



[12] 发明专利说明书

专利号 ZL 200410007817.8

[45] 授权公告日 2006 年 7 月 12 日

[11] 授权公告号 CN 1264089C

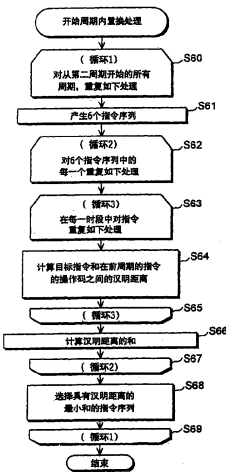
[22] 申请日 2004.1.21
[21] 申请号 200410007817.8
[30] 优先权
[32] 2003.1.28 [33] JP [31] 019365/2003
[71] 专利权人 松下电器产业株式会社
地址 日本大阪府
[72] 发明人 瓶子岳人 小川一 谷丈畅
笹川幸宏
审查员 李楠

[74] 专利代理机构 永新专利商标代理有限公司
代理人 夏青

权利要求书 3 页 说明书 39 页 附图 45 页

[54] 发明名称
编译设备和编译方法

[57] 摘要
一种编译设备，能够产生指令序列、以使具有并行处理能力的处理器以低功耗工作，该编译设备将源程序翻译为用于处理器的机器语言程序，该处理器包括可并行执行指令的多个执行单元、以及多个指令发出单元，它们发出分别由多个执行单元执行的指令，该编译设备包括：语法分析单元，用于对源程序进行语法分析；中间码转换单元，用于将经过语法分析的源程序转换为中间码；优化单元，用于优化中间码，从而减小在连续的指令周期中位于与同一指令发出单元相对应位置的指令之间的汉明距离，而不改变对应于中间码的指令之间的相关性；以及代码产生单元，用于将优化的中间码转换为机器语言指令。



1. 一种编译设备，将源程序翻译为用于处理器的机器语言程序，该处理器包括多个执行单元以及多个指令发出单元，所述执行单元能够并行执行指令，所述指令发出单元发出分别由多个执行单元执行的指令，该编译设备包括：

语法分析单元，用于对源程序进行语法分析；

中间码转换单元，用于将经过语法分析的源程序转换为中间码；

优化单元，用于优化所述中间码，从而减小指令之间的汉明距离，这些指令在连续的指令周期中位于将要提供指令的位置，且该位置与同一个指令发出单元相对应，而不改变对应于中间码的指令之间的相关性；以及

代码产生单元，用于将被优化的中间码转换为机器语言指令。

2. 根据权利要求1的编译设备，

其中，优化单元通过将具有较高优先级的指令安排在将要提供指令的位置上，且该位置与多个指令发出单元中的每一个相对应，来优化中间码，而不改变对应于中间码的指令之间的相关性，所述具有较高优先级的指令与在紧邻的先前周期中被安排在将要提供指令的位置上，且该位置对应于同一个指令发出单元的指令间，具有较小的汉明距离。

3. 根据权利要求2的编译设备，

其中，优化单元将具有较高优先级的指令安排在将要提供指令的位置上，且该位置与多个指令发出单元中的每一个相对应，所述具有较高优先级的指令与在紧邻的先前周期中被安排在将要提供指令的位置上，且该位置对应于同一个指令发出单元的指令间，具有较小的操作码的汉明距离。

4. 根据权利要求2的编译设备，

其中，优化单元将具有较高优先级的指令安排在将要提供指令的位置上，且该位置与多个指令发出单元中的每一个相对应，所述具有较高优先级的指令与在紧邻的先前周期中被安排在将要提供指令的位置上，且该位置对应于同一个指令发出单元的指令间，具有较小的寄存器序号的汉明距离。

5. 根据权利要求1的编译设备，

其中，优化单元通过置换在所述连续指令周期内的一目标指令周期中安排的指令来优化中间码，从而减小与在紧邻的先前周期中被安排在将要提供指令的位置上，且该位置对应于同一个指令发出单元的指令间的汉明距离，而不改变对应于中间码的指令之间的相关性。

6. 根据权利要求5的编译设备，

其中，优化单元置换在所述连续指令周期内的一目标指令周期中安排的指令，从而减小与在紧邻的先前周期中被安排在将要提供指令的位置上，且该位置与多个指令发出单元相对应的多个指令间的汉明距离之和。

7. 根据权利要求5的编译设备，

其中，优化单元置换在所述连续指令周期内的一目标指令周期中安排的指令，从而减小与在紧邻的先前周期中被安排在将要提供指令的位置上，且该位置与同一个指令发出单元相对应的指令间的操作码的汉明距离。

8. 根据权利要求5的编译设备，

其中，优化单元置换在所述连续指令周期内的一目标指令周期中安排的指令，从而减小与在紧邻的先前周期中被安排将要提供指令的位置上，且该位置与同一个指令发出单元对应的指令间的寄存器序号的汉明距离。

9. 一种编译方法，将源程序翻译为用于处理器的机器语言程序，该处理器包括多个执行单元以及多个指令发出单元，所述执行单元能够并行执行指令，所述指令发出单元发出分别由所述多个执行单元执行的指令，该编译方法包括：

语法分析步骤，对源程序进行语法分析；

中间码转换步骤，将经过语法分析的源程序转换为中间码；

优化步骤，优化中间码，从而减小指令之间的汉明距离，这些指令在连续指令周期中位于将要提供指令的位置，且该位置与同一个指令发出单元相对应，而不改变对应于中间码的指令之间的相关性；以及

代码产生步骤，将被优化的中间码转换为机器语言指令。

10. 根据权利要求9的编译方法，

其中，在该优化步骤中，通过将具有较高优先级的指令安排在将要提供指令的位置，且该位置与所述多个指令发出单元中的每一个相对应，来优化中间码，而不改变对应于中间码的指令之间的相关性，所述具有较高优先级的指令与在紧邻的先前周期中被安排将要提供指令的位置，且该位置与同一个指令发出单元相对应的指令间，具有较小的汉明距离。

11. 根据权利要求9的编译方法，

其中，在该优化步骤中，通过置换被安排在所述连续指令周期中的一目标指令周期中的指令，优化所述中间码，从而减小与紧邻的先前周期中被安排在将要提供指令的位置上，且该位置与同一个指令发出单元相对应的指令间的汉明距离，而不改变对应于中间码的指令之间的相关性。

编译设备和编译方法

技术领域

本发明涉及将以高级语言（如C/C++语言）描述的源程序转换为机器语言程序的编译器，尤其是能够输出机器语言程序的编译器，该机器语言程序可以以低功耗执行。

背景技术

在最近几年广泛使用的移动信息处理设备，例如，移动电话和个人数字助理（PDA），需要降低功耗。因此，逐渐需要开发能够有效利用这些信息处理设备中使用的处理器的高级功能的编译器，并且产生机器级指令，这些指令可以由处理器以低功耗执行。

如传统的编译器，通过改变指令的执行顺序，用于降低处理器功耗的指令序列优化设备，在日本未决专利申请No. 8-101777中予以公开。

该指令序列优化设备置换这些指令，从而减小指令的位模式之间的汉明距离，而不改变指令之间的相关性。因此，它可以实现指令序列的优化，从而导致处理器功耗的降低。

然而，传统的指令序列优化设备并不能认定能执行并行处理的处理器。因此，存在一个问题，即，即使将传统的优化处理应用在具有并行处理能力的处理器中，也不能获得最佳指令序列。

发明内容

本发明基于上述背景构思，目的在于提供一种编译器，它能够产生可由具有并行处理能力的处理器以低功耗执行的指令序列。

为了实现上述目的，根据本发明的编译设备，是一种将源程序翻译为

用于处理器的机器语言程序的编译设备：该处理器包括多个执行单元，它们能够并行执行指令，以及多个指令发出单元，它们发出由多个执行单元分别执行的指令，该编译设备包括：语法分析单元，用于对源程序进行语法分析；中间码转换单元，用于将经过语法分析的源程序转换为中间码；优化单元，用于优化中间码，从而减小在连续的指令周期中被安排在与同样的指令发出单元对应位置的指令之间的汉明距离，而不改变对应于中间码的指令之间的相关性；以及代码产生单元，用于将被优化的中间码转换为机器语言指令。优选地，该优化单元通过将具有较高的优先级的指令安排在对应于多个指令发出单元中每一个的位置上来优化中间码，而不改变对应于中间码的指令之间的相关性，所述具有较高优先级的指令与在一个紧接先前的周期中被安排在与同一指令发出单元对应的位置上的指令间，具有较小的汉明距离。

因此，由于可以抑制由每一个执行单元执行的指令的位模式的改变，保存在处理器的指令寄存器中值的位改变保持较小，从而产生可以由该处理器以低功耗执行的指令序列。

根据本发明另一方面的编译设备是一种将源程序翻译为用于处理器的机器语言程序的编译设备，该处理器包括多个执行单元，它们能够并行执行指令，以及多个指令发出单元，发出由多个执行单元分别执行的指令，该编译设备包括：语法分析单元，用于对源程序进行语法分析；中间码转换单元，用于将经过语法分析的源程序转换为中间码；优化单元，用于优化中间码，从而在连续的指令周期中访问同一寄存器，而不改变对应于中间码的指令之间的相关性；以及代码产生单元，用于将被优化的中间码转换为机器语言指令。优选地，该优化单元通过将具有较高优先级的指令安排在对应于多个指令发出单元中的每一个的位置，优化中间码，而不改变对应于中间码的指令之间的相关性，所述具有较高优先级的指令用于访问在紧接先前指令周期中被安排与同一指令发出单元对应位置的指令的寄存器。

因此，重复对一个寄存器的访问，并使用用于选择寄存器的控制信号的

变化变小，从而产生可以由处理器以低功耗执行的指令序列。

根据本发明的又一方面的编译设备是一种将源程序翻译为用于处理器的机器语言程序的编译设备，该处理器包括多个执行单元，它们能够并行执行指令，以及多个指令发出单元，其发出由多个执行单元分别执行的指令，其中所发出的具有较高优先级的指令是预定用于所述多个指令发出单元的每一个，以及该编译设备包括：语法分析单元，用于对源程序进行语法分析；中间码转换单元，用于将经过语法分析的源程序转换为中间码；优化单元，通过将所述具有较高优先级的预定指令放在与多个指令发出单元的每一个对应的位置上，优化中间码，而不改变对应于中间码的指令之间的相关性；以及代码产生单元，用于将被优化的中间码转换为机器语言指令。

因此，如果利用处理器的同一组成部分的指令被分配为将由同一指令发出单元按优先级发出的指令，使用同一组成部分的指令在同一执行单元中被连续执行。因此，产生能够由处理器以低功耗执行的指令序列。

根据本发明的又一方面的编译设备是一种将源程序翻译为用于处理器的机器语言程序的编译设备，该处理器包括多个执行单元，它们能够并行执行指令，以及多个指令发出单元，发出由多个执行单元分别执行的指令，该编译设备包括：语法分析单元，用于对源程序进行语法分析；中间码转换单元，用于将经过语法分析的源程序转换为中间码；间隔检测单元，用于连续检测在预定数量的指令周期中，检测在分别对应于其中将安排指令的多个指令发出单元的多个位置当中、预定数量的位置中没有安排指令的间隔；第一指令插入单元，用于在紧接该间隔之前插入一条指令，以停止对应于没有被安排指令的位置的指令发出单元的操作；以及代码产生单元，用于将被优化的中间码转换为机器语言指令。

因此，当指令没有在某一个间隔被安排在对应用于指令发出单元的位置上时，在该间隔期间可以停止提供给该指令发出单元的电源。因此，产生能够由处理器以低功耗执行的指令序列。

根据本发明的又一方面的编译设备是一种将源程序翻译为用于处理器

的机器语言程序的编译设备，该处理器包括多个执行单元，它们能够并行执行指令，以及多个指令发出单元，它们发出由多个执行单元分别执行的指令，该编译设备包括：语法分析单元，用于对源程序进行语法分析；中间码转换单元，用于将经过语法分析的源程序转换为中间码；优化单元，通过安排指令优化中间码，从而仅使指定数量的指令发出单元工作，而不改变对应于中间码的指令之间的相关性；以及代码产生单元，将被优化的中间码转换为机器语言指令。优选地，源程序包括单元数量指定信息，其指定处理器使用的指令发出单元的数量，以及优化单元，通过安排指令优化中间码，从而仅使由单元数量指定信息所指定数量的指令发出单元工作，而不改变对应于中间码的指令之间的相关性。

因此，根据由数量指定信息指定的指令，该优化单元能够产生不发出指令的指令发出单元，并且停止向该指令发出单元提供电源。因此，产生能够由处理器以低功耗执行的指令序列。

更优选地，上述编译设备进一步包括：接受单元，用于接受处理器使用的指令发出单元的数量，其中优化单元通过安排指令优化中间码，从而仅使由接受单元接受的数量的指令发出单元工作，而不改变对应于中间码的指令之间的相关性。

因此，可以仅使由接受单元接受的数量的指令发出单元工作，并且停止对其它指令发出单元提供电源。因此，产生能够由处理器以低功耗执行的指令序列。

应当注意，本发明不仅可以被实现为如上所述的编译器，也可以被实现为在该编译设备中包括的单元执行的步骤的编译方法，以及实现为用于该特性编译器的程序或者计算机可读记录介质。无须说明，该程序和数据文件能够通过记录介质（如CD-ROM）、以及传输介质（如因特网）广泛分发。

从以上说明中显而易见，根据本发明的编译设备抑制在处理器的指令寄存器中保存的值的位变化，从而产生能够由处理器以低功耗执行的指令序列。

同样，对一个寄存器进行重复访问，并使用于选择寄存器的控制信号的改变变小，从而产生能够由处理器以低功耗执行的指令序列。

同样，由于使用同样组成部分的指令能够连续在某些周期的同一时段中被执行，产生能够由处理器以低功耗执行的指令序列。

此外，由于可以停止向空闲时段提供电源，产生能够由处理器以低功耗执行的指令序列。

如上所述，根据本发明的编译设备允许具有并行处理能力的处理器以低功耗运行。尤其是，可以产生适合于需要低功耗运行的设备使用的处理器的指令序列（机器语言程序），诸如移动信息处理设备（如移动电话、PDA或者类似设备），因此本发明的实际价值非常高。

关于本申请的技术背景的进一步信息，参见2003年1月28日提出的日本专利申请No. 2003-019365，将其结合在此以供参考。

附图简述

从下面结合示出本发明的具体实施例的附图进行描述，使本发明的这些和其它目的、优点和特征将变得明显。在图中：

图1A-图1D表示本发明实施例中由处理器解码和执行的指令的结构示意图；

图2表示本发明实施例中处理器的示意结构框图；

图3是表示分组的例子的图；

图4（a）和图4（b）是用于说明包括在分组中的并行执行边界信息的示意图；

图5A-5C是表示执行指令的单元的实例的示意图，这些指令基于分组的并行执行边界信息创建并且以并行方式执行；

图6表示算术和逻辑/比较操作单元的示意结构框图；

图7表示桶形移位器的示意结构框图；

图8表示除法器的示意结构框图；

图9表示乘法/乘积求和操作单元的示意结构框图;

图10是表示当处理器执行指令时执行的每一个流水线操作的时序图;

图11是表示由处理器执行的指令、处理的细节和指令的位模式的示意图;

图12是表示根据当前实施例的编译器结构的功能框图;

图13是表示指令调度单元操作的流程图;

图14A和图14B是表示相关性图的实例的示意图;

图15是表示指令调度结果的实例的示意图;

图16是表示图13中所示的最佳指令提取处理的操作的流程图;

图17A和图17B是用于说明如何计算操作码中位模式之间汉明距离的示意图;

图18A和图18B是用于说明如何计算具有不同位长的操作码之间的汉明距离的示意图;

图19是表示周期内置换处理单元的操作的流程图;

图20A-图20F是表示指令序列的六种模式的实例的示意图;

图21是表示被安排指令的实例的图;

图22A-图22F是用于说明创建指令序列的处理的示意图(图19中的S61);

图23是说明计算操作码之间汉明距离的处理的示意图(图19中的S64);

图24表示寄存器分配单元的操作的流程图;

图25表示变量作为分配对象的生存期的示意图;

图26表示基于图25中的实例创建的变量的冲突图的示意图;

图27A-图27C是表示在指令调度过程中获得的结果的示意图;

图28表示指令再调度单元的操作的流程图;

图29表示图28中最佳指令提取过程的操作的流程图;

图30A和图30B是用于说明指定安排候选指令的处理的示意图(图29中的S152);

图31A和图31B是用于说明指定安排候选指令的处理的示意图(图29中的

S156);

图32A和图32B是用于说明指定安排候选指令的处理的示意图 (图29中的S160);

图33是表示时段停止/恢复指令产生单元的操作的流程图;

图34是表示其中安排指令的调度结果的实例的示意图;

图35是表示调度结果的示例的示意图, 其中写入指令作为仅连续使用一个指定时段的情况下的处理;

图36是表示调度结果的示例的示意图, 其中写入指令作为仅连续使用两个指定时段的情况下的处理;

图37 (a) 至图37 (d) 是表示程序状态寄存器的实例的示意图;

图38 (a) 至图38 (h) 是表示程序状态寄存器的另一个实例的示意图;

图39是表示图28中所示的最佳指令提取过程的其它操作的流程图;

图40A和图40B是用于说明指定一个安排候选指令的处理的示意图 (图39中的S212);

图41是表示周期内置换处理单元237的操作的第一变型的流程图;

图42是用于说明计算指令之间的汉明距离的处理的示意图 (图41中的S222);

图43是表示周期内置换处理单元237的操作的第二变型的流程图;

图44是用于说明计算寄存器字段之间的汉明距离的处理的示意图 (图43中的S232);

图45是表示周期内置换处理单元237的操作的第三变型的流程图;

图46表示被安排指令的实例的示意图;

图47A-图47F是用于说明创建指令序列的处理的示意图 (图45中的S61);

图48是用于说明计算寄存器字段的数目的处理的示意图 (图45中的S242);

图49是表示周期内置换处理单元237的操作的第四变型的流程图。

优选实施例的描述

以下将参考附图详细描述根据本发明的编译器的实施例。

当前实施例的编译器为交叉编译器，用于将用高级语言（如C/C++语言）描述的源程序翻译为能够由指定处理器（目标）执行的机器语言，并且具有降低处理器功耗的特点。

（处理器）

首先，将参考图1A-图11，说明由本发明实施例中的编译器实现的处理器的实例。

具有的可执行指令的并行性高于微计算机的流水线系统被用于由当前实施例中的编译器实现的处理器，从而并行执行多个指令。

图1A-图1D是表示当前实施例中由处理器解码并且执行的指令的结构示意图。如图1A-图1D所示，由处理器执行的每一条指令具有32位的固定长度。每一条指令的第0位表示并行执行边界信息。当并行执行边界信息为“1”时，在指令和其后指令之间存在一个并行执行的边界。当并行执行边界信息为“0”时，则不存在并行执行的边界。下面将描述如何利用并行执行边界信息。

在从每一条指令的指令长度除去并行执行边界信息之外的31位中确定操作。具体来说，在“OP1”，“OP2”，“OP3”和“OP4”字段中，指定表示操作类型的操作码。在寄存器字段“Rs”，“Rs1”和“Rs2”中，指定了作为源操作数寄存器的寄存器序号。在寄存器字段“Rd”中，指定了作为目的操作数的寄存器的寄存器序号。在字段“Imm”中，指定了操作的恒定操作数。在字段“Disp”中，指定了位移。

操作码的前2位（第30和31位）用于指定操作（一组操作）的类型。下面将描述这两位细节。

操作码OP2-OP4是16位长度的数据，而操作码OP1是21位长度的数据。因此，为方便起见，操作码OP1的前一半（第16-31位）被称为操作码OP1-1，

而其第二半（第11-15位）被称为操作码OP1-2。

图2是表示当前实施例中处理器的示意结构框图。处理器30包括一个指令存储器40，用于存储根据VLIW（超长指令字）描述的指令组（下面称做“分组”）；指令提供/发出单元50；解码单元60；执行单元70和数据存储器100。下面将详细描述这些单元的每一个。

图3是表示分组的一个实例的图。定义一个分组是指令提取的单元，并且由四条指令组成。如上所述，一条指令为32位长。因此，一个分组为128（ $=32 \times 4$ ）位长。

再回到图2，指令提供/发出单元50连接到指令存储器40、解码单元60和执行单元70，并根据执行单元70提供的PC（程序计数器）的数值从指令存储器40接收分组，并且将3条或更少的指令并行发出至解码单元60。

解码单元60连接到指令提供/发出单元50和执行单元70，并将由指令提供/发出单元50发出的指令解码，且将解码的指令发出至执行单元70。

执行单元70连接到指令提供/发出单元50、解码单元60和数据存储器100，并且如果必要，访问存储在数据存储器100中的数据，并且根据基于由解码单元60提供的解码结果的指令执行处理。每次执行处理时，执行单元70的PC数值逐一增加。

指令提供/发出单元50包括：指令提取单元52，它连接到指令存储器40和下面将描述的执行单元70中的PC单元，访问由存储在PC单元的程序计数器表示的指令存储器40中的地址，并从指令存储器40接收分组；指令缓冲区54，连接到指令提取单元52，并且暂时保存分组；以及指令寄存器单元56，连接到指令缓冲区54，并且保存每一个分组中包括的3条或更少的指令。

指令提取单元52和指令存储器40通过IA（指令地址）总线42和ID（指令数据）总线44彼此连接。IA总线42为32位宽，而ID总线44为128位宽。通过IA总线42，地址从指令提取单元52提供给指令存储器40。通过ID总线44，分组从指令存储器40提供给指令提取单元52。

指令寄存器单元56包括指令寄存器56a-56c，这些指令寄存器分别连接

到指令缓冲区54，并且分别容纳一条指令。

解码单元60包括：指令发出控制单元62，控制指令的发出，所述指令保存在指令寄存器单元56中的三个指令寄存器56a-56c中；以及解码子单元64，连接到指令发出控制单元62和指令寄存器单元56，在指令发出控制单元62的控制下，将由指令寄存器单元56提供的指令解码。

解码子单元64包括指令解码器64a-64c，这些指令解码器分别连接到指令寄存器56a-56c，在一个周期内基本上解码一条指令，从而输出控制信号。

执行单元70包括：执行控制单元72，连接到解码子单元64，并且根据从解码子单元64中的三个指令解码器64a-64c输出的控制信号，控制下面所要描述的执行单元70的每一个组成部分；PC单元74，保存接下来将要执行的分组的地址；寄存器文件76，该文件由32个包括R0~R31的32位寄存器构成；算术和逻辑/比较操作单元（AL/C操作单元）78a-78c，执行SIMD（单指令多数据）类型指令的操作；乘法/乘积求和操作单元（M/PS操作单元）80a和80b，这些单元与算术和逻辑/比较操作单元78a-78c相类似，能够执行SIMD类型的指令，并且计算65位或更少长度的结果，而不降低位准确性。

执行单元70进一步包括：桶形移位器82a-82c，分别执行数据的算术移位（求补系统的移位）或者逻辑移位（无符号移位）；除法器84；操作数访问单元88，连接到数据存储器，并且向存储器100发送数据和从存储器100接收数据；32位宽的数据总线90（一条L1总线，一条R1总线，一条L2总线，一条R2总线，一条L3总线和一条R3总线）；以及32位宽的数据总线92（一条D1总线，一条D2总线和一条D3总线）。

寄存器文件76由32个包括R0-R31的32位寄存器构成。用于向L1总线、R1总线、L2总线、R2总线、L3总线和R3总线输出数据的寄存器文件76中的寄存器，是分别根据由执行控制单元72提供给寄存器文件76的控制信号CL1、CR1、CL2、CR2、CL3和CR3选择的。通过D1总线、D2总线和D3总线所传输的数据被写入的寄存器，分别根据由执行控制单元72向寄存器文件76提供的控制信号CD1、CD2和CD3进行选择。

算术和逻辑/比较操作单元78a的两个输入端口分别连接到L1总线和R1总线，它的输出端口连接到D1总线。算术和逻辑/比较操作单元78b的两个输入端口分别连接到L2总线和R2总线，它的输出端口连接到D2总线。算术和逻辑/比较操作单元78c的两个输入端口分别连接到L3总线和R3总线，它的输出端口连接到D3总线。

所述乘法/乘积求和操作单元80a的四个输入端，分别连接至L1总线、R1总线、L2总线和R2总线，且该单元的两个输出端分别连接至D1总线和D2总线。所述乘法/乘积求和操作单元80b的四个输入端，分别连接至L2总线、R2总线、L3总线和R3总线，且该单元的两个输出端分别连接至D2总线和D3总线。

所述桶形移位器82a的两个输入端，分别连接至L1总线和R1总线，且其输出端连接至D1总线。所述桶形移位器82b的两个输入端，分别连接至L2总线和R2总线，且其输出端连接至D2总线。所述桶形移位器82c的两个输入端分别连接至L3总线和R3总线，且其输出端连接至D3总线。

所述除法器84的两个输入端分别连接至L1总线和R1总线，且其输出端连接至D1总线。

所述操作数访问单元88和所述数据存储器100，通过一OA（操作数地址）总线96和一OD（操作数数据）总线94，彼此相互连接。所述OA总线96和OD总线94分别为32位宽。所述操作数访问单元88进一步通过OA总线96，指定所述数据存储器100的地址，并通过OD总线94，在该地址进行数据读取和写入。

所述操作数访问单元88也连接至D1总线、D2总线、D3总线、L1总线和R1总线，并向这些总线中的任意一个发送数据，和从这些总线中的任意一个接收数据。

处理器30能够并行执行三条指令。如下所述，一组电路能够执行包括并行执行的指令分配级、解码级、执行级和写入级的一组流水线处理，且在当前描述中被定义为一个“时段”（slot）。因此，所述处理器30具有三个时段，第一、第二和第三时段。指令寄存器56a和指令解码器64a执行的

一组处理属于第一时段，指令寄存器56b和指令解码器64b执行的一组处理属于第二时段，以及的指令寄存器56c和指令解码器64c执行的一组处理属于第三时段。

称为默认逻辑的指令被分配给相应的时段，且执行指令调度，以便在可能时在同一时段执行同一指令。例如，关于存储器访问的指令（默认逻辑）被分配给第一时段，关于乘法的默认逻辑被分配给第二时段，且其他默认逻辑被分配给第三时段。应注意的是，默认逻辑与附图1A—附图1D所示的一组操作——对应。换句话说，具有前2位为“01”、“10”和“11”的指令，分别是用于第一、第二和第三时段的默认逻辑。

用于第一时段的默认逻辑包括“ld”（载入指令），“st”（存储指令）等等。用于第二时段的默认逻辑包括“mul1”，“mul2”（乘法指令）等等。用于第三时段的默认逻辑包括“add1”，“add2”（加法指令），“sub1”，“sub2”（减法指令），“mov1”，“mov2”（寄存器之间传输的指令）等等。

附图4是说明一分组所包括的并行执行边界信息的示意图。假定一个分组112和一个分组114以该顺序存储在指令存储器40中。还假定分组112中的指令2和分组114中的指令5的并行执行边界信息为“1”，且其他指令的并行执行边界信息为“0”。

所述指令提取单元52基于PC单元74中的程序计数器的值，按顺序读取分组112和分组114，并将其顺序发出至指令缓冲区54。所述执行单元70并行执行所述指令，直到其并行执行边界信息为1的指令。

附图5A～5C示出了基于一个分组的并行执行边界信息产生且并行执行的指令的执行单元的实例。参见附图4和附图5A～5C，通过在其并行执行边界信息为“1”的指令的位置分离分组112和分组114，产生执行122～126的单元。因此，从指令缓冲区54将指令按照执行122～126的单元的顺序，发送至指令寄存器单元56。指令发出控制单元62控制这些指令的发送。

指令解码器64a～64c分别将指令寄存器56a～56c中保存的指令的操作码解码，并将控制信号输出给执行控制单元72。所述执行控制单元72基于

指令解码器64a~64c中的分析结果,对所述执行单元70的各组成部分进行不同类型的控制。

以指令“addl R3, R0”为例。该指令意味着将寄存器R3的值和寄存器R0的值相加,并将相加所得结果写入寄存器R0。在此情况下,所述执行控制单元72实行作为示例的以下控制。所述执行控制单元72将一控制信号CL1提供给寄存器文件76,以便将寄存器R3中保存的值输出给所述L1总线。同时,所述执行控制单元72将一控制信号CR1提供给所述寄存器文件76,以便将寄存器R0中保存的值输出给所述R1总线。

执行控制单元72进一步将一控制信号CD1提供给所述寄存器文件76,以便将通过D1总线获得的执行结果写入所述寄存器R0。所述执行控制单元72进一步控制所述算术和逻辑/比较操作单元78a,通过L1总线和L2总线接收寄存器R3和R0的值,将它们相加,并通过D1总线将相加的结果写入寄存器R0。

附图6是每一个算术和逻辑/比较操作单元78a~78c的示意结构的框图。参见附图6和附图2,算术和逻辑/比较操作单元78a~78c之每一个包括:通过数据总线90连接至寄存器文件76的ALU(算术和逻辑单元)132;通过ALU 132和数据总线92连接至寄存器文件76的饱和处理单元134,并执行诸如饱和、最大/最小值检测和绝对值产生的处理;以及连接至ALU 132的标记单元136,检测溢出并产生状态标记。

附图7是桶形移位器82a~82c之每一个的示意结构的框图。参见附图7和附图2,桶形移位器82a~82c之每一个包括:累加器单元142,具有用于容纳32位数据的累加器M0和M1;选择器146,通过数据总线90连接至所述累加器M0和寄存器文件76,并接收累加器M0和寄存器的值;选择器148,通过数据总线90连接至累加器M1和所述寄存器文件76,并接收累加器M1和寄存器的值;连接至所述选择器146的输出端的高位桶形移位器150;连接至所述选择器148的输出端的低位桶形移位器152;以及与所述高位桶形移位器150和所述低位桶形移位器152的输出端相连接的饱和处理单元154。

所述饱和处理单元154的输出端通过所述数据总线92, 连接至所述累加器单元142和所述寄存器文件76。

桶形移位器82a~82c的每一个通过操作其自身的组成部件, 执行数据的算术移位(在2级补码系统内移位)或逻辑移位(无符号移位)。其通常接收或输出32或64位数据。通过应用存储在另一寄存器中的移位量或一直接值, 指定将要移位的数据的移位量, 所述要移位的数据的移位量存储在寄存器文件76的寄存器中、或累加器单元142的累加器中。在向左63位和向右63位的范围之间, 执行数据的算术或逻辑移位, 且数据以与输入位长相同的位长输出。

桶形移位器82a~82c的每一个能够响应一条SIMD指令, 移位8、16、32、64位数据。例如, 其能够并行处理四个8位数据移位。

算术移位是在2级求补系统中的移位, 它通过在加法和减法、2的幂的乘法(诸如2倍, 2的2次幂, 2的-1次幂, 2的-2次幂)等等时, 按照十进制的小数点的排列执行。

附图8是除法器84的示意结构的框图。根据附图8和附图2, 所述除法器84包括: 具有容纳32位数据的累加器M0和M1的累加器单元162; 以及通过所述累加器单元162与数据总线90和92, 连接至所述寄存器文件76的除法单元164。

由于被除数为64位且除数为32位, 所述除法器84分别输出一个32位的商和一个32位的余数。为获得一个商和一个余数, 占用了34个周期。所述除法器84能够处理所有有符号和无符号的数据。然而, 为二者共同确定是否用符号标记所述被除数和除数。所述除法器84进一步具有输出一溢出标记和一0除法标记的功能。

附图9是乘法/乘积求和运算单元80a和80b的每一个的示意结构框图。根据附图9和附图2, 乘法/乘积求和运算单元80a和80b的每一个包括: 具有分别容纳64位数据的累加器M0和M1的累加器单元172; 以及32位乘法器174a和174b, 具有分别通过数据总线90连接至所述寄存器文件76的两个输入端。

乘法/乘积求和运算单元80a和80b的每一个进一步包括：64位加法器176a，该加法器与所述乘法器174a的输出和所述累加器单元172相连接；64位加法器176b，该加法器与所述乘法器174b的输出和所述累加器单元172相连接；64位加法器176c，该加法器与所述64位加法器176a的输出和64位加法器176b的输出相连接；选择器178，该选择器与所述64位加法器176b的输出和64位加法器176c的输出相连接；以及饱和处理单元180，该单元通过数据总线92与所述加法器176a的输出、所述选择器178的输出、所述累加器单元172和所述寄存器文件76相连接。

乘法/乘积求和运算单元80a和80b的每一个执行如下乘法和乘积求和运算：

- * 32×32 位有符号数据的乘法，乘积和以及乘积差运算；
- * 32×32 位无符号数据的乘法；
- * 并行执行两个 16×16 位有符号数据的乘法、乘积和以及乘积差运算；

及

- * 并行执行两个 32×16 位有符号数据的乘法、乘积和以及乘积差运算。

对整数和定点格式的数据执行上述运算。同时，这些运算的结果被四舍五入并达到饱和状态。

附图10是当上述处理器30执行指令时，所执行的每一个流水线操作的时序图。根据附图2和附图10，在指令提取级，指令提取单元52在PC单元所容纳的程序计数器指定的地址访问所述指令存储器40，并将分组发送至指令缓冲区54。在指令分配级，容纳在所述指令缓冲区54中的指令被分配给指令寄存器56a~56c。在解码级，在指令发出控制单元62的控制下，被分配给所述指令寄存器56a~56c的指令分别通过所述指令解码器64a~64c进行解码。在操作级，所述执行控制单元72操作所述执行单元70的组成部分，从而基于在指令解码器64a~64c中解码的结果，执行不同操作。在写入级，所述操作结果被存储在数据存储器100或寄存器文件76中。根据这些处理，能够并行执行3个或更少的流水线处理。

附图11是示出通过处理器30执行的指令、处理的细节和指令的位模式的示图。指令“ld Rs, Rd”表示将数据存储器100中的数据载入至寄存器Rd中的处理,如附图1A~附图1D中所示,其中所述数据由在指令的Rs字段中指定的一寄存器(以下称为“寄存器Rs”)寻址。所述位模式如附图11中所示。

在附图11中所示出的每一个位模式中,前2位(第30和第31位)用于指定一组操作,且第0位用于指定并行执行边界信息。前2位为“01”的操作涉及存储器访问。前2位为“10”的操作涉及乘法。前2位为“11”的操作涉及其他处理。

指令“st Rs, Rd”表示数据存储器100中,将寄存器Rs的值存储至一通过寄存器Rd寻址的位置中的处理。

指令“mul1 Rs, Rd”表示将寄存器Rs的值和寄存器Rd的值的乘积写入寄存器Rd的处理。指令“mul2 Rs1, Rs2, Rd”表示将寄存器Rs1的值和寄存器Rs2的值的乘积写入寄存器Rd的处理。

指令“add1 Rs, Rd”表示将寄存器Rs的值和寄存器Rd的值的和写入寄存器Rd的处理。指令“add2 Rs1, Rs2, Rd”表示将寄存器Rs1的值和寄存器Rs2的值的和写入寄存器Rd的处理。

指令“sub1 Rs, Rd”表示将寄存器Rs的值和寄存器Rd的值的差写入寄存器Rd的处理。指令“sub2 Rs1, Rs2, Rd”表示将寄存器Rs1的值和寄存器Rs2的值的差写入寄存器Rd的处理。

指令“mov1 Rs, Rd”表示将寄存器Rs的值写入寄存器Rd的处理。指令“mov2 Imm, Rd”表示将Imm字段中的值写入寄存器Rd的处理。

指令“div Rs, rd2”表示将以寄存器Rs的值除以寄存器Rd的值得到的商写入寄存器Rd的处理。指令“mod Rs, Rd”表示将以寄存器Rs的值除以寄存器Rd的值得到的余数写入寄存器Rd的处理。

(编译器)

接下来,将参照图12-38,说明当前实施例中以上述处理器30作为目标的编译器的实例。

（编译器的整体结构）

附图12是当前实施例中编译器200的结构的的功能性框图。该编译器200是一交叉编译器，所述编译器将以高级语言（诸如C/C++语言）描述的源程序202翻译为一机器语言程序204，其目标处理器为上述处理器30。所述编译器200通过在一计算机（诸如个人计算机）上执行的程序来实现，且大致由一语法分析单元210、一中间码转换单元220、一优化单元230和一代码产生单元240构成。

所述语法分析单元210是一预处理单元，该单元提取保留字（关键字）等，从而实现源程序202的词法分析（包含将包括的头文件），它是编译的目标，具有普通编译器的分析功能，

所述中间码转换单元220是一处理单元，该单元连接至所述语法分析单元210，并根据一定的规则，将从语法分析单元210通过的源程序202中的每一语句转换为中间码。这里，中间码一般是以函数调用的格式表现的编码（例如，表示“+ (int a, int b)”；表示“将一整数a与一整数b相加”的编码）。

优化单元230包括：一指令调度单元232，该单元连接至所述中间码转换单元220，并把注意力集中在包括在由中间码转换单元220输出的中间码内包括的指令的操作码，安排这些指令以便降低处理器30的功耗，而不改变指令之间的相关性；以及一寄存器分配单元234，该单元连接至所述指令调度单元232，并把注意力集中在由指令调度单元232执行的调度的结果中包括的指令的寄存器字段，分配寄存器以便降低所述处理器30的功耗。

所述优化单元230进一步包括：一指令再调度单元236，该单元连接至寄存器分配单元234，并把注意力集中在分配寄存器的调度结果内包括的指令的位模式，置换指令以便降低处理器30的功耗，而不改变指令之间的相关性；以及一时段停止/恢复指令产生单元238，该单元连接至所述指令再调度单元236，且基于指令再调度单元236中的调度结果，检测停止某些周期或更多周期的间隔的一时段，并插入指令，以在所述间隔之前和之后停

止和恢复所述时段。

所述优化单元230进一步包括：一并行执行边界信息设置单元239，该单元与所述时段停止/恢复指令产生单元238相连接，且基于调度结果，设置关于所安排指令的并行执行边界信息；以及一周期内置换处理单元237，该单元连接至所述指令调度单元232、所述寄存器分配单元234和所述指令再调度单元236，并置换每一周期的调度结果中的指令，以便降低功耗。

应注意的是，下面所要描述的优化单元230中的处理被执行在每一个基本模块的单元中。一基本模块是一程序的单元，诸如一系列等式和赋值语句，其中在其中部并未出现向外的分支，也没有从外部至其中部的分支。

一代码产生单元240连接至优化单元230中的并行执行边界信息设置单元239，并根据代码产生单元240本身中保存的转换表或其他类似的表格，将从所述并行执行边界信息设置单元239输出的所有中间码置换为机器语言指令，以便产生一机器语言程序204。

接下来，将使用具体实例，说明具有如上所述结构的所述编译器200的特有操作。

（指令调度单元）

附图13是指令调度单元232的操作的流程图。所述指令调度单元232并不执行寄存器的调度，而是执行假定存在无穷多的寄存器时的处理。因此，在以下描述中假设“Vr”（虚拟寄存器），诸如“Vr 0”和“Vr 1”，被附加至将由指令调度单元232调度的寄存器的头部。

所述指令调度单元232基于在中间码转换单元220中产生的中间码创建一指令相关性图（步骤S2）（“步骤”在以下被忽略）。相关性图是表示指令之间的相关性的图，即其中将一个节点分配给每一指令的、且相互相关的指令通过边缘连接的有向图。相关性图是一个众所周知的技术，所以将不再重复对其的详细说明。例如，在此创建了附图14A中示出的包括三个有向图的相关性图。

所述指令调度单元232在相关性图中选择可执行的指令（节点），并调

度用于第一周期的指令，以便与每一时段的默认逻辑相匹配（S4）。例如，在附图14A的相关性图中，假定相对应于节点N1、N6、N7、N11和N12的指令能够被调度，且其中，所述节点N1对应于关于存储器访问的指令，所述节点N11对应于乘法指令，且所述节点N6对应于移位指令。在这种情况下，所述节点N1、N11和N6分别位于第一周期的第一～第三时段。将标记附加至已安排的节点，因而，如附图14B中所示更新相关性图。在第一周期的指令调度之后（S4），获得如附图15所示的指令调度的结果。

所述指令调度单元232根据所述相关性图，产生安排候选指令集（S8）。在附图14B的实例中，对应于所述节点N2、N7、N8和N12的指令是安排候选指令集。

所述指令调度单元232根据以下所要描述的算法，从所述安排候选指令集中提取一个最佳指令（S12）。

所述指令调度单元232判断所提取的最佳指令实际上是否能够被安排（S14）。其是否能够被安排，基于包括为目标周期安排的最佳指令的指令的数目，是否不多于为在前周期安排的指令的数目来进行判断。因此，相同数目的指令被连续安排用于随后的周期。

当判断所述最佳指令能够被安排时（在S14中为“是”），所述指令调度单元232临时安排所述最佳指令，并将其从安排候选指令集中删除（S16）。然后，所述指令调度单元232以与上述判断相同的方式（S14），判断是否能够在时段内安排另一指令（S18）。当它判断另一指令能够被安排时（在S18中为“是”），所述指令调度单元232根据所述相关性图，将一个新的安排候选指令（如果有的话）添加至所述安排候选指令集（S20）。重复用于临时为一目标周期安排指令的上述处理，直到所有的安排候选指令均已被安排（S10～S22）。

当判断在临时安排最佳指令的处理（S16）之后、没有更多的指令可在目标周期内安排时（在S18中为“否”），通过所述指令调度单元232执行的处理，从临时指令安排处理的循环中退出（S10～S22）。

在执行所述临时指令安排处理（S10～S22）之后，所述指令调度单元232明确安排所述临时安排的指令，并结束所述安排候选指令集的调度（S24）。然后，将表示“已安排”的标记添加给相关性图中对应于已安排指令的节点，以更新所述相关性图（S26）。

所述指令调度单元232判断是否相同数目的指令在预定数目的周期内被连续安排（S27）。当判断相同数目的指令被连续安排在预定数目的周期内时（例如，当两个指令在20个或更多的周期内被连续安排时，或当一个指令在10个或更多的周期内被连续安排时）（在S27中为“是”），所述指令调度单元232将能够在一个周期内被安排的指令的最大数目（以下称为“可安排指令的最大数目”）设置为“3”（S28），以使在之后的周期中，尽可能在一个周期中安排三个指令。重复上述处理，直到所有的指令都被安排（S6～S29）。

附图16是附图13中所述最佳指令提取处理（S12）的操作的流程图。

所述指令调度单元232计算每一个安排候选指令和每一个在目标周期之前的周期内已被安排的指令的操作码的位模式之间的汉明距离（S42）。

例如，在附图14B中，对应于节点N2、N7、N8和N12的指令，能够被安排在第二周期的调度的开始。对应于节点N1、N6和N11的指令在第一周期内已被安排。因此，所述指令调度单元232计算对应于节点N1、N6和N11的指令与对应于节点N2、N7、N8和N12的指令的所有组合的操作码的位模式之间的汉明距离。

附图17A和附图17B是说明如何计算操作码的位模式之间的汉明距离的示意图。假定指令“ld Vr11, Vr12”已安排在第N个周期，且第（N+1）个周期的安排候选指令为“st Vr13, Vr14”和“addl Vr13, Vr14”。如果根据附图17A，比较操作码“ld”和“st”，则第12位、第16位、第17位、第24位和第25位的位模式均互不相同。因此，汉明距离为5。如果参见附图17B以与附图17A所示相同方式，比较操作码“ld”和“addl”，则第16位、第17位、第18位、第20位、第25位、第26位、第28位和第31位的位模式均互

不相同。因此，汉明距离为8。

附图18A和附图18B是说明如何计算具有不同位长的操作码的位模式之间的汉明距离的示意图。假定指令“ld Vr11, Vr12”已安排在第N个周期，且第(N+1)个周期的安排候选指令为“mul2 Vr13, Vr14, Vr15”和“st Vr13, Vr14”。如果操作码的位长与附图18A中的操作码“ld”和“mul2”一样是不同的，则计算所述操作码的重叠部分的位模式之间的汉明距离。因此，基于所述操作码的第16~31位的值，计算汉明距离。在操作码“ld”和“mul2”之间，第16位、第18位、第19位、第22位、第23位、第25位、第26位、第27位、第28位、第30位和第31位的位模式均互不相同。因此，汉明距离为11。对于其他安排候选指令“st Vr13, Vr14”的汉明距离是基于附图18B中的操作码的第16~31位的值来计算，以便确保与附图18A中实例的一致性。在操作码“ld”和“st”之间，第16位、第17位、第24位和第25位的位模式均互不相同。因此，汉明距离为4。

返回附图16，所述指令调度单元232指定具有最小汉明距离的安排候选指令(S43)。在附图17A~附图18B的实例中，指定指令“st Vr13, Vr14”。

指令调度单元232判断是否存在两个或更多个具有最小汉明距离的安排候选指令(S44)。当存在一个具有最小汉明距离的安排候选指令时(在S44中为“否”)，则指定该指令作为最佳指令(S56)。

当存在两个或更多个具有最小汉明距离的安排候选指令时(在S44中为“是”)，所述指令调度单元232判断所述安排候选指令的任意一个是否与其中未安排指令的空闲时段的默认逻辑相匹配(S46)。

如果没有安排候选指令与所述默认逻辑相匹配(在S46中为“否”)，则选择具有最小汉明距离的两个或更多个安排候选指令中的任意一个，作为一最佳指令(S54)。

如果任意一个安排候选指令与默认逻辑相匹配，且这样的指令的数目为1(S46中为“是”且S48中为“否”)，则指定这个安排候选指令作为最佳指令(S52)。

如果任意一个安排候选指令与默认逻辑相匹配, 且这样的指令的数目为2或更多(S46中为“是”且S48中为“是”), 则选择与所述默认逻辑相匹配的两个或更多个安排候选指令中的任意一个, 作为最佳指令(S50)。

(周期内置换处理单元)

附图19是周期内置换处理单元237的操作的流程图。所述周期内置换处理单元237基于指令调度单元232中的调度结果, 调整每一个周期内的指令的安排。

所述周期内置换处理单元237置换调度结果中从第二个到最后一个周期中的目标周期的三个指令, 以便产生指令序列的六个模式(S61)。附图20A~附图20F是如上所述产生的指令序列的六个模式的示例的示意图。

所述周期内置换处理单元237执行计算汉明距离的和处理, 用于下面将描述的指令序列的六个模式中的每一个(S62~S67)。所述周期内置换处理单元237从指令序列的六个模式所计算的汉明距离的和, 选择具有汉明距离的最小和的指令序列, 并置换这些指令, 以便实现与所选择指令序列相同的安排(S68)。对第二个到最后一个周期重复上述处理(S60~S69)。

接下来, 将说明对于计算指令序列的六个模式中每一个的汉明距离的和处理(S62~S67)。对于每一指令序列的每一时段, 所述周期内置换处理单元237计算目标周期的指令和在前周期的指令的操作码的位模式之间的汉明距离(S64)。所述周期内置换处理单元237对三个时段内的所有指令(S63~S65), 执行计算汉明距离的处理(S64), 并计算在上述三个时段内的指令之间的汉明距离的和(S66)。对所有指令序列的六个模式, 执行上述处理(S62~S67)。

附图21是已安排指令的实例的示意图。假定指令“ld Vr10, Vr11”, “subl Vr12, Vr13”和“addl Vr14, Vr15”分别作为在第一、第二和第三时段中执行的指令, 安排给第N个周期。还假定指令“st Vr16, Vr17”, “mul Vr18, Vr19”和“mod Vr20, Vr21”分别作为将在第一、第二和第三时段中执行的指令, 安排给第N+1个周期。

附图22A~附图22F是说明指令序列产生处理的示意图(S61)。例如,应用附图21中所示的安排给第N+1个周期的三个指令,产生在附图22A~附图22F中所示的六个指令序列。

附图23是说明计算操作码之间的汉明距离的处理的示意图(S64)。例如,当计算附图21中第N个周期的指令序列和附图22C中第N+1个周期的指令序列的操作码之间的相应时段的汉明距离时,第一、第二和第三时段中的汉明距离分别为10、9和5。

因此,在附图23的实例中,汉明距离的和为24。在计算汉明距离和的处理中(S66),以与上述方式相同的方式,计算附图21中第N个周期的指令序列和附图22A~附图22F中第N+1个周期的指令序列之间的汉明距离的和,且这些值分别为14、16、24、22、24和20。在选择一指令序列的处理中(S68),从指令序列的六个模式中,选择如附图22A所示的具有汉明距离的最小和的指令序列。

(寄存器分配单元)

附图24是寄存器分配单元234的操作的流程图。所述寄存器分配单元234基于在指令调度单元232中和所述周期内置换处理单元237的调度结果,实际分配寄存器。

所述寄存器分配单元234从源程序202中提取分配对象(变量),且计算每一分配对象的生存期和优先级(S72)。一个生存期是从程序中一个变量的定义到参照该变量的结束的一个时间段。因此,一个变量可以具有多个生存期。基于一个分配对象的生存期长度以及参照该对象的频率,确定其优先级。其详细说明在此将不再重复,因为它并不是本发明的必要部分。

所述寄存器分配单元234基于分配对象,产生一冲突图(S74)。该冲突图是表示在不能分配同样的寄存器的分配对象的状态的图。接下来,将说明如何产生一冲突图。

附图25示出是分配对象的变量的生存期的示意图。该例中,三个变量I, J和K是分配对象。

在步骤T1中定义变量I，且最终在步骤T5中涉及该变量I。在步骤T8中再次定义变量I，且最终在步骤T10中涉及该变量。因此，变量I具有两个生存期。该变量I在前一个生存期中，被定义为变量I1，且在后一个生存期中，被定义为变量I2。在步骤T2中定义变量J，且最终在步骤T4中涉及该变量。

在步骤T3中定义变量K，且最终在步骤T6中涉及该变量。在步骤T7中再次定义变量K，且最终在步骤T9中涉及该变量。因此，所述变量K具有与变量I类似的两个生存期。该变量K在前一个生存期中被定义为变量K1，且在后一个生存期中被定义为变量K2。

所述变量I1、I2、J、K1和K2的生存期具有以下重叠区。所述变量I1和J的生存期在步骤T2~T4中重叠。变量J和K1的生存期在步骤T3~T4中重叠。变量I1和K1的生存期在步骤T3~T5中重叠。变量I2和K2的生存期在步骤T8~T9中重叠。如果变量的生存期重叠，则它们将不能分配给同一寄存器。因此，在冲突图中，作为分配对象的变量是节点，其生存期重叠的变量通过边缘连接。

附图26是基于附图25中实例产生的变量的冲突图的示意图。节点I1、K1和J通过边缘彼此连接。在变量I1、K1和J的生存期中存在重叠，因而发现同一寄存器不能分配给这三个变量。节点I2和K2以同样的方式通过一条边来连接。因而，发现同一寄存器不能分配给变量I2和K2。

然而，在未通过一条边连接的节点之间不存在相关性。例如，节点J和K2未通过一条边连接。因此，在变量J和K2之间不存在重叠区，且因而发现不能将同一寄存器分配给它们。

返回附图24，所述寄存器分配单元234在没有给其分配寄存器的分配对象中，选择具有最高优先级的分配对象（S80）。所述指令调度单元232判断一个寄存器是否能够被分配给分配对象（S82），其中该寄存器在涉及分配对象的指令之前的同一时段内将执行的指令的同一字段中，具有与寄存器数目相同的数目。该判断是参照上述冲突图做出的。

附图27A~附图27C是表示在指令调度处理中获得的结果的示意图。例

如，参照附图27A，假定在N+1周期的第一时段中，当前分配对象被分配给一源操作数（寄存器Vr5）。如上所述，寄存器Vr5是临时设置的。因此，在判断寄存器可分配性的处理中（S82），判断一分配对象是否能够在第N周期分配给同一字段中所使用的寄存器（在该例中为寄存器R0）。附图27B示出了在将寄存器R0分配给Vr5的情况下指令的位模式。该图也示出了由于在连续的周期中，通过访问同一寄存器的寄存器特性，从而可以降低功耗。

当判断具有相同数目的寄存器能够被分配时（在S82中为“是”），所述寄存器分配单元234将具有相同序号的上述寄存器分配给分配对象（S84）。当判断具有相同序号的寄存器不能被分配时（在S82中为“否”），所述寄存器分配单元234从可分配寄存器的寄存器序号（二进制表示）中指定若干个寄存器，这些寄存器具有与在前周期中同一时段的相同字段中的寄存器序号之间有最小汉明距离的寄存器序号（S86）。附图27C示出了一个示例，其中从可分配寄存器中选择的寄存器R1，寄存器R1具有与寄存器R0的寄存器序号（00000）之间有最小汉明距离的寄存器序号（00001）。

其中当仅存在一个具有最小汉明距离的可分配寄存器时（在S88中为“否”），该寄存器被分配给分配对象（S92）。当存在两个或更多个具有最小汉明距离的可分配寄存器时（在S88中为“是”），选择两个或更多个寄存器中的任意一个并将其分配给分配对象（S90）。重复上述处理，直到没有更多的分配对象为止（S78～S94）。

在寄存器分配单元234中的处理之后，所述周期内置换处理单元237基于通过寄存器分配单元234的调度结果，调整每一个周期中指令的安排。在周期内置换处理单元237中所执行的处理，与根据附图19和附图20A～附图20F所说明的处理相同。因此，其详细说明在此不再重复。

（指令再调度单元）

附图28示出了指令再调度单元236的操作的流程图。所述指令再调度单元236执行重新调度已被调度的指令的安排结果的处理，以便根据指令调度单元232、寄存器分配单元234和周期内置换处理单元237所执行的处理，在

处理器30中可操作。换句话说,所述指令再调度单元236重新调度已由寄存器分配单元234确切分配给寄存器的指令序列。

指令再调度单元236从调度结果中删除冗余指令。例如,由于指令“movl R0, R0”是一条冗余指令,因为它是用于将寄存器R0的内容写入寄存器R0的指令。当同一周期第一时段中一指令为“mov2 4, R1”,且同一周期第二时段中一指令为“mov2 5, R1”时,它们分别是将4和5写入寄存器R1的指令。在当前实施例中,在时段中数目较大的指令以较高的优先级执行。因此,在第一时段中的指令“mov2 4, R1”是一条冗余指令。

如果删除一个冗余指令,则指令之间的相关性也可改变。因此,所述指令再调度单元236重构一相关性图(S114)。所述指令再调度单元236在所述相关性图中选择可执行的指令(节点),并在第一周期调度它们,以便匹配每一时段中的默认逻辑(S115)。将表示“已安排”的标记添加至所述相关性图中对应于第一周期的指令的节点。

指令再调度单元236参照所述相关性图产生一安排候选指令集(S118)。所述指令再调度单元236根据下述算法从安排周期指令集中取一条最佳指令(S122)。

指令再调度单元236判断所取的最佳指令是否能够被实际安排(S124)。该判断与附图13的S14中的判断相同。因此,其详细说明在此将不再重复。

当指令再调度单元236判断能够安排最佳指令时(在S124中为“是”),则它临时安排该指令,并将其从安排候选指令集中删除(S126)。然后,所述指令再调度单元236判断是否能够以与安排的上述判断相同的方式(S124)安排其他指令(S128)。当其判断能够安排其他指令时(在S128中为“是”),则其根据所述相关性图来判断是否存在一条新的安排候选指令,将其添加至该安排候选指令集(如果有的话)(S130)。重复上述处理,直到不存在更多的安排候选指令为止(S120~S132)。

应注意的是,当在临时安排最佳指令的处理(S126)之后,判断没有更多的指令能够被安排给目标周期时(在S128中为“否”),指令再调度

单元236的处理从临时安排最佳指令的处理的循环中退出（S120～S132）。

在临时安排最佳指令的处理之后（S120～S132），所述指令再调度单元236确切安排所述临时安排的指令，并结束安排候选指令集的调度（S134）。然后，将表示“已安排”的标记添加至对应于相关性图中已安排指令的节点，以便更新所述相关性图（S136）。

指令再调度单元236判断相同数目的指令是否连续安排给预定周期（S137）。当判断相同数目的指令被连续安排给预定数目的周期时（在S137中为“是”），所述指令再调度单元236将可安排指令的最大数目设置为3（S138），从而尽可能将三条指令安排给一个周期。重复上述处理，直到不再留下有未安排的指令（S116～S139）。

附图29是附图28中最佳指令提取处理（S122）的操作的流程图。比较所述安排候选指令中目标周期的指令和在前周期的同一时段中执行的指令，所述指令再调度单元236取得具有相同寄存器数目的字段的数目，并指定一条具有同一寄存器数目的最大数目的字段的安排候选指令（S152）。

附图30A和附图30B是说明指定安排候选指令的处理（S152）的示意图。假定安排指令“addl R0, R2”作为第N周期内第一时段中执行的一条指令，且存在可被安排给第N+1周期的第一时段中的指令，如附图30A中所示“subl R0, R1”和附图30B中所示“div R0, R2”。当将指令“subl R0, R1”安排给附图30A中所示的安排位置时，具有相同寄存器序号的字段，仅是其中安排寄存器R0的字段（寄存器序号00000）。因此，具有同一寄存器数目的字段的数目为1。当将指令“div R0, R2”安排给附图30B中所示的安排位置时，其中分别安排寄存器R0（寄存器序号为000000）和寄存器R2（寄存器序号00010）的两个字段具有相同的寄存器数目。因此，具有相同寄存器序号的字段的数目为1。

当仅存在具有最大数目的这样的字段的一个安排候选指令时（在S154中为“否”），指定这个安排候选指令作为一条最佳指令（S174）。

当不存在具有最大数目的这样的字段的安排候选指令、或存在两个或

更多个这样的指令时（在S154中为“是”），所述指令再调度单元236将在前周期同一时段中将要执行的指令与每一条所述安排候选指令进行比较，以便获得上述两指令的位模式之间具有最小汉明距离的指令（S156）。

附图31A和附图31B是说明指定安排候选指令的处理的示意图（S156）。假定安排指令“mul1 R3, R10”作为第N周期第一时段中将执行的一条指令，且存在可被安排给第N+1周期的第一时段中的指令，如附图31A中所示“add1 R2, R4”和附图31B中所示“sub2 R11, R0, R2”。图中还示出了这些指令的位模式。当将指令“add1 R2, R4”安排在附图31A中所示的安排位置中时，与指令“mul1 R3, R10”的汉明距离为10。当将指令“sub2 R11, R0, R2”安排在附图31B中所示的安排位置中时，与指令“mul1 R3, R10”的汉明距离为8。因此，将指令“sub2 R11, R0, R2”指定为一条安排候选指令。

当仅存在一条具有最小汉明距离的安排候选指令时（在S158中为“否”），这个安排候选指令被指定为一条最佳指令（S172）。

当存在两个或更多个具有最小汉明距离的安排候选指令时（在S158中为“是”），两个或更多个安排候选指令之一条指令与其中执行该安排候选指令的时段的默认逻辑相匹配（S160）。

图32A和图32B是说明指定安排候选指令的处理的示意图（S160）。假定安排指令“st R1, R13”作为第N周期第一时段中将执行的一条指令，且存在可被安排给第N+1周期的第一时段中的指令，如附图32A中所示指令“ld R30, R18”和附图32B中所示的指令“sub1 R8, R2”。图中还示出了这些指令的位模式。如上所述，第一时段的默认逻辑，是关于存储器访问的指令。这可以从所述指令的前2位“01”中得到。由于指令“ld R30, R18”的前2位为“01”，则其与第一时段的默认逻辑相匹配，而由于指令“sub1 R8, R2”的前2位为“11”，则其不与第一时段的默认逻辑相匹配。因此，将指令“ld R30, R18”指定为一条安排候选指令。

当不存在与默认逻辑相匹配的安排候选指令时（在S162中为“否”），选择具有最小汉明距离的安排候选指令的任意一条作为一条最佳指令（S170）。

当存在一个与默认逻辑相匹配的安排候选指令，且这种指令的数目为1时（在 S162中为“是”且 S164中为“否”），指定与所述默认逻辑相匹配的安排候选指令作为一条最佳指令（S168）。

当存在与默认逻辑相匹配的安排候选指令，且上述指令的数目为2或更多时（在S162中为“是”且 S164中为“是”），选择与所述默认逻辑相匹配的这种指令中的任意一条，作为一条最佳指令（S166）。

在指令再调度单元236中的处理之后，所述周期内置换处理单元237基于指令再调度单元236中的调度结果，调整每一周期中的指令的安排。在周期内置换处理单元237中执行的处理，与参照附图19和附图20A～附图20F说明的处理相同。因此，其详细说明将不在此重复。

这就是指令再调度单元236的操作的说明。根据源程序中描述的编译或编译指示的选项，可以限定一个周期中使用的时段的数目。一个“编译指示”给出一编译器的最优化而不改变程序意义的原则的描述。

例如，如以下第一实例中所示，“-para”被设置为以C语言描述的源程序的编译的一个选项，且通过如下数目定义时段的数目。在第一实例中，通过C编译器编译源程序“foo.c”，且在调度结果中，一直将两个指令安排给每一个周期。

同时，如同第二实例中所示，通过一编译指示可以定义源程序中描述的用于每一个函数的时段的数目。在第二实例中，将用于执行一函数func的时段的数目定义为1。因此，一直只将一条指令安排给调度结果中执行函数func的每一个周期。

（第一实例）

```
cc -para 2 foo.c
```

（第二实例）

```
#pragma para=1 func
int func(void) {
.....
```

```
}
```

应注意的是，当同时设置一个选项和一个编译指示时，通过优先级选择具有较小指定值的任一个。例如，当第二实施例所示出的函数func及其“编译指示”被指定在第一实施例所示的源程序“foo.c”中时，通常并行执行在两个时段中的处理，但产生一个调度结果，以使在执行所述函数func的周期中执行在一时段中的处理。

此外，可以基于指令再调度单元236的操作和指令调度单元232或寄存器分配单元234的操作，考虑选项和编译指示。

（时段停止/恢复指令产生单元）

附图33是时段停止/恢复指令产生单元238的操作的流程图。所述时段停止/恢复指令产生单元238基于指令再调度单元236中的调度结果，检测一间隔，该间隔中，只有一个指定时段在一个预定数目（例如4个周期）或更多个周期中连续使用（S182）。时段停止/恢复指令产生单元238插入一条指令，以在紧接上述间隔之前的周期中的一个空闲时段位置停止剩下的两个时段（S184）。当不存在在先前周期中插入指令的空闲时段位置时，将添加一个周期，以插入上述指令。

接下来，所述时段停止/恢复指令产生单元238插入一条指令，从而恢复在紧接上述间隔之后的周期的空闲时段位置被停止的两个时段（S186）。当在随后的周期中，不存在插入指令的空闲时段位置时，将添加一个周期，以插入上述指令。

附图34是其中安排指令的调度结果的实例的示意图。在从第10周期到第18周期的9个周期中，只有第一时段被连续使用。因此，在第9周期的空闲时段中，写入仅操作第一时段并停止剩下两个时段的指令（“set11”）。且在第19周期的空闲时段中，写入恢复剩下两个时段的指令（“clear11”）。附图35是其中基于附图33中连续使用的指定的仅仅一个时段的情况的处理、写入上述指令的调度结果（S182～S186）的实例的框图。

返回附图33，时段停止/恢复指令产生单元238基于调度结果，检测一

间隔, 该间隔中, 只有两个指定时段在一个预定数目 (例如4个周期) 或更多个周期中被连续使用 (S188)。所述时段停止/恢复指令产生单元238插入一条指令, 从而在上述间隔之前的周期中的一个空闲时段位置停止剩下的一个时段 (S190)。当不存在在之前的周期中插入指令的空闲时段位置时, 将添加一个周期, 以插入上述指令。

接下来, 时段停止/恢复指令产生单元238在上述间隔之后的一个空闲时段位置插入一条指令, 从而恢复被停止的一个时段 (S192)。不存在在之后的周期中插入指令的空闲时段位置时, 将添加一个周期, 以插入上述指令。

附图35的调度结果中, 在从第4周期到第8周期的5个周期中, 仅使用了第一时段和第二时段, 而并未使用第三时段。因此, 分别在在前周期和之后的周期中, 存在插入停止第三时段的指令 (“set2 12”) 以及将其恢复的指令 (“clear2 12”) 的需要。然而, 在第3和第9周期的所有时段中已安排指令。因此, 时段停止/恢复指令产生单元238在第4周期之前和第8周期之后插入新的周期, 以便写入上述两个指令。附图36是其中基于附图33中仅仅指定连续使用的指定两个时段的情况的处理 (S188~S192)、写入指令的调度结果的实例的框图。

在当前实施例中, 假定以第一、第二和第三时段的顺序安排指令。因此, 当两个时段在操作中时, 第三时段并不在操作中, 且当只有一个时段在操作中时, 第二和第三时段并不在操作中。

在处理器30中提供一个32位程序状态寄存器 (附图中未示出)。附图37是程序状态寄存器的实例的示意图。例如, 操作中的时段的数目能够使用第15和第16位这2位来表示。该此情况中, 附图37 ((a) ~ (d)) 分别表示操作中的时段的数目为0~3。

附图38是程序状态寄存器的其他实例的示意图。在该程序状态寄存器中, 第14、15和16位分别对应于第一, 第二和第三时段。位的值为 “1” 表示该时段在操作中, 且所述位的值为 “0” 表示该时段被停止。例如, 附图

38 (b) 中所示的程序状态寄存器, 示出了第一时段被停止, 第二和第三时段在操作中。

根据指令“set1”或“set2”, 程序状态寄存器中保存的值被改写。

这就是说当前实施例中所描述编译器的说明, 但编译器200中的每一单元将做如下修改。接下来, 将逐一描述其变体。

(在编译器中的每一单元的变体)

(指令再调度单元236的操作的变体)

在当前实施例中, 将根据附图28和附图29, 已描述指令再调度单元236的操作。然而, 可以执行附图39中所示的提取一条最佳指令的处理, 而不是通过附图29中所示的提取一条最佳指令的处理(S122)。

附图39是附图28中取一条提取最佳指令的处理(S122)的其他操作的流程图。

指令再调度单元236通过以下方法来计算最小汉明距离, 以替代附图29中计算最小汉明距离的处理(S156)。更具体的说, 指令再调度单元236比较在前周期中同一时段执行的一条指令和每一条安排候选指令之间的寄存器字段中的位模式, 以便获得具有最小汉明距离的指令(S212)。

附图40A和附图40B是说明指定安排候选指令的处理(S212)的示意图。假定将指令“add1 R0, R2”安排为第N周期第一时段中执行的指令, 且将附图40A中所示的指令“sub1 R3, R1”和附图40B中所示的指令“div R7, R1”, 安排为可被安排在第N+1周期第一时段中的指令。这些图中还示出了这些指令的位模式。当将指令“sub1 R3, R1”安排在附图40A中示出的上述安排位置时, 该指令和指令“add1 R0, R2”的寄存器字段之间的汉明距离为4。当将指令“div R7, R1”安排在附图40B中示出的上述安排位置时, 该指令和指令“add1 R0, R2”的寄存器字段之间的汉明距离为5。因此, 指定指令“add1 R0, R2”作为一条安排候选指令。

其他处理(S152~S154以及S158~S174)与根据附图29说明的处理相同。因此, 其详细说明在此将不再重复。

（周期内置换处理单元237的第一变体）

周期内置换处理单元237可以执行附图41中所示处理，以替代参照附图19所说明的处理。

附图41是周期内置换处理单元237的操作的第一变体的流程图。

所述周期内置换处理单元237通过以下方法计算最小汉明距离，以替代附图19中所示的计算汉明距离的处理（S64）。更具体地说，周期内置换处理单元237计算一目标指令和每一指令序列中每一时段的在前周期中的指令的位模式之间的汉明距离（S222）。其他处理（S60～S63以及 S65～S69）与根据附图19说明的处理相同。因此，其详细说明在此将不再重复。

附图42是说明计算指令间汉明距离的处理的示意图（S222）。例如，当计算附图21所示第N周期中一个指令序列、与附图22C所示第N+1周期中指令序列的每一时段中的指令之间的汉明距离时，第一、第二和第三时段中的汉明距离分别为12、11和11。

因此，附图42的实例中，汉明距离的和为34。在计算汉明距离和的处理（S66）中，以上述方式，计算附图21中所示第N周期中指令序列中的指令，与附图22A～附图22F中所示的指令序列的6个模式间的汉明距离和，且所计算的和分别为28、26、34、28、34和30。在选择一指令序列的处理（S68）中，从指令序列的6个模式中，选择附图22B所示、具有最小汉明距离和的指令序列。

还应注意的是，假定在当前变体中计算汉明距离的处理（S222）中，寄存器已被分配。因此，在其中尚未分配寄存器的指令调度单元232中的处理之后，不能执行当前变体中的周期内置换处理单元237的处理，但在寄存器分配单元234中的处理或指令再调度单元236的处理之后执行上述处理。

（周期内置换处理单元237的第二变体）

所述周期内置换处理单元237可以执行附图43中所示处理，以替代根据附图19所说明的处理。

附图43是周期内置换处理单元237的操作的第二变体的流程图。

所述周期内置换处理单元237通过以下方法计算最小汉明距离，以替代附图19中所示的计算汉明距离的处理（S64）。更具体地说，周期内置换处理单元237计算一目标指令和每一指令序列中每一时段的在前周期中的指令的寄存器字段的位模式之间的汉明距离（S232）。其他处理（S60～S63以及S65～S69）与根据附图19说明的处理相同。因此，在此将不再重复其详细说明。

附图44是说明计算寄存器字段之间汉明距离的处理的示意图（S232）。例如，当计算附图21所示第N周期中一个指令序列、与附图22C所示第N+1周期中指令序列中的每一时段中的指令之间的汉明距离时，第一、第二和第三时段中的汉明距离分别为2、2和6。

因此，附图44的实例中，汉明距离的和为10。在计算汉明距离之和的处理（S66）中，以上述方式，计算附图21中所示第N周期中指令序列中的指令、与附图22A～附图22F中所示指令序列的6个模式间的汉明距离和，且所计算的和分别为14、10、10、6、10和10。在选择一指令序列的处理（S68）中，从指令序列的6个模式中，选择附图22D所示的具有最小汉明距离和的指令序列。

还应注意的是，假定在当前变体中计算汉明距离的处理（S232）中，寄存器已被分配。因此，在其中仍未分配寄存器的指令调度单元232中的处理之后，不能执行在当前变体中周期内置换处理单元237的处理，但在寄存器分配单元234中的处理或指令再调度单元236的处理之后执行上述处理。

（周期内置换处理单元237的第三变体）

所述周期内置换处理单元237可以执行附图45中所示处理，以替代根据附图19所说明的处理。

附图45是周期内置换处理单元237的操作的第三变体的流程图。

所述周期内置换处理单元237执行以下处理，以替代附图19中所示的获得汉明距离的处理（S64）。更具体地说，周期内置换处理单元237获得一用于每一指令序列中的每一时段的目标指令的寄存器字段的数目，其具有

与在前周期的一条指令相同的寄存器数目（S242）。

所述周期内置换处理单元237执行以下处理，以替代附图19中所示的获得汉明距离和的处理（S66）。更具体地说，周期内置换处理单元237获得三个时段的指令中，具有相同寄存器数目的寄存器字段的数目的和（S244）。

所述周期内置换处理单元237进一步执行以下处理，以替代附图19中所示的置换指令的处理（S68）。更具体地说，周期内置换处理单元237从在六个指令序列的每一个中获得的寄存器字段数目的和中，选择具有最大数目的匹配寄存器字段的指令序列，且置换这些指令，以便成为与所选择指令序列相同的安排（S246）。其他处理（S60～S63，S65和S67和S69）与根据附图19说明的处理相同。因此，在此将不再重复其详细说明。

附图46是已安排指令的实例的示意图。假定将指令“ld R0, R1”，“subl R2, R3”和“addl R4, R5”分别安排为在第N周期第一、第二和第三时段中执行的指令。还假定将指令“st R5, R8”，“mul R2, R3”和“mod R0, R10”分别安排为在第N+1周期第一、第二和第三时段中执行的指令。

附图47A～附图47F是说明产生指令序列的处理（S61）的示意图。例如，从附图46中示出的安排给第N+1周期的三个指令中，产生如附图47A～附图47F中所示的六个指令序列。

附图48是用于说明计算寄存器字段的数目的处理（S242）的示意图。例如，在每一时段内获得附图47F中示出的第N+1周期中指令序列的寄存器字段的数目（具有与附图46中示出的第N周期中指令序列相同的寄存器数目）。就第一时段而言，匹配寄存器字段的数目为1，因为两个周期的寄存器字段中寄存器R0相互匹配，但是其它寄存器字段中的寄存器不同。就第二时段而言，匹配寄存器字段的数目为2，因为两个周期的寄存器字段中寄存器R2和R3相互匹配。就第三时段而言，匹配寄存器字段的数目为0，因为没有两个寄存器字段共用的寄存器。

因此，附图48的实例中，具有相同寄存器数目的寄存器字段的数目之和为3。在计算寄存器字段的数目的和的处理（S244）中，获得匹配寄存器

字段的数目的和，用于附图46中所示第N周期的指令序列和附图47A～附图47F中所示六个指令序列的每一个。所获得的和为2、0、0、0、1和3。因此，在指令序列选择处理（S246）中，从六个指令序列中，选择附图47F中所示的、具有与匹配寄存器字段的数目的最大和的指令序列。

在当前变体中，在假定已进行寄存器分配的基础上，执行获得寄存器字段的数目的处理（S242）。因此，在其中仍未进行寄存器分配的指令调度单元232中的处理之后，不能执行在当前变体中的周期内置换处理单元237中的处理，但在寄存器分配单元234中的处理或指令再调度单元236中的处理之后执行上述处理。

（周期内置换处理单元237的第四变体）

所述周期内置换处理单元237可以执行以下处理，以替代根据附图19所说明的处理。

附图49是周期内置换处理单元237的操作的第四变体的流程图。

所述周期内置换处理单元237执行以下处理，以替代附图19中所示的获得每一指令序列的汉明距离的和的处理（S63～S66）。更具体地说，周期内置换处理单元237获得匹配一目标指令序列所包括的指令之外的一时段的默认逻辑的指令的数目（S252）。

所述周期内置换处理单元237执行以下处理，以替代附图19中所示的置换指令的处理（S68）。更具体地说，周期内置换处理单元237从为六个指令序列的每一个获得的上述指令数目的和中，选择包括与默认逻辑相匹配的最大数目的指令的指令序列，且置换这些指令，以便成为与所选择指令序列相同（S254）。其他处理（S60～S62，S67和S69）与根据附图19说明的处理相同。因此，在此将不再重复其详细说明。

例如，假定在产生指令序列的处理（S61）中，产生附图47A～附图47F中示出的六个指令序列。如上所述，将通过根据一指令序列中的每一个指令的前2位，可以判断该指令是否与被安排在其中的时段的默认逻辑相匹配。例如，由于在附图47B所示的指令序列中，安排在第一时段的指令的前

2位为“01”，则其与上述时段的默认逻辑相匹配。然而，安排在第二和第三时段中的指令的前2位分别为“11”和“10”，则它们与这些时段的默认逻辑并不匹配。因此，一指令与对应时段的默认逻辑相匹配。以这种方式，在计算指令数目的处理中，分别在六个指令序列中获得与默认逻辑相匹配的指令的数目（S252），且所述数目分别为3、1、1、0、0和1。在选择指令序列的处理（S254）中，从六个指令序列中，选择附图47A示出的、具有与默认逻辑相匹配的最大数目的指令的指令序列。

如上所述，当前实施例中的编译器200允许指令安排的优化，以使用于连续周期的同一时段的指令、操作码和寄存器字段之间的汉明距离变小。因此，存储在处理器的指令寄存器中的值的变化保持较小，并且可以产生一用于使处理器以低功耗运行的机器语言程序。

当前实施例中的编译器200也允许指令安排的优化，以使在同一时段内的相同寄存器字段连续访问同一寄存器。因此，由于连续访问同一寄存器，使用于选择寄存器的控制信号的变化保持较小，并且可以产生一用于使处理器以低功耗工作的机器语言程序。

同时，当前实施例中编译器200允许将指令分配给若干个时段，以使这些指令与这些时段的默认逻辑相匹配。因此，在同一时段内连续执行使用处理器的相同组成部分的指令。因此可以产生一用于使处理器以低功耗工作的机器语言程序。

此外，当前实施例中的编译器200允许停止向一个或若干个空闲时段供电，同时仅仅有一个或两个时段在连续指令周期中使用。因此可以产生一用于使处理器以低功耗工作的机器语言程序。

此外，当前实施例中的编译器200允许使用一编译指示或作为编译的一个选项，指定所要用于执行一程序的时段的数目。因此，可以产生空闲时段，并可以停止向空闲时段供电，并且可以产生一用于使处理器以低功耗工作的机器语言程序。

到现在为止，基于当前实施例已说明依据本发明的编译器，但本发明

并不局限于该实施例。

例如，在提取通过指令再调度单元236执行的一最佳指令的处理（S122）中，如参照附图28和附图29所述，根据具有相同寄存器数目的字段的数目、一目标指令和在其之前执行的指令之间的汉明距离（S156）、以及以该顺序的优先级的时段的默认逻辑（S160），指定最佳指令（S152）。然而，本发明并不局限于该优先级顺序，且可以按照其他顺序的优先级顺序来指定最佳指令。

同样，指定最佳指令所应考虑的各种条件，诸如汉明距离和时段的默认逻辑，并不局限于当前实施例中的条件。总而言之，这样的条件需要进行组合或需要将优先级分配给这些条件，使得当根据本发明的编译器使处理器工作时减少总功耗。不用说，对于由指令调度单元232、寄存器分配单元234、周期内置换处理单元237以及指令再调度单元236执行的处理也是一样的。

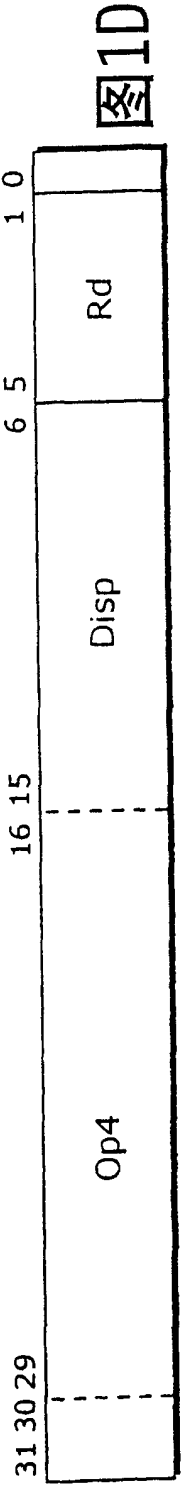
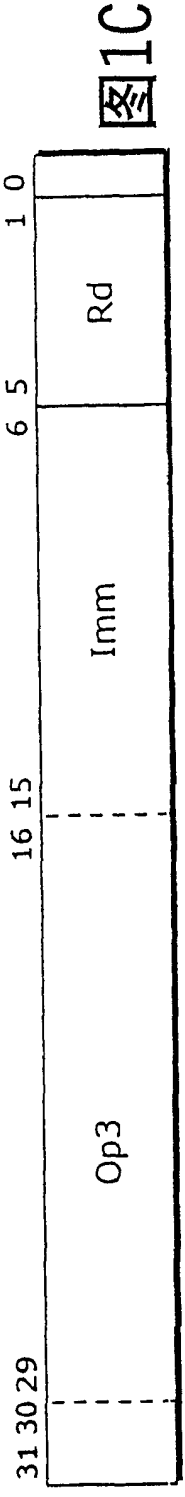
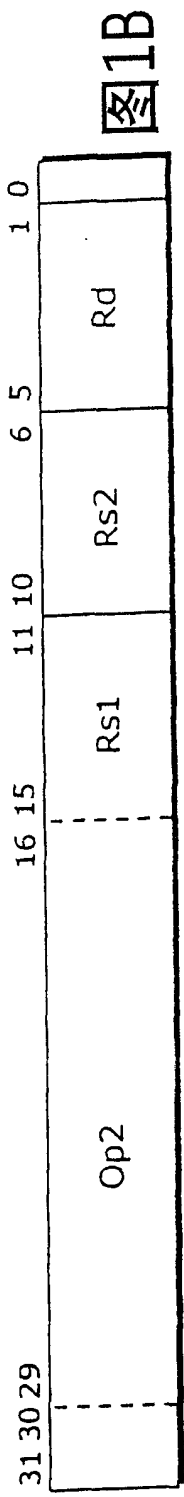
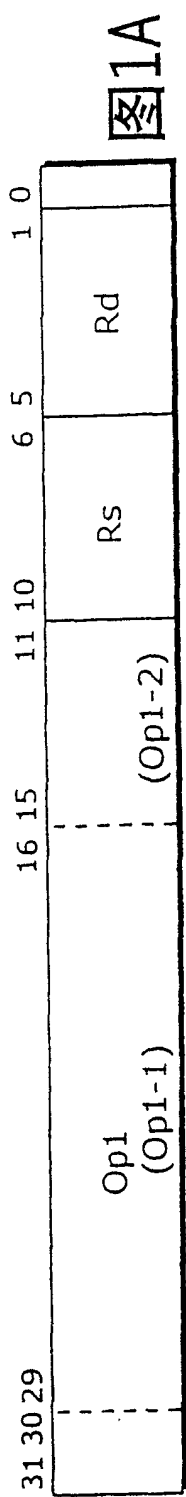
同样，可构造本发明，以使这些条件或优先级的参数化组合被集成为用于编译的源程序202的头文件，或可以指定这些参数作为编译器的一个选项。

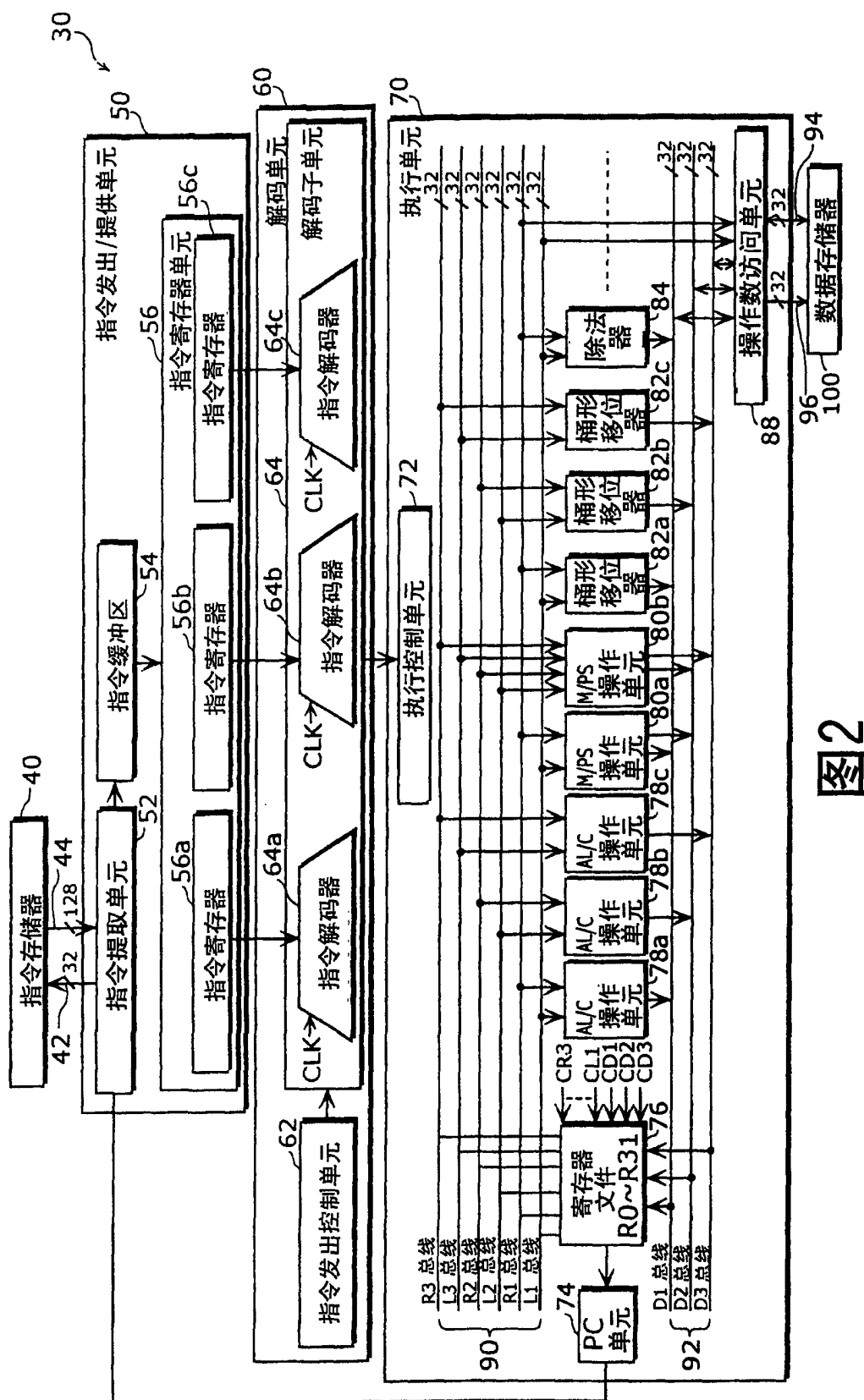
此外，在当前实施例内由优化单元230执行的处理中，可以从多个方法中，为每一基本块选择最佳调度方法。例如，可接受的是，获得用于每一个基本单元的所有多个备用调度方法的调度结果，并选择期望最显著地降低功耗的调度方法，从而得到要降低的功耗。

可以使用诸如反向跟踪之类的方法选择最佳调度方法。例如，当所估计的功耗大于作为寄存器分配单元234的寄存器分配的结果所期望的功耗时，即使在指令调度单元232选择期望最显著降低功耗的调度方法之后，作为试验，指令调度单元232选择期望其次降低功耗的另一调度方法。因此，如果所估计的功耗小于所期望的功耗，则指令再调度单元236可以执行指令再调度处理。

此外，当前实施例中已说明了以C语言描述的一源程序被转换为

一机器语言程序的实例，但该源程序可以以除C语言之外的其他高级语言来描述，或可以是一种已通过其他编译器编译的机器语言程序。当源程序是一机器语言程序时，构造本发明，以便输出通过机器语言程序的优化获得的机器语言程序。





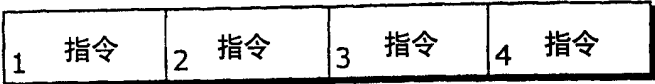


图3

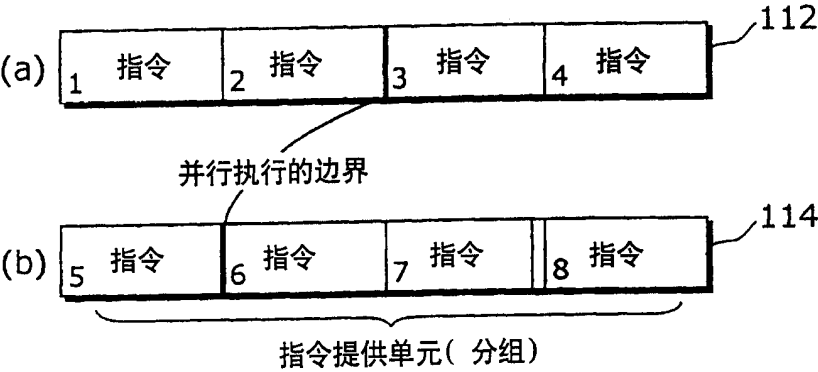


图4

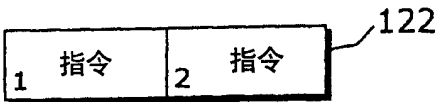


图5A

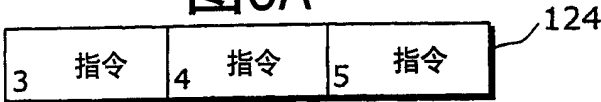


图5B

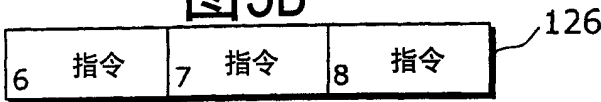


图5C

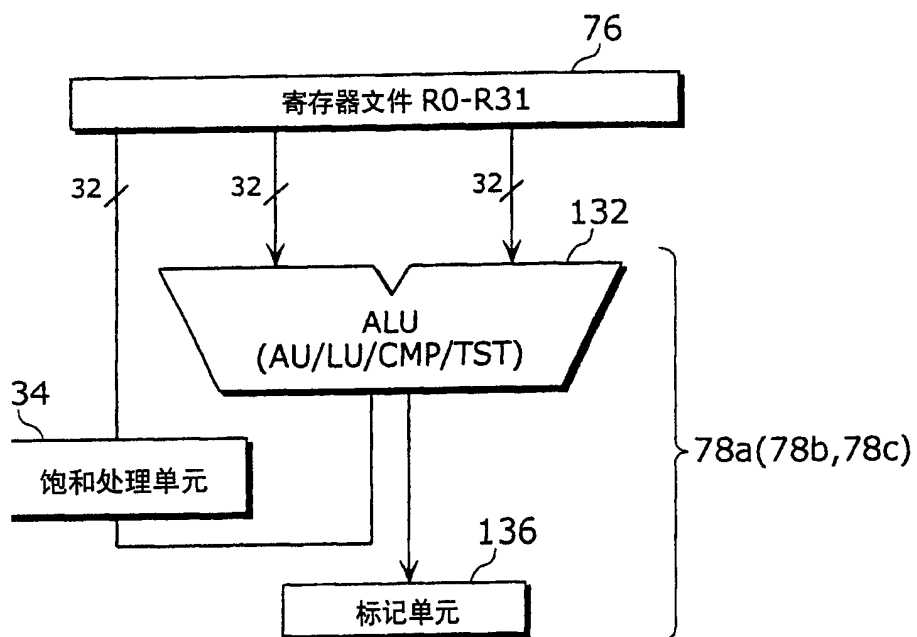


图6

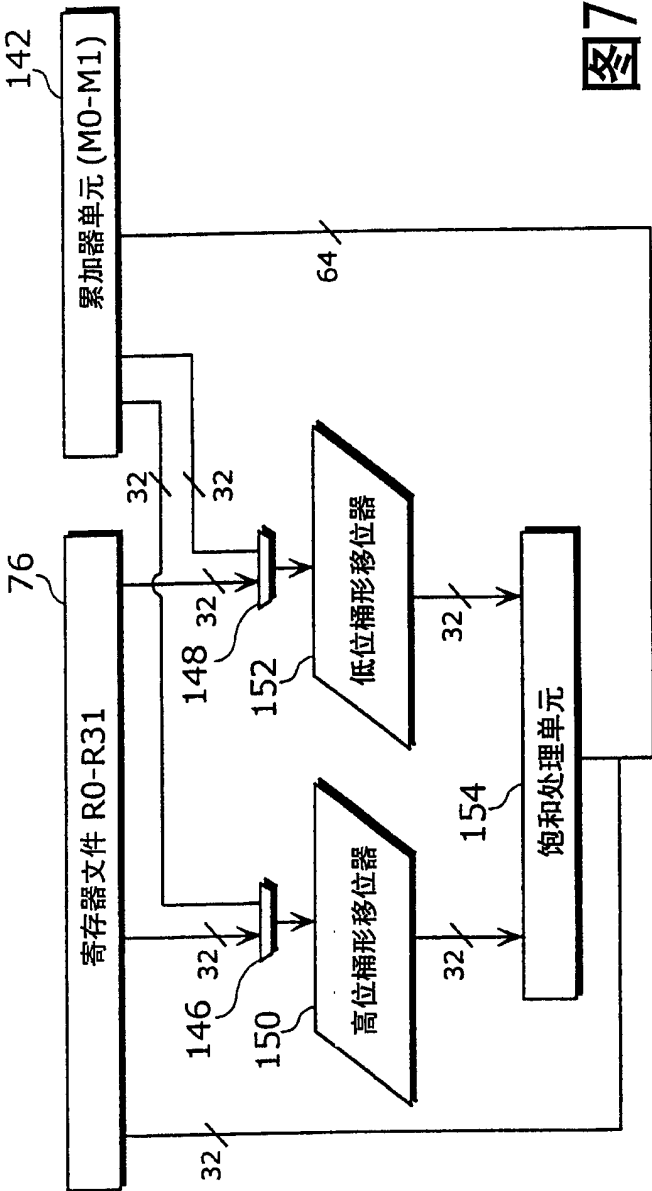


图7

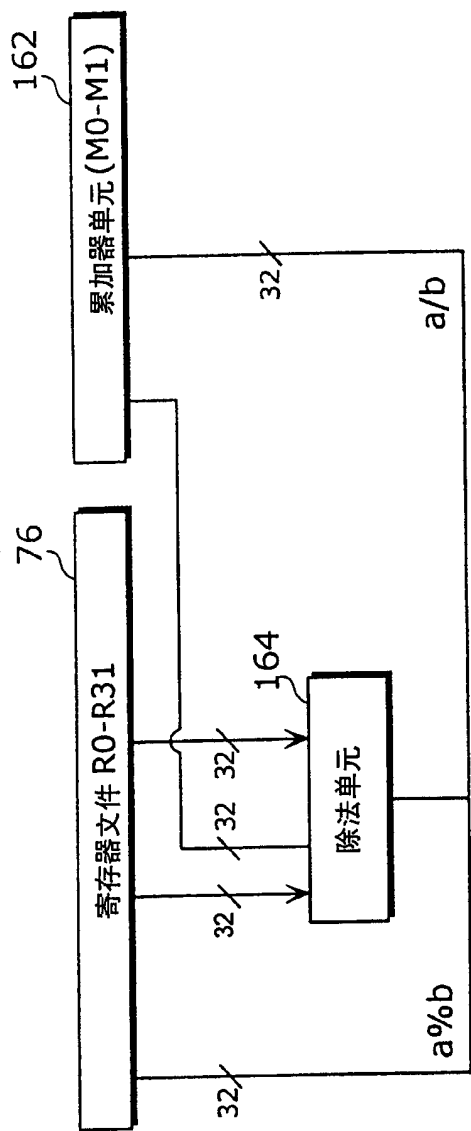


图8

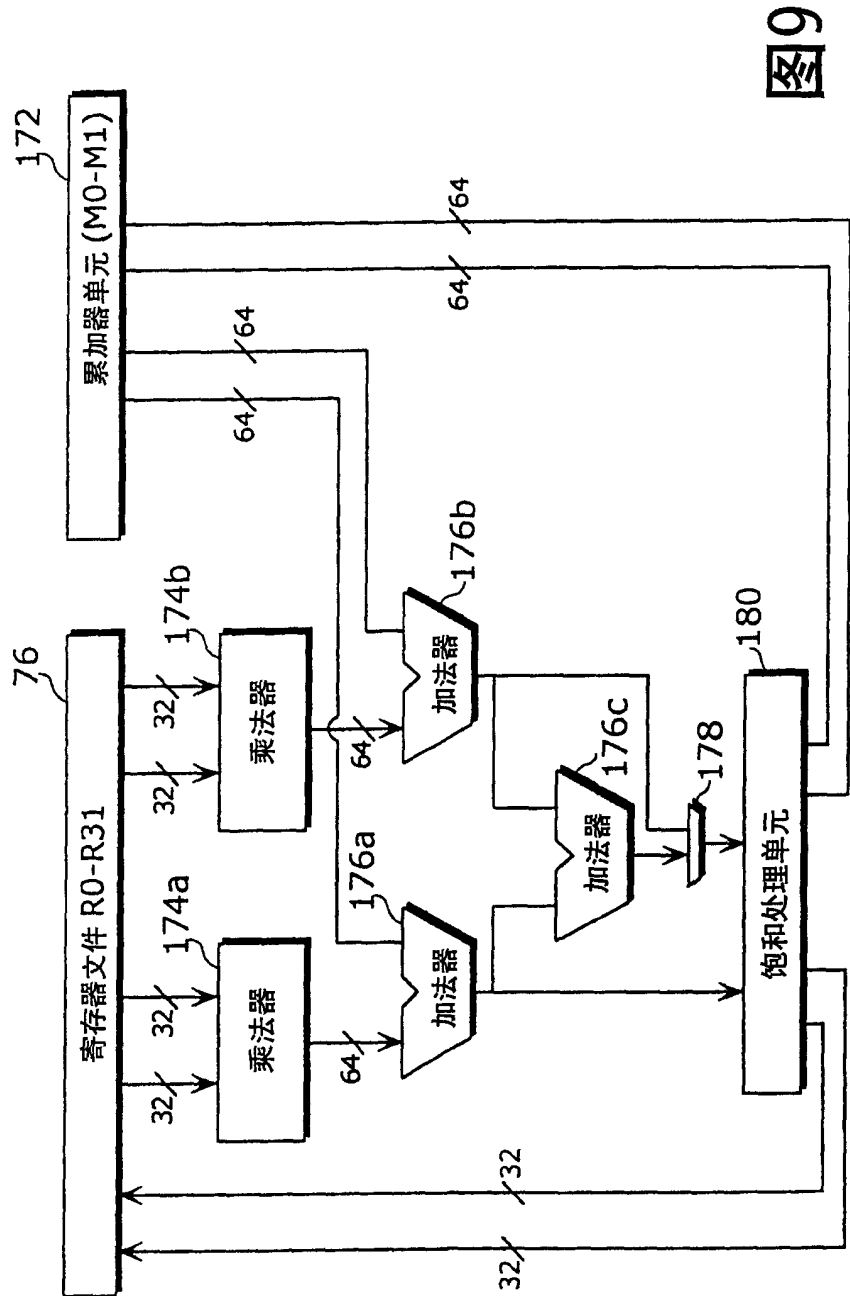


图9

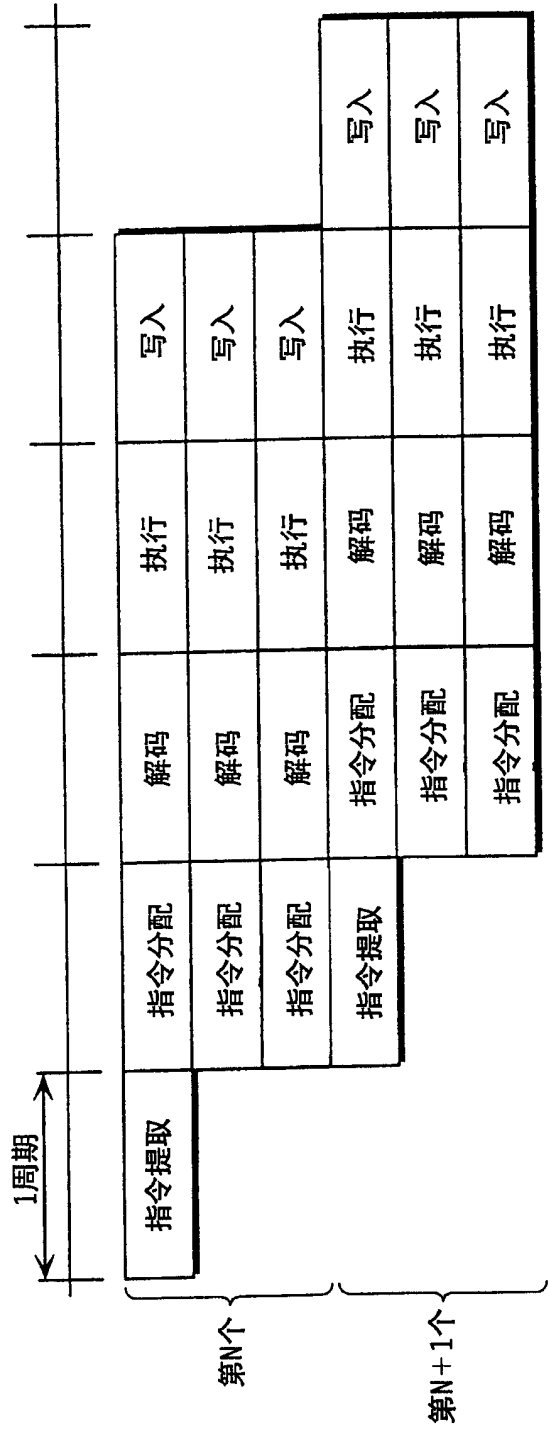


图10

指令	指定处理	31	30	29	16	15	11	10	6	5	1	0
ld Rs, Rd	(Rs)→Rd	0	1	0	1	0	1	0	1	0	0	0
st Rs, Rd	(Rs)→(Rd)	0	1	0	1	0	1	0	1	0	0	0
mul1 Rs, Rd	(Rs)X(Rd)→Rd	1	0	0	0	1	0	0	0	1	0	0
mul2 Rs1, Rs2, Rd	(Rs1)X(Rs2)→Rd	1	0	0	0	1	0	0	1	0	0	0
add1 Rs, Rd	(Rs)+(Rd)→Rd	1	1	0	0	0	0	0	0	1	0	0
add2 Rs1, Rs2, Rd	(Rs1)+(Rs2)→Rd	1	1	0	0	0	0	0	1	0	1	0
sub1 Rs, Rd	(Rs)-(Rd)→Rd	1	1	0	0	0	0	0	1	0	0	0
sub2 Rs1, Rs2, Rd	(Rs1)-(Rs2)→Rd	1	1	0	0	0	0	1	0	0	1	0
mov1 Rs, Rd	(Rs)→Rd	1	1	0	0	0	0	1	0	0	0	0
mov2 Imm, Rd	Imm→Rd	1	1	0	0	0	1	0	0	1	0	0
div Rs, Rd	(Rs)÷(Rd)→Rd	1	1	0	0	1	0	0	1	0	0	0
mod Rs, Rd	(Rs)%(Rd)→Rd	1	1	0	0	1	0	0	1	0	0	0

Op1-1, Op2, Op3, Op4

Op1-2

Op1

并行执行边界信息

图11

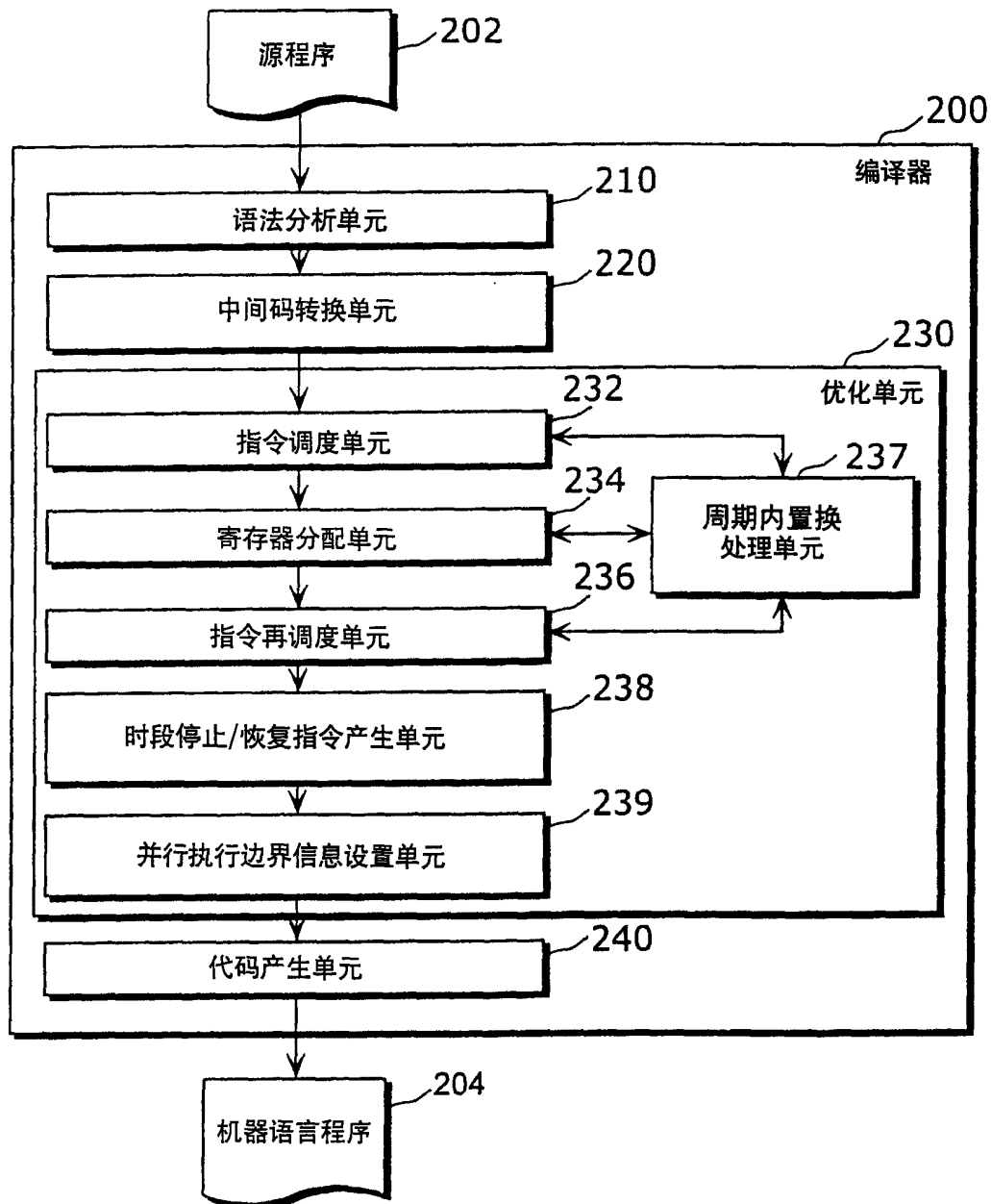


图12

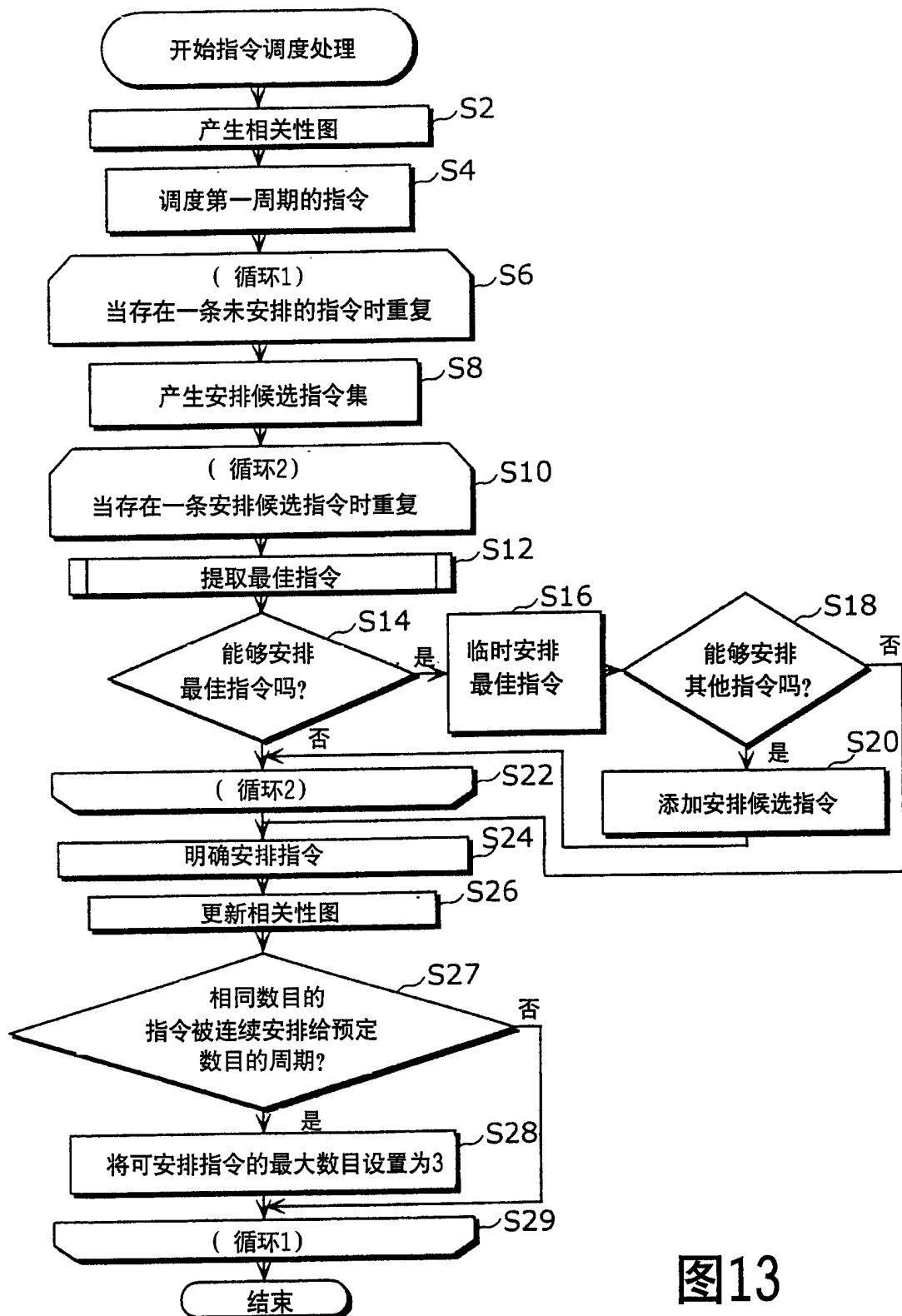


图13

图14A

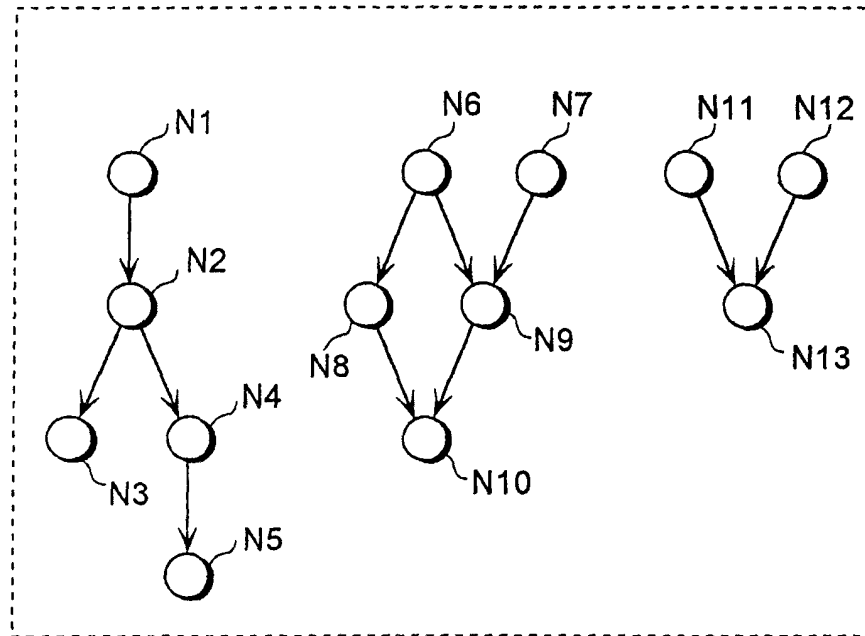
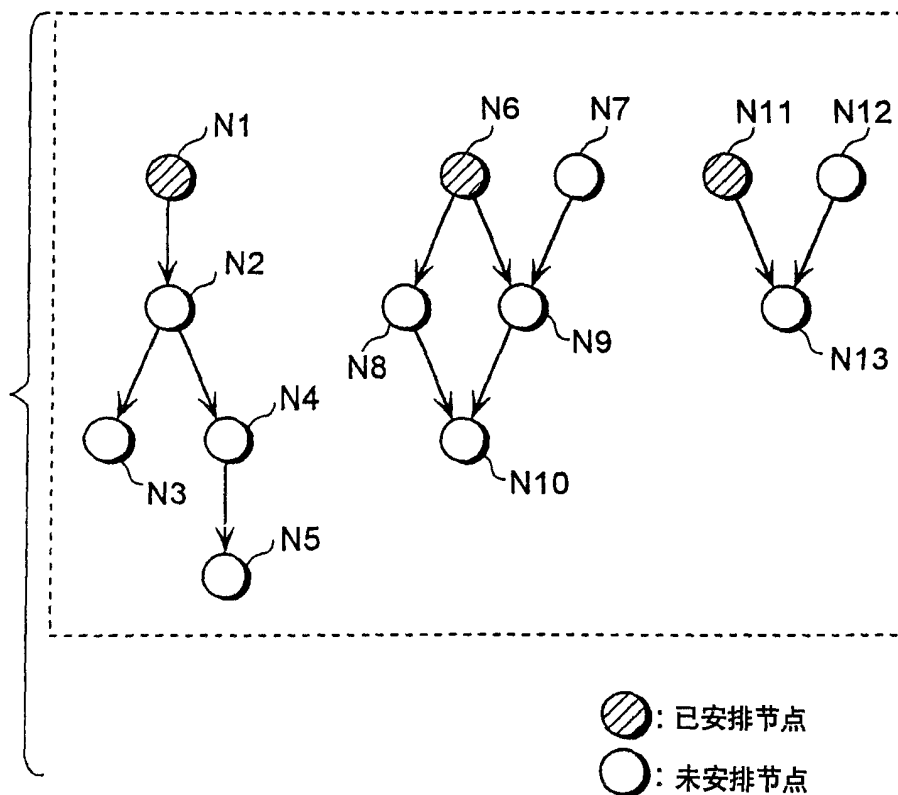


图14B



周期	第一时段	第二时段	第三时段
1	ld Vr0, Vr1 (Node N1)	mul2 Vr2, Vr3, Vr4 (Node N11)	add1 Vr5, Vr6 (Node N6)

图15

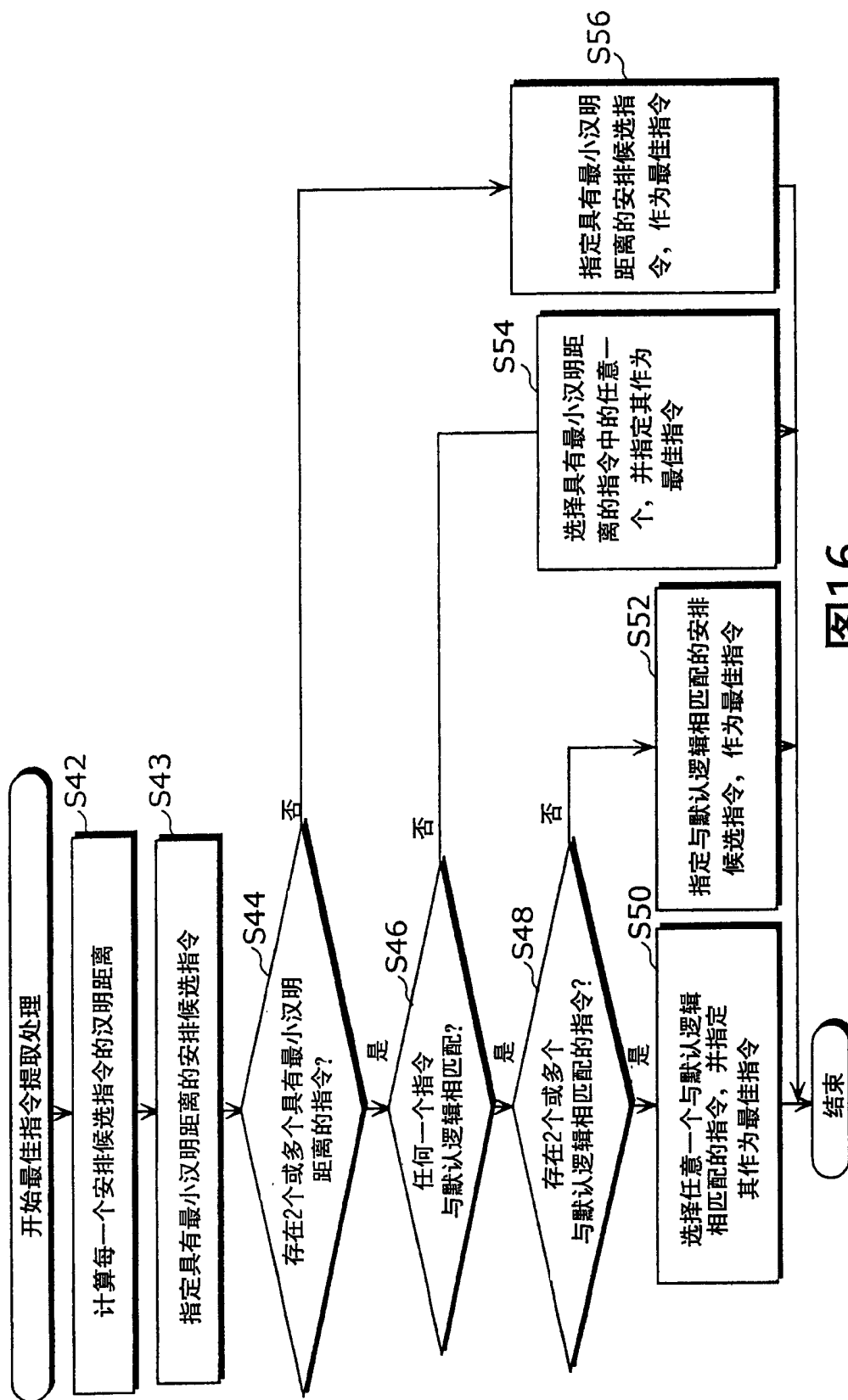
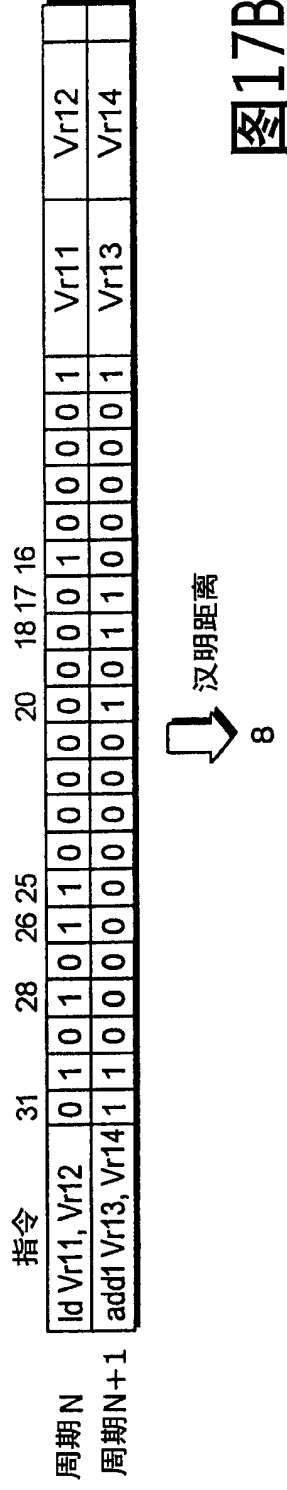
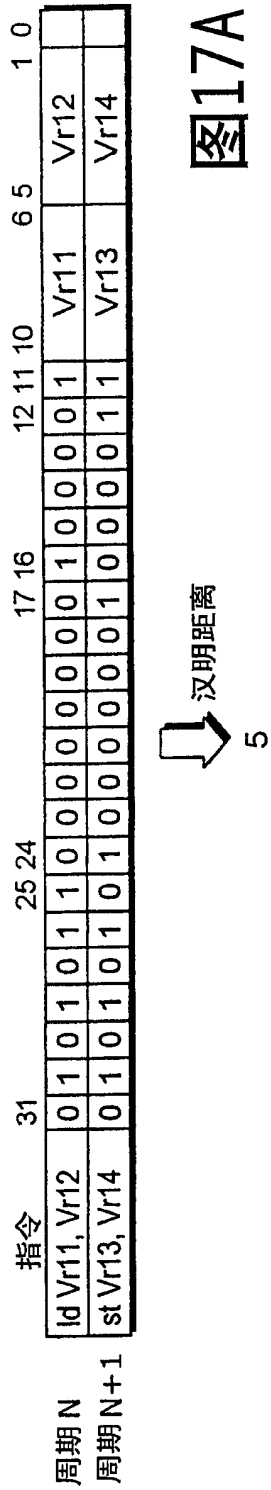
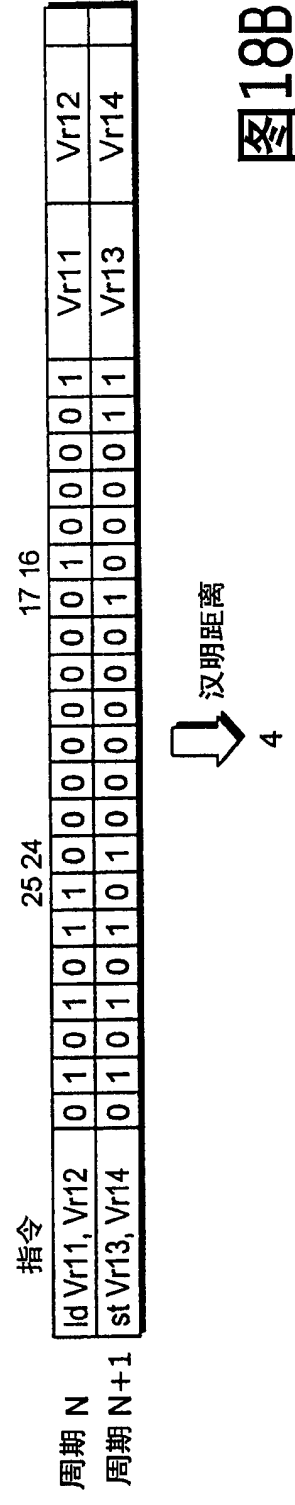
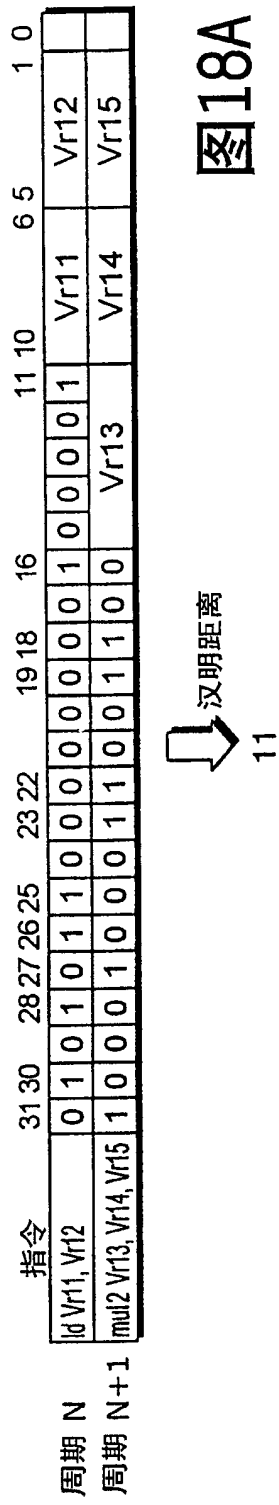


图16





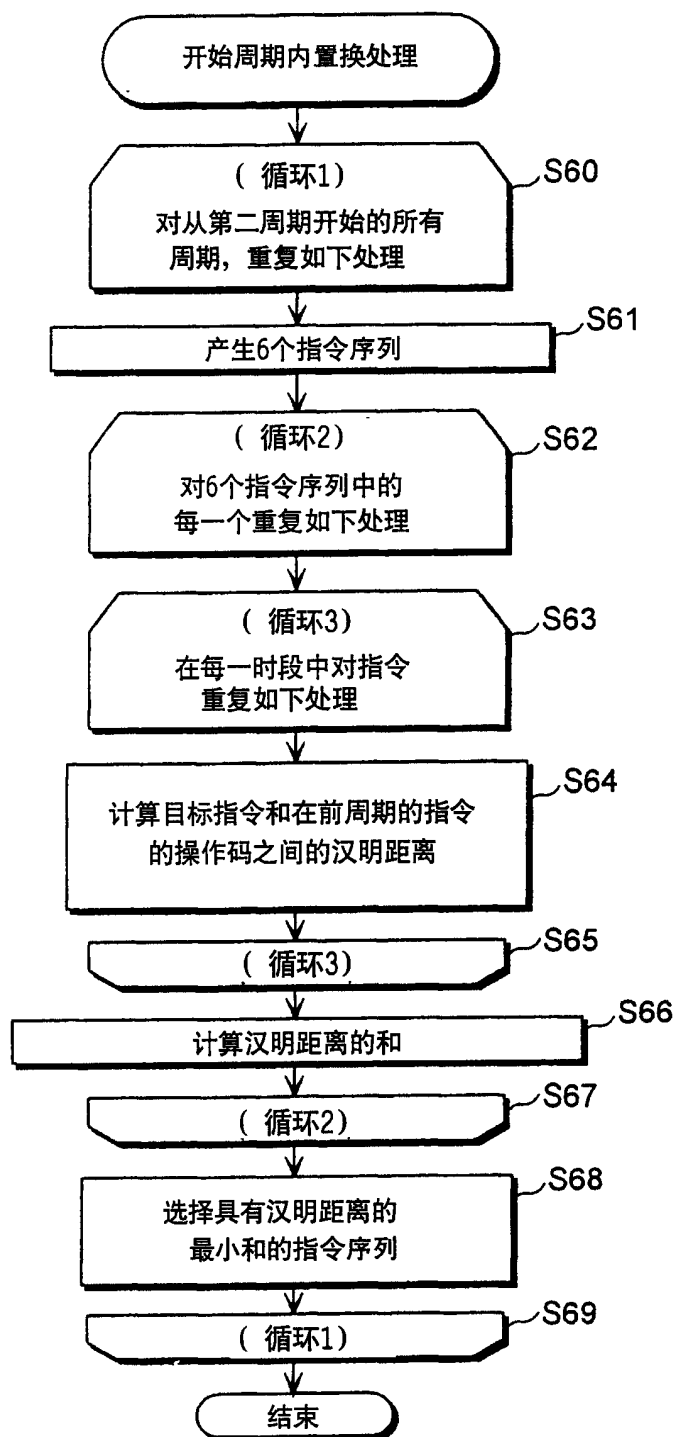
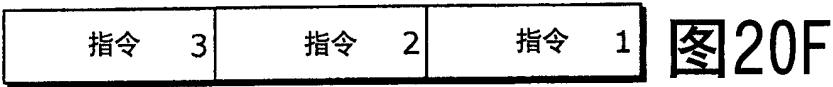
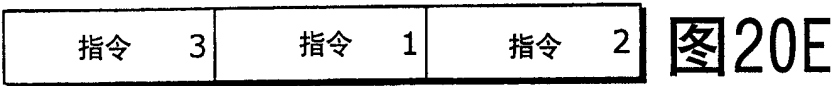
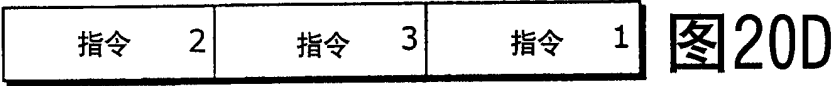
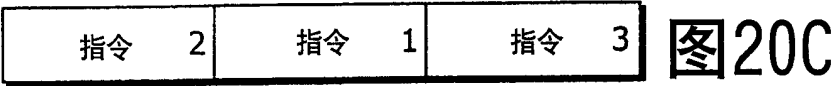
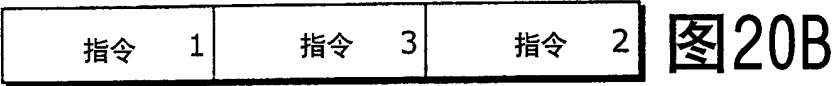
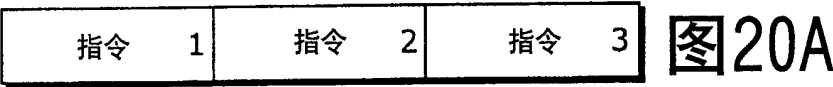


图19



周期	第一时段	第二时段	第三时段
...	...	...	...
N	ld 010101100000000100001	sub1 110000000100010100001	add1 110000000001011000001
N+1	st 0101010100000001000011	mul1 1000100011000100000001	mod 1100100011010000000001
...	...	...	...

图21

图22A

周期	第一时段			第二时段			第三时段		
N+1	st	Vr16	Vr17	mul1	Vr18	Vr19	mod	Vr20	Vr21
	010101010000001000011;			1000100011000100000001			1100100011010000000001		

图22B

周期	第一时段			第二时段			第三时段		
N+1	st	Vr16	Vr17	mod	Vr20	Vr21	mul1	Vr18	Vr19
	010101010000001000011;			1100100011010000000001			1000100011000100000001		

图22C

周期	第一时段			第二时段			第三时段		
N+1	mul1	Vr18	Vr19	st	Vr16	Vr17	mod	Vr20	Vr21
	1000100011000100000001;			010101010000001000011;			1100100011010000000001		

图22D

周期	第一时段			第二时段			第三时段		
N+1	mul1	Vr18	Vr19	mod	Vr20	Vr21	st	Vr16	Vr17
	1000100011000100000001;			1100100011010000000001;			010101010000001000011		

图22E

周期	第一时段			第二时段			第三时段		
N+1	mod	Vr20	Vr21	st	Vr16	Vr17	mul1	Vr18	Vr19
	1100100011010000000001;			010101010000001000011;			1000100011000100000001		

图22F

周期	第一时段			第二时段			第三时段		
N+1	mod	Vr20	Vr21	mul1	Vr18	Vr19	st	Vr16	Vr17
	1100100011010000000001;			1000100011000100000001;			010101010000001000011		

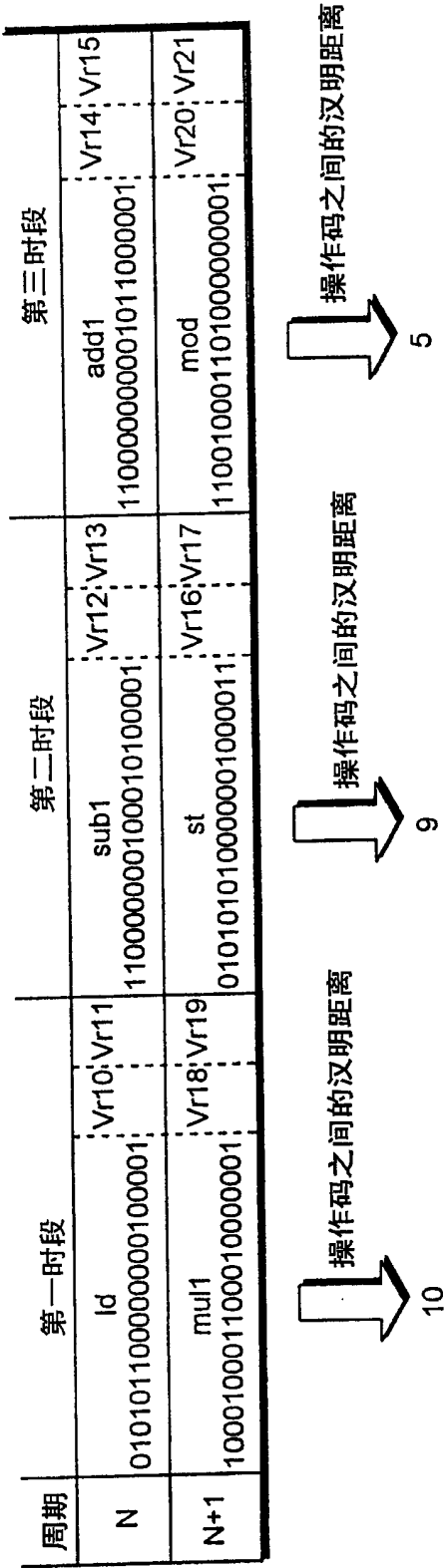


图23

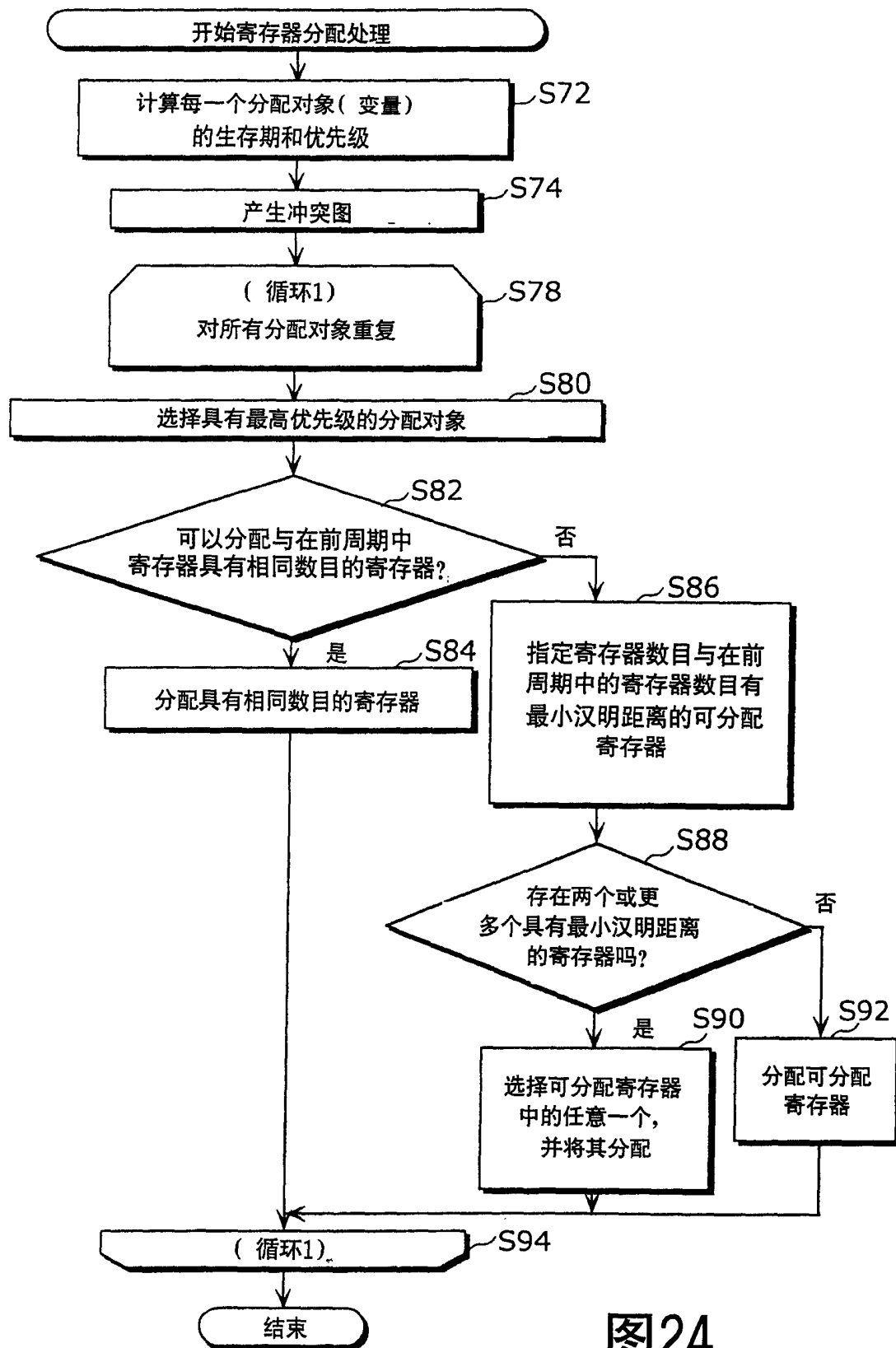


图24

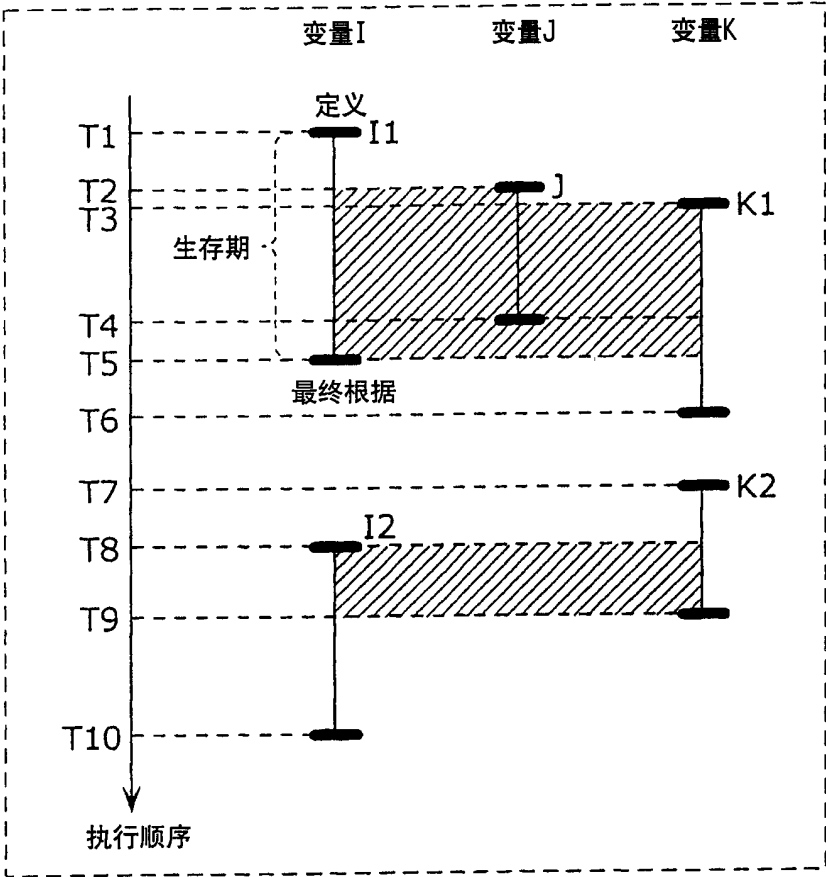


图25

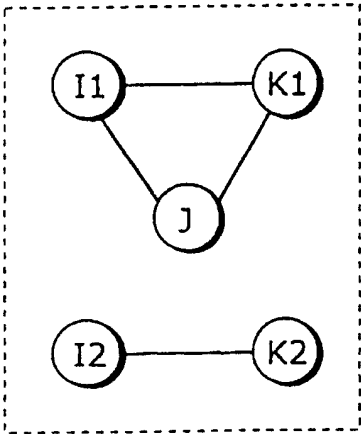


图26

周期	第一时段	第二时段	第三时段
⋮	⋮	⋮	⋮
N	add1 R0, R2		
N+1	sub1 Vr5, Vr6		
⋮	⋮		

图27A

N	add1 110000000001011000001	R0 00000	R2 00010
N+1	sub1 1100000000100010100001	R0 00000	Vr6

图27B

N	add1 1100000000001011000001	R0 00000	R2 00010
N+1	sub1 1100000000100010100001	R1 00001	Vr6

图27C

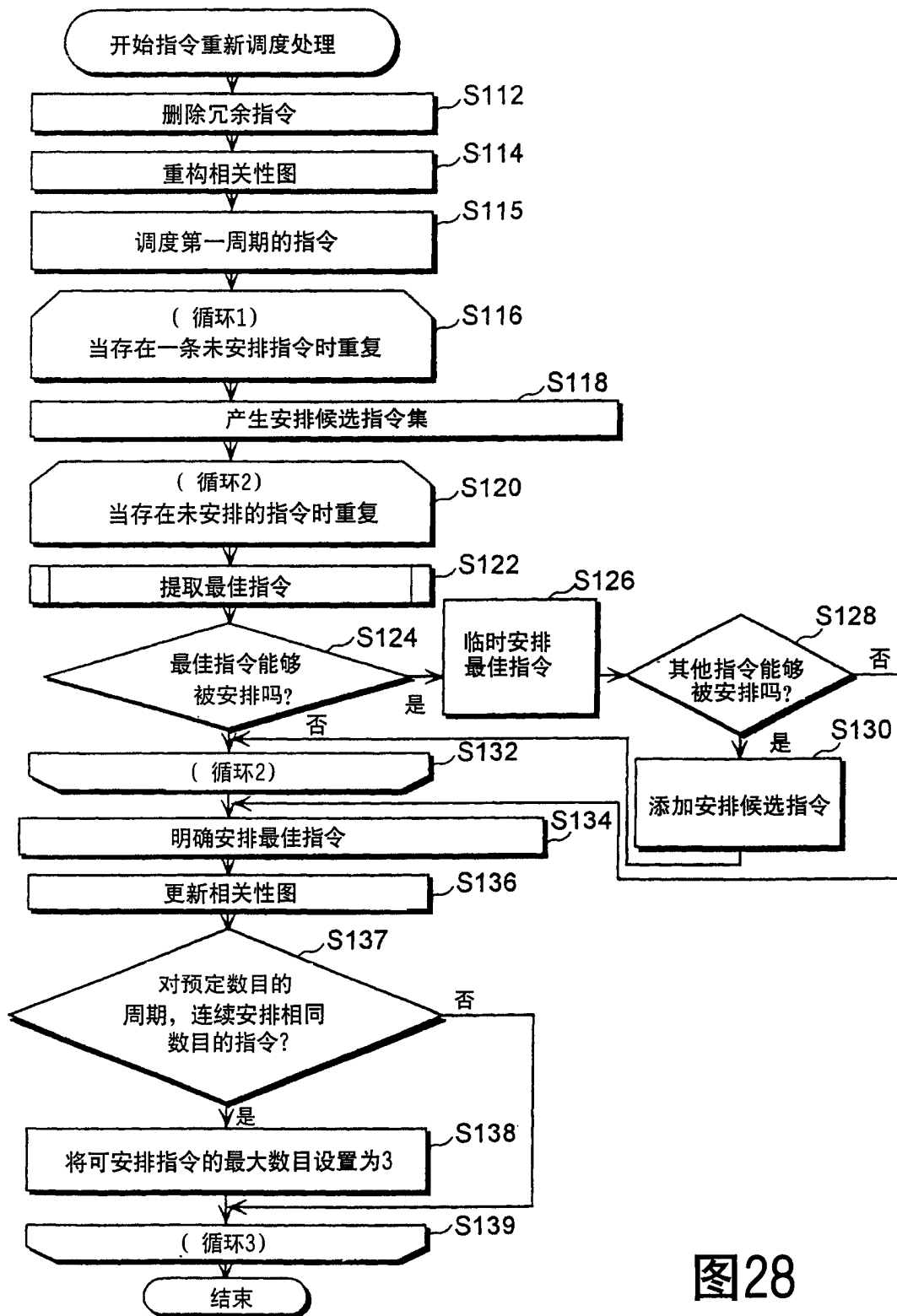


图28

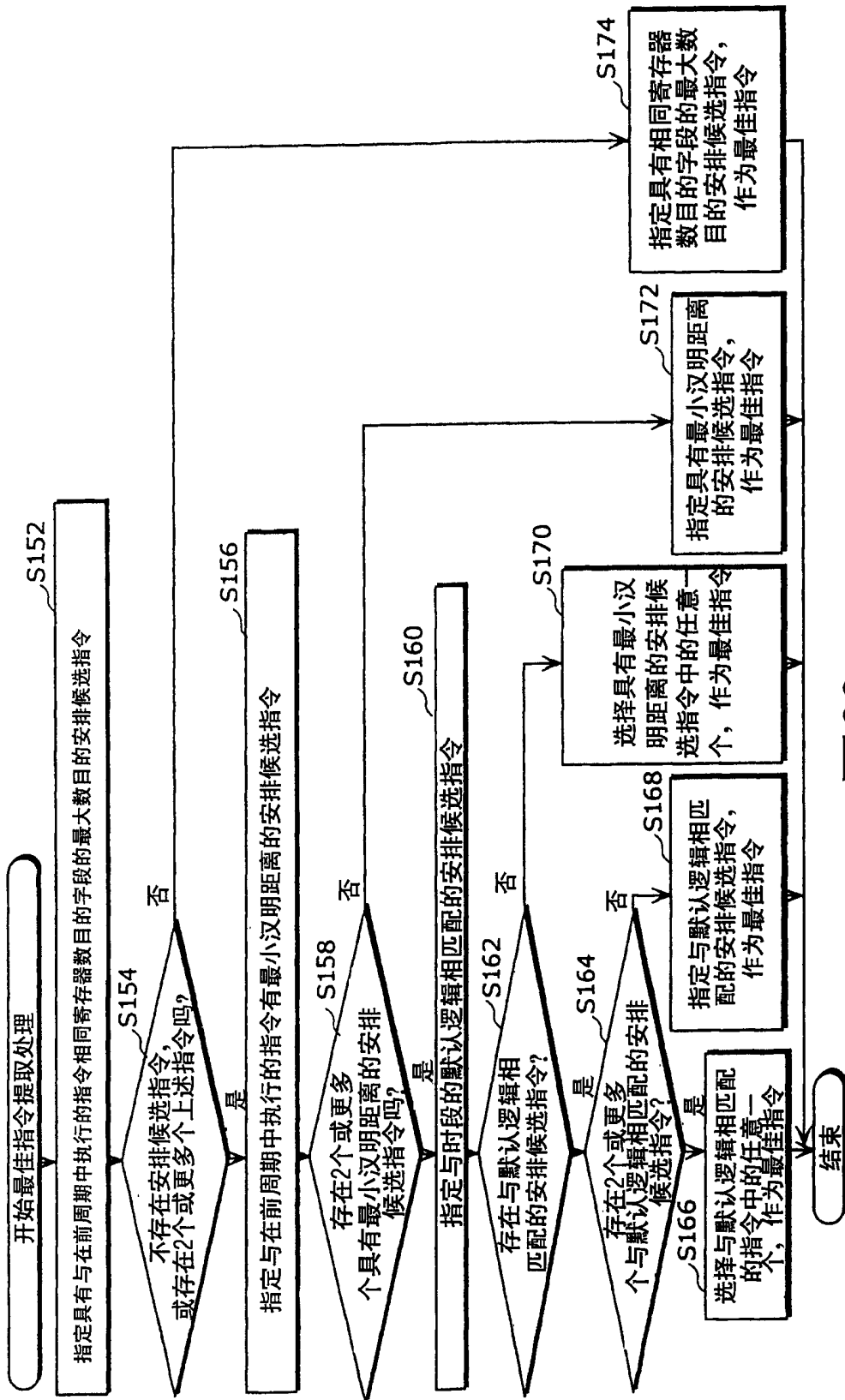
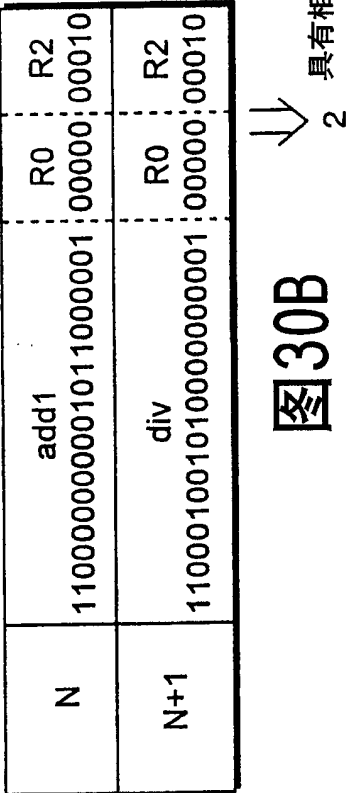
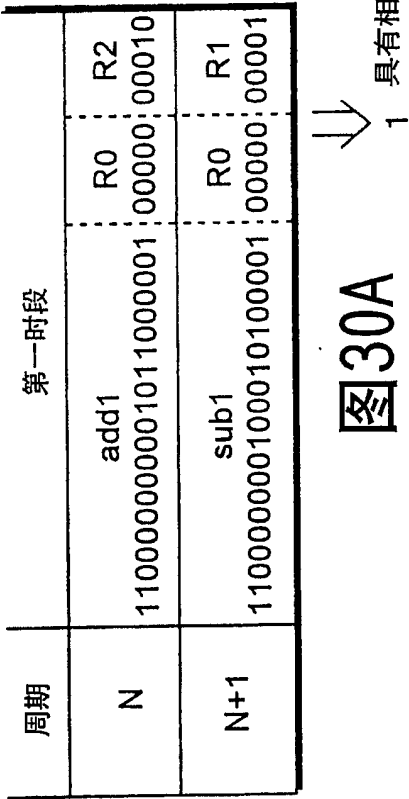
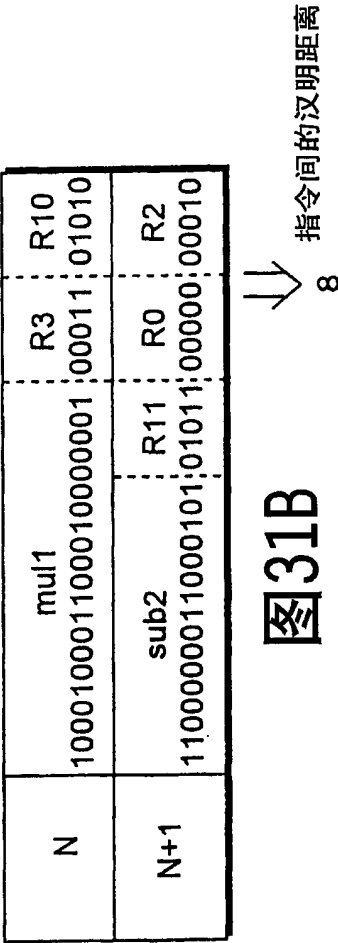
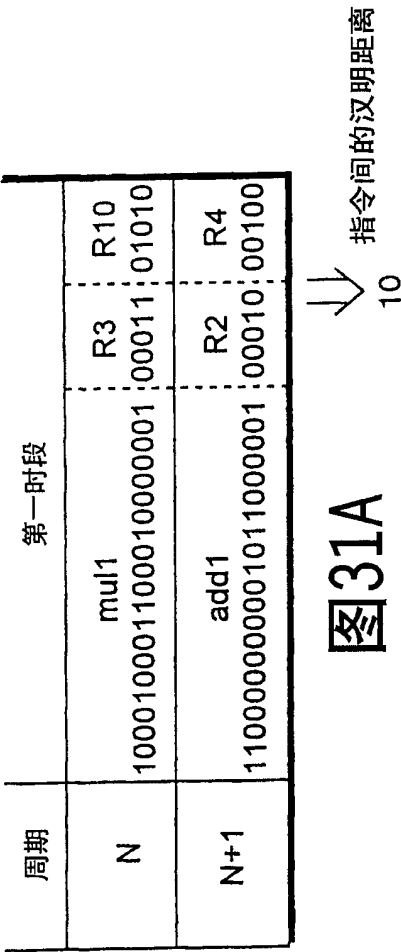


图29





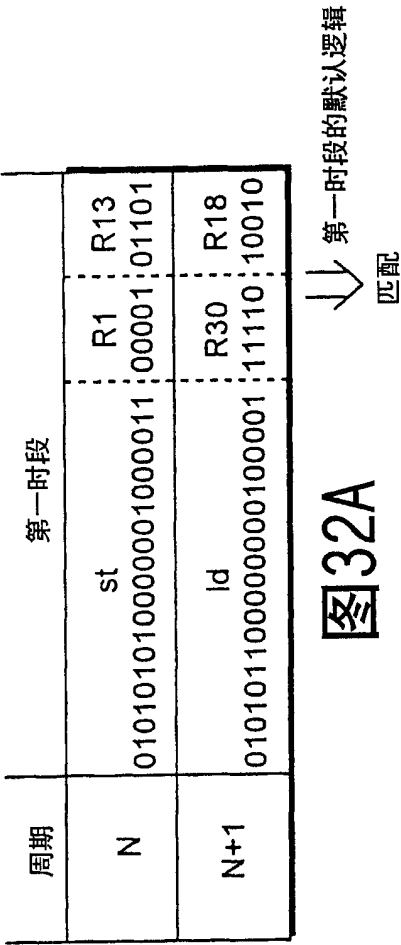


图32A

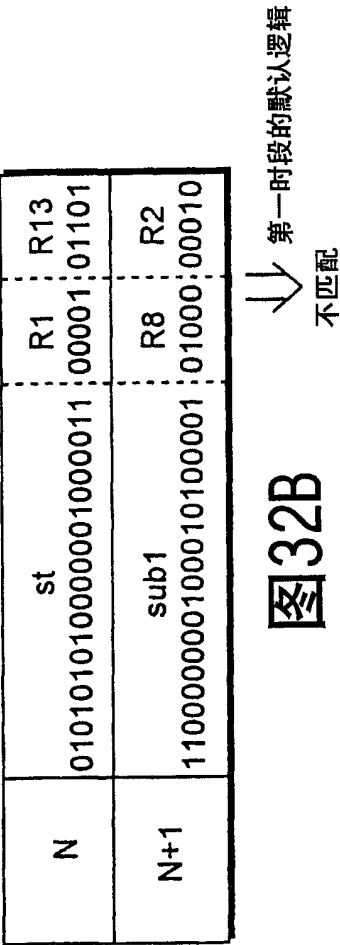


图32B

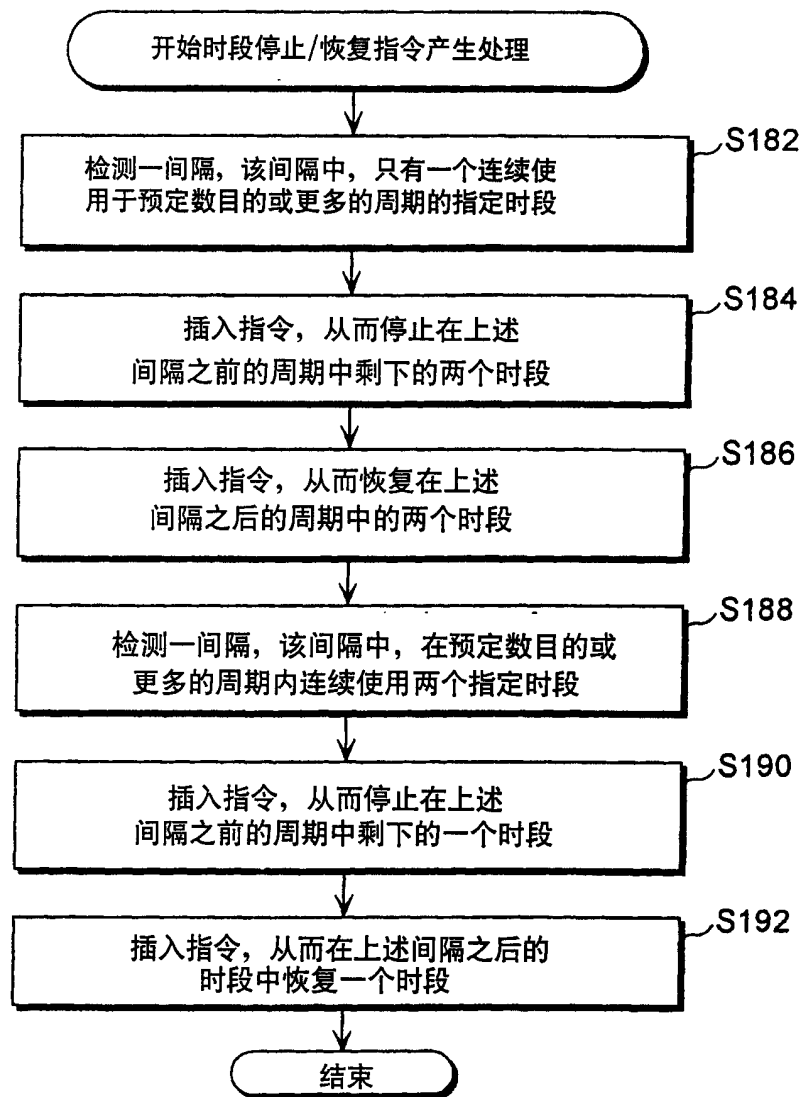


图33

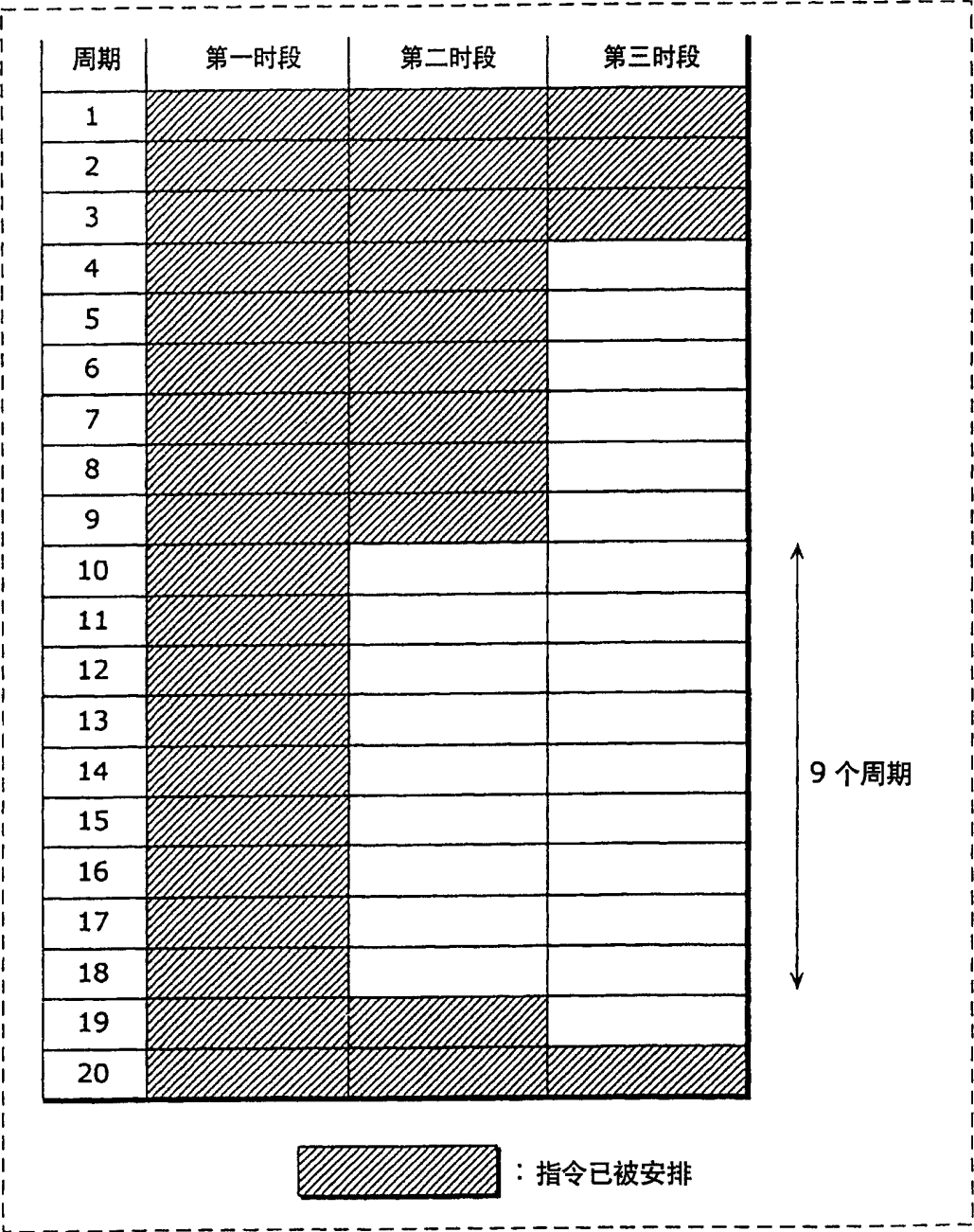


图34

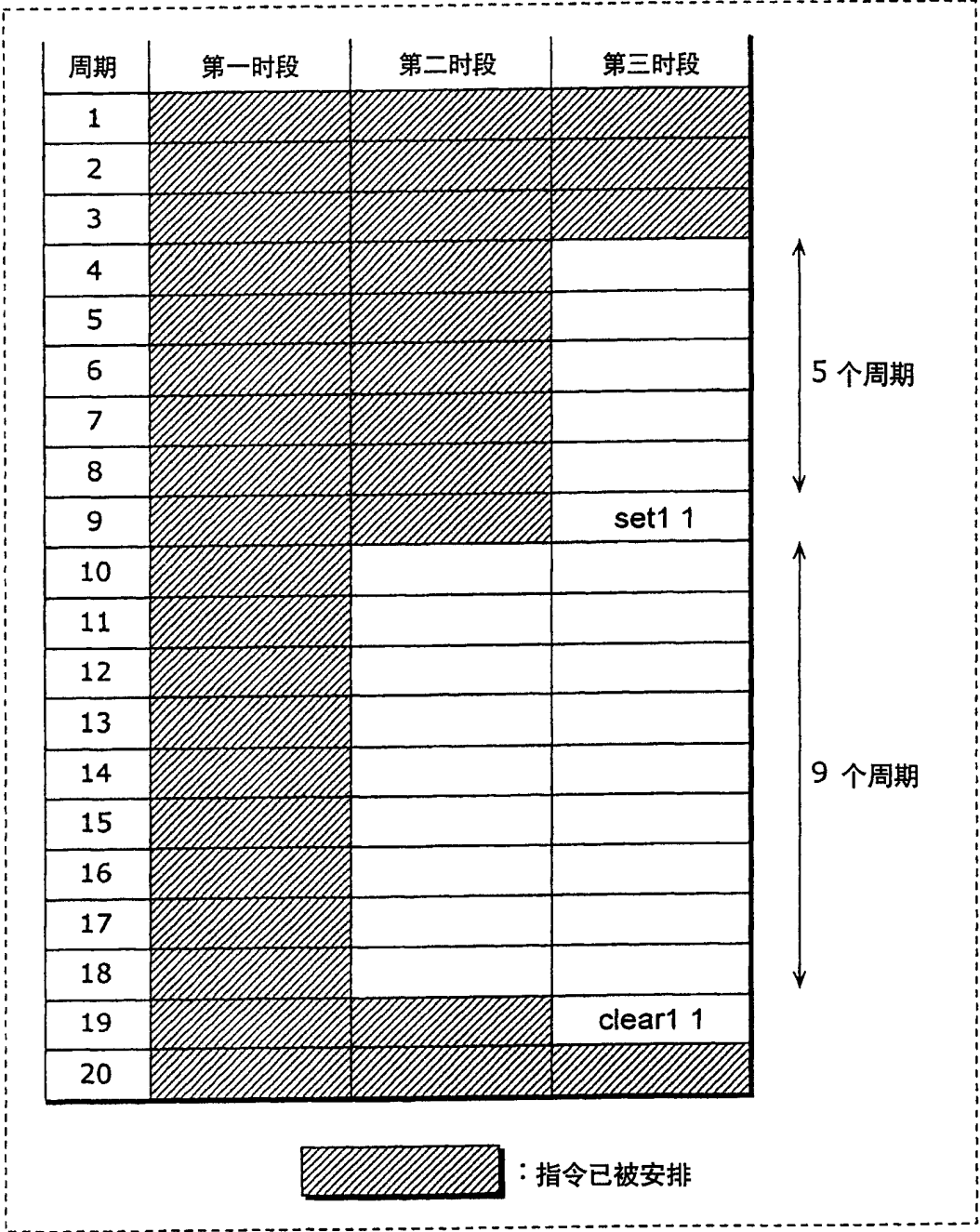


图35

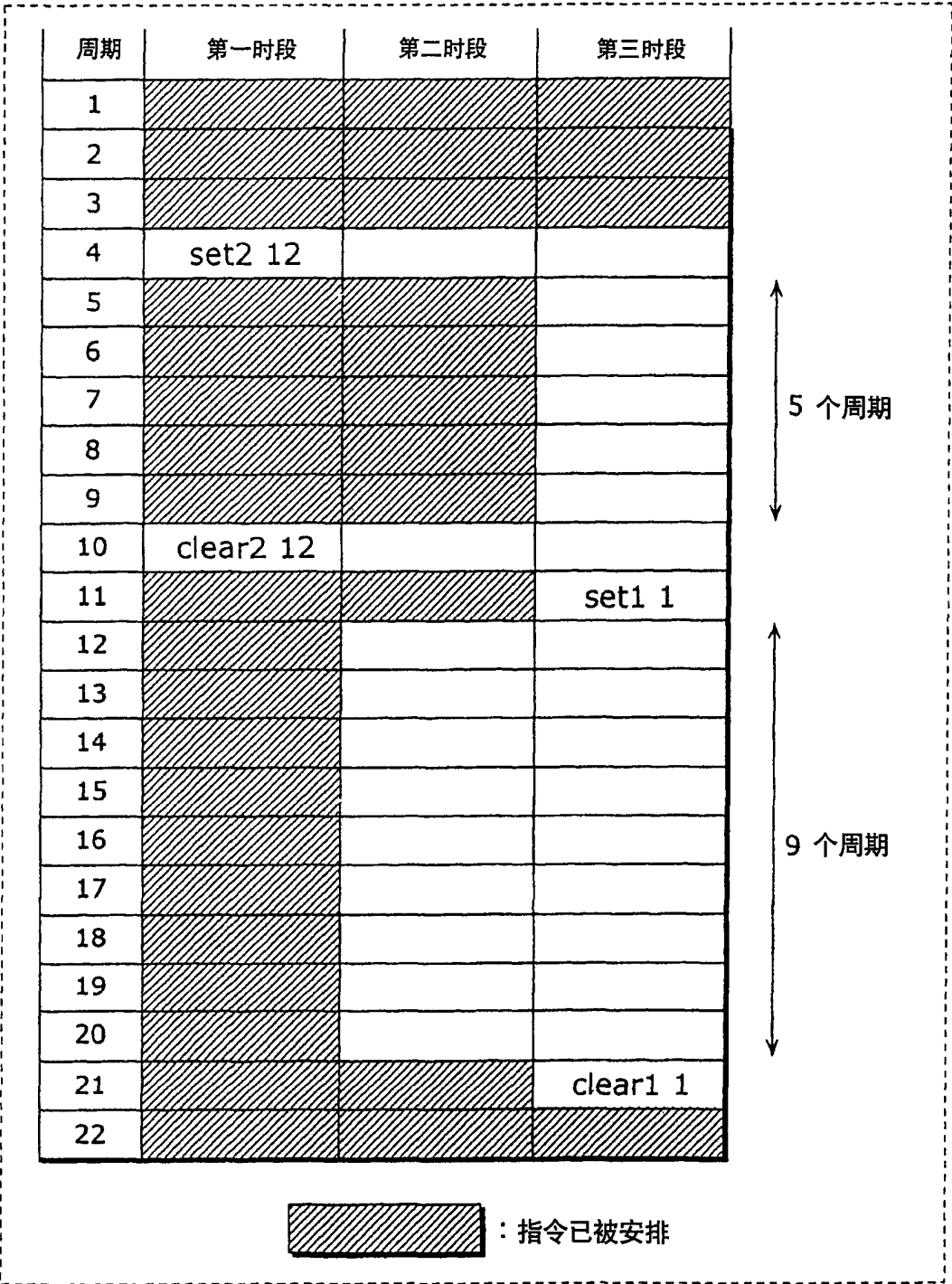


图36

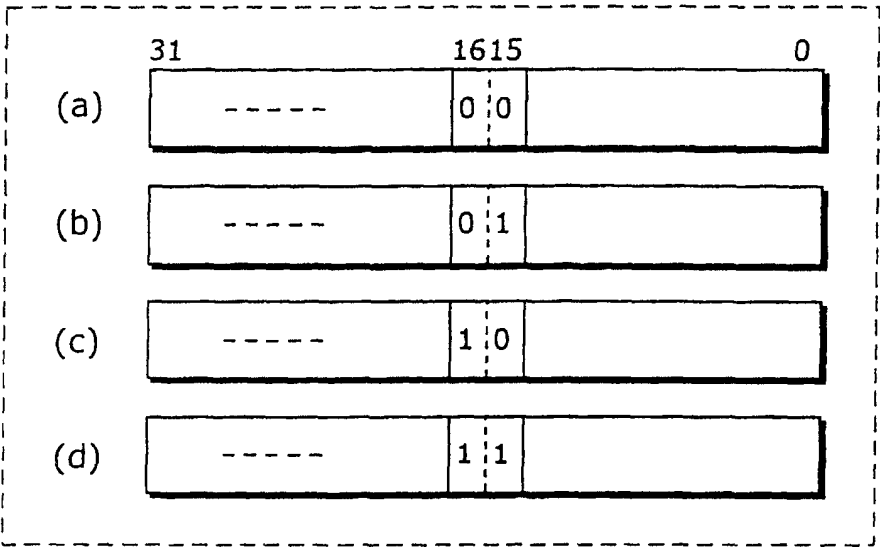


图37

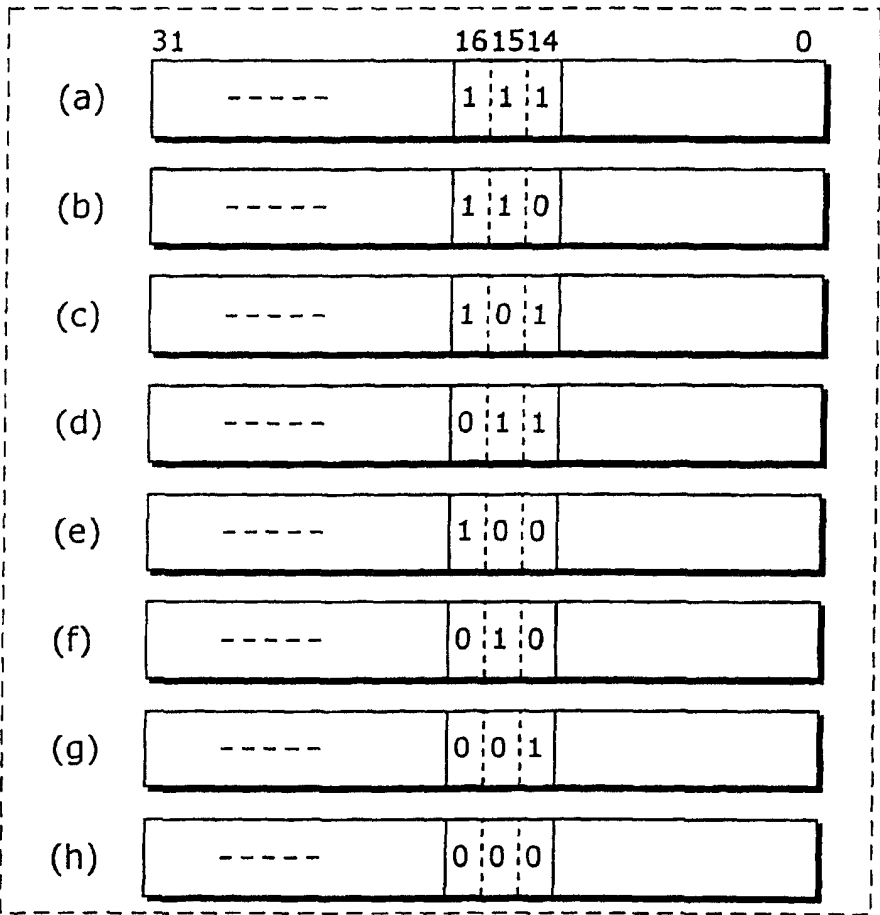


图38

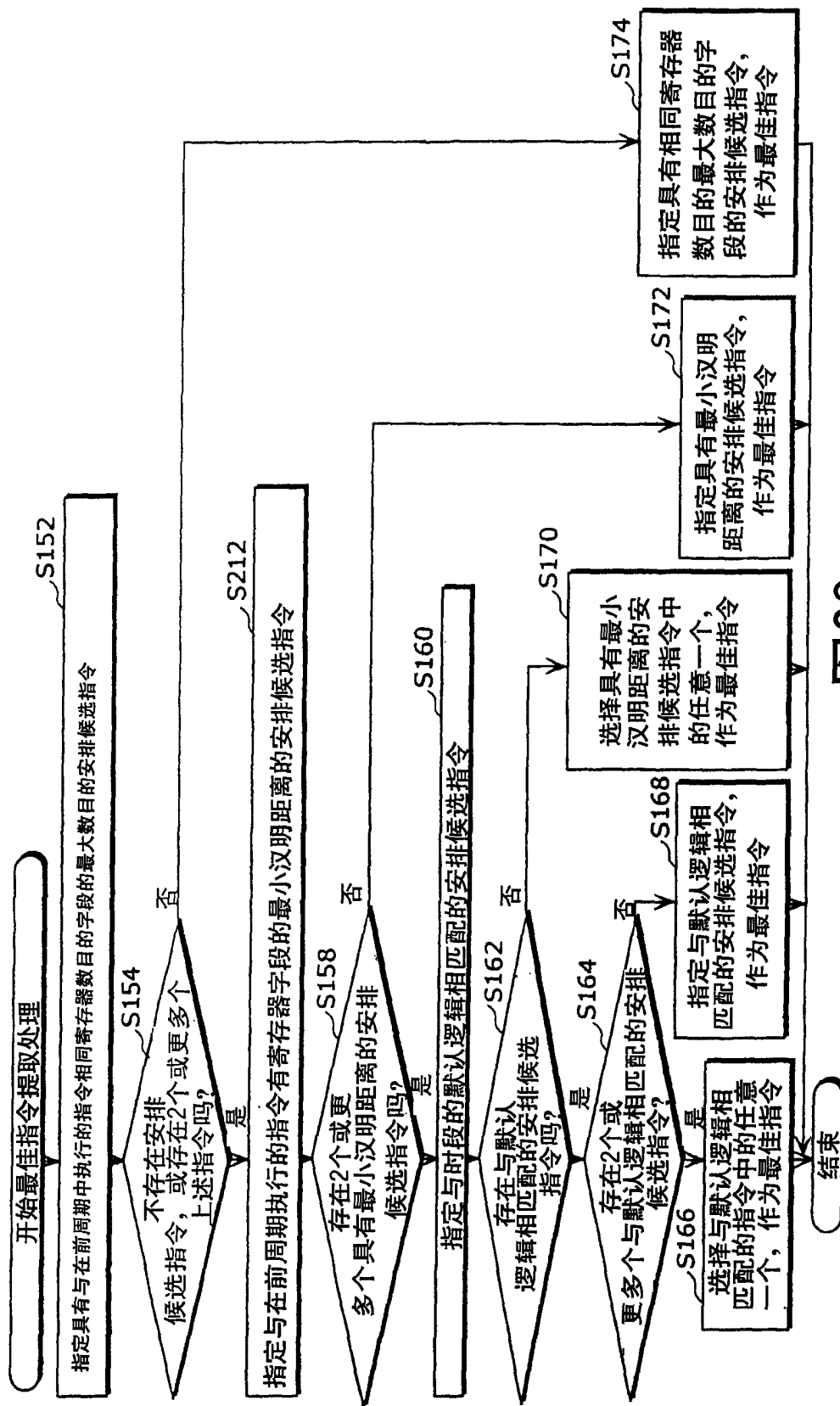
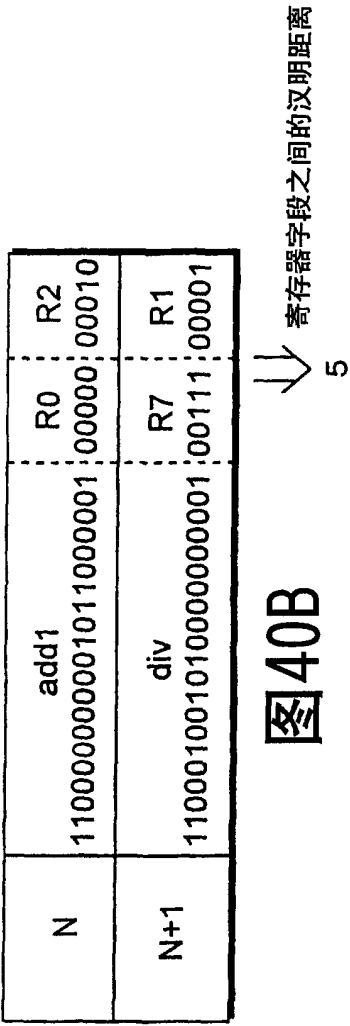
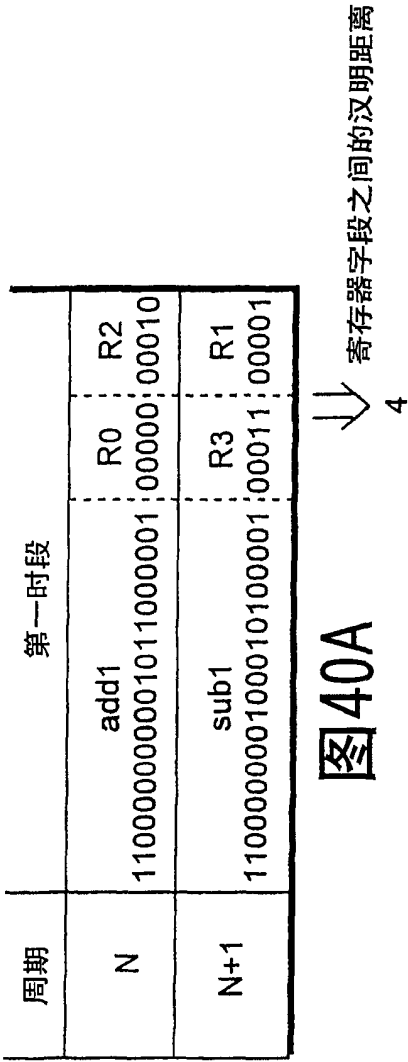


图39



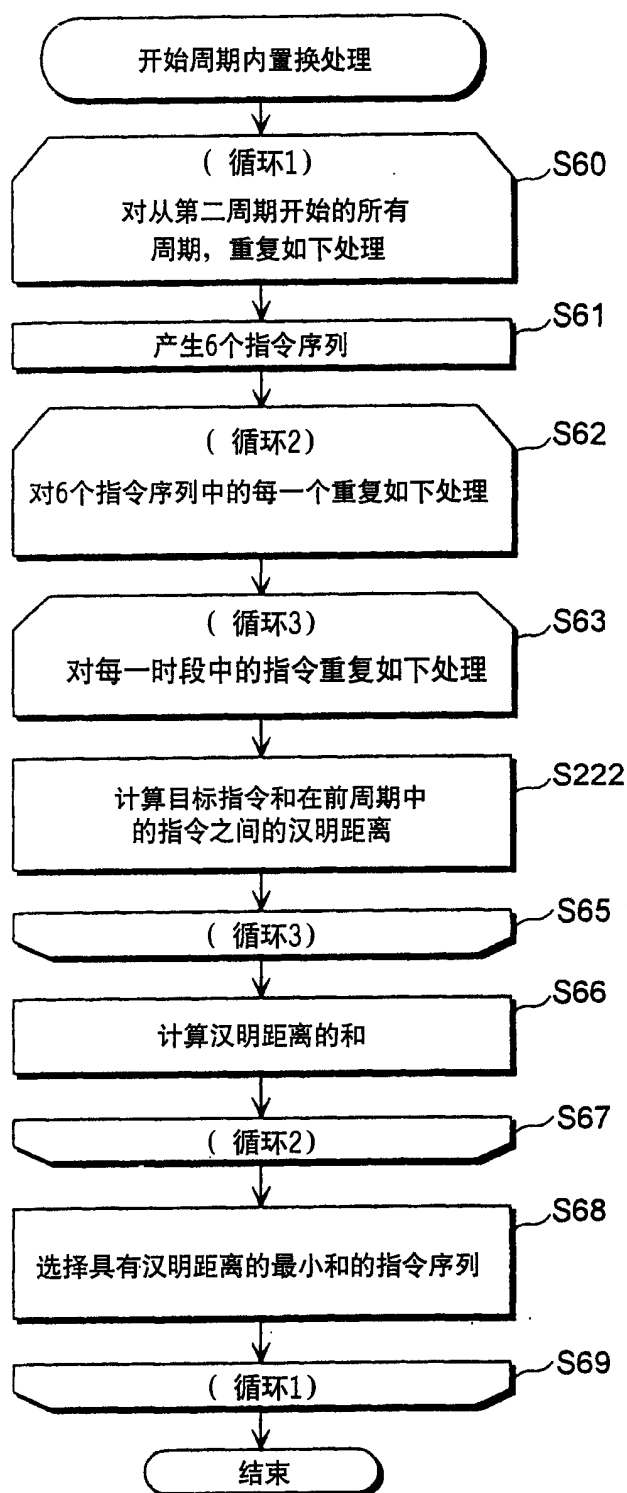


图41

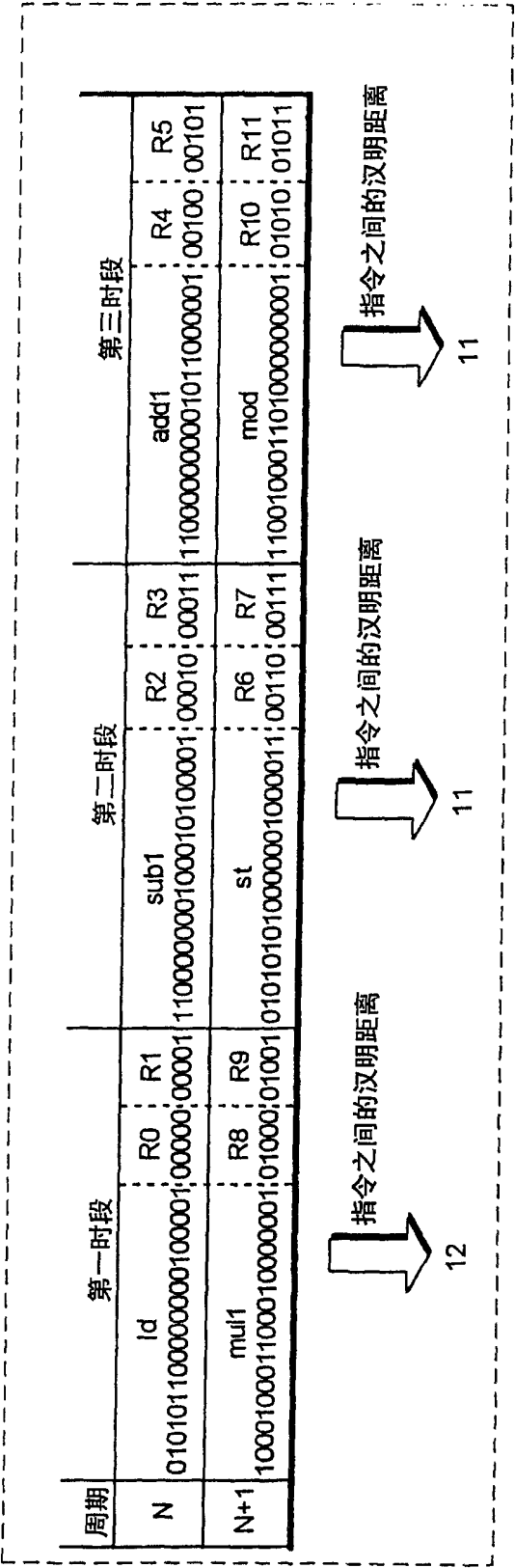


图42

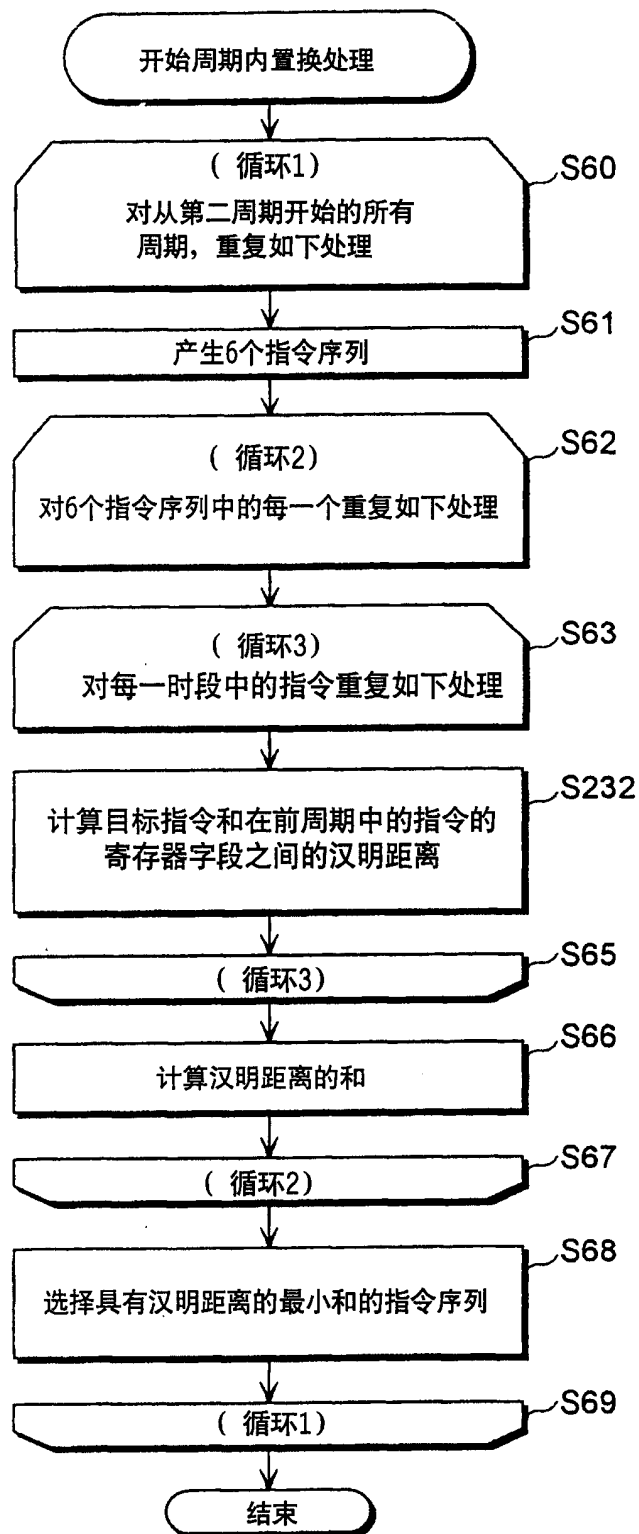


图43

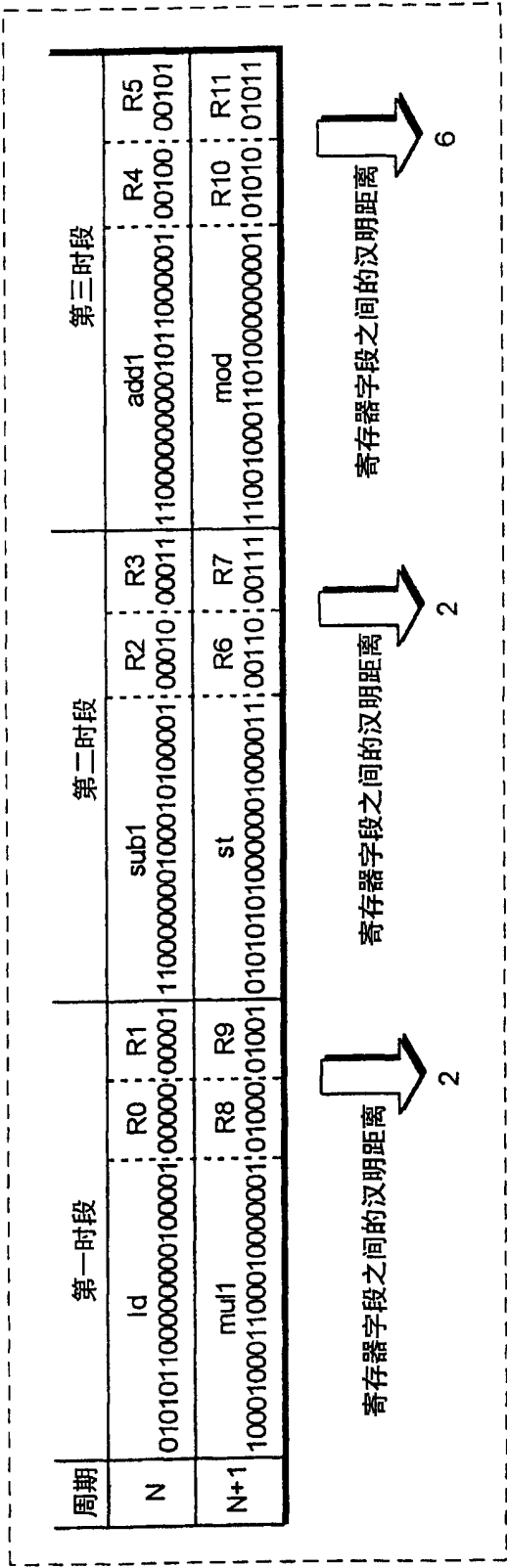


图44

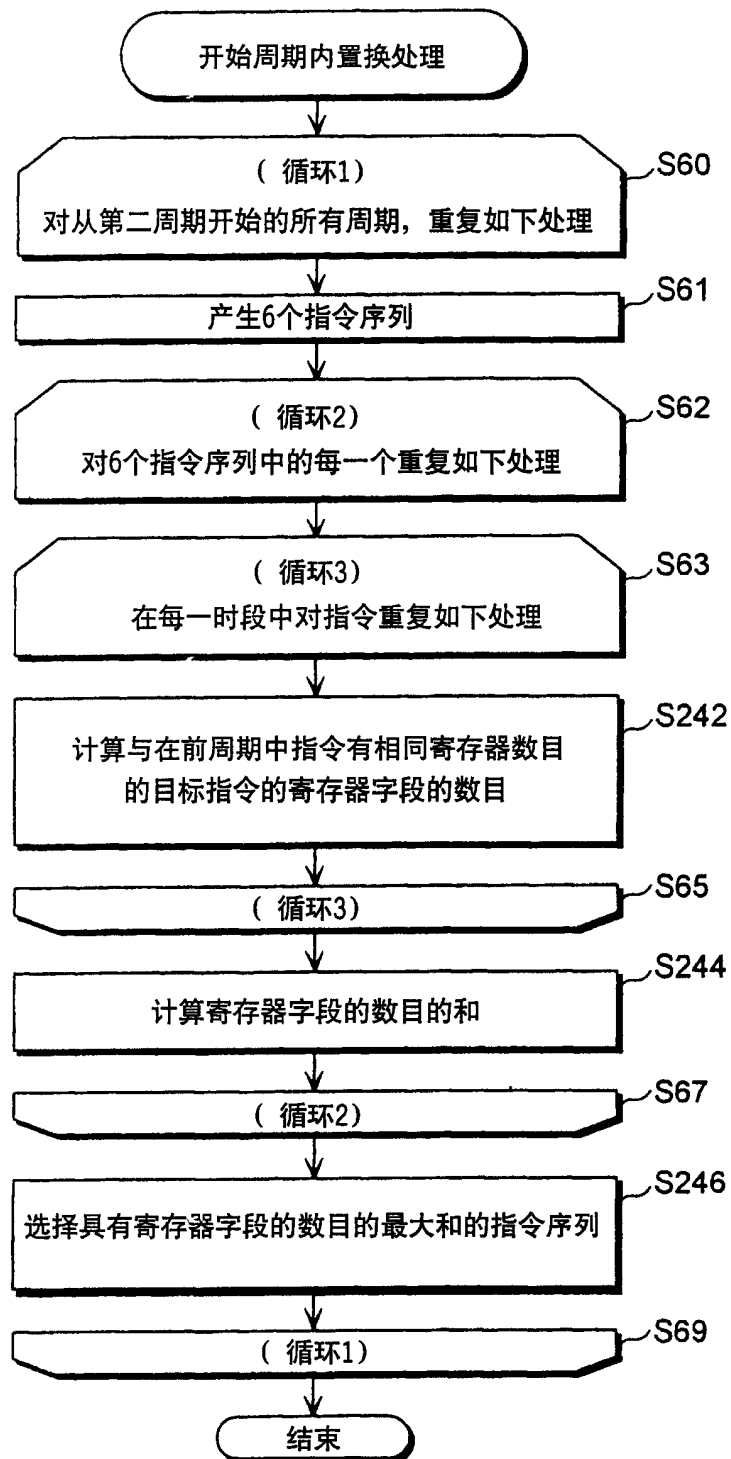


图45

周期	第一时段	第二时段	第三时段
⋮	⋮	⋮	⋮
N	Id 0101011000000001000001:00000:00001	sub1 1100000001000101000001:00010:00011	add1 1100000000001011000001:00100:00101
N+1	st 0101010100000001000011:00101:01000	mul1 1000100011000100000001:00010:00011	mod 1100100011010000000001:00000:01010
⋮	⋮	⋮	⋮

图46

周期	第一时段				第二时段				第三时段			
N+1	st 010101010000001000011;00101;01000				mul1 1000100011000100000001;00010;00011				mod 110010001101000000001;00000;01010			

图47A

周期	第一时段				第二时段				第三时段			
N+1	st 010101010000001000011;00101;01000				mod 10010001101000000001;00000;01010				mul1 1000100011000100000001;00010;00011			

图47B

周期	第一时段				第二时段				第三时段			
N+1	mul1 100010001100010000001;00010;00011				st 1010101010000001000011;00101;01000				mod 110010001101000000001;00000;01010			

图47C

周期	第一时段				第二时段				第三时段			
N+1	mul1 100010001100010000001;00010;00011				mod 110010001101000000001;00000;01010				st 0101010100000001000011;00101;01000			

图47D

周期	第一时段				第二时段				第三时段			
N+1	mod 110010001101000000001;00000;01010				st 0101010100000001000011;00101;01000				mul1 1000100011000100000001;00010;00011			

图47E

周期	第一时段				第二时段				第三时段			
N+1	mod 110010001101000000001;00000;01010				mul1 1000100011000100000001;00010;00011				st 0101010100000001000011;00101;01000			

图47F

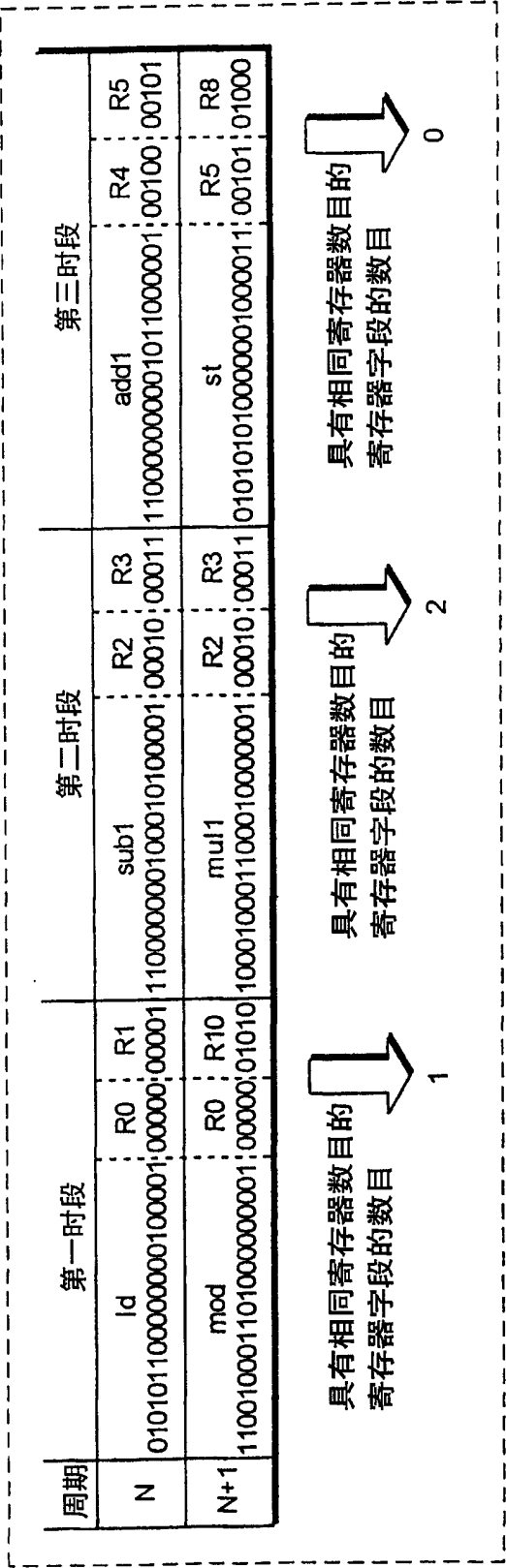


图48

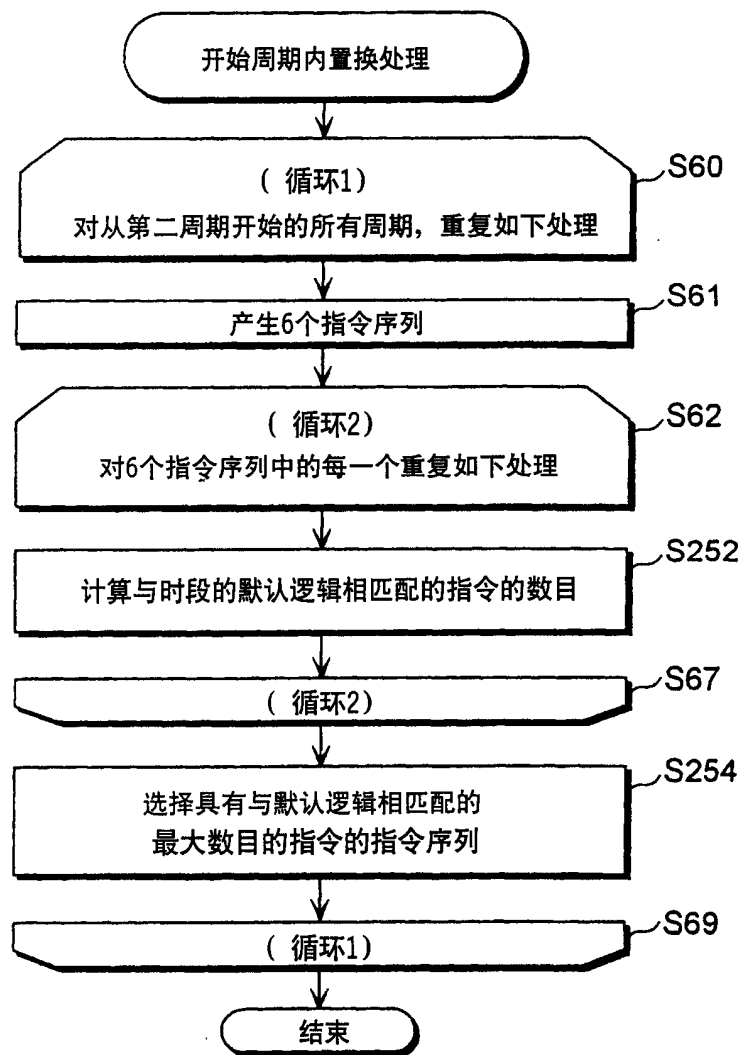


图49