MIPS CPU 需求文档

杜家驹[1] 袁星驰[2] 王硕[3]

2017年1月

[1]电子邮件：dujj14@mails.tsinghua.edu.cn，学号：2014010257
[2]电子邮件：yuanxc13@mails.tsinghua.edu.cn，学号：2013011665
[3]电子邮件：w-s14@mails.tsinghua.edu.cn，学号：2014012276

目录

1 引言

1.1 背景

本项目是一个基于MIPS32的CPU设计项目，通过Verilog进行硬件设计，实现一个MIPS的CPU，能够运行一个简单的操作系统ucore。

本项目是计算机组成原理和软件工程的联合实验。

1.2 实验目标

1 使用提供的开发板，设计一个基于标准32位MIPS指令集的流水CPU，采用小端序，支持异常、中断、TLB等。

2 在CPU上运行操作系统，进入用户态和shell，能够执行shell命令。

3 为CPU增加调试功能，同时修改操作系统，提供设置断点的功能，能够调试程序。

4 实现对VGA、PS/2键盘的支持。

1.3 参考资料

《计算机组成与设计：硬件/软件接口》，David A. Patterson，John L. Hennessy

《See MIPS Run》，D. Sweetman

《自己动手写CPU》，雷思磊

1.4 实验设备与开发环境

开发板中含有：

FPGA一块，为Xilinx的Spartan6 xc6slx100

CPLD一块，用于PS/2 IO

32位RAM两块，各4MB，共计8MB

16位Flash一块，共8MB

开发环境为Xilinx ISE

使用的操作系统为清华大学操作系统课程实验的32位操作系统ucore，编译器为mips-sde-elf-gcc

1.5 性能需求

CPU时钟频率需要达到25M。

2 CPU功能需求

2.1 寄存器

CPU应含有32个通用的寄存器，分别以$0-$31命名。其中$0在任何时刻的值均为0，不受写入的影响。

对于整数乘法，CPU还应含有两个整数乘法寄存器hi，lo。

为实现对操作系统的支持，CPU还应含有协处理器0，简称CP0。CP0的具体描述见2.3节。

2.2 指令系统

CPU应当支持46条指令，可分为下面5类。

1 运算指令，共26条

表 1: 运算指令

addu	add word
addiu	add immediate word
subu	subtract word
slt	set on less than
slti	set on less than immediate
sltu	set on less than unsigned
sltiu	set on less than unsigned immediate
mult	multiply word
mflo	move from lo
mtlo	move to lo

表 1：运算指令

mfhi	move from hi
mthi	move to hi
and	and
andi	and immediate
lui	load upper immediate
nor	not or
or	or
ori	or immediate
xor	exclusive or
xori	exclusive or immediate
sll	shift word left logical
sllv	shift word left logical variable
sra	shift word right arithmetic
srav	shift word right arithmetic variable
srl	shift word right logical
srlv	shift word right logical variable

2 分支指令，共10条

表 2: 分支指令

beq	branch on equal
bne	branch on not equal
bgez	branch on greater than or equal to zero
bgtz	branch on greater than zero
blez	branch on less than or equal to zero
bltz	branch on less than zero
j	jump
jal	jump and link
jr	jump register
jalr	jump and link register

上面10条指令之后都会有延迟槽，无论是否跳转，指令都会执行。

3 访存指令，共5条

表 3: 访存指令

lw	load word
sw	store word
lb	load byte
lbu	load byte unsigned
sb	store byte

4 陷入指令，共1条

表 4: 陷入指令

syscall	system call

5 特权指令，共4条

表 5: 特权指令

eret	exception return
mfc0	move from coprocessor 0
mtc0	move to coprocessor 0
tlbwi	write indexed TLB entry

各条指令的格式和具体功能见第6节。

2.3 协处理器0(CP0)

根据操作系统的需要，CPU需要实现协处理器0(CP0)，其中应含有13个32位寄存器。它们的编号和名称分别为：

0 index
2 entrylo0
3 entrylo1
10 entryhi
上面四个寄存器用于指令tlbwi，详情请见TLB部分。

8 badaddr
导致最近地址相关异常的地址。详情请见中断处理部分。

9 count
计时器，每个CPU工作周期加1。

11 compare
计时器，当compare寄存器等于count寄存器时产生time interrupt，详情请见中断处理部分。

12 status
CPU的状态。各位的描述

表 6: status

bit 15-8	interrupt manager(im) 7-0 使能某种中断
bit 4	user mode(um) 处于用户态
bit 1	exception(exl) 发生了异常，此时禁止中断
bit 0	interrupt enable(ie) 使能中断

13 cause
异常或中断的原因。各位的描述

表 7: cause

bit31	exception in delay slot(bd) 异常发生在延迟槽中
bit15-8	interrupt(ip) 此时是否发生对应的中断
bit6-2	exception code(ec) 异常种类

14 epc

异常或中断的发生或返回地址

15 ebase
异常入口点地址

18 watchlo
19 watchhi
断点的地址，详情请见中断处理部分。

与CP0相关的指令有mfc0和mtc0，它们的作用为在通用寄存器和CP0之间转移数据。CP0中的所有寄存器都是可读写的。

2.4 TLB

为了实现对操作系统的支持，CPU需要能够实现从虚拟地址到物理地址的转换。这需要实现一个TLB来进行转换。

根据操作系统的需求，虚拟地址空间的划分和映射如下：

表 8: 虚拟地址空间的划分和映射

0x00000000-0x7fffffff	用户可用，TLB表项映射
0x80000000-0x9fffffff	线性映射到0x00000000-0x1fffffff
0xa0000000-0xbfffffff	线性映射到0x00000000-0x1fffffff
0xc0000000-0xffffffff	TLB表项映射

TLB中应当有16个表项，每一项有64bit。各位的定义如下：

表 9: TLB表项

bit62-44	virtual page number(vpn) 虚拟页号
bit43-24	physical page number0(ppn0) 物理页号
bit23	valid0 表示这一页是否有对应的物理页
bit22	write0 表示这一页是否可写
bit21-2	physical page number1(ppn1) 物理页号
bit1	valid1表示这一页是否有对应的物理页
bit0	write1表示这一页是否可写

操作系统要求虚拟页和物理页的大小都为4kB。TLB中每一项都对应2个连续的虚拟页。在查找一个虚拟地址对应的物理地址时，首先把这个虚拟地址的31-13位

与每一项的vpn进行匹配，如果没有相同的，则报告TLB缺失。如果有相同的，考虑虚拟地址的第12位，根据它为0或1来选择ppn0或ppn1。以第12位等于0为例，如果valid0为0，直接报告TLB缺失。如果这个操作是写操作，且write0为0，则报告权限错误。之后把ppn0和虚拟地址的11-0位结合起来就得到虚拟地址。

与TLB相关的指令有tlbwi，它的作用为把CP0中index，entrylo0，entrylo1，entryhi这四个寄存器中的相关信息写入TLB中。四个寄存器的含义如下：

index

表 10: index

bit3-0	表示要写入的TLB表项的序号

entrylo0

表 11: entrylo0

bit25-6	physical page number0(ppn0) 物理页号
bit2	write0 表示这一页是否可写
bit1	valid0 表示这一页是否有对应的物理页

entrylo1

表 12: entrylo1

bit25-6	physical page number0(ppn1) 物理页号
bit2	write1 表示这一页是否可写
bit1	valid1 表示这一页是否有对应的物理页

entryhi

表 13: entryhi

bit31-13	virtual page number(vpn) 虚拟页号

2.5 异常处理

CPU需要实现对中断和异常的支持。根据操作系统的需求，CPU需要支持的异常有：

0 interrupt 外部中断。

需要处理的中断有：

4 serial interrupt 由串口控制器产生

7 time interrupt 由CP0产生

中断在每个工作周期检测一次，检测到中断产生的条件为：cp0.status.ie = 1，cp0.status.exl = 0，cp0.status.im中允许相应的中断，并且外部的中断信号为1。

2 tlb miss on load

读数据时如果TLB中没有保存相应的映射，则产生此异常

3 tlb miss on store

写数据时如果TLB中没有保存相应的映射，则产生此异常

4 address error on load

当读取一个字的地址不是4的倍数时，或在用户态读取内核态的地址时，产生此异常。

5 address error on store

当写入一个字的地址不是4的倍数时，或在用户态写了内核态的地址时，或写了只读页面时，产生此异常。

8 syscall

执行syscall指令时产生此异常。

10 reserved instruction

当指令不合法或在用户态执行了mfc0/mtc0/tlbwi时，产生此异常。

23 debug

当前的PC与CP0中的watchlo或watchhi相同时，产生此异常。

异常处理过程：设置cp0.epc指向异常处理之后重新开始执行的地址，如果异常发生在延迟槽外，则设置为发生异常指令的地址，否则为设置为延迟槽前跳转指令的地址，对于中断，则设置为刚结束alu阶段的指令地址。cp0.status.exl置为1，cp0.cause记录下异常号、中断号以及是否在延迟槽种，cp0.badaddr置为最近一次发生错误的地址。然后跳转到异常入口点，异常为tlb miss时目标为cp0.ebase，其余异常目标为cp0.ebase+0x180。

需要实现指令eret，其作用为把cp0.status.exl置为0，并跳转到cp0.epc。

2.6 CPU的启动

CPU启动时应从0xbfc00000开始执行，状态为核心态，关闭所有中断。

3 总线与外部设备

3.1 物理地址映射

表 14: 物理地址映射

ROM	0x1fc00000-0x1fc001ff映射到0x000-0x1ff
RAM	0x00000000-0x003fffff映射到0x000000-0x3fffff
Flash	0x1e000000-0x1effffff映射到0x000000-0x7fffff
串口	0x1fd003f8-0x1fd003ff映射到0x0-0x7
VGA	0x10000000-0x10000fff映射到0x000-0xfff

3.2 ROM

ROM是CPU启动后开始执行的位置，大小为512byte，需要支持以32bit为单位进行读。

3.3 RAM

RAM用作内存，大小为4M，需要支持以32bit或8bit为单位进行读写。但由于SRAM只能以32bit为单位进行读写，因此需要SRAM控制器做相关的处理。

3.4 Flash

Flash用作硬盘，需要支持以32bit为单位进行写。Flash大小为8M，对应16M的地址空间。其原因是Flash实际上为16bit，而MIPS32每次访存都是32bit。每次读Flash时都在高位补上16个0。下表是一个更详细的说明：

表 15: Flash

总线地址	Flash控制器中的地址	Flash地址
0x1e000000	0x0	0x0
0x1e000001	0x1	0x0
0x1e000004	0x4	0x1
0x1e000005	0x5	0x1

3.5 串口与键盘

串口的地址空间为0x0-0x7，其中0x0-0x3为数据寄存器，对它的读写相当于读串口或写串口和键盘。0x4-0x7为数据寄存器，表示串口的输入缓冲区中有数据。当串口或键盘有字符输入时，把字符写入缓冲区中。当CPU读串口时，如果缓冲区中有数据，则输出一个数据，否则输出0xff。当一个数据被写入缓冲区时，串口控制器应该发出中断，持续时间为一个CPU工作周期。

3.6 VGA

VGA控制器对应的地址空间为0x000-0xfff，共4kB。其中，前2400byte代表显存，存放对应位置的字符的ascii码。分辨率为每行80个字符，共30行。需要支持以8bit为单位写。此外，还需要存储每个字符的点阵字体，字体大小为16*8。

3.7 总线协议

为了提高访存效率，需要实现AHB-Lite协议，其中master为CPU，slave为ROM/SRAM/VGA/串口/Flash等各种外设控制器。

4 自检工具

为了便于调试，需要事先验证SRAM和Flash没有问题，所以需要完成一个自检工具，其功能是读写SRAM和Flash，并且通过串口与PC交互。此工具应能够指定SRAM或Flash中的一个地址，向这个地址写入数据，然后再把数据读出来，验证正确性。当验证SRAM和Flash正确后，就可以通过此工具把操作系统写入Flash。

5 GDB

本项目需要完成一个GDB，实现对应用程序的调试。GDB的功能应该有设断点、显示寄存器内容、修改寄存器内容、单步调试、继续运行。为了实现这些功能，应该利用CPU的调试异常，修改操作系统，增加相关的syscall，然后再增加一个用户态程序，用于与人进行交互。

6 指令格式与功能

指令主要分为三种，即R型、I型、J型
R型指令的格式

表 16: R型

31-26bit	op
25-21bit	s
20-16bit	t
15-11bit	d
10-6bit	sa
5-0bit	func

其中op=000000。

I型指令的格式

表 17: I型

31-26bit	op
25-21bit	s
20-16bit	t
15-0bit	imm

J型指令的格式

表 18: J型

31-26bit	op
25-0bit	address

一些特殊的符号：
R[x]：通用寄存器，x为通用寄存器的编号
Mem[x]：内存中的数据

下表描述了各种指令的功能，最后一栏对I型和J型指令来说是op，对R型指令来说是func

表 19: 指令功能

指令	功能描述	op/func
addu(R)	R[d]=R[s]+R[t]	100001
addiu(I)	R[t]=R[s]+SignExtend(imm)	001001
subu(R)	R[d]=R[s]-R[t]	101011
slt(R)	if(R[s]<R[t]) R[d]=1 else R[d]=0	101010
slti(I)	if(R[s]<SignExtend(imm)) R[t]=1 else R[t]=0	001010
sltu(R)	if(R[s]<R[t]) R[d]=1 else R[d]=0(Unsigned)	101011
sltiu(I)	if(R[s]<ZeroExtend(imm)) R[t]=1 else R[t]=0(Unsigned)	001011
mult(R)	{hi,lo}=R[s]*R[t]	011000
mflo(R)	R[d]=hi	010010
mtlo(R)	R[d]=lo	010011
mfhi(R)	lo=R[s]	010000
mthi(R)	hi=R[s]	010001
and(R)	R[d]=R[s] and R[t]	100100
andi(I)	R[t]=R[s] and ZeroExtend(imm)	001100
lui(I)	R[t]=imm*65536	001111
nor(R)	R[d]=not(R[s] or R[t])	100101
or(R)	R[d]=R[s] or R[t]	100110
ori(I)	R[t]=R[s] or ZeroExtend(imm)	001101
xor(R)	R[d]=R[s] xor R[t]	100110
xori(I)	R[t]=R[s] xor ZeroExtend(imm)	001110
sll(R)	R[d]=R[t]<<sa	000000
sllv(R)	R[d]=R[t]<<R[s]	000100
sra(R)	R[d]=R[t]>>sa(arithmetic)	000011
srav(R)	R[d]=R[t]>>R[s](arithmetic)	000111
srl(R)	R[d]=R[t]>>sa(logical)	000010
srlv(R)	R[d]=R[t]>>R[s](logical)	000110
beq(I)	if R[s]=R[t] branch pc+4+SignExtend(imm)	000100
bne(I)	if R[s]!=R[t] branch pc+4+SignExtend(imm)	000101
bgez(I)	if R[s]>=0 branch pc+4+SignExtend(imm)(t=00001)	000001
bgtz(I)	if R[s]>0 branch pc+4+SignExtend(imm)	000111
blez(I)	if R[s]<=0 branch pc+4+SignExtend(imm)	000110
bltz(I)	if R[s]<0 branch pc+4+SignExtend(imm)(t=00000)	000001
j(J)	pc=(pc+4)[31:28]||address||00	000010
jal(J)	R[31]=pc+8, pc=(pc+4)[31:28]||address||00	001001

表 19：指令功能

指令	功能描述	op/func
jr(R)	pc=R[s]	001000
jalr(R)	rd=pc+8, pc=R[s]	001001
lw(I)	R[t]=Mem[R[s]+SignExtend(imm)](Word)	100011
sw(I)	Mem[R[s]+SignExtend(imm)]=R[t](Word)	101011
lb(I)	R[t]=Mem[R[s]+SignExtend(imm)](Byte, SignExtend)	100000
lbu(I)	R[t]=Mem[R[s]+SignExtend(imm)](Byte, ZeroExtend)	100100
sb(I)	Mem[R[s]+SignExtend(imm)]=R[t](Byte)	101000
mfc0(R)	R[t]=CP0[R[d]], s=00100 op=010000	-
mtc0(R)	CP0[R[d]]=R[t], s=00000 op=010000	-

其它指令

表 20: 其它指令

syscall	0x0000000c
tlbwi	0x42000002
eret	0x42000018