[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 9/30 (2006.01)
G06F 9/318 (2006.01)
G06F 9/38 (2006.01)

[12] 发明专利说明书

专利号 ZL 200410006873.X

[45] 授权公告日 2007 年 9 月 26 日

[11] 授权公告号 CN 100339824C

[22] 申请日 1998.6.16
[21] 申请号 200410006873.X
分案原申请号 98102986.8
[30] 优先权
[32] 1997.6.16 [33] JP [31] 159048/97
[73] 专利权人 松下电器产业株式会社
地址 日本大阪府
[72] 发明人 高山秀一 桧垣信生
[56] 参考文献
US5390358A 1995.2.14
EP0689129A1 1995.12.27
审查员 胡徐兵

[74] 专利代理机构 中国专利代理(香港)有限公司
代理人 王 勇

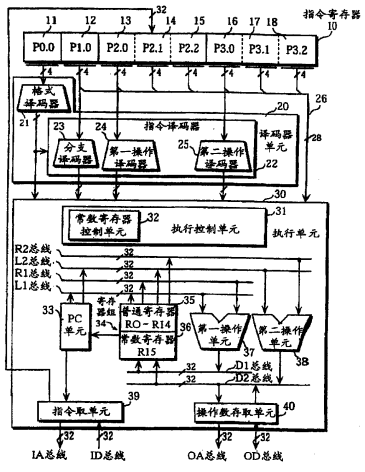
权利要求书 2 页 说明书 19 页 附图 16 页

[54] 发明名称

高效执行特长指令字的处理器和方法

[57] 摘要

32 位指令 50 是由 4 位格式字段 51，4 位操作字段 52，和两个 12 位操作字段 59 和 60 组成。4 位操作代码 52 只能包括(1)用于指示分支操作的操作代码“CC”，该分支操作使用常数寄存器 36 隐含指示的存储值作为分支地址，或(2)常数“const”。4 位操作字段 52 的内容由格式字段 51 中提供的格式代码确定。



1. 一种用以译码和执行包括一个格式字段和至少一个操作字段的指令的特长指令字处理器，其中格式字段包括一个格式代码，

所述特长指令字处理器包括:

取指令单元，用于取指令;

指令寄存器，用于存储由所述取指令单元提取的指令;

译码器单元，包括: 格式译码器，用于将所述指令寄存器中存储的所述格式字段中的所述格式代码译码和指示操作字段中的操作类型;

操作译码器，用于以所述格式译码器所指示的操作类型规定的方式将所述指令寄存器存储的所述操作字段译码;

执行单元，包括:

程序计数器单元，用于指定将要执行的指令的地址;

操作单元，用于根据所述译码器单元提供的译码结果，执行由所述操作字段指示的操作; 以及

寄存器集，用于存储要由所述操作单元执行的多个值; 以及

执行控制单元，用以控制所述执行单元。

2. 根据权利要求 1 的特长指令字处理器，

其中指令具有多个操作字段，而其中的格式代码还表示那些操作字段中的每一个的操作类型。

3. 根据权利要求 2 的特长指令字处理器，

其中那些操作字段中的每一个的操作类型还进一步由指令中的操作字段的位置指定。

4. 根据权利要求 1 的特长指令字处理器，

其中格式译码器将操作字段中包含的操作代码译码; 和

操作单元执行与被译码的操作代码对应的操作。

5. 根据权利要求 1 的特长指令字处理器，其中操作单元包括:

存储单元，用于在格式译码器指示操作字段中的操作类型是存储一个

常数的操作时，将一个常数从操作字段存储到寄存器中。

6. 根据权利要求 1 的特长指令字处理器，其中

格式译码器将包括在操作字段中的操作代码译码；和

在格式译码器还指示操作字段包括操作代码、一个源寄存器代码和一个目标寄存器代码时，操作单元用该源寄存器代码和目标寄存器代码执行与被译码的操作代码对应的操作。

7. 根据权利要求 1 的特长指令字处理器，其中

指令还具有分支字段；

格式译码器将包含在操作字段中的分支操作代码译码；和

在格式译码器还指示操作字段中的操作类型表示该分支字段包括分支操作代码时，操作单元执行与被译码的分支操作代码对应的分支操作。

8. 根据权利要求 1 的特长指令字处理器，其中格式代码还表示除格式字段外保留至少一个字段。

9. 一种用于根据权利要求 1 的特长指令字处理器的指令译码/执行方法，用以将包括格式字段和至少一个操作字段的指令译码和执行，其中格式字段包括一个格式代码，

该指令译码/执行方法包括下列步骤：

由程序计数器单元指定将要执行的指令的地址；

由取指令单元根据程序计数器单元指定的指令地址取指令；

将所述取指令单元提取的指令存储在指令寄存器中；

由包含在译码器单元中的格式译码器将所述指令寄存器中存储的所述格式字段中的所述格式代码译码和指示操作字段中的操作类型；

由包含在译码器单元中的操作译码器以所述格式译码器所指示的操作类型规定的方式将所述指令寄存器存储的所述操作字段译码；

由操作单元根据所述译码器单元提供的译码结果，执行由所述操作字段指示的操作；以及，

将要由所述操作单元执行的多个值存储在寄存器集中。

高效执行特长指令字的处理器和方法

技术领域

本发明涉及具有 VLIW（特长指令字）结构的处理器，特别是执行稍短字长和高编码效率的处理器。

背景技术

近年来随着对多媒体装置和电子电路小型化需求的增加，对微处理器的需求是能以高速处理多媒体数据，例如音频数据和图象数据。一类处理器是能够使用 VLIW 结构的处理器的需求，以后称之为“VLIW 处理器”。

VLIW 处理器包括许多内部操作单元，也就是能够以平行方式同时执行许多种 VLIW 的操作数。这样的 VLIW 是由调查在源程序级平行处理的可能性和执行编目录的程序编制器产生。对于用户装置中使用的被嵌入的微处理器，对抑制程序的代码尺寸是重要的。使得具有高关联的无操作指令（以后称为“无操作指令”）和产生的差的代码效率的结果的 256 位 VLIW 离理想情况很远。

执行相对短字长指令的 VLIW 处理器的一例是日本公开专利申请 H09-26878。该技术是执行 32 位指令的并能同时指示最大两个操作的 VLIW 处理器的数据处理装置。

图 1A 和图 1B 示出图 1 示出的用于同时指示两个操作的指令格式和图 1B 示出的用于指示只一个操作的指令格式的规定技术的指令格式。该技术的目的在于，通过示明在每个指令中的操作数和执行顺序的格式字段 410 中的 2 位值来改进代码效率。

通过单个 32 位指令表明的最大两个操作，无论如何没有达到足够的平行度等级。还有一个问题是，当利用一超过给定字长的常数进行操作时会降低指令的代码效率。作为一个举例，当 32 位常数被提供为上 16 位和下 16 位，使得它能被置入寄存器时，恰需要两个 32 位指令去指示利用该常数的操作。

发明内容

从所述的问题看，本发明的一个目的就是提供一种 VLIW 处理器，能执行相对短字长指令而又具有高等级平行度和高效代码结构，使得能同时指示若干

操作。作为一个举例，由单个32位指令能指示三或更多操作。

本发明的第二个目的是提供一种VLIW处理器，用于执行具有某种结构的相对短字长指令，从而使总的代码效率相对不受影响，甚至当处理相对长的字长的常数时也是如此。

通过VLIW（特长指令字）处理器能够实现第一个目的，它能译码和执行具有至少两个操作字段的指令，第一个操作字段能只包括确定操作类型的一个操作代码和第二操作字段包括一个操作代码和至少一个由第二操作字段指示的操作中使用的操作数的组合，该VLIW处理器包括：第一译码单元，用于译码在第一操作字段中的操作代码；第一执行单元，用于根据第一译码单元的译码结果执行由在第一操作字段中的操作代码指示的操作；第二译码单元，用于译码在第二操作字段中的操作代码；和第二执行单元，用于根据第二译码单元的译码结果，在由第二操作字段中的操作数指示的数据基础上，执行由第二操作字段中的操作代码指示的操作。

在此情况下，由于通过只插入一个操作代码而没有清楚的操作数指示能够指明指令中至少一个操作，从而能降低指令的字长。其结果是，执行相对短字长的而具有高效代码结构的指令，使得能同时指示若干操作的VLIW处理器得以实现。

这里，在第一操作字段中由操作代码占有的多个位数可以等于在第二操作字段中由操作代码占有的多个位数。

作为一种结果就是，在一指令中包括的所有操作代码都由相同的数目的比特组成多位数，从而简化了例如译码电路的单元。

这里，指令可以包括三个操作字段，在三个操作字段中的第三操作字段可以占有如第二操作字段的相同位数和也可以包括一个操作代码和至少一个操作数的组合，该VLIW处理器进一步包括：第三译码单元，当一操作代码存在于第三操作字段时，它译码第三操作字段的操作代码；和第三执行单元，用于根据第三译码单元的译码结果，在由第三操作字段中的操作数指示的数据基础上，执行由在第三操作字段的操作代码指示的操作。

其结果是，具有高等级平行度从而能同时进行三个操作的VLIW处理器能够实现。

这里，第一执行单元可以控制包括该指令的程序流。

其结果是，一般不需要大量比特数的分支操作可被赋予短的操作字段。这

就意味着可以确定高代码效率的指令集合。

这里，第二执行单元可以控制由在第二操作字段中包括的操作数指示的数据的传送，和第三执行单元，在由第三操作字段中包括的操作数指示的数据的基础上，可以控制执行一算术逻辑操作。

其结果是，可以由在一指令中的单个操作指示将数据传送到外部存储器和将数据从外部存储器传送来，使得能简化应在 VLIW 处理器中提供的操作数存取电路。

通过 VLIW 处理器能实现本发明的第二个目的，该处理器能译码和执行具有至少两个操作字段的一指令，第一操作字段只包括（i） 用于确定一操作类型的一单个操作代码的一个和（ii）一常数，和第二操作字段包括（i）一操作代码和由第二操作字段指示的一操作中使用的至少一个操作数的组合的一个和（ii）一常数，该 VLIW 处理器包括：第一译码单元，用于当操作代码出现在第一操作字段中时，译码在第一操作字段的操作代码；第一执行单元，用于根据第一译码单元的译码结果，执行由在第一操作字段中的操作代码指示的操作；第二译码单元，用于当操作代码存在于第二操作字段时，译码在第二操作字段的操作代码；和第二执行单元，用于根据第二译码单元的译码结果，在由第二操作字段中的操作数指示的数据基础上，执行由第二操作字段的操作代码指示的操作。

采用以上结构，当在一指令的操作字段中必须放入无意义的代码时，由不同操作使用的常数可以替换被插入，使得 VLIW 处理器能实现用于执行具有高代码效率的指令，而不管只具有短的字长。

这里，该指令还包括含有指明在第一操作字段中是否只有一个常数和在第二操作字段中是否只有一个常数的格式代码的格式字段，该 VLIW 处理器进一步包括：用于译码格式代码的格式译码单元；和用于当格式译码单元的译码结果表明在第一操作字段和第二操作字段的至少一个中只存在一个常数时，提取在该指令中的该常数并存储该提取的常数的常数存储单元。

其结果是，在操作字段中放置的常数能被存储在常数存储单元，用于稍后指令的操作，使得能避免代码效率的降低，甚至是在当利用相对短字长的指令，处理相对长字长的常数也是如此。

这里，格式字段，第一操作字段，第二操作字段中的操作代码，在第二操

作字段中的每个操作数，在第三操作字段中的操作代码，和第三操作字段中的每个操作数，每个都可占据n位。

采用上述结构，组成一指令的所有字段都具有相同的位数，这就能简化VLIW处理器的内部电路。

这里，VLIW处理器可以包括：取单元，用于取包括有n个操作字段的L位指令；和n个操作单元，每一个都与该取得的指令中的n个操作字段的不同的一个相关联，并且每一个都相互以平行方式单独执行相关操作字段中指示的操作；VLIW处理器的特征在于，n个操作字段并不全是相同尺寸，和L并不是n的整数倍。

采用上述结构，并不需要指令的所有操作字段都具有相同的字长，这就可能以高代码效率确定指令。其结果是，VLIW处理器执行相对短字长而具有高效代码结构的指令，使得能实现同时指示若干操作。

这里，n可以是3，和L可以是32。

采用上述结构实现的VLIW处理器，具有高等级的平行度，从而由一单个32位指令指定的三个操作能被同时进行。

这里，在n个操作字段中的至少一个操作字段中包括的操作数的数目，可以不同于在该n个操作字段中的另一个操作字段中的操作数的数目。

采用上述结构，不需要指令中的每一个操作字段都具有相同数目的操作数，使得能确定具有高等级的代码效率的指令格式。

这里，n个操作字段可以包括只有一个操作代码组成的至少一个操作字段和由一个操作代码和至少一个操作数组成的至少一个操作字段。

采用上述的结构，该指令字长要比当指令中的每个操作字段包含有操作代码和操作数的组合的情况要短，使得能够实现执行具有高效代码结构的指令的VLIW处理器。

如上所述，本发明实现的VLIW处理器能执行相对短字长而又具有允许将由单个指令指定的若干操作的高效代码结构的指令。对于处理多媒体数据的嵌入处理器，其作用尤为显著。

附图说明

随同描述本发明特定实施例的附图的以描述，本发明的这些和其它目的，优点和特点将会变得极为明显。图中：

图 1A 和 1B 是现有技术的指令格式， 图 1A 是同时指示两个操作的指令格式和图 1B 是只指示一个操作的指令格式；

图 2A 是由本发明处理器执行的指令的字段结构；

图 2B 至 2D 示出 16 类指令格式，图 2B 示出三个操作指令， 图 2C 示出二操作指令，和图 2D 示出单个操作指令；

图 3 的表示出由三种类型操作代码指示的特定操作，即由图 2 使用的“CC”、“OP1”，和“OP2”；

图 4 方框图示出本发明的硬件结构；

图 5 方框图示出本处理器的常数寄存器 36 的详细结构和外围电路；

图 6A 至 6D 表示用于由图 5 所示常数寄存器控制单元 32 存储常数的不同方法，图 6A 示出当格式代码是“0”或“1”时的情况，图 6B 示出当格式代码是“4”时的情况，图 6C 示出当格式代码是“5”时的情况，和图 6D 是当格式代码是“2”、“3”，或“A”时的情况；

图 7 方框图是本处理器的 PC 单元 33 的详细结构；

图 8 是处理 32 位常数的过程的流程；

图 9 是本处理器执行图 8 所示过程的程序举例；

图 10 是当执行图 9 所示程序时，本处理操作的定时图；

图 11 是本处理器执行处理 16 位常数的过程的程序举例；

图 12A 示出由标准处理器执行的指令的字段定义；

图 12B 是图 12A 所示指令的指令格式；

图 13 示出标准处理器进行如图 9 所示程序的相同过程的程序举例；

图 14 示出标准处理器执行如图 11 所示程序的相同过程的程序举例；

图 15A 至 15D 示出由本发明的 VLIW 处理器执行的指令的结构的改进；和

图 16 示出能执行图 15A 所示指令的本处理器的硬件结构的改进。

具体实施方式

以下将参照附图描述本发明的处理器的实施例。在该实施例中，表达式“指令”呈现为代码集合，它随着表达式“操作”呈现给由本处理器平行执行的处理单元，例如算术操作，逻辑操作，传送，或分支的处理单元译码和执行，以及指示每个处理单元的代码。

首先描述由本处理器译码和执行的指令结构。本处理器是一 VLIW 处理器，它译码和执行具有 32 位固定字长的指令。

图 2A 示出由本处理器执行的指令 50 的字段结构。同时图 2B 至 2D 示出 16 个指令格式。图 2B 这些指令格式同时指示三种操作，图 2C 两个操作的指令格式，和图 2D 单个操作的指令格式。

指令 50 具有固定 32 位字长，和由依序示出的 8 个 4 位物理字段组成，开始从 MSB（最高有效位）作为图 2A 中的 P0.0 字段 51，P1.0 字段 52，… P3.2 字段 58。从 P2.0 字段 53 至 P2.2 字段 55 的这些范围被称为第一操作字段 59，而从 P3.0 字段 56 至 P3.2 字段 58 的范围被称之为第二操作字段 60。

在图 2B 至 2D 中，代码“const”表示一常数，并取决于它使用的操作，可以是一数字常数或一符号常数，例如，立即，绝对地址，或位移。代号“OP”表示操作代码，指示操作类型，而代号“RS”表示用作为源操作数的寄存器，“Rd”是作为目的操作数的寄存器，和“CC”操作代码指示分支操作，它使用本发明处理器提供的专门的 32 位寄存器（图 4 中所示的常数寄存器 36）的存储值作为分支目的地的绝对地址或相对地址（位移）。

在上述代码之后直接给出的数字值，表示在第一操作字段 59 或第二操作字段 60 中二者之一的操作中使用的值。作为一例，对于具有格式代码“6”的指令格式，位于 P1.0 字段 52 中的 4 位常数“const1” 和位于 P2.1 字段 54 中的 4 位常数“const1” 被组合形成一 8 位常数，即对应于第一操作字段 59 的操作代码“OP1”的源操作数。

不附加数目的常数“const”表示是将存储在本处理器提供的专门的 32 位寄存器（图 4 中所示常数寄存器 36）中的常数。 作为一例，对于具有格式代码“0”的指令格式，位于 P1.0 字段 52 中的 4 位常数“const”意味着该常数将被存储在被隐含指示的常数寄存器 36 中。

图 3 示出能由图 2 给三类操作代码“CC”，“OP1”、和“OP2”表示的操作的特例。以下将详细描述这些操作。

4 位操作代码“CC”表示 16 类分支指令的一个。每个分支指令确定为一种分支条件和分支格式。分支条件的举例包括“等于”（’eq’）”，“不等于（’neq’）”，和：大于“（’gt’）”。该分支格式可以是作为分支目的的绝对地址的常数寄存器 36 的存储值的格式（用不附加“i”到指令助记符表示），

或作为相对地址的常数寄存器36的存储值的格式（附加“i”到指令助记符表示）。作为一例，操作代码“eq”表示，当予先比较确定比较的值相等时，分支到通过绝对地址指示的目的的操作，而操作代码“eqi” 表示，当予先比较确定比较的值相等时，分支到通过相对地址指示的目的的操作。

4位操作数“OP1”可被用于表示算术逻辑操作，例如任何的“add”（加），“Sub”（减），“mul”（乘），“and”（逻辑加），或“or”（逻辑或），或者是内部寄存器传送的操作，例如任何的“mov”（字（32位）数传送），“movh”（半字数据传送），或“movb”（一字节数据传送）。

4位操作数“OP2”可被用于表示任何算术逻辑操作或内部寄存器传送，该变换能由操作数“OP1”表示，而且也能用于指示寄存器—存储器传送操作，例如“ld”（从存储器将一字数据加载到寄存器）或“st”（从寄存器将一字数据存储到存储器）。

以下描述图2A所示字段51，52，59和60的特征。

P0.0字段51保存确定指令50的格式的4位格式代码，特别是该P0.0字段51确定图2B至2D所示16个指令格式中的一个。

P1.0字段52是保存用于分支操作的常数或操作代码的字段。当常数位于P1.0字段52（例如在具有格式代码“0”，“1”，和“4”至“9”的指令中）时，存在这种情况，常数被存储在常数寄存器36中（例如在具有格式代码“0”，“1”，“4”，和“5”的指令中）和常数形成在第一操作字段59或第二操作字段60中的操作数的一部分（例如在具有格式代码“5”，“7”，“8”，“9”和“B”的指令中）的情况。当在P1.0字段52中的常数将被存储在常数寄存器36中的时候，存在这种情况，只有该4位常数被存储（例如在具有格式代码“0”和“1”的指令中），和该常数与位于第一操作字段59或第二操作字段60的二者之一中的12位常数一起存储（例如在具有格式代码“4”和“5”的指令中）的情况。

当用于分支的操作代码“CC”在P1.0字段52中给出时（例如在具有格式代码“2”，“3”，和“A”的指令中），这是指示利用常数寄存器36的存储值作为分支目的的绝对地址或相对地址（位移）的分支操作。

第一操作字段59保存一常数或（a）用于指示在本处理器和外围（存储器）之间不包含数据传送的操作（例如算术逻辑操作或内部寄存器传送）的操作代

码和（b）源和用于操作的目的的操作数的组合这二者之一。

第二操作字段60能保存如上述第一操作字段59的相同内容，而且还能替换地保存（a）用于指示在本处理器和外围之间包含数据传送的操作的（例如存储器—寄存器变换）操作代码和（b）用于操作的操作数的组合。

上述功用的不同操作类型确定的字段停止在用于当前 Von Neumann 类型处理器的假设基础上，从而它不必同时处理两个或更多的分支操作，和只有一个用于在本处理器和外围（处理器）之间提供传输操作数的输入/输出端口。

图2B至2D的指令格式具有下述特征。

首先集中在常数“const”上，可以看出，有用于在常数寄存器36中存储一常数的三个指令类型。

（1）当格式代码是“0”或“1”时：

在这些指令中，位于P1.0字段52中的4位常数被存储在常数寄存器36中。

（2）当格式代码是“4”时：

在该指令中，位于P1.0字段52至P2.2字段55中的16位常数被存储在常数寄存器36中。

（3）在格式代码是“5”时：

当该指令中，位于P1.0字段52和P3.0字段56至P3.2字段58中的16位常数被存储在常数寄存器36中。

第二，对于本处理器，最大三个操作能由单个指令指示，和在此情况下，如从图2B所示三重操作格式所看到的，可以使用下述操作类型组合之一。

（1）一种操作是设置4位常数到常数寄存器36和两个标准操作（当格式代码是“0”或“1”）。

（2）一种操作是利用设置在常数寄存器36中的值，作为绝对地址或相对地址进行分支和两个标准操作（当格式代码是“2”或“3”时）。

如上所述，具有高效字段结构的本处理器的指令能通过32位指令同时指示最大三个操作。

下面描述本处理器的硬件结构。

图4示出本发明处理器的硬件结构的方框图。如上所述，该处理器是VLIW处理器，能平行执行最大三个操作。该处理器的结构可粗略划分为指令寄存器

10，译码器单元20，和执行单元30。

指令寄存器10是32位寄存器，存储从指令取单元39送出的一个指令。

译码器单元20译码在指令寄存器10中保存的指令和根据该译码结果在控制线上完成到执行单元30的输出。译译码器单元20自身能粗略分为格式译码器21和指令译码器22。

指令译码器22的组成有分支译码器23，译码在P1.0字段12中保存的“CC”操作代码并相应地控制PC单元33；第一操作译码器24，译码在P2.0字段13中保存的操作代码并相应地控制第一操作单元37；和第二操作译码器25，译码在P3.0字段16中保存的操作代码并相应地控制第二操作单元38和操作数存取单元40。

格式译码器21译码在P0.0字段11中保存的4位格式代码，以识别在指令寄存器10中保存的作为图2B至2D所示的16种可能的指令格式之一的指令的指令格式。根据译码结果，格式译码器21通过分支译码器23，第一操作译码器24，和第二操作译码器25允许或禁止译码操作，并启动执行单元30的寄存器控制单元32。

格式译码器21，分支译码器23，第一操作译码器24，和第二操作译码器25在一个周期中基本译码一个操作并送出控制信号到执行单元30。这里，连接指令寄存器10同执行单元30的26常数信号线26是用于传送在指令寄存器10中的常数和操作数到执行单元30的总线。

根据译码器单元20的译码结果，执行单元30进行操作，并且是能够平行执行最大三个操作的电路。该执行单元30的组成是，执行控制单元31，PC单元33，寄存器集34，第一操作单元37，第二操作单元38，指令取单元39，和操作数存取单元40。在执行单元30的成份外，寄存器控制单元32，PC单元33，和常数寄存器36的结构更详细地表示在另外图中。

执行控制单元31一般认为是用于根据译码器单元20的译码结果控制在执行单元30中的序号为33至40的各单元的控制电路和线路。该执行控制单元31包括在处理器中标准提供的各成分，例如具有用于定时控制，操作允许/禁止控制，状态管理，和中断控制，以及具有本处理器特征成份的常数寄存器控制单元32的各种电路。常数寄存器控制单元32完成控制，使得在由格式译码器21给出指示的基础上将在指令寄存器10中保存的4或16位常数“const”

存储在常数寄存器36中。

PC（程序计数器）单元33在分支译码器23的控制下操作，和输出将被译码和执行的下一个指令的在外部存储器（未示出）中的地址到取指令单元39。

指令取单元39经由32位IA（指令地址）总线和32位ID（指令数据）总线，从外部存储器（未示出）取指令组。指令取单元39在内部指令的超高速缓冲存储器中存储取得的指令组，并对应于PC单元33输出的地址，提供该指令到指令寄存器10。

寄存器集34由15个32位普通寄存器35和一个32位常数寄存器36组成。根据第一操作译码器24和第二操作译码器25的译码结果，被存储在这些16个寄存器35和36中的这些值被传送到第一操作单元37和第二操作单元38，在这里，在送到寄存器集34或操作数存取单元40之前进行操作或替换地允许这些值通过。此外，由第一操作单元37和第二操作单元38进行的操作中，在常数寄存器36中存储的值也可传送到PC单元33，在这里它被用于产生用作为分支目的的有效地址。

第一操作单元37内部包括一用于在两个32位数据集的基础进行算术逻辑操作的ALU（算术逻辑单元）和用于在两个数据集的基础完成乘法的乘法器。该第一操作单元37在第一操作译码器24的控制下能执行两类操作（即，算术逻辑操作，和内部寄存器变换操作）。

第二操作单元38如第一操作单元37相同方法内部包括用于在两个32位数据集的基础上进行算术逻辑操作的ALU和用于在两个32位数据集的基础上完成乘法的乘法器。该第二操作单元38在第二操作译码器25的控制下能执行两类操作（即，算术逻辑操作，和内部寄存器变换操作）。

操作数存取单元40在第二操作译码器25的控制下操作，它是在寄存器集34和外部存储器（未示出）之间传输操作数的一个电路。操作数存取单元40内部包括用于存储操作数和操作数地址的缓冲器。作为一个特例，当操作代码“ld”是在指令寄存器10的P3.1字段16中时，位于外部存储器中的一数据字经由操作数存取单元40加载到寄存器集34中的一个寄存器中。当当前是操作代码“st”时，与此同时，在寄存器集34中的一个寄存器中的存储值被存储在外部存储器中。

PC单元33，寄存器集34，第一操作单元37，第二操作单元38，和操作

数存取单元40由图4所示的内部总线（L1总线，R1总线，L2总线，R2总线，D1总线，和D2总线）相连接。在这里，L1总线和R1总线的每个连接到第一操作单元的两个输入端口的各自的一个，L2总线和R2总线的每个连接到第二操作单元38的两个输入端口的各自的一个，和D1总线和D2总线被分别连接到第一操作单元37和第二操作单元38的输出端口。

常数寄存器36的详细结构和它的外围。

以下详细描述常数寄存器36的结构和外围电路。

图5是常数寄存器36和外围电路的详细结构的方框图。这里注意，用于进位常数“0”的4个信号线的固定线路的附图中的固定值（“0”）27。

常数寄存器控制单元32由5个3输入选择器32a-32e和3个4输入选择器32f-32h组成，而常数寄存器36由8个4位寄存器36a-36h组成。这里，输入和输出数据的每集是4位平行数据。

根据来自格式译码器21和指令译码器32的控制信号，常数寄存器控制单元32控制8个输入选择器32a-32h，使得在指令寄存器10中存储的常数或零，根据以下的4种存储方法中的一种，存储到常数寄存器36中。

图6A至6D示出本实施例的4种可能的存储方法。

图6A示出当格式译码器21检测到在P0.0字段11中存储的值是“0”或“1”时的存储方法。这等于是当在P1.0字段12中只有4位常数被存储在常数寄存器36中的情况。特别是，在常数寄存器36中存储的数据被向上（图6A中向左）移动4位单元和在指令寄存器10的P1.0字段12中存储的4位常数被存储在常数寄存器36的最低序4位寄存器36h。

图6B示出当格式译码器21检测到在P0.0字段11中的存储值是“4”的存储方法。这等于是当在P1.0字段12和P2.2字段15之间的16位常数被存储在常数寄存器36中的情况。特别是，在常数寄存器36的低16位36e-36h中存储的数据被向上移动16位36a-36d，和在指令寄存器10中的P1.0字段12和P2.2字段15之间16位常数被存储在常数寄存器36的最低序16位36e-36h。

图6C示出当格式译码器21检测到在P0.1字段11中存储值是“5”的情况。这等于是当在P1.0字段12和P3.0字段16与P3.2字段18之间的16位常数被存储在常数寄存器36中的情况。特别是，被存储在常数寄存器36的低16位36e-36h的数据被向上移动16位36a-36d和在指令寄存器10的P1.0字段12

和P3.0字段16与P3.2字段18之间的16位常数被存储在常数寄存器36的最低序16位36e-36h中。

图6D示出当格式译码器21检测到在P0.0字段11中存储值是“2”，“3”，或“A”时，或者当指令译码器22检测到，由P2.1字段14，P2.2字段15，P3.2字段17，和P3.3字段18中的至少一个指示常数寄存器（R15）时的存储方法。这等于是在常数寄存器36的存储值已经由在P1.0字段12中的分支操作，在第一操作字段59中的一操作或在第二操作字段60中的一操作的至少一个操作之后，在常数寄存器36中的存储值被全部复位到零（那就是说，常数寄存器36被清零）的情况。特别是，在常数寄存器36的存储值已经被读出到PC单元33，第一操作单元37或第二操作单元38中的一个之后，立即将具有“0”值的32位常数写入常数寄存器36。

这里，在保证使用具有零扩展的值总是被存储在常数寄存器36中之后，常数寄存器36中的值被清零。零扩展呈现为，当值的有效位数低于予定位数时执行的零插入，零被插入到较高位的位置，使得值提升到予定位数。

如上所述，当在指令寄存器10的P0.0字段11中的值是“0”，“1”，“4”，或“5”时，总是被存储在常数寄存器36中的常数被移位和新的值被存储。还有，在常数寄存器36中存储的值被读出和使用之后，该存储的值被删除。通过这样做，常数寄存器36能够连续累积常数直到下一次它存储的内容被使用为止。

PC单元33的详细结构

以下详述PC单元33的结构。

图7是PC单元33的详细结构方框图。如图7所示，PC单元33的组成是，固定值（“4”）33a，即永久进位常数“4”的线路，2输入选择器33b，加法器33c，用于存储将被译码和执行的下一个指令的地址的PC（程序计数器）33d，和4输入选择器33e。

在PC单元33中，根据来自译码单元20的控制信号，该选择器33b和33e进行操作，使得选择器33e输出下述三种类型值中的一个到指令取单元39，作为有效地址。

1.4的值被加到PC33d的内容

这对应于当没有分支发生和下一个指令将被按序执行时，这就是说，本指

令的译码结果没有分支操作指示。“4”被读出的理由是，一指令的长度是 4 字节，即 32 位。

2.常数寄存器 36 的内容值被加到 PC33d 的内容

这对应于，当常数寄存器 36 的内容被用作为分支的相对地址时，例如，当分支译码器 23 的译码结果是 P1.0 字段 12 指示分支到一相对地址。

3.作为常数寄存器 36 的内容给的值

这对应于，当常数寄存器 36 的内容被用作为分支的绝对地址，例如，当分支译码器 23 的译码结果是 P1.0 字段 12 指示分支到一绝对地址。

如上所述，PC 单元 33 包括专用加法器 33c，它能直接使用由常数寄存器 36 存储的值，使得能使常数寄存器 36 的存储值作为相对地址或绝对地址并独立于由第一操作单元 37 和第二操作单 38 进行的操作的以平行方式执行的分支执行控制。

处理器的操作

以下描述当译码和执行专门操作时的本处理器的操作。

图 8 是处理 32 位常数的过程举例的流程。首先，确定寄存器 R0 和 R1 的存储值之间的差（步 S80），和由 R2 的存储值乘该结果（步 S81）。32 位常数“0×87654321”（16 进制值“87654321”）然后被加到该结果（步 S82，S83），和最后，寄存器 R0 被清零（步 S84）。

图 9 是本处理器已经完成图 8 所示过程的程序举例。该程序包括三个指令 71-73。在图 9 中，一行对应一指令，和每个指令的内容由每个指令的分别字段中的助记符表示。在图 9 中，每个常数的值以 16 进制表示。还有，代号“fmtn（n=o-F）”表示格式代码“n”，而代号“Rn（n=0-15）”表示在寄存器集 34 中的一个寄存器存储的值。所有这些，“R15”表现为常数寄存器 36。

图 10 是执行图 9 中所示程序时，本处理器操作的定时图。图 10 示出时钟周期，普通寄存器 R0-R3 和寄存器 R15 的内容，和在 4 总线 L1，R1，L2，和 R2 上流动的数据。

以下参照图 9 和 10，解释本处理器用于指令 71 至 73 的操作。

指令 71

在指令 71 已经被装入指令寄存器 10 之后，本处理器执行图 10 中时钟周期 t0-t1 的操作。格式译码器 21 从指令寄存器 10 中的 P0.0 字段 11 的值“fmt4”

判断本指令是具有格式代码“4”的双操作指令，这样控制执行单元 30，使得以平行方式执行下述的两个操作。

1.第一操作

常数寄存器控制单元 32 控制它的 8 个内部选择器 32a-32h，使得位于 P1.0 字段 12 至 P2.2 字段 15 之间的 16 位常数（0×8765），根据图 6B 所示存储方法，存储到常数寄存器的低 16 位中。相应地，如图 10 中的时钟周期 t0-t1 所示，寄存器 15 的内容从“0×000000000”变化到“0×00008765”。

2.第二操作

第二操作单元 38 接收普通寄存器 R0 的存储值“0×33333333”和普通寄存器 R1 的存储值“0×22222222”的输入，和从前者减去后者之后，在普通寄存器 R0 中存储该结果。作为一种结果，普通寄存器 R0 的存储内容，在图 10 所示时钟周期 t0-t1 中，从值“0×33333333”变化成值“0×111111111”。

指令 72

接着，在指令 72 已经被装入指令寄存器 10 中以后，本处理器进行图 10 中时钟周期 t1-t2 所示的操作。格式译码器 21 从指令寄存器 10 中的 P0.0 字段 11 的值“fmt4”中判断，本指令是具有格式代码“4”的双操作指令，和控制执行单元 30，使得能以平行方式执行下述的两个操作。

1.第一操作

常数寄存器控制单元 32 控制它的 8 个内部选择器 32a-32h，使得位于 P1.0 字段 12 至 P2.2 字段 15 之间的 16 位常数（0×4321），根据图 6B 所示的存储方法，存储在常数寄存器 36 的低 16 位中。相应地，如图 10 中时钟周期 t1-t2 所示，寄存器 R15 的内容从“0×00008765”变化到“0×87654321”。

2.第二操作

第二操作单元 38 接收普通寄存器 R2 的存储值“0×00000004”和普通寄存器 R0 的存储值“0×11111111”的输入，和在存储普通寄存器 R0 中存储该结果之前，这二者相乘。其结果是，如图 10 所示的时钟周期 t1-t2，普通寄存器 R0 的存储值从值“0×11111111”变化到值“0×44444444”。

指令 73

接着，在指令 73 被装入到指令寄存器 10 中之后，本处理器进行如图 10 的时钟周期 t2-t3 所示的操作。格式译码器 21 从指令寄存器 10 的 P0.0 字段 11

的值“fmt7”中判断，本指令是具有格式代码“7”的双操作指令，和控制执行单元30，使得以平行方式执行下述的两个操作。

1.第一操作

第一操作单元37接普通寄存器R15的存储值“0×87654321”和普通寄存器R0的存储值“0×44444444”的输入，和在普通寄存器R0中存储该结果之前，使二者相加。其结果是，如图10所示的时钟周期t2-t3中，普通寄存器R0的存储内容从值“0×44444444”变化到值“0×CBA98765”。

2.第二操作

第二操作单元38接收位于P1.0字段12和P3.1字段17中的8位常数(“0×00”)的输入，并允许该常数通过，使得它被存储在普通寄存器R3中。其结果是，如图10中的时钟周期t2-t3所示的，普通寄存器R3的内容从原先保存的值“0×FEDCBA98”变化到“0×00000000”。

如上所述的本处理器，32位常数“0×87654321”被分离成两部分，即被规范成两个指令71和72，通过移位它的存储值，这些部分被连续存储在常数寄存器36中。然后根据第三指令，指令73，使用该存储的常数，通过此方法，图8流程图所示过程能由三个指令71-73执行。

以下解释利用涉及16位常数的不同程序的本处理器的操作。

图11示出处理16位常数的程序举例。该程序由5个指令74至78组成。

以下描述指令74至78的每一个的本处理器的操作。

指令74

当指令74已经被装入指令寄存器10时，格式译码器21从指令寄存器10中的P0.0字段11的值“fmt0”中判断，本指令是具有格式代码“0”的三重操作指令，和控制执行单元30，使得能以平行方式执行下述的三个操作。

1.第一操作

常数寄存器控制单元32控制它的8个内部选择器32a-32h，使得位于P1.0字段12中的4位常数，根据图6A所示存储方法，存储到常数寄存器36的最低4位中。

3.第二操作

第一操作单元37接收普通寄存器R6的存储值的输入，和允许该值通过，使得它被存储在普通寄存器R1中。

3.第三操作

以相同方法，第三操作单元38接收普通寄存器R7的存储值的输入，并允许该值通过，使得它被存储在普通寄存器R2中。

指令75

当指令75已经被装入指令寄存器10时，格式译码器21从在指令寄存器10中的P0.0字段11的值“fmt0”中判断，该指令是具有格式代码“0”的三重操作指令，和控制执行单元30，使得能以平行方式执行下述的三个操作。

1.第一操作

常数寄存器控制单元32控制它的8个内部选择器32a-32h，使得位于P1.0字段12中的4位常数（“0×7”），根据图16A所示存储方法，存储到常数寄存器36的最低4位。在操作之后，常数“0×87”被设置在常数寄存器36的最低8位。

2.第二操作

第一操作单元37接收普通寄存器R0和普通寄存器R1的存储值的输入，并将这些值相加。第一操作单元37在普通寄存器R1中存储加的结果。

3.第三操作

以相同方法，第二操作单元38接收普通寄存器R0和普通寄存器R2的存储值的输入，和将这些值相加。第二操作单元38在普通寄存器R2中存储加的结果。

指令76，77

指令76，77以上述相同方法执行，和作为结果“0×8765”被存储在常数寄存器36的低16位中。

指令78

一旦指令78被装入指令寄存器10，本处理器就以如处理指令73相同方法操作。

如上所述的本处理器，16位常数“0×8765”被分离成4部分，规范成指令74-77，通过移位它的存储值，这些部分被连续存储在常数寄存器36中。该被存储的常数依照第五指令，指令78被使用。

同标准处理器比较

以下描述由标准处理器对如图9和11所示相同处理内容的程序执行的处

理与本发明的处理相比较。这里，表示“标准处理器”的处理器执行指令的字长固定为 32 位。和本处理器相同，除了缺乏一种结构以外，例如用于累积已经分派在指令中间的常数的常数寄存器 36 和常数寄存器控制单元 32。

图 12A 示出由标准处理器执行的指令的字段定义，而图 12B 示出指令格式。假设标准处理器能执行三类双操作指令，指令 101-103，和一类单操作指令，指令 104。

图 13 示出由标准处理器执行的程序举例。该程序具有如图 9 所示程序的相同处理内容，就是说与图 8 所示流程的过程相同。

如通过比较图 13 和图 9 所能看到的，用于标准处理器的程序比用于本发明处理器的程序的指令多两个。

被包括在指令 105 和 106 中的“NOP 代码”的原因是指令 106 使用指令 105 的操作结果，使得这些指令不能以平行方式执行。还有，由于常数“0×87654321”被分成设置在常数寄存器 R1 的上 16 位和下 16 位（指令 107 和 108），它就不能在一单个 32 位指令中为建立指令，设置 32 位常数和操作代码。

图 14 还示出标准处理器的程序举例。该程序与图 11 所示程序具有相同处理内容。如通过比较图 14 和图 11 所看到的，用于标准处理器的程序比本发明处理器的程序的指令多一个。

如上所述，由本发明处理器执行的指令具有高效字段结构，从而能利用相对短的 32 位字长指示最多三个操作。

相应地，采用本发明的处理器，已经被交叉划分成多个指令的的 16 位或 32 位常数能在常数寄存器 36 中累积，以复原它原始形式的常数，然后用它进行分支操作或算术逻辑操作。相应地，当在指令中一小区域是有效的时，该区域能被有效用于放置常数的一部分，使得程序的代码尺寸与当相同处理由标准处理器执行时的相比较被降低了。

修改

图 15A 至 15D 示出本发明的各种修改的 VLIW 处理器的指令格式。在这些图中，由垂直线划分的最小间隔代表 1 位，而代号“fmt”示出格式字段。

图 15A 所示指令由 5 位格式字段，7 位操作字段，和两个 10 位操作字段组成。同时，图 15B 所示指令由 2 位格式字段，4 位操作字段，和两个 13 位操作字段组成。图 15C 所示指令由 3 位格式字段，3 位操作字段，和两个 13 位操

作字段组成。最后，图15D所示指令由4位格式字段，2位操作字段，和两个13位操作字段组成。

这4类指令类似于图2A所示上述实施例的指令50。

i.指令字长固定为32位。

ii.每个指令具有一个格式字段和三个操作字段。

iii.三个操作字段不具有相同的结构，就是说，两个操作字段具有相同结构，而一个剩余操作字段是短的。

其结果是，这4类指令具有如上述实施例描述的指令50的相同特性。

I.该指令具有一字段结构，从而能以尽管相对短的32位字长指示最多三个操作。

II.提供的小操作字段，是理想的插入小指令，例如，不需要两个操作数的分支指令，如此，该指令的代码效率是高的。

III.提供一格式字段，通过给出在一操作字段中存在一常数或常数的一部分的指示，其中在该操作字段中，NOP指令是提供的正常需要，该程序的代码尺寸能被降低。

另一方面，上述4类指令还具有不同于上述实施例中指令50的特性。图15A所示指令具有放大的格式字段的优点，意味着能确定较大数量的指令类型，和能够在三个操作字段中的每一个中提供至少一个操作数。同时，在图15B至15D所示指令，具有两个放大的操作代码的优点（“OP2”和“OP3”），使得能确定较大变化的操作。

图16是用于执行图15A所指令的本发明的VLIW处理器的结构的方框图。如通过与图4所示结构相比较能看到的，处理器的基本结构是相同的，虽然在指令寄存器10，110和译码器单元20，120之间的连接中有某些差别。以此方法，用于执行图15A至15D中所示修改的指令的VLIW处理器，只通过局部改变上述实施例中所述的VLIW处理器就能实现。

本发明的处理器已经通过上述实施例给出解释，虽然应该清楚，若干进一步的修改是可能的。以下给出这样的4例。

（1）该实施例和本发明的上述修改的处理情况是指令字长32位指示最多三个操作，虽然本发明并不限于这些数。

作为一种举例，图2A所示指令50可以进一步包括另外4位操作代码和另

一 4 位操作数，使得总的指令字长 40 位。通过这些，它变得可能确定具有高代码效率的指令，从而通过具有 40 位相对短字长的单个指令能执行最多 4 个操作。

（2）上述实施例的指令 50 只包括使用隐含操作数（常数寄存器 36 的存储值）的一个字段（P1.0 字段 52），虽然本发明并不限于此，可以有两个或多个这样的字段。通过近似确定新的指令格式，可以对此进行处理。

（3）在处理给定数字常数的举例的上述实施例中，虽然处理过程可能等价于本发明的处理字符常数。这是因为被分离成多个指令的长字符常数能在常数寄存器 36 中通过连续存储字符常数的不同部分而被累积。

（4）如从上述实施例的图 2B 至 2D 中所示的指令格式中能看到的，只有 4 位或 16 位常数能通过单指令被存储在上述实施例的常数寄存器 36 中，虽然这并不是对本发明的限制。作为举例，确定指令格式可能是等价的，从而 12 位或 28 位常数能通过单个指令被存储在常数寄存器中。为做到这些，只有必须改变常数寄存器 36 的外围电路的连接模式。

虽然参照附图通过举例已全面描述了本发明，应注意各种变化和修改对本领域技术人员是明显的。因此，除非该变化和修改脱离开本发明的范围，否则应包括在本发明的范围之内。

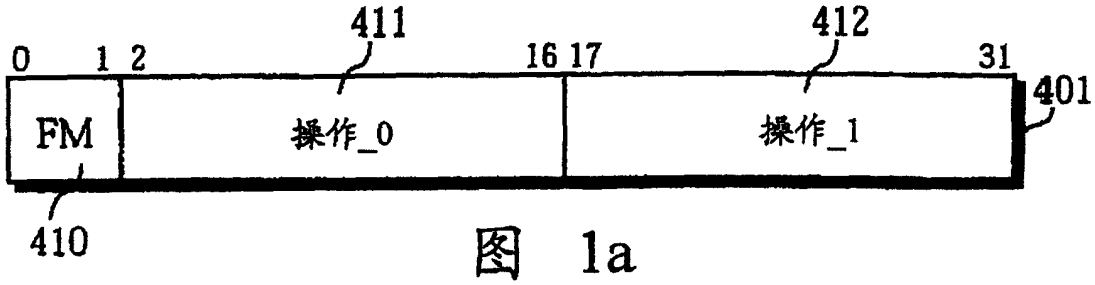


图 1a

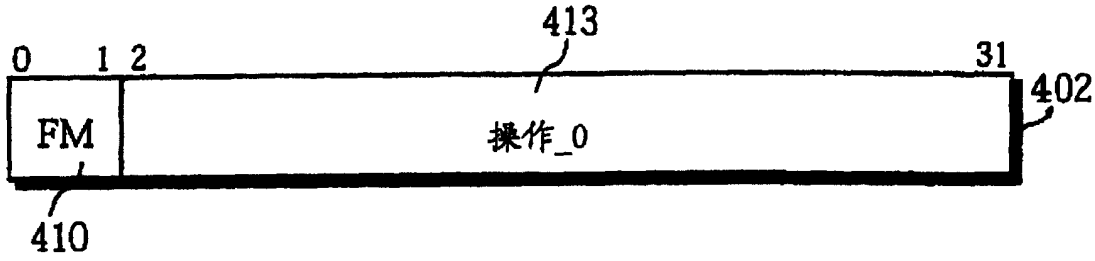


图 1b

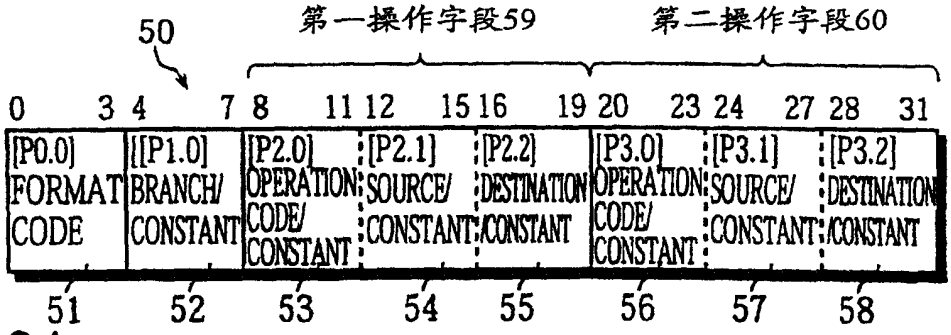


图 2A

0	const	op1	Rs1	Rd1	op2	Rs2	Rd2
1	const	op1	const1	Rd1	op2	Rs2	Rd2
2	cc	op1	Rs1	Rd1	op2	Rs2	Rd2
3	cc	op1	const1	Rd1	op2	Rs2	Rd2

图 2B

4	const	const	const	const	op2	Rs2	Rd2
5	const	op1	Rs1	Rd1	const	const	const
6	const1	op1	const1	Rd1	op2	Rs2	Rd2
7	const2	op1	Rs1	Rd1	op2	const2	Rd2
8	const1	op1	const1	Rd1	op2	const2	Rd2
9	const2	op1	const1	Rd1	op2	const2	Rd2
A	cc	const2	const2	const2	op2	const2	Rd2

图 2C

B	const2	const2	const2	const2	op2	const2	const2
C	保留						
D	保留						
E	保留						
F	保留						

图 2D

符号	操作	助记符
cc	分支	eq, eqi, ne, nei, gt, gti, • • •
op1	ARITHMETIC LOGIC OPERATION	add, sub, mul, and, or, • • •
	INTER-REGISTER TRANSFER	mov, movh, movb
op2	ARITHMETIC LOGIC OPERATION	add, sub, mul, and, or, • • •
	INTER-REGISTER TRANSFER	mov, movh, movb
	REGISTER-MEMORY TRANSFER	ld, ldh, ldb, st, sth, stb

图 3

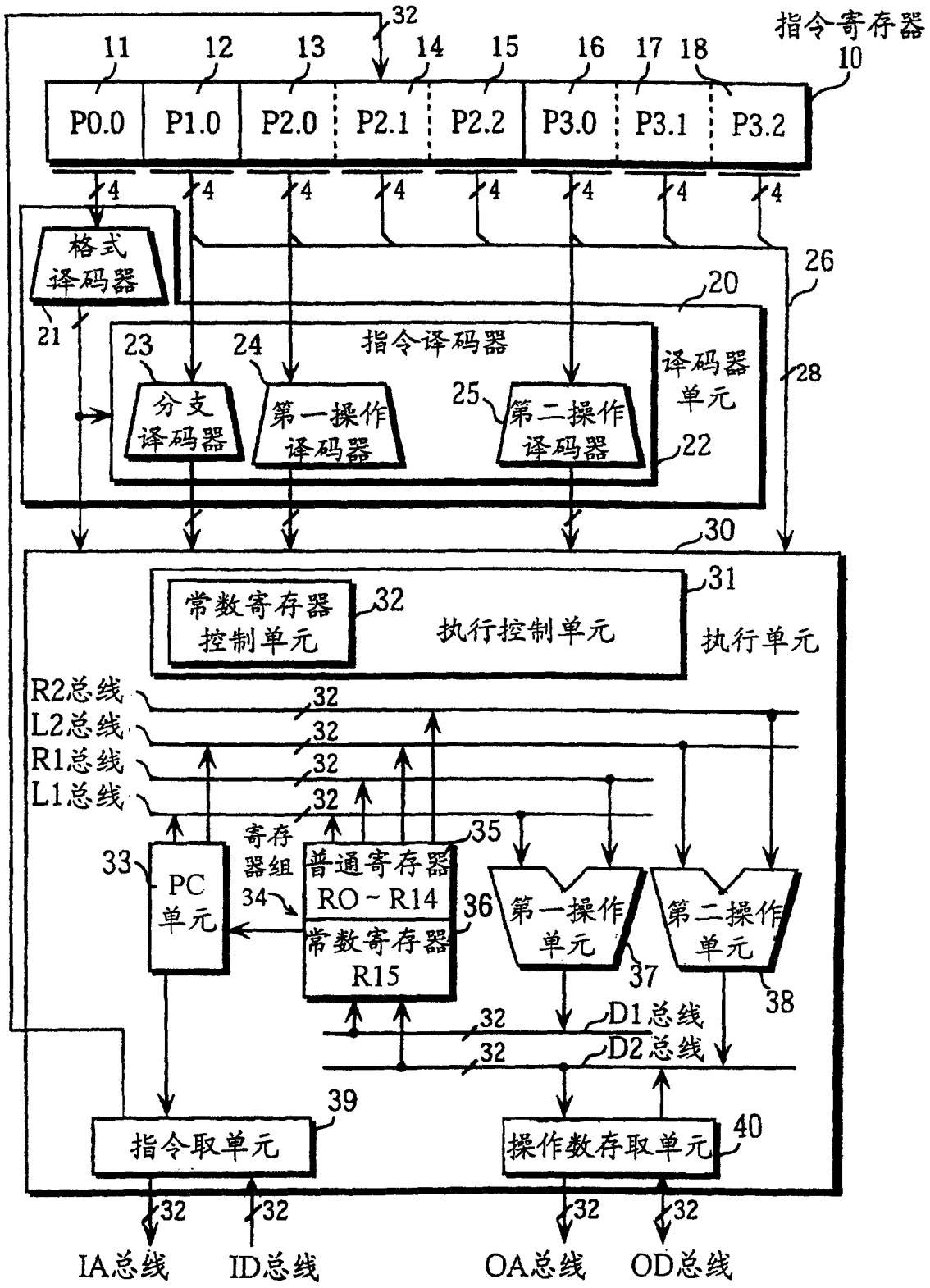

指令寄存器
10
32
11
12
13
14
15
16
17
18
P0.0
P1.0
P2.0
P2.1
P2.2
P3.0
P3.1
P3.2
4
格式
译码器
21
20
26
指令译码器
23
24
25
分支
译码器
第一操作
译码器
第二操作
译码器
22
译码器
单元
28
30
31
常数寄存器
控制单元
32
执行控制单元
执行单元
R2总线
L2总线
R1总线
L1总线
寄存
器组
34
33
PC
单元
普通寄存器
R0~R14
常数寄存器
R15
35
36
第一操作
单元
第二操作
单元
37
38
D1总线
D2总线
39
指令取单元
操作数存取单元
40
IA总线
ID总线
OA总线
OD总线

图 4

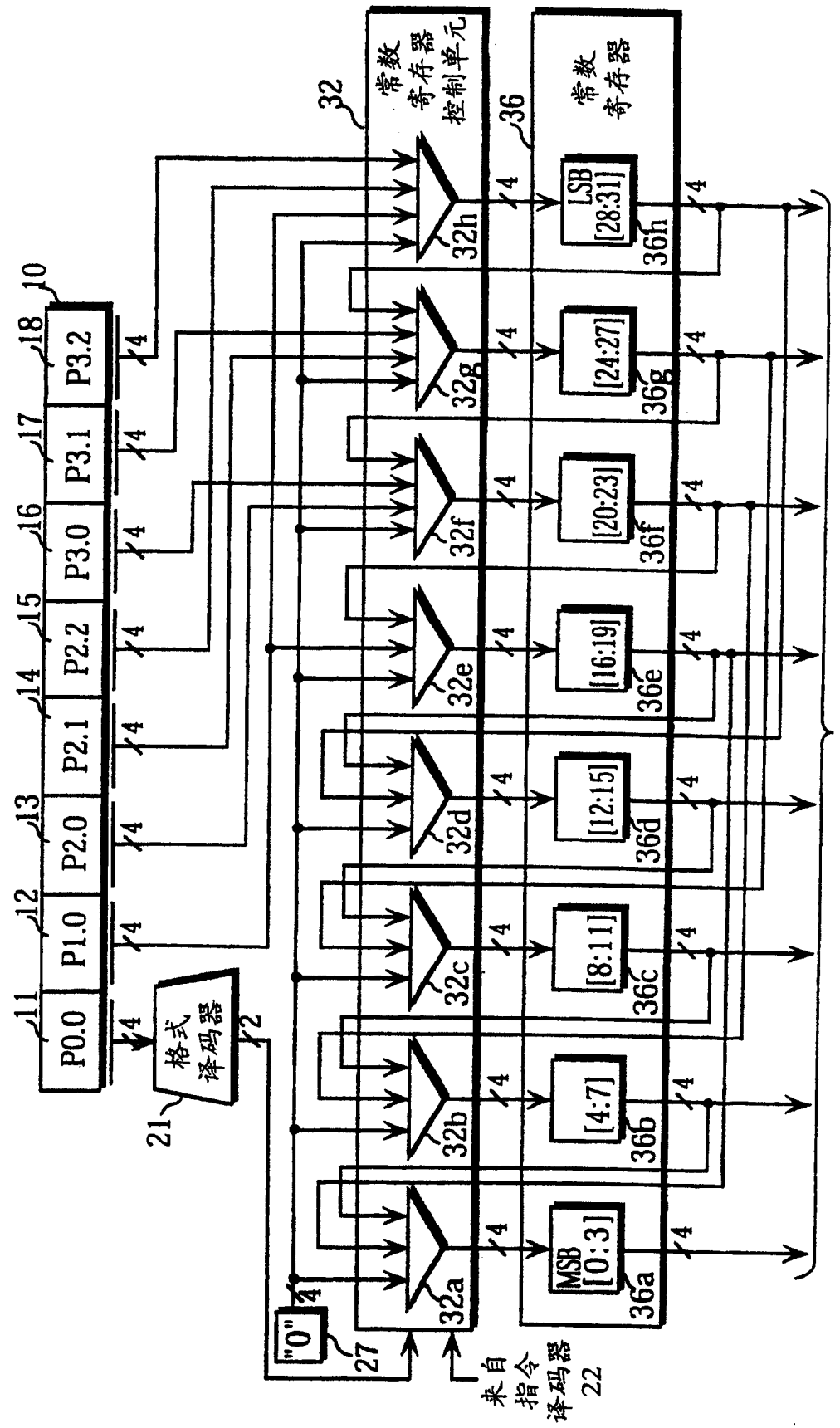

常数寄存器控制单元
常数寄存器
32
36
P0.0
P1.0
P2.0
P2.1
P2.2
P3.0
P3.1
P3.2
格式译码器
"0"
来自指令译码器
22
MSB [0:3]
[4:7]
[8:11]
[12:15]
[16:19]
[20:23]
[24:27]
LSB [28:31]
TO PC单元33，L1总线，R1总线，L2总线，R2总线

图 5

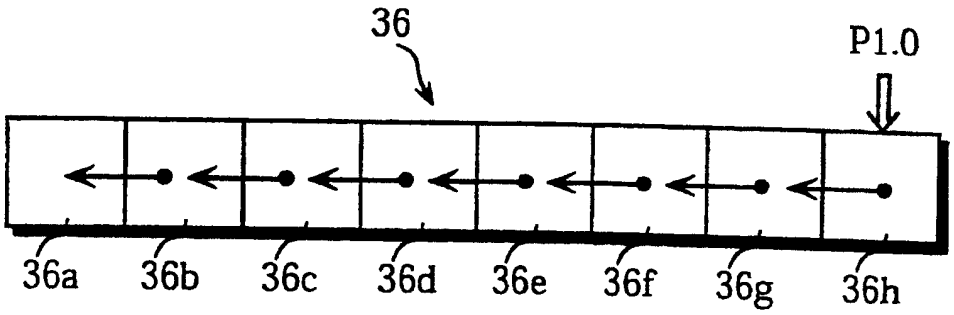


图 6A

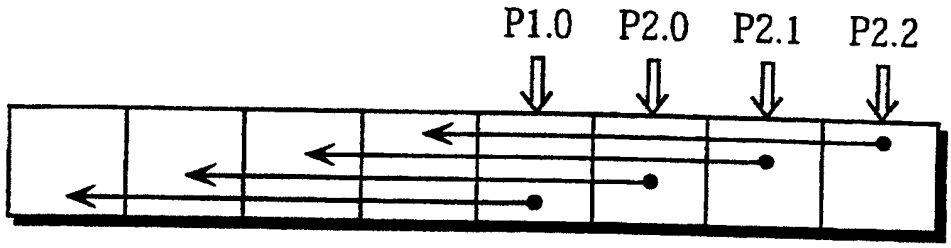


图 6B

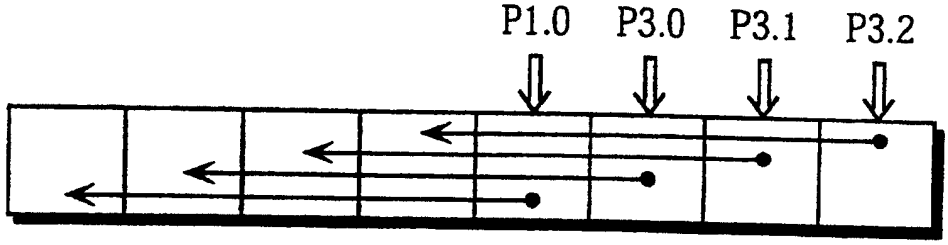


图 6C

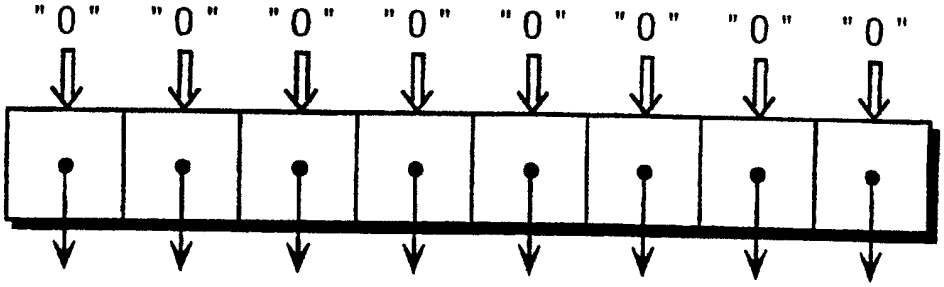


图 6D

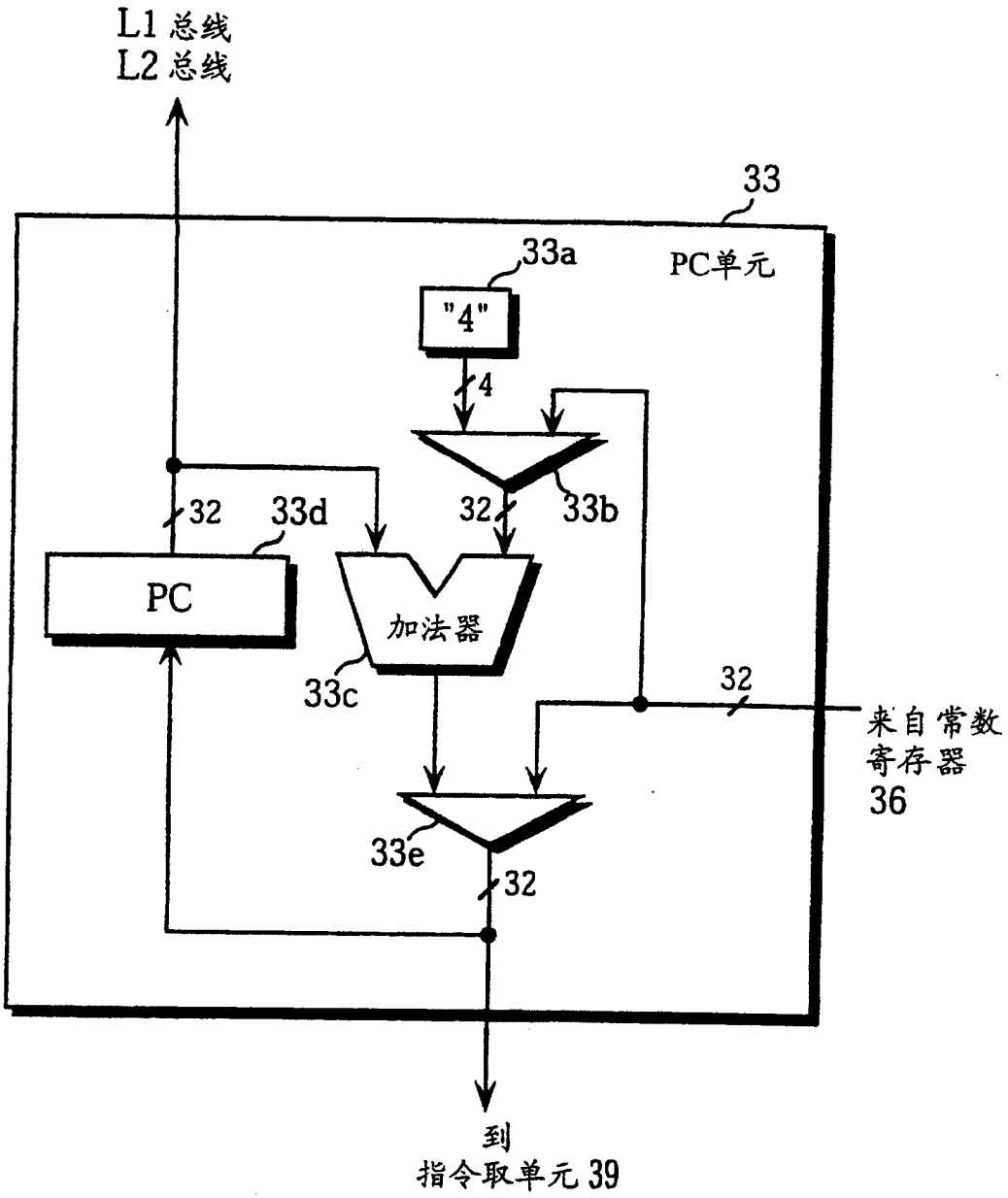

L1 总线
L2 总线
33
33a
PC单元
"4"
4
32
33d
32
33b
PC
加法器
33c
32
来自常数
寄存器
36
33e
32
到
指令取单元 39

图 7

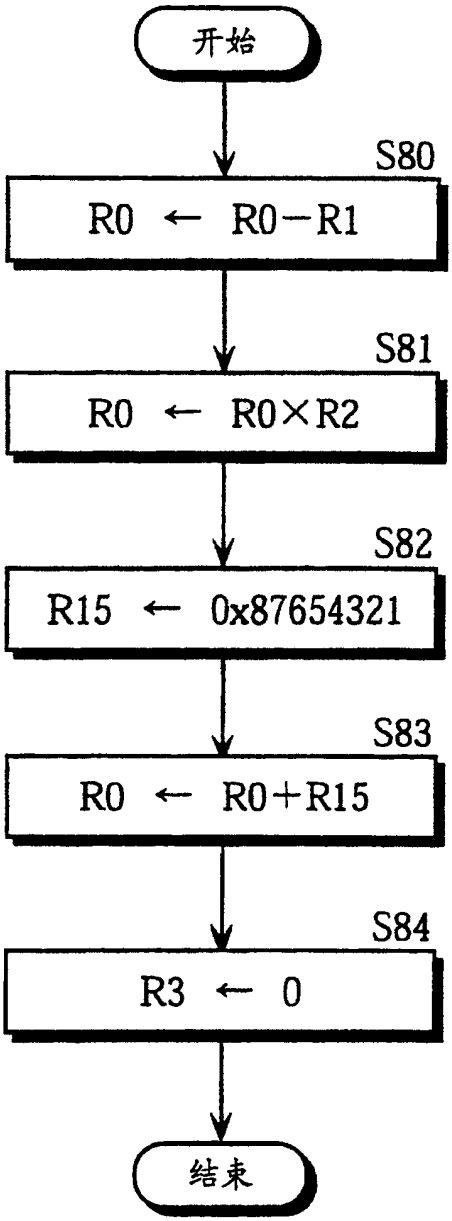

开始
S80
R0 ← R0−R1
S81
R0 ← R0×R2
S82
R15 ← 0x87654321
S83
R0 ← R0+R15
S84
R3 ← 0
结束

图 8

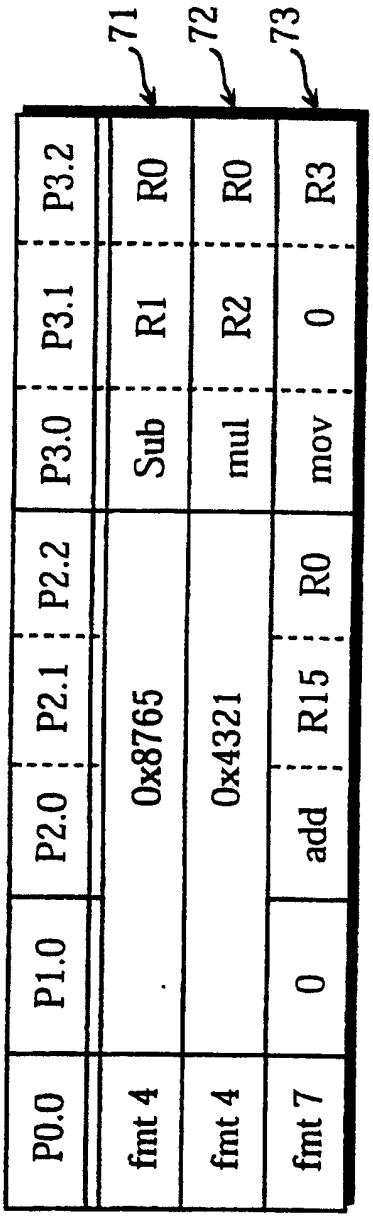

71
72
73
P0.0
P1.0
P2.0
P2.1
P2.2
P3.0
P3.1
P3.2
fmt 4
0x8765
Sub
R1
R0
fmt 4
0x4321
mul
R2
R0
fmt 7
0
add
R15
R0
mov
0
R3

图 9

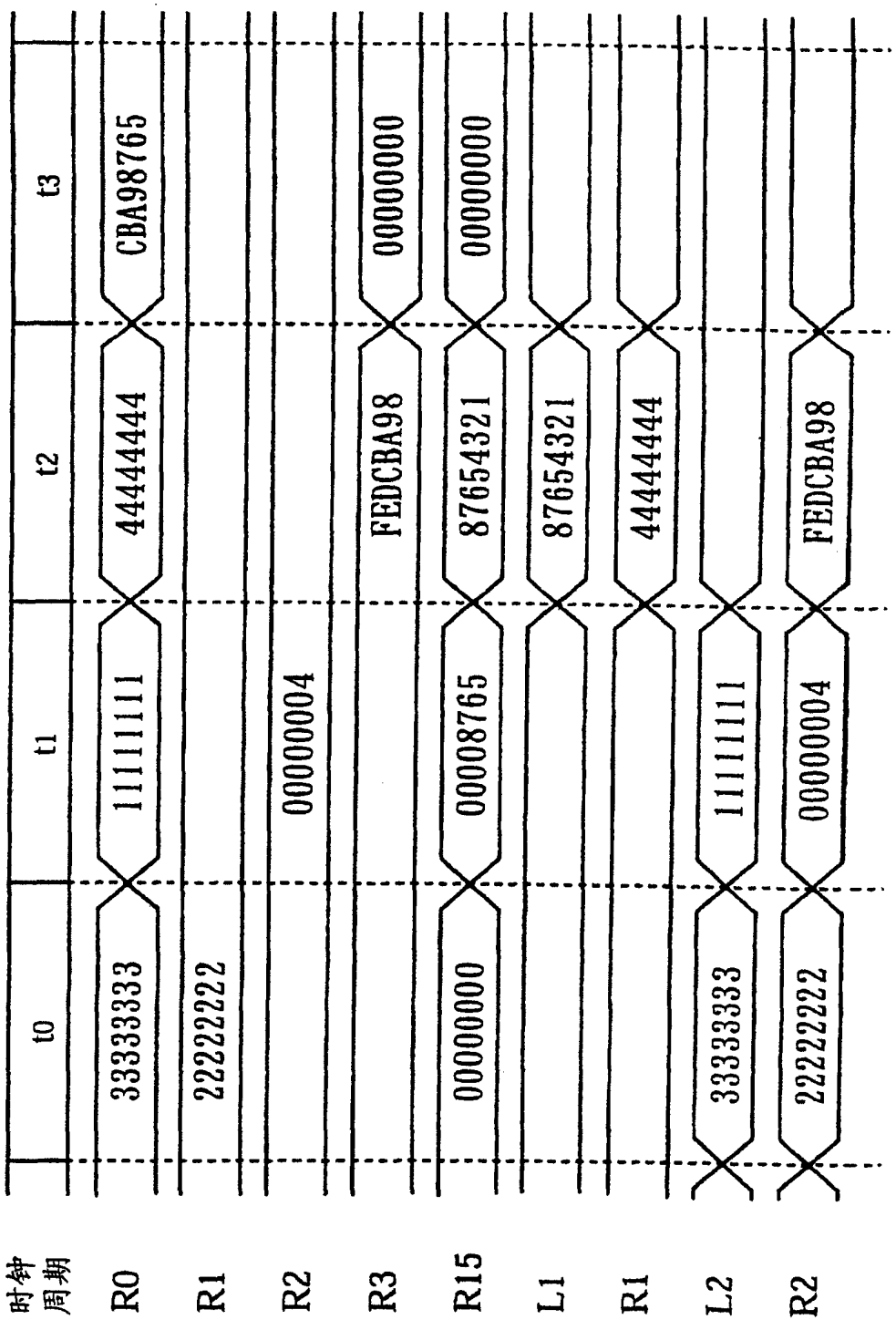

时钟周期
t0
t1
t2
t3
R0
33333333
11111111
44444444
CBA98765
R1
22222222
R2
00000004
R3
FEDCBA98
00000000
R15
00000000
00008765
87654321
00000000
L1
87654321
R1
44444444
L2
33333333
11111111
R2
22222222
00000004
FEDCBA98

图 10

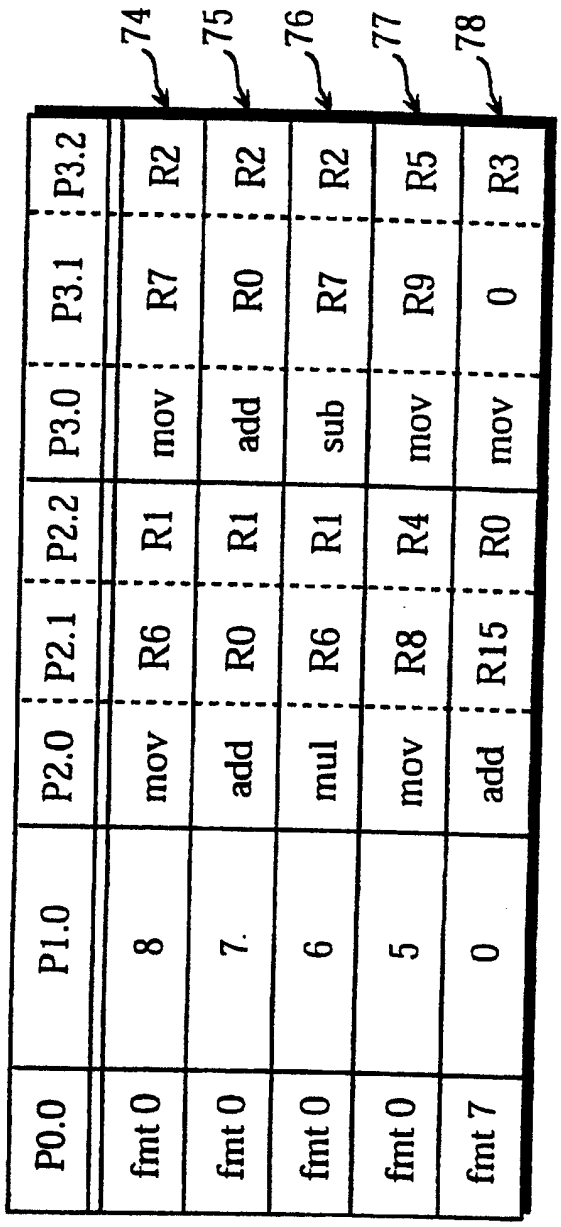

74
75
76
77
78
P0.0
P1.0
P2.0
P2.1
P2.2
P3.0
P3.1
P3.2
fmt 0
8
mov
R6
R1
mov
R7
R2
fmt 0
7.
add
R0
R1
add
R0
R2
fmt 0
6
mul
R6
R1
sub
R7
R2
fmt 0
5
mov
R8
R4
mov
R9
R5
fmt 7
0
add
R15
R0
mov
0
R3

图 11

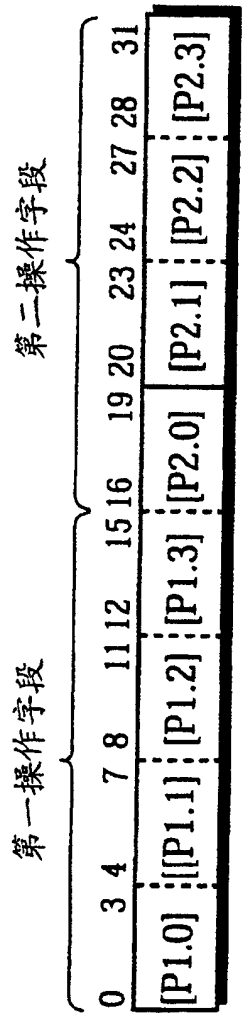

第一操作字段
第二操作字段
0 3 4 7 8 11 12 15 16 19 20 23 24 27 28 31
[P1.0] [[P1.1] [P1.2] [P1.3] [P2.0] [P2.1] [P2.2] [P2.3]

图 12A

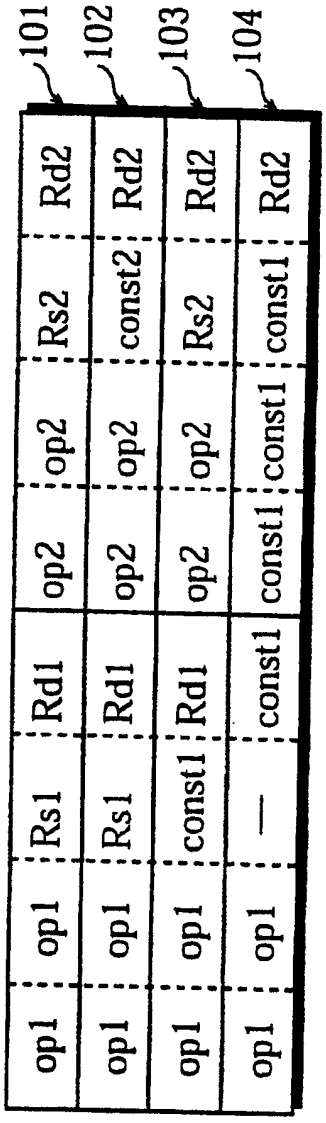

101
102
103
104
op1 op1 Rs1 Rd1 op2 op2 Rs2 Rd2
op1 op1 Rs1 Rd1 op2 op2 const2 Rd2
op1 op1 const1 Rd1 op2 op2 Rs2 Rd2
op1 op1 — const1 const1 const1 const1 Rd2

图 12B

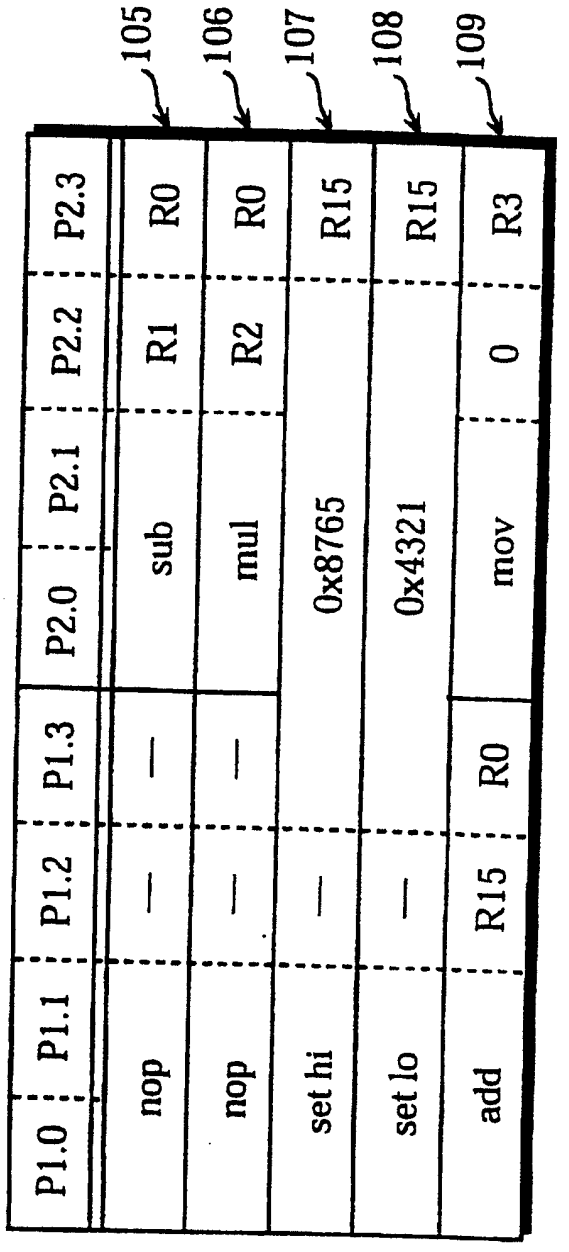

105
106
107
108
109
P1.0
P1.1
P1.2
P1.3
P2.0
P2.1
P2.2
P2.3
nop
—
—
sub
R1
R0
nop
—
—
mul
R2
R0
set hi
—
0x8765
R15
R15
set lo
—
0x4321
R15
R15
add
R15
R0
mov
0
R3

图 13

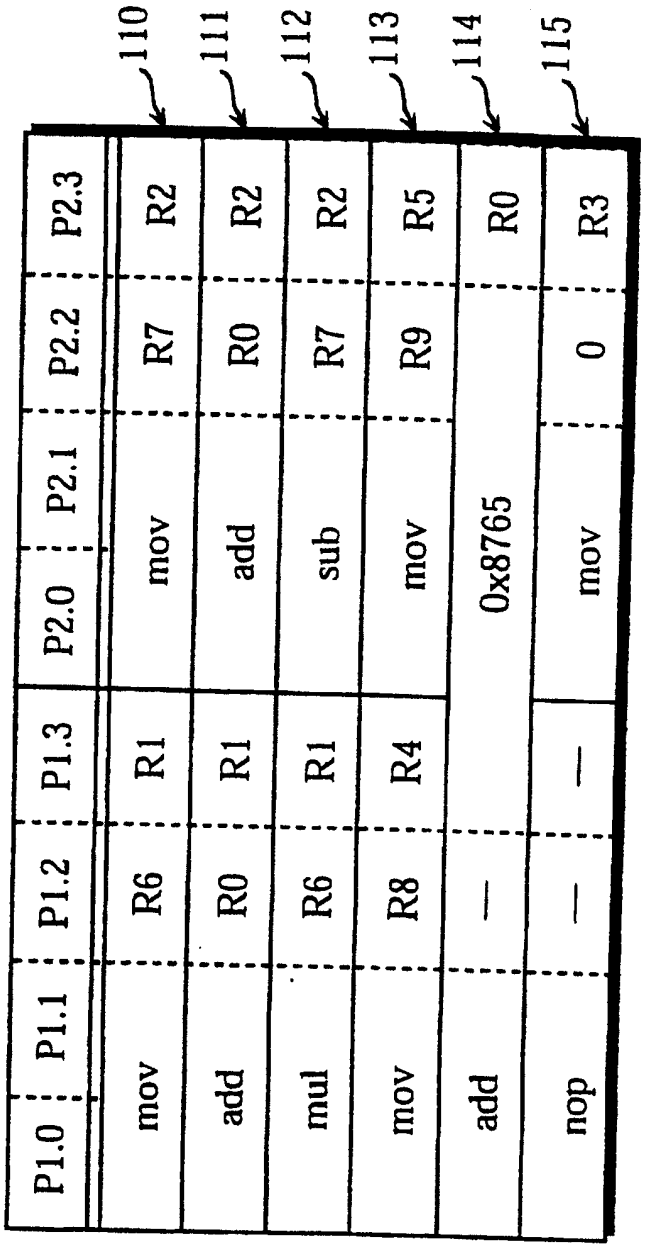

110
111
112
113
114
115
P1.0
P1.1
P1.2
P1.3
P2.0
P2.1
P2.2
P2.3
mov R6 R1 mov R7 R2
add R0 R1 add R0 R2
mul R6 R1 sub R7 R2
mov R8 R4 mov R9 R5
add — — 0x8765 R0
nop — — mov 0 R3

图 14

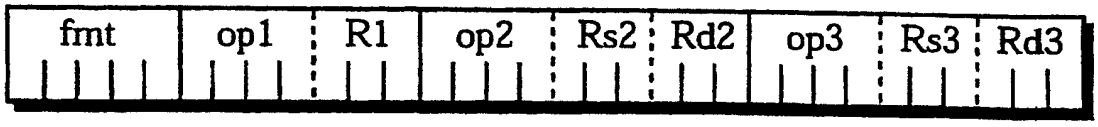


图 15A

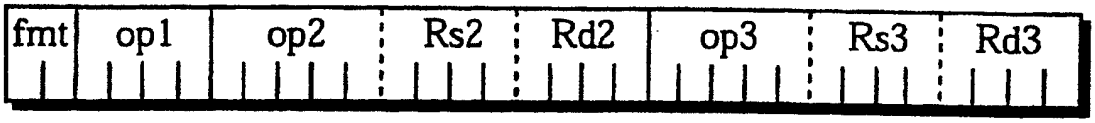


图 15B

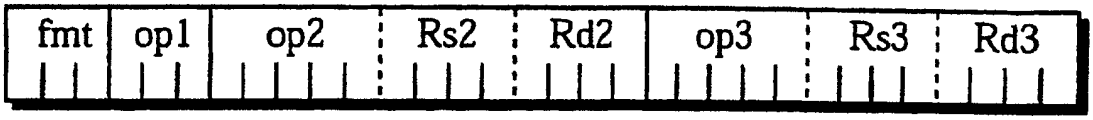


图 15C

图 15D

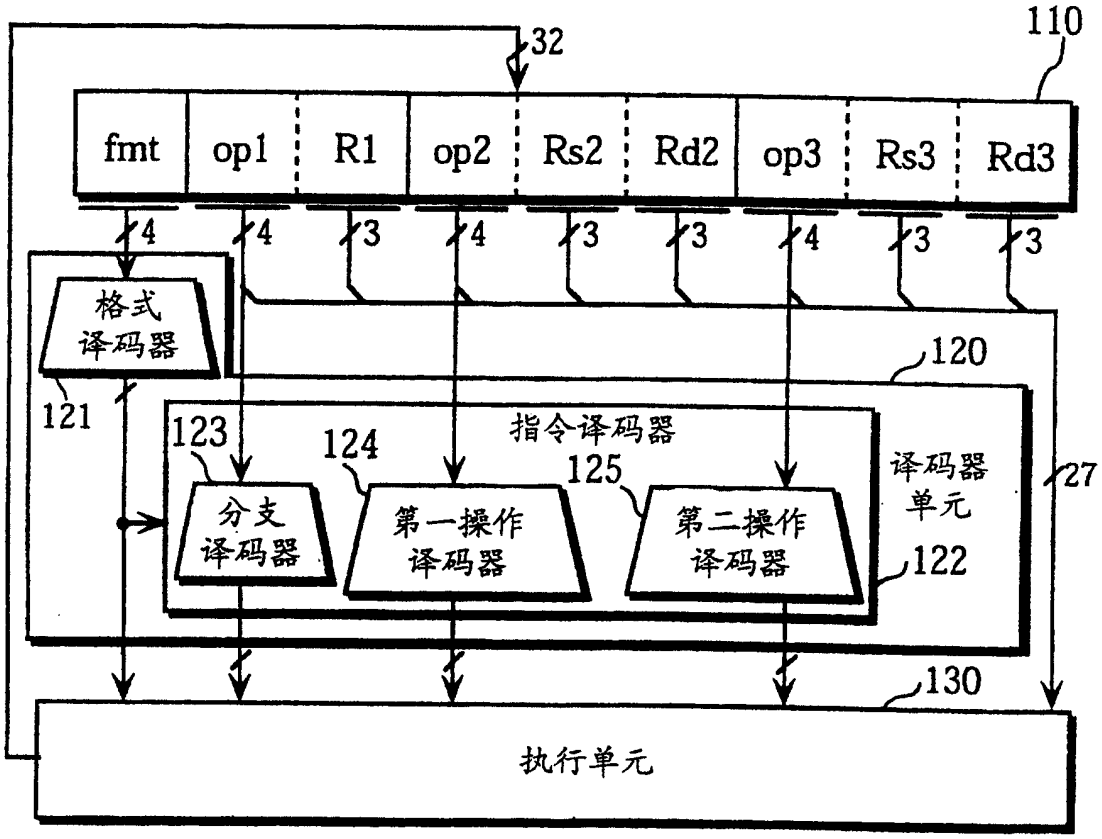

110
32
fmt
op1
R1
op2
Rs2
Rd2
op3
Rs3
Rd3
4
4
3
4
3
3
4
3
3
格式
译码器
121
120
123
124
125
指令译码器
译码器
单元
27
分支
译码器
第一操作
译码器
第二操作
译码器
122
130
执行单元

图 16