Atitit 性能提升 流水线模式 流水线深度

[1.1. 每一个具体步骤就是一条业务指令 1](#_Toc21521)

[2. 流水线的级数（深度 2](#_Toc9464)

[2.1. 如何确定合适的级数 深度 2](#_Toc14398)

[3. 流水线深度 3级流水 5级别 15级别 2](#_Toc10282)

[4. 流水线深度过深的问题 3](#_Toc26650)

[4.1. 复杂度也不能一直细化划 3](#_Toc4702)

[4.2. 分支预测流水线冲刷（Pipeline Flush） 3](#_Toc18890)

[5. 流水线其他概念 4](#_Toc25485)

[5.1. ”流水线停顿(pipeline stall)” 4](#_Toc28196)

[5.2. 两条指令之间的关系可以分为：数据相关和控制相关，数据相关分为； 4](#_Toc16109)

[5.3. 流水线互锁实现 5](#_Toc21076)

[5.4. 动态分支预测 5](#_Toc16520)

## 每一个具体步骤就是一条业务指令

流水线处理器性能的主要障碍是指令相关引起的流水线停顿，其中控制相关引起的分支开销是最重要的部分。

**动态分支预测**能减少这一部分的开销，预测成功时，流水线没有停顿。在检测到预测失败时，必须清空流水线。

减少从取指段到完成分支指令段的距离也可以减少分支指令引起的开销。

在流水线处理器设计中，不仅仅数据通道被流水化了，连控制通道也被流水化了，传统的数据通道和控制通道被合并到同一条流水线。

# ****流水线的级****数（深度

采用流水线技术后，并没有加速单条指令的执行，每条指令的操作步骤一个也不能少，只是多条指令的不同操作步骤同时执行，因而从总体上看加快了指令流速度，缩短了程序执行时间。

为了进一步满足普通流水线设计所不能适应的更高时钟频率的要求，高档次处理器中的****流水线的深度****（级数）在逐代增多。当流水线深度在5~6级以上时，通常称为****超流水线结构****（SuperPipeline）。显然，流水线级数越多，每级所花的时间越短，时钟

## 如何确定合适的级数 深度

产上的装配流水线。在CPU中由5—6个不同功能的电路单元组成一条指令处理流水线，然后将一条X86指令分成5—6步后再由这些电路单元分别执行，这样就能实现在一个CPU时钟周期完成一条指令，因此提高CPU的运算速度。经典奔腾每条整数流水线都分为四级流水，即取指令、译码、执行、写回结果，浮点流水又分为八级流水。

面6 bit标识指令的操作码，即这条指令是干什么的，后面26 bit是这个跳转指令的跳转地址。指令是一串0、1序列，译码单元要将这条指令包含的有用数据解出来，如果是跳转指令，就要知道它的地址，如果是运算指令，就要知道的它的源操作数、目的操作数在哪，这样处理器才能进行后续的运算。

# 流水线深度 3级流水 5级别 15级别

发展到 Cortex-A(32-bit)/ARMv7-A/Cortex-A8 已经支持多达 13 级流水线，后续支持更深度的甚至乱序（out-of-order）的超标量（superscalar）流水线结构。  
以下列举了 List of ARM microarchitectures 中由 ARM 公司设计的 ARM cores（family/architecture/core） 对应的流水线级数发展史

****4.3 MIPS经典五级流水线****

****（1）MIPS的五级流水线****

MIPS体系架构依据流水线结构设计。只要CPU从缓存中获取数据，那么执行每条MIPS指令就被分成五个流水阶段，并且每个阶段占用固定的时间，通常是只耗费一个处理器时钟周期。RD/WB操作只占用半个时钟周期，故MIPS五段流水线只占用四个时钟周期。

MIPS处理器在设计时，将处理器的执行阶段划分为以下五个阶段:

<1> ****IF****:Instruction Fetch，****取指****。从指令缓存（I-Cache）中获取下一条指令。

<2> ****ID****（RD）：InstructionDecode（Read Register），****译码（读寄存器）****。翻译指令，识别操作码和操作数，从寄存器堆中读取数据到ALU输入寄存器。

<3> ****EX****（ALU）：Execute，****执行****（算术/逻辑运算）。在一个时钟周期内，完成算术或逻辑操作。注意，浮点算术运算和整数乘除运算不能在一个时钟周期内完成。

<4> ****MEM****：Memory Access，****内存数据读或者写****。在该阶段，指令可以从数据缓存（D-Cache）中读/写内存变量。平均来说，大约四分之三的指令在这一阶段没有执行任何操作，为每条指令分配这个阶段是为了保证同一时刻不会有两条指令都访问数据缓存。

<5> ****WB****：Write Back，****写回****。操作完成后，将计算结果从ALU输出寄存器写回到通用寄存器中。

****对于运算指令****，在MEM阶段空闲。****对于load指令****，在EX阶段计算要访问的地址，在MEM阶段从内存中将数据读入到MEMregister（MEM和WB之间的流水线寄存器）中，在WB阶段，将MEM register的数据写回到Register File中。****对于store指令****，在EX阶段计算要访问的地址，在MEM阶段将寄存器中的数据写回到存储器中。

大，对流水线性能有一定影响现在很多cpu都是将超标量和超级流水线技术一起使用，例如pentiumIV，流水线达到20级，频率最快已经超过3GHZ。教科书上用于教学的经典MIPS只有5级流水

# 流水线深度过深的问题

以最知名的ARM Cortex-A系列处理器IP为例，Cortex-A7主打的低功耗前提下的能效比，其流水线级数为8级；而Cortex-A15主打高性能，其流水线深度为15级。  
   
当然流水线越来越深也需有其限度，曾有某些商业处理器产品一味地追求极端流水线深度（达到几十级）反而遭遇失败的例子。目前最新的Intel处理器和ARM 高性能Cortex-A系列处理器的流水线深度都在十几级的范围左右。

## 复杂度也不能一直细化划

流水线并不能被无限制的增加下去，因为时钟的限制(setup time/clock uncertainty)，流水线的划分有一个物理局限。

复杂度也不能一直细化划分下去那么远样划分成本很高

## 分支预测流水线冲刷（Pipeline Flush）

较深的处理器流水线还有一个问题，由于在流水线的取指令阶段无法得知条件跳转的结果是跳还是不跳，因此只能进行预测，而到了流水线的末端才能够通过实际的运算得知该分支是真的该跳还是不该跳，如果发现真实的结果（譬如该跳）与之前预测的结果（譬如预测为不跳）不相符，则意味着预测失败，需要将所有预取的错误指令流全部丢弃掉，而重新取正确的指令流，这个过程叫做流水线冲刷（Pipeline Flush），虽然可以使用分支预测器来保证前期的分支预测尽可能的准确，但是也无法做到万无一失。那么，流水线的深度越深，则意味着已经预取了很多的错误指令流，需要将其全部抛弃然后重启，不仅白白的浪费了功耗，还造成了性能的损失。流水线越深则意味着浪费和损失越严重，流水线越浅则浪费和损失越少。这是流水线加深的另一个主要的负面意义。

# 流水线其他概念

## ****”流水线停顿(pipeline stall)”****

## 两条指令之间的关系可以分为：数据相关和控制相关，数据相关分为；

数据相关即可以发生在存储器中也可以发生在寄存器中。

1)：先写后读，(RAW，read after write)，后一指令的操作数是该指令的结果。该指令的结果被后指令使用。真相关。

2)：先读后写，(WAR，write after read)，反相关，防止后一指令的结果对上一指令的操作数产生影响。

3)：写后写，(WAW，write after write)，两条指令的顺序不能改变。输出相关。

4)：读后读，(RAR，read after read)，无影响。

## 流水线互锁实现

2)：流水线互锁实现。检测出所有的流水线相关，并保证所有的相关得到满足。包括停止流水线的某一段，以及，数据在定向路径中的传输控制。

RAW相关的检测是通过比较器，连续比较连续两条指令的寄存器标识来实现的。如果检测到数据的相关性，则使能相关的定向路径。

如果没有相关的定向路径，则停止流水线的某段。

## ****动态分支预测****

流水线处理器性能的主要障碍是指令相关引起的流水线停顿，其中控制相关引起的分支开销是最重要的部分。

**动态分支预测**能减少这一部分的开销，预测成功时，流水线没有停顿。在检测到预测失败时，必须清空流水线。

减少从取指段到完成分支指令段的距离也可以减少分支指令引起的开销。

在流水线处理器设计中，不仅仅数据通道被流水化了，连控制通道也被流水化了，传统的数据通道和控制通道被合并到同一条流水线。

作者简介

艾提拉 艾龙 attilax，法名 st attilax akbar rinpoche 圣阿提拉科斯阿克巴仁波切

头街软件技术大师 uke组织创始人

学术成就，完善的20大知识体系，拥有uke学院硕士博士学位

从事软件互联网行业技术背景十二年，csdn排名TOP57

长年从事软件互联网技术与管理，预计出版多本心得分享书籍

擅长技术与管理与文化 致力于标准化事业

Qq 1466519819 小号112237553

微信attilax 小号attilax201708

作者:: 绰号与头街 :老哇的爪子claw of Eagle 偶像破坏者Iconoclast image-smasher 神的使者（Messenger of God）及守望者（Watch Man

捕鸟王"Bird Catcher kok 虔诚者Pious 宗教信仰捍卫者 Defender Of the Faith. 卡拉卡拉红斗篷 Caracalla red cloak KOA万兽之王 纵火者

头街来源：神的使者（Messenger of God）及守望者（Watch Man来源于圣经

老哇的爪子claw of Eagle来源于印加帝国

KOA万兽之王 来源于婆罗门大神森林中修炼

简称：： st Emir Attilax Akbar 圣 埃米尔 阿提拉克斯 阿克巴

全名：：st Emir Attilax Akbar bin Mahmud bin attila bin Solomon bin adam Al Rapanui 圣 埃米尔 阿提拉克斯 阿克巴 本 马哈茂德 本 阿提拉 本 所罗门 本亚当 阿尔 拉帕努伊

常用名：艾提拉（艾龙）， EMAIL:1466519819@qq.com

喜欢的绰号 捕鸟王纵火者 老瓦的爪子兔子的耳朵

喜欢的头街 st圣 仁波切 大师 马斯塔 艺术家 博士 诗人 国王

头衔：

|  |  |
| --- | --- |
| uke | Emir Uke部落首席大酋长，ati协会创始人 仁波切马斯塔 埃米尔  uke总部o2o负责人，全球网格化项目创始人，  圣阿提拉克斯国王 |
| 科技领域 | UTSC uke技术标准化委员会委员长 uke 首席cto 软件部门总监 技术部副总监 研发部门总监主管 产品部副经理 项目部副经理 uke科技研究院院长 uke软件培训大师  Ati组织科研研究院创始人 |
| 文艺领域 | ， ，， uke机车协会主任 uke纹身协会  uke交友协会会长 uke捕猎协会会长  Ati文艺协会会长 ati文学协会 |
| 行政领域 | Gchsp总裁 gchsp常委 GsP创始人 |
| 媒体传播领域 | uke出版社编辑总编 宣传布道总策划  Ati传媒总部 |
| 渔猎军事领域 | uke保安部首席大队长  Uke 户外运动协会理事长 度假村首席大村长  Ati打猎协会 |
| 法学 | 法学研究会 制度研究会 |
| 管理领域 | 工商管理学 公共管理与社会服务  ,uke制度检查委员会副会长 |
| 教育领域 | uec学院校长， uecip图像处理机器视觉专业系主任 uke文档检索专业系主任  Uke图像处理与机器视觉学院首席院长  uke终身教育学校副校长  靓号研究院 |
| 经济领域 | uke波利尼西亚区大区连锁负责人 汤加王国区域负责人 uke克尔格伦群岛区连锁负责人，莱恩群岛区连锁负责人，uke布维岛和南乔治亚和南桑威奇群岛大区连锁负责人  Uke软件标准化协会理事长理事长 Uke 数据库与存储标准化协会副会长  直达巴士西北区负责人 直达巴士长沙与西安分部部长  润昌通讯软件事业部总裁 执行长 分部负责人 执行委员会主席  Ati经济研究所 |
| 历史领域 | 历史事业部 ati历史研究院 |
| 社会科学领域 | 社科学院 ati文化部 |
| 自然科学领域 | Uke研究院院长兼首席研究员 科学家  Ati自然科学研究院 |
| 宗教神学领域 | uke宗教与文化融合事务部部长 大师master  uke制度与重大会议委员会委员长 ati宗教事务所 |
| 医学领域 | Uke医院 与医学院方面的创始人 |

转载请注明来源：attilax的专栏 <http://blog.csdn.net/attilax>

<http://www.cnblogs.com/attilax/>

Microblog

<http://weibo.com/u/5941179815> (common attilax)

<https://weibo.com/p/1005055941179815> （attilax201707,bek weibo）

<http://weibo.com/u/5487832265> (tech,for blog auto gene)

知乎空间

https://www.zhihu.com/people/ati-att/activities

Qq 1466519819 小号112237553

微信attilax 小号attilax201708

微博 attilax2016 小号attilax201707

--Atiend v23