[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/30
G06F 9/318 G06F 9/32
G06F 9/38

[12] 发明专利申请公开说明书

[21] 申请号 200410005379.1

[43] 公开日 2004 年 8 月 18 日

[11] 公开号 CN 1521618A

[22] 申请日 1998.8.28
[21] 申请号 200410005379.1
分案原申请号 98120299.3
[30] 优先权
[32] 1997. 8.29 [33] JP [31] 234354/1997
[32] 1998. 4. 8 [33] JP [31] 95645/1998
[71] 申请人 松下电器产业株式会社
地址 日本大阪府门真市
[72] 发明人 高山秀一 小谷廉介 田中旭
桧垣信生 铃木正人 田中哲也
瓶子岳人 宫地信哉

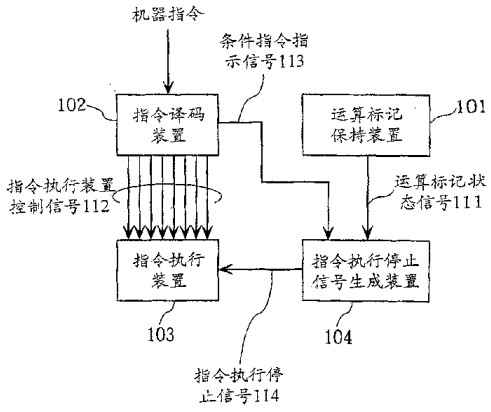
[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 叶恺东

权利要求书 2 页 说明书 30 页 附图 23 页

[54] 发明名称 对指令列译码并执行指令的处理器

[57] 摘要

一种译码和执行指令列的处理器，其特征在于，包括：输入装置，用于输入由分配在该处理器的指令集中的指令构成的指令列；译码装置，可以对分配到所述指令集中的指令译码并对所述输入装置输入的指令列中的每一个指令逐一译码，条件标记，用于保持预定条件成立与否的判断结果；判断装置，在所述译码装置对第一条件译码时，判断该第一条件是否成立，并将该判断结果保持到所述条件标记中，在所述译码装置对第二条件译码时，判断该第二条件是否成立，并将该判断结果保持到所述条件标记中；执行装置，在所述译码装置对条件成立时执行指令所包含的操作代码进行译码的情况下，只有所述条件标记所保持的判断结果成立时，才执行该操作代码表示的操作。



ISSN1008-4274

1.一种译码和执行指令列的处理器，其特征在于，包括：

输入装置，

用于输入由分配在该处理器的指令集中的指令构成的指令列；

其中，上述指令列中，包括一种以上的第一条件标记设定指令、一种以上的第二条件标记设定指令和一种以上的条件成立时执行指令；

其中，第一条件和第二条件设为相互具有排他关系，第一条件标记设定指令包含第一条件，而第二条件标记设定指令包含所述第二条件；

其中，条件成立时执行指令包含操作代码，并且相同操作代码的条件成立时执行指令的种类为一种；

译码装置，

可以对分配到所述指令集中的指令译码并对所述输入装置输入的指令列中的每一个指令逐一译码，

在第一条件标记设定指令译码时，对该第一条件标记设定指令包含的第一条件进行译码，

在第二条件标记设定指令译码时，对该第二条件标记设定指令包含的第二条件进行译码，

在条件成立时执行指令译码时，对各个条件成立时执行指令所包含的操作代码进行译码；

条件标记，

用于保持预定条件成立与否的判断结果；

判断装置，

在所述译码装置对第一条件译码时，判断该第一条件是否成立，并将该判断结果保持到所述条件标记中，

在所述译码装置对第二条件译码时，判断该第二条件是否成立，并将该判断结果保持到所述条件标记中；

执行装置，

在所述译码装置对条件成立时执行指令所包含的操作代码进行译码的情况下，只有所述条件标记所保持的判断结果成立时，才执行该操作代码表示的操

作。

2.如权利要求1所述的处理器，其特征在于，第一条件标记设定指令和第二条件标记设定指令指定两个运算对象甲和乙；

其中，上述译码装置将甲与乙相等和甲与乙不等中的任一方作为第一条件的第一条件标记设定指令，和把另一方作为第二条件的第二条件标记设定指令译码，以及将甲大于或等于乙，和甲小于乙中的任一方作为第一条件的第一条件标记设定指令，和把另一方作为第二条件的第二条件标记设定指令译码，以及把甲小于或等于乙，和甲大于乙的任一方作为第一条件的第一条件标记设定指令，和把另一方作为第二条件的第二条件设定指令译码。

3．如权利要求1所述的处理器，其特征在于，第一条件标记设定指令和第二条件标记设定指令指定两个运算对象甲和乙；

其中，上述译码装置将甲和乙不带符号而作比较，并将甲与乙相等和甲与乙不等中的任一方作为第一条件的第一条件标记设定指令，和把另一方作为第二条件的第二条件标记设定指令译码，以及将甲和乙不带符号而作比较，并将甲大于或等于乙，和甲小于乙中的任一方作为第一条件的第一条件标记设定指令，和把另一方作为第二条件的第二条件标记设定指令译码，以及将甲和乙不带符号而作比较，并把甲小于或等于乙，和甲大于乙的任一方作为第一条件的第一条件标记设定指令，和把另一方作为第二条件的第二条件设定指令译码；以及将甲和乙带符号而作比较，并将把甲大于或等于乙，和甲小于乙中的任一方作为第一条件的第一条件标记设定指令，和把另一方作为第二条件的第二条件标记设定指令译码，以及将甲和乙带符号而作比较，并把甲小于或等于乙，和甲大于乙的任一方作为第一条件的第一条件标记设定指令，和把另一方作为第二条件的第二条件设定指令译码。

4.如权利要求2所述的处理器，其特征在于，所述译码装置在可以译码的条件成立时执行指令所包含的操作代码为传送、运算及分支中的任一种或三种。

对指令列译码并执行指令的处理器

本申请是申请号为98120299.3、申请日为1998年8月28日的原案申请的分案申请，该原案的在先申请为JP97-234354，在先申请日为1997年8月29日。

技术领域

本发明涉及处理器及指令变换装置，特别涉及在装入了条件指令的情况下削减指令种类和硬件数量的技术。

背景技术

随着近年来应用微处理器的产品的多功能化和高速化，希望出现处理能力更高的微处理器（以下简称为“处理器”）。

一种用于处理器高速化的基本技术是流水线处理。

流水线处理就是将1个指令的处理过程分成多个较小的处理单位（流水线级），通过同时执行各个流水线级来提高处理速度的技术。

在这样的流水线处理中，存在在分支时产生流水线的混乱（失速），而使流水线的执行性能低于理想性能的问题。这种现象就是所谓的分支危险性。

在近年的处理器中，为了削减分支危险性，利用条件指令来代替分支指令，以提高处理能力。条件指令的相关内容在例如“The ARM RISC Chip A Programmer's Guide、Addison-Wesley Publishing Company Inc.的p.62-63中予以详细记载。

图30示出了现有的包含条件传送指令的指令列。图30中所示的“r0”、“r1”、“r2”分别表示寄存器。指令3001是将数值“1”传送到寄存器“r0”中的传送指令。指令3002是将寄存器“r1”和“r2”的比较结果反映到各种标记中的比较指令。指令3003是条件传送指令，用于在参照各种标记并由指令3002所比较的两个比较对象相等时，将数值“0”传送到寄存器“r0”中。

图31示出了现有的条件传送指令。这个条件传送指令3101有6种。条件3102是表示各个条件传送指令所指定的条件的记号。在比较指令比较两个运算对象a和运算对象乙时，条件包括“甲等于乙”、“甲不等于乙”、“甲大于乙”、“甲大于等于乙”、“甲小于乙”和“甲小于等于乙”，并且各条件传送指令在各个条件成立时执行传送指令。

图32示出了现有的比较指令（CMP指令）、在条件成立时执行传送指令的条件传送指令、在条件成立时执行分支指令的条件分支指令。这里，用指令缩写的操作码末尾两个字符来指定条件。

由于图32所示的各条件指令和条件分支指令的条件是将比较条件作为符号数据附加到图31所示的条件中，所以各条件指令和条件分支指令各有10种。

从而，比较指令、两个操作的各条件指令和条件分支指令合计共有31种。

也有削减了指令种类的条件指令。关于这样的指令，在例如“日立单片RISC微机SH7000/SH7600系列编程手册：（株）日立制作所发行”P.57-58、P.69-70、P.75-78中有详细记载。

图33示出了削减了指令种类的比较指令、条件相加指令、条件传送指令和条件分支指令。

由于图33所示的各条件指令和条件分支指令的条件只有条件标记被设置或条件标记被清零两种，所以各条件指令和条件分支指令各有两种，并有5种用于设置条件标记/清零条件标记的比较指令。

从而，比较指令、两个操作的条件指令和条件分支指令的指令种类共有11种。

在进行流水线处理的处理器中，为了更多地削减分支危险，需要具备更多种操作的条件指令。

但是，由于处理器的指令为位数定长模式，所以处理器能够具备的指令种类是有限的。

这样，处理器能够具备的条件指令的种类也是有限的，在增加指令种类时用于对增加的指令进行译码的硬件是必要的，从而增加了处理器成本。

发明内容

针对上述问题，本发明的目的在于提供一种在装入了条件指令的情况下，削减指令种类的指令变换装置和削减了硬件数量的处理器。

（1） 为了达到上述目的，依据本发明的一种对指令列译码并执行指令的处理器，其特征在于，包括：状态保持装置，用于在执行预定指令时保持表示其执行结果的状态的运算标记；输入装置，用于输入由分配给该处理器的指令集的指令构成的指令列；条件指令是包含状态条件和操作代码的指令；状态条件用于指定一种以上的运算标记的状态；其中，在上述指令列中，在操作代码

相同而状态条件不同的多个条件指令存在时，在包含于该多个条件指令中的各状态条件中的任何两个之间，排他关系都不成立；译码装置，可以对分配到所述指令集中的指令译码，并对所述输入装置输入的指令列中的每一个指令逐一译码；在这里，译码装置对于操作代码相同而在状态条件排他关系成立的两个条件指令，具有将一方的条件指令译码的功能，而不具有将另一方的条件指令译码的功能；判断装置，用于判断所述状态保持装置所保持的运算标记是否由所述译码装置所译码的条件指令中所包含的状态条件所指定的状态；执行装置，用于只有在所述判断装置作出肯定判断时，才执行所述译码装置所译码的包含在条件指令中的操作代码所示的操作。

由此，由于依据本发明的处理器只将指定具有排他关系的两个条件中的任一个条件的条件指令分配到指令集中而不将指定另一个条件的条件指令分配到指令集中，所以与现有技术相比，减少了条件指令的种类。

这样，能够削减指令译码器的硬件数量并在指令种类有限的情况下，能够具备更多种操作的条件指令，所以在进行流水线处理的处理器中能够更多地削减分支危险。

（2）也可以是所述状态保持装置保持着作为比较指令的执行结果的两个比较对象甲和乙的大小关系；在这里，所述译码装置具有在包含相同操作代码的条件指令中，将包含甲与乙相等，和甲与乙不等的任何一方的状态条件的条件指令译码的能功，而不具有将包括另一方的状态条件的条件指令译码的功能，又具有将包含甲大于，和甲小于乙的任何一方的状态条件的条件指令译码的功能，而不具有将另一方的状态条件的条件指令译码的功能，又，具有将包含甲小于，和甲大于乙的任何一方的状态条件的条件指令译码的功能，而不具有将另一方的状态条件的条件指令译码的功能。

由此，由于依据本发明的处理器对每种操作具备三种条件指令，所以与现有技术的6种相比，条件指令的种类减少一半。

这样，能够削减指令译码器的硬件数量并在指令种类有限的情况下，能够具备现有技术的2倍操作的条件指令，所以能够在进行流水线处理的处理器中更多地削减分支危险。

（3）也可以是所述译码装置能够译码的条件指令所包含操作代码为传送和运算中的一种，或二者兼备。

由此，分配在依据本发明的处理器的指令集的一部分中的条件指令所指定的操作为传送及运算中的任一种或二者兼备。

这样，能够在全部传送指令及运算指令中分配条件指令，从而能够在进行流水线处理的处理器中更多地削减分支危险。

（4）为了达到上述目的，依据本发明的处理器包括：

输入装置，用于输入由与分配在该处理器的指令集中的指令相对应的指令构成的指令列；

其中，（一种以上的）第一条件标记设定指令、（一种以上的）第二条件标记设定指令和（1种以上的）条件成立时执行指令被分配到所述指令集的一部分中；

条件标志设定指令是包含条件的指令，

其中，第一条件标记设定指令包含相互具有排他关系的第一条件和第二条件中的第一条件，而第二条件标记设定指令包含所述第二条件；

条件成立时执行指令是包含操作代码的指令，

其中，条件成立时执行指令包含操作代码，并且相同操作代码的条件成立时执行指令的种类为1种。

译码装置，

可以对分配到所述指令集中的指令译码并对所述输入装置输入的指令列中的每一个指令逐一译码，

在对第一条件标记设定指令译码时，对该第一条件标记设定指令包含的第一条件进行译码，

在对第二条件标记设定指令译码时，对该第二条件标记设定指令包含的第二条件进行译码，

在对条件成立时执行指令译码时，对各个条件成立时执行指令所包含的操作代码进行译码；

（一个）条件标记，

用于保持预定条件成立与否的判断结果；

判断装置，

在所述译码装置对第一条件译码时，判断该第一条件是否成立，并将该判断结果保持到所述（一个）条件标记中，

在所述译码装置对第二条件译码时，判断该第二条件是否成立，并将该判断结果保持到所述（1个）条件标记中；

执行装置，

在所述译码装置对条件成立时执行指令所包含的操作代码进行译码的情况下，只有所述条件标记所保持的判断结果成立时，才执行该操作代码表示的操作。

由此，由于依据本发明的处理器将条件标记设定指令和条件成立时执行的指令分配到指令集中而不将条件不成立时执行的指令分配到指令集中，所以对每种操作有1种分配到指令集的一部分中的条件成立时执行的指令。

由此，与现有技术中的每种操作有10种带有条件时执行的指令和2种条件成立时执行的指令相比，条件成立时执行的指令的操作越来越多了而条件成立时执行的指令的种类减少了。

这样，能够削减指令译码器的硬件数量，实现低成本并在指令种类有限的情况下，能够具备更多种操作的条件指令，所以能够在进行流水线处理的处理器中更多地削减分支危险。

（5）也可以是，第一条件标记设定指令和第二条件标记设定指令指定两个运算对象甲和乙；

其中，第一条件标记设定指令和第二条件标记设定指令共计6种：以甲等于乙、甲不等于乙、甲大于等于乙、甲小于乙、甲小于等于乙、甲大于乙为第一条件及第二条件。

由此，由于依据本发明的处理器具备6种条件标记设定指令并这些条件中的每两个之间各自具有排他关系，所以即使不将条件不成立时执行的指令分配到指令集中，也能够进行与现有技术相同的处理。

从而，能够削减指令种类而不降低功能。

（6）也可以是，所述译码装置能够译码的第一条件标记设定指令和第二条件标记设定指令共计10种：在通过使所述两个比较对象甲和乙带有符号来进行比较的情况下，以甲大于等于乙、甲小于乙、甲小于等于乙、甲大于乙为第一条件及第二条件。

由此，由于依据本发明的处理器具备10种条件标记设定指令并这些条件中的每两个之间各自具有排他关系，所以即使不将条件不成立时执行的指令分配

到指令集中，也能够进行与现有技术相同的处理。

从而，能够削减指令种类而不降低功能。

（7）也可以是，所述译码装置可以译码的条件成立时执行指令所包含操作代码为传送、运算及分支中的任一种，或三者兼备。

由此，分配在依据本发明的处理器的指令集的一部分中的条件成立时执行的指令所指定的操作为传送、运算及分支中的任一种或三者兼备。

这样，能够在全部传送指令、运算指令及分支指令中分配条件成立时执行的指令，从而能够在进行流水线处理的处理器中更多地削减分支危险。

（8）为了达到上述目的，依据本发明的指令变换装置包括：

其中，条件指令是包含条件和操作代码并且只有在此条件成立时才由处理器执行此操作代码所表示的操作的指令；

输入装置，用于输入不包含条件指令的指令列；

指令列检测装置，用于从所述输入装置输入的指令列中检测出根据预定（一个）条件是否成立而向同一存储对象分别传送不同传送对象的变换对象指令列；

判断装置，用于判断包含与所述变换对象指令列所表示的预定条件相同的条件的条件指令是否被分配到专用处理器的指令集中；

变换装置，

当所述判断装置的判断结果为已被分配时，将所述变换对象指令列变换为包含所述预定条件的条件指令的指令列，

当所述判断装置判断结果为未被分配时，将所述变换对象指令列中所述不同传送对象进行相互替换，变换为包含与所述预定条件具有排他关系的条件的条件指令的指令列。

依据本发明而如此构成的指令变换装置不生成不能用专用处理器译码的条件指令，而生成可以用专用处理器译码的条件指令。

由此，由于通过条件指令的变换数相同能够削减其种类，所以用于分配指令的字段减少了，代码长度被削减了并且能够削减可以用专用处理器译码的条件指令的种类而不降低功能。

这样，能够削减专用处理器中的指令译码器的硬件数量，并在指令种类有限的情况下，能够具备更多种操作的条件指令，所以能够在进行流水线处理的

处理器中更多地削减分支危险。

（9）也可以是，包含所述判断装置判断出未被分配时的条件指令的条件和具有排他关系的条件的条件指令被分配到所述专用处理器的指令集中；

在指定相同操作的条件指令之间，被分配到所述专用处理器的指令集中的任一个条件指令的条件与指定相同操作的另外任一个条件指令的条件不具有排他关系。

由此，由于判断装置判断出只将指定具有排他关系的两个条件中的任一个条件的条件指令分配到专用处理器的指令集中，所以依据本发明的指令变换装置不生成指定另一个条件的条件指令。

（10）也可以是，对于作为所述预定比较指令的执行结果的两个比较对象甲和乙的大小关系，分配在专用处理器的指令集中并包含相同操作代码的条件指令包含三种：在执行预定比较指令后执行条件指令的情况下，分别以甲等于乙及甲不等于乙中的任一个、以甲大于等于乙及甲小于乙中的任一个、以甲小于等于乙及甲大于乙中的任一个为条件。

由此，由于判断装置判断出只将每种操作的三种条件指令作为可以用专用处理器译码的指令，所以在依据本发明的指令变换装置中，与现有技术的6种相比，条件指令的种类减少了一半。

（11）也可以是，所述传送对象为数值、不同于所述存储对象的存储对象所表示的数值及它们的运算结果中的任一个。

由此，条件指令所指定的操作为传送及运算中的任一种。

这样，能够在全部传送指令及运算指令中分配条件指令，从而能够在进行流水线处理的专用处理器中更多地削减分支危险。

（12）也可以是，所述变换对象指令列顺序由在所述预定条件成立时分支到后三个指令的条件分支指令、将所述传送对象传送到所述存储对象中的传送指令、跳转到后两个指令的无条件分支指令和将不同于所述传送对象的传送对象传送到所述存储对象中的传送指令构成。

由此，指令列检测装置能够检测出这样的指令列。

（13）为了达到上述目的，依据本发明的指令变换装置包括：

其中，条件指令是包含条件和操作代码并且只有在此条件成立时才由处理器执行此操作代码所表示的操作的指令；

输入装置，用于输入包含条件指令的指令列；

条件指令检测装置，用于检测出包含在所述输入装置输入的指令列中的条件指令；

第一判断装置，用于判断所述条件指令检测装置所检测出的条件指令是否被分配到专用处理器的指令集中；

第二判断装置，用于在所述第一判断装置判断出未分配的情况下，根据所述条件指令检测装置检测出的条件指令所包含的预定条件是否成立，判断所述输入装置输入的指令列是否包含向同一存储对象传送不同传送对象的变换对象指令列；

变换装置，用于在所述第二判断装置判断出包含变换对象指令列的情况下，对该变换对象指令列中的所述不同传送对象进行相互替换，并将所述条件指令变换为包含与所述预定条件具有排他关系的条件的条件指令。

依据本发明而如此构成的指令变换装置能够将不能用专用处理器译码的条件指令变换成能用专用处理器译码的条件指令。

由此，由于一边变换成原有数量的条件指令，一边能够削减其种类，所以能够减少用于指令分配的字段，削减代码长度，并能够削减可以用专用处理器译码的条件指令的种类而不降低功能。

这样，能够削减专用处理器中的指令译码器的硬件数量，并在指令种类有限的情况下，能够具备更多种操作的条件指令，所以能够在进行流水线处理的处理器中更多地削减分支危险。

（14）也可以是，包含所述第一判断装置判断出未被分配时的条件指令的条件和具有排他关系的条件的条件指令被分配到所述专用处理器的指令集中；

在指定相同操作的条件指令之间，被分配到所述专用处理器的指令集中的任一个条件指令的条件与指定相同操作的另外任一个条件指令的条件不具有排他关系。

由此，由于第一判断装置判断出只将指定具有排他关系的两个条件中的任一个条件的条件指令分配到专用处理器的指令集中，所以依据本发明的指令变换装置不生成指定另一个条件的条件指令。

（15）也可以是，所述变换对象指令列顺序由比较两个运算对象的比较指令、将预定传送对象传送到预定存储对象中的传送指令、和在预定条件成立时

将不同于所述预定传送对象的传送对象传送到所述预定存储对象中的条件指令构成。

由此，指令列检测装置能够检测出这样的指令列。

（16）也可以是，所述变换装置包括：

逆变换装置，用于在所述第二判断装置判断出不包含变换对象指令列的情况下，将包含所述条件指令的指令列逆变换为不包含所述条件指令的指令列。

由此，作为不能用专用处理器译码的条件指令，即使不能变换成能用专用处理器译码的条件指令，也能够恢复成原来的指令列。

从而，能够生成专用处理器可以执行的指令列。

（17）为了达到上述目的，依据本发明的指令变换装置包括：

其中，各条件标记设定指令是分别包含一个条件，在专用处理器中判断各自的条件是否成立，并将该判断结果保持到同一（一个）条件标记中；

其中，各条件成立时执行指令是分别包含操作代码，并且只有在所述条件标记成立时，才在专用处理器中执行各自的操作代码所表示的操作的指令；

输入装置，用于输入不包含条件标记设定指令和条件成立时执行指令的指令列；

指令列检测装置，用于从所述输入装置输入的指令列中检测出根据预定（一个）条件是否成立而向同一存储对象传送不同传送对象的变换对象指令列；

变换装置，用于将所述变换对象指令列变换为包含条件标记设定指令和条件成立时执行指令的指令列，所述条件标记设定指令包含所述预定条件，所述条件成立时执行指令包含表示在所述预定条件成立时将应传送的传送对象传送到所述存储对象中的操作的操作代码。

依据本发明而如此构成的指令变换装置生成能够用专用处理器译码的条件标记设定指令和条件成立时执行的指令。

由此，由于一边变换成原有数量的条件指令，一边能够削减其种类，所以能够减少用于指令分配的字段，削减代码长度，并能够削减可以用专用处理器译码的条件指令的种类而不降低功能。

这样，能够削减专用处理器中的指令译码器的硬件数量，并在指令种类有限的情况下，能够具备更多种操作的条件指令，所以能够在进行流水线处理的处理器中更多地削减分支危险。

例如，在具备20个操作的条件成立时执行的指令时，可以将10种条件标记设定指令和20种条件成立时执行的指令（各条件成立时执行的指令1种*20个操作），合计30种指令分配到指令集中。

（18）也可以是，允许所述变换装置进行变换的各条件标记设定指令的条件与允许所述变换装置进行变换的另一个条件标记设定指令的条件具有排他关系。

由此，由于在生成条件不成立时执行的指令的情况下，依据本发明的指令变换装置只要生成用于指定具有排他关系的条件的条件标记设定指令和条件成立时执行的指令就足够了，所以不必将条件不成立时执行的指令分配到指令集中。

从而，能够削除条件不成立时执行的指令而不降低功能。

（19）也可以是，所述各条件标记设定指令分别指定两个运算对象甲和乙，

所述条件标记设定指令有6种：以以甲等于乙、甲不等于乙、甲大于等于乙、甲小于乙、甲小于等于乙、甲大于乙为条件。

由此，由于变换装置生成6种条件标记设定指令，并且这些条件中每两个各自具有排他关系，所以即使不将条件不成立时执行的指令分配到指令集中，依据本发明的指令变换装置也能执行与现有技术相同的处理。

从而，能够削除条件不成立时执行的指令而不降低功能。

（20）也可以是，所述变换对象指令列顺序由比较两个运算对象的比较指令、在预定条件成立时分支到后3个指令的条件分支指令、将预定传送对象传送到预定存储对象中的传送指令、跳转到后两个指令的无条件分支指令和将不同于所述预定传送对象的传送对象传送到所述预定存储对象中的传送指令构成。

由此，指令列检测装置能够检测出这样的指令列。

（21）也可以是，所述变换对象指令列顺序由比较两个运算对象的比较指令、将预定传送对象传送到预定存储对象中的传送指令、和在预定条件成立时将不同于所述预定传送对象的传送对象传送到所述预定存储对象中的条件指令构成。

由此，指令列检测装置能够检测出这样的指令列。

（22）也可以是，所述变换对象指令列顺序由比较两个运算对象的比较指

令、只有在预定条件不成立时才将预定传送对象传送到预定存储对象中的条件指令、和只有在预定条件成立时才将不同于所述预定传送对象的传送对象传送到所述预定存储对象中的条件指令构成。

由此，指令列检测装置能够检测出这样的指令列。

（23）为了达到上述目的，依据本发明的记录媒体包括：

其中，条件指令是包含条件和操作代码并且只有在此条件成立时才由处理器执行此操作代码所表示的操作的指令；

输入步骤，用于输入不包含条件指令的指令列；

指令列检测步骤，用于从所述输入装置输入的指令列中检测出根据预定（一个）条件是否成立而向同一存储对象分别传送不同传送对象的变换对象指令列；

判断步骤，用于判断包含与所述变换对象指令列所表示的预定条件相同的条件的条件指令是否被分配到专用处理器的指令集中；

变换步骤，

当所述判断步骤的判断结果为已被分配时，将所述变换对象指令列变换为包含所述预定条件的条件指令的指令列，

变换步骤，

当所述判断步骤的判断结果为已被分配时，将所述变换对象指令列变换为包含所述预定条件的条件指令的指令列，

当所述判断步骤判断结果为未被分配时，将所述变换对象指令列中所述不同传送对象进行相互替换，变换为包含与所述预定条件具有排他关系的条件的条件指令的指令列。

由此，能够得到与（8）相同的效果。

（24）为了达到上述目的，依据本发明的记录媒体包括：

其中，条件指令是包含条件和操作代码并且只有在此条件成立时才由处理器执行此操作代码所表示的操作的指令；

输入步骤，用于输入包含条件指令的指令列；

条件指令检测步骤，用于检测出包含在所述输入步骤输入的指令列中的条件指令；

第一判断步骤，用于判断所述条件指令检测步骤所检测出的条件指令是否

被分配到专用处理器的指令集中；

第二判断步骤，用于在所述第一判断步骤判断出未分配的情况下，根据所述条件指令检测步骤检测出的条件指令所包含的预定条件是否成立，判断所述输入步骤输入的指令列是否包含向同一存储对象传送不同传送对象的变换对象指令列；

变换步骤，用于在所述第二判断步骤判断出包含变换对象指令列的情况下，对该变换对象指令列中的所述不同传送对象进行相互替换，并将所述条件指令变换为包含与所述预定条件具有排他关系的条件的条件指令。

由此，能够得到与（13）相同的效果。

（25）为了达到上述目的，依据本发明的记录媒体包括：

其中，各条件标记设定指令是分别包含一个条件，在专用处理器中判断各自的条件是否成立，并将该判断结果保持到同一（一个）条件标记中；

其中，各条件成立时执行指令是分别包含操作代码，并且只有在所述条件标记成立时，才在专用处理器中执行各自的操作代码所表示的操作的指令；

输入步骤，用于输入不包含条件标记设定指令和条件成立时执行指令的指令列；

指令列检测步骤，用于从所述输入步骤输入的指令列中检测出根据预定（一个）条件是否成立而向同一存储对象传送不同传送对象的变换对象指令列；

变换步骤，用于将所述变换对象指令列变换为包含条件标记设定指令和条件成立时执行指令的指令列，所述条件标记设定指令包含所述预定条件，所述条件成立时执行指令包含表示在所述预定条件成立时将应传送的传送对象传送到所述存储对象中的操作的操作代码。

由此，能够得到与（17）相同的效果。

如上所述，本发明的技术具有很大的实用价值。

附图说明

图1是执行本发明的流水线所生成的机器指令的处理器构成示意图。

图2示出了实施例1的处理器能够执行的条件传送指令。

图3是描述实施例1中的编译器操作的流程图。

图4（a）、（b）是用C语言描述的C源程序的示例。

图5（a）、（b）示出了不包含由图4（a）、（b）所示的C源程序生成的条件传

送指令的中间代码串。

图6（a）、（b）示出了包含由图5（a）、（b）所示的不包含条件传送指令的中间代码串生成的条件传送指令的中间代码串。

图7（a）、（b）示出了分别由图6（a）、（b）所示的包含条件传送指令的中间代码串生成的机器指令列。

图8是描述实施例2中的编译器操作的流程图。

图9示出了包含了由图4（b）所示的C源程序生成的条件传送指令的中间代码串。

图10示出了实施例3的处理器能够执行的条件分支指令。

图11是描述实施例3中的编译器操作的流程图。

图12示出了由不能用图5（b）所述的实施例3的处理器译码的中间代码串生成的能够用实施例3的处理器译码的中间代码串。

图13示出了由能够用图12中所示的实施例3的处理器译码的中间代码串生成的机器语言指令列。

图14示出了削减了指令种类的实施例4的比较指令、条件相加指令、条件传送指令和条件分支指令。

图15是实施例4的编译器的构成示意图。

图16示出了输入到此编译器中的用C语言描述的C源代码1511。

图17示出了第一中间代码1512。

图18示出了第二中间代码1513。

图19示出了机器语言指令1515。

图20是描述实施例4中的对象代码检测装置1502和条件指令变换装置1503的操作的流程图。

图21示出了第一中间代码1512。

图22是描述实施例5中的对象代码检测装置1502和条件指令变换装置1503的操作的流程图。

图23示出了第一中间代码1512。

图24是描述实施例6中的对象代码检测装置1502和条件指令变换装置1503的操作的流程图。

图25示出了实施例1的处理器能够执行的条件相加指令。

图26示出了用C语言描述的C源程序。

图27示出了不包含由图26所示的C源程序生成的条件相加指令的中间代码串。

图28示出了由不包含图27所示的条件相加指令的中间代码串生成的包含了条件相加指令的中间代码串。

图29示出了分别由包含图28所示的条件相加指令的中间代码串生成的机器语言指令列。

图30示出了现有技术中的包含条件传送指令的指令列。

图31示出了现有技术中的全部条件传送指令。

图32示出了现有技术中的比较指令（CMP指令）、条件成立时执行相加指令的条件相加指令、条件成立时执行传送指令的条件传送指令和条件成立时进行分支的条件分支指令。

图33示出了削减了指令种类的比较指令、条件相加指令、条件传送指令和条件分支指令。

具体实施方式

图1是执行本发明的流水线所生成的机器指令的处理器构成示意图。

处理器包括运算标记保存装置101、指令译码装置102、指令执行装置103和指令执行停止装置104。

运算标记保存装置101用于保存表示指令执行状态的运算标记并输出表示指令执行状态的运算标记状态信号111。

指令译码装置102用于对机器指令译码并输出指令执行装置控制信号112。而且，在对条件指令译码时，输出条件指令指示信号113。

指令执行装置103根据从指令译码装置102输出的指令执行装置控制信号来执行指令。

指令执行停止装置104输入运算标记保存装置101输出的运算标记状态信号111和消除指令译码装置102输出的条件指令指示信号113，在条件不成立时，进行控制以便于向指令执行装置103输出指令执行停止信号113并停止执行指令。

处理器也可以将指令执行停止装置104替换为指令执行完成装置，指令执行完成装置在条件成立时，进行控制以便于向指令执行装置103输出指令完成信号，并执行指令。包括指令执行停止装置104的处理器和包括指令执行完成装

置的处理器只是通过逻辑反相来执行实质相同的操作，所以说这二者是相同的。

（实施例1）

对于图31所示的6种条件传送指令3101中具有排他关系的1对条件，实施例1的处理器只译码和执行指定了其中一种条件的条件指令而不译码和执行指定了另一种条件的条件指令。

具体地说，在通过比较指令比较两个运算对象甲和运算对象乙时，实施例1的处理器只译码和执行指定了“甲等于乙”和“甲不等于乙”中的任一个条件的条件指令。

同样地，实施例1的处理器只译码和执行指定了“甲大于乙”和“甲小于等于乙”中的任一个条件的条件指令，只译码和执行指定了“甲小于乙”和“甲大于等于乙”中的任一个条件的条件指令。

图2示出了实施例1的处理器能够执行的条件传送指令。这个条件传送指令201有三种。条件202是表示各个条件传送指令所指定的条件的记号。在通过比较指令比较两个运算对象甲和运算对象乙时，moveq203表示“甲等于乙”，movgt204表示“甲大于乙”，movge205表示“甲大于等于乙”，并且各条件传送指令在各自的条件成立时执行传送指令。

在图31中，实施例1中的编译器没有生成图2中未示出的条件传送指令，而只生成图2中所示的条件传送指令。

图3是描述实施例1中的编译器操作的流程图。

图4（a）、（b）示出了一个用C语言描述并被输入到这个编译器中的C源程序。

图4（a）的C源程序比较变量甲和变量乙，如果相等则c为1，否则c为0，并调用函数f。

图4（b）的C源程序比较变量甲和变量乙，如果不相等则c为1，否则c为0，并调用函数f。

图5（a）、（b）在各自的图3所示的流程图步骤S301中，分别示出了不包含分别由图4（a）和（b）生成的条件传送指令的中间代码串。图5（a）、（b）分别与现有的编译器在将图4（a）、（b）所示的C源程序变换为机器指令的过程中所生成的中间代码串相同。

在图5（a）中，中间代码501是将变量甲和变量乙的比较结果反映到标记中的代码，中间代码502是在变量甲等于变量乙时分支到标记507“Lt”的代码，中间代码503是向变量c传送立即数0的代码，中间代码504是分支到标记508“L”的代码，中间代码505是向变量c中传送立即数1的代码，中间代码506是调用函数f的代码。

在图5（b）中，中间代码511是将变量甲和变量乙的比较结果反映到标记中的代码，中间代码512是在变量甲不等于变量乙时分支到标记517“Lt”的代码，中间代码513是向变量c传送立即数0的代码，中间代码514是分支到标记518“L”的代码，中间代码515是向变量c中传送立即数1的代码，中间代码516是调用函数f的代码。

图6（a）、（b）分别示出了在图3所示的流程图中步骤S302~S318，实施例1的编译器从不包含图5（a）、（b）所示的条件传送指令的中间代码串中生成的包含条件传送指令的中间代码串。

在图6（a）中，中间代码601是将变量甲和变量乙的比较结果反映到标记中的代码，中间代码602是向变量c传送立即数0的代码，中间代码603是在变量甲等于变量乙时向变量c传送立即数1的代码，中间代码604是调用函数f的代码。

在图6（b）中，中间代码611是将变量甲和变量乙的比较结果反映到标记中的代码，中间代码612是向变量c中传送立即数1的代码，中间代码613是在变量甲不等于变量乙时向变量c传送立即数0的代码，中间代码614是调用函数f的代码。

图7（a）、（b）分别示出了在图3所示的流程图中步骤S319，实施例1的编译器从包含图6（a）、（b）所示的条件传送指令的中间代码串中生成的机器指令列。而且，图7（a）、（b）所示的机器指令列与现有的编译器从包含图6（a）、（b）所示的相同的条件传送指令的中间代码串中生成的机器指令列相同。

在图7（a）中，机器指令701是将寄存器r0和寄存器r1的比较结果反映到标记中的指令，机器指令702是向寄存器r2传送立即数0的指令，机器指令703是在寄存器r0等于寄存器r1时向寄存器r2传送立即数1的指令，机器指令704是调用函数f的指令。

在图7（b）中，机器指令711是将寄存器r0和寄存器r1的比较结果反映到标记中的指令，机器指令712是向寄存器r2传送立即数1的指令，机器指令713是在

寄存器r0等于寄存器r1时向寄存器r2传送立即数0的指令，机器指令714是调用函数f的指令。

下面，利用图3说明将图4（a）所示的C源程序输入到实施例1的编译器中时的处理。

（1a）将输入的C源程序变换为不包含条件传送指令的中间代码串并将初始值代入变量n（步骤S301）。这里，图4（a）所示的C源程序变换为图5（a）所示的中间代码串。

（2a）判断第n个中间代码是否为表示条件分支的中间代码，并且分支目标是否为第n+3个中间代码（步骤S302）。若判断为No，则n加1并反复判断，直到判断结果为Yes为止（步骤S302：No，步骤S314、步骤S318：Yes）。这里，由于图5（a）的中间代码502是表示条件分支的代码并且分支目标为第n+3个中间代码，所以在n=2时步骤S302的判断结果为Yes并进入步骤S303的判断。

（3a）判断第n+1个中间代码是否为表示传送的中间代码（步骤S303）。若为No，则n加1并找到表示条件分支的中间代码返回步骤S302（步骤S303：No，步骤S314、318：Yes）。这里，由于图5（a）中的中间代码503是表示传送的中间代码，所以在n+1=3时步骤S303的判断结果为Yes并进入步骤S304的判断。

（4a）判断第n+两个中间代码是否为表示无条件分支的中间代码，并且分支目标是否为第n+4个中间代码（步骤S304）。若为No，则n加2并找到表示条件分支的中间代码返回步骤S302（步骤S304：No，步骤S315、318：Yes）。这里，由于图5（a）中的中间代码504是表示无条件分支的中间代码并且分支目标为第n+4个中间代码，所以在n+2=4时步骤S304的判断结果为Yes并进入步骤S305的判断。

（5a）判断第n+3个中间代码是否为表示向与第n+1个中间代码相同的变量传送的中间代码（步骤S305）。若为No，则n加3并找到表示条件分支的中间代码返回步骤S302（步骤S305：No，步骤S316、318：Yes）。这里，由于图5（a）中的中间代码505是表示向与中间代码503相同的变量传送的中间代码，所以在n+3=5时步骤S305的判断结果为Yes并进入步骤S306的判断。

（6a）判断实施例1的处理器能否执行指定条件与表示第n个条件分支的中间代码所指定的条件相同的条件传送指令（步骤S306）。这里，表示图5（a）的中间代码502的条件分支的中间代码所指定的条件为“甲等于乙”，由于图2

所示的实施例 1的处理器能够执行指定此条件的条件传送指令，所以进入步骤S307的变换处理。

（7a）消除表示第n个条件分支的中间代码（步骤S307）。

（8a）消除表示第n+两个无条件分支的中间代码（步骤S308）。

（9a）将表示第n+3个传送的中间代码变更为指定条件与表示第n个条件分支的中间代码所指定的条件相同的条件传送指令（步骤S309）。

（10a）n加4并判断是否还存在应处理的中间代码（步骤S317、S318）。如果仍存在应处理的中间代码，就返回步骤S302~S318。这里，生成包含了图6（a）所示的条件传送指令的中间代码。

（11a）将包含条件传送指令的中间代码串变换为机器指令列（步骤S319）。这里，将图6（a）所示的中间代码串变换为图7（a）所示的机器指令列。

下面，说明将图4（b）所示的C源程序输入到实施例1的编译器中时的处理。

（1b）在步骤S301，将图4（b）所示的C源程序变换为图5（b）所示的中间代码串。

（2b）在步骤S302，由于图5（b）的中间代码512是表示条件分支的代码并且分支目标为第n+3个中间代码，所以在n=2时步骤S302的判断结果为Yes并进入步骤S303的判断。

（3b）在步骤S303，由于图5（b）中的中间代码513是表示传送的中间代码，所以在n+1=3时步骤S303的判断结果为Yes并进入步骤S304的判断。

（4b）在步骤S304，由于图5（b）中的中间代码514是表示无条件分支的中间代码并且分支目标为第n+4个中间代码，所以在n+2=4时步骤S304的判断结果为Yes并进入步骤S305的判断。

（5b）在步骤S305，由于图5（b）中的中间代码515是表示向与中间代码513相同的变量传送的中间代码，所以在n+3=5时步骤S305的判断结果为Yes并进入步骤S306的判断。

（6b）在步骤S306，表示图5（b）的中间代码512的条件分支的中间代码所指定的条件为“甲不等于乙”，由于图2所示的实施例 1的处理器不能够执行指定此条件的条件传送指令，所以判断结果为No并进入步骤S310的变换处理。

（7b）与（7a）相同（步骤S310）。

（8b）与（8a）相同（步骤S311）。

（9b）将表示第n+1个传送的中间代码变更为指定条件与表示第n个条件分支的中间代码所指定的条件具有排他关系的条件传送指令（步骤S312）。

（10b）对换第n+3个和第n+1个传送代码（步骤S313）。

（11b）与（10a）相同（步骤S317、S318）。如果仍存在应处理的中间代码，就返回步骤S302~S318。这里，生成包含了图6（b）所示的条件传送指令的中间代码。

（12b）在步骤S319，将图6（b ）所示的中间代码串变换为图7（b）所示的机器指令列。

如上所述，实施例1的编译器不会生成实施例1的处理器不能执行的条件传送指令，而只生成实施例1的处理器能够执行的条件传送指令。

（实施例2）

图8是描述实施例2中的编译器操作的流程图。

图9示出了实施例2的编译器在图8所示的流程图中的步骤S801，由图4（b）所示的C源程序生成的包含条件传送指令的中间代码串。图9与现有的生成条件传送指令的编译器在将图4（b）中所示的C源程序变换为机器指令的过程中所生成的中间代码相同。

在图9中，中间代码901是将变量甲和变量乙的比较结果反映到标记中的代码，中间代码902是向变量c传送立即数0的代码，中间代码903是在变量甲不等于变量乙时向变量c传送立即数1的代码，中间代码904是调用函数f的代码。

这里，实施例2的处理器能够执行的条件传送指令与实施例1的处理器相同，如图2所示。

下面，利用图8说明将图4（b）所示的C源程序输入到实施例2的编译器中时的处理。

在图8所示的各步骤中，用同于图3中各步骤的标号所表示的步骤执行相同的处理。

（1）将输入的C源程序变换为包含条件传送指令的中间代码串并将初始值1代入变量n（步骤S801）。这里，图4（b）所示的C源程序变换为图9所示的中间代码串。

（2）判断第n个中间代码是否为表示条件传送的中间代码（步骤S802）。若判断为No，则n加1并反复判断，直到判断结果为Yes为止（步骤S802：No，步

骤S807、步骤S808：Yes）。这里，由于图9的中间代码903是表示条件传送的中间代码，所以在n=3时步骤S802的判断结果为Yes并进入步骤S803的判断。

（3）判断实施例2的处理器能否执行第n个条件传送（步骤S803）。这里，表示图9的中间代码903的条件传送的中间代码所指定的条件为“甲不等于乙”，由于图2所示的实施例2的处理器不能够执行指定此条件的条件分支指令，所以判断结果为No并进入步骤S804的判断。

（4）判断第n-两个中间代码是否为表示比较的中间代码（步骤S804）。若判断结果为No，则进入步骤S810的逆变换处理，若为Yes，则进入步骤S805的判断。这里，由于图9的中间代码901为表示比较的中间代码，所以在n-2=1时步骤S804的判断结果为Yes并进入步骤S805的判断。

（5）判断第n-1个中间代码是否为表示向与表示第n个条件传送的中间代码相同的变量传送的中间代码（步骤S805）。若为No，则进入步骤S810的条件传送消除处理，若为Yes，则进入步骤S806的变换处理。这里，由于图9中的中间代码902是表示向与表示中间代码903的条件传送的中间代码相同的变量传送的中间代码，所以在n-1=3时步骤S805的判断结果为Yes并进入步骤S806的变换处理。

（6）将表示第n个条件传送的中间代码的条件变更为与之具有排他关系的条件，并交换第n-1个和第n个中间代码的传送值（步骤S806）。这里，将表示图9的中间代码903的条件传送的中间代码所指定的条件“甲不等于乙（≠）”变更为与之具有排他关系的条件“等于（=）”。即，将“c=：ne 1”变更为“c=：eq 1”。

（7）n加1并判断是否还存在应处理的中间代码（步骤S807、S808）。如果仍存在应处理的中间代码，就返回步骤S802~S808。这里，生成包含了图6（b）所示的条件传送指令的中间代码串。

（8）在步骤S319，将图6（b）所示的中间代码串变换为图7（b）所示的机器指令列。

（9）作为实施例2的处理器不能执行的条件传送指令并不能通过步骤S806的变换处理进行变换的中间代码经过逆变换处理，变为原来的中间代码（步骤S810）。这里，不执行此处理。

如上所述，实施例2 的编译器将实施例2的处理器不能执行的条件传送指令

变换为实施例2的处理器能够执行的条件传送指令。

（实施例3）

实施例3的处理器只译码和执行指定图31中所示的6种条件传送指令3101和表示相同条件的6种条件分支指令中具有排他关系的1对条件中的一个条件的条件分支指令，而不译码和不执行指定另一个条件的条件分支指令。

图10示出了实施例3的处理器能够执行的条件分支指令。这个条件分支指令1001有三种。条件1002是表示各个条件分支指令所指定的条件的记号。在通过比较指令比较两个运算对象甲和运算对象乙时，beq1003表示“甲等于乙”，bgt1004表示“甲大于乙”，bge1005表示“甲大于等于乙”，并且各条件分支指令在各自的条件成立时进行分支。这里，实施例3的处理器不能这些其他的条件指令。

图11是描述实施例3中的编译器操作的流程图。

图12示出了实施例3的编译器按照图11所示的流程图中的步骤S302~S318，从不能用图5（b）所述的实施例3的处理器译码的中间代码串中生成能够用实施例3的处理器译码的中间代码串。

在图12中，中间代码1201是将变量甲和变量乙的比较结果反映到标记中的代码，中间代码1202是在变量甲等于变量乙时分支到标记1207“Lt”的代码，中间代码1203是向变量c传送立即数1的代码，中间代码1204是分支到标记1208“L”的代码，中间代码1205是向变量c中传送立即数0的代码，中间代码1206是调用函数f的代码。

图13是实施例3的编译器在图11所示的流程图中的步骤S319，从能够用图12中所示的实施例3的处理器译码的中间代码串中生成机器语言指令列的示意图。而且，图13中生成的机器指令列与现有的编译器从与图12所示相同的中间代码串中生成的机器指令相同。

在图13中，机器指令1301是将寄存器r0和寄存器r1的比较结果反映到标记中的指令，机器指令1302是在变量甲等于变量乙时分支到标记1307“Lt”的代码，机器指令1303是向寄存器r2传送立即数1的指令，机器指令1304是分支到标记1308“L”的代码，机器指令1305是向寄存器r2传送立即数0的指令，机器指令1306是调用函数f的指令。下面，利用图11说明将图4（b）所示的C源程序输入到实施例3的编译器中时的处理。

在图11所示的各步骤中，用同于图3中各步骤的标号所表示的步骤执行相同的处理。

（1）～（5）同于实施例1的（1b）～（5b）。

（6）判断实施例3的处理器能否执行指定条件与表示第n个条件分支的中间代码所指定的条件相同的条件分支指令。若为Yes，则不进行变换处理，若为No，则进入变换处理（步骤S1101）。这里，表示图5（b）的中间代码512的条件分支的中间代码所指定的条件为“甲不等于乙”，由于图10所示的实施例 3的处理器不能执行指定此条件的条件分支指令，所以判断结果为No，并进入步骤S1102的变换处理。

（7）将表示第n个条件分支的中间代码所指定的条件变更为与之具有排他关系的条件（步骤S1102）。这里，将表示图5（b）的中间代码512的条件分支的中间代码所指定的条件“甲不等于乙（≠）”变更为与之具有排他关系的条件“等于（=）”。即，将“bne”变更为“beq”。

（8）对换第n+3个和第n+1个传送代码（步骤S1103）。这里，对换第5个传送代码“c=0”和第3个传送代码“c=1”。

（9）在步骤S317、318，生成图12所示的实施例3的处理器能够执行的中间代码串。

（10）在步骤S319，将图12所示的中间代码串变换为图13所示的机器指令列。

如上所述，实施例3的编译器将实施例3的处理器不能执行的条件分支指令变换为其能够执行的条件分支指令。

（实施例4）

图14示出了削减了指令种类的实施例4的比较指令、条件相加指令、条件传送指令和条件分支指令。

由于图14所示的各条件指令和条件分支指令的条件只是一个被设置的条件标记，所以各条件指令和条件分支指令各有一种，而用于设置/清零条件标记的比较指令有10种。

这样，比较指令、两2种条件指令和条件分支指令合计共13种。这里，在条件指令的操作数为A时，比较指令、各条件指令和条件分支指令合计共11+A种。

关于比较指令、条件相加指令、条件传送指令和条件分支指令的指令种类

合计，通过比较图14、图32及图33可知，在条件指令的操作数A为0~3时，图33的指令种类合计最少，但在条件指令的操作数A为4时，图14和图33的指令种类合计共15种（此时图32的指令种类合计为51种），条件指令的操作数为5种以上时，图14的指令种类合计最少，并且这种差异随着条件指令的操作数A的增加而增大。

实施例4的处理器译码和执行图14所示的比较指令、条件相加指令、条件传送指令和条件分支指令。

图15是实施例4的编译器的构成示意图。

这个编译器包括中间代码生成装置1501、对象代码检测装置1502、条件指令变换装置1503、中间代码最佳化装置1504及机器指令生成装置1505。

图16示出了输入到此编译器中的用C语言描述的C源代码1511。由于图16的C源程序与图4（a）的C源程序相同，故省略其说明。

中间代码生成装置1501变换C源代码1511并生成第一中间代码1512。由于此变换同于现有的编译器所进行的变换，故省略其说明。

图17示出了第一中间代码1512。由于图17的第一中间代码1512同于图5（a）的而省略其说明。

这里，从图16所示的C源代码1511中生成图17所示的第一中间代码1512。

对象代码检测装置1502检测出根据预定条件成立与否而决定是否执行预定操作的指令列。

条件指令变换装置1503将对象代码检测装置1502检测出的指令列变换为条件指令，并从第一中间代码1512中生成第二中间代码1513。后面将详细说明对象代码检测装置1502和条件指令变换装置1503。

图18示出了第二中间代码1513。

在图18中，中间代码1801“a cmpeq b”为条件比较代码，进行变量甲和变量乙的比较，如果比较结果相等就设置运算标记，如果不相等就对运算标记清零。中间代码1802“c=0”向变量c传送立即数0。中间代码1803“c=：true 1”为条件成立时执行代码，只在包含在比较代码中的条件成立（设置标记）时向变量c传送立即数1。中间代码1804“jsr f”调用“f”所指定的函数。

这里，从图17所示的第一中间代码1512中生成图18所示的第二中间代码1513。

中间代码最佳化装置1504使第二中间代码1513最佳化，并生成第三中间代码1514。由于这个最佳化过程同于现有的编译器所进行的最佳化而省略其说明。

这里，在图18所示的第二中间代码1513被输入到中间代码最佳化装置1504中时，由于没有应进行最佳化的代码而不进行任何操作。从而，第三中间代码1514与图18所示的第二中间代码1513相同。

机器指令生成装置1505变换第三中间代码1514并生成机器指令1515。由于此变换同于现有的编译器所进行的变换，故省略其说明。

图19示出了机器语言指令1515。

在图19中，机器指令1901“cmpeq r0，r1”为条件比较指令，比较寄存器r0和寄存器r1的内容，如果比较结果相等就设置运算标记，如果不相等就对运算标记清零。机器指令1902“movt 0，r2”为传送指令，用于向寄存器r2传送立即数0。机器指令1903“movt 1，r2”为条件成立时执行指令，只在比较指令设定的条件成立（设置运算标记）时向寄存器r2传送立即数1。中间代码1904“jsr f”调用“f”所指定的函数。这里，从图18所示的第三中间代码1514中生成图19所示的机器指令1515。

图20是描述实施例4中的对象代码检测装置1502和条件指令变换装置1503的操作的流程图。

下面，利用图17说明图17所示的第一中间代码1512被输入到实施例4的编译器的对象代码检测装置1502和条件指令变换装置1503中时的处理。

（4）将第一中间代码输入到对象代码检测装置1502和条件指令变换装置1503中，并将初始值1代入变量n（步骤S2001）。这里，输入图17所示的第一中间代码。

（2）判断第n个第一中间代码是否为表示比较的中间代码（步骤S2002）。若判断为No，则n加1并反复判断，直到判断结果为Yes为止（步骤S2002：No，步骤S2011、步骤S2016：Yes）。这里，由于图17的第1个第一中间代码是表示比较的中间代码，所以在n=1时步骤S2002的判断结果为Yes并进入步骤S2003的判断。

（3）判断第n+1个第一中间代码是否为表示条件分支的中间代码，并且分支目标是否为第n+4个中间代码（步骤S2003）。若判断为No，则n加1并找到表

示比较的第一中间代码返回步骤S2002（步骤S2003：No，步骤S2011、步骤S2016：Yes）。这里，由于图17的第两个第一中间代码是表示条件分支的中间代码并且分支目标为第n+4个中间代码，所以在n+1=2时步骤S2003的判断结果为Yes并进入步骤S2004的判断。

（4）判断第n+两个第一中间代码是否为表示传送的中间代码（步骤S2004）。若判断为No，则n加2并找到表示比较的第一中间代码返回步骤S2002（步骤S2004：No，步骤S2012、步骤S2016：Yes）。这里，由于图17的第3个第一中间代码是表示传送的中间代码，所以在n+2=3时步骤S2004的判断结果为Yes并进入步骤S2005的判断。

（5）判断第n+3个第一中间代码是否为表示无条件分支的中间代码，并且分支目标是否为第n+5个中间代码（步骤S2005）。若为No，则n加3并找到表示比较的第一中间代码返回步骤S2002（步骤S2005：No，步骤S2013、2016：Yes）。这里，由于图17中的第4个第一中间代码是表示无条件分支的中间代码并且分支目标为第n+5个中间代码，所以在n+3=4时步骤S2005的判断结果为Yes并进入步骤S2006的判断。

（6）判断第n+4个第一中间代码是否为表示向与第n+两个第一中间代码相同的变量传送的中间代码（步骤S2006）。若为No，则n加4并找到表示比较的第一中间代码返回步骤S2002（步骤S2006：No，步骤S2014、2016：Yes）。这里，由于图17中的第5个第一中间代码是表示向与第3个第一中间代码相同的变量传送的中间代码，所以在n+4=5时步骤S2006的判断结果为Yes并进入步骤S2007的变换处理。

（7）将表示第n个比较的中间代码变更为指定条件与表示第n+1个条件分支的中间代码的条件相同的条件比较代码（步骤S2007）。

（8）消除表示第n+1个条件分支的中间代码（步骤S2008）。

（9）消除表示第n+3个无条件分支的中间代码（步骤S2009）。

（10）将表示第n+4个传送的中间代码变更为条件成立时执行代码（步骤S2010）。

（11）n加5并判断是否还存在应处理的中间代码（步骤S2015、S2016）。如果仍存在应处理的中间代码，就返回步骤S2002~S2016。这里，生成图18所示的第二中间代码1513。

（12）条件指令变换装置1503输出第二中间代码（步骤S2017）。

如上所述，实施例4的编译器生成实施例4的处理器能够执行的条件比较代码和条件成立时执行代码。

（实施例5）

与实施例4相同，实施例5的处理器译码和执行图14所示的比较指令、条件相加指令、条件传送指令和条件分支指令。

与实施例4相同，实施例5的编译器的构成如图15所示。

在实施例5中只说明不同于实施例4之处。

图21示出了第一中间代码1512。由于图21的第一中间代码1512与图6（a）中的中间代码相同，故省略其说明。

中间代码生成装置1501从图16中所示的C源代码1511中生成图21所示的第一中间代码1512。由于此变换同于现有的用于生成条件传送指令的编译器所进行的变换，故省略其说明。

对象代码检测装置1502和条件指令变换装置1503从图21所示的第一中间代码1512中生成图18所示的第二中间代码1513。

图22是描述实施例5中的对象代码检测装置1502和条件指令变换装置1503的操作的流程图。

下面，利用图22说明图21所示的第一中间代码1512被输入到实施例5的编译器的对象代码检测装置1502和条件指令变换装置1503中时的处理。

（1）将第一中间代码输入到对象代码检测装置1502和条件指令变换装置1503中，并将初始值1代入变量n（步骤S2201）。这里，输入图21所示的第一中间代码。

（2）判断第n个第一中间代码是否为表示比较的中间代码（步骤S2202）。若判断为No，则n加1并反复判断，直到判断结果为Yes为止（步骤S2202：No，步骤S2207、步骤S2210：Yes）。这里，由于图21的第1个第一中间代码是表示比较的中间代码，所以在n=1时步骤S2202的判断结果为Yes并进入步骤S2203的判断。

（3）判断第n+1个第一中间代码是否为表示传送的中间代码（步骤S2203）。若判断为No，则n加1并找到表示比较的第一中间代码返回步骤S2202（步骤S2203：No，步骤S2207、步骤S2210：Yes）。这里，由于图21的第两个第一中

间代码是表示传送的中间代码，所以在n+1=2时步骤S2203的判断结果为Yes并进入步骤S2204的判断。

（4）判断第n+两个第一中间代码是否为表示向与第n+1个第一中间代码相同的变量进行条件传送的中间代码（步骤S2204）。若为No，则n加2并找到表示比较的第一中间代码返回步骤S2202（步骤S2204：No，步骤S2208、2210：Yes）。这里，由于图21中的第3个第一中间代码是表示向与第两个第一中间代码相同的变量进行条件传送的中间代码，所以在n+2=3时步骤S2204的判断结果为Yes并进入步骤S2205的变换处理。

（5）将表示第n个比较的中间代码变更为指定条件与表示第n+两个条件传送的中间代码的条件具有排他关系的条件比较代码（步骤S2205）。

（6）将表示第n+两个条件传送的中间代码变更为条件成立时执行代码（步骤S2206）。

（7）n加3并判断是否还存在应处理的中间代码（步骤S2209、S2210）。如果仍存在应处理的中间代码，就返回步骤S2202~S2210。这里，生成图18所示的第二中间代码1513。

（8）条件指令变换装置1503输出第二中间代码（步骤S2211）。

如上所述，实施例5的编译器生成实施例5的处理器能够执行的条件比较代码和条件成立时执行代码。

（实施例6）

与实施例4、5相同，实施例6的处理器译码和执行图14所示的比较指令、条件相加指令、条件传送指令和条件分支指令。

与实施例4、5相同，实施例6的编译器的构成如图15所示。

在实施例6中只说明不同于实施例4之处。

图23示出了第一中间代码1512。

在图23中，中间代码2301是将变量甲和变量乙的比较结果反映到标记中的代码，中间代码2302是在变量甲不等于变量乙时向变量c传送立即数0的代码，中间代码2303是在变量甲等于变量乙时向变量c传送立即数1的代码，中间代码2304是调用函数f的代码。

中间代码生成装置1501从图16中所示的C源代码1511中生成图23所示的第一中间代码1512。由于此变换同于现有的用于生成条件传送指令的编译器所进行

的变换，故省略其说明。

对象代码检测装置1502和条件指令变换装置1503从图23所示的第一中间代码1512中生成图18所示的第二中间代码1513。

图24是描述实施例6中的对象代码检测装置1502和条件指令变换装置1503的操作的流程图。

下面，利用图24说明图23所示的第一中间代码1512被输入到实施例6的编译器的对象代码检测装置1502和条件指令变换装置1503中时的处理。

（1）将第一中间代码输入到对象代码检测装置1502和条件指令变换装置1503中，并将初始值1代入变量n（步骤S2401）。这里，输入图23所示的第一中间代码。

（2）判断第n个第一中间代码是否为表示比较的中间代码（步骤S2402）。若判断为No，则n加1并反复判断，直到判断结果为Yes为止（步骤S2402：No，步骤S2409、步骤S2412：Yes）。这里，由于图23的第1个第一中间代码是表示比较的中间代码，所以在n=1时步骤S2402的判断结果为Yes并进入步骤S2403的判断。

（3）判断第n+1个第一中间代码是否为表示条件传送的中间代码（步骤S2403）。若判断为No，则n加1并找到表示比较的第一中间代码返回步骤S2402（步骤S2403：No，步骤S2409、步骤S2412：Yes）。这里，由于图23的第两个第一中间代码是表示传送的中间代码，所以在n+1=2时步骤S2403的判断结果为Yes并进入步骤S2404的判断。

（4）判断第n+两个第一中间代码是否为表示向与第n+1个第一中间代码相同的变量进行条件传送的中间代码（步骤S2404）。若为No，则n加2并找到表示比较的第一中间代码返回步骤S2402（步骤S2404：No，步骤S2410、2412：Yes）。这里，由于图23中的第3个第一中间代码是表示向与第两个第一中间代码相同的变量进行条件传送的中间代码，所以在n+2=3时步骤S2404的判断结果为Yes并进入步骤S2405的变换处理。

（5）判断表示第n+两个条件传送的中间代码的条件与表示第n+1个条件传送的中间代码的条件是否具有排他关系（步骤S2405）。若为No，则n加2并找到表示比较的第一中间代码返回步骤S2402（步骤S2405：No，步骤S2410、2412：Yes）。这里，由于图23中表示第3个条件传送的第一中间代码的条件与表示第两

个条件传送的第一中间代码的条件具有排他关系，所以在n+2=3时步骤S2405的判断结果为Yes并进入步骤S2406的变换处理。

（6）将表示第n个比较的中间代码变更为指定条件与表示第n+两个条件传送的中间代码的条件相同的条件比较代码（步骤S2406）。

（7）将表示第n+1个条件传送的中间代码变更为传送代码（步骤S2407）。

（8）将表示第n+两个条件传送的中间代码变更为条件成立时执行代码（步骤S2408）。

（9）n加3并判断是否还存在应处理的中间代码（步骤S2411、S2412）。如果仍存在应处理的中间代码，就返回步骤S2402~S2412。这里，生成图18所示的第二中间代码1513。

如上所述，实施例6的编译器生成实施例6的处理器能够执行的条件比较代码和条件成立时执行代码。

以上，在各实施例中对本发明相关的处理器及编译器进行了说明，但并不意味着本发明仅限于这些实施例。即，

（1）本发明的编译器所进行的用于将不包含条件传送代码的中间代码或包含本发明相关的处理器不能执行的条件传送代码的中间代码变换为包含本发明相关的处理器能够执行的条件传送代码的中间代码的处理也可以是用于将不包含条件传送指令的机器指令列或包含本发明相关的处理器不能执行的条件传送指令的机器指令列变换为包含本发明相关的处理器能够执行的条件传送指令的机器指令列的处理。

（2）在各实施例中利用传送指令传送立即数，但传送寄存器和存储器的内容等也是可以的。

（3）在各实施例中主要从传送指令中生成条件指令和条件成立时执行指令，但并不仅限于传送指令，算术运算指令和逻辑运算指令也是可以的。例如，可以设想将实施例1适用于加法运算指令并生成包含条件相加指令的指令列的情况。

图25示出了实施例1的处理器能够执行的条件相加指令。这个条件相加指令2501有三种。条件2502是表示各个条件相加指令所指定的条件的记号。在通过比较指令比较两个运算对象甲和运算对象乙时，addeq2503表示“甲等于乙”，addgt2504表示“甲大于乙”，addge2505表示“甲大于等于乙”，并且各条件相加指令在各自的条件成立时执行相加指令。

图26示出了用C语言描述的C源程序。

图27示出了不包含由图26所示的C源程序生成的条件相加指令的中间代码串。

图28示出了由不包含图27所示的条件相加指令的中间代码串生成的包含了条件相加指令的中间代码串。

图29示出了分别由包含图28所示的条件相加指令的中间代码串生成的机器语言指令列。

通过将实施例1适用于条件相加指令中的编译器，图26所示的程序被变换为图27所示的中间代码，然后变换为图28所示的中间代码，然后生成图29所示的包含条件相加指令的指令列。

（4）在实施例4~6中，各条件指令和条件分支指令的条件为“条件标记被设置”，但“条件标记被清零”也是可以的。

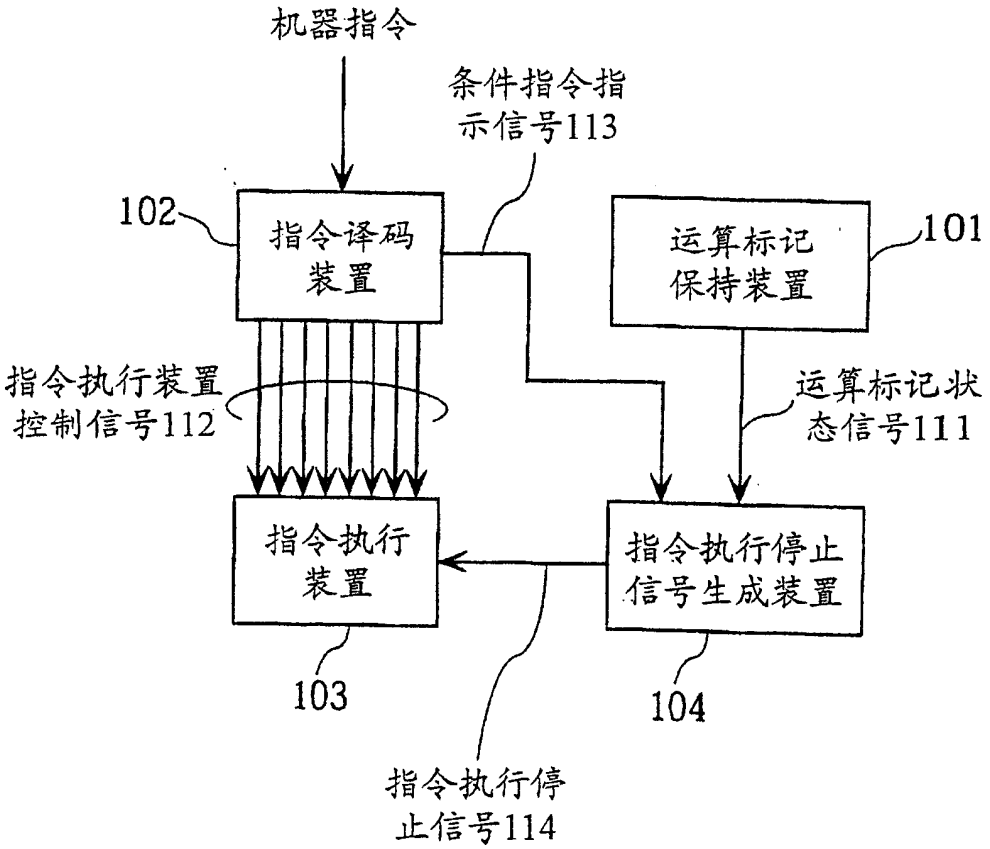

机器指令
条件指令指示信号113
102
指令译码装置
运算标记保持装置
101
指令执行装置控制信号112
运算标记状态信号111
指令执行装置
指令执行停止信号生成装置
103
104
指令执行停止信号114

图 1

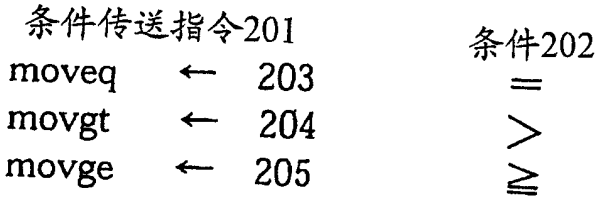


图 2

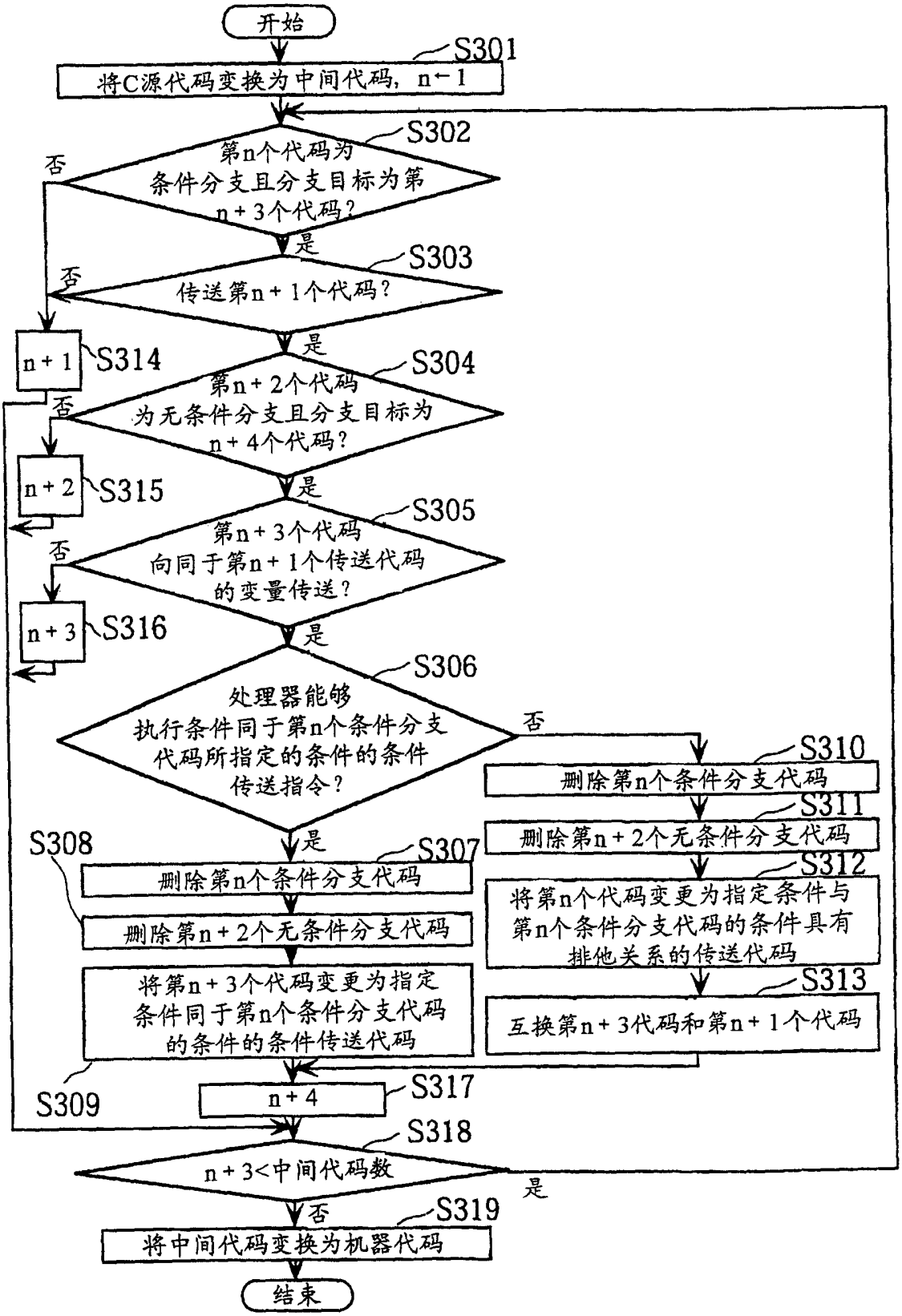

开始
S301
将C源代码变换为中间代码，n←1
S302
第n个代码为条件分支且分支目标为第n+3个代码？
否
是
S303
传送第n+1个代码？
否
是
n+1
S314
S304
第n+2个代码为无条件分支且分支目标为n+4个代码？
否
是
n+2
S315
S305
第n+3个代码向同于第n+1个传送代码的变量传送？
否
是
n+3
S316
S306
处理器能够执行条件同于第n个条件分支代码所指定的条件的条件传送指令？
否
是
S310
删除第n个条件分支代码
S311
删除第n+2个无条件分支代码
S312
将第n个代码变更为指定条件与第n个条件分支代码的条件具有排他关系的传送代码
S313
互换第n+3代码和第n+1个代码
S308
S307
删除第n个条件分支代码
删除第n+2个无条件分支代码
将第n+3个代码变更为指定条件同于第n个条件分支代码的条件的条件传送代码
S309
n+4
S317
S318
n+3<中间代码数
是
否
S319
将中间代码变换为机器代码
结束

图 3

图 4(a)

```
if(a == b)
{
        c = 1;
}
else
{
       c = 0
}
f();
```

图 4(b)

```
if(a != b)
{
       c = 1;
}
else
{
      c = 0
}
f();
```

图 5(a)

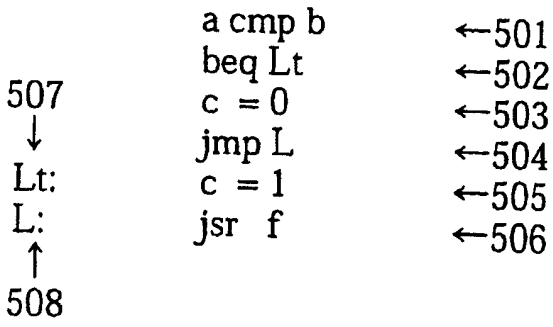

a cmp b
beq Lt
c = 0
jmp L
c = 1
jsr f
←501
←502
←503
←504
←505
←506
507
↓
Lt:
L:
↑
508

图 5(b)

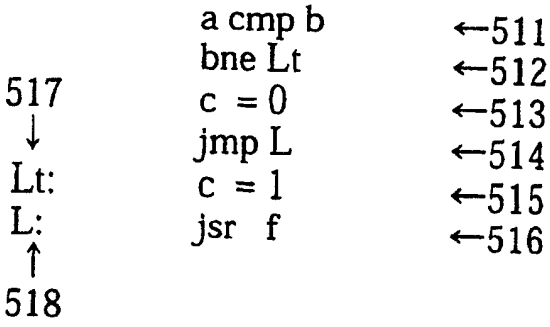

a cmp b
bne Lt
c = 0
jmp L
c = 1
jsr f
←511
←512
←513
←514
←515
←516
517
↓
Lt:
L:
↑
518

图 6 (a)

```
a cmp b      ←601
c = 0        ←602
c = :eq 1    ←603
jsr  f       ←604
```

图 6 (b)

```
a cmp b      ←611
c = 1        ←612
c = :eq 0    ←613
jsr  f       ←614
```

图 7 (a)

```
cmp     r0,r1    ←701
mov     0,r2     ←702
moveq   1,r2     ←703
jsr     f        ←704
```

图 7 (b)

```
cmp     r0,r1    ←711
mov     1,r2     ←712
moveq   0,r2     ←713
jsr     f        ←714
```

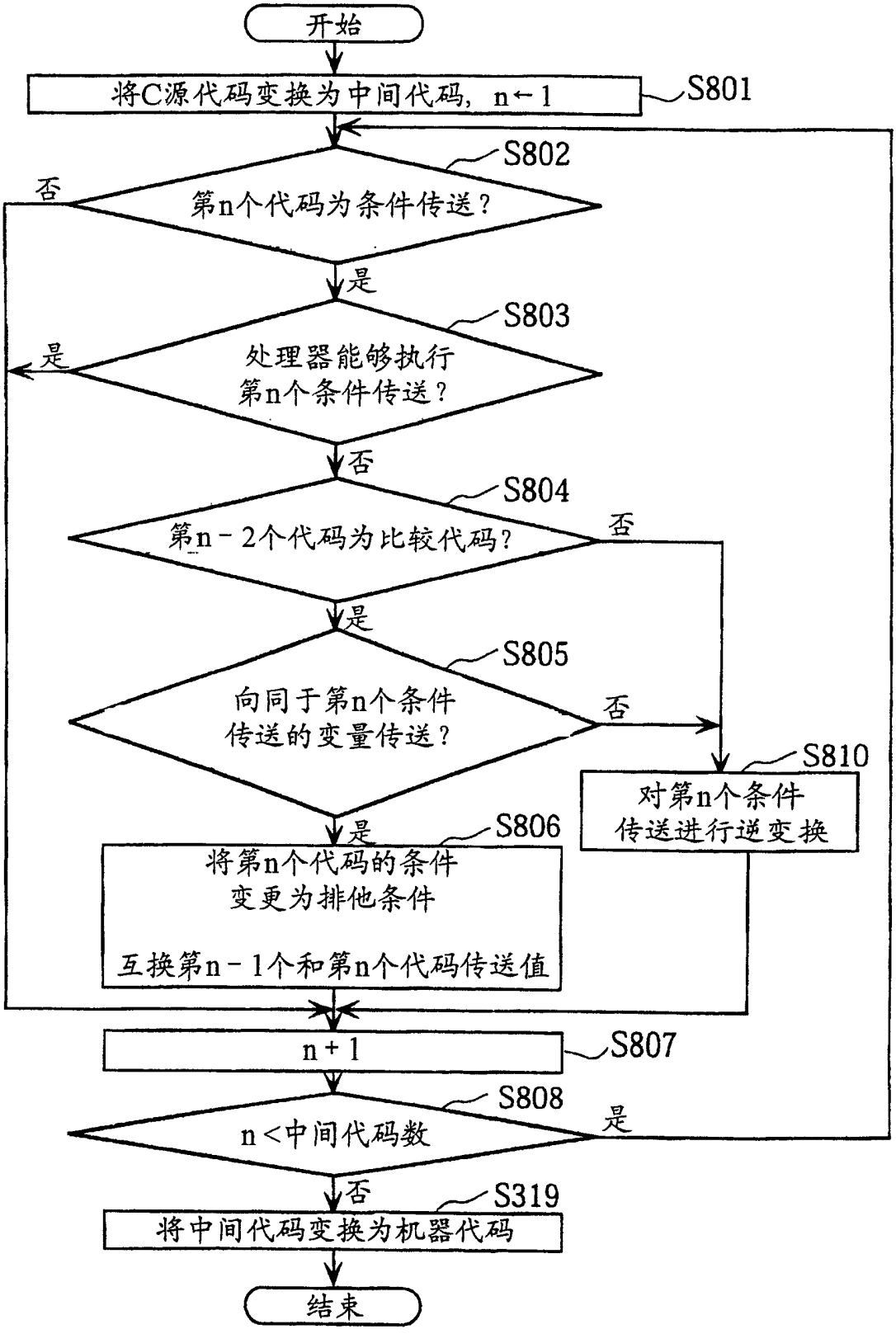

开始
将C源代码变换为中间代码，n←1
S801
S802
第n个代码为条件传送？
否
是
S803
处理器能够执行
第n个条件传送？
是
否
S804
第n－2个代码为比较代码？
否
是
S805
向同于第n个条件
传送的变量传送？
否
是
S806
将第n个代码的条件
变更为排他条件
互换第n－1个和第n个代码传送值
S810
对第n个条件
传送进行逆变换
n＋1
S807
S808
n＜中间代码数
是
否
S319
将中间代码变换为机器代码
结束

图 8

```
a cmp b          ←901
c = 0            ←902
c = :ne 1        ←903
jsr  f           ←904
```

图 9

条件分支指令1001		条件1002
beq	←1003	=
bgt	←1004	>
bge	←1005	≧

图 10

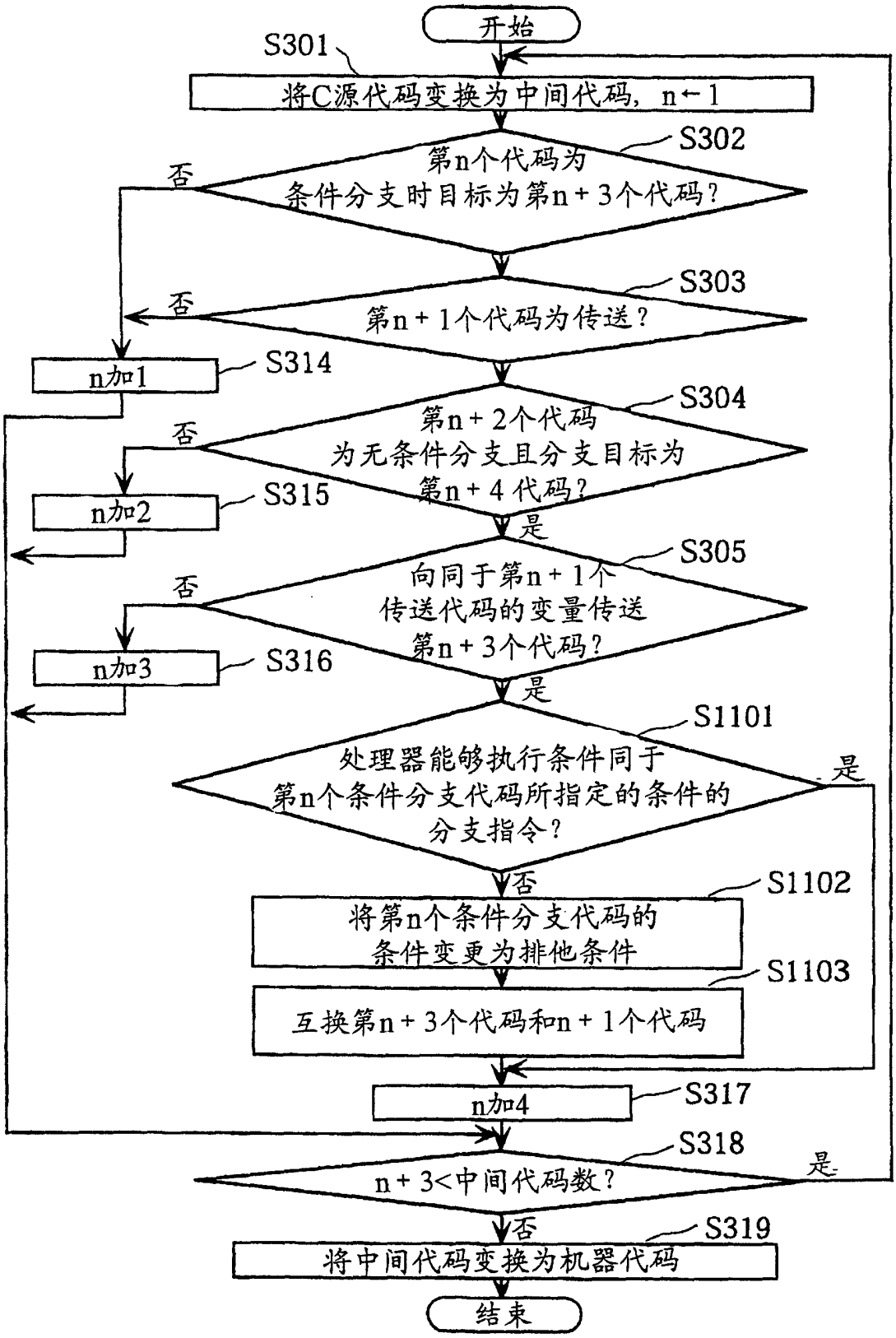

开始
S301
将C源代码变换为中间代码，n←1
S302
第n个代码为
条件分支时目标为第n+3个代码？
否
S303
第n+1个代码为传送？
否
n加1
S314
S304
第n+2个代码
为无条件分支且分支目标为
第n+4代码？
否
n加2
S315
是
S305
向同于第n+1个
传送代码的变量传送
第n+3个代码？
否
n加3
S316
是
S1101
处理器能够执行条件同于
第n个条件分支代码所指定的条件的
分支指令？
是
否
S1102
将第n个条件分支代码的
条件变更为排他条件
S1103
互换第n+3个代码和n+1个代码
n加4
S317
S318
n+3<中间代码数？
是
否
S319
将中间代码变换为机器代码
结束

图 11

```
                a cmp b      ←1201
                beq Lt       ←1202
1207            c = 1        ←1203
 ↓              jmp L        ←1204
Lt:             c = 0        ←1205
L:              jsr  f       ←1206
 ↑
1208
```

图 12

```
                cmp     r0,r1    ←1301
                beq     Lt       ←1302
1307            mov     1,r2     ←1303
 ↓              jmp     L        ←1304
Lt:             mov     0,r2     ←1305
L:              jsr     f        ←1306
 ↑
1308
```

图 13

指令缩写	条件指定	操作
[比较]		
cmpeq Rm,Rn	=	CMP指令结果为相等时设置条件标记，其它时候对条件标记清零
cmpne Rm,Rn	≠	CMP指令结果为不相等时设置条件标记，其它时候对条件标记清零
cmpge Rm,Rn	≧	CMP指令结果为符号数据并为大于等于时设置条件标记，其它时候对条件标记清零
cmple Rm,Rn	≦	CMP指令结果为符号数据并为小于等于时设置条件标记，其它时候对条件标记清零
cmpgt Rm,Rn	>	CMP指令结果为符合数据并为大于时设置条件标记，其它时候对条件标记清零
cmplt Rm,Rn	<	CMP指令结果为符号数据并为小于时设置条件标记，其它时候对条件标记清零
cmpns Rm,Rn	≧	CMP指令结果为无符号数据并为大于等于时设置条件标记，其它时候对条件标记清零
cmpls Rm,Rn	≦	CMP指令结果为无符号数据并为小于等于时设置条件标记，其它时候对条件标记清零
cmphi Rm,Rn	>	CMP指令结果为无符号数据并为大于时设置条件标记，其它时候对条件标记清零
cmplo Rm,Rn	<	CMP指令结果为无符号数据并为小于时设置条件标记，其它时候对条件标记清零
bt label	–	设置条件标记时进行分支
movt Rm,Rn	–	设置条件标记时将Rm传送到Rn中
addt Rm,Rn,Rd	–	设置条件标记时将Rm与Rm相加，并将结果存入Rd中

图 14

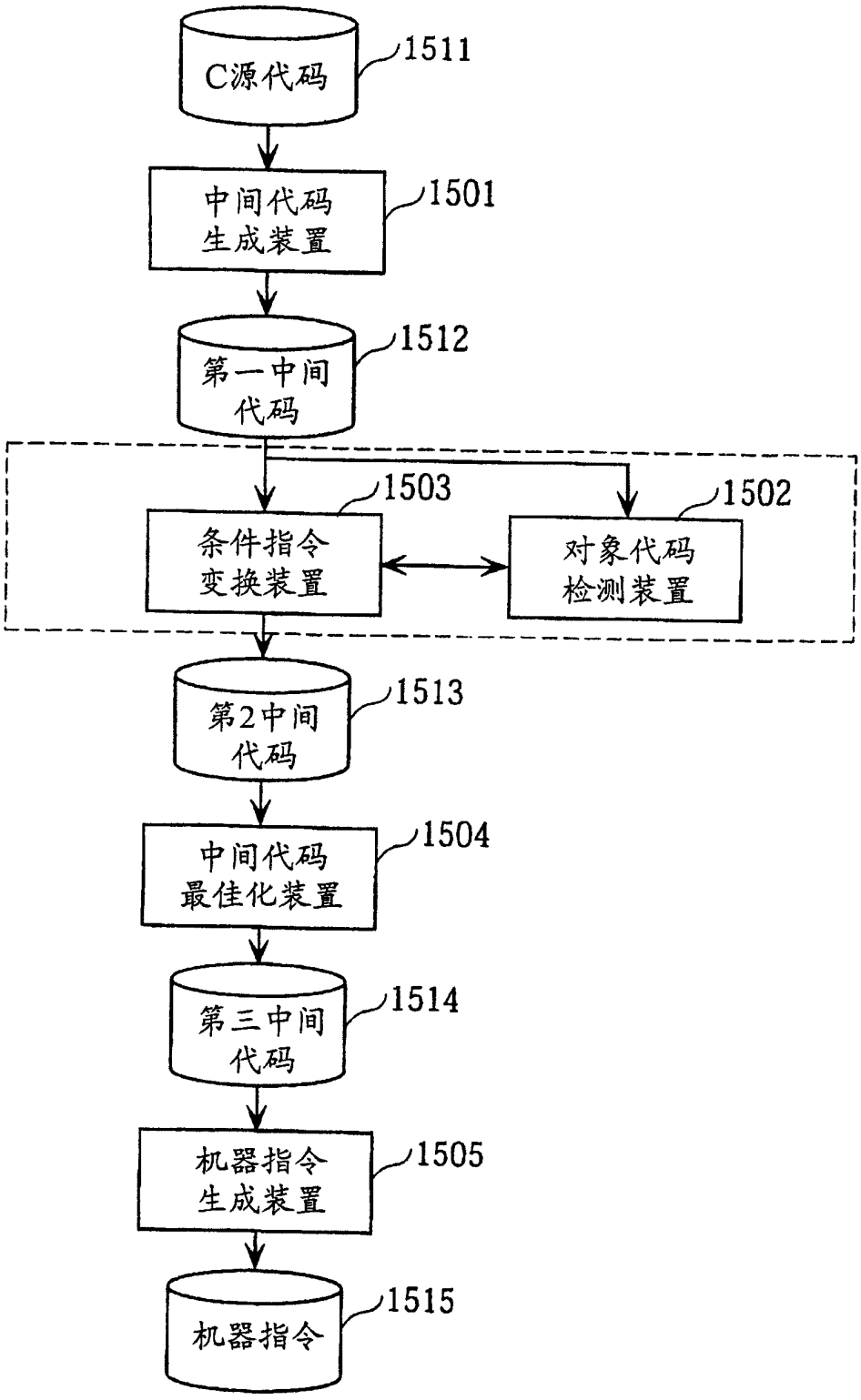

C源代码
1511
中间代码
生成装置
1501
第一中间
代码
1512
1503
条件指令
变换装置
1502
对象代码
检测装置
第2中间
代码
1513
中间代码
最佳化装置
1504
第三中间
代码
1514
机器指令
生成装置
1505
机器指令
1515

图 15

图 16

```
if(a == b)
{
        c = 1;
}
else
{
        c = 0;
}
f();
```

图 17

```
        a cmp b
        beq Lt
        c=0
        jmp  L
Lt:     c=1
L:      jsr  f
```

图 18

```
a cmpeq b       ←1801
c=0             ←1802
c=:true 1       ←1803
jsr  f          ←1804
```

图 19

```
cmpeq   r0,r1   ←1901
mov     0,r2    ←1902
movt    1,r2    ←1903
jsr     f       ←1904
```

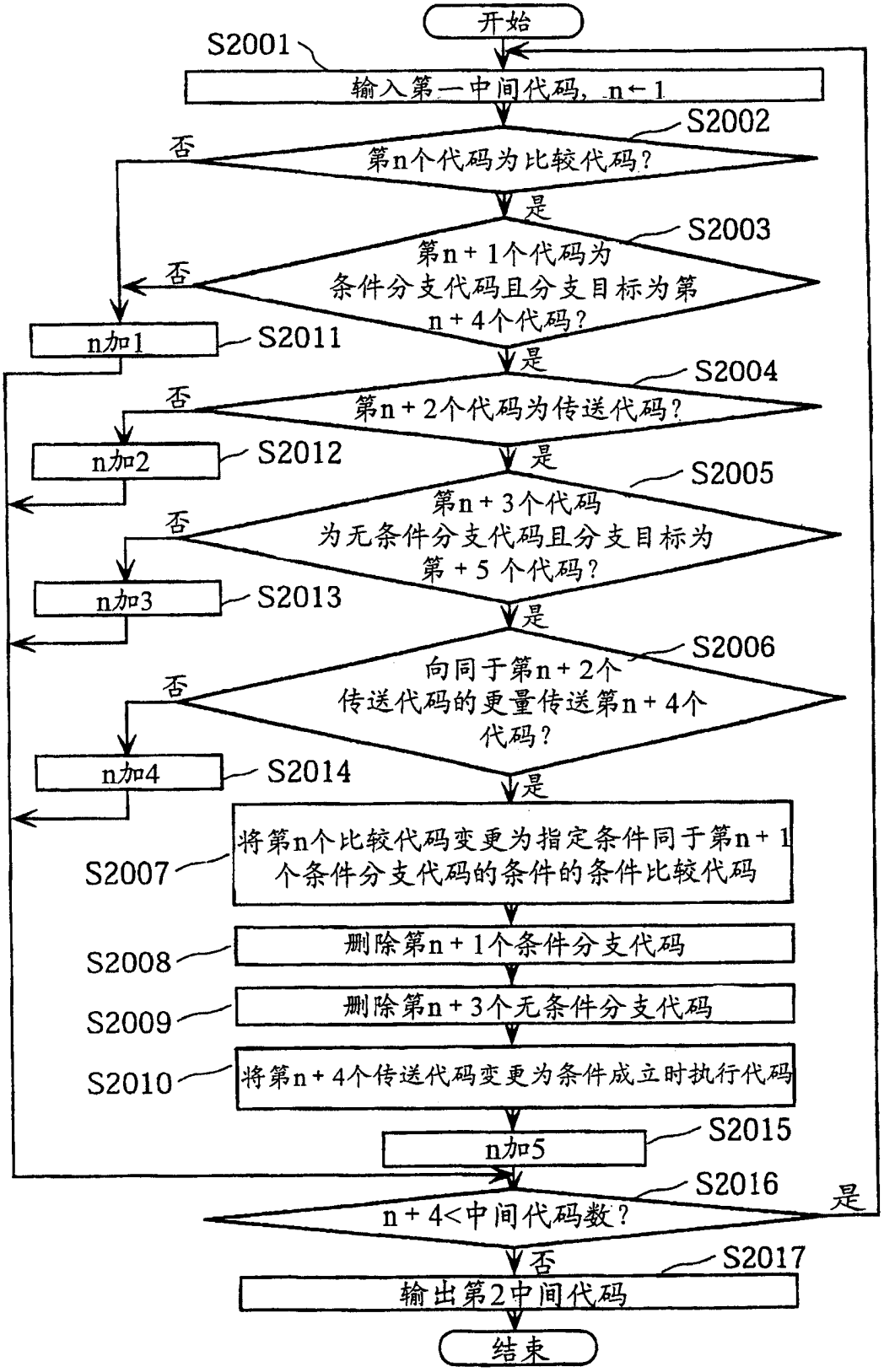

开始
S2001
输入第一中间代码，n←1
S2002
第n个代码为比较代码？
否
是
S2003
第n＋1个代码为条件分支代码且分支目标为第n＋4个代码？
否
是
n加1
S2011
S2004
第n＋2个代码为传送代码？
否
是
n加2
S2012
S2005
第n＋3个代码为无条件分支代码且分支目标为第＋5个代码？
否
是
n加3
S2013
S2006
向同于第n＋2个传送代码的更量传送第n＋4个代码？
否
是
n加4
S2014
S2007
将第n个比较代码变更为指定条件同于第n＋1个条件分支代码的条件的条件比较代码
S2008
删除第n＋1个条件分支代码
S2009
删除第n＋3个无条件分支代码
S2010
将第n＋4个传送代码变更为条件成立时执行代码
n加5
S2015
S2016
n＋4<中间代码数？
是
否
S2017
输出第2中间代码
结束

图 20

```
a cmp b
c=0
c=:eq 1
jsr  f
```

图 21

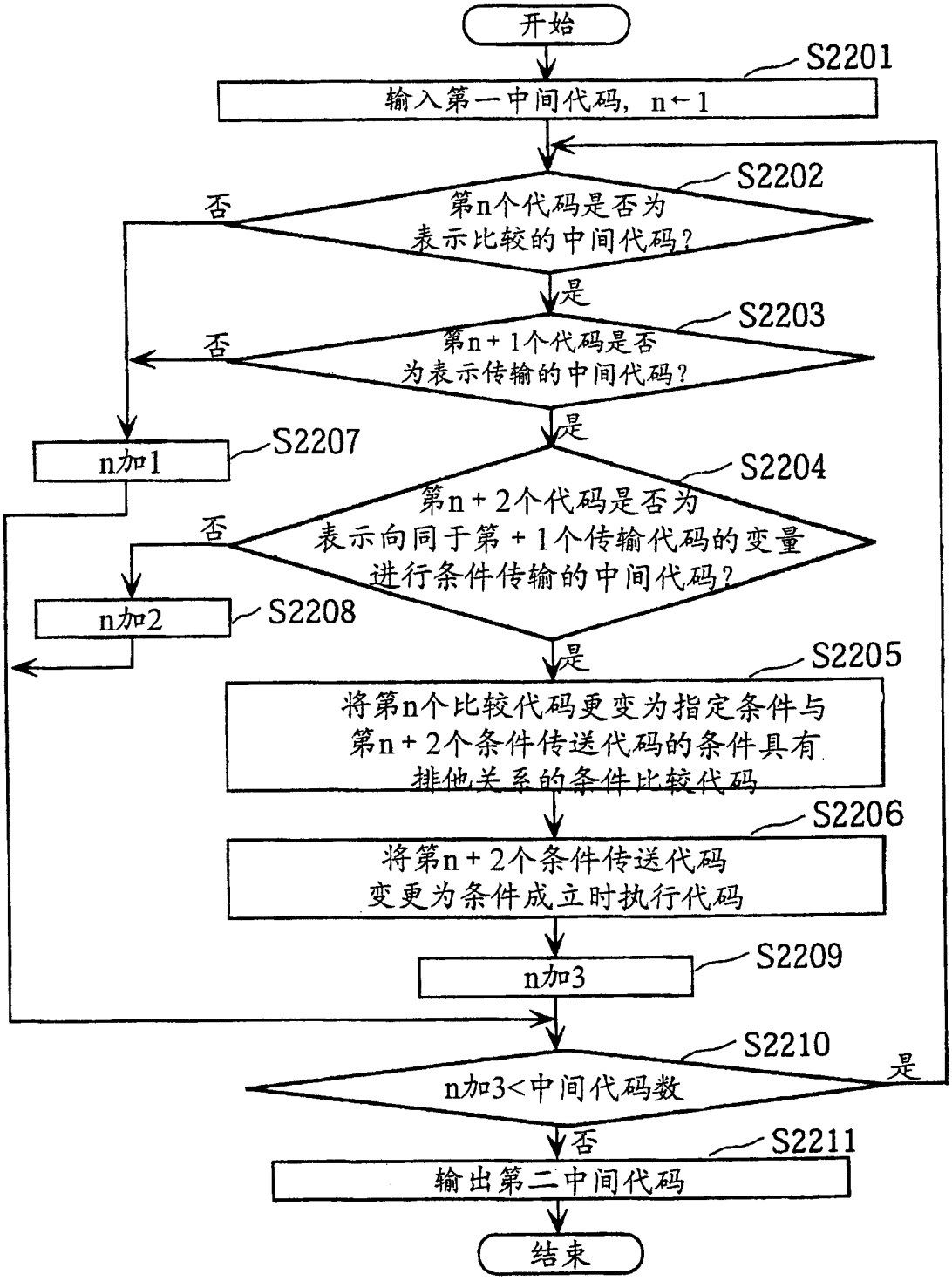

开始
S2201
输入第一中间代码，n←1
S2202
第n个代码是否为
表示比较的中间代码？
否
是
S2203
第n＋1个代码是否
为表示传输的中间代码？
否
是
n加1
S2207
S2204
第n＋2个代码是否为
表示向同于第＋1个传输代码的变量
进行条件传输的中间代码？
否
是
n加2
S2208
S2205
将第n个比较代码更变为指定条件与
第n＋2个条件传送代码的条件具有
排他关系的条件比较代码
S2206
将第n＋2个条件传送代码
变更为条件成立时执行代码
n加3
S2209
S2210
n加3<中间代码数
是
否
S2211
输出第二中间代码
结束

图 22

```
a cmp b      ←2301
c=:ne 0      ←2302
c=:eq 1      ←2303
jsr  f       ←2304
```

图 23

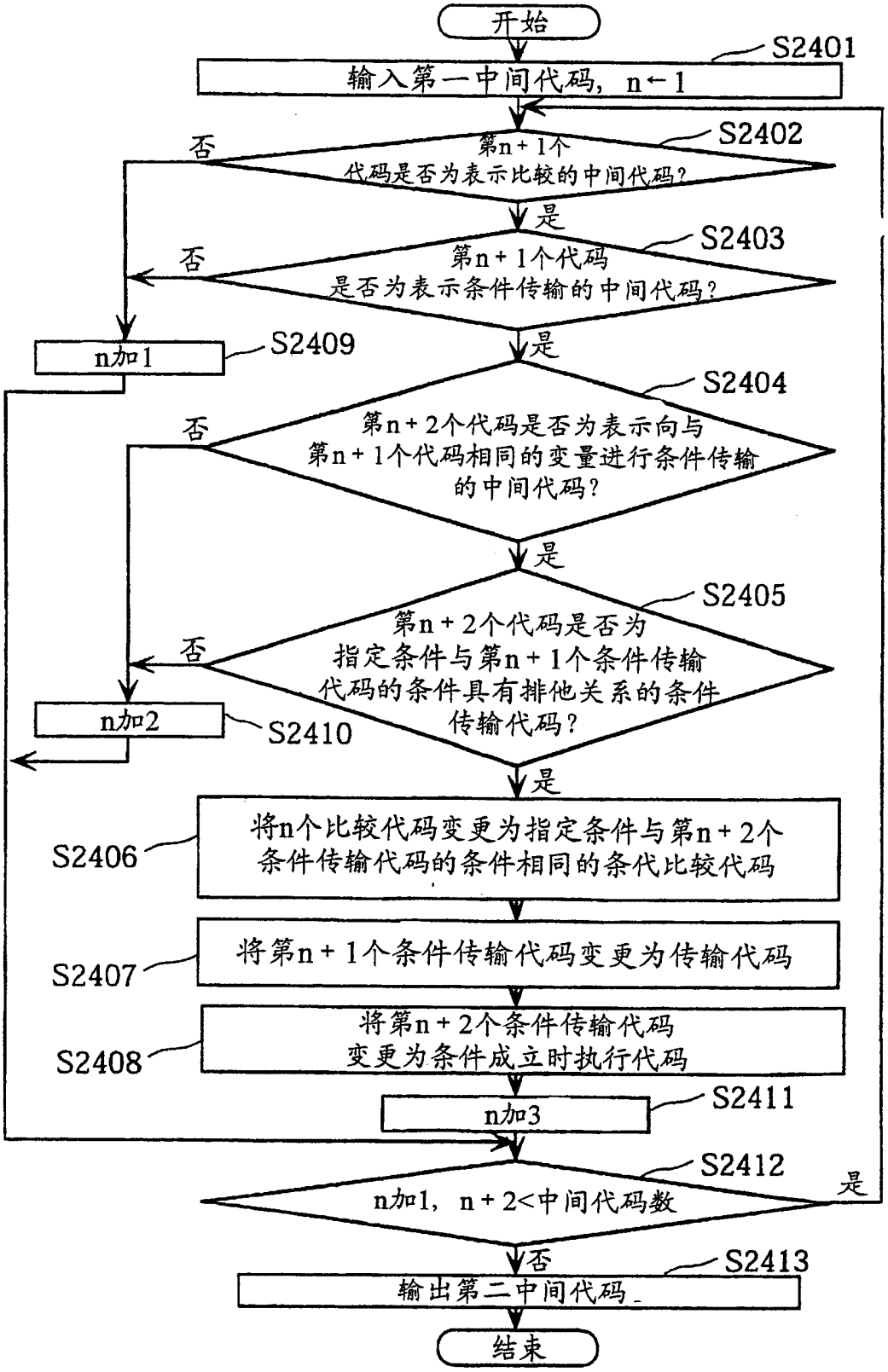

开始
S2401
输入第一中间代码，n←1
S2402
第n+1个代码是否为表示比较的中间代码？
否
是
S2403
第n+1个代码是否为表示条件传输的中间代码？
否
是
n加1
S2409
S2404
第n+2个代码是否为表示向与第n+1个代码相同的变量进行条件传输的中间代码？
否
是
S2405
第n+2个代码是否为指定条件与第n+1个条件传输代码的条件具有排他关系的条件传输代码？
否
是
n加2
S2410
S2406
将n个比较代码变更为指定条件与第n+2个条件传输代码的条件相同的条代比较代码
S2407
将第n+1个条件传输代码变更为传输代码
S2408
将第n+2个条件传输代码变更为条件成立时执行代码
n加3
S2411
S2412
n加1，n+2<中间代码数
是
否
S2413
输出第二中间代码
结束

图 24

条件运算指令	←2501	条件 ←2502
addeq	←2503	=
addgt	←2504	>
addge	←2505	≧

图 25

```
if(a != b)
{
        d = c + 1;
}
else
{
        d = c + 2;
}
f ();
```

图 26

```
        a cmp b
        bne  Lt
        d =  c + 2
        jmp L
Lt:     d =  c + 1
L:      jsr  f
```

图 27

```
a cmp b
d =  c + 1
d =  c +:eq 2
jsr  f
```

图 28

```
cmp       r0,r1
add       1,r2,r3
addeq     2,r2,r3
```

图 29

```
mov     1,r0    ←3001
cmp     r1,r2   ←3002
moveq   0,r0    ←3003
```

图 30

条件运算指令3101	条件 3102
moveq	=
movne	≠
movgt	>
movge	≧
movlt	<
movle	≦

图 31

指令缩写	条件指定	操作
[比较]		
cmp Rn,Rm		比较Rm和Rn屏将结果反映到运算标记中
[条件相加]		
addeq Rd,Rn,Rm	=	CMP指令结果为相等时将Rm与Rn相加，并将结果存入Rd中
addne Rd,Rn,Rm	≠	CMP指令结果为不相等时将Rm和Rn相加，并将结果存入Rd中
addge Rd,Rn,Rm	≧	CMP指令结果为符号数据并为大于等于时将Rm与Rn相加,将结果存入Rd中
addle Rd,Rn,Rm	≦	CMP指令结果为符号数据并为小于等于时将Rm与Rn相加,将结果存入Rd中
addgt Rd,Rn,Rm	>	CMP指令结果为符号数据并为大于时将Rm与Rn相同，将结果存入Rd中
addlt Rd,Rn,Rm	<	CMP指令结果为符号数据并为小于时将Rm与Rn相加，将结果存入Rd中
addhs Rd,Rn,Rm	≧	CMP指令结果为无符号数据并为大于等于时将Rm与Rn相加,将结果存入Rd中
addls Rd,RnRm	≦	CMP指令结果为无符号数据并为小于等于时将Rm与Rn相加,将结果存入Rd中
addhi Rd,Rn,Rm	>	CMP指令结果为无符号数据并为大于时将Rm与Rn相加，将结果存入Rd中
addlo Rd,Rn,Rm	<	CMP指令结果为无符号数据并为小于时将Rm与Rn相加，将结果存入Rd中
[条件传送]		
moveq Rd,Rm	=	CMP指令结果为相等时将Rm传送到Rn中
movne Rd,Rm	≠	CMP指令结果为不相等时将Rm传送到Rn中
movge Rd,Rm	≧	CMP指令结果为符号数据并为大于等于时将Rm传送到Rn中
movle Rd,Rm	≦	CMP指令结果为符号数据并为小于等于时将Rm传送到Rn中
movgt Rd,Rm	>	CMP指令结果为符号数据并为大于时将Rm传送到Rn中
movlt Rd,Rm	<	CMP指令结果为符号数据并为小于时将Rm传送到Rn中
movhs Rd,Rm	≧	CMP指令结果为无符号数据并为大于等于时将Rm传送到Rn中
movls Rd,Rm	≦	CMP指令结果为无符号数据并为小于等于时将Rm传送到Rn中
movhi Rd,Rm	>	CMP指令结果为无符号数据并为大于时将Rm传送到Rn中
movlo Rd,Rm	<	CMP指令结果为无符号数据并为小于时将Rm传送到Rn中
[条件分支]		
beq label	=	CMP指令结果为相等时分支到label
bne label	≠	CMP指令结果为不相等时分支到label
bge label	≧	CMP指令结果为符号数据并为大于等于时分支到label
ble label	≦	CMP指令结果为符号数据并为小于等于时分支到label
bgt label	>	CMP指令结果为符号数据并为大于时分支到label
blt label	<	CMP指令结果为符号数据并为小于时分支到label
bhs label	≧	CMP指令结果为无符号数据并为大于等于时分支到label
bls label	=	CMP指令结果为无符号数据并为小于等于时分支到label
bhi label	>	CMP指令结果为无符号数据并为大于时分支到label
blo label	<	CMP指令结果为无符号数据并为小于时分支到label

图 32

指令缩写	条件指定	操作
[比较]		
cmp/eq Rm,Rd	=	CMP指令结果为相等时设置条件标记,其它时候对条件标记清零
cmp/ge Rm,Rd	≧	CMP指令结果为符号数据并为大于等于时设置条件标记，其它时候对条件标记清零
cmp/gt Rm,Rd	>	CMP指令结果为符号数据并为大于时设置条件标记，其它时候对条件标记清零
cmp/hs Rm,Rd	≧	CMP指令结果为无符号数据并为大于等于时设置条件标记，其它时候对条件标记清零
cmp/hi Rm,Rd	>	CMP指令结果为无符号数据并为大于时设置条件标记，其它时候对条件标记清零
[条件相加]		
addt Rd,Rd,Rm	—	设置条件标记时将Rm与Rd相加，并将结果存入Rd中
addf Rd,Rd,Rm	—	清零条件标记时将Rm与Rd相加，并将结果存入Rd中
[条件传送]		
movt Rd,Rm	—	设置条件标记时将Rm传送到Rd中
movf Rd,Rm	—	清零条件标记时将Rm传送到Rd中
[条件分支]		
bt label	—	设置条件标记时进行分支
bf label	—	清零条件标记时进行分支

图 33