

一种 RISC-V 指令集余数指令的实现方法及系统

技术领域

本发明涉及微处理器技术领域，具体涉及一种 RISC-V 指令集余数指令的实现方法及系统。

背景技术

微处理器经过 50 多年的发展，微处理器的架构伴随着半导体工艺经历了蓬勃的发展。从单核到物理多核及逻辑多核；从顺序执行到乱序执行；从单发射到多发射；尤其在服务器领域，不断的追求处理器的性能。随着数据中心和科学计算的要求越来越高，对除法及余数指令的性能要求也越高，同时指令中的除法和余数指令比重逐渐升高。除法和余数指令执行周期比较长，同时执行周期与数据相关，执行周期可变。这些因素对 CPU 的性能影响很大。

发明内容

针对现有技术的不足，本发明公开了一种 RISC-V 指令集余数指令的实现方法及系统，用于解决余数指令不能用除法指令的执行结果，每条余数指令需要在执行单元执行，然后得到余数，余数指令执行周期长，造成效率低的问题。

本发明通过以下技术方案予以实现：

第一方面，本发明公开一种 RISC-V 指令集余数指令的实现方法，包括以下步骤：

S1 对 CPU 乱序执行，指令从取指单元进入指令译码单元，进行指令译码；

S2 译码完成后的指令在重命名单元进行目的寄存器的重命名，同时对余数指令进行优化；

S3 若余数指令不满足优化条件时，重命名之后的指令进入保留站后，进入执行单元执行；

S4 执行完成后的指令通过重排序缓存提交，并释放在重命名阶段分配的除法指令编码缓存资源。

更进一步的，所述方法中，当除法和余数指令配对出现时，通过余数指令的目的寄存器映射到除法指令写余数的物理寄存器，取到除法指令产生的余数。

更进一步的，所述方法中，在重命名阶段出现余数指令时，检索余数指令加速单元中的除法指令编码缓存，如果除法指令的编码和余数指令的编码匹配成功，那么该条余数指令可以优化；

如果除法指令的编码和余数指令的编码没匹配成功，那么该条余数指令需要到执行单元执行，计算得到余数。

更进一步的，所述方法中，判断余数指令匹配不成功的规则为：当连续不同类型的除法指令、连续不同类型的余数指令或者除法指令与余数指令不匹配；

当除法指令与余数指令判断为不匹配时，余数指令加速单元中的是否配对的域置为 0。

更进一步的，所述方法中，当除法指令写入到除法指令编码缓存的时候，需要判断是否有空闲的 entry，将除法指令的信息写到对应的 entry；当余数指令的标识 rem_val 有效时，表示当前指令为余数指令，如果有效位 valid 有效，那么该条余数指令匹配成功。

更进一步的，所述方法中，除法指令在重命名阶段申请物理寄存器 div_phy_quo 和 div_phy_rem，用于分别存放除法指令的商和余数，其中除法指令的 div_phy_quo 更新到除法指令目的寄存器重命名映射表 RAT，div_phy_rem 存放到除法指令编码缓存的写余数目的寄存器 PHY_REG。

更进一步的，所述方法中，当除法指令进入重命名阶段时，当不存在配对的余数指令，除法指令根据编码，将除法指令信息写入除法指令编码缓存，除法指令写缓存时，先从缓存中找到空闲的位置，然后把除法指令的编码 DIV_N_OP、保存配对余数指令结果的物理寄存器地

址 div_phy_rem 及除法指令除法指令的重排序 ROB_ID 写到缓存中，此时除法指令编码缓存的有效位 valid 置为 1；

当余数指令进入重命名阶段时，余数指令编码 REM_N_OP 和除法指令编码 DIV_N_OP 进行匹配检查，根据除法指令 DIV 是和余数指令 REM 配对规则进行检查；如果余数指令比较命中，则将目的寄存器 rem_rd 的映射关系映射到 rem_phy_reg，并且更新到目的寄存器重命名映射表 RAT，余数指令执行完成，更新在重排序缓存中指令执行完成指示，并且释放除法指令编码缓存资源。

更进一步的，所述方法中，当出现刷新、复位或者后续新的除法指令或者余数指令时，释放除法指令申请的物理寄存器，并且释放除法指令编码缓存；

当除法指令在 ROB 提交的时候，根据提交指针 cm_ptr 得到除法指令的 ROB_ID，ROB_ID 去检索除法指令编码缓存，如果除法指令编码缓存不是因为异常刷新或者分支指令预测错误刷新释放，那么该位置为余数指令配对时释放的情况；

当存在指令与正在提交的除法指令配对时，配对的余数指令释放物理寄存器 div_phy_quo，当不存在余数指令与正在提交的除法指令配对时，除法指令同时释放物理寄存器 div_phy_quo 和物理寄存器 div_phy_rem。

更进一步的，所述方法中，当一条余数指令在重命名阶段，除法指令编码缓存没有匹配的除法指令时，这条余数指令需要送到指令执行单元，指令计算得到余数，并且更新到余数目的寄存器。

第二方面，本发明公开一种 RISC-V 指令集余数指令的实现系统，所述系统用于执行第一方面所述的 RISC-V 指令集余数指令的实现方法，包括寄存器、执行单元、除法单元、指令译码单元和取指令单元。

本发明的有益效果为：

本发明在重命名阶段通过增加一个余数指令加速单元实现余数指令的功能，当除法和余数指令配对出现时，余数指令不需要发射到后

续除法执行单元，而是通过余数指令的目的寄存器映射到除法指令写余数的物理寄存器，取到除法指令产生的余数，余数指令执行效率高。

附图说明

为了更清楚地说明本发明实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本发明的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动的前提下，还可以根据这些附图获得其他的附图。

图 1 是余数指令实现架构图；

图 2 是余数指令加速处理单元图；

图 3 是执行除法指令图；

图 4 是除法与余数指令相邻图；

图 5 是除法与余数指令配对不相邻图；

图 6 是除法指令与余数指令配对不同拍图；

图 7 是除法指令与余数指令配对检查图。

具体实施方式

为使本发明实施例的目的、技术方案和优点更加清楚，下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有作出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

实施例 1

本实施例公开一种实现余数指令的新方法。该方法在重命名阶段通过增加一个余数指令加速单元实现余数指令的功能，如图 1 所示。当除法和余数指令配对出现时，余数指令不需要发射到后续除法执行单元，而是通过余数指令的目的寄存器映射到除法指令写余数的物理

寄存器，取到除法指令产生的余数。

本实施例以具体的 RISC V 指令为例进行详细阐述。在乱序执行的 CPU 中，指令从取指单元进入指令译码单元，进行指令译码；译码完成后的指令在重命名单元进行目的寄存器的重命名，并且在重命名阶段对余数指令进行优化；如果余数指令不满足优化条件时，重命名之后的指令进入保留站后，进入执行单元执行。

本实施例主要侧重强调除法指令与余数指令；执行完成后的指令通过重排序缓存提交，并且释放在重命名阶段分配的除法指令编码缓存等资源，如图 1 所示。

本实施例解决了余数指令不能用除法指令的执行结果，每条余数指令需要在执行单元执行，然后得到余数，余数指令执行周期长，造成效率低的问题。

实施例 2

本实施例在指令译码阶段产生除法指令和余数指令的一个新操作码 N_OP，为了便于说明，对除法和余数指令的 N_OP 编码，如表 1. 所示。

表格 1 除法与余数指令编码。

指令	扩展操作码[31:25]	扩展操作码[14:12]	操作码	新编码 N_OP
DIV	0000001	100	0110011	000
DIVU	0000001	101	0110011	001
DIVUW	0000001	101	0111011	010
DIVW	0000001	100	0111011	011
REM	0000001	110	0110011	100
REMU	0000001	111	0110011	101
REMU W	0000001	111	0111011	110
REMW	0000001	110	0111011	111

在本实施例中，主要以表 1. 中的指令为例进行说明。除法指令和余数指令可以配对的组合为：000 与 100，001 与 101，010 与 110 和 011 与 111。N_OP 中除法指令的编码称为 DIV_N_OP；N_OP 中余数指令的编码称为 REM_N_OP。

本实施例中，在重命名阶段出现余数指令时，检索余数指令加速单元中的除法指令编码缓存，如果 DIV_N_OP 和 REM_N_OP 匹配成功，那么该条余数指令可以优化。如果 DIV_N_OP 和 REM_N_OP 没匹配成功，那么该条余数指令需要到执行单元执行，计算得到余数。判断余数指令匹配不成功的规则：当连续不同类型的除法指令、连续不同类型的余数指令或者除法指令与余数指令不匹配。当除法指令与余数指令判断为不匹配时，余数指令加速单元中的是否配对的域置为 0。

本实施例中，除法指令在重命名阶段申请 2 个物理寄存器 div_phy_quo 和 div_phy_rem。这 2 个物理寄存器分别存放除法指令的商和余数。除法指令的 div_phy_quo 更新到除法指令目的寄存器重命名映射表 RAT。div_phy_rem 存放到除法指令编码缓存的写余数目的寄存器 PHY_REG。除法指令在表 1. 中映射的新编码 N_OP 写入除法指令编码缓存的 DIV_N_OP。除法指令的重排序缓存编号也写入除法指令编码缓存的 ROB_ID 域。当除法指令的信息写完后，置 valid 为有效，如图 2. 所示。

本实施例中，余数指令加速单元中的除法指令编码缓存保存了需要配对的除法指令及相关信息。当除法指令写入到除法指令编码缓存的时候，需要判断是否有空闲的 entry，将除法指令的信息写到对应的 entry。当余数指令的标识 rem_val 有效时，即表示当前指令为余数指令。余数指令的编码 REM_N_OP 匹配除法指令编码缓存中的除法指令编码 DIV_N_OP。同时如果有效位 valid 有效，那么该条余数指令匹配成功，即 div_rem_hit 为 1。余数指令目的寄存器 rem_rd 映射到除法物理寄存器 rem_phy_reg。

本实施例中，除法指令进入除法执行单元时，同时得到商和余数。除了将商和余数 Forward 到提前唤醒逻辑，同时把商和余数写到物理寄存器堆。写的地址分别为 div_phy_quo 和 div_phy_rem。当保留站中存在与该条除法指令存在依赖关系时，通过比较 div_phy_quo 和 div_phy_rem，如果物理寄存器地址匹配，那么提前得到数据，并且提

前发射到执行单元执行。

实施例 3

本实施例公开除法指令与余数指令实现的几种情况：第一种情况，除法指令与余数指令配对出现，并且在同一拍流水线：

在重命名阶段的时候，如果流水线中的除法和余数指令存在配对，并且除法和余数指令之间不存在其它指令，那么配对指令中的余数指令不需要执行，即该余数指令不需要发到后续的流水线，余数指令的功能完全被配对的除法指令实现，即如图 4. 所示，相邻的除法指令与余数指令。

在重命名阶段的时候，如果流水线中的除法和余数指令存在配对，但是除法和余数指令之间存在其它指令，那么配对指令中的余数指令也不需要执行，即该余数指令不需要发到后续的流水线，余数指令的功能完全被配对的除法指令实现，即如图 5. 所示，配对但是不相邻的除法指令与余数指令。

第二种情况，除法指令与余数指令配对出现，但是不在同一拍流水线：

当除法指令进入重命名阶段时，当拍不存在配对的余数指令，除法指令根据表 1. 中的编码，将除法指令信息写入除法指令编码缓存。除法指令写缓存的时候，先从缓存中找到一个空闲的位置，然后把除法指令的编码 DIV_N_OP、保存配对余数指令结果的物理寄存地址 div_phy_rem 及除法指令除法指令的重排序 ROB_ID 写到缓存中。除法指令编码缓存的有效位 valid 置为 1。

当余数指令进入重命名阶段时，余数指令编码 REM_N_OP 和除法指令编码 DIV_N_OP 进行匹配检查。根据表 1. 中除法指令 DIV 是和余数指令 REM 配对规则进行检查。如果余数指令比较命中，即 div_rem_hit 为 1，说明该条余数指令需要的余数可以由之前的除法指令产生得到，并且余数保存在 rem_phy_reg 中。因此，余数指令只需要将目的寄存器 rem_rd 的映射关系映射到 rem_phy_reg，并且更新到目的寄存器重

命名映射表 RAT。余数指令执行完成，不需要进入后续的执行单元执行，只需要更新在重排序缓存中指令执行完成指示，并且释放除法指令编码缓存资源，即置 valid 为 0，如图 2. 所示。余数指令在重排序缓存提交时释放物理寄存器。

第三种情况，除法指令与余数指令没有配对：

除法指令本身不能确定是否能够与后续的余数指令配对，因此在重命名阶段除法指令为余数推测申请一个物理寄存器，当出现刷新、复位或者后续新的除法指令或者余数指令时，释放除法指令申请的物理寄存器，并且释放除法指令编码缓存。

当除法指令在 ROB 提交的时候，根据提交指针 cm_ptr 得到除法指令的 ROB_ID，ROB_ID 去检索除法指令编码缓存，如图 7. 所示。如果除法指令编码缓存不是因为异常刷新或者分支指令预测错误刷新释放，那么该位置为余数指令配对时释放的情况。当存在指令与正在提交的除法指令配对时，除法指令不需要释放物理寄存器 div_phy_rem，只需要释放物理寄存器 div_phy_quo。配对的余数指令会释放物理寄存器 div_phy_quo。当不存在余数指令与正在提交的除法指令配对时，除法指令需要同时释放物理寄存器 div_phy_quo 和物理寄存器 div_phy_rem。

第四种情况，除法指令与余数指令没有配对出现，并且只有 1 条余数指令：

当一条余数指令在重命名阶段，除法指令编码缓存没有匹配的除法指令时，这条余数指令需要送到指令执行单元，指令计算得到余数，并且更新到余数目的寄存器。这种情况下的余数指令和其它指令的处理流程完全相同。

为了便于进一步阐述原理，假设 CPU 的带宽为每个时钟周期一条指令，以如下表的 RISC V 指令序列为例进行说明。

表格 1RISC V 指令序列

序号	指令	指令操作数	补充说明
----	----	-------	------

1	divw	a4, s10,s6	除法指令, 编码为 011
2	sd	a4, -300(s0)	
3	sw	a4, -200(s0)	
4	auipc	a4,0x1	
5	lbu	a5,1731(a4)	
6	divu	a5,a5,a4	除法指令, 编码为 001
7	c.addi	a3,4	
8	addiw	s10,s5,1	
9	remu	s3,a5,a4	余数指令, 编码为 101
10	sw	s3, -4(a3)	
11	bgeu	a5,a4,pc -20	
12	bge	s10,s9,pc+18	
13	slli	s8,s5,2	
14	c.ld	a5,0(a1)	
15	c.li	a4,10	
16	remu	s3,a5,a4	余数指令, 编码为 101
17	c.addiw	s8,0	
18	c.addi4spn	a3,a5,a4	
19	c.li	s5,1	
20	c.swsp	s3,0(sp)	
21	bgeu	a5,a4,pc+8	
22	divuw	a5,a5,a4	除法指令, 编码为 010
23	c.addi	a3,4	
24	addiw	s10,s5,1	
25	remu	s3,a5,a4	余数指令, 编码为 101
26	sw	s3,-4(a3)	
.....			

表格 2 写入序号 1 divw 除法指令

除法指令编码 DIV_N_OP	写余数目的物理 寄存器 PHY_REG	该项有效位 valid	重排序缓存索引 ROB_ID	是否配对
011	div_phy_rem_1	1	rob_id_1	1

序号 1 divw 除法指令在重命名的时候, 申请 2 个物理寄存器 div_phy_quo_1 和 div_phy_rem_1。序号 1 divw 指令分配的 ROB_ID 为 rob_id_1。并且将这些信息写入到除法指令编码缓存。同时默认该条除法指令存在与它配对的余数指令, 即是否配对的域置为 1。从序号为 1 的指令到序号为 6 的指令之间, 没有出现与 divw 配对的余数指令 REMW。当序号 1 divw 除法指令在 ROB 提交的时候, 同时释放物理寄存

器 div_phy_quo_1 和 div_phy_rem_1。序号 6 divu 除法指令在重命名的时候，申请 2 个物理寄存器 div_phy_quo_2 和 iv_phy_rem_2。序号 6 divu 指令分配的 ROB_ID 为 rob_id_2。并且将这些信息写入到除法指令编码缓存。同时标记序号为 1 divw 指令的是否配对域为 0，即该条指令不存在配对的余数指令。序号 6 divu 除法指令对应的是否配对的域置为 1。

表格 3 写入序号 6 divu 除法指令

除法指令编码 DIV_N_OP	写余数目的物理 寄存器 PHY_REG	该项有效位 valid	重排序缓存索引 ROB_ID	是否配对
011	div_phy_rem_1	1	rob_id_1	0
001	div_phy_rem_2	1	rob_id_2	1

序号 9 remu 余数指令在重命名的时候，会发现除法指令编码缓存中存在配对的除法指令，即序号为 6 的 divu 指令。此时序号为 9 的 remu 余数指令译码为一条 MOV 指令，即将配对除法指令分配的物理寄存器 div_phy_rem_2 映射到序号 9 remu 余数指令的目的寄存器。序号 9 remu 余数指令不需要到发射到除法执行单元执行。释放序号为 6 的 divu 指令在除法指令编码缓存中的资源。

表格 4 释放序号 6 divu 除法指令

除法指令编码 DIV_N_OP	写余数目的物理 寄存器 PHY_REG	该项有效位 valid	重排序缓存索引 ROB_ID	是否配对
011	div_phy_rem_1	1	rob_id_1	0

序号 16 remu 余数指令在重命名的时候，会发现除法指令编码缓存中不存在配对的除法指令。此时，该条指令需要发射到除法执行单元计算余数。

序号 22 divuw 除法指令在重命名的时候，申请 2 个物理寄存器 div_phy_quo_3 和 div_phy_rem_3。序号 22 divuw 指令分配的 ROB_ID 为 rob_id_3。并且将这些信息写入到除法指令编码缓存。同时默认该条除法指令存在与它配对的余数指令，即是否配对的域置为 1。

表格 5 写入序号 22 divuw 除法指令

除法指令编码 DIV_N_OP	写余数目的物理 寄存器 PHY_REG	该项有效位 valid	重排序缓存索引 ROB_ID	是否配对
011	div_phy_rem_1	1	rob_id_1	0
010	div_phy_rem_3	1	rob_id_3	1

序号 25 remu 余数指令在重命名的时候，会发现除法指令编码缓存中不存在配对的除法指令。此时，该条指令需要发射到除法执行单元计算余数。同时标记序号为 22 divuw 指令的是否配对为 0，即该条指令不存在配对的余数指令。

表格 6 释放序号 22 divuw 除法指令

除法指令编码 DIV_N_OP	写余数目的物理 寄存器 PHY_REG	该项有效位 valid	重排序缓存索引 ROB_ID	是否配对
011	div_phy_rem_1	1	rob_id_1	0
010	div_phy_rem_3	1	rob_id_3	0

实施例 4

本实施例公开一种 RISC-V 指令集余数指令的实现系统，所述系统用于执行 RISC-V 指令集余数指令的实现方法，包括寄存器、执行单元、除法单元、指令译码单元和取指令单元。

本发明在重命名阶段通过增加一个余数指令加速单元实现余数指令的功能，当除法和余数指令配对出现时，余数指令不需要发射到后续除法执行单元，而是通过余数指令的目的寄存器映射到除法指令写余数的物理寄存器，取到除法指令产生的余数，余数指令执行效率高。

以上实施例仅用以说明本发明的技术方案，而非对其限制；尽管参照前述实施例对本发明进行了详细的说明，本领域的普通技术人员应当理解：其依然可以对前述各实施例所记载的技术方案进行修改，或者对其中部分技术特征进行等同替换；而这些修改或者替换，并不使相应技术方案的本质脱离本发明各实施例技术方案的精神和范围。

1、一种 RISC-V 指令集余数指令的实现方法，其特征在于，所述方法包括以下步骤：

S1 对 CPU 乱序执行，指令从取指单元进入指令译码单元，进行指令译码；

S2 译码完成后的指令在重命名单元进行目的寄存器的重命名，同时对余数指令进行优化；

S3 若余数指令不满足优化条件时，重命名之后的指令进入保留站后，进入执行单元执行；

S4 执行完成后的指令通过重排序缓存提交，并释放在重命名阶段分配的除法指令编码缓存资源。

2、根据权利要求 1 所述的 RISC-V 指令集余数指令的实现方法，其特征在于，所述方法中，当除法和余数指令配对出现时，通过余数指令的目的寄存器映射到除法指令写余数的物理寄存器，取到除法指令产生的余数。

3、根据权利要求 1 所述的 RISC-V 指令集余数指令的实现方法，其特征在于，所述方法中，在重命名阶段出现余数指令时，检索余数指令加速单元中的除法指令编码缓存，如果除法指令的编码和余数指令的编码匹配成功，那么该条余数指令可以优化；

如果除法指令的编码和余数指令的编码没匹配成功，那么该条余数指需要到执行单元执行，计算得到余数。

4、根据权利要求 3 所述的 RISC-V 指令集余数指令的实现方法，其特征在于，所述方法中，判断余数指令匹配不成功的规则为：当连续不同类型的除法指令、连续不同类型的余数指令或者除法指令与余数指令不匹配；

当除法指令与余数指令判断为不匹配时，余数指令加速单元中的是否配对的域置为 0。

5、根据权利要求1所述的RISC-V指令集余数指令的实现方法，其特征在于，所述方法中，当除法指令写入到除法指令编码缓存的时候，需要判断是否有空闲的entry，将除法指令的信息写到对应的entry；当余数指令的标识rem_val有效时，表示当前指令为余数指令，如果有效位valid有效，那么该条余数指令匹配成功。

6、根据权利要求1所述的RISC-V指令集余数指令的实现方法，其特征在于，所述方法中，除法指令在重命名阶段申请物理寄存器div_phy_quo和div_phy_rem，用于分别存放除法指令的商和余数，其中除法指令的div_phy_quo更新到除法指令目的寄存器重命名映射表RAT，div_phy_rem存放到除法指令编码缓存的写余数目的寄存器PHY_REG。

7、根据权利要求1所述的RISC-V指令集余数指令的实现方法，其特征在于，所述方法中，当除法指令进入重命名阶段时，当不存在配对的余数指令，除法指令根据编码，将除法指令信息写入除法指令编码缓存，除法指令写缓存时，先从缓存中找到空闲的位置，然后把除法指令的编码DIV_N_OP、保存配对余数指令结果的物理寄存地址div_phy_rem及除法指令除法指令的重排序ROB_ID写到缓存中，此时除法指令编码缓存的有效位valid置为1；

当余数指令进入重命名阶段时，余数指令编码REM_N_OP和除法指令编码DIV_N_OP进行匹配检查，根据除法指令DIV是和余数指令REM配对规则进行检查；如果余数指令比较命中，则将目的寄存器rem_rd的映射关系映射到rem_phy_reg，并且更新到目的寄存器重命名映射表RAT，余数指令执行完成，更新在重排序缓存中指令执行完成指示，并且释放除法指令编码缓存资源。

8、根据权利要求1所述的RISC-V指令集余数指令的实现方法，其特征在于，所述方法中，当出现刷新、复位或者后续新的除法指令或者余数指令时，释放除法指令申请的物理寄存器，并且释放除法指

令编码缓存；

当除法指令在 ROB 提交的时候，根据提交指针 `cm_ptr` 得到除法指令的 `ROB_ID`，`ROB_ID` 去检索除法指令编码缓存，如果除法指令编码缓存不是因为异常刷新或者分支指令预测错误刷新释放，那么该位置为余数指令配对时释放的情况；

当存在指令与正在提交的除法指令配对时，配对的余数指令释放物理寄存器 `div_phy_quo`，当不存在余数指令与正在提交的除法指令配对时，除法指令同时释放物理寄存器 `div_phy_quo` 和物理寄存器 `div_phy_rem`。

9、根据权利要求 1 所述的 RISC-V 指令集余数指令的实现方法，其特征在于，所述方法中，当一条余数指令在重命名阶段，除法指令编码缓存没有匹配的除法指令时，这条余数指令需要送到指令执行单元，指令计算得到余数，并且更新到余数目的寄存器。

10、一种 RISC-V 指令集余数指令的实现系统，所述系统用于执行如权利要求 1-9 任一项所述的 RISC-V 指令集余数指令的实现方法，其特征在于，包括寄存器、执行单元、除法单元、指令译码单元和取指令单元。