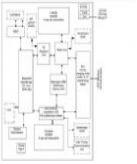
MIPS

MIPS即Million Instructions Per Second（每秒百万条指令）的简称，衡量计算机性能的指标之一，一种采取精简指令集的处理器架构。MIPS是高效精简指令集计算机(RISC)体系结构中雅的一种。



目录

- 处理器
- 构架简介

处理器

MIPS技术公司是一家设计制造高性能、次及嵌入式32位和64位处理器的厂商，在RISC处理器方面占有重要地位。1984年，MIPS计算机公司成立。1992年，SGI收购了MIPS计算机公司。1998年，MIPS脱离SGI，成为MIPS技术公司。

MIPS公司设计RISC处理器始于二十世纪八十年代初，1986年推出R2000处理器，1988年推R3000处理器，1991年推出款64位商用微处器R4000.之后又陆续推出R8000（于1994年）、R10000（于1996年）和R12000（于1997年）等型号。

随后，MIPS公司的战略发生变化，把重点放在嵌入式系统。1999年，MIPS公司发布MIPS32和MIPS64架构标准，为未来MIPS处理器的开发奠定了基础。新的架构集成了所有原来MIPS指令集，并且增加了许多更强大的功能。MIPS公司陆续开发了高性能、低功耗的32位处理器内核（core）MIPS32 4Kc与高性能64位处理器内核MIPS64 5Kc.2000年，MIPS公司发布了针对MIPS32 4Kc的版本以及64位MIPS64 20Kc处理器内核。

构架简介

MIPS体系结构首先是一种RISC架构1 MIPS32架构中有32个通用寄存器，其中$0（无论你怎么设置，这个寄存器中保存的数据都是0）和$31（保存函数调用jal的返回地址）有着特殊的用途，其它的寄存器可作为通用寄存器用于任何一条指令中。

虽然硬件没有强制性的指定寄存器使用规则，在实际使用中，这些寄存器的用法都遵循一系列约定。这些约定与硬件确实无关，但如果你想使用别人的代码，编译器和操作系统，你是遵循这些约定。

寄存器编号助记符用法0 zero永远返回值为0 1 at用做汇编器的暂时变量2-3 v0， v1子函数调用返回结果4-7 a0-a3子函数调用的参数8-15 t0-t7 24-25 t8-t9暂时变量，子函数使用时不需要保存与恢复16-25 s0-s7子函数寄存器变量。子函数必须保存和恢复使用过的变量在函数返回之前，从而调用函数知道这些寄存器的值没有变化。

26，27 k0，k1通常被中断或异常处理程序使用作为保存一些系统参数28 gp全局指针。一些运行系统维护这个指针来更方便的存取"static"和"extern"变量。

29 sp堆栈指针30 s8/fp第9个寄存器变量。子函数可以用来做桢指针31 ra子函数的返回地2 MIPS32中如果有FPA（浮点协处理器），将会有32个浮点寄存器，按汇编语言的约定为$f0~$f31，MIPS32中只能实用偶数号的浮点寄存器，奇数号的用途是：在做双精度的浮点运算时，存放该奇数号之前的偶数号浮点寄存器的剩余无法放下的32位。比如在做双精度的浮点运算时，$1存放$0的剩余的部分，所以在MIPS32中可以通过加载偶数号的浮点寄存器而把64位的双精度数据加载到两个浮点寄存器中，每个寄存器存放32位。

比如：l.d $02， 24（t1）

被扩充为两个连续的寄存器加载：lwc1 $f0， 24（t1）

lwc1 $f1， 28（t1）

3 MIPS架构中没有X86中的PC（程序计数）寄存器，它的程序计数器不是一个寄存器。因为在MIPS这样具有流水线结构的CPU中，程序计数器在同一时刻可以有多个给定的值，如jal指令的返回地址跟随其后的第二条指令。

……

jal printf move $4， $6 xxx # return here after call MIPS32中也没有条件码，比如在X86中常见的状态寄存器中的Z、C标志位在MIPS32中是没有的，但是MIPS32中是有状态寄存器4 MIPS32中不同于其它的RISC架构的地方是其有整数乘法部件，这个部件使用两个特殊的寄存器HI、LO，并且提供相应的指令mfhi/mthi，mthi/mtlo来实现整数乘法结果——hi/lo寄存器与通用寄存器之间的数据交换5数据加载与存储MIPS CPU可以在一个单一操作中存储1到8个字节。文档中和用来组成指令助记符的命名约定如下：C名字MIPS名字大小（字节）汇编助记符long long dword 8 "d"代表ld int/long word 4 "w"代表lw short halfword 2 "h"代表lh char byte 1 "b"代表lb 5.1数据加载包括这样几条记载指令LB/LBU、LH/LHU、LW byte和short的加载有两种方式。带符号扩展的lb和lh指令将数据值存放在32位寄存器的低位中，剩下的高位用符号位的值来扩充（位7如果是一个byte，位15如果是一个short）。这样就正确地将一个带符号整数放入一个32位的带符号的寄存器中。

不带符号指令lbu和lhu用0来扩充数据，将数据存放纵32位寄存器的低位中，并将高位用零来填充。

例如，如果一个byte字节宽度的存储器地址为t1，其值为0xFE（-2或254如果是非符号数），那么将会在t2中放入0xFFFFFFFE（-2作为一个符号数）。t3的值会是0x000000FE（254作为一个非符号数）

lb t2， 0（t1）

lbu t3， 0（t1）

5.2数据存储包括这样几条指令SB、SH、SW由于加载就是把寄存器中的数据加载到内存中，所以不存在位扩展的问题。

6 CP0 （协处理器0）——MIPS处理器控制用于控制和设置MIPS CPU，里面包含了一些寄存器，同过对这些寄存器的不同的位的操作可以实现对处理器的设置CP0类似于X 86只能有内核（高优先级权限）访问的一些处理器资源而前面提到的通用寄存器GPR和FPR则可以有由所有的优先级权限访问CP0提供了中断异常处理、内存管理（包括CACHE、TLB）、外设管理等途径（而这些只能由高优先级的内核才能访问到）。

6.1常见的MIPS CPU控制寄存器包括：SR（状态寄存器） 12 Config （CPU参数设置寄存器）-16 EPC （例外程序寄存器）13、CAUSE（导致中断和异常的原因寄存器） 14、BadVaddr（地址错误时存放地址的寄存器）8 Index- 0、Random-1、EntryLo0-2、EntryLo1-3、EntryHi-10、PageMask Count-9、Compare-11共同组成了高精度的时钟6.2CP0寄存器的访问指令mtc0 ts， #把通用寄存器ts中的数据送到协处理器0中的寄存器nn mfc0 ts， #把送到协处理器0中寄存器nn的值送到通用寄存器ts dmtc0 ts， #把通用寄存器ts中的数据送到协处理器0中的寄存器nn dmfc0 ts， #把送到协处理器0中寄存器nn的值送到通用寄存器ts 6.3起作用的寄存器及其作用时机加电后：你需要设置SR和Config寄存器，以确保CPU进入正确的引导状态，并且SR寄存器还设置了中断码。以决定系统响应哪些中断。

进入任何异常：处理任何例外都会调用一个“通用异常处理程序”。MIPS体系并没有为进入异常作任何的寄存器保存，也没有关于堆栈方面的任何支持，进入异常时保存的就是异常返回地址保存在EPC中。所以这些都需要操作系统的软件实现。操作系统可以使用K0或者K1寄存器作为堆栈指针，指向某个位置，并且在需要保存的寄存器保存到这个栈上。然后通过Cause寄存器找到发生异常的原因，这样才能跳转到对应的中断处理程序中。

从异常返回：从异常返回到EPC制定的地址之前要把CPU的状态回复到异常之前，好象什么事情都没有发生一样。在R3000中使用rfe指令作这样的事情，但是着条指令仅仅恢复了一些寄存器中的内容，但是并没有转移控制指令，你需要把EPC内容保存到一个通用寄存器中，然后调用jr指令返回到EPC指向的地址处，7 MIPS的存储管理模型MIPS32中的存储器模型被划分为四个大块，其中：0x0000，0000~0x7fff，ffff（0~2G-1） USEG must be mapped （set page table and TLB）and set cache before use 0x8000，0000~0x9fff，ffff（2G~2.5G-1） KSEG0 directly mapped（no need to set page table and TLB） but need to set cache before use 0xa000，0000~0xbfff，ffff（2.5G~3G-1） KSEG1 directly mapped（no need to set page table and TLB） and never use cache 0xc000，0000~0xffff，ffff（3G~4G-1） KSEG2 muse be mapped（set page table and TLB） and set cache before use这样的存储器管理模型和X86差距比较大，X86有一个实模式，内核在启动保护模式之前，运行在实模式之下，直到设定了保护模式之后才能运行在保护模式下。在MIPS32中没有保护模式那么系统是如何启动的呢?

MIPS32中的系统启动向量位于KSEG1中0xbf10，0000，由于KSEG1是directly mapped的，所以直接对应了物理地址0x1fc0，0000，你可以在内核中一直使用0xa000，0000~0xbfff，ffff之间的虚拟地址来访问物理地址0~512M-1，在设置了KSEG0的cache之后，也可一使用0x8000，0000~0x9fff，ffff之间的虚拟地址来访问0~512M-1之间的物理地址。对于512M以上的物理地址只能由KSEG2和USEG通过页表访问。

8 MIPS32中的状态寄存器SR状态寄存器来设置处理器的一些功能集合，包括设置设置协处理器0~3的可用性的位CU0~CU3（28~31）

复位向量BEV中断屏蔽位8~15 KUc、IEc0~1，基本的CPU保护单位KUc为1时表示运行在内核态，为0时运行在用户模式。在内核态下，可以访问所有的地址空间和协处理器0，运行在用户态下值只能访问0x0000，0000~0x7fff，ffff之间的地址空间。

KUp、IEp2~3当异常发生时，硬件把KUc、IEc的值保存到KUp、IEp中，并且将KUc、IEc设置为[1，0]——内核态、关中断。异常返回时rfe指令可以把KUp、IEp的内容复制到KUc、IEc中KUo、IEo当异常发生时，硬件把KUp、IEp的值保存到KUo、IEo中；返回时把KUo、IEo的内容复制到KUp、IEp中。

上面三对KU/IE位构成了深度2的栈，异常发生时，硬件自动压栈，rfe指令从异常返回时，从栈中恢复数值还有一些其它的功能和状态位，可以参考相应的文档9 MIPS32中的Cause寄存器BD位：EPC中正常情况下存放了发生异常的指令，但是当着条指令存放在调转指令的延迟槽中时，那么EPC中存放的是这个跳转指令，否则这条跳转指令将得不到执行。

IP位：告诉用户来临的中断ExcCode：这是一个5位的代码，告诉你哪一条异常发生了，可以根据这个从通用异常处理程序跳装到特定异常处理程序中。

10 MIPS32的C语言中参数传递和返回值的约定caller至少使用16bytes堆栈空间存放参数，然后把这16 bytes存放到通用寄存器a0~a3中，called subroutine直接使用寄存器中的参数，同时caller堆栈中的16bytes的数据可以不去理会了。

需要理解的是带有浮点参数和结构体的参数传递，对于带有浮点参数的传递需要看个参数是否是浮点，如果是浮点则将参数放到$f12和$f14这两个浮点寄存器中，如果个参数不是浮点数，则不用浮点寄存器存放参数。对于结构体的参数传递和x86类似对于整数和指针类型的参数返回值一般通过通用寄存器v0（$2）返回，对于浮点返回类型，一般存放在$f0中返回。

MIPS相关资讯 更多>>

1. MIPS 宣布推出其第一个 RISC-V IP 内核
2. MIPS 宣布推出其首款 RISC-V IP 内核 - eVocore P8700 多处理器
3. MIPS Technologies将放弃继续设计MIPS架构，全球三大芯片架构之一将成为历史?
4. 上海芯联芯发布承接mips在中国的经营权
5. 国产芯片重大利好，MIPS宣布CPU构架开源，下面请看龙芯表演！
6. AI芯片初创公司Wave收购硅谷老前辈MIPS

7. AI初创公司买下MIPS，蓄力叫板Nvidia
8. 复盘MIPS，为什么成不了另一个ARM
9. 联发科技选用MIPS开发LTE调制解调器
10. Sierraware 开发的兼容Global Platform的可信任执行环境TEE已支持MIPS-based 设备

MIPS相关技术资料

更多>>

基于MIPS CPU和微控制器来实现IoT节点和IoT网关的高安全性
基于MIPS处理器内核的通用MPU
MIPS - Lauterbach TRACE32工具的新增支持功能可简化MIPS 和ARM CPU 结合设计的调试工作
Imagination新推MIPS I-class I6400 CPU
基于VLSI平台的MIPS处理器仿真与设计
MIPS推新Aptiv，逆袭ARM的4个理由
MIPS将Myriad应用带向电视
基于MIPS的普适农业监测系统设计