

# 32位MIPS综合实验需求文档

原子小组

October 21, 2015

## 目 录

<b>1 引言</b>	<b>2</b>
1.1 编写目的	2
1.2 背景	2
1.3 定义	2
1.4 Mips架构下CPU运行概述	3
1.5 参考资料	4
<b>2 功能需求</b>	<b>4</b>
2.1 CPU	4
2.1.1 ALU	5
2.1.2 乘法器	6
2.1.3 寄存器堆	6
2.1.4 CP0	6
2.1.5 异常中断处理	10
2.1.6 MMU	12
2.1.6.1 虚拟地址映射	12
2.1.6.2 TLB	12
2.2 Ucore	12
2.2.1 BIOS	12
2.2.2 远程文件执行	12
2.3 外设	13
2.3.1 串口	13
2.3.2 VGA	13
2.3.3 ps/2键盘	13
2.3.4 网口	13
2.4 Decaf编译器	13
2.4.1 汇编指令的生成	13
2.4.2 库函数调用及calling convention	13
2.4.3 程序入口及退出	14
2.5 指令集与数据通路	14
<b>3 性能需求</b>	<b>14</b>
<b>4 运行环境需求</b>	<b>14</b>
4.1 设备	14
4.2 控制	14
<b>5 附录</b>	<b>14</b>
5.1 指令系统	14

# 1 引言

## 1.1 编写目的

计算机组成原理32位MIPS实验是在计原16位实验的基础上的扩展。在实验原理方面与软件工程、操作系统、编译原理等多门课程相结合。实验初期学习曲线较陡，在原理方面比较难以掌控。在编程实现方面涉及到大规模硬件语言的代码书写，需要对数字逻辑设计有比较清晰的思路，工作量非常大。

撰写本篇需求文档的目的如下：

1. 明确项目需求，对该系统进行一个能够达成一致认可的描述，明确需要完成的功能；
2. 明确开发资源与项目目标；
3. 控制整个开发过程，作为开发手册的一部分，降低开发中小组讨论的成本。

文档预期读者包括开发人员、任务提出者及其他使用该资源的用户。

## 1.2 背景

本项目的系统名称为32位MIPS处理器。

本项目任务由计算机组成原理课程刘卫东老师、李山山老师和软件工程课程白晓颖老师共同提出。

承担本项目的开发者为计33班的徐炜杰、王楠和黄欢。此外还受到了刘卫东、白晓颖、李山山三位老师的指导和王钧奕、张乐两位助教的帮助。

## 1.3 定义

本段中给出了需求文档中对于一些术语的基本定义，具体如下。

表 1: 定义列表

术语	描述
RISC	Reduced Instruction Set Computer的缩写，指精简CPU指令集
MIPS	Microprocessor without Interlocked piped stages的缩写，指无内部互锁流水级的微处理器，为一种典型RISC指令集
CPU	中央处理器，为本项目的最终目标
CP0	系统控制协处理器，是CPU和操作系统交互的窗口；CPU通过CP0向操作系统传递运行信息和异常信息等，而操作系统则通过操作CP0设置硬件的运行状态
MMU	内存管理单元，负责对虚拟地址进行映射，并总管访问各种存储器和输入输出接口的功能
TLB	传输后备缓冲器（快表），即页表的高速缓存，用于加快地址转换。本项目需对内存进行分页管理，即虚拟地址通过页表映射得到物理地址。
ALU	算术逻辑单元
RAM	存储程序的硬件，断电不保存信息，速度较快，常用于电脑主存储器
ROM	存储程序的硬件，断电可保存信息
Flash	存储程序的硬件，断电可保存信息，实验中用作硬盘
BIOS	主板上的启动程序，负责初始化硬件引导操作系统
BootLoader	由BIOS启动的一段特殊程序，用于将操作系统从Flash中加载到内存中，并且跳转到开始地址执行
Controller	CPU模块之一，根据指令和硬件反馈信息给出控制信号，控制各个部件和整个CPU的运行
组合逻辑元件	与运行时钟无关的元件，某一时刻的输出只取决于当前输入
时序逻辑元件	根据系统的时钟信号，在时钟的上升沿工作的元件。某一时刻的稳定输出由当前输入和历史状态共同决定。注意组合逻辑元件和时序逻辑元件往往没有明确的分界；一个寄存器往往仅在上升沿写入，但却随时按照组合逻辑方式将其内容进行输出

## 1.4 Mips架构下CPU运行概述

本段给出MIPS架构下CPU运行的基本方式。这是设计MIPS架构CPU需要掌握的基础知识，也是以下需求分析的基础，望读者详加掌握。

MIPS架构下的计算机和所有计算机一样，基本功能都是从内存中提取指令并执行，保存所提取的指令地址的寄存器称为PC。

硬件运行分为用户态和内核态以进行权限管理，内核态可使用的内存空间更大，可使用的指令条数和硬件模块更多，地址映射方法也和用户态不同。如此可以方便操作系统完成TLB管理、异常处理等系统级操作，并将以上内容向用户隐藏，既可以简化用户的操作，也可以防止用户的误操作造成难以预测的危险。上述过程中需要利用协处理器CP0，如在异常处理过程中硬件检测到异常后，将异常信息与发生异常的指令地址存入CP0寄存器中，并跳转到操作系统的通用异常处理入口处。随后该段代码通过检查CP0保存的数据识别异常的种类，并分发到对对应的代码段进行处理。异常处理完成后，CPU跳转回被打断的指令继续执行，如此即完成了一次异常处理。

内存管理方面采用虚拟地址机制，将虚拟地址映射为物理地址，对应到实际的存储模块。如此既可以设置权限，防止访问未经授权的资源而造成危险，同时也使得各个进程之间互不干扰，虚拟地址可以复

用，大大简化了程序的设计。从虚拟地址到物理地址的映射方法有多种，而方法的选择则是依据虚拟地址的值本身。其中用户态下虚拟地址的映射较为复杂，需要通过查表进行对应，这就是所谓的页表管理机制。考虑到该查找操作非常频繁，而页表本身存放在内存中，对其进行访问代价很大，所以将页表的一部分放入CPU做成高速缓存，这就是快表TLB。最后，访存操作还为CPU和外界的互动提供了统一接口，如从串口的输入输出就是通过读写某一特定地址来实现的。硬件上总管从接收虚拟地址开始映射到读写数据完成这整个过程的模块就是MMU。

## 1.5 参考资料

- [1] 实验指导文档
- [2] OSLab实验参考文档
- [3] 计算机组成原理综合实验报告 贾开
- [4] 刘卫东, 李山山, 宋佳兴, 等. 计算机硬件系统实验教程[M]. 清华大学出版社, 2013.
- [5] 帕特森, 亨尼斯, 康继昌, 等. 计算机组成与设计: 硬件/软件接口[M]. 机械工业出版社, 2012.
- [6] Sweetman D. See MIPS run[M]. Morgan Kaufmann, 2010.
- [7] 雷思磊. 自己动手写CPU[M]. 电子工业出版社, 2014.

## 2 功能需求

### 2.1 CPU

本实验将实现基于标准32位MIPS指令集的子集的五级流水CPU，支持异常、中断、TLB等。

MIPS指令集的各条指令可以分解为取指（IF）、译码（ID）、执行（EX）、访存（MEM）和回写（WB）五个阶段，分别对应于本次实验需要实现的五个核心模块。模块内部采用组合逻辑电路实现，相邻模块之间的数据传输采用时序逻辑电路实现，每经过一个时钟周期，所有阶段将准备好的数据分别交给下一个阶段，从而实现流水CPU。

此外，我们还需要实现寄存器堆模块（Regfile）、HILO模块（HILO）、协处理器模块（CP0）和控制模块（CTRL）。寄存器堆模块在译码阶段实现，便于指令在译码阶段访问寄存器并得到相应的数据。HILO模块和协处理器模块在回写阶段实现。控制模块则用于控制整个流水线的暂停、清除等动作，因此不将其归入流水线中的某一个阶段。

流水线主要有结构冲突、数据冲突和控制冲突三种冲突类型。由于结构冲突不可避免，而控制冲突可以通过延时槽来解决，因此本实验重点需要解决的冲突是数据冲突。为此，我们引入数据前推技术，将计算结果从其产生处直接送到其他指令需要或所有需要的功能单元处，避免流水线暂停，其实现需要各个阶段相互协作。上述解决方案的前提是新的寄存器的值必须在执行阶段计算出来，如果是加载指令则不满足这个前提。对于load相关的情况就需要暂停流水线，为此需要给流水线实现暂停机制，使得Controller可以接收到来自各个阶段的暂停请求，并实现对流水线的暂停和恢复操作。

CPU的总体设计图如下图所示。



2.1.2 乘法器

乘法器负责实现乘法功能，输入为两个32位整数，输出为两个32位整数，分别存放在LO和HI寄存器中。

乘法运算采用Verilog语言提供的乘法运算符实现。我们没有将乘法器作为一个独立的元件来实现，而是在ALU中加入乘法的操作码，将乘法器并入ALU中实现。

考虑到乘法运算需要的时间比较长，如果乘法指令需要多周期完成，那么Controller就必须暂停流水线，等待乘法指令完成后再让流水线继续运作。为避免这种情况发生，可适当降低时钟频率，以使得乘法运算可以在一个时钟周期内完成运算。

2.1.3 寄存器堆

寄存器堆负责实现通用寄存器的读写和在数据通路中的控制，在流水线译码阶段读取一个或两个通用寄存器的数据（组合逻辑），并在流水线写回阶段将结果写入通用寄存器（时序逻辑）。

寄存器堆采用FPGA的逻辑单元来实现数据的存储，在32位MIPS架构下需要实现32个32位通用寄存器。

2.1.4 CP0

CP0是系统控制协处理器，本次实验我们需要通过CP0实现对TLB、MMU及异常处理的管理机制。下表列出了CP0的寄存器及其功能，这里我们只给出必须实现的11个寄存器。

编号	寄存器名称	寄存器功能
0	Index	用于TLBWI指令访问TLB入口的索引序号
2	EntryLo0	作为TLBWI及其他TLB指令接口，管理偶数页入口
3	EntryLo1	作为TLBWI及其他TLB指令接口，管理奇数页入口
9	BadVAddr	捕捉最近一次地址错误或TLB异常（重填、失效、修改）时的虚拟地址
10	Count	每隔一个时钟增加1，用作计数器，并可使能控制
11	EntryHi	TLB异常时，系统将虚拟地址部分写入EntryHi寄存器中用于TLB匹配信息
12	Compare	保持一定值，当Count值与Compare相等时，SI.TimerInt引脚变高电平直到有数值写入Compare，用于定时中断
13	Status	表示处理器的操作模式、中断使能及诊断状态
15	Cause	记录最近一次异常的原因，控制软件中断请求以及中断处理派分的向量
16	EPC	存储异常处理之后程序恢复执行的地址
18	EBase	识别多处理器系统中不同的处理器异常向量的基地址

(1) Index寄存器 Index寄存器是一个32位读/写寄存器，可用于TLBP、TLBR和TLBWI指令访问TLB入口的索引序号。

Index区域的大小根据具体实现方式随TLB的入口个数而定。对于基于TLB的内存管理单元MMU来说，该区域的最小值为 $\lceil \log_2(TLBEntries) \rceil$ 。

如果一个写入Index寄存器的值大于等于TLB入口数，则该处理器的操作是未定义的。

该寄存器仅对TLB有效。

31	30	6	5	0
P	0			Index

区域		描述	读/写	重置状态
名称	比特			
P	31	检测故障。当先前的TLBProbe (TLBP) 指令没有在TLB中寻得匹配时，硬件会将其置为1。	R/W	未定义
0	30: 6	必须写为0；读取时返回0.	0	0
Index	5:0	TLB入口的索引值，收到TLBRead和TLBWrite指令影响。 对于16或32个入口的TLB，如果索引指向一个不存在的入口，则该行为是未定义的。	R/W	未定义

- (2) EntryLo1/EntryLo0寄存器 这对EntryLo寄存器的作用等同于TLB、TLBR、TLBWI和TLBWR指令间的接口。对基于TLB的MMU而言，EntryLo0管理偶数页的入口，EntryLo1管理奇数页的入口。如果出现了地址错误，TLB失效，TLB修改或是TLB重填异常的行为，那么EntryLo0和EntryLo1寄存器的内容将会成为未定义的。只有当基于TLB的存储管理单元存在时，这些寄存器才有效。

31	30	29	26	25	6	5	3	2	1	0
R		O		PFN		C		D	V	G

区域		描述	读/写	重置状态
名称	比特			
R	31:30	保留区域。写入时应被忽略；读取时返回0。	R	0
O	29:26	在一般情况下，这4个位为PFN的一部分。但由于24k仅支持32位的物理地址，PFN仅有20位的位宽，所以，该寄存器的29:26位强制写入0。	R	0
PFN	25: 6	页帧号：有助于物理地址高位的定义。PFN区域对应了物理地址的31 12位。	R/W	未定义
C	5: 3	页面一致性属性。	R/W	未定义
D	2	已使用或写使能位：表示了该页面已经被写入，并且/或者是可写入的。如果D=1，则允许写入该页；如果D=0，则不允许写入该页，否则会引起TLB修改异常。	R/W	未定义
V	1	有效位：指明当前TLB入口是否有效，即虚拟页面映射是否有效。如果该位为1，那么允许进入该页；如果V=1，则允许访问该页；如果V=0，访问该页会引起TLB无效异常。	R/W	未定义
G	0	全局位：在对TLB入口进行写操作时，EntryLo0和EntryLo1寄存器中的G位与运算的结果作为TLB入口的G位。如果TLB入口的G位为1，ASID比较将在TLB匹配中被忽略掉。在对TLB入口进行读操作时，EntryLo0和EntryLo1的G位都反映了TLB的G位的状态。	R/W	未定义

C[5:3]值	缓存一致性属性
0	可缓存、非一致性、写通、无写分配
1	保留
2	不可缓存
3	缓存可、非一致性、写回、写分配
4,5,6	保留
7	无缓存加速

- (3) **EntryHi寄存器** EntryHi寄存器包含了用于TLB读、写和访问操作的虚拟地址匹配信息。当TLB异常（TLB Refill, TLB Invalid或TLB Modified）发生时，系统将虚拟地址的[31:13]位写入EntryHi寄存器的VPN2区域。TLBR指令将选中的TLB入口相应的区域写入EntryHi寄存器。软件（通常是操作系统）将当前地址空间标识符写入ASID区域，该区域在TLB比较过程中用于确定TLB是否可以匹配。

由于ASID区域被TLBR指令重填覆盖了，软件必须保存和重新存储有关TLBR使用的ASID的值。这在发生TLB失效和TLB修改异常时，以及在其它存储管理软件中尤为重要。

在发生了地址错误的异常后，EntryHi寄存器的VPN2区域将成为未定义的，并且该区域可能在发生地址错误异常的过程中被硬件修改。EntryHi寄存器的软件写操作（通过MTC0）不会导



致BadVAddr和Context寄存器中的地址相关区域发生隐式的写入（implicit write）。

31	13	12	8	7	0
VPN2			0		ASID

区域		描述	读/写	重置状态
名称	比特			
VPN2	31..13	虚地址（虚拟页面数目的一半）的VA <sub>31..13</sub> 。在TLB异常或TLB进行读取时，该区域被硬件写入；在TLB写之前，该区域被软件写入。	R/W	未定义
0	12..8	必须写为0；在读取时返回0。	0	0
ASID	7..0	地址空间标识符。在TLB读取时，该区域由硬件写入；通过软件写该区域，可以为TLB写操作建立当前的ASID值，这样可以避免对TLB的访问匹配所有TLB入口的ASID值。	R/W	未定义

该寄存器仅对TLB有效。

- (4) **Status寄存器** Status寄存器是一个读/写寄存器，可以表示处理器的操作模式、中断使能以及诊断状态。该寄存器的区域联合作用，可以创建处理器的工作模式。

中断使能：当一下所有条件成立时启用中断：

Status[0]:IE = 1

Status[0]:EXL = 0

Status[0]:ERL = 0

额外的：Debug[0]:DM = 0

当这些条件都符合时，设置IM（Status[16:9]）位和IE位可以使能中断。

EXL与ERL任一位置1都可使系统进入Kernel模式，否则为用户模式。

- (5) **Cause寄存器** Cause寄存器主要记录最近一次异常的原因，也控制软件中断请求以及中断处理派分的向量。除了IP1..0、DC、IV和WP区域，Cause寄存器中其它的所有区域都是只读的。第二版架构在外部中断控制器（EIC）的中断模式下，增加了可选项。在这个模式下，IP7..2表示请求中断优先级（RIPL）。

Cause[6:2]表示ExcCode，即异常号。

- (6) **Ebase寄存器** EBase寄存器是一个读/写寄存器，包含了在StatusBEV为0时所使用的异常向量的基地址及一个只读CPU号，该CPU号可以被软件用来区分多处理器系统中不同的处理器。

EBase寄存器使软件能在多处理器系统中识别特定的处理器，并允许每个处理器的异常向量可以不同，特别是对由多种处理器组成的系统。当StatusBEV为0时，EBase寄存器的位31:12由0构成，形成异常向量的基地址。当StatusBEV为1时，或在任何EJTAG调试异常的情况下，异常向量的基地址由固定的缺省值确定。

EBase寄存器的位31:30固定为2#10，从而强制异常基地址在kseg0或kseg1的无映射的虚拟地址段中。在缓存错误异常中，异常基地址的位29将被强制置为1，从而使异常处理器从无缓存的kseg1段开始执行。如果需要改变异常基地址寄存器的值，则操作必须在StatusBEV为1的情况下进行。如果在StatusBEV为0时，在异常基地址区域写入了一个不同的值，处理器的操作将成为未定义的。

将位31:12与异常基地址区域相结合，可允许将异常向量的基地址置于任何4KB字节大小的页面的边界上。

31	30	29		12	11	10	9	0
1	0	Exception Base			0	CPUNum		

区域		描述	读/写	重置状态
名称	比特			
1	31	在写入时，该位被忽略；在读取时，返回0.	R	1
Exception	29:12	与位31:30结合，该区域在 $Status_{SEV}$ 为0时，指定了异常的基地址。	R/W	0
CPUNum	9:0	该区域包含了一个标识符，它对多处理器系统的每个CPU都是唯一的。软件可以用此来判断它运行的位置。这个区域的值由连接到内核的 $SI\_CPUNum[9:0]$ 静态输入管脚设置。	R	外部设置
0	30, 11:10	必须写为0；在读取时返回0.	0	0

通过调用MFC0，MTC0指令，CP0提供了统一的对外接口以完成对寄存器组的访问。

实现步骤:

- (1) 选择要实现的CP0寄存器，可考虑推荐实现的寄存器或自行决定额外寄存器实现。
- (2) 按照CP0寄存器的功能分别在CPU的不同模块完成各寄存器的赋值。
  - (a) 在译码、运算及访存阶段发生地址错误时将错误地址赋值给BadVAddr
  - (b) 异常处理开始时，若为TLB异常，则将错误地址高20位赋值给EntryHi高20位
  - (c) 异常处理开始时，将Status[1]赋值为1；在执行ERET指令时将Status[1]赋值为0
  - (d) 异常处理开始时，将Cause[6:2]赋值为异常号
  - (e) 异常处理开始时，将EPC赋值为Victim指令地址
  - (f) 每个周期，Count加1
  - (g) 其他写操作由软件完成
- (3) 实现MFC0，MTC0指令访问CP0寄存器的功能。
- (4) 通过各寄存器值控制相应功能。

### 2.1.5 异常中断处理

本次实验要求实现精确异常处理，以下是一些可能用到的中断及异常的情况。

异常号	异常名	描述
0	Interrupt	外部中断、异步发生，由硬件引起
1	TLB Modified	内存修改异常，发生在Memory阶段
2	TLBL	读未在TLB中映射的内存地址触发的异常
3	TLBS	写未在TLB中映射的内存地址触发的异常
4	ADEL	读访问一个非对齐地址触发的异常
5	ADES	读访问一个非对齐地址触发的异常
8	SYSCALL	系统调用
10	RI	执行未定义指令异常
11	Co-Processor Unavailable	试图访问不存在的协处理器异常
23	Watch	Watch寄存器监控异常

下表列出可能用到的中断号。

中断号	设备
0	系统计时器
1	键盘
3	通讯端口COM2
4	通讯端口COM1

中断/异常处理的一般流程如下：

- (1) 保存中断信息，主要是EPC，BadVAddr，Status，Cause等寄存器的信息

EPC:存储异常处理之后程序恢复执行的地址。对于一般异常，当前发生错误的指令地址即为EPC应当保存的地址；而对于硬件中断，由于是异步产生则可以任意设定一条并未执行完成的指令地址保存，但在进入下一步处理之前，该指令前的指令都应当被执行完。

BadVAddr: 捕捉最近一次地址错误或TLB异常（重填、失效、修改）时的虚拟地址。

Status: 将EXL位置为1，进入kernel模式进行中断处理。

Cause: 记录下异常号。

EntryHi: TLB异常时，记录下BadVAddr的部分高位。

- (2) 根据Cause中的异常号跳转到相应的异常处理函数入口
- (3) 中断处理
- (4) 通过调用ERET指令恢复现场，返回EPC所存地址执行并且将Status中的EXL重置为0表示进入user模式。

实现步骤:

- (1) 在可能发生异常的位置实现对异常的记录。
  - (a) 访存时可能发生ADEL，ADES，TLBM，TLBL，TLBS，Watch异常
  - (b) 译码后可能发生RI，SYSCALL，Co-ProcessorU异常

- (2) 实现对中断的记录。
  - (a) 硬件产生中断时将信息写入CP0寄存器
- (3) 根据异常记录信息判断是否产生异常。
- (4) 进入异常处理流程。

## 2.1.6 MMU

MMU负责实现虚拟地址转换到物理地址，再访问相应元件读取或写入数据的全过程，并实现内核态与用户态的区分。同时MMU还需要集成TLB模块，并实现相应的TLB异常处理。这里我们只考虑硬件需要提供给操作系统的支持，不考虑操作系统已经实现的功能。

### 2.1.6.1 虚拟地址映射

通过划分虚拟地址，将所有外设映射到特定的虚拟地址来实现操作系统对外设的I/O操作，对于未映射的地址，读操始终返回0，写操作忽略。

虚拟地址	设备	说明
[0x00000000,0x007FFFFF]	RAM	共8MB
[0x10000000,0x10000FFF]	引导ROM	共4KB
[0x1E000000,0x1EFFFFFF]	flash	地址空间共16MB，但对于32位字，只有低16位有效
[0x1A000000,0x1A096000]	VGA	
0x1FD003F8	串口数据	
0x1FD003FC	串口状态	
0x1FD00400	数码管	
0x0F000000	键盘码	

**2.1.6.2 TLB** TLB是一个寄存器组，用作页表的缓存。根据表大小以及相连度不同而不同。以下是设计方案：

对于一个拥有 $2^N$ 个表项的TLB来说，每个表项有64位长，每次更新时它的信息被存放在如下寄存器中：目录为 $INDEX[N-1:0]$ ，对应的每个表项即每个入口为

$\{EntryHi[31:13], EntryLo1[25:6], EntryLo1[2:1], EntryLo0[25:6], EntryLo0[2:1]\}$

即高22位存储了虚拟地址高22位，后面分别跟着两组物理地址（奇偶）及其标志位（Valid，Global），匹配虚拟地址与标志后将实地址合并得到物理地址。

## 2.2 Ucore

### 2.2.1 BIOS

BIOS即为启动ucore所用的Bootloader程序，通常是放在Flash中。而本实验中我们将在FPGA里建立一块ROM，将Bootloader放置在该ROM中，并且设置CPU的访问地址从该ROM开始。这样能避免由于Flash的读写不稳定而对BIOS造成的破坏，还能将ucore与其独立开来。

BIOS启动时，Flash中的操作系统加载到内存中，然后跳转到操作系统的初始化代码，从而开始操作系统的工作。

### 2.2.2 远程文件执行

实验要求修改ucore，实现简单的远程文件执行功能，即通过串口从PC上获取ELF文件，并在本地执行。

## 2.3 外设

### 2.3.1 串口

串口的功能需求为实现与PC机的通信，通过计算机键盘输入数据，向计算机输出数据。

串口模块的主要部分位于板子上的CPLD中，在FPGA端，对串口的控制通过data\_ready、tbre、tsre、rdn、wrn进行。

当data\_ready= '1' 时串口数据就绪，可以读出。

当tbre and tsre= '1' 是，表示可以向串口写入。

将rdn置 '0' 且数据线写高阻，可以从数据线得出串口数据。

将数据写入数据线且将wrn置 '0'，可以向串口发出数据。

串口模块1FD003F8地址表示数据（只有低8位有效），必须使用SW指令写入。

1FD003FC地址表示状态寄存器Status，Status&1=1时可写，Status&2=2时可读。

### 2.3.2 VGA

Device.VGA模块是显示控制模块，接受CPU写入的ASCII码数据，并维护一个字符矩阵，以字符中断的形式通过VGA接口输出到显示器。

CPU需要从1FC03000地址读出数据，如果结果为 '1' 表示可以写入，否则不能吸入；可以写入时，向1FC03000地址写入ASCII数据即可。

### 2.3.3 ps/2键盘

Device.Keyboard模块位键盘控制模块。只要从0F000000地址读出数据，结果为0表示没有新数据，否则读到的就是键入的ASCII码。

PS/2键盘读到的是扫描码，为了方便软件，可以在硬件层面加入编码转换，CPU读到的直接就是ASCII码。

### 2.3.4 网口

这是拓展要求。实验提供了DM9000A网口芯片与PC机进行网络通讯。该芯片带有通用处理器接口的以太网控制器，一个10/100M PHY和4K双字的SRAM，IO端口支持3.3V与5V容限值。

当收到数据包时，芯片通过中断信号方式通知CPU触发异常，由操作系统对数据包进行处理。若要完成此需求，则需对操作系统进行改写：在初始化时添加网口中断使能、添加网口芯片初始化代码、手动实现网络通信协议和网口驱动等。

## 2.4 Decaf编译器

### 2.4.1 汇编指令的生成

由于简化的CPU中并未实现add、sub指令，需要把decaf的MIPS后端里生成add、sub指令的部分改成addu、subu，区别仅在于溢出时后者不会产生异常。eps 另外、CPU中也未实现除法指令，不过由于所用测试程序中没有除法运算，因此不进行相关修改。如果需要除法，可以用其它指令手动实现除法函数，并把除法翻译成函数调用。

### 2.4.2 库函数调用及calling convention

标准MIPS32使用O32 ABI，函数调用的前四个参数通过\$a0-\$a3四个寄存器传输；但decaf编译出的程序的参数全都在栈上传递。当然，无论什么calling convention，只要能自恰，程序本身就应该能正常运行，所以需要解决的问题只有用于程序与C实现的库函数及操作系统交互的部分。

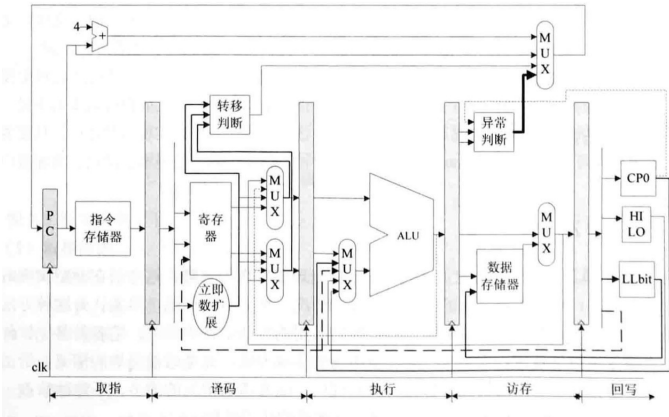
在这里，我们在decaf和库函数之间增加一个适配器层，将decaf的调用约定翻译成O32 ABI再调用库函数。

2.4.3 程序入口及退出

直接使用ucore里的linker script(user.Id)以及用户静态函数库libuser.a，在该环境下系统会设置好一些全局变量，然后跳转到main执行。我们将修改decaf编译器，将其输出的main重命名为decaf\_main，然后汇编实现一个新的main函数。由于decaf的main是void类型，我们默认其执行成功返回0，于是在decaf\_main返回后直接调用exit(0)。

2.5 指令集与数据通路

本次实验需要实现的MIPS32指令集详见附录。  
本次实验的基本数据通路图参照《自己动手写CPU》P56，并做一定的扩展。



3 性能需求

实现流水线CPU，通过良好的内部设计使流水线各阶段的用时相当，以减少因某些指令带来整体延迟的情况。

针对流水线CPU的开发目标，本项目的主频目标暂定为12.5MHz。

4 运行环境需求

4.1 设备

在硬件上提供了一块开发板，主要核心部件为Xilinx Spartan6 xc6slx100 FPGA。

4.2 控制

本工程将会充分利用开发板资源，控制部分采用键盘作为输入设备，PVGA显示作为输出，从终端通过串口和网口实现PC和开发板的双向通信和数据交流。如果有需要，也可以启用开发板上的第二个串口。

5 附录

5.1 指令系统

表 2: 设备与外部接口

环境	描述
FPGA	Xilinx Spartan6 xc6slx100
RAM	32-bit字长, 4块, 共8MB
Flash	16-bit字长, 共8MB
CPLD	与FPGA相连, 用于I/O
串口	2个
ps/2接口	1个
以太网接口	1个
VGA接口	1个

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 is immediate															
指令功能	$R[t] \leftarrow R[s] + \text{Sign-extend}(\text{immediate})$															
功能说明	对立即数进行符号扩展后与寄存器rs的值求和，结果保存到寄存器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	0	0	1
MIPS语言	ADDU rd rs rt															
指令功能	$R[d] \leftarrow R[s] + R[t]$															
功能说明	将寄存器rs与寄存器rt的值求和，结果保存到寄存器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	1	0	1	0	1	0
MIPS语言	SLT rd rs rt															
指令功能	if( $R[s] < R[t]$ ) then $R[d] = 1$ , else $R[d] = 0$															
功能说明	比较寄存器rs与寄存器rt的值并根据结果对寄存器rd赋值															

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	if( $R[s] < \text{Sign-extend}(\text{immediate})$ ) $R[t] = 1$ , else $R[t] = 0$															
功能说明	比较寄存器rs与立即数进行符号扩展后的值并根据结果对寄存器rt赋值															



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	if(R[s] < Zero-extend(immediate)) R[t] = 1, else R[t] = 0															
功能说明	比较寄存器rs与立即数进行零扩展后的值并根据结果对寄存器rt赋值															

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	if(R[s] < R[t]) R[d] = 1, else R[d] = 0															
功能说明	比较寄存器rs与寄存器rt的值并根据结果对寄存器rd赋值															

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] $\leftarrow$ R[s] - R[t]															
功能说明	用寄存器rs的值减寄存器rt的值，结果保存到寄存器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
	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 $\leftarrow$ R[s] * R[t]															
功能说明	将寄存器rs与寄存器rt的值相乘，保存到寄存器HI/LO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	$R[d] \leftarrow LO$															
功能说明	将LO寄存器的值保存到rd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	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	$R[d] \leftarrow HI$															
功能说明	将HI寄存器的值保存到rd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	$LO \leftarrow R[s]$															
功能说明	将寄存器rs的值保存到LO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	0	1
MIPS语言	MTHI rs															
指令功能	$HI \leftarrow R[s]$															
功能说明	将寄存器rs的值保存到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	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	BEQ rs rt immediate															
指令功能	if(R[s] = R[t]) PC $\leftarrow$ PC + Sign-extend(immediate)															
功能说明	如果寄存器rs与寄存器rt的值相等，则跳转到目的地址执行，否则顺序执行下一条指令															

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	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
	immediate															
MIPS语言	BGEZ rs immediate															
指令功能	if(R[s] >= 0) PC $\leftarrow$ PC + Sign-extend(immediate)															
功能说明	如果寄存器rs的值大于等于0，则跳转到目的地址执行，否则顺序执行下一条指令															

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	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
	immediate															
MIPS语言	BGTZ rs immediate															
指令功能	if(R[s] > 0) PC $\leftarrow$ PC + Sign-extend(immediate)															
功能说明	如果寄存器rs的值大于0，则跳转到目的地址执行，否则顺序执行下一条指令															

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	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
	immediate															
MIPS语言	BLEZ rs															
指令功能	if(R[s] <= 0) PC $\leftarrow$ PC + Sign-extend(immediate)															
功能说明	如果寄存器rs的值小于等于0，则跳转到目的地址执行，否则顺序执行下一条指令															

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	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
	immediate															
MIPS语言	BLTZ rs															
指令功能	if(R[s] < 0) PC $\leftarrow$ PC + Sign-extend(immediate)															
功能说明	如果寄存器rs的值小于0，则跳转到目的地址执行，否则顺序执行下一条指令															

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	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	BNE rs rt															
指令功能	if(R[s] != R[t]) PC $\leftarrow$ PC + Sign-extend(immediate)															
功能说明	如果寄存器rs与寄存器rt的值不相等，则跳转到目的地址执行，否则顺序执行下一条指令															

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	1	0	immediate(26bit)									
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	immediate(26bit)															
MIPS语言	J immediate															
指令功能	PC $\leftarrow$ PC + Sign-extend(immediate)															
功能说明	无条件跳转目的地址执行															

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	immediate(26bit)									
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	immediate(26bit)															
MIPS语言	JAL immediate															
指令功能	PC $\leftarrow$ PC + Sign-extend(immediate), RA $\leftarrow$ RPC															
功能说明	无条件跳转目的地址执行，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 rs rd															
指令功能	$PC \leftarrow R[s], R[d] \leftarrow RPC$															
功能说明	无条件跳转目的寄存器rs中所存地址执行，将延时槽后一条指令存入R[d]															

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	$PC \leftarrow R[s]$															
功能说明	无条件跳转至寄存器rs所存地址执行															

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	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	LW rt rs immediate															
指令功能	$R[t] \leftarrow \text{MEM}[R[s] + \text{Sign-extend}(\text{immediate})]$															
功能说明	将寄存器rs的值与立即数immediate符号扩展后相加所得存至rt中															

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	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	SW rt rs immediate															
指令功能	$\text{MEM}[R[s] + \text{Sign-extend}(\text{immediate})] \leftarrow R[t]$															
功能说明	将寄存器rt的值存入寄存器rs的值与立即数immediate符号扩展后相加所得地址中															

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	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	LB rt rs immediate															
指令功能	$R[t] \leftarrow \text{Sign-extend}(\text{MEM\_Byte}[R[s] + \text{Sign-extend}(\text{immediate})])$															
功能说明	将寄存器rs的值与立即数immediate符号扩展后相加所得地址中第一个字节取出来符号扩展后保存在寄存器rt中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	1	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
	immediate															
MIPS语言	LBU rt rs immediate															
指令功能	$R[t] \leftarrow \text{Zero-extend}(\text{MEM\_Byte}[R[s] + \text{Sign-extend}(\text{immediate})])$															
功能说明	将寄存器rs的值与立即数immediate符号扩展后相加所得地址中的第一个字节取出来零扩展后保存在寄存器rt中															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	1	0	1	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
	immediate															
MIPS语言	SB rt rs immediate															
指令功能	$\text{MEM\_Byte}[R[s] + \text{Sign-extend}(\text{immediate})] \leftarrow \text{LOW\_BYTE}[R[t]]$															
功能说明	将寄存器rt的最低字节取出来保存在rs的值与立即数immediate符号扩展后相加所得地址中															

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	0
MIPS语言	AND rd rs rt															
指令功能	$R[d] \leftarrow R[s] \& R[t]$															
功能说明	将寄存器rs与寄存器rt的值相与后的结果保存至寄存器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	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	$R[t] \leftarrow R[s] \& \text{Zero-extend}(\text{immediate})$															
功能说明	将寄存器rs的值与立即数零扩展后相与的结果保存至寄存器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	$R[t] \leftarrow \text{immediate} * 65536$															
功能说明	将16为立即数放至寄存器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	1	1
MIPS语言	NOR rd rs rt															
指令功能	$R[d] \leftarrow \sim(R[s]   R[t])$															
功能说明	将寄存器rs与寄存器rt的值或非后的结果保存至寄存器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	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	$R[d] \leftarrow R[s]   R[t]$															
功能说明	将寄存器rs与寄存器rt的值相或后的结果保存至寄存器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	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	$R[t] \leftarrow R[s] \mid \text{Zero-extend}(\text{immediate})$															
功能说明	将寄存器rs与立即数immediate零扩展后相或的结果保存至寄存器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	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	$R[d] \leftarrow R[s] \wedge R[t]$															
功能说明	将寄存器rs与寄存器rt的值异或后的结果保存至寄存器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	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	$R[t] \leftarrow R[s] \wedge \text{Zero-extend}(\text{immediate})$															
功能说明	将寄存器rs与立即数immediate零扩展后相异或的结果保存至寄存器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	rt					
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					immediate						0	0	0	0	0
MIPS语言	SLL rd rt immediate															
指令功能	$R[d] \leftarrow R[t] \ll \text{immediate}$															
功能说明	将寄存器rt中的值逻辑左移寄存器rs中的值位后的结果保存至寄存器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	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	$R[d] \leftarrow R[t] \ll R[s]$															
功能说明	将寄存器rt中的值逻辑左移寄存器rs中的值位后的结果保存至寄存器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	rt					
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					immediate						0	0	0	0	1
MIPS语言	SRA rd rt immediate															
指令功能	$R[d] \leftarrow R[t] \gg \text{immediate}(\text{arithmetic})$															
功能说明	将寄存器rt中的值算数右移立即数immediate位后的结果保存至寄存器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	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	$R[d] \leftarrow R[t] \gg R[s](\text{arithmetic})$															
功能说明	将寄存器rt中的值算数右移寄存器rs中的值位后的结果保存至寄存器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	rt					
二进制（低位）	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	rd					immediate						0	0	0	0	1
MIPS语言	SRL rd rt immediate															
指令功能	$R[d] \leftarrow R[t] \gg \text{immediate}(\text{logical})$															
功能说明	将寄存器rt中的值逻辑右移立即数immediate位后的结果保存至寄存器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	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	$R[d] \leftarrow R[t] \gg R[s](\text{logical})$															
功能说明	将寄存器rt中的值逻辑右移寄存器rs中的值位后的结果保存至寄存器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
	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
MIPS语言	SYSCALL															
指令功能	中断号 $\leftarrow$ SYSCALL															
功能说明	执行后除法中断															

二进制（高位）	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	1	0	1	1	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	CACHE															
指令功能																
功能说明	不做cache，视为NOP															

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	$PC \leftarrow EPC$															
功能说明	返回至EPC寄存器的地址执行，并设置Status寄存器的EXL位为0															

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	$R[t] \leftarrow CP0[R[d]]$															
功能说明	将协处理器0中的rd寄存器的值保存到rt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 rd rt															
指令功能	$CP0[R[d]] \leftarrow R[t]$															
功能说明	将寄存器rt的值保存到协处理器0中的rd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	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	$R[d] \leftarrow R[t] \ll R[s]$															
功能说明	写索引TLB项															

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	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	LHU rt rs immediate															
指令功能	$R[t] \leftarrow \text{Zero-extend}(\text{MEM.HALFWORD}[R[s] + \text{Sign-extend}(\text{immediate})])$															
功能说明	将寄存器rt中的值与立即数immediate符号扩展后相加所得地址中的低两个字节取出来零扩展后保存在寄存器rt中															