Tsinghua University

Computer Organization Experimentation

# Project Requirement Document

Author:
Shizhi Tang
Xihang Liu
Zixi Cai

Supervisor:
Yuxiang Zhang
Prof. Weidong Liu

January 21, 2018

# Contents

# 1 引言

## 1.1 项目背景

本项目旨在设计一个基于 MIPS32 的 CPU 及其相关硬件，最终结果是，可以支持一个 MIPS32 指令的一个子集，最终能够运行 UCORE 操作系统。

也有一些可选的扩展任务，包括以下内容：支持了 UBoot（共需要 58 条指令），此外也完成了以太网、USB、HDMI 的扩展工作。我们最终实现的需求是，可以通过以太网或者 USB 进行 Boot 操作，最终达到加载系统的目的。

本项目是计算机组成原理课程和软件工程课程的联合实验，需求方为计算机组成原理课程和软件工程课程。承担此项目的是 BnG 小组，成员包括计 53 班的唐适之、刘熙航和蔡子熙。

## 1.2 编写目的

本文档为该项目的需求文档，初版编写于项目正式启动之前，用于明确 CPU 中主要需要实现的功能，硬件设计中主要用到的技术和 UCORE 操作系统对硬件的需求，以保证在项目进行过程中可以保持在正确的方向上。此外，在整个设计过程中也将会对该文档进行部分的修改。

## 1.3 项目概览

本项目所需要实现的主要部分包括：CPU、MMU、通信等部分，项目将采用 VHDL 和 Verilog HDL 这两种语言实现，项目基于的软件是 Vivado2017.2.以下是项目所需要的各个部分的概览：

### 1.3.1 CPU

CPU 的总体设计主要包括指令系统、基本数据通路、流水线结构、异常与中断处理等。

**指令系统** UCORE 操作系统用到的指令有 47 条，其中算术指令 8 条，逻辑指令 8 条，移位指令 6 条，分支跳转指令 10 条，访存指令 5 条，移动指令 4 条，陷入指令 1 条，特权指令 5 条。该 47 条指令也是本次项目支持的 MIPS 32 指令集的子集。详细的内容我们将在附录中提到。

而 UBoot 相对 UCore 来说，需要多支持 11 条指令，这 11 条指令包含 2 条算数指令，5 条访存指令，1 条特权指令。此外也有 4 条与 Cache 相关的指令和 Watch Hi/ Lo 相关的机制，但是对于 UBoot 来说，这些指令并不需要实现，所以我们在最终的操作中，将这四条指令等效于 nop 指令进行处理。

**流水线** 计划实现的流水线为五级流水线，包括取指、译码、执行、访存、回写五个阶段，需要解决的冲突有数据冲突、结构冲突和控制冲突。

**异常与中断处理** 能够正确进行异常和中断处理，支持精确异常。

### 1.3.2 MMU

MMU 主要完成的功能为虚拟地址到物理地址的转换，除外，还需要进行地址到外设端口的映射等。

### 1.3.3 通信

通信部分主要包括串口通信，VGA 通信和键盘通信。

## 1.4 项目目标

项目主要需要达成的目标有以下三点：

- 能够设计实现 CPU、MMU、通信等功能
- 能够通过引动程序将 CPU 从 Flash 加载到 RAM 中，并顺利接受用户输入，执行相应功能。
- 逐步提高主频，并处理主频提高过程中遇到的一系列问题。

而项目的扩展目标也有以下三个方面：

- 能够支持 UBoot 和全部的功能测例
- 支持 USB、以太网和 HDMI 模块，能够进行图像和字符的输出，读取 U 盘等操作。

## 1.5 一些重要定义

以下是本文档及后续文档中用到的英文简称及其含义：

| 英文简称 | 含义 |
|---|---|
| MIPS | 无内部互锁流水级的微处理器 |
| CPU | 中央处理器 |
| CP0 | 协处理器 0 |
| ALU | 算术逻辑单元 |
| MMU | 内存管理单元 |
| PA | 物理地址 |
| VA | 虚拟地址 |
| TLB | 旁路快表缓冲 |
| BIOS | 基础输入输出系统 |
| ROM | 只读存储器 |
| RAM | 随机存取存储器 |
| SRAM | 静态随机存取存储器 |
| Flash | 快闪存储器 |

### 1.6 项目开发环境

#### 1.6.1 硬件环境

1. 开发板: THINPAD 2. RAM: 32 位 8MB 3. Flash: 32 位 8MB

#### 1.6.2 软件环境

- EDA 工具：Xilinx Vivado 2017.2
- 编译运行平台：Windows 10 x64
- 所需要支持的目标操作系统：ucore-thumips

## 2 UCORE 操作系统解析

### 2.1 简介

该部分主要分析操作系统中与硬件直接相关的部分，在接下来的内容中，我们首先简介 UCORE 操作系统及其编译、仿真方法。然后针对 UCORE 操作系统的 boot 阶段、内存布局和异常处理作深入分析。

UCORE 是一款类似 UNIX 的教学操作系统，本项目使用的是 UCORE 的 MIPS 移植版，称为 UCORE-THUMIPS。上述移植版的 GitHub 项目中含有一份简要文档。

### 2.2 准备工作

#### 2.2.1 编译

由于监控程序和 u-core 的编译较为类似，故这里忽略了监控程序的编译过程。

我们的做法是：

- 首先安装 mips-sde-elf 交叉编译器。并在 Linux 上为其配置了 32 位兼容。
- 将编译器的 bin/目录加入系统的$PATH 环境变量中。
- 执行 make ON_FPGA=y。（需要说明的是，因为监控程序有支持和不支持 TLB 的版本，在编译监控程序的过程中，我们没有设置 EN_TLB=y 来避免打开 tlb，因为需要支持 tlbr 和 tlbp 两条不属于需求的指令）。

### 2.3 Boot 阶段

Boot 阶段是操作系统启动阶段，在硬件进行初始化后，引导程序 BootLoader 从硬盘中加载操作系统到内存，然后将控制权交由操作系统，操作系统初始化相关状态，随后进行进程调度。整个阶段可以分为 3 部分:硬件初始化阶段，BootLoader 阶段，操作系统初始化阶段。

### 2.3.1 硬件初始化阶段

硬件接收到 Reset 信号后，根据 MIPS 标准，将进行如下初始化：

- 初始化 CP0 寄存器:根据我们的需要，这一阶段需要初始化的有 Random 和 Status 寄存器。
- 初始化 TLB:TLB 中有一位表示是否可以匹配的“隐藏”位，这一阶段需要将之置为禁止。
- Bus 状态机和配置初始化。
- PC 初始化:取值地址将从 VA 0xBFC00000 开始。

硬件完成初始化后，将从 VA 0xBFC00000 开始执行指令，进入 BootLoader 阶段。

### 2.3.2 BootLoader 阶段

### 2.3.3 操作系统初始化阶段

## 2.4 内存管理

### 2.4.1 内存管理器

### 2.4.2 MMU

## 2.5 异常处理

### 2.5.1 异常处理流程

### 2.5.2 异常类型

| Syscall 序号 | Syscall 说明 |
|---|---|
| 0 | sys_exit |
| 1 | sys_fork |
| 2 | sys_wait |
| 3 | sys_exec |
| 4 | sys_yield |
| 5 | sys_kill |
| 6 | sys_getpid |
| 7 | sys_putc |
| 8 | sys_pgdir |
| 9 | sys_gettime |
| 10 | sys_sleep |
| 11 | sys_open |
| 12 | sys_close |
| 13 | sys_read |
| 14 | sys_write |
| 15 | sys_seek |
| 16 | sys_fstat |
| 17 | sys_fsync |
| 18 | sys_getcwd |
| 19 | sys_getdirentry |
| 20 | sys_dup |

#### 2.5.3 ALU

# 3 UBoot 系统解析

## 3.1 简介

UBoot 是一个 Github 上的开源系统[1]，其主要作用是加载系统并启动 Kernel，以下内容主要也参考了该文档。

## 3.2 准备工作

### 3.2.1 编译

我们的编译过程使用了 mipsel-linux-gnu 工具链，与 UCore 和监控程序一致。此外，我们也需要修改 configs/naivemips_thinpad_defconfig 的配置：将 CONFIG_BAUDRATE 置为串口的波特率，CONFIG_DEBUG_UART_CLOCK 置为串口时钟频率（在我们的工程下，这一频率为 25,000,000)

### 3.2.2 UBoot 与 Windows 笔记本电脑直连

以下介绍实验板与 Windows 笔记本电脑直连的方法，如果是其他情况，也可以参考下面的做法。

- 将电脑与实验板用双绞线相连。如果笔记本电脑不支持双绞线，则需要一根交叉线。
- 运行 UBoot，这一步需要在 UBoot 中进行默认设置，需要设置的有：ipaddr 表示实验板的 IP 地址，默认是 192.168.1.60，serverip 表示 TFTP 服务器的地址，默认是 192.168.1.30，我们需要将笔记本与实验板相连的网卡设为此 IP，并将环境变量设为笔记本电脑该网卡的 IP。bootfile 表示需要被 boot 的镜像文件名，没有默认值，需要进行手动设置。
- 下载并安装 Windows 下的 TFTP 服务端，设置好要用于文件共享的文件夹。
- 编译 UCore，将专用于 u-boot 的镜像 obj/ucore.ub 复制到共享文件夹下。
- 在 u-boot 中运行 tftpboot，此命令会下载镜像，并将其加载到内存。
- 在 u-boot 中运行 bootm，进行 boot。

对于采用 USB Boot 的操作，需要注意以下两点：

- 准备好一个 FAT32 格式的 U 盘，我们建议，U 盘仅有一个分区，作为准备工作，需要将 ucore.ub 置于其根目录。
- 在 u-boot 中依次运行 USB start, fatls usb 0, run s_ucore, fatload usb 0, bootm 这些指令。

[1] https://github.com/u-boot/u-boot

### 3.2.3 对 UCore 的定制化

为了通过 uboot 对 ucore 进行 boot 操作，我们进行了两个改进：

- 将程序的起始地址移动到 0x80000040 以便为镜像头部腾出空间。
- 修改了 ucore 的 makefile 中 checkdir 的相关操作。

为了支持这一改动，需要安装新的依赖 mkimage。

# 4 基础部件

## 4.1 ALU

ALU 全称为算术逻辑单元(Arithmetic Logic Unit)，是能实现多种算术运算和逻辑运算的组合逻辑电路，也是 CPU 中的核心组成部分。例如，存储访问指令用 ALU 计算目标数据的地址，算术逻辑指令用 ALU 执行运算，而分支跳转指令用 ALU 进行比较。根据 ucore 操作系统的实际需求，我们的 ALU 只需要进行基本的整数算术逻辑运算即可，不需要支持浮点数运算。最后一章所示 UBoot 操作系统的 58 条基本指令中，包含了 10 条算术指令、6 条分支指令、8 条逻辑指令和 6 条移位指令。从这些指令出发，ALU 运算需求可以整理为下表。

具体实现需求中，ALU 接受两个 32 位整数与控制信号作为输入，以一个 32 位整数作为输出。很显然，ALU 根据不同的控制信号执行不同的操作。这些控制信号来源于指令当中的特定字段，在流水线结构的译码阶段生成。如果是算术逻辑指令，ALU 的结果将写回寄存器;如果是存取指令，ALU 的结果可作为读写寄存器的地址;如果是分支指令，ALU 的结果会被用于决定下一条指令地址。

## 4.2 乘法器

乘法器(Multiplier)是现代计算机 CPU 中必不可少的一部分，常见的乘法器模型基于“移位相加”算法实现。从乘数的最低位开始，若其为 1，则被乘数左移一位并与上一次的和相加;若为 0，则被乘数左移后以全零相加，如此循环至乘数的最高位。

MIPS32 指令集的乘法指令与其余算术逻辑指令略有差异。它接受两个 32 位整数为输入，以 64 位整数保存乘法运算结果，存储在 HILO 寄存器内。其中，HI 寄存器保存结果的高 32 位，LO 寄存器保存结果的低 32 位。这里运算结果的位数和存储位置发生了变化，因此我们需要将乘法器视为独立的运算单元，而不再是 ALU 的一部分。此外，由于我们所需要完成的内容并未涉及到 MADD、MADDU、MSUB、MSUBU 等乘法与加减法混合指令（虽然在 UBoot 反编译得到的文件中有对应的指令，但是实际上这些指令都是可以当作 nop 进行处理的），我们的乘法器只需要一个时钟周期就可以完成计算，从而避免了流水线调度的问题。

乘法器在具体实现中，主要用于执行 MULT、MULTU 两种指令，输入两个 32 位整数，将计算结果保存在 HILO 寄存器中。HILO 寄存器的读写操作，可以通过 MFLO、MFHI、MTLO 和 MTHI 四种指令完成。

| 操作码 | 功能 | 描述 |
|---|---|---|
| ADD | $A + B$ | 加法 |
| SUB | $A - B$ | 减法 |
| AND | $A\ and\ B$ | 按位与运算 |
| OR | $A\ or\ B$ | 按位或运算 |
| XOR | $A\ xor\ B$ | 按位异或运算 |
| NOT | $not\ A$ | 按位非运算 |
| SLL | $A\ sll\ B$ | 将 A 逻辑左移 B 位 |
| SRL | $A\ SRL\ B$ | 将 A 逻辑右移 B 位 |
| SRA | $A\ SRA\ B$ | 将 A 算数右移 B 位 |
| EQU | $A = B$ | 判断 A 是否等于 B |
| SLT | $A < B$ | 判断 A 是否小于 B |
| DIV | $A / B$ | 进行 A 除以 B 的操作 |

### 4.3 寄存器组

寄存器(Register)是 CPU 中的重要组成部分。它是容量大小有限的高速存储部件，可用作暂存指令、数据、地址等信息。依据存储内容的不同，寄存器可以分为整数寄存器、浮点寄存器、指令寄存器等;而依据设计用途的不同，寄存器也可以分为通用目的寄存器(General Purpose Registers, GPR)、程序计数器(Program Counter, PC)、堆栈寄存器(Stack Pointer, SP)、状态寄存器等。基本寄存器由 D 触发器构成，在 CP 脉冲信号的控制下，每到时钟上升沿时进行数据写入，其余时段数据保持不变。寄存器通常拥有非常高的读写速度，所以在寄存器之间传递数据效率很高，相比处理器缓存和主存储器而言有着显著的性能优势。

MIPS32 CPU 有 32 个通用目的寄存器，被命名为 031。各个寄存器的功能及汇编程序中的使用约定如下表所示:

| 寄存器号 | 寄存器名 | 功能 |
|---|---|---|
| \$0 | $zero$ | 常量 0 |
| \$1 | $a_t$ | 保留给汇编器使用 |
| \$2∼\$3 | $v_0 \sim v_1$ | 函数调用返回值 |
| \$4∼\$7 | $a_0 \sim a_3$ | 函数调用参数 |
| \$8∼\$15 | $t_0 \sim t_7$ | 调用者保存寄存器 |
| \$16∼\$23 | $s_0 \sim s_7$ | 被调用者保存寄存器 |
| \$24∼\$25 | $t_8 \sim t_9$ | 调用者保存寄存器 |
| \$26∼\$27 | $k_0 \sim k_1$ | 保留给异常处理使用 |
| \$28 | $gp$ | 全局指针(Global Pointer) |
| \$29 | $sp$ | 堆栈指针(Stack Pointer) |
| \$30 | $fp$ | 帧指针(Frame Pointer) |
| \$31 | $ra$ | 返回地址(Return Address) |

在采用 MIPS32 架构的硬件系统中，寄存器组内包括上述的 32 个通用目的寄存器，并且每个寄存器的位宽均为 32 位。寄存器内的值可以在一个时钟周期内的任意时间读出，但只能在时钟上升沿处更改。MIPS32 指令集中的 R 型指令(如算术逻辑指令、条件移动指令等)最多涉及三个不同寄存器的访问，并且规定从两个寄存器中读取数据，并将结果写入最后一个寄存器中。因此，寄存器组模块需要接收全局的时钟信号，支持对任意两个寄存器同时进行的读操作，和在时钟边沿处对某一个寄存器进行的写操作。此外我们注意到，并非所有指令都需要将结果写入寄存器，而且对 0 号寄存器的写入操作不符合寄存器功能要求，这意味着我们还需要为寄存器组添加必要的写使能信号，以规避潜在风险。

### 4.4 CP0

协处理器(Coprocessor)一词通常用来表示处理器的一个可选部件，通过扩展指令集或提供配置寄存器的方式来扩展内核处理功能。MIPS32 架构提供了最多 4 个协处理器，分别命名为 CP0 CP3，其功能如下表所示。协处理器可以借助一组专门的、提供装载或存储类型接口的 MIPS 指令来访问。

| 协处理器 | 功能 |
|---|---|
| CP0 | 控制 CPU，实现 MMU、异常处理、乘除法等功能 |
| CP1 | 浮点处理器(Floating Point Unit, FPU) |
| CP2 | 保留，各生产厂商用来实现自己的特色功能 |
| CP3 | 浮点处理器 |

实验用操作系统 ucore 没有实现浮点运算，因此我们不需要实现 CP1 和 CP3。CP2 没有特殊功能，同样不用实现。只有 CP0 是唯一不可选的协处理器，由于它涉及中断处理、提供可选配置、观察并控制系统缓存或时钟、地址转换等关键功能，我们必须加以足够重视。MIPS32 架构定义的协处理器 CP0 所负责的主要工作如下。

Table 1: CP0 Specification

| 寄存器号 | Select 段编号 | 寄存器名 | 功能 |
|---|---|---|---|
| 0 | 0 | Index | TLB 阵列的入口索引 |
| 1 | 0 | Random | 提供 tlbwr 指令所需的随机数位置 |
| 2 | 0 | EntryLo0 | 偶数虚拟页入口地址的低 32 位部分 |
| 3 | 0 | EntryLo1 | 奇数虚拟页入口地址的低 32 位部分 |
| 4 | 0 | Context | 提供页表所在的位置 |
| 5 | 0 | PageMask | 提供与 TLB 相关的 PageMask 功能 |
| 6 | 0 | Wired | 提供 Wired TLB Entries 和 Random TLB Entries 的分隔 |
| 8 | 0 | BadVAddr | 记录最近一次存储发生异常时的虚拟地址 |
| 9 | 0 | Count | 与 Compare 寄存器组成片内计时器，两者相等时发出时钟中断信号 |
| 10 | 0 | EntryHi | TLB 入口地址的高 32 位部分 |
| 11 | 0 | Compare | 与 Count 寄存器组成片内计时器，两者相等时发出时钟中断信号 |
| 12 | 0 | Status | 处理器状态和控制寄存器，决定 CPU 特权等级和中断使能等 |
| 13 | 0 | Cause | 保存最近一次异常原因 |
| 14 | 0 | EPC | 保存最近一次异常时的程序计数器 |
| 15 | 0 | Identification | 保存处理器的相关信息，如公司等 |
| 15 | 1 | Ebase | 保存异常处理程序的入口地址 |
| 16 | 0 | Config | 提供处理器的 Config 等相关操作 |
| 16 | 1 | Config1 | 保存 MMU size，Cache size 等相关内容 |
| 16 | 2 | Config2 | 保存 L2 和 L3 Cache 等的 Config 相关内容 |
| 16 | 3 | Config3 | 提供一些额外的配置信息 |
| 17 | 0 | Load Linked | 存储最近一次读取指令的地址 |
| 18 | N/A[2] | WatchLo | 与 WatchHi 寄存器共同支持 watch point debug 相关内容 |
| 19 | N/A | WatchHi | 与 WatchLo 寄存器共同支持 watch point debug 相关内容 |
| 23 | N/A | Debug | 提供与 EJTAG 相关的操作[3] |
| 24 | N/A | DEPC | 提供与 EJTAG 相关的操作 |

[2]任何值均可，下同

[3]EJTAG 相关操作需要参见 EJTAG 相关的文档

| 寄存器号 | Select 段编号 | 寄存器名 | 功能 |
|---|---|---|---|
| 25 | N/A | Performace | 提供与评估系统表现相关的操作 |
| 26 | N/A | ErrCtl | 提供与错误检查相关的内容 |
| 27 | N/A | CacheErr | 提供与 Cache 错误检查相关的内容 |
| 28 | 0, 2 | TagLo | 与 TagHi 寄存器共同提供与 Cache Tag Array 相关的操作 |
| 28 | 1, 3 | DataLo | 与 DataHi 寄存器提供与 Cache Tag Array 相关的检错操作 |
| 29 | 0, 2 | TagHi | 与 TagLo 寄存器共同提供与 Cache Tag Array 相关的操作 |
| 29 | 1, 3 | DataHi | 与 DataLo 寄存器提供与 Cache Tag Array 相关的检错操作 |
| 30 | 0 | ErrorEPC | 提供错误异常返回地址 |
| 31 | N/A | DESAVE | 提供与 EJTAG 相关的操作 |

而对于 UBoot 而言，我们需要支持的寄存器有：Index, Random, EntryLo0, EntryLo1, BadVAddr, Count, EmtryHi, Compare, Status, Cause, EPC, EBase, WatchLo, WatchHi 这 14 个寄存器，其中，WatchHi 寄存器和 WatchLo 寄存器的相关功能可以不进行实现。因此，我们将重点对这 12 个寄存器的功能进行说明。

从表中不难看出，这 12 个寄存器大多与 MMU、TLB 和异常处理相关。以下部分将介绍这 12 个寄存器的具体功能和实现需求。

- **Index 寄存器**：用来存储 TLBWI 指令向 TLB 中写的 Entry ID。
- **Count 寄存器**：计数用寄存器，计数频率与 CPU 计数频率相同，达到计数上限之后不再进行累加操作。
- **Random 寄存器**：初始为 TLB Entries 的个数（一般满足 $2^n - 1$ 的格式），之后每一个 CPU 时钟周期进行-1 的操作，当减少至 0，则恢复至 TLB Entries 的个数。
- **EntryLo0 寄存器**：在 TLB 异常时使用，用来存储 TLB EntryLo0 将被填写的数值。
- **EntryLo1 寄存器**：在 TLB 异常时使用，用来存储 TLB EntryLo1 将被填写的数值。
- **EntryHi 寄存器**：在 TLB 异常时使用，用来存储 TLB EntryHi 将被填写的数值。
- **BadVAddr 寄存器**：在 TLB 发生异常时，被置为导致 TLB miss 的虚拟地址。
- **Compare 寄存器**：与 Count 寄存器配合使用。
- **EPC 寄存器**：用来记录异常处理的返回地址。
- **EBASE 寄存器**：用来存储异常向量的基地址（最终的地址为 EBase + Offset）。

- **Status 寄存器**：用来控制处理器的操作模式，中断使能及诊断状态。

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 标志名 | CU3∼CU0 | | | | RP | R | RE | 0 | | BEV | TS | SR | NMI | 0 | | |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 标志名 | IM7∼IM0 | | | | | | | | R | | | UM | R | ERL | EXL | IE |

上表中表示为 R 的字段是保留字段。比较重要的非保留字段介绍如下：

- **CU3-CU0**：表示协处理器 CP3 ~ CP0 是否可用。
- **BEV**：表示是否启动异常向量。
- **TS**：表示是否关闭 TLB。
- **SR**：表示是否为软重启，如果为 1 则表示软重启。
- **NMI**：表示是否为可屏蔽中断，为 1 表示重启异常是由不可屏蔽中断造成的。
- **IM7-IM0**：表示是否屏蔽相应的中断，MIPS 处理器的 8 个中断源对应的 IM 字段的 8 位。其中有 6 个中断是外部硬件中断，剩下的两个中断是软件中断。
- **UM**：表示是否为用户模式，为 1 表示处理器运行在内核模式，为 0 表示处理器运行在用户模式。
- **EXL**：表示是否处于异常级，当异常发生时，本字段置 1。
- **IE**：表示是否使能中断，这是全局中断使能标志位。为 1 表示中断使能，为 0 表示中断禁止。

- **Status 寄存器**

| Bit | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 标志名 | BD | R | CE | | DC | PCI | 0 | | IV | WP | 0 | | | | | |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 标志名 | IP7∼IP2 | | | | | | IP1∼IP0 | | 0 | ExcCode | 0 | | | | | |

上表中表示为 R 的字段是保留字段。比较重要的非保留字段介绍如下：

- **BD**:当发生的异常指令处于分支延迟槽时，该字段置 1。
- **IP7∼IP2**：中断挂起字段，相应位用来指明外部硬件中断是否可用。
- **IP1∼IP0**：中断挂起字段，对应软件中断。
- **ExcCode**：用来记录发生了哪些异常，UBoot 中未涉及异常，而 UCore 涉及到的异常种类如下表所示：

| ExcCode | 助记符 | 描述 |
|---|---|---|
| 0 | Int | 中断 |
| 1 | Mod | TLB 加载异常或保留 |
| 2 | TLBL | TLB 加载异常、取指异常或保留 |
| 3 | TLBS | TLB 存储异常或保留 |
| 4 | AdEL | 加载或取指过程中，地址错误异常 |
| 5 | AdES | 存储过程中，地址错误异常 |
| 8 | Sys | 系统调用指令 SYSCALL |
| 10 | RI | 执行未定义指令引起的异常 |
| 11 | CpU | 协处理器不可用异常 |
| 12 | Ov | 整数溢出异常 |

### 4.5 MMU

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、物理存储器的控制模块，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。我们知道，计算机的虚拟地址空间大小由 CPU 位数所决定，对于一个 32 位的 CPU，地址范围为 0-0xFFFFFFFF，总大小为 4G。而实际物理地址往往只是虚拟地址空间的子集，因此在具备 MMU 的计算机上，虚拟地址将通过 MMU 模块转化为物理地址。

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 标准的内存映射方案如下表所示:

| 段 | 虚拟地址 | 权限 | 物理地址 |
|---|---|---|---|
| kuseg | 0x00000000~0x7FFFFFFF | 用户态 | TLB |
| kseg0 | 0x80000000~0x9FFFFFFF | 内核态 | 0x00000000~0x1FFFFFFF |
| kseg1 | 0xA0000000~0xBFFFFFFF | 内核态 | 0x00000000~0x1FFFFFFF |
| kseg2 | 0xC0000000~0xFFFFFFFF | 内核态 | TLB |

在本次实验中，整体的内存映射方案如上表所示，由于开发板上的 SRAM 不能覆盖 kseg0 和 kseg1 段的物理地址空间，因此该段空间实际有效的地址范围是:VA 0x80000000 - 0x800FFFFF(对应 PA 0x0000000-0x000FFFFF). 此外，kseg1 段的部分地址将不被映射到物理内存地址，而是映射到特殊设备，具体见下表:

| 虚拟地址 | 映射目标 说明 | |
|---|---|---|
| 0xBE000000~0xBEFFFFFF | Flash | 存放操作系统 |
| 0xBFC00000~0xBFC00FFF | ROM | 存放 BootLoader |
| 0xBFD003F8~0xBFD003FC | 串口 | 通信 |

## 4.6 TLB

旁路快表缓冲(Translation Lookaside Buffer, TLB)，简称快表，可以理解为一种地址变换高速缓存。

在上一个部分中，计算机内部虚拟地址与实际物理地址之间的对应关系通常由一种特殊的数据结构——页表(Page Table)进行储存。因此，虚拟内存到物理内存的转换可以通过查表操作进行。由于页表存放在主存中，程序每次访问内存至少需要两次:一次访存获取物理地址，第二次访存才获得数据。提高访存性能的关键在于依靠页表的访问局部性。当一个转换的虚拟页号被使用时，它可能在不久的将来再次被使用到。TLB 就是符合这些要求的一种高速缓存，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。

CPU 访问虚拟地址时，会首先根据虚拟地址的高 20 位在 TLB 中查找。如果快表中没有相应的表项，则触发页表缺失异常(TLB miss)，需要通过访问内存中的页表计算出相应的物理地址。与此同时，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，称为页表命中(TLB hit)。

结合操作系统 ucore 及 THINPAD 教学实验平台的实际情况，我们最终确定将 TLB 条目数设置为 16。虚拟地址的高 20 位作为虚页号，用于在 TLB 中查找对应的页表项，返回实际物理地址。

## 4.7 异常及中断处理

MIP32 架构中的异常包括中断(Interrupt)、陷阱(Trap)、系统调用(System Call)以及其它任何可以打断程序正常执行流程的操作。在 ucore 异常处理一节和 CP0 一节的 ExcCode 表中，我们已经列举了操作系统 ucore 涉及的异常，而下表则进一步按照优先级对 MIPS CPU 所需处理异常类型进行了排序。

检测到异常发生之后，CPU 会执行一系列指令以处理异常。MIPS32 架构的异常处理过程如下:

MIPS CPU 有 8 个独立的中断位(在 Cause 寄存器中)，其中 6 个为外部硬件中断，2 个为软件中断。具体实现方面，CPU 能够处理的中断类型如下表所示。遇到中断时，CPU 将跳转至通用的异常处理程序入口。

对可能出现的异常，其优先级排序如下：

| 优先级 | 异常 | 描述 |
|---|---|---|
| 1 | Reset | 硬件复位 |
| 2 | Soft Reset | 发生致命错误后对系统进行恢复位 |
| 3 | NMI | 不可屏蔽中断 |
| 4 | Interrupt | 监测到八个中断之一 |
| 5 | AdEL | 取址地址对齐异常 |
| 6 | TLB Refill | TLB 缺失 |
| 7 | TLB Invalid | TLB 无效 |
| 8 | Sys | 调用 Syscall |
| 9 | RI | 无效指令 |
| 10 | Ov | 算数指令运算溢出 |
| 11 | AdEL | 加载数据的地址未对齐 |
| 12 | AdES | 存储数据的地址未对齐 |
| 13 | TLB Refill | 数据 TLB 缺失 |
| 14 | TLB Invalid | 数据 TLB 无效 |

监测到异常发生之后，CPU 会执行一系列的指令用来处理异常，MIPS32 架构的异常处理过程如下：

- 检测 CP0 中 Status 寄存器的 EXL 字段，如果 EXL 为 0，将异常原因保存到 CP0 协处理器 Cause 寄存器的 ExcCode 寄字段。
- 检查发生异常的指令是否在延迟槽中，如果在则设置 EPC 寄存器的值为指令地址减 4，Cause 寄存器的 BD 字段为 1；否则设置 EPC 寄存器的值为指令地址，Cause 寄存器的 BD 字段为 0。
- 设置 Status 寄存器的 EXL 字段为 1，表示进入内核态处理异常，禁止中断。
- 处理器根据异常种类，转移到相应异常处理例程的入口地址，运行异常处理程序。
- 处理结束，调用异常返回指令 ERET 转移到异常发生前的状态，ERET 指令会清除 Status 寄存器的 EXL 字段，并且将 EPC 寄存器的值复制回 PC 中。

# 5 流水线结构

## 5.1 流水线概述

流水线是指将计算机指令处理过程拆分为多个步骤，并通过多个硬件处理单元并行执行来加快指令执行速度。大多数现代处理器选择采用流水线方式执行指令。通常，一条 MIPS 指令包含如下五个处理步骤:

- Instruction Fetch: 从指令存储器中读取指令。
- Instruction Decomposition: 指令译码的同时读取寄存器.

- Execution: 执行操作或计算地址。
- Memory Access: 从数据存储器中读取操作数。
- Write Back: 将结果写回寄存器。

MIPS CPU 按照这五个处理步骤，建立了五级流水线结构。它的加速原理可用一个简单的公式说明。我们假设所有的流水级都只花费 t 的时间，该时间能够满足最慢操作的执行需要。在单周期、非流水线模型中，任意 n 条指令所需的执行时间均为 tnp = 5n t;而在五级流水线模型中，执行 n 条指令所用的时间为 tp = (n + 4) t。那么采用流水线结构 CPU 的性能加速比为:

也就是说，在指令数量足够大的情况下，流水线所带来的加速比近似等于流水线级数。这里五级流水线结构的加速比接近于 5。考虑到实际情况当中指令数目有限，以及各流水级所需要的时间不尽相同，对于本实验来说，MIPS32 架构的 CPU 大约可带来 4 倍左右的实际性能提升。

总而言之，流水线提升性能的方法是通过增加指令的吞吐率，而非减少单条指令的执行时间。因此执行程序时，指令的吞吐率是一个很重要的性能参数。

## 5.2 数据通路

五级流水线结构的设计，意味着任何一个单时钟周期内，CPU 最多会执行 5 条指令。因此必须把 CPU 数据通路划分为 5 个部分，每部分用相对应的指令执行阶段来进行命名:

- **取指阶段**：从指令存储器中读出指令，确定下一条指令地址。
- **译码阶段**：对去除的指令进行译码，从通用存储器中读出要使用的寄存器值，如果指令含有立即数，需要将立即数进行符号扩展或无符号扩展，如果指令时是跳转指令，需要给出跳转指令的目标地址。
- **执行阶段**：按照译码阶段给出的操作数和运算类型，进行运算并给出结果，如果是访存类指令，还会给出目标地址。
- **访存阶段**：如果是访存类指令，那么此阶段会访问数据存储器，否则只是将执行阶段的运算结果向下传递至回写阶段。同时，此阶段若发生异常，需跳转到异常处理例程的入口地址。
- **回写阶段**：将运算结果保存值目标寄存器。

除去这五个阶段外，我们还需要增加 IF-ID, ID-EX, EX-MEM, MEM-WB 这四个阶段，其作用是存储前一阶段的运算结果，并在上升沿将结果传递给下一阶段，进而保证程序的鲁棒性。

程序执行过程中，指令与数据由上而下地顺序通过五级流水线。通过增加保存中间数据的寄存器，指令执行过程中可以实现对数据通路的共享。然而数据通路中同样存在一些例外。比如，写回阶段需要把结果写回数据通路中间的寄存器堆中;程序计数器的下一个取

值，需在自增的 PC 和 ID 阶段产生的分支目标之间选择。第一个例外可能导致数据冒险，第二个例外可能导致控制冒险。关于流水线冒险，下一节将给出具体说明和解决方案。

## 5.3 冒险及解决方案

### 5.3.1 流水线冒险及其类型

由之前的分析，我们可以看到，流水线虽然显著的提升了效率，但是同时也会导致不同的阶段访问同一资源时产生问题。首先，冒险分为以下的三种类型：

- **结构冒险**：指令执行过程中，由于多条指令对硬件资源的使用产生冲突而导致的冒险，比如指令和数据共享同一个寄存器，在单时钟周期内，访存操作和取指操作会发生存储器访问冲突导致结构冒险。
- **数据冒险**：流水线内部的几条指令中，一条指令依赖于先前指令的执行结果，比如当前译码阶段的指令需要读取上一条指令即将写回寄存器的值，这与流水线的时间顺序矛盾。
- **控制冒险**：流水线中的分支跳转指令在 ID 阶段才能确定目标地址，但取指过程中每时钟周期必须进行，这种为确保取得正确指令而铲产生的延迟成为控制冒险。

MIPS CPU 主要处理的内容为：数据冒险和控制冒险。

### 5.3.2 消除数据冒险

对于相邻或相隔一条指令存在的数据冒险，MIPS CPU 采用如下两种解决方法：

- **流水线暂停**：检测到数据相关时，通过重置流水线寄存器的方式，插入一些空指令周期（或称作“气泡”），暂停流水线的运行。
- **数据旁路**：通过额外的信号，将 EX 阶段和 MEM 阶段得到的结果回送回 ID 阶段。

### 5.3.3 规避控制冒险

分支转移指令或者其它指令修改了 PC 的值，导致已经进入流水线的指令无效，从而引发控制冒险。MIPS CPU 采用分支延迟槽(Branch Delay Slot)来减小控制冒险的代价。

我们规定分支指令后面的指令位置为分支延迟槽，延迟槽内的指令称为“延迟指令”。延迟指令在计算分支目标地址时已经进入流水线，因此它总是被执行，与跳转发生与否没有关系。为保证延迟槽中只有一条指令，分支判断需要在译码阶段完成，MIPS CPU 会通过额外的计算单元实现这一功能。很显然，并非所有的指令都可以放入延迟槽。部分编译器能够对指令顺序进行调整，在结果不变的情况下使用延迟指令来优化性能。但大多数情况下，延迟指令由程序员依据实际情况设置。

需要注意，如果延迟指令导致异常，那么进入异常处理程序之前，CP0 协处理器的 EPC 寄存器应保存之前的分支跳转指令地址，而不是该延迟指令地址。

## 5.4 功能需求

### 5.4.1 功能测例的功能需求

对于功能测例而言，我们需要实现以下的一些功能：

- 首先，支持最后一部分中提到的全部的 57 条指令，并且需要支持表格中所提到的异常。
- 能够正确解决结构冒险、数据冒险和控制冒险。
- 需要自己编写框架，能够对功能测例进行集成，进而进行仿真和硬件测试。
- 对已有的功能测试框架进行补充，添加针对 tlbwr 和 tlbwi 两条指令的功能测例。

### 5.4.2 监控程序的功能需求

对于监控程序而言，我们所需要实现的功能如下：

- 能够支持 ucore 所需要支持的全部指令（对于带 TLB 的监控程序，还需要能够支持 tlbr 和 tlbp 两条指令）。
- 能够支持中断，通过串口首发数据。

### 5.4.3 UCore 的功能需求

对于 UCore 而言，流水线所需要实现的功能总结如下：

- 按照五级流水线结构，建立数据通路，划分指令执行阶段。
- 设置流水线寄存器，保存每个流水级的数据和控制信号，在时钟上升沿处传递至下一级。
- 利用使能信号，对寄存器堆、指令寄存器和数据存储器进行读写控制。
- 设置多路选择器，按照译码结果或其他控制信号选择 ALU、PC 和存储器等的数据来源。
- 检测数据冒险，实现流水线的暂停和数据旁路。
- 添加额外的运算器，在译码阶段完成分支判断和目标地址计算，
- 检测异常发生并调用异常处理程序，并且能够识别发生在延迟槽内的异常指令。
- 整合 TLB、CP0、中断控制等其他功能。

### 5.4.4 UBoot 的功能需求

- 能够支持 57 条指令，这些指令几乎是 C++ 编译器所可能产生的全部指令。
- 需要实现以太网模块、USB 模块中的至少一个，用以检验 UBoot 的正确性。
- 对 UCore 进行一定的定制化，最终目的是保证在通过 UBoot 进行完 UCore 的 Boot 操作之后，可以通过 reset 正确的返回 Uboot。
- 需要正确的实现 CP0 中的相关机制，可以通过 CPU 进行简单的记时操作。

### 5.5 性能需求

流水线结构的性能瓶颈取决于最慢的流水级。考虑到 THINPAD 教学实验平台上的 RAM 读取速度为 50MHz，各阶段引入的控制单元存在延时，MIPS CPU 基于结果正确、运行稳定的原则，通过不断测试以提升时钟频率。其中，可以实现的频率的理论上限是 100MHz，但由于 100MHz 已经接近门延迟，所以一般认为，其可以达到的频率上限为 50MHz。

综合访存等考虑，我们组最终选择了 25MHz 作为 CPU 的时钟频率。

外设方面，键盘输入将用软件中断方式实现，VGA 信号通过串口输出并且由硬件直接处理，从而尽可能减小对性能的影响。

此外，为进一步优化性能，我们将对 CPU 各个逻辑部件的时序做深入分析，努力逼近和提升 CPU 频率上限。

## 6 存储器和外围设备

### 6.1 SRAM

SRAM(Static Random Access Memory)，即静态随机存取存储器，是一种具有静止存取功能的内存，不需要刷新电路就可以保存内部数据。SRAM 的优点是速度快，不需要周期性刷新内存数据; 缺点是集成度低，掉电后数据会丢失，相同容量下的体积和功耗较大，价格较高。因此 SRAM 一般少量用于关键性系统中，以提高运行效率。

THINPAD 教学计算机设置了两片 256×1024×16b 的 SRAM 作为内存，其访问过程可通过设计状态机和内存读写逻辑而实现。需要注意的是，教学实验平台上的内存芯片 RAM1 和串口共同连接在一条总线上，访问 SRAM 的前提是串口不工作，这可以通过调节读写控制信号加以实现。

对存储器的访问是 CPU 流水线非常关键的一级。SRAM 读写时序的优化，可以使得整个流水线数据通路的设计得到简化，有助于提升 CPU 频率和增强 CPU 设计的鲁棒性。不得当的读写时序设计，将给 CPU 的调试运行带来各种难以定位和解决的麻烦。比如依据流水线结构一节的叙述，同时进行取指和访存操作将导致存储器结构冒险。为解决这一问题，我们必须更加精细地设计 SRAM 访问时序，或是实现指令和数据分立存储。不仅如此，由于 RAM 运行频率成为限制 CPU 主频的瓶颈，CPU 性能提升也需要通过优化访存操作或改善存储器体系结构来完成。

### 6.2 Flash 存储器

Flash 存储器又称为闪存，它不仅具备电子可擦除和可编程的功能，还能够快速读取数据。Flash 不像 SRAM，保存的数据断电后也不会丢失。由于这些特点，它在便携式设备中被大量使用，如手机、平板电脑、U 盘和 SD 卡等。

THINPAD 教学实验平台上用到的 Flash 型号为 28F640J3，涉及操作主要有读、写和擦除。本实验的具体需求中，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。操作系统启动之前，写入 FPGA 的 BootLoader 程序将会把 Flash 内部的操作系统代码转移至 SRAM。下表列出了和 Flash 操作相关的部分控制信号。

| 信号名称 | 描述 |
|---|---|
| CE0、CE1、CE2 | 使能信号，本实验中仅控制 CE0 |
| BYTE | 操作模式，置 1，表示字模式 |
| VPEN | 写保护，置 1，表示写保护开启 |
| RP | 重置信号，置 1，表示工作 |
| OE | 读使能信号，0 有效 |
| WE | 写使能信号，0 有效 |
| ADDR[22:0] | 23 位地址线，字模式下使用 ADDR[22:1] |
| DATA[15:0] | 16 位数据线，读写操作共用 |
| SR[7:0] | 8 位状态寄存器，与 DATA[7:0] 共用 |

## 6.3 串行接口

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。本实验中，RS232 接口将连接至 PC 上的串口，实现 PC 和教学计算机之间的程序通信。串口的功能需求具体表现在:

- 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。开始运行时，BootLoader 负责将这些代码转移至内存。
- 调试阶段，我们借助调试器，通过串口向计算机发送调试命令，并接收计算机传回的调试信息，如寄存器状态、内存数据、运行提示信息等。

## 6.4 VGA 设备

VGA(Video Graphics Array)是一种标准的显示接口，在现实中得到广泛应用。本实验所使用的 VGA 设备为光栅扫描显示器，分辨率为 640×480 像素，屏幕刷新率为 60Hz。MIPS CPU 中的 VGA 模块接受 ASCII 码为输入，将 ASCII 码转换为对应的图形点阵，并通过 VGA 接口输出至屏幕上的字符终端。VGA 模块所需实现的具体功能如下:

- 固定 VGA 显示器的分辨率和屏幕刷新率，输出正确的行场像素数据和同步信号，保持图像稳定。• 支持全部 95 个可显示的 ASCII 字符。
- 支持输入行光标的显示和移动。
- 支持屏幕平滑滚动和翻页控制(Page Up/Down)。全部 95 个 ASCII 字符的像素矩阵会事先存储在计算机的 ROM 区域中。在输出信号正确且稳定的基础上，VGA 模块

将通过接口调用的方式，提供对光标、滚动和翻页等功能的支持。

### 6.5 PS2 键盘

本实验对于支持键盘输入的需求主要表现为:

- 添加特定硬件模块，对键盘时钟信号和数据信号进行解析，得到输入的 ASCII 码值，随后引发 CPU 硬件中断，将 ASCII 码值传递至操作系统处理。
- 为操作系统添加对键盘中断信号的支持，使其能够接受来自键盘的输入并进行处理。具体实现中，可在被配置为 UART 的 CPLD 芯片中对键盘信号进行预处理，将提取出的 ASCII 码直接传递给 FPGA 芯片内相应模块。该预处理电路接收来自键盘的串行数据，依据状态机提取扫描码，最后对键盘数据进行译码和输出

## 7 工具

### 7.1 设备自动化检测工具

### 7.2 调试工具

## 8 UCORE 需要的 46 条指令

### 8.1 算数指令

| **指令** | $ADDIU\ rt, rs, immediate$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 001001 | | | | | | rs | | | | | rt | | | | |
| **格式(31-16)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | immediate | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= R[s] + SignedExtend(immediate)$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| **指令** | $ADDU\ rd, rs, rt$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 100001 | | | | | |
| **操作** | $R[d] <= R[s] + R[t]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $MULT\ rs, rt$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 0000000000 | | | | | | | | | | 011000 | | | | | |
| **操作** | $(LO, HI) <= R[s] * R[t]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $SLT\ rd, rs, rt$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 101010 | | | | | |
| **操作** | $R[d] <= R[s] < R[t]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $SLTI\ rt, rs, immediate$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 001010 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | immediate | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= R[s] < SignedExtend(immediate)$ | | | | | | | | | | | | | | | |
| **其他** | 符号数比较 | | | | | | | | | | | | | | | |

| 指令 | $SLTIU\ rt, rs, immediate$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 001011 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | immediate | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= R[s] < SignedExtend(immediate)$ | | | | | | | | | | | | | | | |
| **其他** | 无符号数比较 | | | | | | | | | | | | | | | |

| 指令 | $SLTU\ rd, rs, rt$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 001011 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 101011 | | | | | |
| **操作** | $R[t] <= R[s] < R[t]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $SUBU\ rd, rs, rt$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 100011 | | | | | |
| **操作** | $R[t] <= R[s] - R[t]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

## 8.2 逻辑指令

| 指令 | $AND\ rd, rs, rt$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 100100 | | | | | |
| **操作** | $R[t] <= R[s] \& R[t]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $ANDI\ rt, rs, immediate$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 001100 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | immediate | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= R[s] < ZeroExtend(immediate)$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $LUI\ rt, immediate$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 001100 | | | | | | 00000 | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | immediate | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= immediate \|\| 0^{16}$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $NOD\ rd, rs, rt$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 100111 | | | | | |
| **操作** | $R[t] <= \sim (R[s] \| R[t])$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $OR\ rd, rs, rt$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 100101 | | | | | |
| **操作** | $R[t] <= (R[s] \| R[t])$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $ORI\ rt, rs, immediate$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 001101 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | immediate | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= (R[s] \| ZeroEctend(immediate))$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $XOR\ rd, rs, rt$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(15-0)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 001110 | | | | | | rs | | | | | rt | | | | |
| **格式(31-16)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 100110 | | | | | |
| **操作** | $R[t] <= (R[s]^R[t])$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $XORI\ rt, rs, immediate$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 001110 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | immediate | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= (R[s]^{Z}eroExtend(immediate))$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

## 8.3 移位指令

| 指令 | $SLL\ rd, rs, sa$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | 00000 | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | sa | | | | | 000000 | | | | | |
| **操作** | $R[d] <= (R[t] << sa)$ | | | | | | | | | | | | | | | |
| **其他** | sa 解释为无符号数 | | | | | | | | | | | | | | | |

| 指令 | $SLLV\ rd, rs, rs$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 000100 | | | | | |
| **操作** | $R[d] <= (R[s] << R[s]\{4-0\})$ | | | | | | | | | | | | | | | |
| **其他** | R[s]{4-0}解释为无符号数 | | | | | | | | | | | | | | | |

| 指令 | $SRA\ rd, rs, sa$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | 00000 | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | sa | | | | | 000011 | | | | | |
| **操作** | $R[d] <= (R[t] >> sa)$ | | | | | | | | | | | | | | | |
| **其他** | sa 解释为无符号数 | | | | | | | | | | | | | | | |

| 指令 | $SRAV\ rd, rt, rs$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 000111 | | | | | |
| **操作** | $R[d] <= (R[s] >>_A R[s]\{4-0\})$ | | | | | | | | | | | | | | | |
| **其他** | R[s]{4-0}解释为无符号数 | | | | | | | | | | | | | | | |

| 指令 | $SRL\ rd, rs, ra$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | 00000 | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | ra | | | | | 000010 | | | | | |
| **操作** | $R[d] <= (R[s] >>_L R[s]\{4-0\})$ | | | | | | | | | | | | | | | |
| **其他** | sa 解释为无符号数 | | | | | | | | | | | | | | | |

| 指令 | $SRLV\ rd, rt, rs$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 000110 | | | | | |
| **操作** | $R[d] <= (R[s] >>_L R[s]\{4-0\})$ | | | | | | | | | | | | | | | |
| **其他** | R[s]{4-0}解释为无符号数 | | | | | | | | | | | | | | | |

## 8.4 分支跳转指令

| 指令 | $BEQ\ rt, rs, offset$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000100 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $if\ R[s] = R[t]\ then\ PC <= PC + SignExtend(offset\|\|00))$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $BGEZ\ rs, offset$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000001 | | | | | | rs | | | | | 00001 | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $if\ R[s] \geq 0\ then\ PC <= PC + SignExtend(offset\|\|00))$ | | | | | | | | | | | | | | | |
| **其他** | 符号数 | | | | | | | | | | | | | | | |

| 指令 | $BGTZ\ rs, offset$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000111 | | | | | | rs | | | | | 00000 | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $if\ R[s] > 0\ then\ PC <= PC + SignExtend(offset\|\|00))$ | | | | | | | | | | | | | | | |
| **其他** | 符号数 | | | | | | | | | | | | | | | |

| 指令 | $BLEZ\ rs, offset$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000110 | | | | | | rs | | | | | 00000 | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $if\ R[s] \leq 0\ then\ PC <= PC + SignExtend(offset\|\|00))$ | | | | | | | | | | | | | | | |
| **其他** | 符号数 | | | | | | | | | | | | | | | |

| 指令 | $BLTZ\ rs, offset$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000001 | | | | | | rs | | | | | 00000 | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $if\ R[s] < 0\ then\ PC <= PC + SignExtend(offset\|\|00))$ | | | | | | | | | | | | | | | |
| **其他** | 符号数 | | | | | | | | | | | | | | | |

| 指令 | $BNE\ rs, rt, offset$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000001 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $if\ R[s] \neq R[t]\ then\ PC <= PC + SignExtend(offset\|\|00))$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $J\ instr\_index$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000010 | | | | | | instr_index | | | | | | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | instr_index | | | | | | | | | | | | | | | |
| **操作** | $PC <= PC\{31-28\}\|\|instr\_index\|\|00$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $JAL\ instr\_index$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000011 | | | | | | instr_index | | | | | | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | instr_index | | | | | | | | | | | | | | | |
| **操作** | $R[31] <= PC+8; PC <= PC\{31-28\}\|\|instr\_index\|\|00$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $JALR\ rs(rd=31) or\ JALR\ rd,rs$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | 00000 | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000 | | | | | 001001 | | | | | |
| **操作** | $R[d] <= PC+8; PC <= R[s]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $JR\ rs$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | 00000 | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 00000 | | | | | 00000 | | | | | 001000 | | | | | |
| **操作** | $PC <= R[s]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

## 8.5 访存指令

| 指令 | $LB\ rt, offset(base)$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 100000 | | | | | | base | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= SignedExtend(MB[base + SignExtend(offset)])$ | | | | | | | | | | | | | | | |
| **其他** | 可能的异常：TLB Refill; TLB Invalid; Address Error | | | | | | | | | | | | | | | |

| 指令 | $LBU\ rt, offset(base)$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 100100 | | | | | | base | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= ZeroExtend(MB[base + SignExtend(offset)])$ | | | | | | | | | | | | | | | |
| **其他** | 可能的异常：TLB Refill; TLB Invalid; Address Error | | | | | | | | | | | | | | | |

| 指令 | $LW\ rt, offset(base)$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 100011 | | | | | | base | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= MW[base + SignExtend(offset)]$ | | | | | | | | | | | | | | | |
| **其他** | 可能的异常：TLB Refill; TLB Invalid; Address Error | | | | | | | | | | | | | | | |

| 指令 | $SB\ rt, offset(base)$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 101000 | | | | | | base | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= M[base + SignExtend(offset)] <= R[t]\{7-0\}$ | | | | | | | | | | | | | | | |
| **其他** | 可能的异常：TLB Refill; TLB Invalid; TLB Modified; Bus Error; Address Error | | | | | | | | | | | | | | | |

| 指令 | $SW\ rt, offset(base)$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 101011 | | | | | | base | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $R[t] <= M[base + SignExtend(offset)] <= R[t]$ | | | | | | | | | | | | | | | |
| **其他** | 可能的异常：TLB Refill; TLB Invalid; TLB Modified; Address Error | | | | | | | | | | | | | | | |

## 8.6 移动指令

| 指令 | $MFHI\ rd$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 格式(31-16) | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | 0000000000 | | | | | | | | | |
| 格式(15-0) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | | 00000 | | | | 010000 | | | | | |
| 操作 | $R[d] <= HI$ | | | | | | | | | | | | | | | |
| 其他 | 无 | | | | | | | | | | | | | | | |

| 指令 | $MFLO\ rd$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 格式(31-16) | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | 0000000000 | | | | | | | | | |
| 格式(15-0) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | | 00000 | | | | 010010 | | | | | |
| 操作 | $R[d] <= LO$ | | | | | | | | | | | | | | | |
| 其他 | 无 | | | | | | | | | | | | | | | |

| 指令 | $MTHI\ rs$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 格式(31-16) | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | 00000 | | | | |
| 格式(15-0) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 0000000000 | | | | | | | | | | 010001 | | | | | |
| 操作 | $HI <= R[s]$ | | | | | | | | | | | | | | | |
| 其他 | 无 | | | | | | | | | | | | | | | |

| 指令 | $MTLO\ rs$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 格式(31-16) | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | 00000 | | | | |
| 格式(15-0) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 0000000000 | | | | | | | | | | 010011 | | | | | |
| 操作 | $LO <= R[s]$ | | | | | | | | | | | | | | | |
| 其他 | 无 | | | | | | | | | | | | | | | |

## 8.7 陷入指令

| 指令 | $SYSCALL\ rs$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | code | | | | | | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | code | | | | | | | | | | 001100 | | | | | |
| **操作** | 系统调用 | | | | | | | | | | | | | | | |
| **其他** | 异常：System Call | | | | | | | | | | | | | | | |

## 8.8 特权指令

| 指令 | $ERET$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 010000 | | | | | | 1 | code | | | | | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 0000000000 | | | | | | | | | | 011000 | | | | | |
| **操作** | 异常返回 | | | | | | | | | | | | | | | |
| **其他** | ERET 没有延迟槽 | | | | | | | | | | | | | | | |

| 指令 | $MFC0\ rt, rd$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 010000 | | | | | | 00000 | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000000 | | | | | | | | sel(000) | | |
| **操作** | $R[t] <= CP0[R[d]]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $MTC0\ rt, rd$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 010000 | | | | | | 00100 | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | rd | | | | | 00000000 | | | | | | | | sel(000) | | |
| **操作** | $CP0[R[d]] <= R[t]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $TLBWI$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 010000 | | | | | | 1 | 000000000 | | | | | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 0000000000 | | | | | | | | | | 000010 | | | | | |
| **操作** | 写 TLB | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $TLBWR$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 010000 | | | | | | 1 | 000000000 | | | | | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 0000000000 | | | | | | | | | | 000010 | | | | | |
| **操作** | 写 TLB | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

# 9 UBOOT 额外需要的 11 条指令

## 9.1 算数指令

| 指令 | $DIV\ rs, rt$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 0000000000 | | | | | | | | | | 011010 | | | | | |
| **操作** | $(LO, HI) <= R[s]/R[t]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

| 指令 | $DIVU\ rs, rt$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | rs | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 0000000000 | | | | | | | | | | 011011 | | | | | |
| **操作** | $(LO, HI) <= R[s]/R[t]$ | | | | | | | | | | | | | | | |
| **其他** | 无 | | | | | | | | | | | | | | | |

## 9.2 访存指令

| 指令 | $LH\ rt, offset(base)$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 100001 | | | | | | base | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $rt <= MEMORY[BASE + OFFSET]$ | | | | | | | | | | | | | | | |
| **其他** | 可能会导致 TLB Refill, TLB Invalid, Bus Error, Address Error | | | | | | | | | | | | | | | |

| 指令 | $LHU\ rt, offset(base)$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 100101 | | | | | | base | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $rt <= MEMORY[BASE + OFFSET]$ | | | | | | | | | | | | | | | |
| **其他** | 可能会导致 TLB Refill, TLB Invalid, Address Error | | | | | | | | | | | | | | | |

| 指令 | $LWL\ rt, offset(base)$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 100010 | | | | | | base | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $rt <= rtmergeMEMORY[BASE + OFFSET]$ | | | | | | | | | | | | | | | |
| **其他** | 可能会导致 TLB Refill, TLB Invalid, Bus Error, Address Error | | | | | | | | | | | | | | | |

| 指令 | $LWR\ rt, offset(base)$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 100110 | | | | | | base | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $rt <= rtmergeMEMORY[BASE + OFFSET]$ | | | | | | | | | | | | | | | |
| **其他** | 可能会导致 TLB Refill, TLB Invalid, Bus Error, Address Error | | | | | | | | | | | | | | | |

| 指令 | $SH\ rt, offset(base)$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 101001 | | | | | | base | | | | | rt | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| **操作** | $MEMORY[BASE + OFFSET] <= rt$ | | | | | | | | | | | | | | | |
| **其他** | 可能会导致 TLB Refill, TLB Invalid, Bus Error, Address Error | | | | | | | | | | | | | | | |

### 9.3 其他指令

| 指令 | $BREAK$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 格式(31-16) | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | code | | | | | | | | | |
| 格式(15-0) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | code | | | | | | | | | | 001101 | | | | | |
| 操作 | 系统调用 | | | | | | | | | | | | | | | |
| 其他 | 异常：Breakpoint | | | | | | | | | | | | | | | |

| 指令 | $CACHE\ op, offset(base)$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 格式(31-16) | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 101001 | | | | | | base | | | | | op | | | | |
| 格式(15-0) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | offset | | | | | | | | | | | | | | | |
| 操作 | $OPcache$ | | | | | | | | | | | | | | | |
| 其他 | 可能会导致 Coprocessor Unusuable, Cache Error 和访存指令的四种异常 | | | | | | | | | | | | | | | |

| 指令 | $SYNC$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 格式(31-16) | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | code | | | | | | | | | |
| 格式(15-0) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | code | | | | | | | | | | 001101 | | | | | |
| 操作 | To order loads and stores | | | | | | | | | | | | | | | |
| 其他 | 无 | | | | | | | | | | | | | | | |

| 指令 | $WAIT$ | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 格式(31-16) | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 000000 | | | | | | 1 | Implementation-Dependent code | | | | | | | | |
| 格式(15-0) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | Implementation-Dependent code | | | | | | | | | | 100000 | | | | | |
| 操作 | 等待特定事件 | | | | | | | | | | | | | | | |
| 其他 | 无 | | | | | | | | | | | | | | | |

| 指令 | *SDBBPcode* | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **格式(31-16)** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | 011100 | | | | | | code | | | | | | | | | |
| **格式(15-0)** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | code | | | | | | | | | | 111111 | | | | | |
| **操作** | To case a debug breakpoint exception | | | | | | | | | | | | | | | |
| **其他** | Debug breakpoint exception | | | | | | | | | | | | | | | |