[19]中华人民共和国专利局

[51]Int.Cl[6]
G06F 15/00

[12] 发明专利申请公开说明书

[21] 申请号 97110525.1

[43]公开日 1997 年 12 月 24 日

[11] 公开号 CN 1168505A

[22]申请日 97.4.14

[30]优先权

[32]96.4.15 [33]US[31]632,187

[71]申请人 摩托罗拉公司

地址 美国伊利诺斯

[72]发明人 约瑟夫·C·西尔塞罗

杰弗逊·高金科

[74]专利代理机构 中国国际贸易促进委员会专利商标事务所

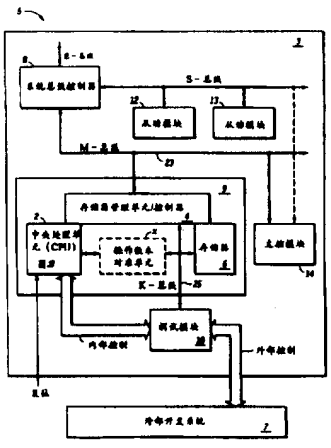
代理人 于 静

权利要求书 2 页 说明书 19 页 附图页数 6 页

[54]发明名称 具有自对准堆栈指针的数据处理系统及其方法

[57]摘要

当堆栈操作未对准的硬件支持作为任选部件时，处理器利用单个系统堆栈指针建立监控器和用户二者的堆栈操作的记录。单个系统堆栈指针作为自对准堆栈指针用于把它自己自动地对准到低于当前地址的最近的模 4 余 0 地址，从而即使在没有硬件支持未对准操作数的情况下也不会发生对准系统错。一旦发生了自动对准，数据处理器在异常事件堆栈帧中存储一个格式字段以指示在出错时刻堆栈指针对准的有关信息，当对异常事件的服务完成时，把堆栈指针恢复到原来值。



权 利 要 求 书

1. 一个数据处理系统的运转方法，其特点在于如下步骤:

接收表明已发生第一预定状态的第一信号;

从堆栈指针寄存器中取出当前地址值;

将当前地址值对准到当前地址值以下的最近的模 4 余 0（ 0 - modulo-4 ）地址; 以及

在一存储器中存储 - 格式字段，该格式字段指明在接收表明已发生第一预定状态的第一信号时当前地址值对准的方式。

2. 权利要求 1 的方法，这里第一预定状态是异常事件处理状态。

3. 权利要求 1 的方法，这里的对准步骤进一步包括步骤:

否定从堆栈指针寄存器中取出的当前地址值的预定部分。

4. 权利要求 1 的方法，其特征还在于如下步骤:

接收表明已发生第二预定状态的第二信号;

从存储器中取出格式字段; 并把当前地址值恢复到堆栈指针寄存器。

5. 一种数据处理器，其特征在于:

总线装置，用于提供多个地址值和多个数据值;

控制装置，用于确定何时已发生操作中的异常事件并响应这一操作异常事件而提供异常事件处理信号;

一个寄存器，用于指明当发生操作异常事件时存取的当前地址值;

一个逻辑电路，与寄存器相连以接收当前地址值的第一部分，与控制装置相连以接收异常事件处理信号，该逻辑电路有选择地修改当前地址值以构成调整后的地址值; 以及

一个格式产生电路与寄存器相连以接收至少是当前地址值的第一部分，该格式产生电路修改当前地址值的第一部分，以产生一个堆栈对准值，用以指示当操作中已发生异常事件时当前地址值的对准。

6. 权利要求 5 的数据处理器，其特点在于一个存储器与格式产生电路相连以存储堆栈对准值。

7. 权利要求 5 的数据处理器，这里，当操作中的异常事件已经发生

并已被处理完毕时，当前地址值被存储于该寄存器。

8. 权利要求5的数据处理器，这里被调整的地址值是一个小于当前地址值的最近一个模4余0的地址。

9. 一种数据处理器，其特点在于:

控制装置，用于产生第一中断信号;

一个堆栈指针寄存器，用于存储和提供当前地址值;

一个逻辑电路，与控制装置相连以接收第一中断信号，还与堆栈指针相连以接收当前地址值的第一部分，该逻辑电路建立一个调整了的地址值；以及

一个转换电路，与堆栈指针寄存器相连，以接收当前地址值的第二部分，该转换电路产生一个格式字，该格式字与调整过的地址值求和时将产生当前地址值。

10. 权利要求9的数据处理器，这里的格式字段存储于数据处理器的存储器中。

11. 权利要求9的数据处理器，这里的格式字段从数据处理器的存储器中取出并用于把当前地址值恢复到堆栈指针寄存器中。

12. 权利要求9的数据处理器，这里被调整的地址值是小于当前地址值的最近的模4余0地址值。

说　　明　　书

具有自对准堆栈指针的
数据处理系统及其方法

本发明一般涉及数据处理系统，特别是涉及数据处理系统中的堆栈指针。

在数据处理系统中，堆栈指针通常是硬件寄存器，用作限定程序变量和临时存储的存储器位置的存储器地址指针。通常，硬件使用程序设计模型中一个或多个寄存器作为系统堆栈指针。在许多系统中，在执行某些操作期间，堆栈指针由指令集隐含使用。通常，该地址寄存器定义了当前堆栈帧的顶，在此操作数通过将数据“推进”存储器堆栈而在各函数之间传送。该推进操作包括将堆栈指针减“ 1 ”，然后将数据拷贝到新堆栈指针地址定义的位置。在例行程序使用了该数据之后，通过简单地将堆栈指针加“ 1 ”使其恢复到原来值，而再分配该数据。该过程称为“弹出”堆栈。

由于这些堆栈指针通常定义了存储器地址，用户程序例行处理限定堆栈顶的堆栈指针内容。各类长度的操作数，如字节、字、长字，都可能被推入堆栈，这样在任何时候堆栈顶都可能指向任何允许的对准。据说在任何操作数地址对准上提供任何操作数长度的自动硬件支持的处理器也支持未对准。如果操作数的存储器地址与其自然边界不对应，则称该操作数未对准。自然边界简单地定义成一个对操作数长度模运算余 0 的地址。例如，如果存储器字节地址是模 2 余 0 则 16 位字操作数是对准的，如果存储器字节地址是模 4 余 0 则 32 位长字是对准的。不符合这种定义的对准会产生未对准操作数。

大多数处理器包括对各种异常事件处理的硬件支持。这些异常事件，也称为故障，通常包括中断、系统调用和算术运算错误条件。也可能发生其他异常事件。通常在检测到异常事件时，处理器将在发生故障时在描述机器环境的系统堆栈顶部产生一特殊的异常事件帧。在产生该异常事件堆栈帧之后，处理器通过将控制传送给异常事件类型确定的指

令地址，而将控制传送给特殊异常事件处理例程。通常异常事件处理例程将处理可能会启动某些校正动作的故障，然后返回到异常事件发生时执行的应用程序。

大多数处理器使用某种类型的出错检测，这样在处理任何其他异常事件期间发生的某些灾难性故障状态会导致整个系统失效从而使处理器停止工作。灾难性故障状态这一特殊的集合包括在某些边界上系统堆栈指针的对准。为了补偿这种可能性，现有技术已实施了一些不同的解决方案，它们要求大量的硬件。例如，大多数现有技术解决方案要求能够确保无操作数未对准的硬件电路。该方案通常要求基于处理器操作模式的多堆栈指针。该操作模式通常定义与执行程序有关的特权级，其最简单形式通常包括两级特权：监督者（最高特权）和用户（最低特权）。在监督方式中执行提供对系统所有资源的访问，而用户方式中执行不能访问某些硬件资源。此外，许多处理器在数据处理系统中提供专用硬件电路来强制所需的堆栈指针的对准。注意，在每种现有技术方案中，都需要专用硬件电路和相应的额外开销。

对于无法实施这种昂贵的硬件电路的数据处理系统，已开发出软件解决方案。在使用这种软件方案时，不允许并发异常事件。换句话说，一旦数据处理系统已经获取了一个异常事件，则不允许其他异常事件。在这种情况下，如果另一异常事件不能被标记和确认，则机器环境的状态和程序计数器信息将被重写。于是，数据处理系统将不能回到处理第一个异常事件之前的状态。在第二种类型的现有技术系统中，数据处理系统无法有效地正确处理所有异常事件。

现有技术的解决方案需要大量额外的硬件电路，以在确认和处理完异常事件之后将数据处理系统恢复到其初始状态。于是需要这样一种解决方案：它使数据处理系统的设计者有选择地包括用于操作数未对准的硬件支持，同时又能保证数据处理系统无论系统堆栈指针是否对准都能正确地处理所有的异常事件并在处理了异常事件之后恢复初始状态。

图 1 以方框图形式显示根据本发明一个实施例的数据处理系统；

图 2 以方框图形式显示图 1 所示数据处理系统的中央处理单元部分；

图 3 以方框图形式显示图 2 所示中央处理单元的取操作数电路;

图 4 以方框图形式显示图 3 所示取操作数电路的寄存器存储器电路（ register file circuit ）;

图 5 以方框图形式显示图 2 所示中央处理单元的地址产生执行电路;

图 6 以方框图形式显示图 4 所示地址产生执行电路的另一个执行机构的一部分;

图 7 以方框图形式显示用于多个截栈指针 SP 寄存器值的一个异常事件堆栈帧结构; 以及

图 8 以方框图形式显示包括 4 位格式字段的一个异常事件堆栈帧的结构。

本发明提供一种数据处理器以及操作一个数据处理系统的方法，在该系统中当把硬件支持堆栈操作数对准作为一种可选功能时，可以使用单个系统堆栈指针来建立系统和用户堆栈操作二者的记录。该单个系统堆栈指针被用作为自对准堆栈指针。简单地说，这个自对准堆栈指针自动地把自己对准到当前地址设置之下的最近的模 4 余 0 （ 0-modulo-4 ）地址，从而即使当没有硬件支持未对准操作数时也不会发生系统错误。一旦发生了自动对准，本发明的数据处理器便在一个异常事件堆栈帧中存储一个 4 位格式字段（ field ），以指示关于出错时堆栈指针对准的信息。当已经对异常事件提供了服务并执行从异常事件返回（ RTE ）指令时，处理器使用这个存储在异常事件堆栈帧中的 4 位格式字段，以把堆栈指针恢复到异常事件发生时它的原来值。

本发明提供一种数据处理器和操作方法，其中使用单一堆栈指针来支持一个数据处理系统的监控和用户二个堆栈区。再有，在本发明中可以采用专用于操作数未对准的硬件，但不是必需的。这一单个堆栈指针对自己对准的能力允许数据处理器保证当发生异常事件而堆栈指针未对准时也不会使数据处理系统出现灾难性错误。下面将更详细地讨论本发明的操作。

在下面的讨论中将更详细地提供采用本发明的数据处理系统的连接和操作情况。

本发明的连接

在下面描述本发明的连接时，将使用“总线（ bus ）”一词代表可能用于传送一种或多种类型信息（诸如数据、地址、控制、或状态）的一组信号或导线。术语“认定（ assert ）”和“否定（ negate ）”将用于表示一个信号、状态位、或类似装置分别进入了它的逻辑真或逻辑伪状态。如果逻辑真状态是逻辑电平 1，则逻辑伪状态将是逻辑电平 0。而如果逻辑真状态是逻辑电平 0，则逻辑伪状态将是逻辑电平 1。

再有，如果在一个数字前冠以“ $ ”符号，则表明这个数是以其 16 进制形式或者说以 16 为基的形式表示的。在数字前冠以“ % ”符号，则表明这个数是以其 2 进制形式或者说以 2 为基的形式表示的。

现在参考图 1，图 1 显示了根据本发明一个实施例的数据处理系统 5。数据处理系统 5 由数据处理器 3 和外部开发系统 7 组成。数据处理器 3 包括系统总线控制器 8、处理器核心 9、从动模块 12、从动模块 13、主控模块 14、以及调试模块 10。系统总线控制器 8 通过 E - 总线与一外部设备（这里未画出）相连。系统总线控制器 8 通过 S - 总线与从动模块 12 及从动模块 13 相连。 S - 总线是从动模块总线。系统总线控制器 8 通过 M - 总线 23 与处理器核心 9 及主控模块 14 相连。 M - 总线 23 是主控模块总线。

处理器核心 9 由中央处理单元（ CPU ） 2 、存储器管理单元（ MMU ）/控制器 4、以及存储器 6 组成。在处理器核心 9 中也可以有选择地包括一个操作数未对准单元 5。中央处理单元 2、 MMU/控制器 4、存储器 6 以及调试模块 10 通过 K - 总线 25 彼此相连。中央处理单元 2 和存储器 6 还直接与 MMU/控制器 4 相连。 MMU/控制器 4 通过 M - 总线 23 向数据处理系统 5 的其余部分提供信息。中央处理单元 2 与调试模块 10 相连以提供内部控制总线。 CPU2 还从外部设备（这里未画出）接收一复位信号。调试模块 10 使用外部控制总线与外部开发系统 7 双向通信。外部开发系统 7 是数据处理系统 5 外部的一个可选设备。

图 2 更详细地显示出 CPU2 部分。 CPU2 包括寄存器 208、取指令流水线（ pipeline ） 210、缓冲器 216、操作数执行流水线 220、以及多路转换器（ MUX ） 222 。取指令流水线 210 由指令地址产生电路

202、取指令电路204、FIFO（先进先出）指令缓冲器206、以及多路转换器219构成。此外，操作数执行流水线220由取操作数电路212、地址产生/执行电路214、以及流水线控制逻辑218构成。

取指令流水线210的指令地址产生电路202向多路转换器222提供指令地址信号。多路转换器222在来自取指令流水线210的指令地址与来自操作数执行流水线220的操作数地址之间进行选择，以用于K总线25上的处理。寄存器208输出一个KADDR（K－总线地址）信号。取指令流水线210的取指令电路204与取操作数电路212及缓冲器216相连，以接收缓冲的KRDATA信号。取指令电路204与取操作数电路212相连以提供取指令地址。取指令电路204还与FIFO指令缓冲器206及多路转换器219相连，以提供指令缓冲器写数据信号。FIFO指令缓冲器206与多路转换器219相连，以提供指令缓冲器读数据信号。多路转换器219与操作执行流水线220的取操作数电路212相连，以提供指令信号。

操作数执行流水线220的流水线控制逻辑218与取操作数电路212相连，以提供异常事件处理信号。流水线控制逻辑218还与地址产生/执行电路214相连，以提供一控制总线。流水线控制逻辑218与取操作数电路212及地址产生/执行电路214相连，以提供向量信号。操作数执行流水线220的地址产生/执行电路214与多路转换器222相连以提供操作数地址信号。再有，地址产生/执行电路214还与缓冲器216相连，以提供操作数写数据信号。缓冲器216提供KRDATA（K－总线读数据）信号和KWDATA（K－总线写数据）信号。KRDATA、KWDATA及KADDR信号都提供给K－总线25。

图3更详述地显示取操作数电路212。取操作数电路212由寄存器224、多路转换器1（MUX）226、寄存器存储器228、程序计数器寄存器229、多路转换器2（MUX）230、多路转换器231、多路转换器3（MUX）232、加法器233、寄存器A234、以及寄存器B236组成。

一个指令信号从图2的多路转换器219提供给寄存器224。图3的寄存器224与多路转换器1 226的第一输入相连，而向量信号与多路转

换器 1　226 的第二输入相连。多路转换器 1　226 向多路转换器 2　230 及多路转换器 3　232 提供常数和立即操作数，如寄存器 224 中存储的指令所指明的。一个执行结果信号与寄存器存储器 228 及多路转换器 2　230 和多路转换器 3　232 每一个相连。寄存器存储器 228 与多路转换器 2　230 相连以提供“A”信号，并与多路转换器 3 232 相连以提供“B”信号。此外，寄存器存储器 228 提供一个 SP（1:0）信号。SP（1:0）信号提供存储在寄存器存储器 228 中的堆栈指针值的位 0 和位 1。多路转换器 2　230 与寄存器 A 234 相连，多路转换器 3　232 与寄存器 B 236 相连。寄存器 A 234 提供一个 RA 信号。寄存器 B 236 提供一个 RB 信号。RA 和 RB 信号作为被寄存的输入操作数提供给地址产生/执行电路 214。

取指令地址信号被提供给多路转换器 231 的第一输入。多路转换器 231 的输出提供给程序计数器寄存器 229。程序计数器寄存器提供一个 PC（程序计数器）信号作为输出信号送给加法器 233 的输入。加法器 233 被用于计算下一个顺序的程序计数器值，其输出与多路转换器 231 的第二输入相连。当 CPU2 执行指令时，程序计数器寄存器 229 的内容代表了存储在寄存器 224 中指令的地址。

图 4 更详细地显示寄存器存储器 228。寄存器存储器 228 通常包括所有通用的可在 CPU 中实现的可编程机器寄存器。寄存器存储器 228 通常由多个寄存器组成，例如寄存器 240、寄存器 242、堆栈指针 244、多路转换器 246、多路转换器 248、与门 272、以及与门 274。执行结果信号提供给寄存器 240、寄存器 242、以及堆栈指针 244 中每一个。寄存器 240、寄存器 242、堆栈指针 244 中每一个的内容以及这里没特别显示的但在寄存器存储器 228 中包括的其他寄存器的内容都提供共多路转换器 246 和 248 中每一个。多路转换器 246 提供一个“A”信号，而多路转换器 248 提供一个“B”信号。

堆栈指针 244 是定义为系统堆栈指针的可编程序寄存器，它提供堆栈指针寄存器内容的低两位，作为 SP（1:0）。该低两位定义了 32 位宽地址空间中的字节地址。SP（1）信号提供存储在寄存器存储器 228 的堆栈指针 244 中的堆栈指针值的位 1。类似地，SP（0）信号提供存储在寄存器存储器 228 的堆栈指针 244 中的堆栈指针值的位 0。SP

（1）信号提供给与门 272 的第一输入，而$\overline{\text{异常事件处理}}$信号提供给与门 272 的第二输入。与门 272 的输出与多路转换器 248 的输入相连。此外，SP（0）信号提供给与门 274 的第一输入，而$\overline{\text{异常事件处理}}$信号提供给与门 274 的第二输入。

图 5 更详细地显示地址产生/执行电路 214。地址产生/执行电路 214 由多路转换器 4 250、多路转换器 5 252、逻辑电路 253、算术逻辑单元（ALU）254、其他执行机构电路 256、状态寄存器 257、以及多路转换器 6 258。

由取操作数电路 212 提供的 RA 信号与多路转换器 4 250、其他执行机构电路 256、以及多路转换器 6 258 相连。由取操作数电路 212 提供的 RB 信号与多路转换器 5 252、其他执行机构电路 256、以及多路转换器 6 258 相连。多路转换器 4 250 及多路转换器 5 252 各与 ALU254 相连。ALU254 与多路转换器 6 258 相连以提供执行结果信号。多路转换器 6 258 提供操作数地址信号。ALU254 的输出是执行结果信号并耦合到多路转换器 6 258 的一个输入上。多路转换器 6 258 然后提供一个操作数地址信号。逻辑电路 253 与状态寄存器 257 相连，而状态寄存器 257 提供 SR（状态寄存器）信号。这是一个可编程序寄存器，用于定义处理器特权级和杂项控制功能。正如已知的，也可以作为其他结构中的处理器状态字。此外，SP（1:0）信号、PC 信号、以及 SR（状态寄存器）信号都提供给其他执行机构电路 256。其他执行机构电路 256 提供操作数写数据信号和执行结果信号。

图 6 更详细地显示其他执行机构电路 256。其他执行机构电路 256 由逻辑电路 259、多路转换器 260、寄存器 261、以及寄存器 262 构成。SP（1:0）信号被提供给逻辑电路 259 的输入。逻辑电路 259 的输出与多路转换器 260 相连以提供格式信号。SR、PC、格式、向量、以及 SP（1:0）信号都提供给多路转换器 260。多路转换器 260 与寄存器 262 相连。寄存器 262 提供操作数写数据信号。此外，RB 信号提供给寄存器 261。寄存器 261 提供一已提取格式信号及执行结果信号。

操作描述

在数据处理系统 5 的操作过程中，处理器核心 9 使用 K－总线 25

连接 CPU2 、 MMU/控制器 4 及存储器 6 。此外，在本发明中，使用者可选择包括操作数未对准单元 5 。如果在数据处理器 3 中包括了操作数未对准单元 5，则 K - 总线 25 把它与 CPU2 、 MMU/控制器 4 及存储器 6 中的每一个连接起来。在本发明的本实施例中， K - 总线 25 是一个高速、单周期存取总线。存储器 6 可以包括随机存取存储器（ RAM ）、只读存储器（ ROM ）、高速缓冲存储器块以及它们的任何组合。所有其他系统模块和外部设备通过 M - 总线 23 与处理器核心 9 相连。 M - 总线 23 是一个内部多主控器总线，用于完成由多总线主控器之一发起的数据传送。系统总线控制器 8 提供多重功能。如果外部总线 E - 总线存在，则系统总线控制器 8 提供内部 M - 总线 23 和外部 E - 总线之间的接口。此外，系统总线控制器 8 作为控制 S - 总线上所有数据传送的焦点。 S - 总线用于把简单从动外部模块（ 12 和 13 ）（如计时器和串行通信通道和存储器）连到数据处理系统 5 中。

在本发明中，数据处理器 3 有几层总线带宽，以提供低成本的数据传送机制。处理器核心 9 与高速单周期 K - 总线 25 互连以获得最好性能。对于不直接与这一高速总线相连的传送， M - 总线 23 提供了来自任何一个内部总线主控器（如处理器核心 9 和主控模块 14 ）的带宽。系统总线控制器 8 提供内部 M - 总线 23 和外部 E - 总线（如果存在的话）之间的连接，同时还提供在低成本、较低带宽的 S - 总线上所有数据传送的控制功能，用于诸如从动模块 12 和 13 等从动外部设备模块。调试模块 10 连于 K - 总线 25 以允许对所有处理器发动的存储器存取进行非入侵监测。调试模块还提供与可选择的外部开发系统 7 的连接。外部开发系统 7 通常在程序调试期间相连和在系统集成测试时开发。数据处理器 3 与外部开发系统 7 相连以互通信息。外部开发系统 7 通过外部控制总线与数据处理器 3 相连。

在下面关于数据处理系统 5 的操作的讨论中，假定数据处理器 3 的用户已决定应降低与设备相关的附加费用，并已选定采用不带操作数未对准单元 5 的数据处理器 3 。如果在数据处理器 3 中采用了操作数未对准单元 5，那么它便提供了 CPU2 和 K - 总线 25 之间的逻辑电路，它将侦听由 CPU2 输出的未对准的参考。于是，操作数未对准单元 5 将把未

对准的参考信号重新组织成对准的参考序列。然而，在本发明的当前实施例中，用户没有在数据处理器3中采用操作数未对准单元5。

如前面讨论的那样，由于没有采用操作数未对准单元5，在异常事件处理过程中现有技术数据处理器将会对任何被恢复的未对准操作数参考产生一个异常事件。如果对于其他类型错误，作为异常事件处理部分发生了未对准参考，则会发生灾难性的双重错误状态。对于大多数数据处理系统，这种双重异常事件对系统性能是灾难性的，常常会导致数据处理系统停机直至对系统复位为止。如以前提到的那样，本发明认识到这个问题并提供了一在自对准的堆栈指针，所以保证了 CPU2 在响应一个异常事件时不会产生未对准参考。下面将更详细地描述本发明的处理器核心9和CPU2。

当数据处理器3在运行时，处理器核心9控制执行内部程序和外部中断。提供CPU2以利用取指令流水线210和操作数执行流水线220来快速有效地处理指令。在本发明的当前实施例中，取指令流水线210和操作数执行流水线220是独立的和互不相连的。

在运行过程中，取指令流水线210预先取指令以供处理器核心9中执行。指令地址产生电路202形成一个预取地址，用于存取下一个指令。这个预取地址通过多路转换器222提供给寄存器208。该预取地址作为KADDR信号提供给K－总线25。CPU 2启动所需的K－总线25传送，以取出由KADDR定义的地址中存放的指令。所希望的指令可以在存储器6、从动模块12、从动模块13、或E－总线相连的设备中，如果存在的话。无论所需的指令位于什么地方，数据最终都通过KRDATA（K总线读数据）信号送回CPU 2。KRDATA信号向缓冲器216提供指令。缓冲器216继而将取来的指令传送给取指令电路204。一旦向取指令电路204提供了取来的指令，取指令电路204向FIFO指令缓冲器206或者多路转换器219提供指令缓冲器写数据信号。当操作数执行流水线220在等待下一指令时，FIFO指令缓冲器206便被越过，于是指令缓冲器写数据信号便被直接提供给多路转换器219。否则，指令缓冲器写数据信号先提供给FIFO指令缓冲器206然后再到多路转换器219。

指令信号从取指令流水线210的多路转换器219提供给操作数执行

流水线 220 的流水线控制逻辑 218 及取操作数电路 212 。再有，取指令流水线 210 的取指令电路 204 向取操作数电路 212 提供取指令地址信号。该取指令地址信号用于根据指令流变化，如分支，建立新的程序计数器寄存器 229 的值。对于涉及寄存器操作数的简单指令，操作数执行流水线作用是两级流水线。在第一级，流水线控制逻辑 218 对指令解码并向取操作数电路 212 提供控制信息。于是，取操作数电路 212 取存储在寄存器存储器中的值。在第二级，流水线控制逻辑 218 向地址产生/执行电路 214 提供控制信号以使用 ALU254 或其它执行机构 256 完成对寄存的输入操作数、RA234 和 RA236 所需的数据运算。所产生的数据在执行结果信号上提供，并通常在一个机器周期结束时写入到寄存器存储器 228 中。

在本发明的当前实施例中，流水线控制逻辑 218 将认定异常事件处理信号以指出已发生了异常事件。例如，可以根据外部中断请求产生异常事件。在数据处理环境中产生异常事件是技术上公知的，这里将不再更详细地讨论。当发生异常事件时，数据处理器 3 不再执行正常的处理。相反，数据处理器 3 必须存取和执行一个特殊的异常事件处理例程。在从正常处理到异常事件处理的过渡过程中，数据处理器 3 的 CPU2 建立一个异常事件堆栈帧，它在系统存储器中保存了当前处理器信息。实际上该存储器可以是存储器 6 、从动模块 12 、从动模块 13 或与 E 总线相连的设备。

在存储于异常事件堆栈帧的当前处理器信息当中包括一个在发生异常事件时正被执行指令的当前地址。该异常事件堆栈帧置于驻留在系统存储器中当前系统堆栈的“顶”部。如前已讨论的那样，由于操作数可以驻留在任何字节边界，所以在检测到异常事件时，硬件堆栈指针可以指向任何字节边界。在本发明的当前实施例中没有采用操作数未对准单元，如果在开始异常事件处理时，堆栈指针未对准，则异常事件堆栈帧的试图产生将导致另一异常事件，并引起灾难性双重错误状态，导致系统操作停止。

为保证在检测到异常事件时，不会由于未对准堆栈指针导致系统性能被破坏而停机，本发明已采用了一种电路和方法来改正这种情况。当

在本发明中发生异常事件时，数据处理器 3 的 CPU2 首先检验堆栈指针寄存器 244 的当前值。然后 CPU2 自动地将它对准到当前值之下最近的模 4 为 0 地址。一旦完成这一调整， CPU 2 在系统存储器中存储一个 8 字节堆栈帧，它包括异常事件的类型和位置。该 8 字节堆栈帧包括一个 4 位格式字段，该字段包括在发生异常事件时有关堆栈指针对准情况的信息。下面将更详细地描述这一操作。

当产生一个异常事件时，流水线控制逻辑 218 认定$\overline{\text{异常事件处理}}$信号并把它提供给取操作数电路 212 。再有，流水线控制逻辑 218 并发执行一个异常事件序列以把当前处理器状态存于系统存储器中它可能是存储器 6 、从动模块 12 、从动模块 13 或通过 E － 总线相连的其他设备。在执行这个异常事件序列过程中，流水线控制逻辑 218 使取操作数电路 212 和地址产生/执行电路 214 在 K － 总线 25 上产生写周期。这个写周期将把构成前述堆栈帧的必须信息写入存储器 6 。请注意，堆栈帧一词和异常事件堆栈帧一词可以互换使用。

当产生堆栈帧时，流水线控制逻辑 218 首先提供控制信息，使寄存器存储器 228 （如图 4 所示）的堆栈指针寄存器 244 向多路转换器 246 输出调整的堆栈指针值。该调处理包括通过将低两位置 0 将堆栈指针寄存器 244 的当前值截断成模 4 余 0 。该截断操作由与门 272 和 274 完成。在正常处理时，$\overline{\text{异常事件处理}}$信号是求反的，于是与门 272 和 274 仅简单地传送堆栈指针寄存器 244 低两位 SP （ 1:0 ）的当前值。一旦检测到异常事件并且流水线控制逻辑 218 认定$\overline{\text{异常事件处理}}$信号，则与门 272 和 274 的输出被强迫置 0 ，这样产生模 4 余 0 地址。此外，堆栈指针寄存器 244 向其他执行机构 256 输出最低两有效位， SP （ 1:0 ）。向多路转换器 246 提供经调整的堆栈指针，它将该值在 A 输出上传送给多路转换器 2 230 ，然后该值被装入寄存器 A 234 。

在经调整的堆栈指针值被从寄存器存储整器 228 传送给寄存器 A 234 时，多路转换器 226 产生一个常数“－ 4 ”。然后，这个常数“－ 4 ”被送到多路转换器 232 并存于寄存器 B 236 。在这一点，经调整的栈指针值存于寄存器 A 234 ，而常数“－ 4 ”存于寄存器 B 236 。经调整的堆栈指针值对应于图 7 所示各例中的“ X ”。

接下来，寄存器 A 234 通过 RA 信号向地址产生/执行电路 214 的多路转换器 250 提供经调整的堆栈指针值。类似地，寄存器 B 236 通过 RB 信号向地址产生/执行电路 214 的多路转换器 252 提供常数“－ 4”。然后 ALU254 对这两个值求和，提供一个值为经调整的堆栈指针值（ X ）加常数－ 4（ X － 4 ）的输出。 ALU254 的输出通过执行结果信号提供给多路转换器 258。 X － 4 的值从多路转换器 258 提供给多路转换器 222 作为操作数地址信号。多路转换器 222 向寄存器 208 提供操作数地址信号。用这种方式，产生用于 K － 总线 25 周期的操作数地址，以完成第一个 32 位异常事件堆栈帧存储。

X － 4 地址值还在执行结果信号上提供给多路转换器 230，然后送给寄存器 A234 。在下一个时间周期开始处，寄存器 A 234 存储 X － 4 值，而寄存器 B 236 存储常数－ 4 。流水线控制逻辑 218 再次提供控制信号，使寄存器 A 234 能通过 RA 信号向地址产生/执行电路 214 的多路转换器 250 提供 X － 4 值。类似地，寄存器 B 236 通过 RB 信号向地址产生/执行电路 214 的多路转换器 252 提供常数－ 4 。其后 ALU254 对这两个值求和，提供一个值为当前堆栈指针值（ X － 4 ）加－ 4（ X － 8 ）的输出。 ALU254 的输出通过执行结果信号提供给多路转换器 258 。多路转换器 258 向多路转换器 222 提供操作数地址信号，然后再提供给寄存器 208 。用此方式，对于 K 总线 25 周期产生操作数地址，以形成第二个 32 位异常事件堆栈帧存储。此外，在执行结果信号上提供的 X － 8 值被写入堆栈指针寄存器 244 ，以在异常事件堆栈帧写结束时反映系统堆栈顶状态。

在产生写数据所需地址的同时，还必须存取要存储在异常事件堆栈帧中的数据。在 X － 4 地址值被形成之时， PC 值被提供给其他执行机构 256 。 PC 值是出错指令的程序计数器值。该 PC 值定义了当前操作数执行流水线 220 中指令的地址，该值从取指令电路 212 中存取。

当产生了程序值并通过 PC 信号提供给其他执行机构 256 时，该 PC 值也和状态寄存器（ SR ）值、格式（ F ）值、以及向量（ V ）值一起提供给多路转换器 260 。 SR 值是响应地址产生/执行逻辑 214 的 ALU254 所产生的状态信息而产生的。由 ALU254 产生的状态信息作为执行结果

信号输出给逻辑电路 253 。逻辑电路 253 处理这个状态信息以提供一个状态值存于状态寄存器 257 。然后由状态寄存器 257 向其他执行机构 256 通过 SR 值提供该状态值。

由流水线控制逻辑 218 产生一个向量值用以代表当前异常事件的类型。该值以后被用于访问定义异常事件处理软件例程的指令地址的向量。

格式值是一个 4 位格式，它确定了在异常事件被认定之时堆栈指针的对准。这个格式值是由逻辑电路 259 在发生异常事件时响应堆栈指针值的第 0 位和第 1 位，即 SP（ 1:0 ）而产生的。下表指出堆栈指针值与逻辑电路 259 产生的格式值之间的关系。

在异常事件之时的原来 SP（ 1:0 ）	在异常事件处理程序第 1 指令处的 SP	格式字段
00	原来 SP － 8	0100
01	原来 SP － 9	0101
10	原来 SP － 10	0110
11	原来 SP － 11	0111

表 1

所以，参考表 1，当 SP（ 1:0 ）有二进制值 00 时，逻辑电路 259 提供的格式字段有相应的二进制值 0100 。类似地，当 SP（ 1:0 ）有二进制值 01，逻辑电路 259 提供的格式字段有二进制值 0101 。当 SP（ 1:0 ）有二进制值 10 时，逻辑电路 259 提供的格式字段有二进制值 0110 。再有，当 SP（ 1:0 ）有二进制值 11 时，逻辑电路 259 提供的格式字段有二进制值 0111 。一旦处理程序例程处理了异常事件，处理器将使用这个异常事件堆栈帧格式字段将堆栈寄存器 244 恢复成其原始的值。。

当处理异常事件时，流水线控制逻辑 218 提供控制信息（这里未画出）以使多路转换器 260 能向寄存器 262 提供 SR、 PC、格式（ F ）、以及向量（ V ）值之一。首先，流水线控制逻辑 218 提供控制信息，它使多路转换器 260 能选择和向寄存器 262 传送

PC值。通过操作数写数据信号，PC值从寄存器262提供给缓冲器216。然后，缓冲器216通过KWDATA（K总线写数据）信号把PC值驱动到K－总线25。其后，PC值存储在存储器中的一个位置，该位置由第1时间周期中并发产生的X－4之值确定。

在第2时间周期期间产生X－8地址，期间流水线控制逻辑218提供控制信息使多路转换器260能够选择的向寄存器262传送SR、格式（F）及向量（V）值。由寄存器262把SR、格式及向量值按图7所示顺序连接起来并通过操作数写数据信号提供给缓冲器216。然后，缓冲器216通过KWDATA（K总线与数据）信号把这个连接起来的值驱动到K－总线25上。其后，格式/向量/状态寄存器值存储在系统存储器中的一个位置，该位置由第2时间周期中并发产生的X－8之值确定。

在产生了异常事件堆栈帧并把PC、SR、格式以及向量之值存于系统存储器中之后，便由数据处理器3处理异常事件。在这一点上，由流水线控制逻辑218提供的向量值被提供给取操作数电路212中的多路转换器226。向量值由多路转换器226提供给多路转换器230，然后存储在寄存器A 234中。向量值由寄存器A 234通过RA信号提供给多路转换器250。在向量产生的同时，在其他执行机构256读取向量基本寄存器（VBR）并通过执行结果信号发送给多路转换器232。VBR具有在系统存储器中重新定位异常事件向量表的能力。通过多路转换器232的VBR值进行门控并装入寄存器B 236中。然后向量值与VBR值在ALU254中相加从而产生一个向量地址，它对应于所产生的异常事件。由ALU254提供的和数被提供给多路转换器258。当由流水线控制逻辑218向多路转换器258提供适当控制时，多路转换器258提供一个向量地址作为操作数地址信号。然后把操作数地址信号提供给多路转换器222。该向量数从多路转换器222由缓冲器208作为KADDR信号输出。KADDR信号由K－总线25使用以读出所希望的异常事件向量。

被取向量通过KRDATA信号传送给缓冲器216。被取向量由KRDATA缓冲器信号从缓冲器216中提供给取操作数电路212。

KRDATA 缓冲器信号与执行结果信号相连以传送被取向量。然后该被取向量被提供给多路转换器 230 并存于寄存器 A 234。通过 RA 信号，把被取向量从寄存器 A 234 提供给多路转换器 250。多路转换器 250 把被取向量传送给 ALU254，在那里在流水线控制逻辑 218 的控制下它与多路转换器 252 产生的零求和。ALU254 必须将被取向量在执行结果信号上传送，在此它将输入提供给多路转换器 258，然后又被传送给多路转换器 222。该被取向量然后装入 KADDR 寄存器 208。该地址代表用于异常事件处理程序起始指令的取指令地址。该异常事件取指令地址从寄存器 208 通过 KADDR 信号提供给 K 总线 25。

随着处理异常事件的软件程序最初指令被取出而且数据被取出，由 KRDATA 信号把该数据提供回 CPU2。特别是 KRDATA 信号从缓冲器 216 提供给取指令电路 204。然后由多路转换器 219（作为选择可使用 FIFO 指令缓冲器 206）对取出指令进行门控，以将取出的指令装入操作数执行流水线 220。上文中详细描过的处理过程发动和执行处理异常事件的软件程序。

当处理异常事件的软件程序完成时，数据处理器 3 将停止异常事件处理并返回正常处理。为了返回正常处理，必须恢复异常事件发生之前数据处理器 3 操作所处前后关系（context）。所以，程序计数器、状态寄存器、以及堆栈指针都必须恢复到它们在异常事件发生之前的值。把异常事件堆栈帧唯一地用于存储格式字段，而该格式字段确定了在异常事件发生时刻堆栈指针的对准，这对于正确和有效地把数据处理器 3 恢复到一个原来状态是至关重要的。

处理异常事件的软件程序中最后一条指令必须是 RTE（由异常事件返回）指令。当执行 RTE 指令时，二个存于异常事件堆栈帧（如图 7 所示）上的长字值将被取出。为了取出这些值，以寄存器存储器 228 的堆栈指针寄存器 244 中取出一个当前堆栈指针值。请记住，在软件执行程序开始时，堆栈指针值是 X - 8。尽管在异常事件处理例程的执行过程中，必须使用堆栈区域用于暂时存储，但堆栈指针必须保持为 X - 8，向 RTE 指令开始执行时那样。通

过 A 信号将堆栈指针值提供给要进行通信的多路转换器 246。A 信号被提供给多路转换器 230，然后传送给寄存器 A 234。与此同时，多路转换器 226 产生一个常数值“+ 4”。常数值 4 从多路转换器 226 传送到多路转换器 232 并存于寄存器 B 236。

在下一个时间周期，寄存器 A 234 的内容（即 X - 8 之值）被 RA 信号提供给多路转换器 258。然后，多路转换器 258 通过操作数地址信号把 X - 8 之值提供给多路转换器 222。从多路转换器 222 取出的 X - 8 值被存于寄存器 208。其后这个 X - 8 之值通过 KADDR 信号提供给 K - 总线 25。然后在 X - 8 之值所指定的系统存储器中的地址位置存取数据。

在形成操作数地址 X - 8 的同时，分别将包含 X - 8 值的寄存器 A 234 和包含 + 4 值的寄存器 B 236 传送给多路转换器 250 和 252。在流水线控制逻辑 218 提供的控制下，多路转换器 250 和 252 将 X - 8 值和常数 4 送到 ALU254。ALU254 将两值相加产生 X - 4。将该结果送给执行结果信号，它将 X - 4 值送给多路转换器 230。多路转换器将 X - 4 提供给寄存器 A 234。

参考图 7，当由 X - 8 之值指定的地址被存取时，格式（F）、向量（V）、及 SR 值被存取。格式、向量、及 SR 值通过 KRDATA 信号提供给缓冲器 216。从缓冲器 216，通过缓冲器 KRDATA 信号和执行结果信号、格式、向量和 SR 值被提供给取操作数电路 212 的多路转换器 232。然后，这些值进入寄存器 B 236。从寄存器 B 236，通过 RB 信号，格式、向量及 SR 值被提供给多路转换器 252。其后，寄存器 B 236 的整个内容不加修改地穿过 ALU 254 并由逻辑电路 253 处理以提供一个状态值。然后该状态值被存于状态寄存器 257 中以表示发生异常事件时数据处理器 3 的操作前后关系。类似地，从寄存器 B 236，通过 RB 信号，格式值被提供其他执行机构 256。其后该格式值被存于寄存器 261。

在下一时间周期中，存于寄存器 A 234 的 X - 4 值被 RA 信号提供给多路转换器 258。RA 信号把 X - 4 值传送给多路转换器 222，然后再传送给寄存器 208。然后 X - 4 值通过 KADDR 信号

提供给 K - 总线 25。然后在由 X - 4 值指定的系统存储器中地址位置存取数据。

再参考图 7，当由 X - 4 值指定的地址位置被存取时，程序计数器（PC）值被存取。PC 值通过 KRDATA 信号提供给缓冲器 216。从缓冲器 216，通过缓冲器 KRDADT 信号，PC 值被提供给取操作数电路 212，将 PC 值通过执行结果信号送到多路转换器 233，在此传送并装入寄存器 B 236。此时，在这一点寄存器 A 234 包括 X - 4 值，而寄存器 B 236 包括新的程序计数器地址。

在下一时间周期，包含新的程序计数器地址的寄存器 B 236 值被送入多路转换器 258。该地址然后通过操作数地址信号送入多路转换器 222，在此将其装入寄存器 208。该地址定义指令地址以便在处理异常事件之后继续执行。然后使用 KADDR 寄存器 208 由系统存储器访问所需的指令以继续执行程序。该所需指令在 KRDATA 上返回，通过缓冲器 216 送入取指令电路 204。

在下一时间周期，通过执行结果信号由寄存器 261 中取出格式值。然后将该值发送到多路转换器 232 并装入寄存器 B 236 中。此时，寄存器 A 234 还包含 X - 4 值，而寄存器 B 包含格式值。在下一时间周期，多路转换器 250 和 252 的内容都送到 ALU254，它对两值求和，即：X - 4 + 格式值。

该求和又产生异常事件发生时的原始堆栈指针值。该 ALU 输出通过执行结果信号存储在寄存器存储器 228 的堆栈指针寄存器 244 中作为当前的堆栈指针。

此时，处理器 3 已完全将机器状态恢复到异常事件发生时其原始值：取指令流水线 210 从恢复的程序计数器值取指令，并且堆栈指针寄存器 244 已恢复到异常事件发生时其原始值。本发明的自对准特点使堆栈指针在接收到异常事件时自动将其本身对准到当前堆栈指针设置之下的最接近于模 4 余 0 的地址。当已处理了异常事件并且处理器 3 开始按正常操作方式运行，堆栈指针必须回到其原始值，即使在接收到异常事件时该值未对准。对堆栈指针的中间值 X - 4 提供附加格式字段是使堆栈指针返回其初始值的唯一有效方

法。

考虑已知基准的下列例子:

```
指令地址          操作码
$ 00005064:   mov.b(a0), -(sp)  # 由源拷贝字节到堆栈
$ 00005066:   bra.b L % 1       # 在 L % 1 处继续
```

在执行地址 $ 00005064 处指令之前，堆栈指针的内容是 $ 00006ed0。地址 $ 00005064 的操作码 mov.b 由寄存器 a0 内容定义的源位置将一字节数据拷贝到系统堆栈并将堆栈指针减“1”。在该指令结束时，堆栈指针值是 $ 00006ecf。

假设，然后有一中断请求到来，使地址 $ 00005056 处的指令的执行延期，以服务于该中断请求。当处理该中断异常事件时，产生下列异常事件堆栈帧:

```
地址          数据
$ 00006ec4    $ 70780008   # 格式=$ 7  向量=$ 078，状
                           态=$ 0008
$ 00006ec8    $ 00005066   # 指令地址
$ 00006ecc    $ --------aa # 由 mov.1 操作码拷贝的数据
```

其中，在异常事件处理软件程序开始时堆栈指针寄存器值是 $ 00006ec4。

在为中断请求服务之后，异常事件处理器执行 RTE 指令。在堆栈顶部的 32 位数据被取出，以及按前所述由 CPU2 处理的格式、向量和 SR 字段。$ 00006ec8 的中间地址 X - 4 被形成并临时存储在寄存器 A 234 中。由位置 $ 00006ec8 取出指令地址 $ 00005066，并使用该地址建立新的指令流。同时在 ALU 254 中对保存的格式字段（$ 7）及寄存器 234 的内容（$ 00006ec8）求和，以形成恢复的堆栈指针（$ 00006ec8 + $ 00000007= $ 00006ecf），将其装入寄存器存储器 228 的堆栈指针寄存器 244 中。然后控制继续执行地址 $ 00005066 处的指令。

概括地说，本发明提供一种数据处理器以及操作一个数据处理系统的方法，在该系统中当把硬件支持堆栈操作数对准作为一种可

选功能时，可以使用单个系统堆栈指针来建立监督者和用户堆栈操作二者的记录。该单个系统堆栈指针被用作为自对准堆栈指针，它把自己对准到当前地址设置之下的最近的模4余0地址，从而即使当没有硬件支持未对准操作数时也不会发生系统错误。一旦发生了自动对准，本发明的数据处理器便在一个异常堆栈帧中存储一个4位格式字段，以指示关于出错时堆栈指针对准的信息。当已经对异常事件提供了服务并执行从异常事件返回（RTE）指令时，处理器使用这个存储在异常事件堆栈帧中的4位格式字段，以把堆栈指针恢复到异常事件发生时它的原来值。

这里描述的本发明的实现只是以举例方式提供的，而且可以有许多其他实现来执行这里所描述的功能。例如，产生并用于建立异常事件堆栈帧地址的常数不限于是－4，也可以是数据处理系统5的设计者或使用者所希望的任何数。

尽管已参考具体实施例图示和描述了本发明，但对于本行专家而言，可以有进一步的修改和改进。所以要理解本发明不限于所展示的特定形式，所附权利要求覆盖了不偏离本发明范围的所有修改。

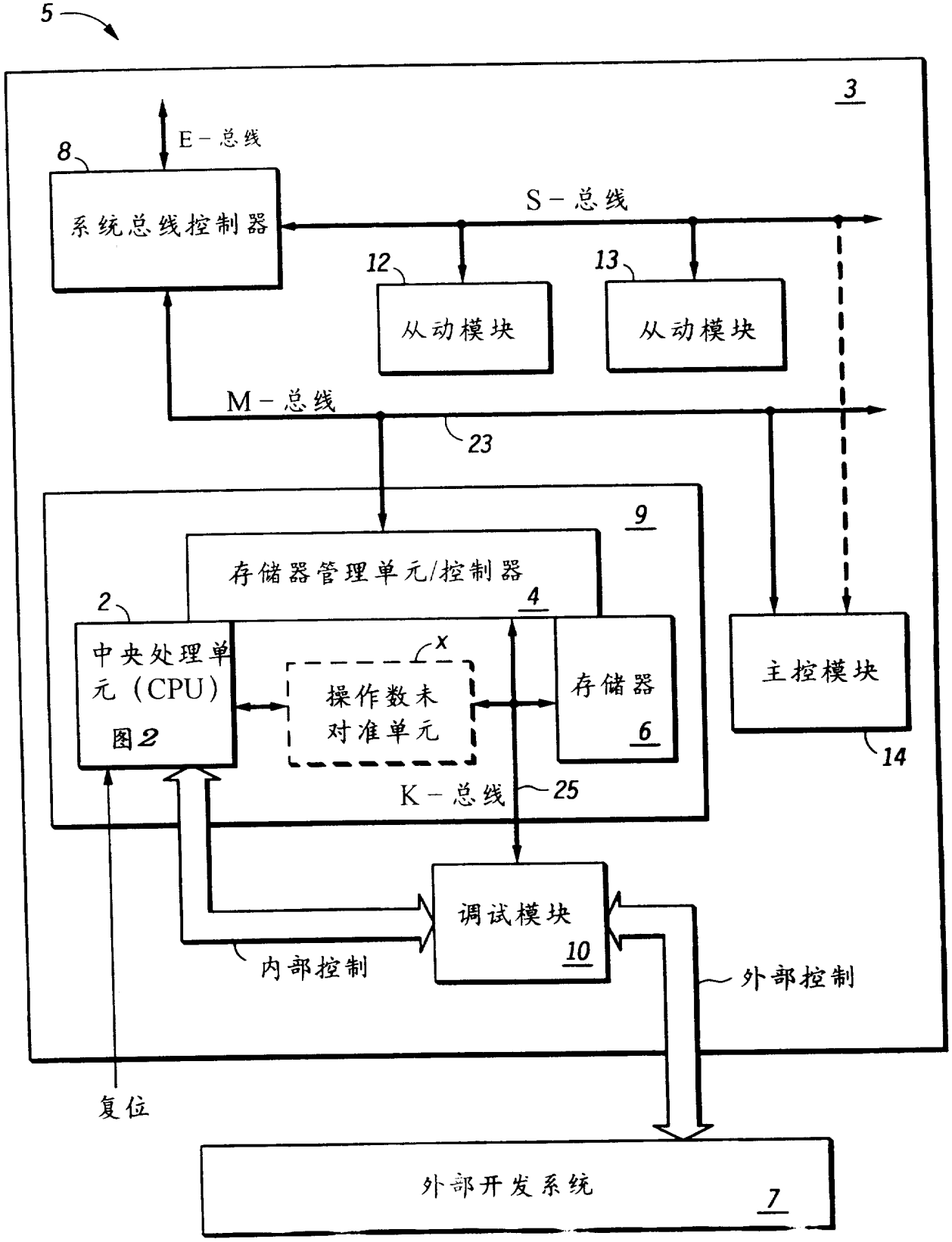

5
3
E－总线
8
系统总线控制器
S－总线
12
从动模块
13
从动模块
M－总线
23
9
存储器管理单元/控制器
4
2
中央处理单元（CPU）
图2
X
操作数未对准单元
存储器
6
主控模块
14
K－总线
25
调试模块
10
内部控制
外部控制
复位
外部开发系统
7

图 1

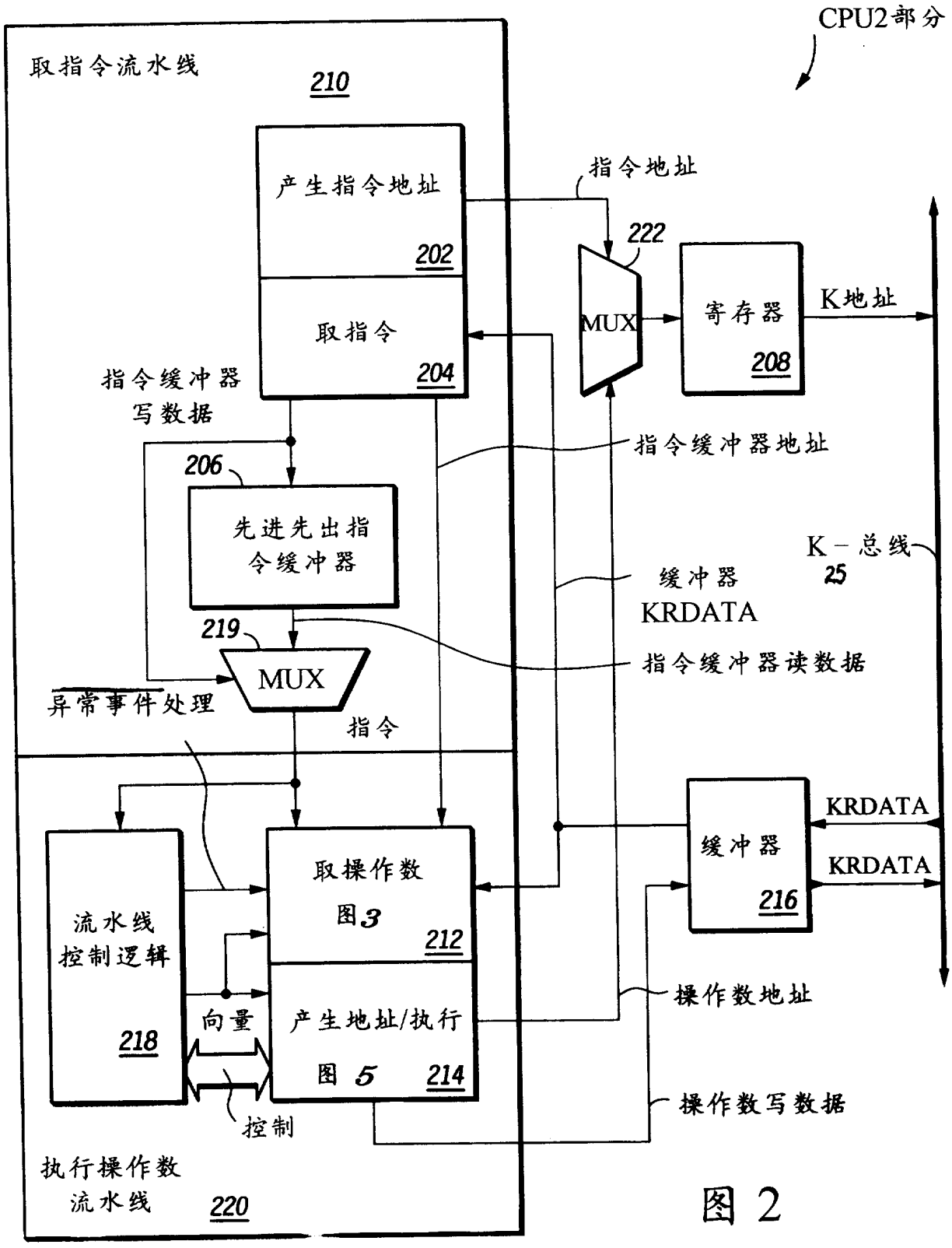

CPU2部分
取指令流水线
210
产生指令地址
202
取指令
204
指令地址
222
MUX
寄存器
208
K地址
指令缓冲器
写数据
206
先进先出指
令缓冲器
指令缓冲器地址
K－总线
25
缓冲器
KRDATA
219
MUX
指令缓冲器读数据
异常事件处理
指令
KRDATA
KRDATA
缓冲器
216
流水线
控制逻辑
218
取操作数
图3
212
向量
产生地址/执行
图 5
214
操作数地址
操作数写数据
控制
执行操作数
流水线
220

图 2

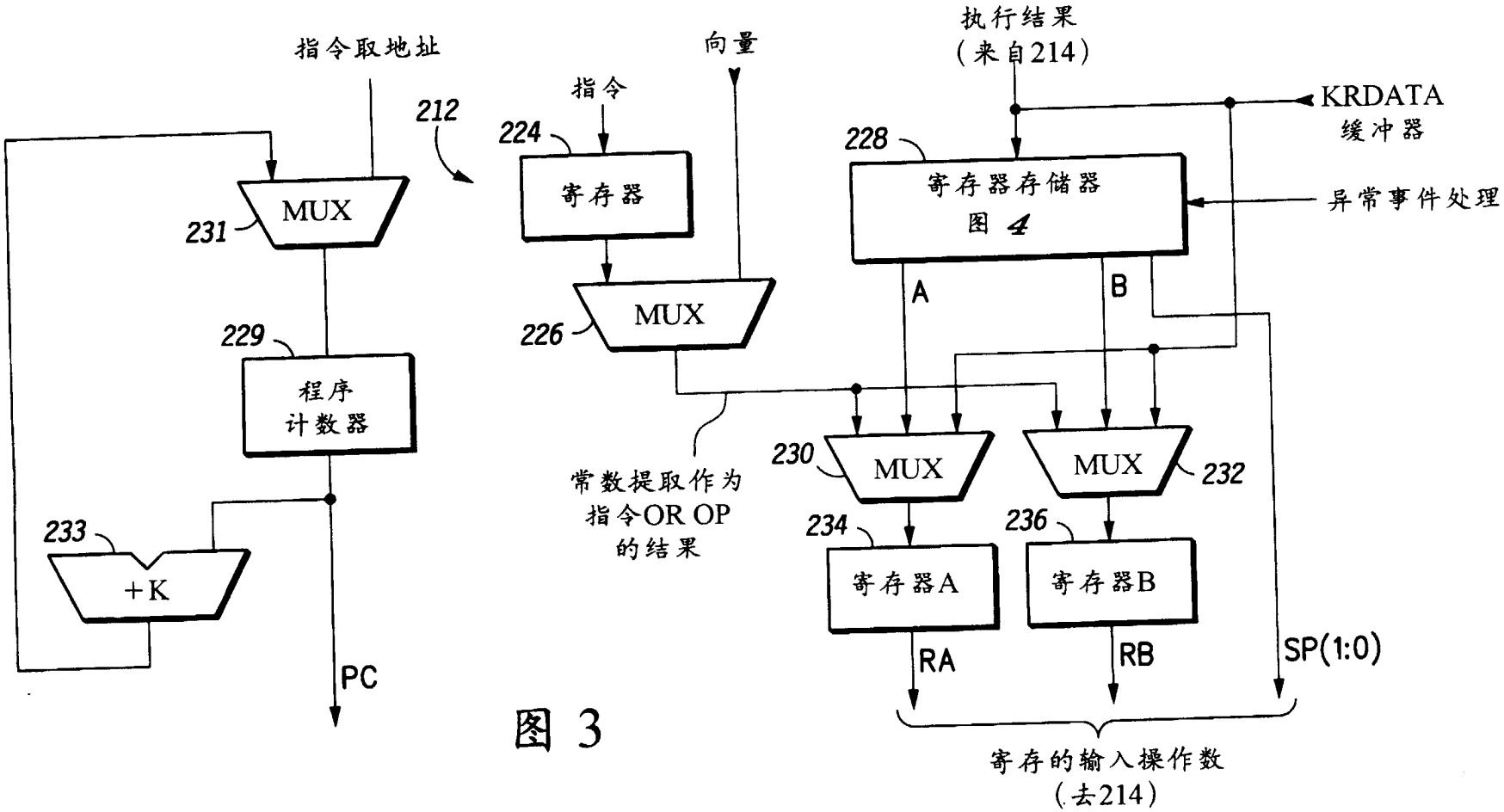

指令取地址
212
指令
向量
执行结果
(来自214)
KRDATA
缓冲器
224
寄存器
228
寄存器存储器
图 4
异常事件处理
231
MUX
226
MUX
A
B
229
程序
计数器
常数提取作为
指令OR OP
的结果
230
MUX
MUX
232
233
+K
234
寄存器A
236
寄存器B
PC
RA
RB
SP(1:0)
寄存的输入操作数
(去214)

图 3

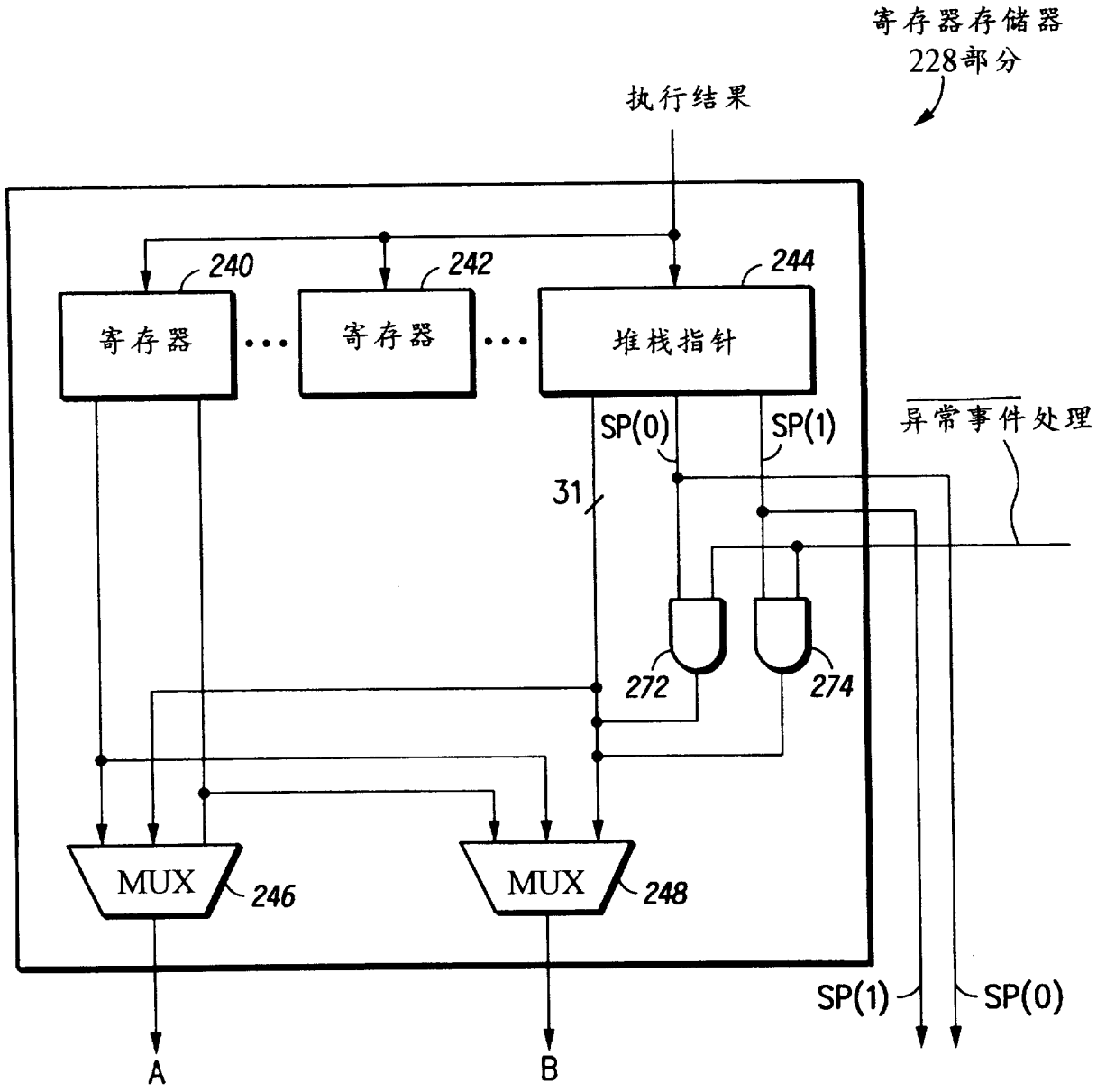

寄存器存储器
228部分
执行结果
240
242
244
寄存器
寄存器
堆栈指针
SP(0)
SP(1)
异常事件处理
31
272
274
MUX
246
MUX
248
SP(1)
SP(0)
A
B

图 4

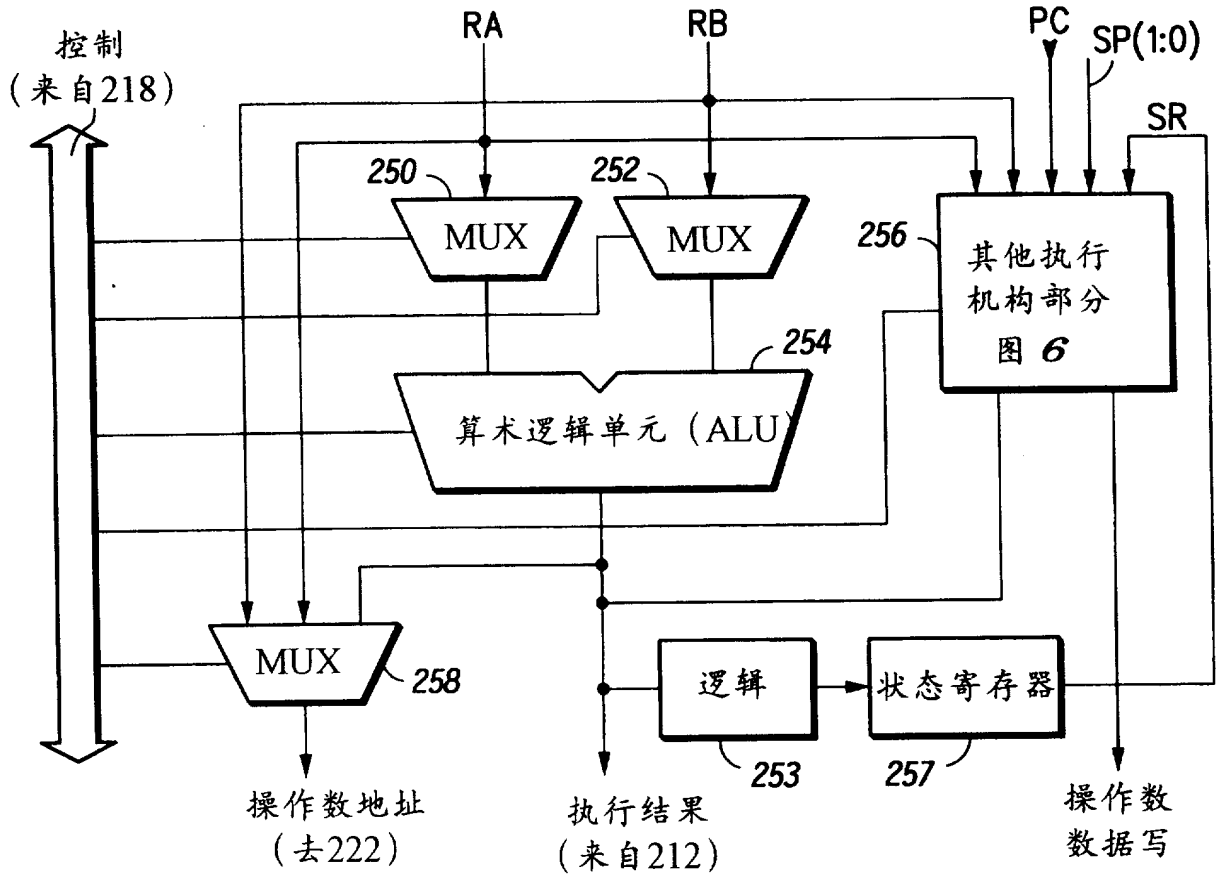

控制
（来自218）
RA
RB
PC
SP(1:0)
SR
250
252
MUX
MUX
256
其他执行
机构部分
图 6
254
算术逻辑单元（ALU）
MUX
258
逻辑
253
状态寄存器
257
操作数地址
（去222）
执行结果
（来自212）
操作数
数据写

图 5

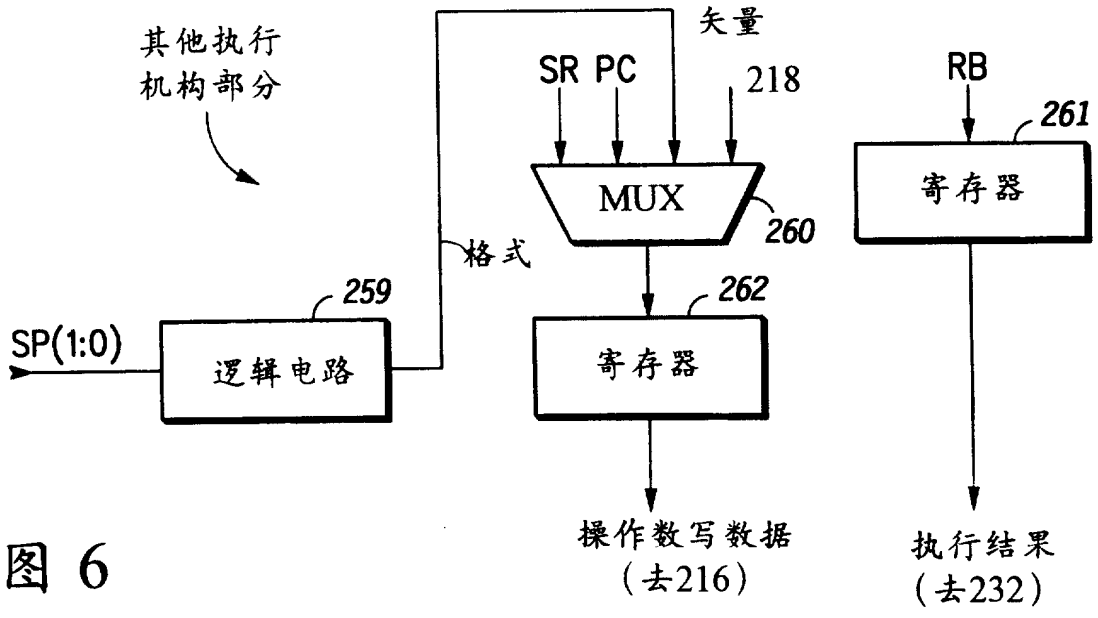

其他执行
机构部分
矢量
SR PC
218
RB
261
MUX
260
寄存器
格式
259
262
SP(1:0)
逻辑电路
寄存器
操作数写数据
（去216）
执行结果
（去232）

图 6

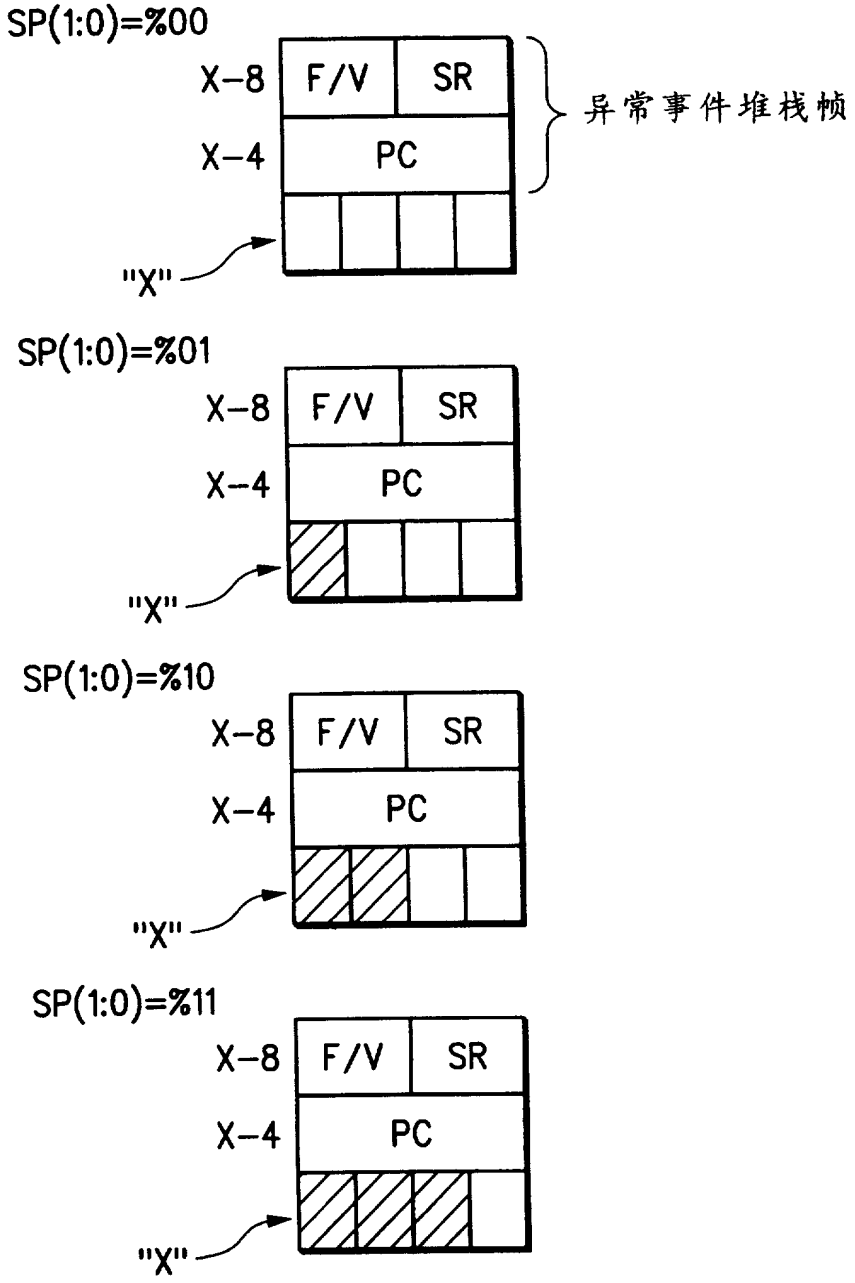

SP(1:0)=%00
X-8
F/V
SR
X-4
PC
异常事件堆栈帧
"X"
SP(1:0)=%01
X-8
F/V
SR
X-4
PC
"X"
SP(1:0)=%10
X-8
F/V
SR
X-4
PC
"X"
SP(1:0)=%11
X-8
F/V
SR
X-4
PC
"X"

图 7

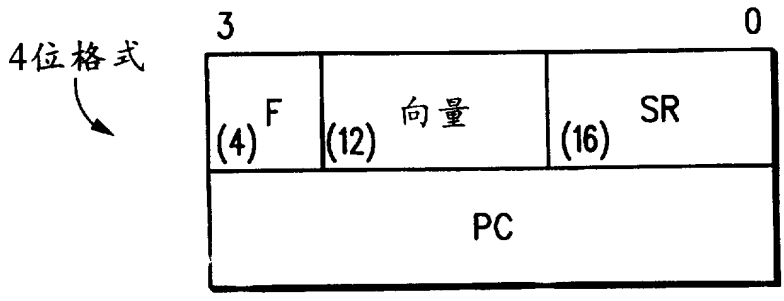

4位格式
3
0
(4) F
(12) 向量
(16) SR
PC

图 8