



(12) 发明专利申请

(10) 申请公布号 CN 114968356 A

(43) 申请公布日 2022. 08. 30

(21) 申请号 202110527272.7

(22) 申请日 2021.05.14

(66) 本国优先权数据

202110188725.8 2021.02.19 CN

(71) 申请人 华为技术有限公司

地址 518129 广东省深圳市龙岗区坂田华为总部办公楼

(72) 发明人 张嘉临 麻鹏豪 王哲

(51) Int. Cl.

G06F 9/30 (2006.01)

G06F 9/38 (2006.01)

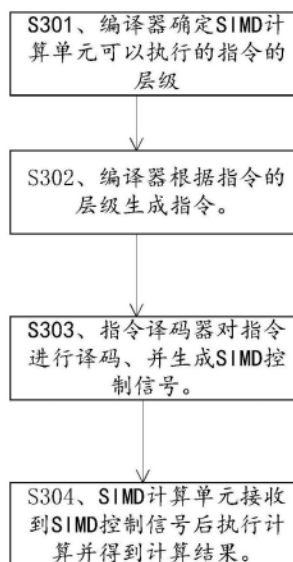
权利要求书3页 说明书14页 附图5页

(54) 发明名称

一种数据处理的方法、装置、处理器和计算设备

(57) 摘要

本申请公开了一种数据处理的方法,该方法应用于处理器,其中,处理器包括单指令多数据流计算单元、编译器、指令译码器和微码只读存储器,微码只读存储器用于存储第一指令和至少两个子指令之间的对应关系。则该方法具体包括:编译器获取数据处理的任务并将该任务编译为第一指令;指令译码器获取第一指令,并根据第一指令和至少两个子指令之间的对应关系,将第一指令分解为子指令;最后SIMD计算单元执行子指令获得数据处理的任务的处理结果。通过编译器和指令译码器之间的配合实现复杂数学任务的向量化计算。



1. 一种数据处理的方法,其特征在于,所述方法应用于处理器,所述处理器包括单指令多数据流(single instruction multipledata, SIMD)计算单元、编译器、指令译码器和微码只读存储器,所述微码只读存储器用于存储第一指令和至少两个子指令之间的对应关系,所述方法包括:

编译器获取数据处理的任务并将所述任务编译为第一指令;

所述指令译码器获取所述第一指令,并根据所述对应关系,将所述第一指令分解为所述至少两个子指令;

所述指令译码器将所述子指令发送给SIMD计算单元;

所述SIMD计算单元执行所述子指令以获得所述数据处理的任務的处理结果。

2. 根据权利要求1所述的数据处理的方法,其特征在于,所述第一指令为第一层级的指令,所述至少两个子指令为第二层级的指令,所述第一层级的指令的操作的复杂程度高于所述第二层级的指令的操作的复杂程度,所述SIMD计算单元支持执行所述第二层级的指令。

3. 根据权利要求1或2任一所述的数据处理的方法,其特征在于,所述至少两个子指令包括Neon指令集和可伸缩适量扩展(SVE)指令集中任意一个指令。

4. 根据权利要求3所述的数据处理的方法,其特征在于,所述至少两个子指令还包括由所述Neon指令集和所述SVE指令集中至少两个指令组成的指令。

5. 根据权利要求1至4任一所述的数据处理的方法,其特征在于,所述编译器将所述任务编译为第一指令之前,所述方法还包括:

所述编译器获取向量化函数库中的函数,所述向量化函数库包括所述Neon指令集和所述SVE指令集中任意一个指令的操作所对应的函数。

6. 根据权利要求5所述的数据处理的方法,其特征在于,所述向量化函数库还包括由所述Neon指令集和所述SVE指令集中至少两个指令组成的指令的操作所对应的函数。

7. 根据权利要求5至6任一所述的数据处理的方法,其特征在于,当所述任务的代码所包含的函数属于所述向量化函数库中的函数时,所述第一指令包括向量指令,所述编译器将所述任务编译为第一指令,具体包括:

所述编译器将所述任务编译为所述向量指令。

8. 根据权利要求5至6任一所述的数据处理的方法,其特征在于,当所述任务的代码所包含的函数不属于所述向量化函数库中的函数时,所述第一指令包括第一跳转指令和向量指令,所述编译器将所述任务编译为第一指令,具体包括:

所述编译器将所述任务编译为所述第一跳转指令;

所述编译器将所述任务的操作在向量化函数库中的函数编译为所述向量指令,所述第一跳转指令指向所述向量指令的地址。

9. 根据权利要求8所述的数据处理的方法,其特征在于,所述第一指令还包括批量数据处理指令,所述第一跳转指令指向所述批量数据处理指令的地址,所述编译器将所述任务编译为第一指令,还包括:

所述编译器将批量数据处理函数编译为所述批量数据处理指令,所述批量数据处理指令包括第二跳转指令,所述第二跳转指令指向所述向量指令的地址。

10. 根据权利要求1至9任一所述的数据处理的方法,其特征在于,所述数据处理的任务

包括初等函数的计算或者非线性函数的计算。

11. 一种数据处理的装置,其特征在于,所述装置包括编译单元、译码单元、执行单元和第一存储单元:

所述第一存储单元用于存储第一指令和至少两个子指令之间的对应关系;

所述编译单元,用于获取数据处理的任务并将所述任务编译为第一指令;

所述译码单元,用于根据所述对应关系,将所述第一指令分解为至少两个所述子指令;还用于将所述子指令发送给执行单元;

所述执行单元,用于执行所述子指令获得以所述数据处理的处理结果。

12. 根据权利要求11所述的数据处理的装置,其特征在于,所述第一指令为第一层级的指令,所述至少两个子指令为第二层级的指令,所述第一层级的指令的操作的复杂程度高于所述第二层级的指令的操作的复杂程度,所述执行单元的硬件电路支持执行所述第二层级的指令。

13. 根据权利要求11或12任一所述的数据处理的装置,其特征在于,所述至少两个子指令包括Neon指令集和可伸缩适量扩展指令集(SVE)中任意一个指令。

14. 根据权利要求13所述的数据处理的装置,其特征在于,所述至少两个子指令还包括由所述Neon指令集和所述SVE指令集中任意指令中至少两个指令组成的指令。

15. 根据权利要求11至14任一所述的数据处理的装置,其特征在于,所述装置还包括第二存储单元,所述第二存储单元用于存储向量化函数库,所述向量化函数库包括所述Neon指令集和所述SVE指令集中任意一个指令的操作所对应的函数,则所述编译器还用于获取所述向量化函数库中的函数。

16. 根据权利要求15所述的数据处理的装置,其特征在于,所述向量化函数库还包括由所述Neon指令集和所述SVE指令集中至少两个指令组成的指令的操作所对应的函数。

17. 根据权利要求15至16任一所述的数据处理的装置,其特征在于,当所述任务的代码所包含的函数属于所述向量化函数库中的函数时,所述第一指令包括向量指令,所述编译单元还用于:

将所述任务编译为所述第一指令。

18. 根据权利要求15至16任一所述的数据处理的装置,其特征在于,当所述任务的代码所包含的函数不属于所述向量化函数库中的函数时,所述第一指令包括第一跳转指令和向量指令,所述编译单元还用于:

将所述任务编译为所述第一跳转指令;

将所述任务的操作在向量化函数库中的函数编译为所述向量指令,所述第一跳转指令指向所述向量指令的地址。

19. 根据权利要求18所述的数据处理的装置,其特征在于,所述第一指令还包括批量数据处理指令,所述第一跳转指令指向所述批量数据处理指令的地址,所述编译单元还用于:

将批量数据处理函数编译为所述批量数据处理指令,所述批量数据处理指令包括第二跳转指令,所述第二跳转指令指向所述向量指令的地址。

20. 根据权利要求11至19任一所述的数据处理的方法,其特征在于,所述数据处理的任任务包括初等函数的计算或者非线性函数的计算。

21. 一种处理器,其特征在于,所述处理器包括SIMD计算单元、指令译码器、微码只读存

储器和编译器：

所述微码只读存储器，用于存储第一指令和至少两个子指令之间的对应关系；

所述编译器，用于获取数据处理的任务并将所述任务编译为第一指令；

所述指令译码器，用于获取第一指令，并根据所述对应关系，将所述第一指令分解为所述子指令；还用于将所述子指令发送给SIMD计算单元；

所述SIMD计算单元，用于执行所述子指令获得所述数据处理的处理结果。

22. 根据权利要求21所述的处理器，其特征在于，所述第一指令为第一层级的指令，所述至少两个子指令为第二层级的指令，所述第一层级的指令的操作的复杂程度高于所述第二层级的指令的操作的复杂程度，所述执行单元的硬件电路支持执行所述第二层级的指令。

23. 根据权利要求21或22任一所述的处理器，其特征在于，所述至少两个子指令包括Neon指令集和可伸缩适量扩展指令集(SVE)中任意一个指令。

24. 根据权利要求23所述的处理器，其特征在于，所述至少两个子指令还包括由所述Neon指令集和所述SVE指令集中任意指令中至少两个指令组成的指令。

25. 根据权利要求21至24任一所述的处理器，其特征在于，所述编译器还用于获取向量化函数库中的函数，所述向量化函数库包括所述Neon指令集和所述SVE指令集中任意一个指令的操作所对应的函数。

26. 根据权利要求25所述的数据处理的装置，其特征在于，所述向量化函数库还包括由所述Neon指令集和所述SVE指令集中至少两个指令组成的指令的操作所对应的函数。

27. 根据权利要求25至26任一所述的处理器，其特征在于，当所述任务的代码所包含的函数属于所述向量化函数库中的函数时，所述第一指令包括向量指令，所述编译器还用于：将所述任务编译为所述第一指令。

28. 根据权利要求25至26任一所述的处理器，其特征在于，当所述任务的代码所包含的函数不属于所述向量化函数库中的函数时，所述第一指令包括第一跳转指令和向量指令，所述编译器还用于：

将所述任务编译为所述第一跳转指令；

将所述任务的操作在向量化函数库中的函数编译为所述向量指令，所述第一跳转指令指向所述向量指令的地址。

29. 根据权利要求28所述的处理器，其特征在于，所述第一指令还包括批量数据处理指令，所述第一跳转指令指向所述批量数据处理指令的地址，所述编译器还用于：

将批量数据处理函数编译为所述批量数据处理指令，所述批量数据处理指令包括第二跳转指令，所述第二跳转指令指向所述向量指令的地址。

30. 根据权利要求21至29任一所述的处理器，其特征在于，所述数据处理的任务包括初等函数的计算或者非线性函数的计算。

31. 一种计算设备，其特征在于，所述计算设备包括如权利要求21至30任一所述的处理器和存储器，所述存储器用于存储向量化函数库。

一种数据处理的方法、装置、处理器和计算设备

技术领域

[0001] 本申请涉及计算机领域,尤其涉及一种数据处理的方法、装置、处理器和计算设备。

背景技术

[0002] 随着数据量的增大和对实时性要求的提高,各类高性能计算(high performance computing,HPC)应用对计算的性能提出了更高的要求,通过单指令多数据流(single instruction multiple data,SIMD)计算单元实现向量化计算就是其中重要方法之一。SIMD通过复杂的电路实现对一组数据(即向量)同时处理,在批量处理数据时,相对于单指令单数据流(single instruction single data,SISD)理论上可以做到成倍的性能提升,任何需要对大量数据作相同处理的场景都可以用到SIMD技术实现性能提升,HPC应用也包括在其中。

[0003] 处理器的指令集从体系结构上分为精简指令集和复杂指令集,与CISC指令集相比,RISC指令集的指令数目少,并且每条指令采用相同的字节长度,在字边界上对齐,字段位置固定,因此RISC指令集可以大大简化处理器的控制器和其他功能单元的设计,不必使用大量专用寄存器,特别是允许以硬件线路来实现指令操作,从而节约处理器的制造成本。然而,在针对SIMD计算的场景下,RISC并不具有直接实现对应功能的硬件结构,需要依赖软件实现对应功能,并且由于RISC指令集中非线性函数计算这样相对复杂的数学计算并没有对应单条指令实现。因此,目前处理器上的向量化函数库的功能并不完备,大多为基础的四则运算,且存在功能拓展困难、使用复杂、性能不够高等问题,并不能在HPC场景下充分发挥SIMD指令集的优势,最终导致处理器实现SIMD功能的计算的性能低下。

[0004] 目前针对采用RISC架构的处理器上的SIMD指令集主要包括Neon指令集和可伸缩适量扩展(scalable vector extension,SVE)指令集,如何提供一种利用现有的SIMD指令集,使得处理器能使用SIMD技术处理复杂的数学计算成为亟待解决的技术问题。

发明内容

[0005] 本申请提供了一种数据处理的方法、装置、处理器和计算设备,以此使得处理器能使用SIMD技术处理复杂的数学计算。

[0006] 第一方面,提供一种数据处理的方法,该方法应用于处理器,其中,处理器包括单指令多数据流(single instruction multiple data,SIMD)计算单元、编译器、指令译码器和微码只读存储器,微码只读存储器用于存储第一指令和至少两个子指令之间的对应关系,则该方法包括:首先编译器获取数据处理的任务并将该任务编译为第一指令。进一步,由指令译码器获取第一指令,并根据第一指令和至少两个子指令之间的对应关系,将第一指令分解为至少两个子指令。然后,指令译码器将子指令发送给SIMD计算单元,最后SIMD计算单元执行子指令以获得数据处理的任务的处理结果。通过上述方法,可以协同使用编译器和指令译码器,根据第一指令和至少两个子指令之间的对应关系,指令译码器可以将复

杂指令分解为简单指令,减少指令译码器取指令的时间消耗,加快计算速度。同时还可以将数据处理的任务转化为处理器的SIMD计算单元可以执行的指令,从而使得可以使用SIMD技术实现更为复杂的计算,减小硬件的复杂度。

[0007] 在一种可能的实现方式中,第一指令为第一层级的指令,至少两个子指令为第二层级的指令,其中,第一层级的指令的操作的复杂程度高于第二层级的指令的操作的复杂程度,SIMD计算单元支持执行所述第二层级的指令。需要说明的是,本申请对指令的层级的数量并不做限定,示例性地,指令可以被分为现有指令层指令、基础计算层指令、计算步骤层指令和函数层指令,其中第一指令可以为函数层指令,对应地,至少两个子指令可以为至少两个基础计算层指令,也可以为至少两个现有指令层指令,还可以为至少一个现有指令层指令和至少一个计算步骤层指令。

[0008] 在另一种可能的实现方式中,至少两个子指令包括Neon指令集和可伸缩适量扩展(SVE)指令集中任意一个指令。则指令译码器支持将负责指令解码为现有指令集的指令,可以对现有指令集进行扩展。其中,两个子指令可以均为Neon指令集中至少两个指令,也可以为SVE指令集中至少两个指令,还可以分别为至少一个Neon指令集的指令和至少一个SVE指令集中指令。

[0009] 在另一种可能的实现方式中,至少两个子指令还包括由Neon指令集和SVE指令集中至少两个指令组成的指令。其中,两个子指令可以均为Neon指令集中至少两个指令组成的指令,也可以为可伸缩适量扩展指令集中至少两个指令组成的指令,还可以分别为至少一个Neon指令集和至少一个可伸缩适量扩展指令集中组成的指令。

[0010] 在另一种可能的实现方式中,编译器将数据处理的任务编译为第一指令之前,还可以首先获取向量化函数库中的函数,其中,向量化函数库包括Neon指令集和SVE指令集中任意一个指令的操作所对应的函数。通过上述方法,编译器可以根据数据处理的任务和向量化函数库中函数的关系采取不同的编译策略,满足不同形式的任务的需求。

[0011] 在另一种可能的实现方式中,向量化函数库还包括由Neon指令集和SVE指令集中至少两个指令组成的指令的操作所对应的函数。

[0012] 在另一种可能的实现方式中,当任务的代码所包含的函数属于向量化函数库中的函数时,第一指令包括向量指令,编译器将任务编译为第一指令,具体包括:编译器将任务编译为向量指令。通过上述方法,当任务的代码中直接使用向量化函数库中的函数时,可以直接使用编译器将任务编译为可以使用SIMD技术的指令。

[0013] 在另一种可能的实现方式中,当任务的代码所包含的函数不属于向量化函数库中的函数时,第一指令包括第一跳转指令和向量指令,则编译器将任务编译为第一指令,具体包括:编译器将任务编译为第一跳转指令,并且将任务的操作在向量化函数库中的函数编译为向量指令,跳转指令指向向量指令的地址。通过上述方法,当数学处理的任务的代码中未使用向量化函数库中的函数时,编译器可以将数学处理的任务编译为跳转指令,指向向量化函数库中与数学处理任务的操作相同的函数编译后的指令,从而在执行时使用SIMD技术的指令。

[0014] 在另一种可能的实现方式中,第一指令还包括批量数据处理指令,第一跳转指令指向批量数据处理指令的地址,则方法还包括:编译器将批量数据处理函数编译为批量数据处理指令,其中,批量数据处理指令包括第二跳转指令,第二跳转指令指向向量指令的地

址。通过上述方法,在可以在执行向量指令之前首先执行批量数据处理指令,对数据处理的输入数据进行预处理。

[0015] 在另一种可能的实现方式中,数据处理的任务包括初等函数的计算或者非线性函数的计算。

[0016] 在另一种可能的实现方式中,指令的层级按照对应操作的复杂程度,从简单到复杂被分为现有指令层指令、基础计算层指令、计算步骤层指令和函数层指令。处理器的SIMD计算单元支持基础计算层的指令,微码只读存储器支持计算步骤层到基础计算层以及计算步骤层到现有指令层的解码。则计算 $\ln(x)$ 的方法包括:编译器将 $\ln(x)$ 任务的源代码编译为第一跳转指令,同时编译器将向量化函数库中的 $\ln(x)$ 函数编译为计算步骤层的指令,即乘法缩减指令、近似指令和范围重构指令,第一跳转指令指向乘法缩减指令的地址。指令译码器分别获取上述指令,并协同微码只读存储器106将乘法缩减指令分解为抽取指数位指令和抽取尾数位指令并译码生成相应的SIMD控制信号,将近似指令分解为多项式近似指令并译码生成相应的SIMD控制信号,以及将范围重构指令分解为浮点数加法指令和浮点数乘法指令并译码生成相应的SIMD控制信号。最后现有指令层的指令分别由SIMD计算单元执行,得到 $\ln(x)$ 的计算结果。通过上述方法,可以扩展现有的SIMD指令,使用硬件电路直接实现基础操作层的指令,提高计算速度,同时,利用微码只读存储器将微指令组合为计算步骤层的复杂指令,进一步减少取指令所消耗的计算时间,提高计算速度。

[0017] 在另一种可能的实现方式中,指令的层级按照对应操作的复杂程度,从简单到复杂被分为现有指令层指令、基础计算层指令、计算步骤层指令和函数层指令。处理器的SIMD计算单元仅支持基础计算层的指令,微码只读存储器仅支持基础计算层的指令到现有指令层的解码。则计算 $\ln(x)$ 的方法包括:编译器将 $\ln(x)$ 任务的源代码编译为第一跳转指令,同时编译器将向量化函数库中的 $\ln(x)$ 函数编译为基础计算层的指令,即抽取指数位指令、抽取尾数位指令、多项式近似指令、浮点数加法指令和浮点数乘法指令,第一跳转指令指向抽取指数位指令的地址。指令译码器分别获取上述指令,并将抽取指数位指令、抽取尾数位指令、多项式近似指令、浮点数加法指令和浮点数乘法指令译码并生成相应地SIMD控制信号,分别由SIMD计算单元执行,得到 $\ln(x)$ 的计算结果。通过上述方法,可以扩展现有的SIMD指令,使用硬件电路直接实现基础操作层的指令,提高计算速度。

[0018] 在另一种可能的实现方式中,指令的层级按照对应操作的复杂程度,从简单到复杂被分为现有指令层指令、基础计算层指令、计算步骤层指令和函数层指令。处理器的SIMD计算单元仅支持现有指令层的指令,即现有SIMD指令,微码只读存储器支持计算步骤层到现有指令层的解码。则计算 $\ln(x)$ 的方法包括:编译器将向量化函数库中的 $\ln(x)$ 函数编译为计算步骤层的指令,即乘法缩减指令、近似指令和范围重构指令,第一跳转指令指向乘法缩减指令的地址。指令译码器分别获取上述指令,并协同微码只读存储器依次将计算步骤层的指令分解为现有指令层的指令,具体为:指令译码器协同微码只读存储器将乘法缩减指令分解为与指令和移位指令并译码生成相应的SIMD控制信号,将近似指令分解为FMA指令并译码生成相应的SIMD控制信号,以及将范围重构指令分解为浮点数加法指令和浮点数乘法指令并译码生成相应的SIMD控制信号。最后现有指令层的指令分别由SIMD计算单元执行,得到 $\ln(x)$ 的计算结果。通过上述方法,可以扩展现有的SIMD指令,使用硬件电路直接实现基础操作层的指令,提高计算速度。

[0019] 在另一种可能的实现方式中,指令的层级按照对应操作的复杂程度,从简单到复杂被分为现有指令层指令、基础计算层指令、计算步骤层指令和函数层指令。假设处理器的SIMD计算单元仅支持现有指令层的指令,微码只读存储器不支持解码。则计算 $\ln(x)$ 的方法包括:编译器将 $\ln(x)$ 任务的源代码编译为第一跳转指令,同时编译器将向量化函数库中的 $\ln(x)$ 函数编译为现有指令层的指令,即与指令、移位指令、FMA指令、浮点数加法指令以及浮点数乘法指令,第一跳转指令指向与指令的地址。指令译码器分别获取上述指令,并将与指令、移位指令、位指令、FMA指令、浮点数加法指令以及浮点数乘法指令译码并生成相应地SIMD控制信号,分别由SIMD计算单元执行,得到 $\ln(x)$ 的计算结果。通过上述方法,在处理器的硬件上并没有针对数据处理的优化,所有层级的操作均由编译器实现。分层操作的函数设计不仅保证了每一层的操作的完整性和扩展性,用户可以分别控制每一层操作的具体流程以完成对数据处理的细节上的优化。

[0020] 第二方面,提供一种数据处理的装置,包括编译单元、译码单元、执行单元和存储单元,所述数据处理的装置包括用于执行第一方面或第一方面任一种可能实现方式中的数据处理的各个模块。

[0021] 第三方面,提供一种处理器,包括SIMD计算单元、指令译码器、微码只读存储器和编译器,其中,SIMD计算单元、指令译码器、微码只读存储器和编译器分别用于执行第一方面或第一方面任一种可能实现方式中相应主体所执行的方法的操作步骤。

[0022] 第四方面,提供一种数据处理的计算设备,包括处理器和存储器,其中,存储器用于存储向量化函数库,处理器包括SIMD计算单元、编译器、指令译码器和微码只读存储器。SIMD计算单元、指令译码器、微码只读存储器和编译器分别用于执行第一方面或第一方面任一种可能实现方式中相应主体所执行的方法的操作步骤。

[0023] 本申请在上述各方面提供的实现方式的基础上,还可以进行进一步组合以提供更多实现方式。

附图说明

[0024] 图1为本申请实施例提供的一种数据处理的计算设备100的结构示意图;

[0025] 图2是本申请实施例提供的一种数据处理的处理器10的硬件结构图;

[0026] 图3是本申请实施例提供的一种数据处理的方法的流程示意图;

[0027] 图4是本申请实施例提出的一种4层的操作的分层的示意图;

[0028] 图5是本申请实施例提出的一种4层的 $\ln(x)$ 操作的函数的示意图;

[0029] 图6是本申请实施例提供的一种实现 $\ln(x)$ 的数据处理的方法的流程示意图;

[0030] 图7为本申请实施例提供的一种数据处理的装置700的示意图。

具体实施方式

[0031] 下面结合附图对本申请实施例中的技术方案进行描述,图1为本申请实施例提供的一种数据处理的计算设备100的结构示意图,如图所示,计算设备100包括处理器10、存储器20和总线30。其中,处理器10和存储器20通过总线30连接,总线30可以是数据总线,还可以是电源总线、控制总线和状态信号总线等,该总线也可以为其他类型实现计算设备内器件间连接的总线。

[0032] 应用程序的源代码用于存储需要进行数据处理的任务的指令,为了便于用户使用和保持较高的可读性,这些任务的指令通常使用高级语言编写,并保存在存储器20中。

[0033] 存储器20用于存储应用程序运行的数据和指令,该存储器20可以包括只读存储器和随机存取存储器,还可以包括非易失性随机存取存储器。

[0034] 该存储器20可以是易失性存储器或非易失性存储器,或可包括易失性和非易失性存储器两者。其中,非易失性存储器可以是只读存储器(read-only memory,ROM)、可编程只读存储器(programmable ROM,PROM)、可擦除可编程只读存储器(erasable PROM,EPROM)、电可擦除可编程只读存储器(electrically EPROM,EEPROM)或闪存。易失性存储器可以是随机存取存储器(random access memory,RAM),其用作外部高速缓存。通过示例性但不是限制性说明,许多形式的RAM可用,例如静态随机存取存储器(static RAM,SRAM)、动态随机存取存储器(DRAM)、同步动态随机存取存储器(synchronous DRAM,SDRAM)、双倍数据速率同步动态随机存取存储器(double data rate SDRAM,DDR SDRAM)、增强型同步动态随机存取存储器(enhanced SDRAM,ESDRAM)、同步连接动态随机存取存储器(synchlink DRAM,SLDRAM)和直接存储器总线随机存取存储器(direct rambus RAM,DR RAM)。

[0035] 编译器50运行于处理器10中。由于处理器无法识别与执行高级语言编写的指令,编译器50用于将数据处理的任务中使用的高级语言的指令转化为二进制代码表示的指令,存放在存储器20中。处理器10通过从存储器20中取出计算任务的指令并执行来得到数据处理的计算结果。处理器10可以为中央处理器(central processing unit,CPU)或者是其他通用处理器,处理器10可以为使用ARM架构的处理器,或者是采用RISC指令集的其他架构的处理器。

[0036] 图2是本实施例提供的一种数据处理的处理器10的硬件结构图,可以进一步阐述处理器10执行应用程序发布的数据处理的任务的指令的流程。

[0037] 如图2所示,处理器10包括存储器地址寄存器101、存储器数据寄存器102、程序计数器103、指令寄存器104、指令译码器105、微码只读存储器106、操作控制器107、计算单元108、通用寄存器109、累加器110、状态寄存器111、时序电路112以及处理器总线120。其中,处理器总线120也可以是数据总线、电源总线、控制总线或者状态信号总线等。

[0038] 在处理器10中,存储器地址寄存器101和存储器数据寄存器102分别用于保存当前处理器10所访问的存储器20的地址,以及处理器10从该地址中读取或写入的数据和读取或写入的指令,以便弥补处理器和存储器之间存在的操作速度上的差异。

[0039] 时序电路112通过固定时钟为各部件提供一个时间基准,处理器10执行一条指令的时间为一个指令周期。程序计数器103用于存放下一条指令的地址,当指令顺序执行时,每取一条指令后程序计数器103自动加上一条指令的字节数;当遇到转移指令时,程序计数器103通过转移指令中的地址码字段来指定下一条指令的地址。指令寄存器104用于保存当前正在执行的指令。指令包括操作码和地址码两个字段,操作码部分由指令译码器105对操作码部分进行译码,以产生指令所要求操作的控制电位。操作控制器107可以根据指令译码器输出的控制电位信号和时序电路112产生的时序信号,生成各种操作控制信号,控制处理器10其余部件完成取指令和执行指令的操作。

[0040] 微指令是处理器执行指令的最小单位,一条指令可以是单个微指令,也可以由若干个微指令组成,例如弹出寄存器ebx中存储的数据的指令,可以由载入指令、存储指令以

及加法指令组合而成,其中,载入指令、存储指令以及加法指令都是微指令,如下代码所示:

```
        load temp [esp];  
[0041]  pop [ebx]; = store [ebx], temp;  
        add esp, 4;
```

[0042] 由多条微指令组合的指令称为复杂指令,复杂指令对应的微指令的步骤存储在微码只读存储器106,指令译码器105在译码的过程中,可以从微码只读存储器106中查询并得到组成复杂指令的微指令的操作码和地址码,依次对微指令的操作码部分进行译码,产生微指令所要求操作的控制电位。

[0043] 通用寄存器组109用于根据指令的地址码,保存地址码对应的数据。计算单元108用于接收操作控制器107的操作控制信号并对保存在通用寄存器组109中的数据执行计算,包括算术运算(包括加减乘数等基本运算及其附加运算)和逻辑运算(包括移位、逻辑测试或两个值比较)。计算中产生的临时变量存放在累加器110中,产生的状态的信息存放在程序状态字寄存器111中,例如运算结果进/借位标志(C)、运算结果溢出标志(O)、运算结果为零标志(Z)、运算结果为负标志(N)、运算结果符号标志(S)等。程序状态字寄存器还用来保存中断和计算设备工作状态等信息,以便处理器10及时了解机器运行状态和程序运行状态。

[0044] 计算单元108中包含多种不同的电路模块,可以分别用于执行不同的指令。特别的,在指令中,SIMD指令用于完成对向量的算数运算和逻辑运算,则相应地,计算单元108也包括SIMD计算单元1081,用于执行SIMD指令。

[0045] 下面以一个指令周期为例,阐述处理器执行一条SIMD指令的步骤,具体如下:

[0046] 1) 初始化程序计数器103:由程序计数器103指定指令在存储器20中的地址,该指令可以为一条SIMD指令,也可以为多条SIMD指令组成的复杂指令。

[0047] 2) 取指令:根据程序计数器103的地址以及操作控制器107的控制信号,指令通过处理器总线120从存储器数据寄存器102中送入指令寄存器104中。

[0048] 3) 指令解码:指令译码器105从指令寄存器104中获取指令,判断指令是否为复杂指令。当指令为复杂指令时,指令译码器105从微码只读存储器106中查询并得到组成复杂指令的SIMD指令的操作码和地址码;当指令为简单指令时,指令译码器105直接获得SIMD指令的操作码和地址码。根据SIMD指令的操作码,指令译码器105生成指令所要求操作的控制电位,并将其送到操作控制器107中,由操作控制器107生成不同的SIMD操作控制信号。

[0049] 4) 执行指令:根据指令的操作码的不同,操作控制器107生成不同的SIMD操作控制信号,控制处理器10不同的组件完成指令的操作。例如,当指令为读取数据指令时,在操作控制器107的操作控制信号的协调下,SIMD指令的地址码对应的数据通过处理器总线120,从存储器数据寄存器102中送入通用寄存器组109中。每个通用寄存器的长度是固定的,为SIMD指令所能执行元素的最大值。又例如,当指令为SIMD的加法指令时,计算单元108从通用寄存器组109中获取两个数据,调用SIMD计算单元完成计算,并将计算结果保存在通用寄存器组109中。如果加法运算产生超出处理器能处理的数据大小的结果,在程序状态字寄存器111中设置运算结果溢出标志。

[0050] 5) 写回计算结果:运算结果被写进处理器的存储器数据寄存器102中,以供随后指令快速存取。

[0051] 6) 更新程序计数器103:程序计数器103中生成下一条指令的地址。

[0052] SIMD计算单元1081根据硬件电路的不同可以执行不同的SIMD指令,现有的SIMD计算单元1081的硬件电路比较简单,可以用于执行基础的四则运算的SIMD指令。本实施例可以根据用户需求在SIMD计算单元801中使用定制的复杂硬件电路,直接接收操作控制器107发送的复杂指令,而无需将指令通过微码只读存储器106译码为不同的SIMD指令的组合。对于复杂指令,定制的SIMD计算单元只需要执行一条指令,因此可以减少运算所需的时钟周期数,缩短运算时间。

[0053] 通过本申请提出的方法,既可以采用通过硬件电路直接实现复杂指令的方法提升计算的速度,也可以采用将多条SIMD指令组合成一条复杂指令的方法,减少取指令阶段以及指令译码阶段的时钟周期数,而无需修改计算单元的硬件电路。用户可以根据计算任务的需要和实际情况,选择适当的方法,平衡处理器的计算速度和硬件复杂度。

[0054] 下面将结合图3-6对本申请实施例提供的数据处理的方法进行说明,图3是本申请实施例提供的一种数据处理的方法的流程示意图,该方法可以使用图1或图2所示的处理器10执行。如图所示,具体方法包括:

[0055] S301、编译器50确定SIMD计算单元可以执行的指令的层级。

[0056] SIMD计算单元1081完成计算时,编译器可以通过向量化实现将数据处理的任务转化为能使用SIMD计算单元1081执行的指令。向量化是指使指令同时应用于一整组数据的编程方法,应用程序的源代码中需要使用向量计算的数据处理的操作会由编译器编译为指令,该指令包括SIMD指令以及由SIMD指令组成的复杂指令。指令被执行时,由指令译码器105协同微码只读存储器106译码后生成数据处理操作的控制电位,指示SIMD计算单元完成计算。

[0057] 现有的SIMD指令集中的指令可以实现的功能并不丰富,因此对于复杂的初等函数以及非线性函数等数据处理的实现,可以在现有的SIMD指令集上进行扩展。具体地,一个数据处理的操作可以由多个子操作构成,每个子操作能实现该数据处理的部分操作。按照执行的复杂程度,可以将操作分成多个层级,其中,最低层级的操作为现有的SIMD指令集中的指令的操作,包括Neon指令集和SVE指令集,高层级操作由多个低层级操作组合得到。通过分层,可以利用最低层级的现有SIMD指令组合完成高层级的操作,从而实现复杂数据处理对应的高层级操作的向量化。在本申请的实施例中,对于分层的数量不做限定。

[0058] 图4是本申请实施例提出的一种4层的操作的分层的示意图,层级从低到高分别为现有指令层、基础计算层、计算步骤层和函数层。如图4所示,现有指令层的操作均为现有的SIMD指令集中的指令的操作,例如与操作、位操作、整形除法操作、整形乘法操作、整形减法操作以及积和熔加运算(fused multiply add,FMA)操作。基础计算层的操作可以由现有指令层的操作构成,例如,抽取指数位操作和抽取尾数位操作都可以由与操作和移位操作构成;整形取模操作由整形除法操作、整形乘法操作和整形减法操作构成;多项式近似操作由FMA;系数计算操作由减法操作和移位操作构成。进一步地,计算步骤层的每个操作又可以分别由基础计算层的操作组成,例如,乘法缩减操作可以由抽取指数位操作和抽取尾数位操作构成;加法缩减操作可以由抽取指数位操作、抽取尾数位操作以及整形取模操作构成;近似操作和插值操作可以分别由多项式近似操作和系数计算操作构成;计算步骤层的操作还可以直接由现有指令层的操作构成,例如,范围重构操作直接由浮点数加法操作和浮点

数乘法操作构成。函数层的操作则可以由计算步骤层的三个操作组合而成,分别为范围缩减操作、近似操作(或者插值操作)以及范围重构操作。其中,范围缩减操作用于将输入A通过乘法或者加法缩放到一定的数值范围中,乘法缩减可以采用以2为底的指数展开,加法缩减可以采用浮点数取余的方式,缩放后的值记为C;近似操作或者插值操作用于在数值范围内,通过查表得到数值C经过该数据处理的表达式计算后的值D,当表中没有数值C对应的值时,使用近似或者插值的方法找到与C最接近的数E,再通过查表得到数值E经过该数据处理的表达式计算后的值D;范围重构操作用于使用值D经过与范围缩减相反的操作得到输出B。最终函数层为复杂的初等函数或者非线性函数的表达式,可以直接得到输入A经过数据处理后的输出B。

[0059] 可以将实现操作的代码封装为函数,保存在向量化函数库中,向量化函数库又可以保存在存储器20中。每一个函数会由编译器编译为一个指令,指令被执行时,即可实现该函数的操作。例如,实现系数计算操作的函数可以由编译器编译为系数计算指令,指令的层级与对应的函数实现的操作的层级相同,即实现高层级操作的函数编译后得到高层级的指令。按照操作之间的组合关系,高层级操作可以通过在函数中调用低层级操作的函数来实现。其中,低层级操作的函数接收高层级参数的数据并向高层级的函数返回计算结果,函数之间的数据均以向量为单位进行计算和传输。因此按照函数之间的调用关系,每一个高层级的函数也可以由编译器编译为多个低层级的指令组合而成,所有低层级的指令都被执行完毕时,即可实现该函数的操作。例如,实现系数计算操作的函数也可以由编译器编译为减法指令和移位指令的组合。

[0060] 下面以具体的数据处理为例,进一步说明4层的操作的函数的分层方式。图5是本申请实施例提出的一种4层的 $\ln(x)$ 操作的函数的示意图,以图3中的分层方式为例,如图5所示,在现有指令层,将与操作、移位操作、FMA操作、浮点数加法操作和浮点数乘法操作分别封装为 $op_and(x)$ 函数、 $op_move(x)$ 函数、 $op_FMA(x)$ 函数、 $op_fadd(x)$ 和 $op_fmul(x)$ 函数,均为可以实现现有SIMD指令的操作的函数。在基础计算层,将抽取指数位操作、抽取尾数位操作和多项式近似操作分别封装为 $v_exindex(x)$ 函数、 $v_ex(x)$ 函数以及 $v_polyapprox(x)$ 函数,每个函数可以调用现有SIMD指令的函数。在计算步骤层,乘法缩减操作、近似操作和范围重构操作分别封装为 $v_mulreduct(x)$ 函数、 $v_approx(x)$ 函数以及 $v_construct(x)$ 函数。 $v_mulreduct(x)$ 函数中可以依次调用 $v_exindex(x)$ 函数和 $v_ex(x)$ 函数将操作分解为抽取指数位操作和抽取尾数位操作, $v_approx(x)$ 函数可以调用 $v_polyapprox(x)$ 函数将操作分解为多个多项式近似操作。同样,计算步骤层的函数中也可以直接调用现有SIMD指令操作的函数,例如 $v_construct(x)$ 函数可以调用 $op_fadd(x)$ 和 $op_fmul(x)$ 函数。函数层的操作封装为 $v_ln(x)$ 函数,提供完整的 $\ln(x)$ 函数计算功能。 $v_ln(x)$ 函数中依次调用三个函数将 $\ln(x)$ 操作分解为乘法缩减操作、近似操作和范围重构操作。

[0061] 在不同层级的指令中,现有SIMD指令所在的层级的指令可以直接指示SIMD计算单元1081完成计算,而其它层级的指令可以通过两种方式执行,具体为:

[0062] 方式一:定制SIMD计算单元1081的硬件电路执行指定层级的指令。

[0063] 由于高层级的指令由多个低层级的指令构成,直接执行高层级的指令可以减少运算所需的时钟周期数,缩短运算时间,因此在硬件设计阶段,可以根据数据处理所需的时间,设计SIMD计算单元1081硬件电路,从而使SIMD计算单元1081可以直接执行指定层级的

指令发送的控制信号,此时,该层级的指令是一条微指令,可以加入SIMD指令集中,称为扩展SIMD指令。当SIMD计算单元1081可以直接执行指定层级的指令时,也可以直接执行层级低于指定层级的指令,这些指令也将作为扩展SIMD指令加入SIMD指令集中。

[0064] 方式二:通过微码只读存储器106生成SIMD计算单元1081可以执行的SIMD指令。

[0065] 微码只读存储器106可以存储复杂指令对应的微指令的步骤,指令译码器105在译码的过程中,可以从微码只读存储器106中查询并得到组成复杂指令的微指令的操作码和地址码,将复杂指令分解为SIMD指令,从而指示SIMD计算单元完成计算。通过将微指令组合的方式实现高层级的指令,可以减少取指以及译码阶段的时钟周期数,提升运算速度,并且不用改变SIMD计算单元1081的硬件电路。因此在硬件设计阶段,也可以根据数据处理所需的时间,将指定层级的指令的分解方式预置在微码只读存储器106中,从而使SIMD计算单元1081可以处理更高层级的指令。需要说明的是,这里SIMD计算单元1081可以执行的SIMD指令包括现有SIMD指令以及扩展SIMD指令,为了描述方便,在之后的表述中,将现有SIMD指令以及扩展SIMD指令统称为SIMD指令。当将指定层级的指令的分解方式存储在微码只读存储器106中时,也可以将低于指定层级的指令的分解方式存储在微码只读存储器106中,则SIMD计算单元1081可以协同指令译码器106共同处理指定层级的指令以及低于指定层级的指令。

[0066] 在处理器执行数据处理前,可以根据SIMD计算单元以及微码只读存储器,将SIMD指令的层级,以及微码只读存储器106存储支持的复杂指令的层级作为SIMD计算单元1081可以处理的指令的层级。

[0067] S302、编译器50根据指令的层级生成指令。

[0068] 根据S201中得到的SIMD计算单元1081可以处理的指令的层级的最高值,编译器50可以获取需要向量化的数据处理的任务的源代码,并将该任务编译为层级不高于该最高值的指令,为了描述方便,在之后的表述中,将指令的层级的最高值称为第一阈值。存在以下两种场景:

[0069] 场景一:任务的源代码中直接调用该数据处理的操作在向量化函数库中的函数。

[0070] 当数据处理的操作的层级不高于第一阈值时,编译器50将该函数编译为对应层级的指令。当数据处理的操作的层级高于第一阈值时,编译器50会根据S201步骤中确定的该函数与其他函数之间的调用关系,按照层级从高到低的顺序,逐层编译较低层级操作的函数,直至较低层级的函数对应操作的层级不高于第一阈值,最后将该函数编译为多个层级为第一阈值的指令。

[0071] 以图5中操作的分层方式为例,假设第一阈值为基础计算层,当数据处理为多项式近似时,可以在数据处理的代码中直接调用 $v_polyapprox(x)$ 函数,然后由编译器直接生成多项式近似指令,该多项式近似指令的层级为基础计算层。当数据处理为 $\ln(x)$ 时,可以在数据处理的代码中直接调用 $v_ln(x)$ 函数,编译器根据 $v_ln(x)$ 中的调用关系,编译 $v_mulreduct(x)$ 、 $v_approx(x)$ 和 $v_construct(x)$ 函数、再编译 $v_mulreduct(x)$ 调用的 $v_exindex(x)$ 和 $v_ex(x)$ 函数,以及 $v_approx(x)$ 函数调用的 $v_polyapprox(x)$ 函数,以及 $v_construct(x)$ 函数调用的 $op_fadd(x)$ 函数和 $op_fmul(x)$ 函数。由于 $v_exindex(x)$ 、 $v_ex(x)$ 、 $v_polyapprox(x)$ 、 $op_fadd(x)$ 函数和 $op_fmul(x)$ 函数对应的操作的等级不高于第一阈值,编译器停止编译,最终生成抽取指数位指令、抽取位数指令、多项式近似指令、浮点

数加法指令和浮点数乘法指令。

[0072] 场景二:任务的源代码中未调用该数据处理的操作在向量化函数库中的函数。

[0073] 可以使用编译器、并行编程模型(例如OpenMP)等优化工具,对代码进行扫描,识别出需要进行向量化的数据处理的代码区块,并对比向量化函数库,由编译器50将该数据处理的操作在向量化函数库中的函数按照场景一的方法编译为向量指令。然后,编译器50将识别出的代码区块编译为第一跳转指令,第一跳转指令指向向量指令中优先被执行的指令的地址。则当第一跳转指令执行时,可以通过第一跳转指令跳转到向量指令中优先被执行的指令,完成向量化的数据处理。例如,假设第一阈值为基础计算层,当原代码中调用 $v_ln(x)$ 函数执行数据处理 $ln(x)$ 时,可以使用GNU C语言编译器,通过`-free-vectorize`编译选项激活自动向量化功能,则编译器50自动识别出需要向量化的数据处理,将向量化函数库中的 $v_ln(x)$ 按照方式一的方法编译为抽取指数位指令、抽取位数指令、多项式近似指令、浮点数加法指令和浮点数乘法指令。然后编译器将源代码中的 $ln(x)$ 编译为第一跳转指令`br addr`,其中,`addr`为抽取指数位指令、抽取位数指令、多项式近似指令、浮点数加法指令和浮点数乘法指令中优先执行的指令的地址,即抽取指数位指令的地址。通过优化工具可以完成原代码中非向量化数据处理的自动向量化替换,无需进行代码重构即可使实现更大比例的向量化加速,适用于经过了大量时间开发和完善的高性能计算的应用。

[0074] 可选地,在场景二的情况下,可以在向量化函数库中添加批量数据处理函数,用于对输入数据进行预处理。批量数据处理函数可以接收数据处理中待计算的输入数据的地址和长度,并将大块的输入数据拆分为多个向量,通过调用数据处理的操作在向量化函数库中的函数将输入数据以向量为单位传递给该函数。另外,批量数据处理函数还可以接收数据处理的操作在向量化函数库中的函数的计算结果,并将多个输入数据的向量的计算结果汇和为最终的计算结果。在编译阶段,编译器50将批量数据处理函数编译为批量数据处理指令,批量数据处理函数中调用的函数被编译为向量指令,批量数据处理指令中包括第二跳转指令,用于指向向量指令的地址。编译器50将第一跳转指令指向批量数据处理指令的地址。则当指令执行时,可以通过第一跳转指令跳转执行批量数据处理指令中优先被执行的指令,后继续通过第二跳转指令跳转执行向量指令。

[0075] S303、指令译码器105对指令进行译码,并生成SIMD控制信号。

[0076] S302步骤中编译器50生成的指令存储在存储器20中,指令被执行时,根据程序计数器103中的地址,指令依次送入存储器数据寄存器102以及指令寄存器104中,最后由105指令译码器105从指令寄存器104中获取指令。

[0077] 当指令的层级等于或低于S301步骤中得到的SIMD指令的层级时,指令译码器105独立译码后通过操作控制器107向SIMD计算单元1081发出SIMD控制信号。当指令的层级高于S201步骤中得到的SIMD指令的层级时,指令译码器105从微码只读存储器中查询并得到指令分解后的SIMD指令,将SIMD指令译码后通过操作控制器107向SIMD计算单元1081发出SIMD控制信号。

[0078] 当指令为普通指令时,指令译码器105译码后通过操作控制器107向计算单元108的其余计算单元或者其他部件发出控制信号。

[0079] S304、SIMD计算单元1081接收到SIMD控制信号后执行计算并得到计算结果。

[0080] 通过本申请提出的一种数据处理的方法,用户可以根据数据处理的需求以及SIMD

计算单元的能力划分软件执行和硬件执行的分界线。既可以使用编译器将数据处理的任务编译为SIMD计算单元可以执行的指令,减少硬件的复杂度,还可以使用定制复杂SIMD计算单元以及定制微码只读存储器实现复杂指令分解的方式执行编译器生成的指令,提高计算速度。

[0081] 下面以四个具体计算的示例进一步阐述本申请的数据处理的执行的方法。在示例一中,数据处理任务为计算 $\ln(x)$ 的结果,采用图5的操作分层的方法,假设处理器的SIMD计算单元1081支持基础计算层的指令,微码只读存储器支持计算步骤层到基础计算层以及计算步骤层到现有指令层的解码,则根据S301和S302,编译器将 $\ln(x)$ 任务的源代码编译为第一跳转指令,同时编译器将向量化函数库中的 $\ln(x)$ 函数编译为计算步骤层的指令,即乘法缩减指令、近似指令和范围重构指令,第一跳转指令指向乘法缩减指令的地址。图6是本申请实施例提供的一种实现 $\ln(x)$ 的数据处理的方法的流程示意图,如图6所示,示例一的具体方法如下:

[0082] S601、程序计数器103存储第一跳转指令在存储器20中的地址。

[0083] S602、参考S303的描述,,指令译码器105对第一跳转指令进行译码,并生成SIMD控制信号。

[0084] 由于第一跳转指令为普通指令,指令译码器105译码后通过操作控制器107向程序计数器103发出跳转控制信号。

[0085] S603、程序计数器103中存储的地址更新为第一跳转指令指向的地址,即乘法缩减指令在存储器20中的地址。

[0086] S604、参考S303的描述,指令译码器105对乘法缩减指令进行译码。

[0087] 指令译码器105查询微码只读存储器106解码后生成抽取指数位指令和抽取尾数位指令,经过译码后通过操作控制器107向SIMD计算单元1081发出相应的SIMD控制信号。

[0088] S605、参考S304的描述,,SIMD计算单元1081接收到SIMD控制信号后执行计算并得到乘法缩减指令的计算结果。

[0089] S606、程序计数器103中存储的地址更新为近似指令的地址。

[0090] S607、参考S303的描述,,指令译码器105对近似指令进行译码。

[0091] 指令译码器105查询微码只读存储器106解码后生成多项式近似指令,经过译码后通过操作控制器107向SIMD计算单元1081发出相应的SIMD控制信号。

[0092] S608、参考S304的描述,,SIMD计算单元1081接收到SIMD控制信号后执行计算并得到近似指令的计算结果。

[0093] S609、程序计数器103中存储的地址更新为范围重构指令的地址。

[0094] S610、参考S303的描述,,指令译码器105对范围重构指令进行译码。

[0095] 指令译码器105查询微码只读存储器106解码后生成浮点数加法指令和浮点数乘法指令,经过译码后通过操作控制器107向SIMD计算单元1081发出相应的SIMD控制信号。

[0096] S611、参考S304的描述,,SIMD计算单元1081接收到SIMD控制信号后执行计算并得到范围重构指令计算结果。由于乘法缩减指令、近似指令和范围重构指令依次执行完毕,最后范围重构指令的计算结果即为 $\ln(x)$ 的计算结果。

[0097] 通过上述方法,可以扩展现有的SIMD指令,使用硬件电路直接实现基础操作层的指令,提高计算速度,同时,利用微码只读存储器将微指令组合为计算步骤层的复杂指令,

进一步减少取指令所消耗的计算时间,提高计算速度。

[0098] 示例二是本申请实施例提供的另一种实现 $\ln(x)$ 的数据处理的方法,与图6中的示例一不同的是,在实例二中,假设处理器的SIMD计算单元1081仅支持基础计算层的指令,微码只读存储器106仅支持基础计算层的指令到现有指令层的解码,则根据S201和S202,编译器将 $\ln(x)$ 任务的源代码编译为第一跳转指令,同时编译器将向量化函数库中的 $\ln(x)$ 函数编译为基础计算层的指令,即抽取指数位指令、抽取位数指令、多项式近似指令、浮点数加法指令和浮点数乘法指令,第一跳转指令指向抽取指数位指令的地址。与图6的数据处理的方法类似,不同的是,在执行跳转指令之后,程序计数器103依次存储抽取指数位指令、抽取位数指令、多项式近似指令、浮点数加法指令和浮点数乘法指令的地址;根据程序计数器103存储的每一个指令地址,指令译码器105也依次直接将抽取指数位指令、抽取位数指令、多项式近似指令、浮点数加法指令和浮点数乘法指令译码并生成相应地SIMD控制信号,分别由SIMD计算单元1801执行,得到 $\ln(x)$ 的计算结果。

[0099] 通过上述方法,可以扩展现有的SIMD指令,使用硬件电路直接实现基础操作层的指令,提高计算速度。

[0100] 示例三是本申请实施例提供的另一种实现 $\ln(x)$ 的数据处理的方法,与图6中的示例一不同的是,在实例三中,假设处理器的SIMD计算单元1081仅支持现有指令层的指令,即现有SIMD指令,微码只读存储器支持计算步骤层到现有指令层的解码,则根据S201和S202,编译器将 $\ln(x)$ 任务的源代码编译为第一跳转指令,同时编译器将向量化函数库中的 $\ln(x)$ 函数编译为计算步骤层的指令,即乘法缩减指令、近似指令和范围重构指令,第一跳转指令指向乘法缩减指令的地址。与图6的数据处理的方法类似,不同的是,在执行跳转指令之后,根据程序计数器103存储的每一个指令地址,指令译码器105协同微码只读存储器106依次将计算步骤层的指令分解为现有指令层的指令,具体为:指令译码器105协同微码只读存储器106将乘法缩减指令分解为与指令和移位指令并译码生成相应的SIMD控制信号,将近似指令分解为FMA指令并译码生成相应的SIMD控制信号,以及将范围重构指令分解为浮点数加法指令和浮点数乘法指令并译码生成相应的SIMD控制信号。最后现有指令层的指令分别由SIMD计算单元1801执行,得到 $\ln(x)$ 的计算结果。

[0101] 通过上述方法,可以扩展现有的SIMD指令,使用硬件电路直接实现基础操作层的指令,提高计算速度。

[0102] 示例四是本申请实施例提供的另一种实现 $\ln(x)$ 的数据处理的方法,与图6中的示例一不同的是,在实例四中,假设处理器的SIMD计算单元1081仅支持现有指令层的指令,微码只读存储器不支持解码,则根据S201和S202,编译器将 $\ln(x)$ 任务的源代码编译为第一跳转指令,同时编译器将向量化函数库中的 $\ln(x)$ 函数编译为现有指令层的指令,即与指令、移位指令、FMA指令、浮点数加法指令以及浮点数乘法指令,第一跳转指令指向与指令的地址。与图6的数据处理的方法类似,不同的是,在执行跳转指令之后,程序计数器103依次存储抽取与指令、移位指令、FMA指令、浮点数加法指令以及浮点数乘法指令的地址;根据程序计数器103存储的每一个指令地址,指令译码器105也依次直接将与指令、移位指令、位指令、FMA指令、浮点数加法指令以及浮点数乘法指令译码并生成相应地SIMD控制信号,分别由SIMD计算单元1801执行,得到 $\ln(x)$ 的计算结果。

[0103] 通过上述方法,在处理器的硬件上并没有针对数据处理的优化,所有层级的操作

均由编译器实现。分层操作的函数设计不仅保证了每一层的操作的完整性和扩展性,用户可以分别控制每一层操作的具体流程以完成对数据处理的细节上的优化

[0104] 值得说明的是,对于上述方法实施例,为了简单描述,故将其都表述为一系列的动作组合,但是本领域技术人员应该知悉,本申请并不受所描述的动作顺序的限制。

[0105] 本领域的技术人员根据以上描述的内容,能够想到的其他合理的步骤组合,也属于本申请的保护范围内。其次,本领域技术人员也应该熟悉,说明书中所描述的实施例均属于优选实施例,所涉及的动作并不一定是本申请所必须的。

[0106] 上文中结合图3至图6,详细描述了本申请实施例所提供的数据处理的方法,下面将结合图7,进一步介绍本申请实施例所提供的数据处理的装置。

[0107] 图7为本申请实施例提供的一种数据处理的装置700的示意图,包括第一存储单元701、编译单元702、译码单元703和执行单元704。装置700用于实现图1或图2所示的处理器

的功能。

[0108] 其中,第一存储单元701,用于存储第一指令和至少两个子指令之间的对应关系。在产品实际实现中,第一存储单元701可以是图2所示的微码只读存储器106,也可以是其他存储器,本实施例不对第一存储单元701的形态进行限定。

[0109] 编译单元702,用于获取数据处理的任务并将任务编译为第一指令,具体实现可参考图3中S301和S302的步骤,这里不再赘述。在产品实际实现中,编译单元702可以是图1所示的编译器50,也可以是其他编译器,本实施例不对编译单元702的形态进行限定。

[0110] 译码单元703,用于获取第一指令,并根据对应关系,将所述第一指令分解为至少两个子指令,还用于将子指令发送给SIMD计算单元。译码单元703的功能的具体实现可参考图3中S303的步骤,这里不再赘述。在产品实际实现中,译码单元703可以是图2所示的指令译码器105,也可以是其他指令译码器,本实施例不对译码单元703的形态进行限定。

[0111] 执行单元704,用于执行子指令以获得数据处理的处理结果,具体实现可参考图3中S304的步骤,这里不再赘述。在产品实际实现中,执行单元704可以是图2所示的SIMD计算单元1081,也可以是其他SIMD计算单元,本实施例不对执行单元704的形态进行限定。

[0112] 可选地,第一指令可以为第一层级的指令,子指令为第二层级的指令,其中,第一层级的指令的操作的复杂程度高于第二层级的指令的操作的复杂程度,SIMD计算单元支持执行所述第二层级的指令。

[0113] 可选地,至少两个子指令包括Neon指令集和可伸缩适量扩展指令集中任意一个指令。则指令译码器支持将负责指令解码为现有指令集的指令,可以对现有指令集进行扩展。其中,两个子指令可以均为Neon指令集中至少两个指令,也可以为SVE指令集中至少两个指令,还可以分别为至少一个Neon指令集的指令和至少一个SVE指令集中指令。

[0114] 可选地,至少两个子指令还包括由Neon指令集和可伸缩适量扩展指令集中至少两个指令组成的指令。其中,两个子指令可以均为Neon指令集中至少两个指令组成的指令,也可以为可伸缩适量扩展指令集中至少两个指令组成的指令,还可以分别为Neon指令集和可伸缩适量扩展指令集中至少两个指令组成的指令。

[0115] 可选地,数据处理的装置还包括第二存储单元705,用于存储向量化函数库,向量化函数库包括Neon指令集和SVE指令集中任意一个指令的操作所对应的函数。则编译单元702还用于在数据处理的

际实现中,第二存储单元705可以是图1或图2所示的存储器20,也可以是其他存储器,本实施例不对第二存储单元705的形态进行限定。

[0116] 可选地,向量化函数库还包括由Neon指令集和SVE指令集中至少两个指令组成的指令的操作所对应的函数。

[0117] 可选地,当任务的代码所包含的函数属于向量化函数库中的函数时,第一指令包括向量指令,编译单元702还用于将任务编译为向量指令。

[0118] 可选地,当任务的代码所包含的函数不属于向量化函数库中的函数时,第一指令包括第一跳转指令和向量指令,编译单元702还用于将任务编译为第一跳转指令,并且将任务的操作在向量化函数库中的函数编译为向量指令,跳转指令指向向量指令的地址。

[0119] 可选地,第一指令还包括批量数据处理指令,第一跳转指令指向批量数据处理指令的地址,编译单元702还用于将批量数据处理函数编译为批量数据处理指令,其中,批量数据处理指令包括第二跳转指令,第二跳转指令指向向量指令的地址。

[0120] 可选地,数据处理的任务包括初等函数的计算或者非线性函数的计算。

[0121] 综上所述,本申请实施例提供的数据处理装置700,可以基于已有的SIMD指令集,通过编译单元和译码单元实现复杂数学计算的数据向量化加速处理,从而提高装置的计算性能。

[0122] 上述实施例,可以全部或部分地通过软件、硬件、固件或其他任意组合来实现。当使用软件实现时,上述实施例可以全部或部分地以计算机程序产品的形式实现。所述计算机程序产品包括一个或多个计算机指令。在计算机上加载或执行所述计算机程序指令时,全部或部分地产生按照本申请实施例所述的流程或功能。所述计算机可以为通用计算机、专用计算机、计算机网络、或者其他可编程装置。所述计算机指令可以存储在计算机可读存储介质中,或者从一个计算机可读存储介质向另一个计算机可读存储介质传输,例如,所述计算机指令可以从一个网站站点、计算机、服务器或数据中心通过有线(例如同轴电缆、光纤、数字用户线(digital subscriber line,DSL))或无线(例如红外、无线、微波等)方式向另一个网站站点、计算机、服务器或数据中心进行传输。所述计算机可读存储介质可以是计算机能够存取的任何可用介质或者是包含一个或多个可用介质集合的服务器、数据中心等数据存储设备。所述可用介质可以是磁性介质(例如,软盘、硬盘、磁带)、光介质(例如,DVD)、或者半导体介质。半导体介质可以是固态硬盘(solid state drive,SSD)。

[0123] 以上所述,仅为本申请的具体实施方式。熟悉本技术领域的技术人员根据本申请提供的具体实施方式,可想到变化或替换,都应涵盖在本申请的保护范围之内。

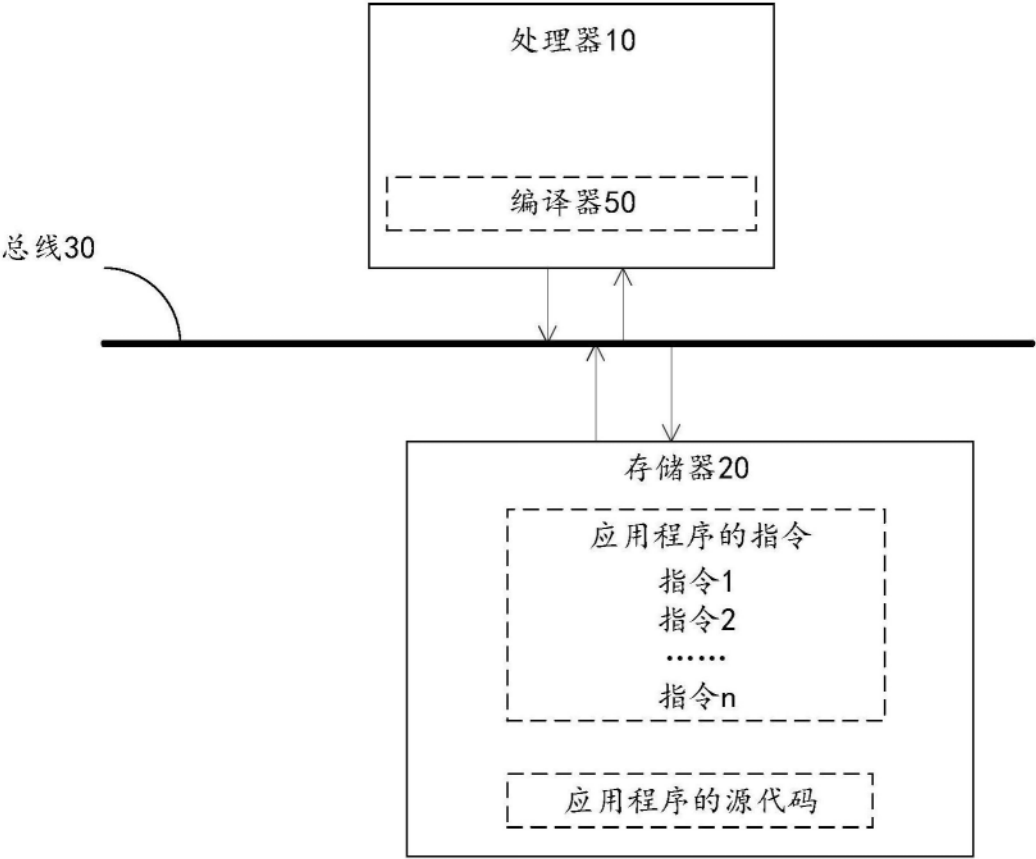


图1

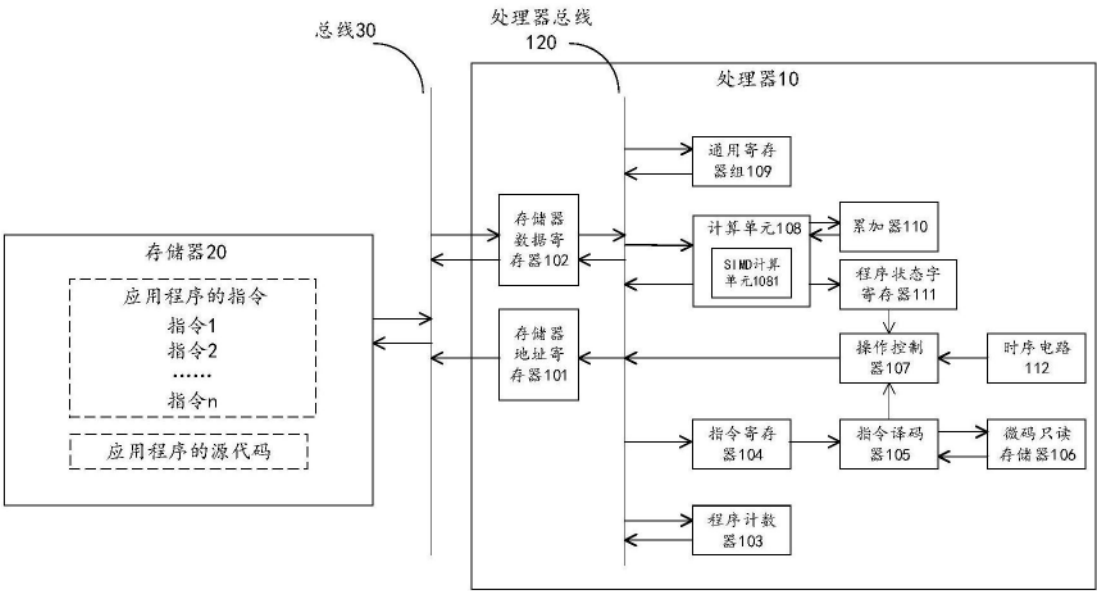


图2

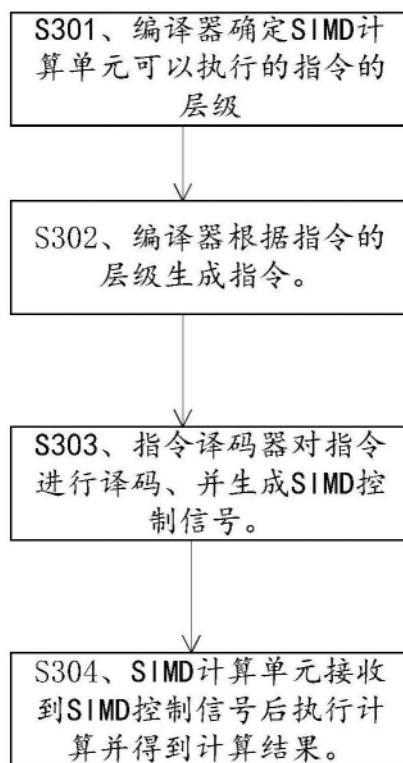


图3

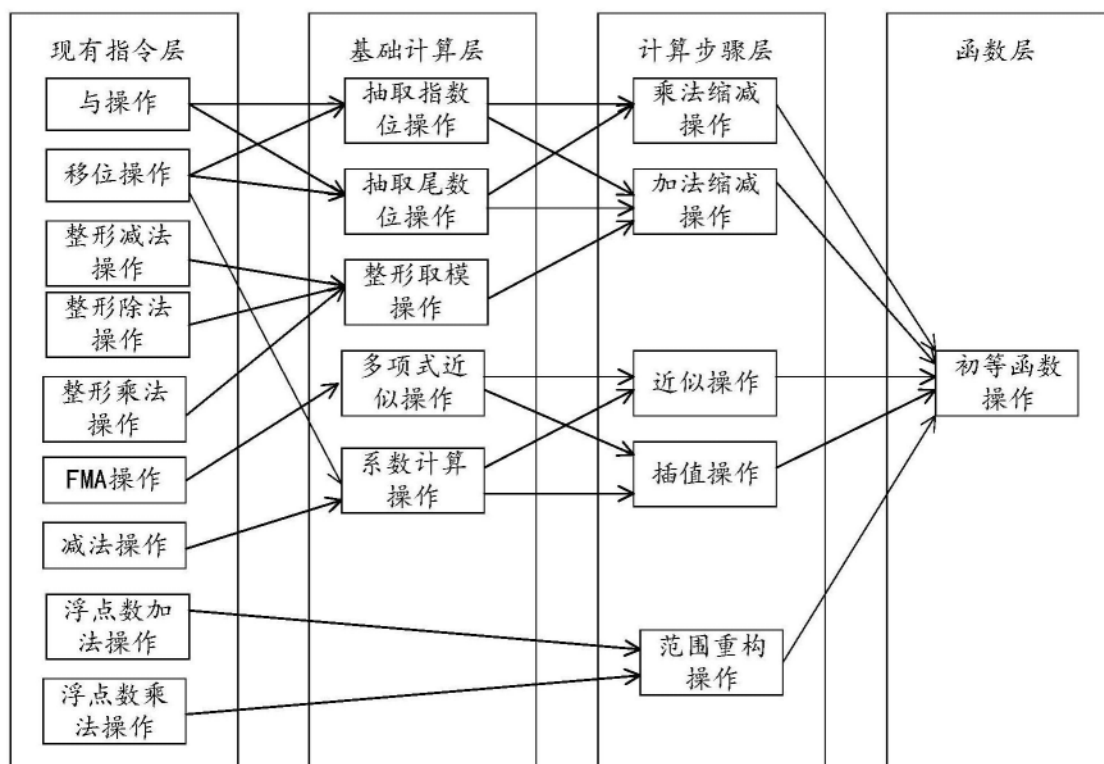


图4

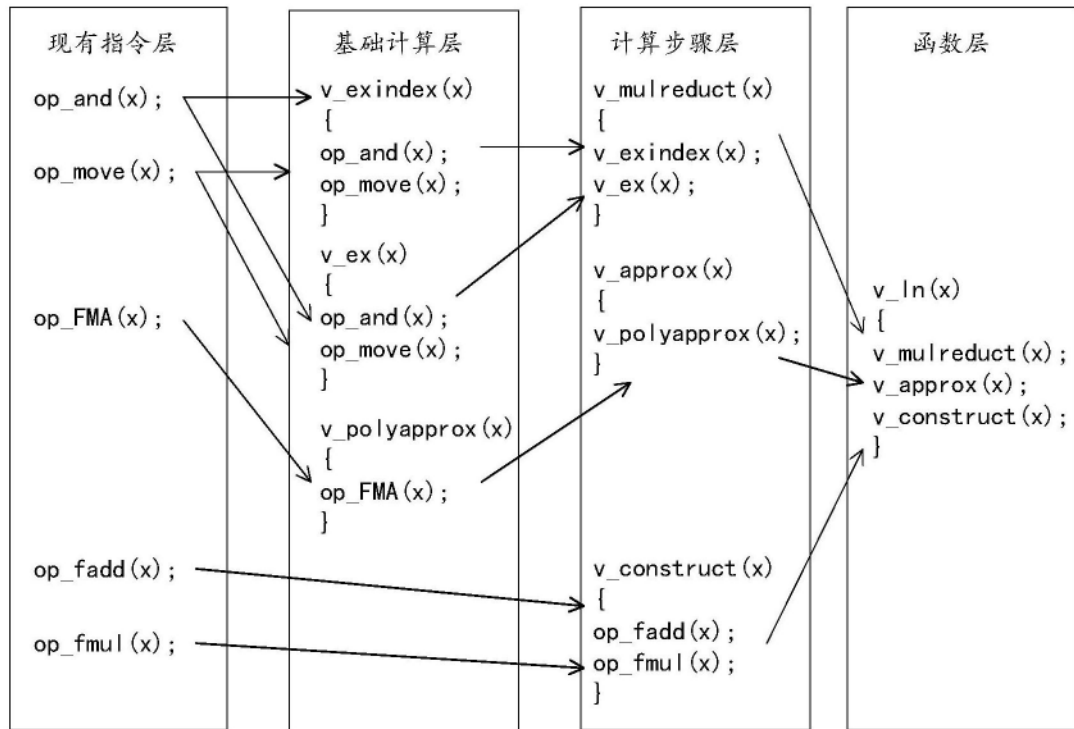


图5

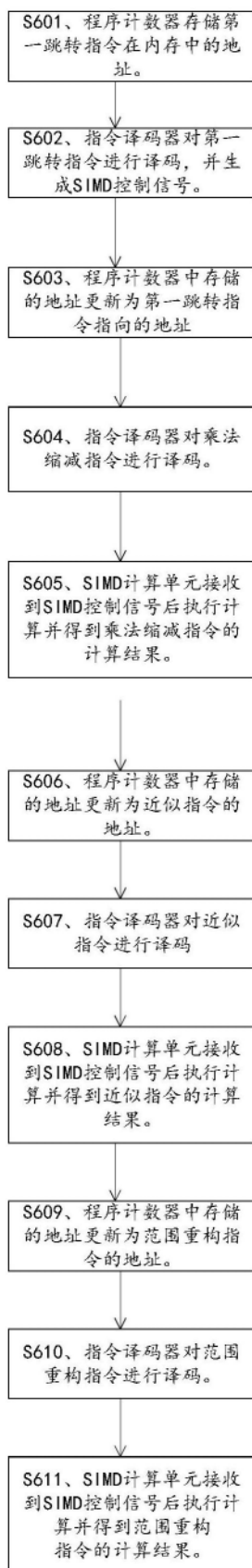


图6

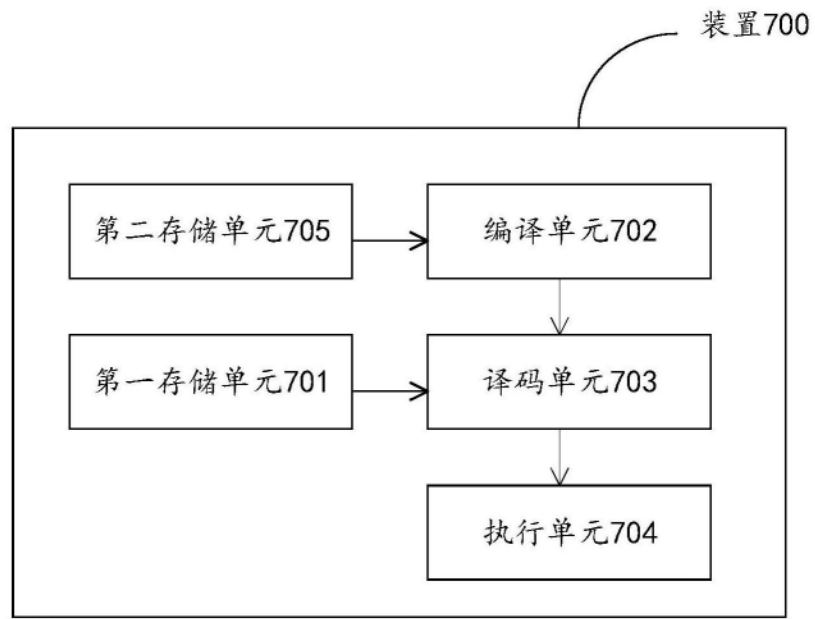


图7