

## 第五章 64位微处理器

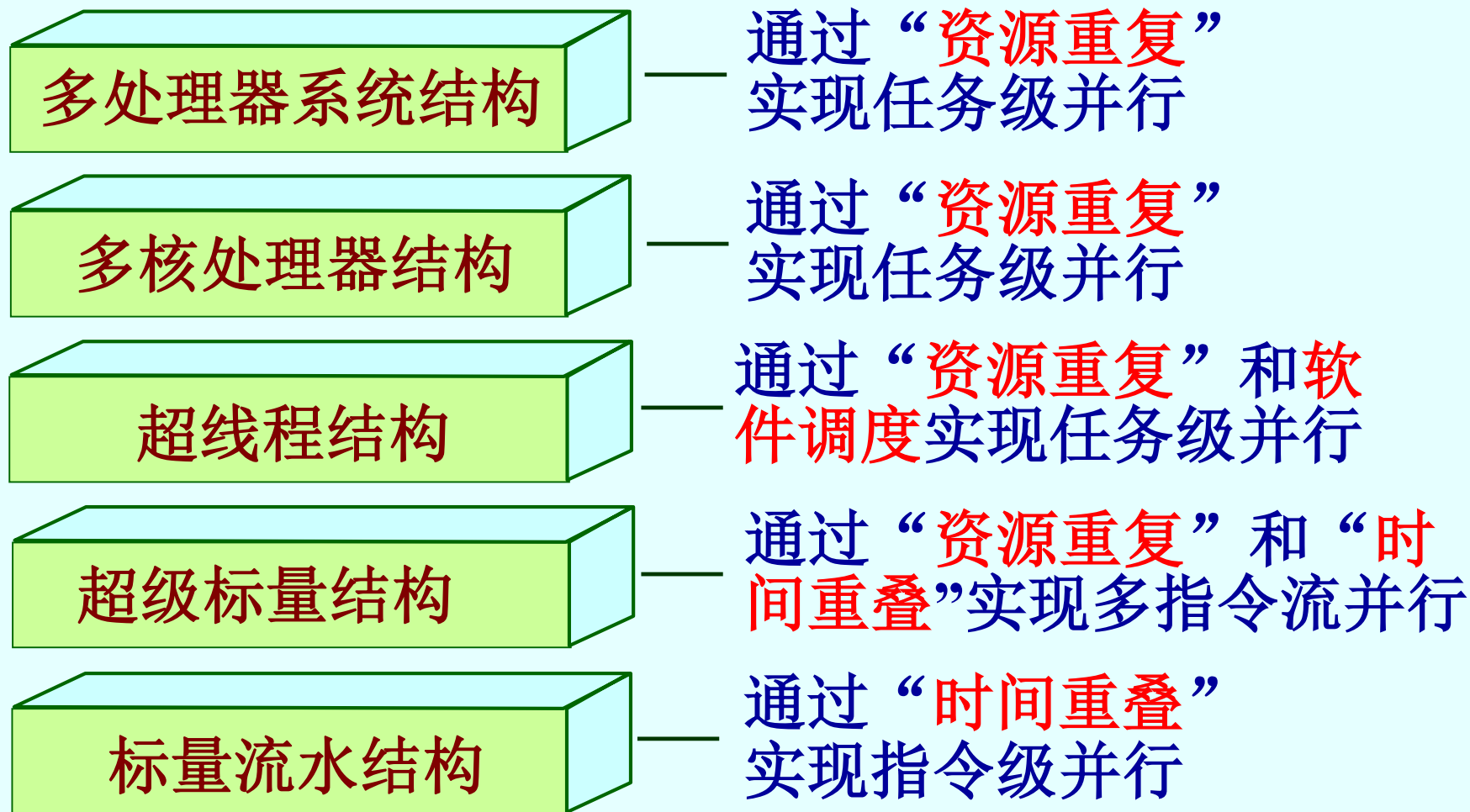
64位处理器的先驱:

**Alpha 21064 和 MIPS R4000**

相继有:

- HP的PA-RISC 8000系列
- Alpha 21×64
- Sun的 SPARC
- IBM的Power G4、G5、G6、G7等
- Intel的 Itanium
- AMD的 Opteron和Athlon 64 (FX系列)

# 回顾“并行性”技术(从结构上提高性能的主要手段):



**超长指令字 VLIW**

## 超长指令字 VLIW

### VLIW 基本概念 (Very Long Instruction Word):

一种实现指令并行的技术。即把多条指令(包含多个不同的操作)连在一起, 构成一条长指令(超长指令), 以增强并行处理能力。

### VLIW实现的基本思路:

将一个长指令字所包含的的不同操作调度(分配)到不同的功能部件。

比如: 将 “ $R1+R2 \rightarrow R3$ ” 和 “ $R4+R5 \rightarrow R6$ ” 这两条指令组合到同一个指令字中(两条指令无寄存器相关)

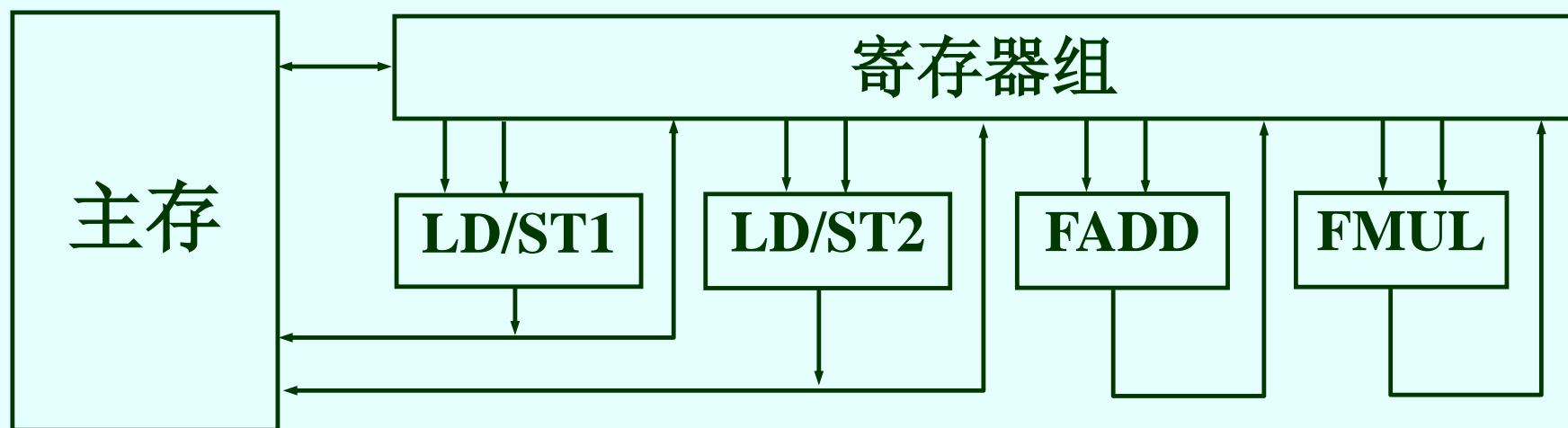
前提条件: 至少有两个可进行加法运算的部件

## VLIW的主要技术特点:

- (1) 超长指令字被分成多个字段, 各字段**独立控制各功能部件**;
- (2) 由**编译**来识别可能出现的数据相关和资源冲突等, **并进行指令乱序(重调度), 硬件控制较为简单**;
- (3) 由**编译器**在编译阶段来完成超长指令字中**多个可并行操作的调度**;
- (4) 并行操作在流水线的**执行阶段**完成;
- (5) 指令字长度(即可并发的操作)与功能部件的数量有关(取决于功能部件的数量)。

例:

假设一个有2个存取部件、1个浮点加法部件、1个浮点乘法部件(乘法需2个时钟周期)的超长指令字机器,所有功能部件由同一时钟驱动,如图所示。



在同一时刻控制每个功能部件的操作字段组成一个超长指令字:

**VLIW的操作字段:**

LD/ST1	LD/ST2	FADD	FMUL
--------	--------	------	------

编译将**串行**的操作序列合并为可并行执行的指令序列。

例：执行以下几个语句：

**C = A + B**

**K = I + J**

**L = M - K**

**Q = C × K**

如果按**串行**操作进行，则所用指令序列是(右表)：

源代码	操作	周期数
<b>C = A + B</b>	<b>Load A</b>	<b>1</b>
	<b>Load B</b>	<b>1</b>
	<b>C = A + B</b>	<b>1</b>
	<b>Store C</b>	<b>1</b>
<b>K = I + J</b>	<b>Load I</b>	<b>1</b>
	<b>Load J</b>	<b>1</b>
	<b>K = I + J</b>	<b>1</b>
	<b>Store K</b>	<b>1</b>
<b>L = M - K</b>	<b>Load M</b>	<b>1</b>
	<b>L = M - K</b>	<b>1</b>
	<b>Store L</b>	<b>1</b>
<b>Q = C × K</b>	<b>Q = C × K</b>	<b>2</b>
	<b>Store Q</b>	<b>1</b>

**13条指令, 共计14个周期**

如果采用**超长指令字**的编译方法,可将原来的13条指令压缩成**6条**长指令,如下表所示。

$C = A + B$   
 $K = I + J$   
 $L = M - K$   
 $Q = C \times K$

周期1	Load A	Load B		
周期2	Load I	Load J	$C = A + B$	
周期3	Load M	Store C	$K = I + J$	
周期4		Store K	$L = M - K$	$Q = C \times K$
周期5		Store L		
周期6	Store Q			

共6个  
周期

注: 乘法运算需要2个周期(即第4和第5周期), 所以“Store Q”操作在第6个周期完成)。

## 一、Itanium 处理器

基于EPIC(Explicitly Parallel nstruction Computing;  
显性并行指令计算)的Itanium体系结构(2001年5月)。

EPIC最基本的特点是从VLIW继承而来: 支持编译器来决定指令执行过程。既不是RISC也不是CISC, 是一种吸收两者特点的体系结构。



## (一) Itanium1的主要特点

乱序执行技术是多年来32位处理器和64位RISC芯片设计的主流;但缺点是(Intel公司认为):

- (1) 要求处理器具有强大功能和性能,使芯片的结构越来越复杂,又妨碍了主频和性能的提高;
- (2) 设计难度越来越大,使芯片设计周期越来越长,难以满足应用发展的需要;
- (3) 处理器运行时,未充分利用编译来提高指令并行性,使软硬件相结合的能力未得到充分发挥。

根据以上认识,基于EPIC技术的Itanium由此产生。

## EPIC体系结构的**基本设计思想**:

1. 利用**编译**调度硬件资源来提高指令并行度  
指令中设置**指明**哪些指令可以并行的**属性字段**。  
**编译过程**中分析源代码中的**并行属性**(如推测执行、减少数据相关等),并将这些信息填写到执行代码中(显示并行的核心思想)。
2. 简化芯片逻辑结构,以提高主频和性能。
3. 提供大量的资源来实现EPIC,包括:
  - ▶ 多处理单元
  - ▶ 大容量高速缓存(三级Cache:  $L_0$   $L_1$   $L_3$ )
  - ▶ 128个64位整数寄存器, 128个82位浮点寄存器
  - ▶ 64个1位预测寄存器, 8个转移寄存器
  - ▶ 128个专门的应用寄存器 等

## 4. 指令集结构

每条指令可以引用2个输入寄存器和1个输出寄存器，提高了单条指令的并发能力并更易编程。

- ▶ 引入64位寻址和新的指令集，也包含IA-32的指令集，所有IA-64处理器都能执行IA-32程序；
- ▶ 指令长度41位；
- ▶ 存取式体系结构；
- ▶ 由编译支持指令调度；比如：  
利用编译来消除某些转移指令，让多个分支能同时执行，即“指令断定”（指令预测的进一步发展）：



## “指令断定” 技术(断定执行)

- (1) 将分支指令断定为true值或false值
- (2) 将为true和false的指令并行执行。当判断结果出来时, 再将断定错误的指令结果删除。
- (3) 提供64个断定寄存器, 每个断定寄存器可保存一位(true或false)。其值由比较指令比较的结果来决定。分支指令使用一对存有相反结果的断定寄存器。

比如：通过设置一条比较指令来产生预测结果。该指令抽象为以下形式：

**pT, pF CMP(crel r2, r3)**

用比较规则crel比较r2和r3。比较结果写入预测寄存器pT, 相反状态写入预测寄存器pF。

**例1: 指令: if (a>b) then c=c+1 else d=d×e+f**

用以下指令组完成：

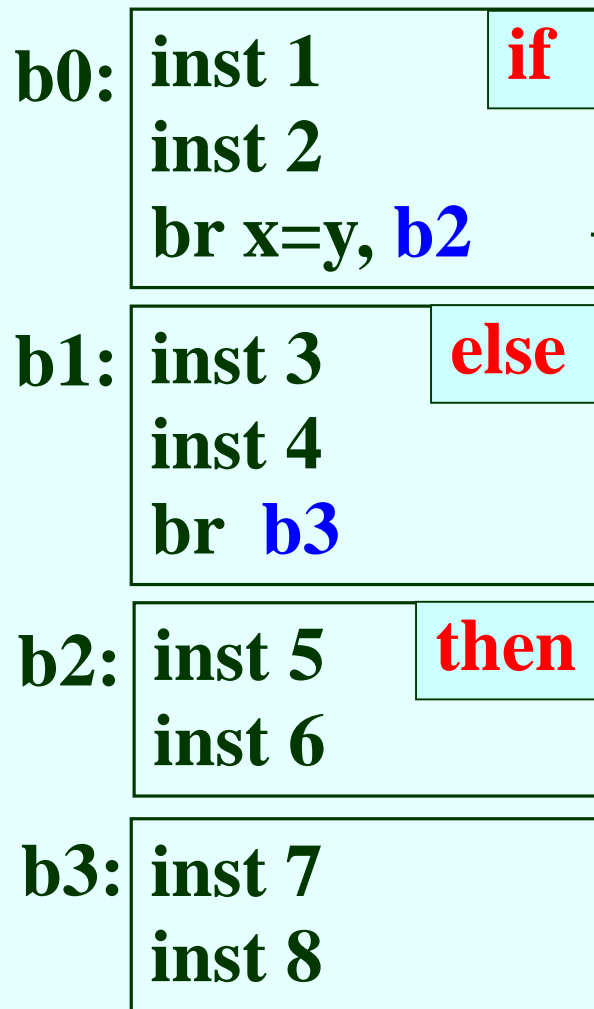
**pT, pF = CMP(a > b)**  
**if (pT) c=c+1**  
**if (pF) d=d×e+f**

或者

**pT, pF = CMP(a > b)**  
**(pT) c=c+1 || (pF) d=d×e+f**

由编译将为 true或false的两条指令并行来消除转移

## 例2: 对程序块的情况



四个基本块

断定

inst 1  
inst 2  
**p1, p2 ← cmp(a=b)**  
(p1) inst 5 || (p2) inst 3  
(p1) inst 6 || (p2) inst 4  
inst 7  
inst 8

## 5. 高并行性

可用包(bundle)的形式(每个包3条指令)发射不相关的、可并行执行的指令(VLIW), 每个时钟周期可发射2个包, 即每时钟周期发射6条指令。

但由于资源限制, 据统计, 每时钟周期发送6条指令的概率为25%, 每周期平均发送3.85条指令。

## 6. 推测执行技术

提前执行指令时, 针对访存不命中高速缓存时, 导致整个流水线阻塞的问题的解决办法(每个周期处理的指令和数据越多, 不命中概率越高, 阻塞越严重)。

推测执行技术分为“**控制推测**”和“**数据推测**”。

### 控制推测

将分支指令之后的取数指令提前若干个周期执行，以消除访存延迟，提高并行度。

由此产生的问题：

取数指令不命中Cache(需访问内存)，更严重的是：还可能产生缺页，需通过异常中断处理，由操作系统启动I/O，产生大量时间损耗。而当分支断定结果产生后，该取数指令还是不需要执行的！

**Itanium采用的解决方式是：**

产生缺页时，不进入异常处理，仅做一个标记，以后需要时，再进行异常处理并通过操作系统启动I/O。



## 数据推测

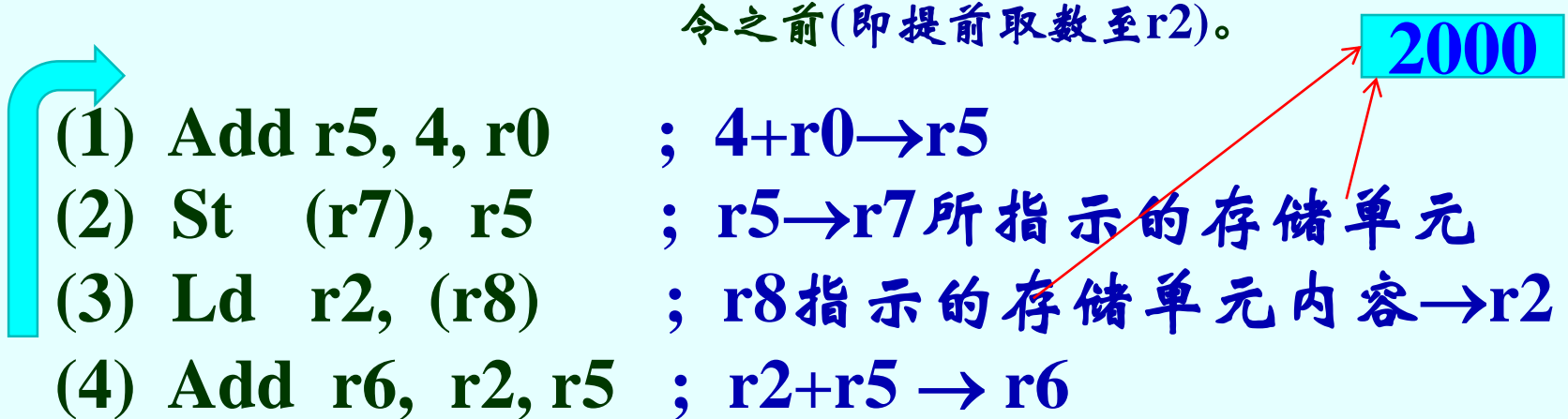
用于解决**提前取数**指令后所产生的**数据相关**。

比如以下情况:

将一条**取数指令**提前到一条存数指令之前,而**存数指令**又恰好修改了被访问的存储单元,那么,这条**被提前执行的取数指令**实际上取得的是**过期的旧数据**。

例: 以下程序段:

第(4)条指令需要使用第(3)条指令的结果(即r2的内容)。为减少延时,编译器可以将第(3)条指令提到第(1)条指令之前(即提前取数至r2)。



(1) Add r5, 4, r0 ;  $4+r0 \rightarrow r5$   
(2) St (r7), r5 ; r5  $\rightarrow$  r7 所指示的存储单元  
(3) Ld r2, (r8) ; r8 指示的存储单元内容  $\rightarrow$  r2  
(4) Add r6, r2, r5 ;  $r2+r5 \rightarrow r6$

- (1) Add r5, 4, r0 ;  $4+r0 \rightarrow r5$
- (2) St (r7), r5 ;  $r5 \rightarrow r7$ 指示的存储单元
- (3) Ld r2, (r8) ; r8指示的存储单元内容 $\rightarrow r2$
- (4) Add r6, r2, r5 ;  $r2+r5 \rightarrow r6$

但可能的风险是: 如果r8的内容与r7的内容相等(比如都等于2000, 即所指示的内存单元地址为2000), 这意味着第(3)条取数指令与第(2)条存数指令访问的是同一单元。

当把取数指令提前到存数指令之前, 取数指令所取得的数据并不是程序本意需要的值。

## Itanium采用数据推测技术解决该问题:

(1) 编译器将一条取数指令分解为两条指令:

Load.a 称为高级取数指令

Load.c 称为取数检查指令

(2) 设置了一个高级取数地址表(ALTA), 该地址表中保存了所有被取数指令访问过、但未被存数指令修改过的内存单元的地址。

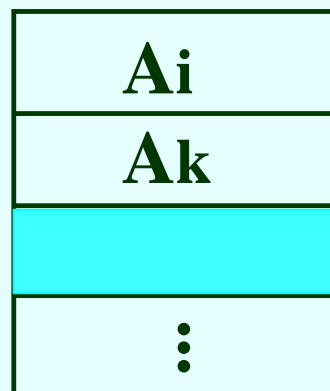
① Ld.a r2, (r8) ;

② Add r5, 4, r0 ;

③ St (r7), r5 ;

④ Ld.c r2, (r8) ;

⑤ Add r6, r2, r5 ;



ALTA

被ld指令访问过、但未被St指令修改过的内存单元的地址

- (3) 当编译器遇到取数指令时, **首先**执行“**高级取数指令Load.a**”, 在完成取数的同时, 将所访问的内存地址(如假设的2000)写入高级取数地址表。存数指令在执行时检查该表, 将所使用的地址去**匹配**ALTA表中的内容, 若存在该地址, 则将该地址从表中删除, 以表明该**地址内容已被修改**。

A <sub>i</sub>
A <sub>k</sub>
2000
⋮

**ALTA**

- (4) 编译器在原来取数指令的位置安排一条“**取数检查指令Load.c**”, 该指令用取数地址去检索高级取数地址表**ALTA**中的内容, 若存在匹配项, 意味该地址内容未被修改, “**取数检查指令Load.c**”不再执行其他操作, **否则**, 需要重新执行一次取数操作。

按照上述处理方式,前面的程序段被编译为以下代码段:

- ① **Ld.a r2, (r8)** ; r8指示的存储单元内容→r2
- ② **Add r5, 4, r0** ;  $4+r0 \rightarrow r5$
- ③ **St (r7), r5** ;  $r5 \rightarrow r7$ 指示的存储单元
- ④ **Ld.c r2, (r8)** ; 以r8的内容检索ALTA
- ⑤ **Add r6, r2, r5** ;  $r2+r5 \rightarrow r6$

上例中,执行“**Ld.a r2, (r8)**”,地址**2000**被写入**ALTA**,当执行**St**存数指令时,以r7的内容(假设也等于2000)检索ALTA,由于2000号单元内容被修改,因此**将2000从ALTA中被删除**。因此,**Ld.c取数检查还将再执行一次实际的取数操作。**

但如果r7的内容**不是**2000(即与r8的内容不同),则2000仍被保留在ALTA中,取数检查**Ld.c**不用再取数。

## (二) Itanium2 的主要特点

**性能**比Itanium1提高**50%**到**100%**。

### 1、新的高速缓存结构

片内集成3MB L3高速缓存, 更高效查询机制, 约为Itanium1 Cache的2倍。

### 2、每时钟周期执行更多指令

每个时钟周期最多也能发送6条指令, 但发送6条指令概率为90%, 平均可以发送5.7条指令。

### 3、更高的并行能力

增加2个指令整数单元, 提高了指令并行能力。

### 4、提高了带宽和吞吐量

前端总线频率由266MHz提高到400MHz、带宽由64位提高到128位, 整体带宽提高了3倍。

## Itanium1 和Itanium2的基本参数对照:

处理器	Itanium-1	Itanium-2
主频	800 MHz	1 GHz
线宽	0.18微米	0.18微米
晶体管数	25M	214 M
前端总线	266 MHz	400 MHz
系统总线接口	64位	128位
最大带宽	2.1 GB/s	6.4 GB/s
一级缓存	32 KB(芯片内)	32 KB(芯片内)
二级缓存	96 KB(芯片内)	256 KB(芯片内)
三级缓存	4 MB(外置)	3 MB(芯片内)
流水线级数	10	8
寄存器	328个	328个
执行单元	4个整数单元 2FP/2 SIMD	6个整数单元 /2FP/2 SIMD





**2017年5月, Intel推出最后一代安腾处理器,  
代号Kittson的安腾9700系列处理器**

- 集成 31亿个晶体管**
- 8核16线程**
- 频率2.66GHz**
- 三级32MB缓存(缓存总容量54MB)**



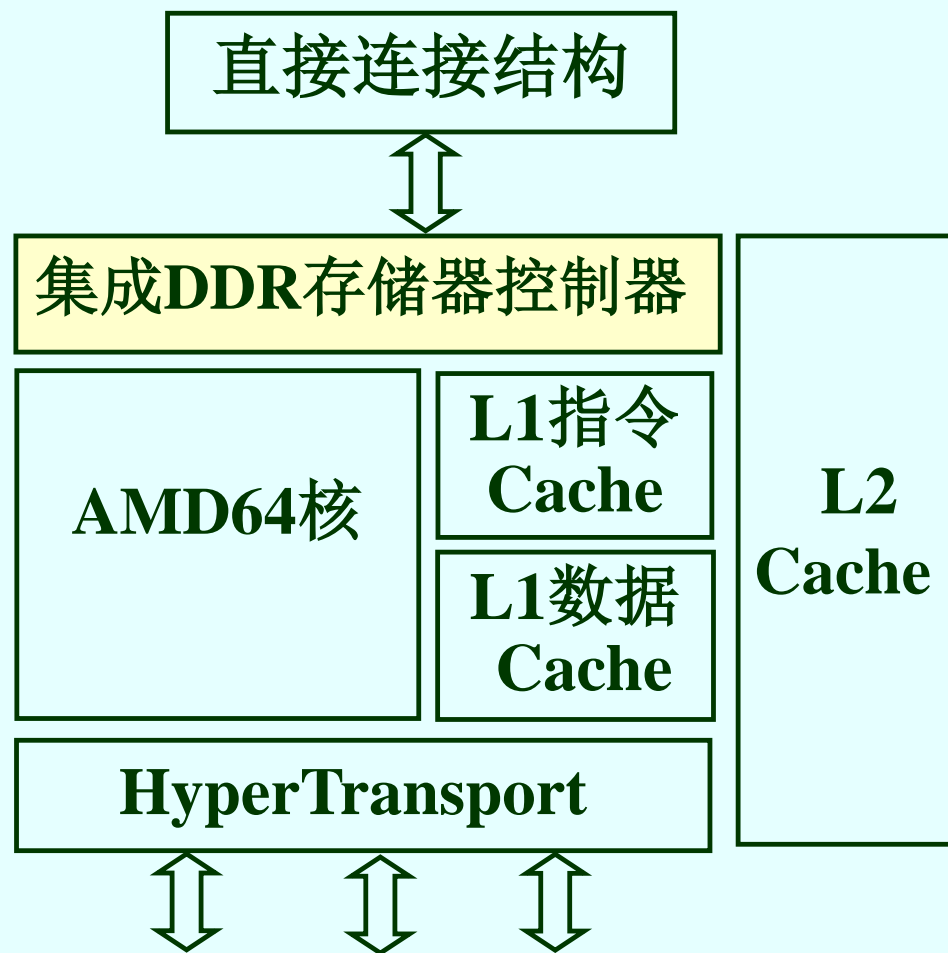
## 二、AMD 64位处理器

### (一) AMD 64位Opteron处理器(皓龙)

#### 1. 主要技术特点

- (1) 兼容X86指令集, 支持32位和64位两种程序模式
- (2) 直接连接结构(Direct Connect Architecture)减少系统瓶颈。包括:
  - 存储器与CPU直接连接, 以优化存储器性能;
  - I/O与CPU直接连接, 有利于I/O吞吐量;
  - CPU直接与CPU连接, 有利于SMP的设计。

# Opteron处理器结构框图:



## (3) HyperTransport技术

高速主板上点对点互连集成电路。在同等条件下比PCI总线速度有显著提高。可支持3条H-T链路, 提供了19.2 GB/s带宽。

## (4) 执行单元

3个整数执行单元(12级流水)和3个浮点执行单元(18级流水)。

## 2. AMDx86的64位扩展

### (1) 指令扩展

添加了一个称为“**长模式**”的新模式(Long Mode Active); 由一个称为**LMA**的控制位来启动:

**当LMA关闭时(LMA=0):**

处理器按照标准x86处理器工作(传统模式)。这时处理器与所有的16、32位操作系统以及应用程序兼容, 不能执行64位功能。

**如果长模式启动(LMA=1):** 处理器进行扩展操作。

长模式包含两种子模式: 64位模式和兼容模式。

## (二) AMD Athlon64 位处理器(速龙)

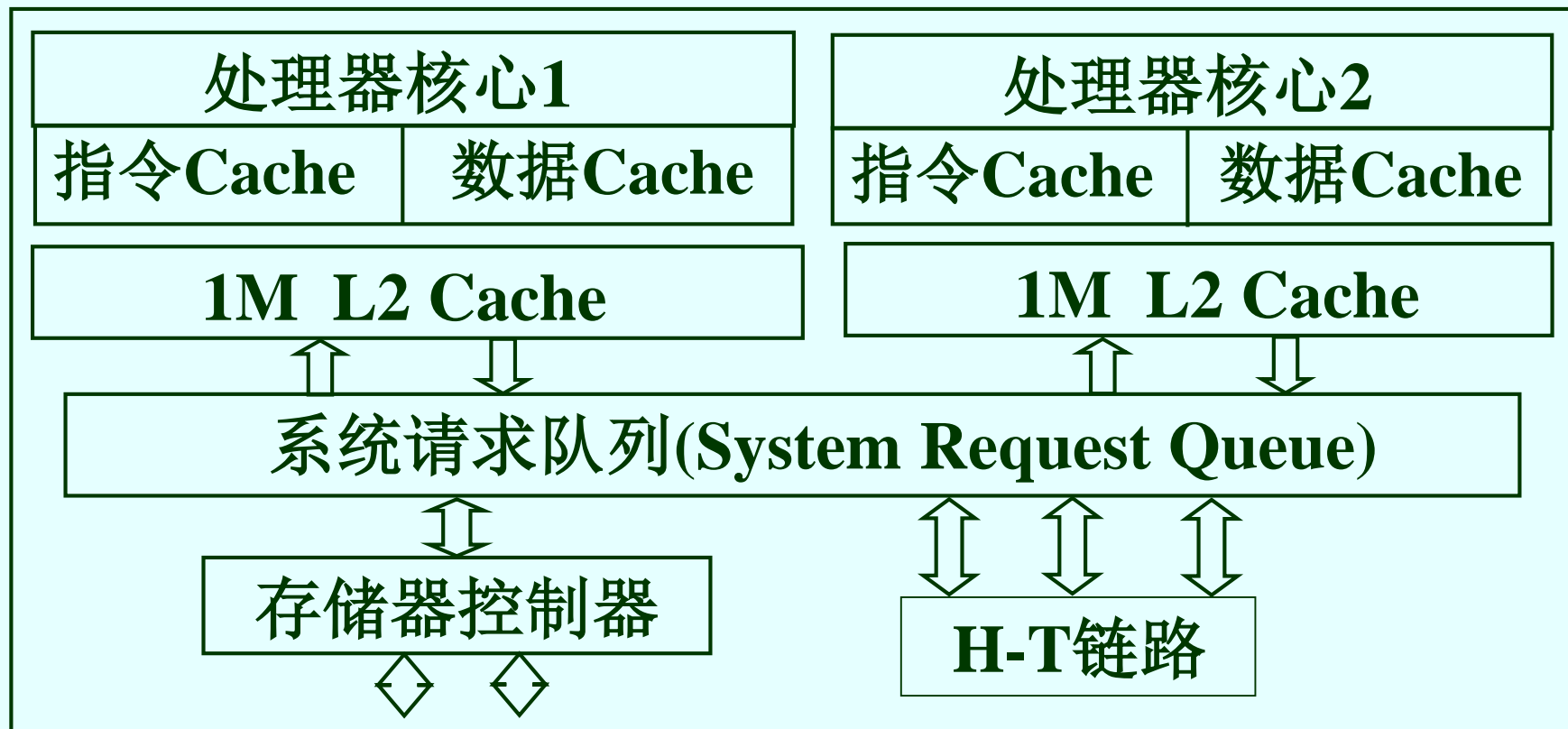
**64位桌面型处理器, 结构和工作模式与Opteron处理器基本相同。**

**双核Athlon64处理器 – Athlon64 X2**

**Athlon 64 X2内部整合了两个重要部件:**

- **仲裁部件(System Request Queue -SRQ)**  
对两个核心的任务进行仲裁和调度
- **交叉开关(Crossbar Switch)**  
对两个核心之间的通信进行协调

**Athlon 64 X2内部架构:**



2个核心整合在一起, 双核不需要通过FSB通信, 所有进程在核心内完成。在多线程/多任务环境下具有较高性能。

### 三、多核UltraSPARC T2

- 处理器内建PCI Express总线协议和Gigabit网络(包括2个Gigabit网和4个内存控制器);
- 8个核心, 每个核心支持8个线程, 使一颗处理器便具有64个逻辑处理器。Solaris操作系统支持该处理器的多核与多线程架构;
- 每个核拥有独立浮点运算器。
- 在VLIW方面, 与IA-64的区别:
  - 1) IA-64是定长128位的指令包, 并行指令不足时, 需要插入NOP指令; UltraSpace是可变长指令包(32到128位), 无须插入NOP指令。
  - 2) IA-64各功能单元基本专用(如整数/浮点/SIMD)  
每个功能单元与数据类型无关, 任意功能单元可操作任意数据类型, 即实际运行中可以同时使用所有功能单元。

## 被称为首款“系统架构于芯片”的处理器

- **网络连接** – 两个Gigabit以太网口, 内建包分类功能, 确保快速网络连接和服务端之间高速通信;
- **安全特性** – 内置8个密码加速单元(加密算法获得美国国家安全局-National Security Agency批准), 且在性能上没有任何损失;
- **输入/输出** – 8条I/O通道, 可满足PCI Express I/O高速应用(如流媒体、数据库读/写、数据备份等);
- **内存访存** – 有4个内存控制器, 提供了每秒50GB以上的内存访存速度。

## UltraSPARC T3–T8

适用于服务器整合和任务关键型企业应用程序的集成式系统，集强大的计算能力、**Oracle Solaris 11**、**Oracle Exadata**的数据库优化以及中间件云服务于一身。



# 关于国产龙芯处理器简介:

## 1、2002年推出龙芯1号

- 266MHz
- 32位处理器内核，采用类MIPS III指令集
- 七级流水线
- 32位整数单元和64位浮点单元

2. 龙芯2号在指令级并行上实现了超发射、超流水、乱序执行、分支预测等,同时具有兼容32位计算能力的64位计算功能;

龙芯2号的功耗仅有3w-5w,仅为Intel公司同等性能的Pentium芯片的3%-5% ;

- 主频1GHz
- 四发射超标量超流水结构
- 片内一级指令和数据高速缓存(各64KB), 片外二级高速缓存8MB
- 约为中等Pentium4水平

### 3. 龙芯3号(多内核、多线程技术)

龙芯3A1000、3B1000、3B1500、3A2000/3B2000、3A3000/3B3000等系列芯片

- 至少为四核处理器
- 专门服务于Java程序的协处理器, 以提高Linux环境下Java程序的执行效率
- 指令缓存追踪技术等。预计峰值每秒500-1000亿次的计算速度。
- 片内集成内存控制器以及HyperTransport (每年交\$5000的产权费)
- 处理器核通过交叉开关进行全相连

预计: 2018年推出八核龙芯B4000, 28nm工艺, 主频2GHz