接口实验 MIPS 机器实现计划

一、 MIPS 指令系统

OpCode 说明

	000	001	010	011	100	101	110	111
000	R_TYPE	BCON(see note 2)	j	jal	beq	bne	blez	bgtz
001	addi	addiu	slti	sltiu	andi	ori	xori	lui
010	mfc0/mtc0(see note 1)							
011								
100	lb	lh		lw	lbu	lhu		
101	sb	sh		sw				
110								
111								

R_TYPE:

	000	001	010	011	100	101	110	111
000	sll		srl	sra	sllv		srlv	srav
001	jr	jalr			syscall(see note 4)			
010	mfhi	mthi	mflo	mtlo				
011	mult	multu	div	divu				
100	add	addu	sub	subu	and	or	xor	nor
101			slt	sltu				
110								
111								

Note:

1.mfc0:

010000 *00000* *rt* *rd* 00000000000
mtc0:
010000 *00100* *rt* *rd* 00000000000
注意是 rt<=rd

2.BCON:

BLTZ 0000 01ss sss0 0000 iiii iiii iiii iiii

BGEZ 0000 01ss sss0 0001 iiii iiii iiii iiii

BGEZAL 0000 01ss sss1 0001 iiii iiii iiii iiii

BLTZAL 0000 01ss sss1 0000 iiii iiii iiii iiii

3.支持的其他特殊指令

ERET:

0100 0010 0000 0000 0000 0000 0001 1000

break 0cccc 0x0d
比如：break 7 cause 9 异常向量 0x+80，用于调试

4. SYSCALL

$V0	功能	参数返回值
1	print integer	$a0: the integer
4	print string	$a0: the start address
5	read integer	$v0: the readed integer
8	read string	$a0: address of input buffer; $a1: maximum num
9	sbrk (allocate heap memory) 尚没有决定	$a0 = number of bytes to allocate; $v0:address
10	exit	
11	print character	$a0: the character
12	read character	$v0: contain the character
13	open file	$a0 = address of string containing filename

		$a1 = flags $a2 = mode
14	read from file	$a0 = file descriptor $a1 = address of input buffer $a2 = maximum number of characters to read
15	write to file	$a0 = file descriptor $a1 = address of output buffer $a2 = number of characters to write
16	close file	$a0 = file descriptor
下面是我们定义的系统调用		
30	get time	read the timer counter from register
31	put a pixel	$a0：x $a1: y $v1: color(least 4 bit)
32	设置文本模式还是图形模式	$a0=1: 图形模式 $a1=0:文本模式

5.伪指令
现在我们已经实现：
lwd $t1, label => load a word labeled by lable, and also lbd, lhd, and so on;
将会扩展。

二、硬件和系统部分

内存空间分布:

图形模式的显存空间，640x480，每个点 4bit。 ：e000_0000
文本模式的显存空间，我们支持 40*30 的文本显示，在内存中只放对应字符的 ascii，每个字模大小 16x16。 ：d000_0000
串口：尚未确认 显存：1byte=> 1: 图形模式 0:文本模式 键盘：2byte => 一个是是否有效，另一个是扫描码 ：c000_0000（开始 mapped io）
：b000_0000（开始文件缓冲：1k 空间） ：8000_0080（异常向量） ：8000_0000
：7fff_fffc （sp 初始指向） .data 段 ：1001_0000
：1000_ffff .dss 段，存放未初始化的(全局变量或者静态变量), gp 被初始化为 1000_8000 ：1000_0000
：0040_0000 TEXT 段

注：光标等都是有用户程序控制的所以在上面没有展示
我们的 os 会实现虚拟内存，暂定物理内存为 1M，一个页表 512 个字节，我们会根据实际的 3e 板上的空间进行调整；磁盘读写，我们将采用 FAT 格式的文件系统，具体的数据结构还没有设计。

三、内部命令

我们的系统会支持文件系统因此有：

ls 列出所有的文件
rm 删除文件
touch 新建文件
cat 显示文件内容
mv 文件重命名

同时也有

time 查看开机时间
run 可以打开某个磁盘文件，并且运行程序（格式必须是我们定义格式的文件才行）

四、应用程序

我们将完成一个文本编辑器，可能会对一种语言，比如 C 或者 mips 进行代码高亮。

五、小组成员及任务分配，具体实现的简要的说明

首先简要说明分工（到目前为止）：
陈译： 汇编器，模拟器的 debug 功能
朱霖潮： 模拟器模拟 cpu，包括指令处理 syscall, exception,io 等
汪乾文： VGA 设计+键盘
魏铭： 硬件 cpu 指令集实现

陈译：

从零开始写汇编器，根据课程要求，汇编器提供了对大多数指令和部分伪指令的支持。汇编器接受 mips 代码文本的文件名，对其进行汇编工作，将得到的二进制码写入二进制文件中。

我们讲整个程序分为头部，数据段和代码段。头部主要提供数据段起始位置，数据段长度，代码段起始位置等信息，格式由事先讨论预定如下：
头部：
头部的长度
Data 段的长度
Data 段的开始地址
Text 段的长度
Text 段的开始地址
Debug 信息的长度
Debug 信息的开始地址
每个参数都是 2byte

接着是数据段，和代码段，代码段中如果需要引用数据段的内容，只需到数据段相应地址处寻找即可。操作系统把程序装载到内存后，程序开始地址是 0040 0000h 开始，已初始化的数据从 1001 0000h 开始，未初始化的数据从 1000 0000h 开始，sp 初始化为 7fffffffh。
得到了汇编后的二进制文件，此时是无法做 debug 操作的。为此，我在二进制文件的末尾加上 debug 信息。其格式定为：
代码： add $t1,$t1, $t2 + 机器码的地址（如果是伪指令，那么将配上多个机器码）

有了这些 debug 信息，我又在此基础上，写好 debug 程序。

此 debug 程序的要点就是实现单步调试，和设置断点后调试。实现的方法是，将需要停住打印寄存器信息的代码的下一行修改成一个会打印各个寄存器信息并暂停等待键盘输入的中断，这个中断由写系统调用的同学实现，我只需要调用即可。当调用好用户按了一下键盘，我就再次把原来的机器码写回去，cpu 再

次执行改行代码，程序正常继续。

汪乾文：

VGA 设计：屏幕扫描，通过指定的参数顺利实现将显存里的内容通过 VGA 扫描到计算机屏幕上。

键盘读取设计：当键盘按键时，通过 PS/2 端口，用其端口协议，将键盘产生的信号，串行读入到硬件中，然后放到自己定义（非内存）指定的位置中。

显存图形模式和字符模式的编写：

字符模式：将内存指定段的数据最高位设为是否有效，低 7 位视为 ascii 码，将指定位置的 ascii 码与指定字模（字模库存放在非内存的地方）对应，然后打印到屏幕指定位置，其中指针闪烁的 ascii 码为 127 号

图形模式：将内存指定字段的内容每 4bit（1bit 为空，其他 3bit 分别为 RGB）视为一个屏幕的一个点，然后将点根据对应的颜色放到指定位置。

字符模式还是图形模式从内存中的指定字段中读取。

魏铭：

CPU 的 MIPS 指令集实现：按照前面制定的 MIPS 指令实现表格，在硬件中实现。

朱霖潮：

实现模拟器。支持我们规定的所有指令的模拟执行。处理 syscall，键盘的 IO，同时实现‘操作系统’加载程序到内存空间，分析文件结构，分配内存等。还有包括内存映射，中断和异常的机制的模拟。