转换装置的工作选项的指示。

34. 如权利要求 31 所述的代码转换装置,

所述编程员的指示是,对所述代码转换装置的、依据编译指示指令的指示。

35. 如权利要求 31 所述的代码转换装置,

与寄存器的用途有关的所述编程员的指示是,在补充时是否应该显示警告消息的指示,该补充是在数值操作数溢出时进行的补充。

36. 如权利要求 30 所述的代码转换装置,还包括,

程序类别判断单元,判断所述程序是否是由编程员记述的程序,

所述警告消息显示单元,只在所述程序是由编程员记述的程序的情况下显示警告消息。

37. 如权利要求 36 所述的代码转换装置,

所述程序类别判断单元,根据对所述代码转换装置的虚拟指令,判断所述程序是否是由编程员记述的程序。

38. 如权利要求 36 所述的代码转换装置,

所述程序类别判断单元,根据所述代码转换装置的工作选项,判断所述程序是否是由编程员记述的程序。

39. 如权利要求 36 所述的代码转换装置,

所述程序类别判断单元,根据对所述代码转换装置的编译指示指令,判断所述程序是否是由编程员记述的程序。

40. 如权利要求 36 所述的代码转换装置,

所述代码转换装置,包括:

编译器,将以高级语言记述的程序转换为汇编程序;

所述汇编器包括所述程序类别判断单元或所述指示受理单元,

所述链接器包括所述警告消息显示单元。

41. 如权利要求 40 所述的代码转换装置，
所述编译器，将抑制所述警告消息的显示的指示输出到所述汇编程序，
所述警告消息显示单元，与抑制所述警告消息的显示的指示对应，来抑制警告消息的
显示。

代码转换装置

技术领域

[0001] 本发明涉及一种代码转换装置,将以 C 语言或汇编语言记述的源程序转换为机械语言程序;尤其涉及一种代码转换装置,有效利用处理器中具备的寄存器,并且确保源程序的检错性。

背景技术

[0002] 在使用处理器所执行的立即值或地址等常数的指令中,在常数的位宽溢出指令格式的规定的宽度的情况下,不能以单一的指令实现该功能。据此,需要将一个指令分解为多个指令,例如分为设定常数的指令和进行运算的指令。

[0003] 对此,在以往的技术中有一种方法:例如像专利文献(日本国特开平 8-6797 号公报)所记载的发明那样,在源程序的编译阶段将单一的指令分解为多个指令,并且实现常数的参照的效率化。

[0004] 而且,有另一种方法:由于到链接时为止不能知道地址等的值是否溢出,因此在编译时以单一的指令预生成机械语言代码,在链接阶段只对需要的指令适当地活用预备的寄存器等,从而分解为多个指令。

[0005] 地址或置换值等较多的值,到链接时为止不确定,但是,若在编译时被分解为多个指令,则导致不必要的指令的分解。因此,在执行性能或代码大小方面的效率不好。

[0006] 而且,在链接阶段只对需要的指令活用预备的寄存器从而分解指令的方法中,需要保存预备的寄存器。因此,存在的问题是:依据寄存器压迫的性能劣化,该寄存器压迫是因不能将预备的寄存器作为通用寄存器使用而引起的;硬件资源的增加;以及硬件使用效率的降低。

[0007] 对此,也可以考虑以编程员的责任将预备的寄存器共享于通用寄存器用途,但不存在对使用错误能够进行检错的工具。因此,导致调试性的降低、开发效率的降低。

发明内容

[0008] 为了解决所述课题,本发明的目的在于,提供一种编译系统,能够实现所述指令分解用的预备的寄存器的有效利用,并且,能够确保将预备的寄存器共享于多种用途时的检错性。

[0009] 为了实现所述目的,本发明涉及的代码转换装置是一种代码转换装置,将以高级语言或汇编语言记述的程序转换为以目标处理器工作的机械语言代码,包括:第一用途指令判断单元,对于所述程序中的各个指令,判断是否将所述目标处理器的资源用于第一用途;以及第二用途指令生成单元,对于由所述第一用途指令判断单元判断为不将所述资源用于第一用途的指令,生成将所述资源用于与所述第一用途不同的第二用途的机械语言代码。

[0010] 根据该结构,能够将处理器所具备的寄存器资源自动地共享于多种用途。据此,能够提高执行性能、降低代码大小。而且,也能够提高指令分解用的预备的寄存器等硬件资源

的利用效率。

[0011] 优选的是，所述第一用途或所述第二用途的用途是，对在将所述程序中的指令转换为机械语言代码时数值操作数溢出指令的操作数宽度的指令进行的补充。

[0012] 而且，所述补充是指，使用所述资源将数值操作数溢出指令的操作数宽度的所述指令，分割为多个指令。

[0013] 根据该结构，也能够将指令分解用的预备的寄存器用于指令的补充用途。另外，也能够将具备的寄存器用于通用用途。据此，能够实现预备的寄存器的有效利用。

[0014] 优选的是，所述第一用途或所述第二用途的用途是，控制所述目标处理器的特定的功能。

[0015] 根据该结构，也能够将目标处理器的资源用于处理器的控制用途。另外，也能够将该资源用于通用用途。据此，能够实现处理器的有效利用。

[0016] 本发明涉及的代码转换装置是一种代码转换装置，将以高级语言或汇编语言记述的程序转换为以目标处理器工作的机械语言代码，包括：指示受理单元，受理与所述目标处理器的资源的用途有关的程序员的指示；以及代码生成单元，生成在所述指示所指定的用途上使用所述资源的机械语言代码。

[0017] 根据该结构，能够依据程序员的意图，将处理器的资源在多种用途上灵活使用。据此，能够提高执行性能、降低代码大小。而且，也能够提高处理器的资源的利用效率。

[0018] 优选的是，所述代码转换装置包括：汇编器，将汇编程序转换为包含机械语言代码的对象文件；以及链接器，通过连接一个或多个对象文件，从而将所述一个或多个对象文件转换为执行形式文件，所述汇编器包括：所述指示受理单元和所述代码生成单元，所述代码生成单元，具有：用途判断单元，根据所述指示受理单元所受理的指示，判断在所述汇编程序的各个指令记述中的所述资源的用途；以及转换单元，将所述汇编程序的指令记述转换为在被判断的所述用途上使用所述资源的机械语言代码。

[0019] 而且，所述用途判断单元，根据所述指示受理单元所受理的指示，判断将相当于所述资源的目标处理器的寄存器是用于通用寄存器用途的区间、还是用于数值操作数溢出时的补充用途的区间，所述转换单元：对于由所述用途判断单元判断为将所述寄存器用于所述通用寄存器用途的区间内包含的指令记述，将该指令记述转换为基于该指令记述的机械语言代码；对于由所述用途判断单元判断为将所述寄存器用于所述补充用途的区间内包含的指令记述，将该指令记述转换为对该指令记述补充数值操作数的溢出的状态的机械语言代码，所述链接器包括地址解决单元：连接所输入的多个对象文件，从而确定内存上的配置地址，所述地址解决单元：在所解决的数值操作数的值溢出指令的操作数宽度的情况下检测错误。

[0020] 据此，能够将数值操作数溢出的补充用途的寄存器在通用寄存器用途上活用。同时，能够确保依据编程错误的检错性，从而提高软件的开发效率。

[0021] 并且，本发明，除了可以作为包括如上所述的特征性单元的代码转换装置来实现以外，也可以作为将代码转换装置中包括的特征性单元作为步骤的代码转换方法来实现，还可以作为用于使计算机执行代码转换装置中包括的特征性步骤的程序来实现。并且，当然也可以通过CD-ROM(Compact Disc-Read Only Memory)等存储介质或互联网等通信网络来分发这些程序。

[0022] 通过将用于以往在特定的用途的寄存器活用于多种用途,从而能够提高执行性能、降低代码大小。而且,能够确保将寄存器活用于多种用途时的检错性。因此,能够提高软件的开发效率。

附图说明

- [0023] 图 1 是用于实现本发明的实施例 1 涉及的编译系统的计算机的外观图。
- [0024] 图 2 是本发明的实施例 1 涉及的编译系统的结构框图。
- [0025] 图 3 是示出在单一指令的操作数的数值溢出的情况下、到多个指令的置换模式的一个例子的图。
- [0026] 图 4 是指令定义资源设定部的处理内容的流程图。
- [0027] 图 5 是示出最佳化信息的一个例子的图。
- [0028] 图 6 是示出编译器生成汇编程序的一个例子的图。
- [0029] 图 7 是示出包含虚拟指令“ENABLE_R29”的程序员记述汇编程序的一个例子的图。
- [0030] 图 8 是汇编器的处理的流程图。
- [0031] 图 9 是链接器的处理的流程图。
- [0032] 图 10 是示出编译器生成汇编程序的一个例子的图。
- [0033] 图 11 是示出包含虚拟指令“noexpandinst”等的程序员记述汇编程序的一个例子的图。
- [0034] 图 12 是汇编器的处理的流程图。
- [0035] 图 13 是示出最佳化信息的一个例子的图。
- [0036] 图 14 是链接器的处理的流程图。

具体实施方式

- [0037] (实施例 1)
 - [0038] 下面,参照附图说明本发明的实施例 1 涉及的编译系统。并且,通过图 1 所示的计算机上使程序执行,从而实现所述编译系统。
 - [0039] 图 2 是本发明的实施例 1 涉及的编译系统 1 的结构框图。
 - [0040] 编译系统 1 包括驱动器 2、编译器 3、汇编器 4 以及链接器 5。
 - [0041] 驱动器 2 具有下列功能:根据用户所指定的选项,按需要以适当的选项使作为编译系统 1 的其它构成要素的编译器 3、汇编器 4 以及链接器 5 启动。例如,驱动器 2,可以只使汇编器 4 启动,也可以只使汇编器 4 和链接器 5 启动。
 - [0042] 编译器 3,将程序员所记述的 C 语言程序 101 作为输入接受,而将 C 语言程序 101 转换为内部中间表示来实施最佳化或资源的分配后,生成对目标处理器的编译器生成汇编程序 102。
 - [0043] 汇编器 4,将编译器 3 所生成的编译器生成汇编程序 102、或程序员所记述的程序员记述汇编程序 103 作为输入接受,而将编译器生成汇编程序 102 或程序员记述汇编程序 103 转换为机械语言代码,从而生成由机械语言代码而成的对象文件 104。对象文件 104 中包含在链接器 5 用于最佳化的最佳化信息 106。
 - [0044] 链接器 5,将一个或多个对象文件作为输入接受,而实施对象文件 104 中包含的未

解析的外部符号的地址或置换值的解析、以及代码的最佳化，从而生成执行形式文件 105。

[0045] 在此，说明本发明的实施例 1 涉及的编译系统 1 中成为目标的处理器的指令方法。

[0046] 本处理器的指令是 32 位固定长度，在处理超过可以以一个指令表示的数值的宽度的数值的情况下，通过将多个指令组合起来，从而可以实现功能。

[0047] 在本发明的实施例 1 涉及的编译系统 1 中，如后述，在链接时知道成为汇编程序中的指令的操作数的数值超过可以以指令表示的数值的宽度的情况下，将该指令置换为多个指令的组合，从而可以生成代码。在该置换时使用作为特定的通用寄存器的“R29”。

[0048] 图 3 示出在单一指令的操作数的数值溢出的情况下的、到多个指令的置换模式的一个例子。

[0049] 在图中，“add”、“ld”、“br”、“setlo”、“sethi”、“mov”、“jmp”表示指令助记符，“R0”、“R1”、“R29”表示通用寄存器，“TAR”表示目标地址设定用寄存器。而且，“imm16”示出 16 位的常数，“imm32”示出 32 位的常数，“disp10”示出表示置换（位移）的 10 位的常数，“disp20”示出表示置换的 20 位的常数，“disp32”示出表示置换的 32 位的常数，“addr32”示出表示地址的 32 位的常数。再者，“LO()”是宏汇编，将括号内的常数的低 16 位作为返回值返回。同样，“HI()”将括号内的常数的高 16 位作为返回值返回。

[0050] 最初的单一的 add 指令是加法指令，使 imm16 和寄存器 R1 存储值相加，从而将加法结果存储到寄存器 R1。在操作数的数值超过 16 位的情况下，可以依据在右边所述的多个指令来实现功能。具体而言，首先，以 setlo 指令在寄存器 R29 的低 16 位存储数值的低 16 位，其次，以 sethi 指令在寄存器 R29 的高 16 位存储数值的高 16 位，从而在寄存器 R29 设定最大 32 位的数值。最后，以 add 指令使 R1 的存储值和 R29 的存储值相加，从而将加法结果存储到寄存器 R1。

[0051] 第二个单一的 ld 指令是内存存取指令，从使寄存器 R1 的存储值和 disp10 相加而得的内存地址中装入数据，从而将装入后的数据存储到寄存器 R0。在置换值超过 10 位的情况下，可以依据在右边所述的多个指令来实现功能。具体而言，首先，与第一个 add 指令相同，以 setlo 指令和 sethi 指令在寄存器 R29 存储最大 32 位的置换值。其次，以 add 指令使寄存器 R29 的存储值和 R1 的存储值相加，从而将加法结果存储到寄存器 R29。最后，以 ld 指令从寄存器 R29 的存储值的内存地址中装入数据，从而将装入后的数据存储到寄存器 R0。

[0052] 第三个单一的 br 指令是转移指令，转移到使当前的程序计数器的值和 disp20 值相加而得的指令地址。在置换值超过 20 位的情况下，可以依据在右边所述的多个指令来实现功能。具体而言，首先，与第一个 add 指令相同，以 setlo 指令和 sethi 指令在寄存器 R29 存储 32 位的转移目标地址 addr32。其次，以 mov 指令将寄存器 R29 的存储值存储到目标地址设定寄存器 TAR。最后，以 jmp 指令转移到 TAR 所存储的指令地址。

[0053] 下面，说明本发明的实施例 1 涉及的编译系统 1 的各构成要素的详细结构和工作。

[0054] 编译器 3 包括指令定义资源设定部 31 以及寄存器分配部 32。

[0055] 指令定义资源设定部 31，作为由寄存器分配部 32 处理的预处理，按每个指令在中间代码设定依据该指令来定义以及参照的寄存器等资源。作为特征性部分，检测各个指令被分解为多个指令的可能性，而对于有可能被分解的指令，视为存在用于分解的寄存器 R29 的定义以及参照，在中间代码设定寄存器 R29。具体而言，指令定义资源设定部 31 保持相当

于图 3 的指令转换模式的表,检测是否符合转换模式的单一指令的指令。再者,在该指令的 imm16 或 disp20 等的操作数是像标签或标签间运算的结果值等的值那样、不能静态确定的数值的情况下,视为该指令定义以及参照寄存器 R29,在中间代码设定寄存器 R29。

[0056] 图 4 是指令定义资源设定部 31 的处理内容的流程图。

[0057] 设想,在指令定义资源设定部 31 执行处理之前,C 语言程序 101 被编译器 3 已转换为中间代码。

[0058] 指令定义资源设定部 31,按每个指令反复进行以下处理,该指令是从所输入的源程序(C 语言程序 101)转换为中间代码的指令。首先,指令定义资源设定部 31,在该中间代码设定依据该指令来定义的寄存器以及参照的寄存器(步骤 S1)。其次,指令定义资源设定部 31,判断该指令是否符合图 3 的转换模式的单一指令(步骤 S2)。指令定义资源设定部 31,只在符合单一指令的情况下(步骤 S2 的“是”),判断是否静态确定该指令的操作数的数值(步骤 S3)。在判断为不能静态确定的情况下(步骤 S3 的“否”),指令定义资源设定部 31,视为该指令定义以及参照寄存器 R29,在该中间代码设定寄存器 R29(步骤 S4)。指令定义资源设定部 31,到不存在未处理的指令为止进行所述处理。

[0059] 寄存器分配部 32,与普通的编译器的寄存器分配部相同,根据指令定义资源设定部 31 所设定的信息求出各个变量的生存区间,向各个生存区间分配包含寄存器 R29 的实际寄存器群,以便不使同一寄存器被分配的变量的生存区间不重叠。

[0060] 通过如上构成编译器 3,从而在不跨越指令中的寄存器 R29 的生存区间的范围内,也可以将寄存器 R29 作为通用寄存器活用,所述指令有可能产生向多个指令的分解。因此,可以将寄存器 R29 共享于指令分解用途和通用寄存器用途的两种用途。

[0061] 汇编器 4 的程序类别设定部 41、寄存器用途虚拟指令受理部 42 以及指令代码转换部 43,对所输入的编译器生成汇编程序 102 以及编程员记述汇编程序 103 的各行,依次进行工作。

[0062] 程序类别设定部 41,通过解释该行的虚拟指令的记述,从而判断所输入的汇编程序是编译器 3 所生成的编译器生成汇编程序 102、还是编程员所记述的编程员记述汇编程序 103。程序类别设定部 41,在所输入的汇编程序是编译器生成汇编程序 102 的情况下,将内部的程序类别模式设定为“1”,在是编程员记述汇编程序 103 的情况下,设定为“0”。具体而言,程序类别设定部 41,在汇编程序文件内的指令代码之前出现只编译器 3 作为首部(header)输出的“FILE”虚拟指令的情况下,判断为所输入的汇编程序是编译器生成汇编程序 102。

[0063] 寄存器用途虚拟指令受理部 42,在该行记述有示出寄存器的用途的虚拟指令的情况下,对应该虚拟指令而切换内部的寄存器使用模式。具体而言,寄存器用途虚拟指令受理部 42,在出现虚拟指令“ENABLE_R29 ON”的情况下,将寄存器使用模式变更为示出可以使用 R29 的“1”,在出现虚拟指令“ENABLE_R29 OFF”的情况下,将寄存器使用模式变更为示出不可使用 R29 的“0”,所述虚拟指令“ENABLE_R29 ON”示出将寄存器 R29 作为通用寄存器使用的区间的开始,所述虚拟指令“ENABLE_R29 OFF”示出将寄存器 R29 作为通用寄存器使用的区间的结束。如上所述,寄存器用途虚拟指令受理部 42,依据虚拟指令的记述的解释来识别将寄存器 R29 作为通用寄存器使用的区间。

[0064] 指令代码转换部 43,进行对应于该行的指令记述的、到机械语言代码的转换。在转

换时,指令代码转换部 43,参照所述程序类别模式和寄存器使用模式。指令代码转换部 43,在程序类别模式示出编程员记述的“0”、且寄存器使用模式为示出不可使用 R29 的“0”的状态下,并且在该行的指令记述出现寄存器 R29 的情况下,输出表示不可使用寄存器 R29 的错误消息,从而结束汇编器 4 的处理。

[0065] 而且,指令代码转换部 43,保持相当于图 3 的指令转换模式的表,检测该行的指令记述是否符合转换模式的左边所述的单一指令的指令。再者,在该指令的数值操作数是像标签或标签间运算的结果值等的值那样、不能静态确定的数值的情况下,指令代码转换部 43,生成与转换模式的右边所述的多个指令相对应的机械语言代码。但是,在寄存器使用模式为示出可以使用 R29 的“1”的情况下,不进行该处理。据此,除了寄存器使用模式为的“1”的区间以外,符合转换模式的所有的指令以被分解为多个指令的状态下被生成,但是,设想,将在后述的链接器 5 的最佳化中不需要分解的指令恢复为原来的指令。为了该最佳化,指令代码转换部 43,向对象文件 104 附加最佳化信息 106,该最佳化信息 106 用于识别分解为多个指令而生成的部分。

[0066] 图 5 示出最佳化信息 106 的一个例子。

[0067] 在最佳化信息 106 中,将汇编器 4 所转换的机械语言代码列的开头到该部分为止的指令地址的偏置值、和图 3 中的该转换模式的模式 ID 的对排列。各个对与各个转换部分相对应。

[0068] 图 6 示出编译器生成汇编程序 102 的一个例子。

[0069] 在图 6 中,(a) 的行是用于编译器 3 传达 C 语言程序 101 的文件名而输出的“FILE”虚拟指令。汇编器 4,依据该虚拟指令,可以识别为所输入的汇编程序是编译器所生成的编译器生成汇编程序 102。而且,在 (b) 的行以及 (d) 的行使用寄存器 R29,这是因为,如上所述,针对在编译器 3 的处理中知道在链接时不需要分解为多个指令的部分,将寄存器 R29 用于通用寄存器用途。具体而言,寄存器 R29 的生成区间,在 (b) 的行开始、在 (d) 的行结束。在 (b) 的行中,将使寄存器 R0 和寄存器 R1 的内容相加的结果存储到寄存器 R29。在 (c) 的行中,从向寄存器 R2 的内容加上常数 4 的值的内存地址中装入的值存储到寄存器 R0。在 (d) 的行中,寄存器 R0 的内容减去寄存器 R29 的内容。在此期间,不存在有可能在链接时需要分解为多个指令的指令。因此,编译器 3 解析所述内容,而将寄存器 R29 作为通用寄存器使用。特别是对于 (c) 的行,指令的种类本身符合图 3 的转换模式的单一指令,但是,数值操作数确定为值 4,因此可以静态知道在链接时不需要分解为多个指令。因此,编译器 3 判断为不存在因向多个指令的分解而使用寄存器 R29 的可能性。

[0070] 图 7 示出包含“ENABLE_R29”的编程员记述汇编程序 103 的一个例子。

[0071] 在图 7 中,(e) 的行是虚拟指令,以编程员指示将寄存器 R29 用于通用寄存器用途的区间的开始。而且,(h) 的行是虚拟指令,以编程员指示将寄存器 R29 用于通用寄存器用途的区间的结束。其结果是,从 (e) 的行到 (h) 的行之间的 (X) 的区间成为将寄存器 R29 用于通用寄存器用途的区间。在该区间,不实施在依据编译系统 1 的数值操作数溢出的情况下扩张处理,即,不实施向多个指令的转换处理。并且,其它区间成为实施依据编译系统 1 的、使用了寄存器 R29 的操作数扩张处理的区间。

[0072] 图 8 是汇编器 4 的处理的流程图。

[0073] 首先,汇编器 4,将作为内部的变量的程序类别模式和寄存器使用模式都初始化为

“0”(步骤 S11)。并且,汇编器 4,按所输入的汇编程序的每个行反复进行以下处理。

[0074] 程序类别设定部 41,判断是否出现了“FILE”虚拟指令(步骤 S12),在出现了的情况下(步骤 S12 的“是”),在程序类别模式设定“1”,该“1”示出编辑器所生成的程序(步骤 S13)。

[0075] 其次,寄存器用途虚拟指令受理部 42,判断是否出现了虚拟指令“ENABLE_R29 ON”(步骤 S14),在出现了的情况下(步骤 S14 的“是”),在寄存器使用模式设定“1”,该“1”示出将寄存器 R29 作为通用寄存器使用(步骤 S15)。而且,是否出现了虚拟指令“ENABLE_R29 OFF”(步骤 S16),在出现了的情况下(步骤 S16 的“是”),在寄存器使用模式设定“0”,该“0”示出将寄存器 R29 用于在数值操作数溢出的情况下的扩张(步骤 S17)。

[0076] 指令代码转换部 43 判断:程序类别模式是否示出编程员记述汇编程序 103 的“0”;并且寄存器使用模式是否示出将寄存器 R29 用于在数值操作数溢出的情况下的扩张的“0”;并且是否出现了寄存器 R29 的记述(步骤 S18)。在判断是真的情况下(步骤 S18 的“是”),可以知道编程员的记述错误,因此,指令代码转换部 43,输出错误消息,从而汇编器 4 的处理结束(步骤 S19)。

[0077] 而且,指令代码转换部 43 判断是否全部满足下列(a)~(c)的三个条件(步骤 S20):(a) 寄存器使用模式是示出将寄存器 R29 用于在数值操作数溢出的情况下的扩张的“0”; (b) 指令与图 3 的转换模式的任何单一指令符合; (c) 不能静态确定该指令的操作数的数值。在全部满足(a)~(c)的条件的情况下(步骤 S20 的“是”),指令代码转换部 43,将该行的指令转换为转换模式的右边所述的多个指令。同时,指令代码转换部 43,作为关于转换的最佳化信息 106,将从指令列的开头到该指令为止的指令地址偏置值和图 3 的转换模式的模式 ID 的对,附加给生成代码(对象文件 104)(步骤 S21)。并且,指令代码转换部 43,将指令转换为机械语言代码,从而输出对象文件 104(步骤 S22)。

[0078] 汇编器 4,到不存在未处理的行为止执行所述处理。

[0079] 链接器 5 包括地址解决部 51 和代码最佳化部 52。到不存在依据代码最佳化部 52 中的处理的代码的变化为止,地址解决部 51 和代码最佳化部 52 反复进行工作。

[0080] 地址解决部 51,与普通的链接器相同,连接输入到链接器 5 的多个对象文件 104 而确定内存上的配置地址,从而确定各个对象文件 104 中包含的未解决的符号的地址或置换值。此时,确定的地址或置换值超过该指令中可以表示的数值宽度的情况下,输出表示值溢出了的消息,从而链接器 5 结束处理。

[0081] 代码最佳化部 52 保持相当于图 3 的转换模式的表。代码最佳化部 52,参照所述汇编器 4 的指令代码转换部 43 所附加的、示出分解为多个指令的部分和转换模式的最佳化信息 106,根据地址解决部 51 所确定的地址值或置换值,从而判断是否可以以转换模式的左边所述的单一指令来表示该部分。在可以以单一指令表现的情况下,即,在可以将所确定的操作数的数值的宽度放到单一指令的情况下,代码最佳化部 52,进行将该部分置换为转换模式左边所述的单一指令的处理。

[0082] 图 9 是链接器 5 的处理的流程图。

[0083] 链接器 5,首先将示出代码的变化的变化标记初始化为“0”(步骤 S31)。

[0084] 地址解决部 51,与普通的链接器相同,连接所输入的多个对象文件而确定内存上的配置地址,从而确定各个对象文件中包含的未解决的符号的地址或置换值(步骤 S32)。

地址解决部 51, 判断所确定的地址或置换值的指令中是否产生了针对可以表示的数值宽度的溢出 (步骤 S33)。在产生了溢出的情况下 (步骤 S33 的“是”), 地址解决部 51, 输出表示溢出了的消息, 从而链接器 5 结束处理 (步骤 S34)。

[0085] 代码最佳化部 52, 针对与所输入的各个对象文件 104 对应的代码块反复进行以下处理。代码最佳化部 52, 对于各个代码块, 对该对象文件 104 的最佳化信息 106 的各个项目、即图 5 的各个行反复进行以下处理。

[0086] 首先, 代码最佳化部 52, 依据最佳化信息 106 的指令地址偏置值, 确定由汇编器 4 展开为多个指令的部分。并且, 代码最佳化部 52, 还依据转换模式的模式 ID 确定转换模式。再者, 代码最佳化部 52, 判断是否可以将地址解决部 51 所确定的操作数的值, 放到图 3 所示的转换模式的左边所述的形式的数值操作数宽度 (步骤 S35)。即, 代码最佳化部 52, 判断是否可以以单一指令来表示多个指令。

[0087] 在判断为可以放到数值操作数宽度的情况下 (步骤 S35 的“是”), 代码最佳化部 52, 将符合的多个指令代码置换为与转换模式的左边所述的单一指令对应的代码, 并且将变化标记设定为“1”, 该“1”表示发生了代码的变化 (步骤 S36)。

[0088] 代码最佳化部 52, 对最佳化信息 106 的所有的项目反复进行所述处理。并且, 代码最佳化部 52, 对与所输入的各个对象文件对应的代码块反复进行所述处理。

[0089] 所述处理结束后, 链接器 5, 判断变化标记是否是表示发生了代码的变化的“1”(步骤 S37)。在是“1”的情况下 (步骤 S37 的“是”), 链接器 5 返回到步骤 S31 而反复进行到此为止的处理, 而在不是“1”的情况下 (步骤 S37 的“否”), 链接器 5 结束处理。

[0090] 通过这些汇编器 4 的指令代码转换部 43 和链接器 5 的代码最佳化部 52 的配合, 在最后生成的执行形式文件 105 中, 生成只在需要向多个指令的展开的部分使用了寄存器 R29 的、多个指令的模式的代码。

[0091] 通过如上构成所述的汇编器 4 以及链接器 5, 编程员可以将操作数的数值溢出时的、向多个指令的展开用的寄存器 R29, 用于通用寄存器用途, 从而程序的性能会提高。

[0092] 而且, 也可以确保编程员将寄存器 R29 用于通用寄存器用途时的检错性, 从而开发效率会提高。具体而言, 通过在汇编程序记述虚拟指令 ENABLE_R29, 从而可以将指令分解用的预备的寄存器 R29 活用于通用寄存器用途, 而且, 可以由汇编器检测在不意图的部分的寄存器 R29 的记述错误。

[0093] 而且, 在编程员将寄存器 R29 活用于通用寄存器用途的区域, 抑制依据汇编器以及链接器的配合的、使用了寄存器 R29 的向多个指令的展开。据此, 可以防止的问题是: 编程员在通用寄存器用途上使用了的寄存器 R29 非意图地被破坏, 从而执行结果成为不正当。具体而言, 在作为通用寄存器活用的区域内操作数的数值溢出的情况下, 在链接时可以检测为错误。

[0094] (实施例 2)

[0095] 下面, 说明本发明的实施例 2 涉及的编译系统。实施例 2 涉及的编译系统的大部分的结构与实施例 1 涉及的编译系统 1 相同, 但是, 对汇编器 4 的虚拟指令的方法不同。下面, 以不同之处为中心进行说明。

[0096] 图 10 示出编译器生成汇编程序 102 的一个例子。

[0097] 在图 10 中, (a) 的行是本实施例特有的虚拟指令“expandinst_without_

warning”。该虚拟指令是用于“在链接时不表示警告而实施数值操作数的扩张处理”的指示。编译器 3, 在生成汇编程序时, 在汇编程序的开头输出该虚拟指令。

[0098] 图 11 示出编程员记述汇编程序 103 的一个例子。

[0099] 在图 11 中, (b) 的行是本实施例特有的虚拟指令“noexpandinst”。该虚拟指令是用于“抑制数值操作数的扩张处理”的指示。而且, (e) 的行是本实施例特有的虚拟指令“expandinst”。该虚拟指令是用于“在链接时表示警告而使数值操作数的扩张处理有效”的指示。从 (b) 的行到 (e) 的行之间的 (X) 的区间成为将寄存器 R29 用于通用寄存器用途的区间, 并且, 在该区间不实施在依据编译系统 1 的数值操作数溢出的情况下扩张处理。并且, 其它区间成为实施依据编译系统 1 的、使用了寄存器 R29 的数值操作数扩张处理的区间。

[0100] 本实施例的编译系统 1, 与实施例 1 涉及的编译系统 1 相同作为构成要素包括编译器 3、汇编器 4 以及链接器 5。

[0101] 编译器 3, 在生成代码时总是在文件(编译器生成汇编程序 102)的开头, 输出如图 10 所示的虚拟“expandinst_without_warning”。据此, 对于由编译器 3 自动生成的代码部分, 实施数值操作数的扩张处理, 并且, 在链接时不表示警告。

[0102] 图 12 是汇编器 4 的处理的流程图。

[0103] 首先, 汇编器 4, 将作为内部的变量的寄存器使用模式初始化为“0”(步骤 S41)。并且, 汇编器 4, 按所输入的汇编程序的每个行反复进行以下处理。

[0104] 寄存器用途虚拟指令受理部 42, 判断是否出现了虚拟指令“expandinst_without_warning”(步骤 S42)。在出现了的情况下(步骤 S42 的“是”), 寄存器用途虚拟指令受理部 42, 在寄存器使用模式设定“2”, 该“2”示出将寄存器 R29 用于数值操作数溢出的情况下扩张、而不表示警告(步骤 S45)。而且, 寄存器用途虚拟指令受理部 42, 判断是否出现了虚拟指令“noexpandinst”(步骤 S44)。在出现了的情况下(步骤 S44 的“是”), 寄存器用途虚拟指令受理部 42, 在寄存器使用模式设定“0”, 该“0”示出将寄存器 R29 作为通用寄存器使用(步骤 S45)。而且, 寄存器用途虚拟指令受理部 42, 判断是否出现了虚拟指令“expandinst”(步骤 S46)。在出现了的情况下(步骤 S46 的“是”), 在寄存器使用模式设定“1”, 该“1”示出将寄存器 R29 用于数值操作数溢出的情况下扩张、而表示警告(步骤 S47)。

[0105] 指令代码转换部 43 判断: 寄存器使用模式是“0”以外(将寄存器 R29 用于数值操作数溢出的情况下扩张); 并且是否出现了寄存器 R29 的记述(步骤 S48)。在判断是真的情况下(步骤 S48 的“是”), 可以知道编程员的记述错误, 因此, 指令代码转换部 43, 输出错误消息, 从而汇编器 4 的处理结束(步骤 S49)。

[0106] 而且, 指令代码转换部 43 判断是否全部满足下列(a) ~ (c) 的三个条件(步骤 S20): (a) 寄存器使用模式是“0”以外(将寄存器 R29 用于数值操作数溢出的情况下扩张); (b) 指令与图 3 的转换模式的任何单一指令符合; (c) 不能静态确定该指令的操作数的数值(步骤 S50)。在全部满足(a) ~ (c) 的条件的情况下(步骤 S50 的“是”), 指令代码转换部 43, 将该行的指令转换为转换模式的右边所述的多个指令。同时, 指令代码转换部 43, 作为关于转换的最佳化信息 106, 将从指令列的开头到该指令为止的指令地址偏置值和图 3 的转换模式的模式 ID, 附加给生成代码(对象文件 104)(步骤 S51)。并且, 指令代码

转换部 43, 将指令转换为机械语言代码, 从而输出对象文件 104(步骤 S52)。

[0107] 汇编器 4, 到不存在未处理的行为止执行所述处理。

[0108] 图 13 示出所述最佳化信息 106 的一个例子。针对实施例 1 中的最佳化信息 106 增加了寄存器使用模式, 将汇编器 4 所转换的机械语言代码列的开头到该部分为止的指令地址的偏置值、图 3 中的该转换模式的模式 ID、和寄存器使用模式的组合排列。各个组合与各个转换部分相对应。

[0109] 图 14 是链接器的处理的流程图。针对图 9 所示的实施例 1 中的链接器 5 的处理的流程图增加了步骤 S38 和步骤 S39。其它各个步骤的工作与实施例 1 中的链接器 5 的各个步骤的工作相同。下面, 说明代码最佳化部 52 的处理。

[0110] 代码最佳化部 52, 针对与所输入的各个对象文件 104 对应的代码块反复进行以下处理。代码最佳化部 52, 对于各个代码块, 对该对象文件 104 的最佳化信息 106 的各个项目、即图 13 的各个行反复进行以下处理。

[0111] 首先, 代码最佳化部 52, 依据最佳化信息 106 的指令地址偏置值, 确定由汇编器 4 展开为多个指令的部分。代码最佳化部 52, 还依据转换模式的模式 ID 确定转换模式。再者, 代码最佳化部 52, 判断是否可以将地址解决部 51 所确定的操作数的值, 放到图 3 所示的转换模式的左边所述的形式的数值操作数宽度(步骤 S35)。即, 代码最佳化部 52, 判断是否可以以单一指令来表示多个指令。

[0112] 在判断为可以将操作数的值放到数值操作数宽度的情况下(步骤 S35 的“是”), 代码最佳化部 52, 将符合的多个指令代码置换为与转换模式的左边所述的单一指令对应的代码, 并且将变化标记设定为“1”, 该“1”表示发生了代码的变化(步骤 S36)。

[0113] 在判断为不能将操作数的值放到数值操作数宽度的情况下(步骤 S35 的“否”), 代码最佳化部 52 判断最佳化信息的寄存器使用模式是否“1”(步骤 S38)。在寄存器使用模式是示出带有警告而进行数值操作数溢出时的扩张的“1”的情况下(步骤 S38 的“是”), 代码最佳化部 52 表示警告消息, 从而示出该部分在被展开为多个指令的状态下留下(步骤 S39)。在寄存器使用模式是“2”的情况下(步骤 S38 的“否”)即使该部分在被展开为多个指令的状态下留下, 也不表示警告消息。

[0114] 代码最佳化部 52, 对最佳化信息 106 的所有的项目反复进行所述处理。并且, 代码最佳化部 52, 对与所输入的各个对象文件对应的代码块反复进行所述处理。

[0115] 所述处理结束后, 链接器 5, 判断变化标记是否表示发生了代码的变化的“1”(步骤 S37)。在是“1”的情况下(步骤 S37 的“是”), 链接器 5 返回到步骤 S31 而反复进行到此为止的处理, 在不是“1”的情况下(步骤 S37 的“否”), 链接器 5 结束处理。

[0116] 通过如上构成所述的汇编器 4 以及链接器 5, 针对汇编器 3 所生成的汇编生成程序, 操作数的数值溢出时的向多个指令的展开时可以抑制警告消息的表示, 针对程序员记述汇编程序 103, 通过虚拟指令可以控制操作数溢出时的警告消息的表示。据此, 可以容易确定因程序员记述部分而引起的多个指令的展开部分, 从而性能以及代码大小调谐等的开发效率会提高。

[0117] 如上所述, 对于本发明涉及的编译系统的构成要素, 根据实施例进行了说明, 但是, 本发明不仅限于所述实施例。例如:

[0118] (1) 在所述实施例中设想对 C 语言的编译系统, 但是本发明不仅限于 C 语言。在采

用其它程序语言的情况下也可以保持本发明的意义。

[0119] (2) 在所述实施例中,作为目标处理器设想固定长度指令的处理器,但是本发明不仅限于此。本发明也可以适用于以一个指令可以处理的操作数的数值宽度有限制的、较多的可变长指令的处理器。

[0120] (3) 在所述实施例中,编译系统由驱动器、编译器、汇编器构成,但是本发明不仅限于此结构。例如,可以分解为更多的构成要素或辅助工具,也可以以一个构成要素来实现所有的功能。

[0121] (4) 在所述实施例中,作为指令的转换模式采用了单一指令和多个指令之间的转换,但是本发明不仅限于此转换模式。即,也可以是:多个指令和多个指令之间的转换;单一指令和其它单一指令之间的转换,该单一指令和其它单一指令的助记符不同;单一指令和其它单一指令之间的转换,该单一指令和其它单一指令的操作数宽度不同。为了实现这些转换,适当地修正图 3 的转换模式即可。据此,本发明,不仅可以适用于数值操作数的扩张,也可以适用于按照性能、成本或消耗电力的代码的选择的用途。

[0122] (5) 在所述实施例中,作为指令的转换模式采用了单一指令和多个指令之间的两种模式间的转换,但是本发明不仅限于此转换模式。也可以是,依据三种以上的模式的多个阶段的转换。例如,单一指令的短指令、单一指令的长指令和多个指令的三个阶段,单一指令、较少的多个指令和较多的多个指令的三个阶段。为了实现所述内容将图 3 的转换模式修正为三个阶段以上即可。据此可以进行更仔细的指令模式的选择。

[0123] (6) 在所述实施例中,作为汇编器所输入的汇编程序是编译器所生成的、还是程序员所记述的的判断单元,采用了汇编虚拟指令,但是本发明不仅限于此接口。例如,可以依据选项指定来传达汇编程序的种类,也可以依据编译指示指令 (#pragma 指令) 来传达汇编程序的种类。在源程序的编辑困难的情况下,或在希望更仔细地指定的情况下等,根据需要选择单元即可。

[0124] (7) 在所述实施例中,作为程序员指定寄存器的用途和该区间的用户接口,采用了对汇编器的虚拟指令,但是本发明中的传达单元不仅限于此接口。例如,可以依据选项指定来传达,也可以依据 #pragma 指令来传达。而且,作为传达方法,不仅是所述实施例那样的、在区间的入口和出口记述虚拟指令的方法,也可以是在文件首部记述区间的信息的形式。再者,也可以是,作为指定范围的粒度,进行以作为更小的粒度的操作数寄存器为单位的指定。例如,也可以是,根据用途使操作数寄存器的表示方法不同 (R29 和 Rx29 等),而按每个寄存器进行指定。例如,R29 是使用寄存器 R29 的指定,Rx29 是使用寄存器 R29 以外的寄存器的指定。或者,可以使按每个指令指定寄存器用途,也可以是作为大的粒度例如以文件为单位指定。

[0125] (8) 在所述实施例中,作为程序员指定寄存器的用途和该区间的用户接口,采用了对汇编器的虚拟指令,但是本发明中的用途以及区间的指定方法不仅限于此接口的有无。例如,可以如下构成:在没有虚拟指令的情况下由汇编器解析汇编程序,将使用寄存器 R29 的区间判断为通用寄存器用途区间。据此,程序员不需要明显地指示,因此程序制作效率会提高。

[0126] (9) 在所述实施例中,将寄存器 R29 共享于通用寄存器用途、和数值操作数溢出时的指令展开用途的两种用途,但是本发明不仅限于此寄存器的用途。例如,也可以是,将寄

存器 R29 共享于通用寄存器用途、和用于 DMA(Direct Memory Access) 传送的 DMA 控制器或专用运算器等的硬件控制寄存器用途的两种用途。据此,可以以软件使用作为硬件专用预备的资源,从而所生成的代码中的寄存器使用效率会提高。

[0127] (10) 在所述实施例中,作为在数值操作数溢出时的指令分解用固定地使用寄存器 R29,但是本发明不仅限于此。例如,可以如下构成:将在各个部分可以用于指令分解用途的寄存器的集合作为最佳化信息,从编译器传达给汇编器以及链接器,由汇编器以及链接器将从最佳化信息所示的集合中选择的寄存器活用于指令分解用途。据此,寄存器的使用灵活性会提高,从而所生成的代码中的寄存器的使用效率会提高。

[0128] (11) 并且,也可以是,编译器,对目标处理器具有的多个寄存器,在 C 语言程序中,确定寄存器用于指令展开用途的指令,而将关于该指令的信息传达给汇编器以及链接器。也可以是,汇编器以及链接器,接受该信息后根据该信息选择用于通用寄存器用途的寄存器。

[0129] 应该认为的是,这次所公开的实施例中的所有内容是举例说明的,因此不仅限于此例子。对于本发明的范围,不是依据所述说明示出的、而是依据权利要求的范围示出的,并且,意味着包含与权利要求的范围相等的意思以及范围内的所有的变更。

[0130] 本发明可以适用于将以 C 语言或汇编语言记述的源程序转换为机械语言程序的编译系统。

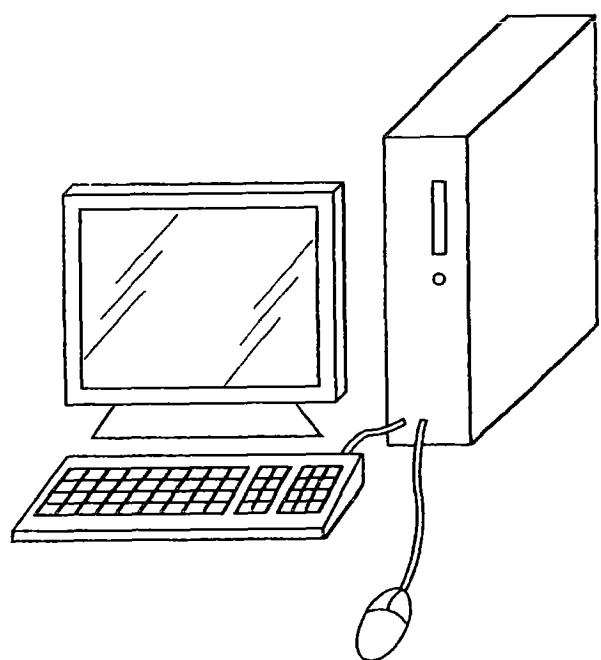
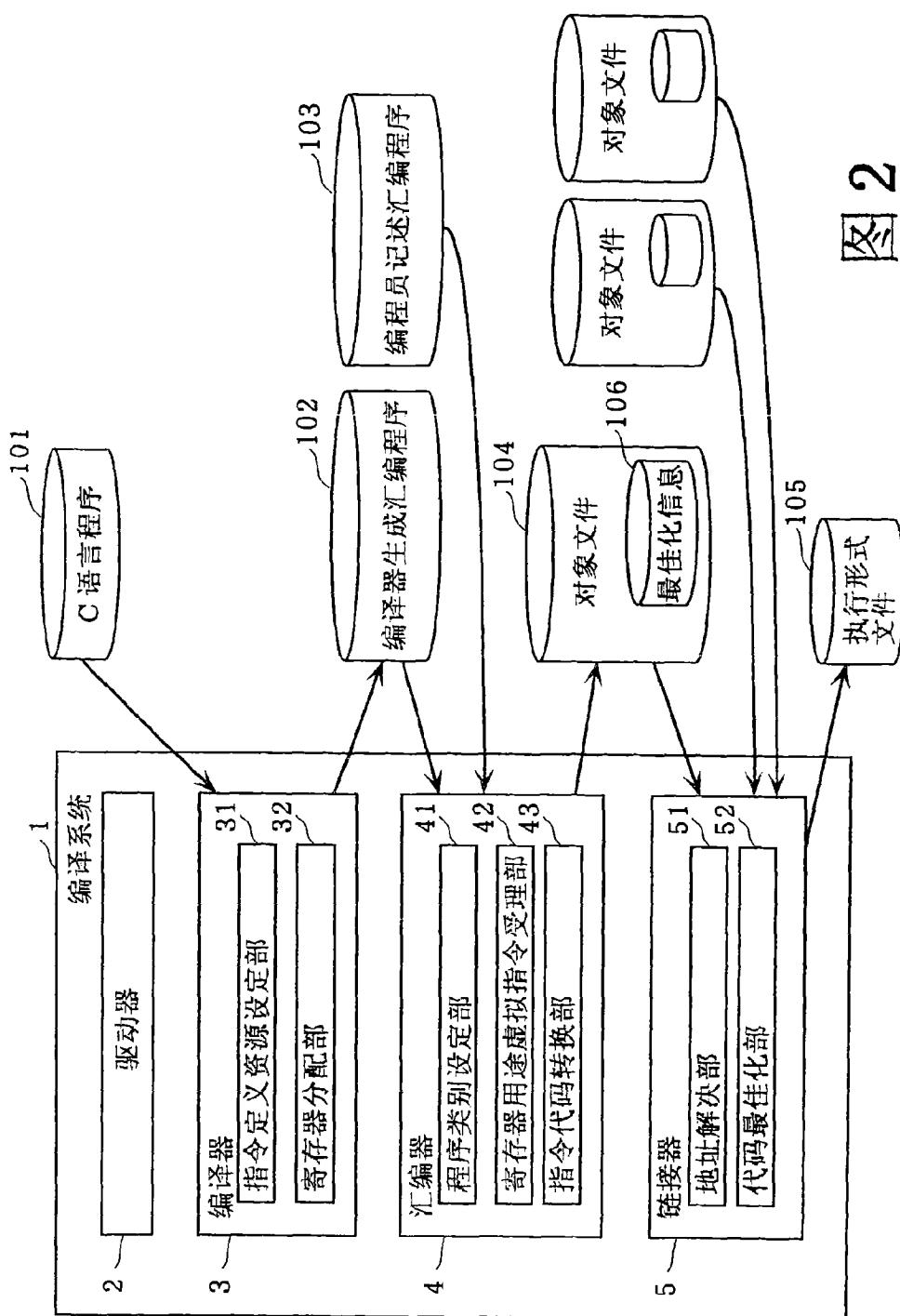


图 1



模式 ID	单一指令	多个指令
1	add R1, imm16	setlo R29 ,LO(imm32) sethi R29 ,HI(imm32) add R1, R29
2	ld R0,(R1,disp10)	setlo R29 ,LO(disp32) sethi R29 ,HI(disp32) add R29 ,R1 ld R0, (R29)
3	br disp20	setlo R29 ,LO(addr32) sethi R29 ,HI(addr32) mov TAR, R29 jmp TAR

图 3

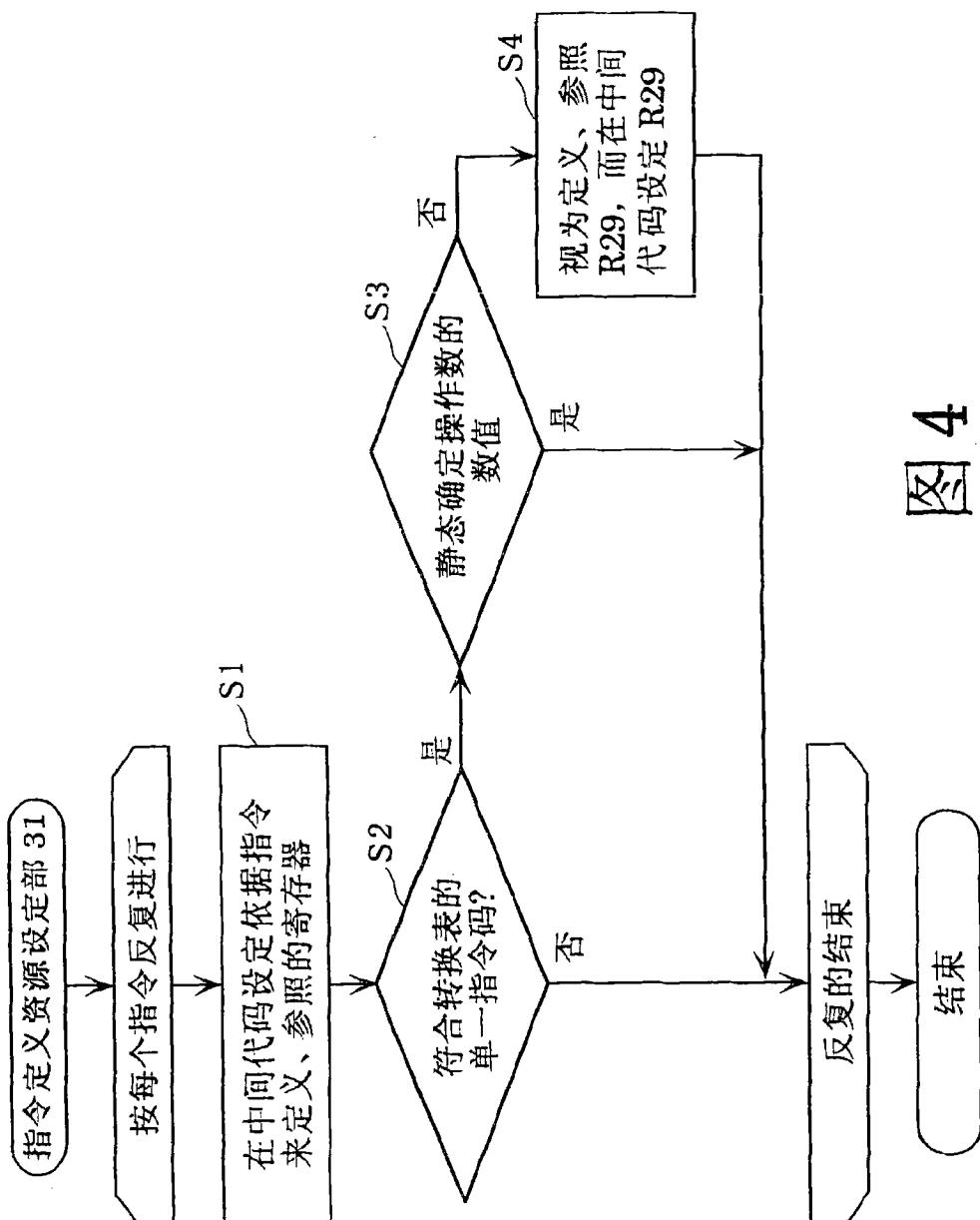


图 4

指令地址偏置值	转换模式 ID
32	2
160	3
248	2
536	1

图 5

编译器生成汇编程序 102

```
FILE test . c ... (a)

func1 :
    add  R29, R0, R1      ... (b)
    ld   R0, (R2, 4)       ... (c)
    sub  R0, R29           ... (d)
    ret

END
```

图 6

编程员记述汇编程序

103

```
func2 :  
    ld R3, (gp, _value - _gptop)  
    add R3, R0  
    ...  
  
    ENABLE_R29 ON      ... (e)      —————↑  
    mov R29, R0          ... (f)      |  
    ...  
    add R1, R29          ... (g)      |  
    ...  
    ENABLE_R29 OFF     ... (h)      —————↓  
    call func3  
    ret  
  
END
```

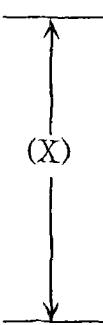


图 7

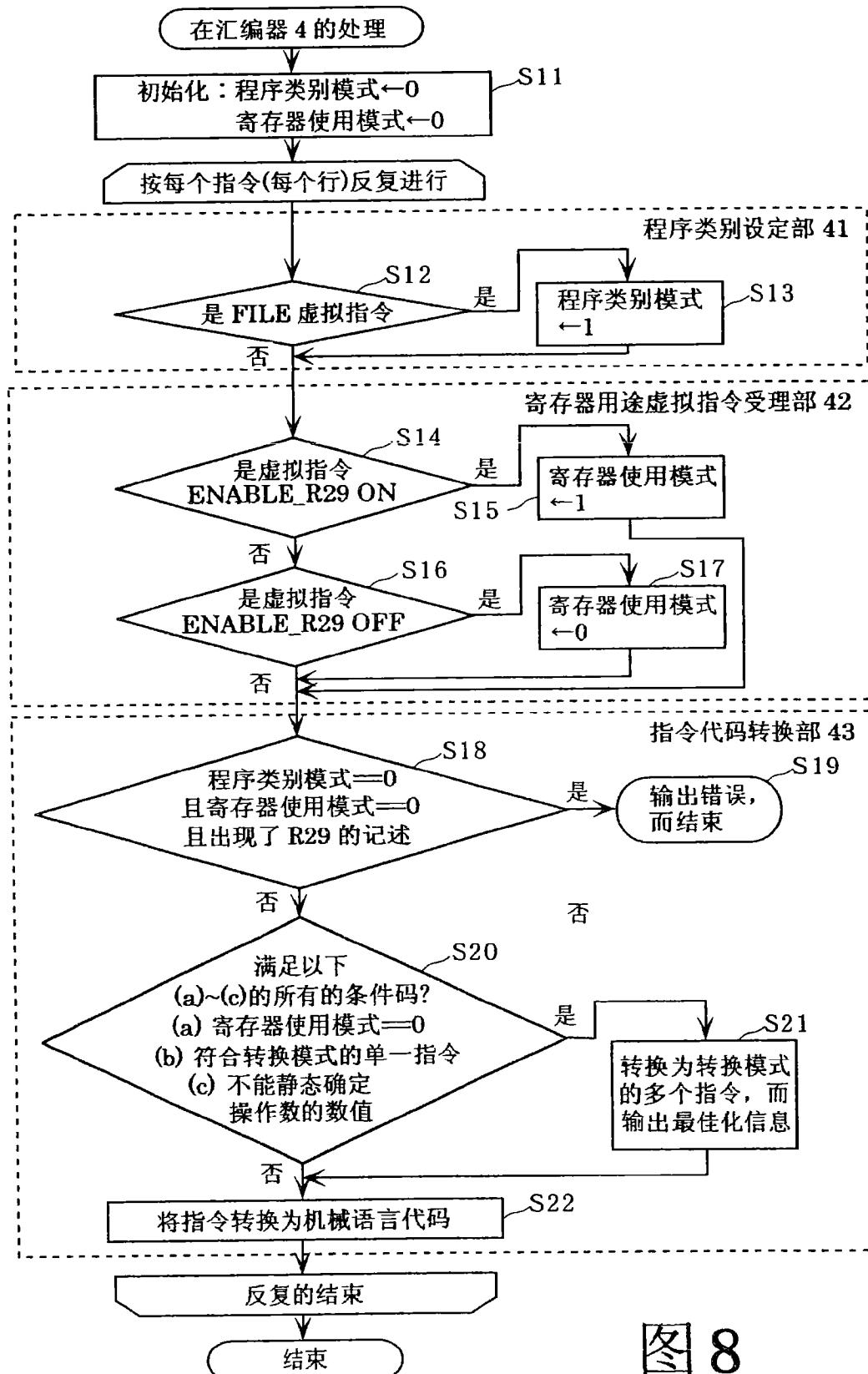


图 8

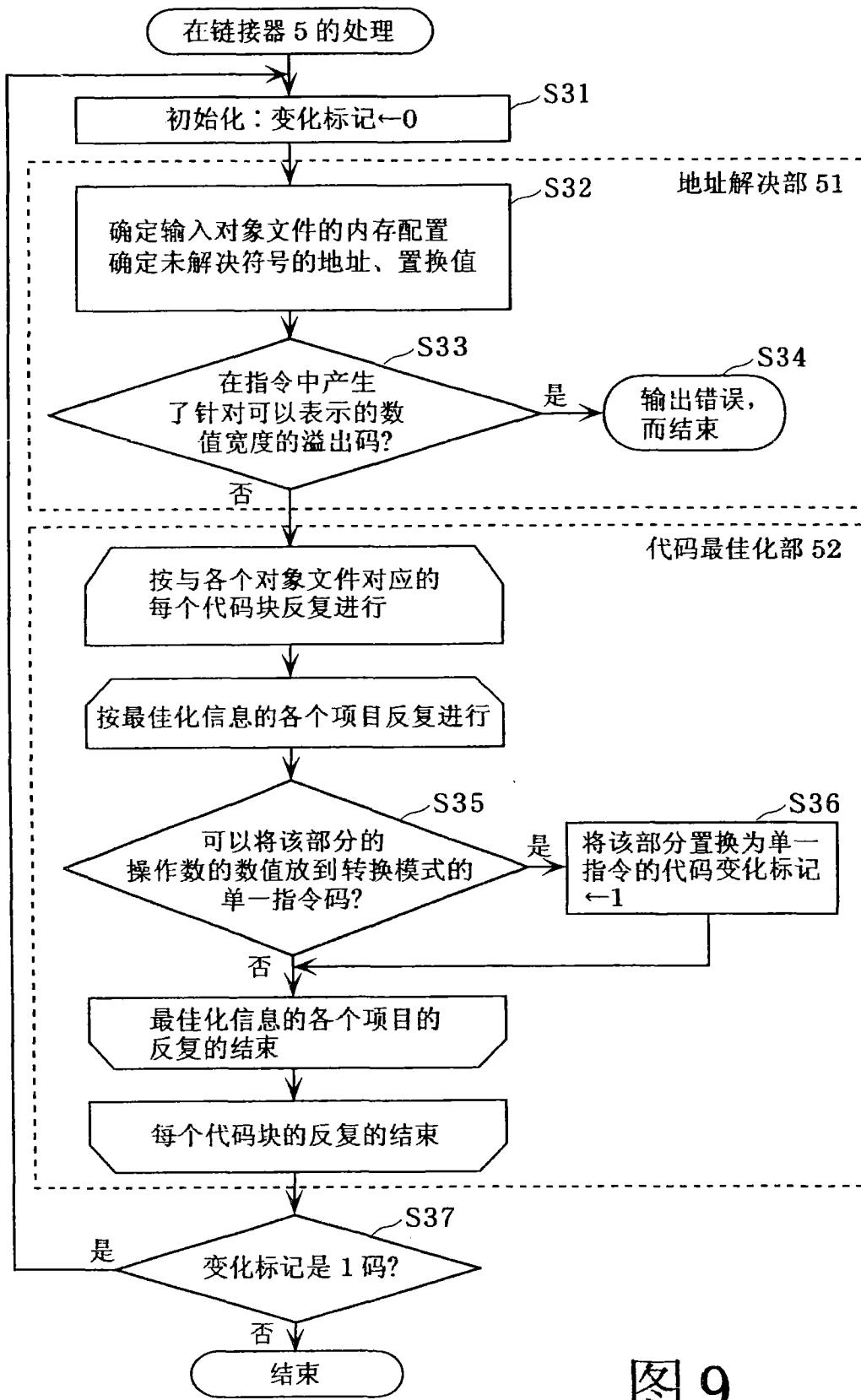


图 9

编译器生成汇编程序 102

```
expandinst_without_warning ... (a)

func1 :
    add  R25, R0, R1
    ld   R0, (R2, 4)
    sub  R0, R25
    ret

END
```

图 10

编程员记述汇编程序

103

```
func2 :  
    ld  R3, (gp, _value - _gptop)  
    add R3, R0  
    ...  
  
    noexpandinst      ... (b)      ——↑  
    mov  R29, R0      ... (c)      |  
    ...  
    add  R1, R29      ... (d)      |  
    ...  
  
    expandinst        ... (e)      ——↓  
    call func3  
    ret
```

END

图 11

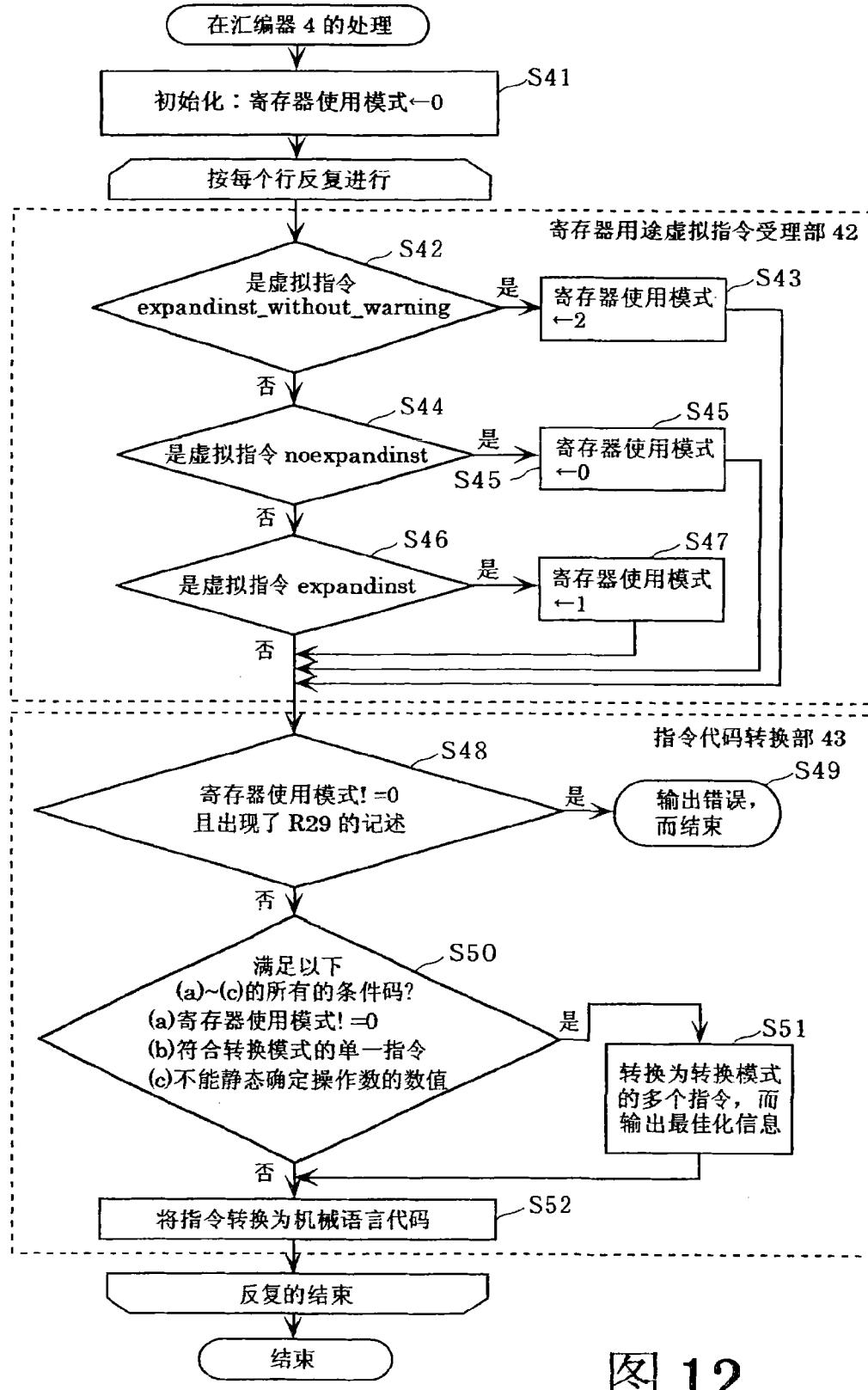


图 12

指令地址偏置值	转换模式 ID	寄存器使用模式
32	2	1
160	3	2
248	2	2
536	1	1

图 13

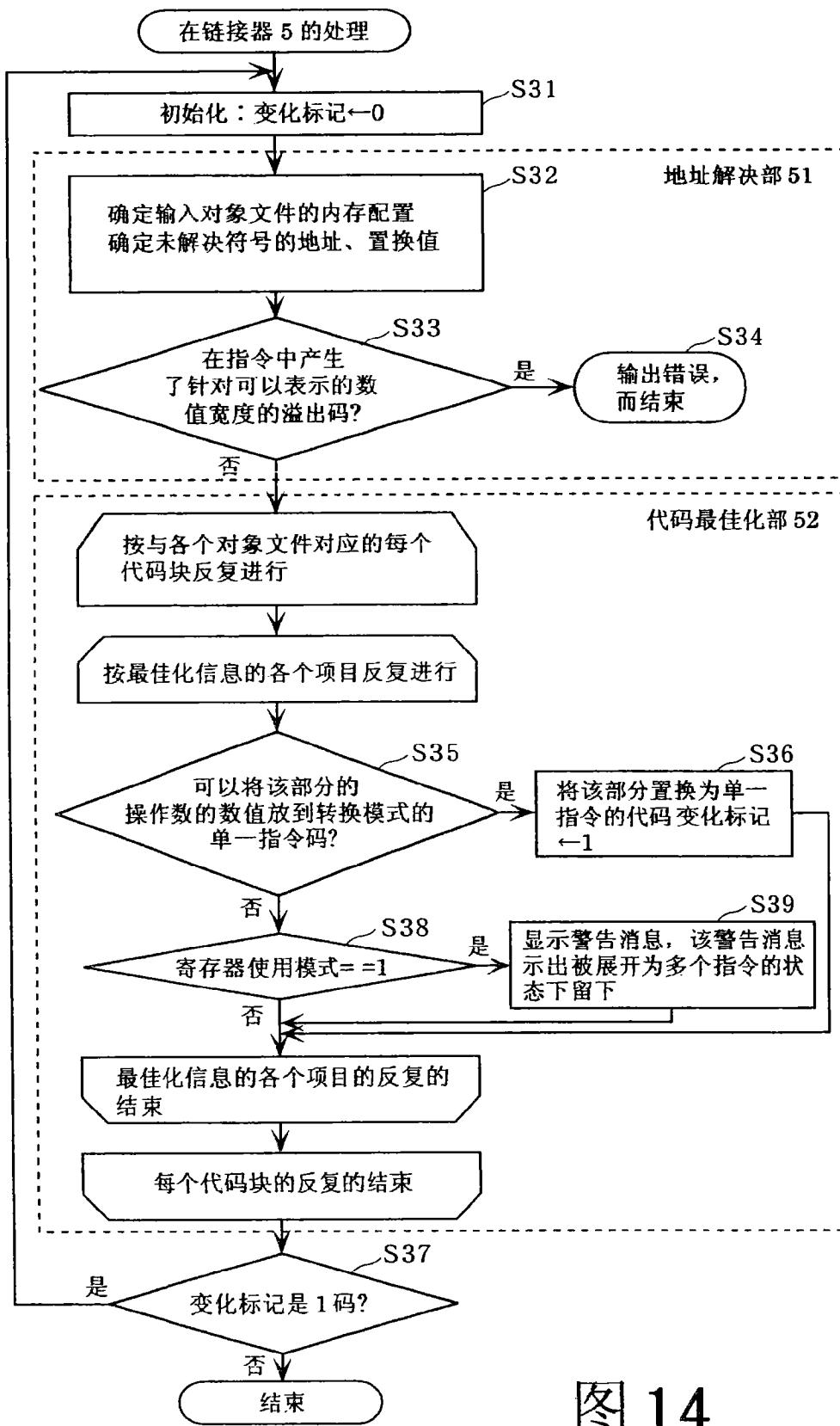


图 14