**Serenity MIPS语言规范**

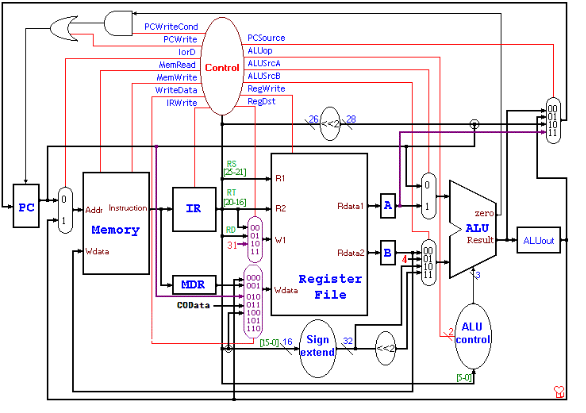
**小组成员：海杰文、李其迈、蔡武威、胡雪燕**

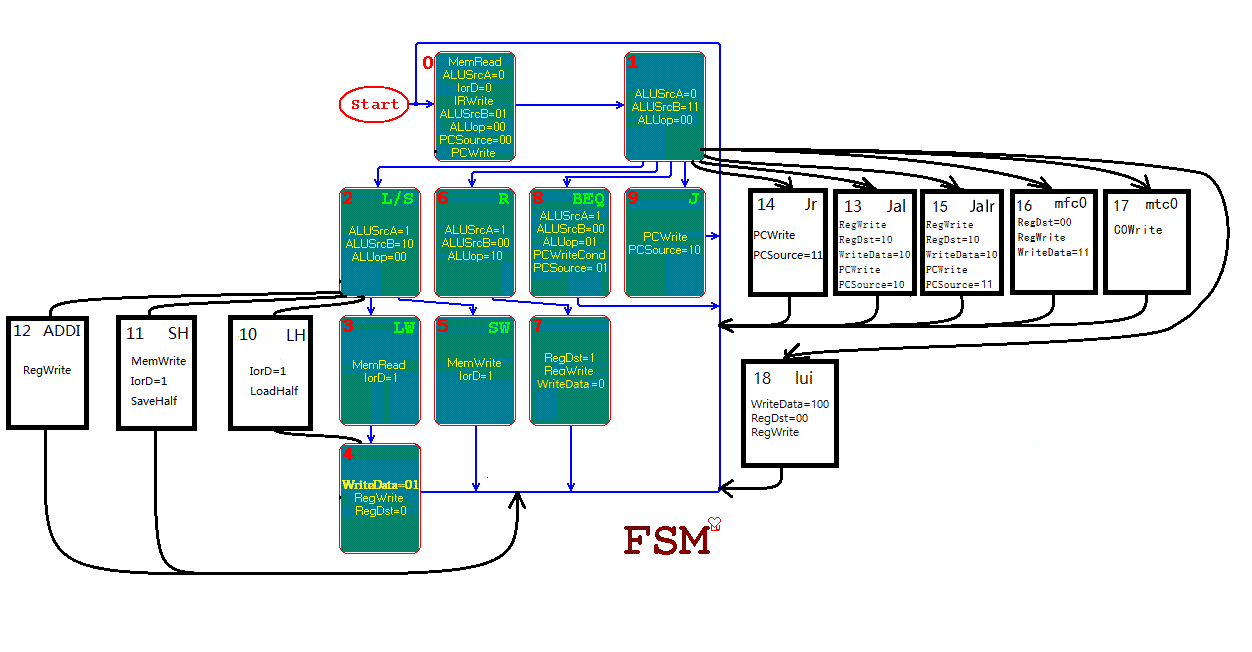
* **前言**

每一个特殊的架构，都需要为其特别准备适合自己的指令集，适合的指令集不但可以简化设计、理清思路，而且可以提升CPU的运行效率。此外，CPU架构与指令集的相互依赖，也造成了MIPS汇编语言的特殊性，这使得我们必须根据自己的需要和设计来规定我们的MIPS语言规范。下面就来介绍Serenity MIPS的具体规范。

* **MIPS汇编指令集**

下图为SerenityPC的系统架构图，为了实现lui指令，我们多加了若干控制信号，并添加了导向寄存器的数据，将指令寄存器第十六位作为高十六位输入，而低十六位输入为16’B0。



此外，为了实现DDR，我们为控制单元加入了rdy信号，只有当rdy信号为1时，CPU的状态才能转移。而中断信号的扫描在0状态完成，若有探测到终端，系统会将EPC和PC分别置好，然后重新回到0状态。

下表为目前CPU支持的指令集，一共36条，可以满足我们的编程需求。

|  |  |  |
| --- | --- | --- |
| **功能** | **格式** | **执行** |
| 逻辑左移 | sll rd,rt,sft | rd=rt<<sft |
| 逻辑右移 | srl rd,rt,sft | rd=rt>>sft |
| 算术右移 | sra rd,rt,sft | rd=rt>>>sft |
| 逻辑左移 | sllv rd,rt,rs | rd=rt<<rs |
| 逻辑右移 | srlv rd,rt,rs | rd=rt>>rs |
| 算术右移 | srav rd,rt,rs | rd=rt>>>rs |
| 寄存器转移 | jr rs | PC=$rs |
| 寄存器转移 | jalr rs,rd | $rd=PC+4, PC=$rs |
| 加法，有溢出 | add rd,rs,rt | rd=rs+rt |
| 加法，不溢出 | addu rd,rs,rt | rd=rs+rt |
| 减法，有溢出 | sub rd,rs,rt | rd=rs-rt |
| 减法，不溢出 | subu rd,rs,rt | rd=rs-rt |
| 与 | and rd,rs,rt | rd=rs&rt |
| 或 | or rd,rs,rt | rd=rs|rt |
| 异或 | xor rd,rs,rt | rd=rs^rt |
| 或非 | nor rd,rs,rt | rd=~(rs|rt) |
| 比较 | slt rd,rs,rt | rd=0;if(rs<rt)rd=1; |
| 小于无符号数 | sltu rd,rs,rt | rd=0;if(rs<rt)rd=1; |
| 转移 | j L | PC=PC高4|(L<<2) |
| 调子程序 | jal L | $ra=PC+4, j L |
| 相等转移 | beq rs,rt,L | PC+=4;if(rs==rt)PC+=L |
| 立即数加，有溢出 | addi rt,rs,imm | rt=rs+imm |
| 立即数加，不溢出 | addiu rt,rs,imm | rt=rs+imm |
| 小于立即数 | slti rt,rs,imm | rd=0;if(rs<imm)rt=1; |
| 小于无符号数 | sltiu rt,rs,imm | rd=0;if(rs<imm)rt=1; |
| 立即数与 | addi rt,rs,imm | rt=rs&imm |
| 立即数或 | ori rt,rs,imm |  |
| 立即数异或 | xori rt,rs,imm | rt=rs^imm |
| 高位立即数 | lui rt,imm | rt高=imm |
| 读协0寄存器 | mfc0 rt,rd | $rt=协0.rd |
| 写协0寄存器 | mtc0 rt,rd | 协0.rd=rd |
| 短乘法 | mul rd,rs,rt | rd=rs\*rt |
| 读2字节 | lh rt,D(rs) | rt=Memory[rs+D] |
| 读字(4字节) | lw rt,D(rs) | rt=Memory[rs+D] |
| 写2字节 | sh rt,D(rs) | Memory[D+rs]=rt |
| 写字(4字节) | sw rt,D(rs) | Memory[D+rs]=rt |

* **伪指令**

下表是目前汇编器支持的所有伪指令，一共14条。这些指令可以大大方便我们的MIPS汇编编程，也能提高可读性。

|  |  |  |
| --- | --- | --- |
| **功能** | **格式** | **执行** |
| 寄存器间数据转移 | move rd,rs | rd = rs |
| 逻辑按位取反 | not rd, rs | rd = ~rs |
| 取负数 | neg rd, rs | rd = -rs |
| 压栈 | push $r | $r进栈 |
| 出栈 | pop $r | 出栈到$r |
| 比较转移，小于 | blt rs, rt, rr | if (rs < rt) goto rr |
| 比较转移，大于 | bgt rs, rt, rr | if (rs > rt) goto rr |
| 比较转移，小于等于 | ble rs, rt, rr | if (rs <= rt) goto rr |
| 比较转移，大于等于 | bge rs, rt, rr | if (rs >= rt) goto rr |
| 绝对值 | abs rs, rt | rs=|rt| |
| 交换 | swap rs, rt | rs <->rt |
| 判断不等 | sne rd, rs, rt | rd = (rs != rt) ? 1 : 0 |
| 判断相等 | seq rd, rs, rt | rd = (rs == rt) ? 1 : 0 |
| 取地址 | la $r, 标号 | $r = 标号地址 |

* **格式指令**

|  |  |  |
| --- | --- | --- |
| **功能** | **格式** | **执行** |
| 指定初始地址 | .origin addr | 随后的程序被加载到指定地址 |
| 定义字符串 | .asciiz “<>” | 0结束ascii字符串变量 |
| 定义数据变量 | .2byte <> | 2字节变量,顺序填充 |

* **表达式**

由于MIPS的特殊架构的特殊性，我们只支持最为基础的指令结构。类似于X86里使用[]来进行取内容操作并不打算被实现。表达式功能的特性就注定其不能在汇编语言执行的时候存在，必须要被汇编器翻译为指令。

而对于汇编器层面，能操纵的只有立即数与标号。对它们使用表达式并没有太大意义，立即数能够事先动态，而标号我们则采用lea指令来取得其地址，这样就可以在汇编代码里使用了。

出于以上两点原因，我们没有设计汇编器解析表达式的功能。

* **错误信息**

支持报错功能，报错时以行为单位，每次只能报一行的一种错误。报错时输出如下：

*Line <lineNumber> : <instruction>*

*<errorInfo>*

*lineNumber*为出错的汇编码行号（伪指令展开前，空行也算行）

*errorInfo*为错误信息，有以下几种

1. ***No error*** 没有错误，前面不会有行号
2. ***Wrong formation*** 各种格式错误，如除eret和syscall外的指令只有指令名而无操作数；指令名和操作数间出现逗号，如”add, $s1,$s2,$s3”；操作数与操作数间无分隔符”,“或括号，如”add $s1 $s2 $s3”；重复的分隔符，如”add $s1,,$s2,$s3”；逗号出现在括号间，如”lw $s1,0(,$s2)”；括号不匹配或出现多于一个括号”lw $s1,0(($s2)”；括号间无操作数，如”lw $s1,0,()”；逗号后无操作数,如“add $s1,$s2,$s3,”……
3. ***Illegal characters in label*** label中有非法字符（合法字符为大小写字母、数字和下划线）
4. ***Label duplication*** 连续的label，即两个label连在一起，如：

*Line 0：RR*

*Line 1：RRR syscall*

报错为：

*Line 1：RRR syscall*

*Label duplication*

1. ***The amount of operand is wrong*** 操作数的数量错误，如add指令本应恰有3个操作数，因此add $s1,$s2，操作数的数量错误
2. ***No such register*** 没有这样的寄存器，如某行若有非法寄存器名“$r1”，改行则会报此类错误
3. **No such instruction or pseudo instruction** 没有这样的指令或伪指令，即指令名不合法或不存在
4. ***Redefined label*** 重定义的label，当同一个label出现第二次时在第二次那行报错，如

*Line 0: RR: add $s0,$s1,$s2*

……

*Line 3: RR: sw $s0,0($s2)*

报错为：

*Line 3: RR: sw $s0,0($s2)*

*No such register*

1. ***Wrong immediate number or offset*** 错误（超过位数或出现非法字符）的立即数或偏移（目前仅在li指令中报此类错误）
2. ***No such label*** 没有这样的label，即找不到在跳转类指令中的label
3. ***Can not assemble*** 其余已被发现的错误
4. ***Unknown error*** 未知错误

* **例程**

*addi $s0, $zero, 0x0c00*

*addi $s1, $zero, 26*

*addi $t0, $zero, 0x40*

*loop: sh $t0,0($s0)*

*addi $t0, $t0, 1*

*addi $s1, $s1, -1*

*beq $s1, $zero, exit*

*j loop*

*exit: sll $t0, $t0, 3*

*sllv $t0, $t0, $ra*

*sra $t0, $t0, 1*