32 位 CPU 计软联合实验组
需求文档

Totoro 组

1 引言

1.1 背景

本项目为计算机组成原理与软件工程联合实验项目，目标为设计并实现一个基于 MIPS32 指令集的 CPU 及其周边硬件。

项目的开发者为计 64 班的杨乐、董原良、李林涵，计 65 班的琚锡廷。

1.2 编写目的

本文档的编写时间处于项目初期，目的是希望在正式开始工作之前明确工作目标，对工作量有一个大体的把握。

1.3 定义

MIPS	Microprocessor without interlocked piped stages / Millions of Instructions Per Second 采取精简指令集的处理器架构
RISC	精简指令集
CPU	中央处理器
ALU	算术逻辑单元
MMU	内存管理单元
TLB	加速页表的查询
CP0	0 号协处理器，用于处理异常
RAM	随机存取存储器
SRAM	静态随机存取存储器
ROM	只读存储器
Flash	随机存取存储器
BIOS	基础输入输出系统

1.4 开发环境

开发板：THU-MIPS

FPGA：Xilinx Spartan-3E XC3S1200EFGG320

CPLD：Xilinx XC9500

RAM：ISSI 公司生产的异步高速 CMOS SRAM，型号为 IS61LV256-10TI

Flash：MT28F640J3

1.5 模块概述

1.5.1 CPU

CPU 的是本项目的运算和控制核心，其中包括指令系统、数据通路、流水线结构、运算器、寄存器、异常与中断处理等，其中的 MMU 负责管理虚拟存储器、物理存储器的控制线路。

1.5.2 MMU

MMU 负责衔接系统与 CPU、外设之间虚拟地址、物理地址之间的转换。

1.5.3 串口与外围设备

串口是连接项目各个主要部分中的通讯方式。外围设备主要包括 VGA 通信和 PS2 键盘通信。

1.6 参考资料

1. 《计算机硬件系统实验教程》刘卫东、李山山、宋佳兴
2. 《自己动手写 CPU》雷思磊

2 功能需求

2.1 MIPS32 指令集

Ucore 所依赖的 MIPS 指令共的 50 条。包括逻辑操作指令、移位操作指令、移动操作指令、算术操作指令、转移指令、加载存储指令、协处理器访问指令、异常相关指令，如下所示：

逻辑操作指令	OR、ORI、AND、ANDI、XOR、XORI、NOR、LUI
移位操作指令	SLL、SLLV、SRL、SRLV、SRA、SRAV
算术操作指令	ADDIU、ADDU、SUBU、SLT、SLTU、 SLTI、SLTIU、MULT、DIVU
移动操作指令	MFHI、MFLO、MTHI、MTLO
分支跳转指令 (B 型)	BEQ、BGEZ、BGTZ、BLEZ、BLTZ、BNE
分支跳转指令 (J 型)	J、JAL、JALR、JR
访存指令	LB、LW、SB、SW、LBU、LHU
协处理器访问指令	MFC0、MTC0
异常相关指令	SYSCALL、ERET、TLBWI、TLBWR

另外，Ucore 系统反汇编后会产生一些伪指令，这些指令是无需实现的：MOVE、NEGU、LI、BEGZ、B、BNEZ、EHB。

上述指令的具体列表将附录于文档后。

2.2 Ucore 操作系统

本实验使用移植于 MIPS32S 的 Ucore 系统。Ucore 是一个包括存储管理、多级流水、的类 linux 操作系统。

2.2.1 编译

Ucore 主要利用的编译工具为 Gcc，最好使用网上提供的交叉编译工具链。

在 Ucore 的编译选项中，提供了 ON_FPGA 编译开关；当使用 qemu 仿真时，置 ON_FPGA=n；当生成在板子上运行的机器代码时，ON_FPGA=y。

2.2.2 仿真

实验过程中需要经过 qemu 仿真流程；qemu 仿真的原理是为 Ucore 操作系统提供一个底层物理硬件的仿真环境，在 qemu 可以测试操作系统的正确性。

仿真的目的有两点：1，在调试阶段可以对比 qemu 仿真和板子的输出结果，从而找出可能的错误；2，在实验后期对操作系统进行拓展功能的修改时，可以验证其正确性。

2.2.3 Boot 与 Bootloader

Boot 为操作系统的启动阶段。第一阶段是硬件初始化，第二阶段由 Bootloader，即从硬盘中加载操作到内存的引导程序执行。第三阶段则是对操作系统进行初始化。

2.2.4 内存管理

在 memlayout.h 中定义的常量 KERNBASE 和 KMEMSIZE。KERNBASE 定义 kseg0 的起始地址，KMEMSIZE 为计算机可用的物理内存大小。在 BootLoader 中定义了 Flash 的起始地址和大小。项目中使用 MMU 来实现对其的操作。

2.2.5 异常处理

异常处理的流程为：

1. 硬件检测到异常发生后，讲异常信息储存于 CP0 的 Ebase 寄存器，跳转到异常处理入口，汇报给 Ucore 系统。

2. 系统保存异常状态后进行相应的处理，如果异常类型不支持，则陷入 panic 状态，并输出陷入帧 (Trapframe)。

3. 处理结束后，进入异常返回过程，恢复保存的异常状态，继续执行被中断程序。CPU 跳转到被异常打断的指令重新运行。

2.3 CPU

2.3.1 ALU

算术逻辑单元 (arithmetic and logic unit) 是能实现多组算术运算和逻辑运算的组合逻辑电路，用以计算机指令集中的执行算术与逻辑操作。ALU 通常有两个数据输入端 A 和 B 输入操作数，一个数据输出端 Y 以及标志位输出结果，通过输入操作吗 op 来确定所要进行的操作。通常，输入操作数、操作数、累加和以及转换结果的存储位置都在 ALU 中。在算术单元中，乘除操作是通过一系列的加减运算得到的。在机器码中有多种方式用以表示负数。在逻辑单元中，每次执行 16 个可能的逻辑运算中的一个。

2.3.2 乘法器

由于乘法的结果超出了 ALU 输出格式的位数，需要单独元件实现，但输入格式和 ALU 基本保持一致，只是输出结果为 64 位。

2.3.3 除法器

由于 Ucore 需要支持 DIVU 指令，所以需要用到除法器；除法器一般采用加减交替法来模拟执行，并且在除法执行时，需要暂停流水线。

2.3.4 寄存器

RISC 的特点时提到一点：大量使用寄存器。这是因为寄存器的存取可以在一个时钟周期内完成，同时也简化了寻址方式。MIPS32 的指令中除加载/存储指令外，都是使用寄存器或立即数作为操作数的。MIPS32 中的寄存器分为三类：通用寄存器（GPR：General Purpose Register）、特殊寄存器、CP0 寄存器。

通用寄存器共 32 个，这里就不再细述；

特殊寄存器有三个：PC（Program Counter 程序计数器）、HI（乘除结果高位寄存器）、LO（乘除结果低位寄存器）。进行乘法运算时，HI 和 LO 保存乘法运算的结果，其中 HI 存储高 32 位，LO 存储低 32 位；进行除法运算时，HI 和 LO 保存除法运算的结果，其中 HI 存储余数，LO 存储商；

CPO 寄存器在下面部分介绍。

2.3.5 CP0 与异常处理

协处理器 CP0 是体系结构中必须实现的。它起到控制 CPU 的作用。MMU、异常处理等功能，都依赖于协处理器 CP0 来实现。

MIPS 的 CP0 包含 32 个寄存器，Ucore 需要使用的寄存器如下表所列:

Ucore	寄存器号	寄存器名	寄存器功能
*	0	Index	TLB 阵列的索引，在 TLBP、TLBP 和 TLBWI 指令中访问 TLB 入口
*	2	EntryLo0	访问 TLB 中偶数页的地址低 32 位
*	3	EntryLo1	访问 TLB 中奇数页的地址低 32 位
	5	PageMask	控制 TLB 的页大小
*	8	BadVAddr	捕获最近一次引起异常的虚拟地址
*	9	Count	用作计时器
*	10	EntryHi	访问 TLB 中地址的高 32 位
*	11	Compare	与 Count 寄存器一同完成定时器和定时中断功能
*	12	Status	表示处理器的操作模式、中断使能和诊断状态
*	13	Cause	记录最近一次异常的原因
*	14	EPC	存储异常处理后程序恢复执行的地址
*	15	Ebase	保存异常处理程序的入口地址

CP0 中常用的几个寄存器是：BadVAddr，Count/Compare，Status/Cause，EPC，Ebase。

以下是对上述寄存器的一些注记：

1. **BadVAddr**：此寄存器用于捕获最近一次引起异常的虚拟地址。但实际上，这个寄存器在 TLB 重填/失效/修改，以及地址错误 (Address Error) 两种异常的时候，才能成功捕获。

2. **Count/Compare**：寄存器 Count 与 Compare 协同工作共同完成系统的计时与中断功能。Count 寄存器是一个稳定的计时器，以一个稳定的速度增加 (计数频率一般与 CPU 时钟频率相同)；而 Compare 寄存器则不会变动，维持稳定。在寄存器 Count 与 Compare 相等时，系统会发出一个中断的请求；这个特性用于维持一个稳定的时钟。

3. **Status**: Status 寄存器用于表示处理器的操作模式、中断使能和诊断状态; 其具体的字段如下所示:

Bits	31	30	29	28	27	26	25	24	23	22	21
Name	CU3-CU0				RP	FR	RE	MX	0	BEV	TS
Bits	20	19	18	17	16	15	14	13	12	11	10
Name	SR	NMI	ASE	Impl		IM7-IM2					
Bits	9	8	7	6	5	4	3	2	1	0	-
Name	IM1-IM0		0			UM	R0	ERL	EXL	IE	-

以下介绍部分字段的含义及相应的功能:

- **CU3-CU0**: 其中 CU2-CU0 分别表示协处理器 CP2、CP1、CP0 是否可用; 在实验中由于只使用协处理器 CP0, 故这几位可置为 4'b0001;
- **RP**: 是否启用低功耗模式;
- **FR**: 用于指示 64 位浮点数单元控制: 当浮点寄存器可以支持 64 位时, 该位被置为 1;
- **BEV**: 控制异常向量的位置: 当启用位置置于 Bootstrap 时, 该位被置为 1;
- **TS**: 控制关断 TLB: 当 TLB 被关断时, 该位被置为 1 (该位只能通过写入 0 实现清 0, 不能强制实现 0-1 的转换);
- **SR**: 表示是否为软重启: 当重启异常为软重启造成, 该位被置为 1;
- **NMI**: 表示是否为不可屏蔽中断: 当重启异常为不可屏蔽中断造成, 该位被置为 1;
- **IM7-IM0**: 表示是否屏蔽对应的中断: 8 个中断位分别控制 8 个中断源, 其中 6 个为硬件中断源, 2 个为软件中断源; 当允许该中断源的中断请求时, 该位被置为 1;
- **KSU**: UM 与 R0 两位组成 KSU, 该区域定义了处理器的基本工作模式:

  > 00: 工作于 Kernel 模式

> 01: 工作于 Supervisor 模式

> 10: 工作于 User 模式

> 11: 保留

- **ERL**：是否处于错误级：当重启、软重启或 NMI 异常发生时，该位被置于 1；
- **EXL**：是否处于异常集：当除重启、软重启或 NMI 异常以外的任何异常发生时，该位被置为 1；
- **IE**：表示中断使能：当全局允许中断时，该位被置为 1。

4. **Cause**：Cause 寄存器用于记录最近一次异常的原因，其具体的字段如下所示：

Bits	31	30	29	28	27	26	25	24	23	22	21
Name	BD	TI	CE		DC	PCI	ASE		IV	WP	FDCI
Bits	20	19	18	17	16	15	14	13	12	11	10
Name	000			ASE		IP7-IP2					
Bits	9	8	7	6	5	4	3	2	1	0	-
Name	IP1-IP0		0	Exc Code					0		-

以下介绍部分字段的含义及相应的功能：

- **BD**：指示最近一次异常是否处于分支延迟槽中：如果处于延迟槽中则置 1；
- **IP7-TP2**：说明有中断挂起；其中 IP7-IP2 分别指示对应硬件中断是否发生；
- **IP1-TP0**：控制软中断请求；其中 IP1-IP0 分别指示软件是否请求中断；
  - **Exc Code**：异常编码；其中 ucore 系统涉及到的异常如下所示：

异常代码值		助记符	描述
十进制	十六进制		
0	16#00	Int	中断
1	16#01	Mod	TLB 异常（修改）
2	16#02	TLBL	TLB 异常（载入、取指）
3	16#03	TLBS	TLB 异常（存储）
4	16#04	AdEL	地址错误异常（加载、取指）
5	16#05	AdES	地址错误异常（存储）
8	16#06	Sys	系统调用异常
10	16#0a	RI	保留的指令异常
11	16#0b	CpU	协处理器不可用异常
12	16#0c	Ov	算术溢出异常

2.4 流水线结构

2.4.1 数据通路

本实验希望实现一条五级流水线。每条 MIPS 指令的执行过程就分为五级，每一级称为一个流水线阶段，每个阶段占用固定的时间。这个固定的时间通常就是一个处理器时钟周期。在这种设计下，每条 MIPS 指令需要经过如下操作。

1. IF 取指 (insturction fetch), 从指令高速缓存 (I-cache) 获取下一条指令；
2. RD 读取寄存器 (read register), 读取该指令的源寄存器域指定的 CPU 寄存器的内容；
3. ALU 算术逻辑单元 (arithmetic/logic unit) 在一个时钟周期内完成算术或者逻辑操作 (浮点运算和整数乘除无法在一个时钟周期内完成，对其处理有所不同，后面会讲到这一点)；
4. MEM 访问内存 (memory), 该阶段指令可以读写数据高速缓存 (D-cache) 中的内存变量。平均而言，每四条指令中就有三条指令在该阶段没有任何操作，

但是为每条指令都分配了这一阶段以确保不会出现两条指令同时需要访问数据高速缓存的情形；

5. WB 写回寄存器 (write back), 将操作结果值写道寄存器堆中。

2.4.2 冒险

流水线中每个节拍硬件被充分利用，这只存在于理想的乌托邦中，现实总会出现这样或那样的问题，让原本顺畅的流水线出现停顿，断断续续。这些导致流水线出现停顿的因素称为流水线冒险（Hazard）。

1．结构冒险 因为处理器资源冲突，而无法实现某些指令的组合，就称该处理器有结构冒险。

在 MIPS 流水线中，指令和数据都存储在存储器中，IF 阶段需要访问存储器，MEM 阶段也需要访问存储器，在早期的处理器中，程序和数据存储器没有分开，下图在第 4 个 Cycle，IF 和 MEM 同时访问存储器导致其中一个操作要等待。现在的处理器，程序存储在 L1P Cache 中，数据存储在 L1D Cache 中（关于 Cache 在下一章介绍），单独访问，因此不存在这个问题。

2．数据冒险 流水线使原先有先后顺序的指令同时处理，当出现某些指令的组合时，可能会导致指令使用了错误的数据。

一种解决方法是在这两条语句之间增加两个 Cycle 的等待，但对效率的影响比较大。

另一种方法是使用直通（Forwarding）来解决这个问题。

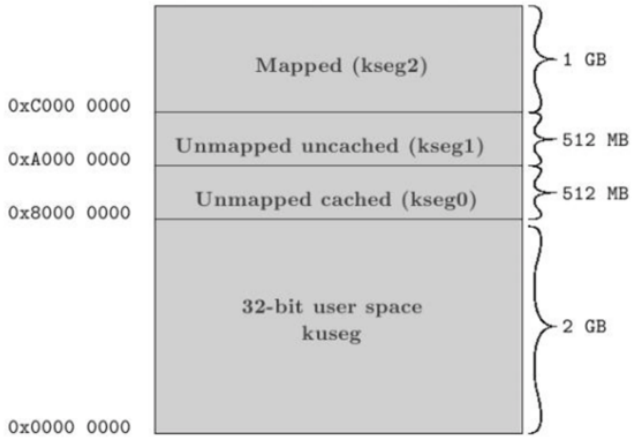
当硬件检测到当前指令的源操作数正好在 EX/MEM 流水线寄存器中时，就直接将 EX/MEM 寄存器的值传递给 ALU 的输入，而不是从寄存器堆中读数据。

3．控制冒险 从微观的角度，在流水线处理器中，指令是并行处理的，在当前指令正在执行时，后面的很多条指令已经完成了取指和译码等步骤。然而，在程序中会存在很多的跳转语句，如果程序的实际执行路径是要跳转到其他的地址去执行，那么流水线中已经做的这些取指和译码工作就白做了，这就是流水线的控制冒险。此时，处理器需要排空流水线，跳转到新的地址处重新进入流水线。由此可知，跳转对程序性能的损失是巨大的，流水线越深，损失越大。

2.5 MMU 存储控制单元

MMU 是 Memory Management Unit 的缩写，中文名是内存管理单元，它是 CPU 中用来管理虚拟存储器、物理存储器的控制线路，同时也负责虚拟地址映射为物理地址，以及提供硬件机制的内存访问授权，多用户多进程操作系统。

MIPS32 架构的 MMU 主要实现虚拟内存映射的功能，以读写控制信号和虚拟地址为输入，实现对应物理地址数据的存储和访问。



MIPS 存储器映射：32 位情形

其中，kseg1 是唯一在系统重启时也能正常工作的内存映射地址空间，是重新启动的入口向量所在区域。这段空间用来访问初始化程序的 ROM、I/O 寄存器。

2.5.1 TLB

TLB 为 translation lookaside buffer 的简称，俗称快表，直译为旁路快表缓冲，也可以理解为页表缓冲，地址变换高速缓存。

TLB 是一种高速缓存，内存管理硬件使用它来改善虚拟地址到物理地址的转换速度。当前所有的个人桌面，笔记本和服务器处理器都使用 TLB 来进行虚拟地址到物理地址的映射。使用 TLB 内核可以快速的找到虚拟地址指向物理地址，而不需要请求 RAM 内存获取虚拟地址到物理地址的映射关系。这与 data cache 和 instruction caches 有很大的相似之处。

当 cpu 要访问一个虚拟地址/线性地址时，CPU 会首先根据虚拟地址的高 19 位在 TLB 中查找。如果是表中没有相应的表项，称为 TLB miss，需要通过访问主存中的页表计算出相应的物理地址。同时，物理地址被存放在一个 TLB 表项中，以后对同一线性地址的访问，直接从 TLB 表项中获取物理地址即可，称为 TLB hit。

2.6 存储器

2.6.1 SRAM

静态随机存取存储器（Static Random-Access Memory，SRAM）是随机存取存储器的一种。所谓的“静态”，是指这种存储器只要保持通电，里面储存的数据就可以恒常保持。相对之下，动态随机存取存储器（DRAM）里面所储存的数据就需要周期性地更新。然而，当电力供应停止时，SRAM 储存的数据还是会消失（被称为 volatile memory），这与在断电后还能储存资料的 ROM 或闪存是不同的。

THINPAD 教学计算机的实验板使用了 2 片 SRAM 作为主要存储器，每片存储容量为 2MB。其中 ExtSRAM 为内存，BaseRAM 由于有 CPLD 串口的关系，使用为扩展内存。

2.6.2 Flash

flash 是存储芯片的一种，通过特定的程序可以修改里面的数据。FLASH 在电子以及半导体领域内往往表示 Flash Memory 的意思，即平时所说的“闪存”，全名叫 Flash EEPROM Memory。flash 存储器又称闪存，它结合了 ROM 和 RAM 的长处，不仅具备电子可擦除可编程（EEPROM）的性能，还可以快速读取数据（NVRAM 的优势），使数据不会因为断电而丢失。

THINPAD 教学计算机的实验板使用的 Flash 储存器大小为 8MB，其数据线为 16 位，地址线为 23 位。Flash 存储器被当做计算机的“硬盘”来使用，除存储操作系统 Ucore 的核心代码外，还可以存储用户程序和数据。

2.7 串口与外围设备

2.7.1 串口

计算机内部数据通常以字节为单位进行并行访问，但与外部设备进行数据交互或与其他计算机进行通信时，也经常需要串行数据通信。

THINPAD 教学计算机的 FPGA 芯片通过 UART 芯片连接 RS232 接口，作为串行接口与其它设备相连。在计算机启动测试之前，操作系统 Ucore 的 ELF 文件将通过串口写入计算机的 Flash 存储器中。

2.7.2 VGA 设备

VGA 是 Video Graphics Array 的缩写，是一种标准的显示接口，在现实中得到了广泛应用。VGA 必须工作在字符显示模式或图形显示模式下。

本实验分辨率为 640X480 像素，帧频率为 75Hz 时需要像素时钟为 50MHz。

3 MIPS32 指令集

以下为 ucore 系统所要求的 MIPS32 指令集。

1. 逻辑操作指令

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	1	0	0	1	0	0
MIPS 语言	AND rd rs rt															
指令功能	R[d] ⇐ R[s] & R[t]															
功能说明	将寄存器 rs 与 rt 的值进行与操作，并将结果保存在寄存器 rd 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	1	1	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	immediate															
MIPS 语言	ANDI rt rs immediate															
指令功能	R[t] ⇐ R[s] & zero_extend(immediate)															
功能说明	将寄存器 rs 与立即数进行与操作，并将结果保存在寄存器 rt 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	1	1	1	1	0	0	0	0	0	rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	immediate															
MIPS 语言	LUI rt immediate															
指令功能	R[t] ⇐ immediate || 0^{16}															
功能说明	将立即数存放在寄存器 rt 的高 16 位中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	1	0	0	1	0	1
MIPS 语言	OR rd rs rt															
指令功能	R[d] ⇐ R[s] | R[t]															
功能说明	将寄存器 rs 与 rt 的值进行或操作，并将结果保存在寄存器 rd 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	1	1	0	1	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	immediate															
MIPS 语言	ORI rt rs immediate															
指令功能	R[t] ⇐ R[s] | zero_extend(immediate)															
功能说明	将寄存器 rs 与立即数进行或操作，并将结果保存在寄存器 rt 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	1	0	0	1	1	0
MIPS 语言	XOR rd rs rt															
指令功能	R[d] ⇐ R[s] ∧ R[t]															
功能说明	将寄存器 rs 与 rt 的值进行异或操作，并将结果保存在寄存器 rd 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	1	1	1	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	immediate															
MIPS 语言	XORI rt rs immediate															
指令功能	R[t] ⇐ R[s] ∧ ZeroExtend(immediate)															
功能说明	将寄存器 rs 与立即数进行异或操作，并将结果保存在寄存器 rt 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	1	0	0	1	1	1
MIPS 语言	NOR rd rs rt															
指令功能	R[d] ⇐ ∼ (R[s] | R[t])															
功能说明	将寄存器 rs 与 rt 的值进行或非操作，并将结果保存在寄存器 rd 中															

2. 移位操作指令

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	0	0	0	0	0	rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					sa					0	0	0	0	0	0
MIPS 语言	SLL rd rt sa															
指令功能	R[d] ⇐ R[t] « sa															
功能说明	将寄存器 rt 的值左移 sa 位，并保存在寄存器 rd 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	0	0	0	1	0	0
MIPS 语言	SLLV rd rt rs															
指令功能	R[d] ⇐ R[t] « R[s]4-0															
功能说明	将寄存器 rt 的值左移 rs 位，并保存在寄存器 rd 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	0	0	0	0	0	rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					sa					0	0	0	0	1	1
MIPS 语言	SRA rd rt sa															
指令功能	R[d] ⇐ R[t] $»_A$ sa															
功能说明	将寄存器 rt 的值算术右移 sa 位，并保存在寄存器 rd 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	0	0	0	1	1	1
MIPS 语言	SRAV rd rt rs															
指令功能	R[d] ⇐ R[t] $»_A$ R[s]{4-0}															
功能说明	将寄存器 rt 的值算术右移 rs 位，并保存在寄存器 rd 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	0	0	0	0	0	rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					sa					0	0	0	0	1	0
MIPS 语言	SRL rd rt sa															
指令功能	R[d] ⇐ R[t] $»_L$ sa															
功能说明	将寄存器 rt 的值逻辑右移 sa 位，并保存在寄存器 rd 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	0	0	0	1	1	0
MIPS 语言	SRLV rd rt rs															
指令功能	R[d] ⇐ R[t] $»_L$ R[s]{4-0}															
功能说明	将寄存器 rt 的值逻辑右移 rs 位，并保存在寄存器 rd 中															

3. 算术操作指令

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	1	0	0	1	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	immediate															
MIPS 语言	ADDIU rt rs immediate															
指令功能	R[t] ⇐ R[s] + Sign-extend(immediate)															
功能说明	将寄存器 rs 加上作符号扩展的立即数后，存入寄存器 rt															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	1	0	0	0	0	1
MIPS 语言	ADDU rd rs rt															
指令功能	R[d] ⇐ R[s] + R[t]															
功能说明	将寄存器 rs 加上寄存器 rt 的值后，存入寄存器 rd															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	1	0	0	0	1	1
MIPS 语言	SUBU rd rs rt															
指令功能	R[d] ⇐ R[s] - R[t]															
功能说明	将寄存器 rs 减去寄存器 rt 的值后，存入寄存器 rd															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
MIPS 语言	MULT rs rt															
指令功能	(HI, LO) ⇐ R[s] * R[t]															
功能说明	寄存器 rs 与寄存器 rt 做乘法； 将低 32 位存储在 LO 寄存器中，将高 32 位存储在 HI 寄存器中.															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1
MIPS 语言	DIVU rs rt															
指令功能	(HI, LO) ⇐ R[s] / R[t]															
功能说明	寄存器 rs 与寄存器 rt 做除法； 将商存储在 LO 寄存器中，将余数存储在 HI 寄存器中.															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	1	0	1	0	1	0
MIPS 语言	SLT rd rs rt															
指令功能	R[d] ⇐ R[s] < R[t]															
功能说明	寄存器 rs 与寄存器 rt 作比较有符号数比较； 当 rs 较小时，rd 赋值为 1；否则赋值为 0.															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	1	0	1	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	immediate															
MIPS 语言	SLTI rt rs immediate															
指令功能	R[t] ⇐ R[s] < sign_extend(immediate)															
功能说明	寄存器 rs 与作符号扩展后的立即数作有符号数比较； 当 rs 较小时，rd 赋值为 1；否则赋值为 0.															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	1	0	1	1	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	immediate															
MIPS 语言	SLTIU rt rs immediate															
指令功能	R[t] ⇐ R[s] < zero_extend(immediate)															
功能说明	寄存器 rs 与作零扩展后的立即数作无符号数比较; 当 rs 较小时，rd 赋值为 1；否则赋值为 0.															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	1	0	1	0	1	1
MIPS 语言	SLTU rd rs rt															
指令功能	R[d] ⇐ R[s] < R[t]															
功能说明	寄存器 rs 与寄存器 rt 作比较无符号数比较; 当 rs 较小时，rd 赋值为 1；否则赋值为 0.															

4. 移动操作指令

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	0	1	0	0	0	0
MIPS 语言	MFHI rd															
指令功能	R[d] ⇐ HI															
功能说明	将 HI 寄存器的值保存至寄存器 rd 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	0	1	0	0	1	0
MIPS 语言	MFLO rd															
指令功能	R[d] ⇐ LO															
功能说明	将 LO 寄存器的值保存至寄存器 rs 中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
MIPS 语言	MTHI rs															
指令功能	HI ⇐ R[s]															
功能说明	将寄存器 rs 的值保存至 HI 寄存器中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1
MIPS 语言	MTLO rs															
指令功能	LO ⇐ R[s]															
功能说明	将寄存器 rs 的值保存至 LO 寄存器中															

5. 分支跳转指令

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	1	0	0	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	offset															
MIPS 语言	BEQ rs rt offset															
指令功能	PC ⇐ PC + (R[s] = R[t]) * offset															
功能说明	当寄存器 rs 与 rt 相等时，跳转至目的地址															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	1	rs					0	0	0	0	1
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	offset															
MIPS 语言	BGEZ rs offset															
指令功能	PC ⇐ PC + (R[s] >= 0) * offset															
功能说明	当寄存器 rs 不小于 0 时，跳转至目的地址															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	1	1	1	rs					0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	offset															
MIPS 语言	BGTZ rs offset															
指令功能	PC ⇐ PC + (R[s] > 0) * offset															
功能说明	当寄存器 rs 大于 0 时，跳转至目的地址															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	1	1	0	rs					0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	offset															
MIPS 语言	BLEZ rs offset															
指令功能	PC ⇐ PC + (R[s] <= 0) * offset															
功能说明	当寄存器 rs 不大于 0 时，跳转至目的地址															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	1	rs					0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	offset															
MIPS 语言	BLTZ rs offset															
指令功能	PC ⇐ PC + (R[s] < 0) * offset															
功能说明	当寄存器 rs 小于 0 时，跳转至目的地址															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	1	0	1	rs					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	offset															
MIPS 语言	BNE rs rt offset															
指令功能	PC ⇐ PC + (R[s] ≠ R[t]) * offset															
功能说明	当寄存器 rs 与 rt 不等时，跳转至目的地址															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	1	0	instr_index									
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	instr_index															
MIPS 语言	J instr_index															
指令功能	PC ⇐ PC{31-28} || instr_index || 00															
功能说明	无条件跳转至目的地址															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	1	1	instr_index									
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	instr_index															
MIPS 语言	JAL instr_index															
指令功能	PC ⇐ PC{31-28} || instr_index || 00；RA ⇐ RPC															
功能说明	无条件跳转目的地址 rs，并将延迟槽后一条指令存入 RA															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	0	0	1	0	0	1
MIPS 语言	JALR rd rs															
指令功能	R[d] ⇐ RPC；PC ⇐ R[s]															
功能说明	无条件跳转目的地址 rs，并将延迟槽后一条指令存入 rd															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	rs					0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
MIPS 语言	JR rs															
指令功能	PC ⇐ R[s]															
功能说明	无条件跳转至目的地址 rs															

6. 访存指令

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	1	0	0	0	0	0	base					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	offset															
MIPS 语言	LB rt, offset(base)															
指令功能	R[t] ⇐ Memory_Byte[base + offset])															
功能说明	在内存中加载一个 byte 类型数															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	1	0	0	1	0	0	base					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	offset															
MIPS 语言	LBU rt, offset(base)															
指令功能	R[t] ⇐ zero_extend(Memory_Byte[base + offset])															
功能说明	在内存中加载一个 byte 类型数，并作零扩展															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	1	0	0	0	1	1	base					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	offset															
MIPS 语言	LW rt, offset(base)															
指令功能	R[t] ⇐ Memory_Word[base + offset]															
功能说明	在内存中加载一个 word 类型数															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	1	0	1	0	0	0	base					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	offset															
MIPS 语言	SB rt, offset(base)															
指令功能	Memory [base + offset] ⇐ R[t]{7-0}															
功能说明	往内存中存储一个 byte 类型数															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	1	0	1	0	1	1	base					rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	offset															
MIPS 语言	SW rt, offset(base)															
指令功能	Memory [base + offset] ⇐ R[t]															
功能说明	往内存中存储一个 word 类型数															

7. 协处理器访问指令

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	0	0	0	0	0	0	0	0	0	rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	0	0	0	0	0	0
MIPS 语言	MFC0 rt, rd															
指令功能	R[t] ⇐ CP0[R[d]]															
功能说明	将 CP0 寄存器值移动到普通寄存器中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	0	0	0	0	0	0	1	0	0	rt				
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					0	0	0	0	0	0	0	0	0	0	0
MIPS 语言	MTC0 rt, rd															
指令功能	CP0[R[d]] ⇐ R[t]															
功能说明	将普通寄存器值移动到 CP0 寄存器中															

8. 异常相关指令

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	code									
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	code										0	0	1	1	0	0
MIPS 语言	SYSCALL															
指令功能	SignalException(SystemCall)															
功能说明	系统调用															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	0	0	0	0	0	code									
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	code										0	0	1	1	0	1
MIPS 语言	BREAK															
指令功能	SignalException(Breakpoint)															
功能说明	产生系统断点异常															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
MIPS 语言	ERET															
指令功能	返回到异常发生的位置															
功能说明	异常返回															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
MIPS 语言	TLBWI															
指令功能	写 index 寄存器索引的 TLB															
功能说明	写 TLB															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
MIPS 语言	TLBWR															
指令功能	写 random 寄存器索引的 TLB															
功能说明	写 TLB															