## MIPS处理器设计实验

## 实验目的

* 进一步加深对运算器、存储器及数据通路的理解。
* 掌握硬布线控制器设计原理。
* 为整机实验以及课程设计做准备。

## 实验内容

利用运算器实验，存储系统实验中构建的运算器、寄存器文件、存储系统等部件以及Logisim中其他功能部件构建一个32位MIPS CPU单周期处理器，该处理器应支持基础指令集中列出的所有指令，见表1，另外还必须支持扩展指令集中的2条C类运算指令，1条M类存储指令，1条B类分支指令（详见表2），具体任务每位同学不一样，任务要求见任务分配清单，每位同学得到一个4位数据表示的任务编号，分别对应CCMB指令的选择。具体指令功能参见附件中的MIPS标准文档。最终设计完成的CPU应能运行教师提供的标准测试程序，程序存储在Logisim ROM模块中（指令存储器、数据存储器分开）。

表1 基础指令集

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **指令** | **格式** | **备注** |
| 1 | Add | add $rd, $rs, $rt | 指令功能及指令格式  参考MIPS32指令集 |
| 2 | Add Immediate | addi $rt, $rs, immediate |
| 3 | Add Immediate Unsigned | addiu $rt, $rs, immediate |
| 4 | Add Unsigned | addu $rd, $rs, $rt |
| 5 | And | and $rd, $rs, $rt |
| 6 | And Immediate | andi $rt, $rs, immediate |
| 7 | Shift Left Logical | sll $rd, $rt, shamt |
| 8 | Shift Right Arithmetic | sra $rd, $rt, shamt |
| 9 | Shift Right Logical | srl $rd, $rt, shamt |
| 10 | Sub | sub $rd, $rs, $rt |
| 11 | Or | or $rd, $rs, $rt |
| 12 | Or Immediate | ori $rt, $rs, immediate |
| 13 | Nor | nor $rd, $rs, $rt |
| 14 | Load Word | lw $rt, offset($rs) |
| 15 | Store Word | sw $rt, offset($rs) |
| 16 | Branch on Equal | beq $rs, $rt, label |
| 17 | Branch on Not Equal | bne $rs, $rt, label |
| 18 | Set Less Than | slt $rd, $rs, $rt |
| 19 | Set Less Than Immediate | slti $rt, $rs, immediate |
| 20 | Set Less Than Unsigned | sltu $rd, $rs, $rt |
| 21 | Jump | j label |
| 22 | Jump and Link | jal label |
| 23 | Jump Register | jr $rs |
| 24 | syscall（display or exit） | syscall | If $v0==10  halt(停机指令)  else  数码管显示$a0值  注意数码管输入数据应该用寄存器锁存 |

表2 扩展指令集

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **指令分类** | **编号** | **指令助记符** | **简单功能描述** | **备注** |
| **运算**  **(C)** | 1 | SLLV | 逻辑可变左移 | 指令格式参考MIPS32指令集，最终功能以MARS模拟器为准。 |
| 2 | SRLV | 逻辑可变右移 |
| 3 | SRAV | 算术可变右移 |
| 4 | SUBU | 无符号减 |
| 5 | XOR | 异或 |
| **6** | XORI | 异或立即数 |
| 7 | LUI | 立即数加载至高位 |
| **8** | SLTIU | 小于立即数置 1(无符号) |
| 9 | MULTU | 乘无符号 |
| A | DIVU | 无符号除 |
| B | MFLO | 读 LO 寄存器 |
| **存储**  **访问**  **（M）** | 1 | LB | 加载字节 |
| **2** | LBU | 加载字节(无符号) |
| 3 | LH | 加载半字 |
| 4 | LHU | 加载半字(无符号) |
| 5 | SB | 存储字节 |
| 6 | SH | 存储半字 |
| **跳转**  **(B)** | **1** | BLEZ | 小于等于 0 转移 |
| 2 | BGTZ | 大于 0 转移 |
| 3 | BLTZ | 小于 0 转移 |
| 4 | BGEZ | 大于等于 0 转移 |

## 实验步骤

**1、简单迭代法**

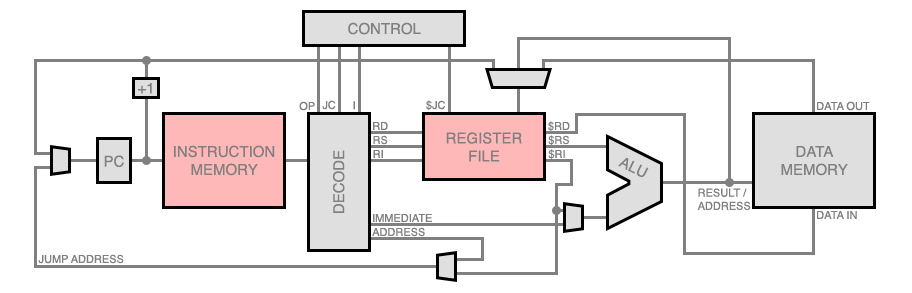


图1 单周期CPU示意图

1. 可以首先完成取下一条指令的逻辑,即PC(程序计数器)的逻辑,最初你可以简单的实现,只用让该寄存器的值,在每次时钟信号到达时加一即可，以后再考虑更复杂的情形。
2. 也许此时你可以用上面构建的模块，搭建一个最简单的CPU了，该CPU只能进行加法运算。你可以先做一个大致的模型，实现加法。而我们希望是，你的CPU应该包括以下器件：

* ALU、寄存器文件
* PC及PC+4的逻辑。
* 指令内存（为了简单，建议你使用系统提供的ROM，而不是RAM，ROM可以方便的加载镜像，重启后也不需要单独加载，RAM重启程序，或者CTRL+R后数据清零），由于mips 32位地址总线是按字节编址，而系统提供的ROM是32位的，且ROM地址总线也无法达到32位，所以连接时可以将高位地址屏蔽，低位字节偏移地址也进行屏蔽，使得取指令工作能正常进行。

试着将以下汇编指令翻译成机器码：

nor $s0, $0, $1  
然后将机器码存入指令ROM中。如果你成功地完成了此最简单的CPU，则时钟每跳一次，将执行一条指令。

1. 你可以进一步编写一些新的CPU指令，如SUB，AND，OR等，来对你前面做的工作进行一些进一步的测试。当然，对于SUB，AND，OR，你可能需要修改ALU的相应控制，否则，还是在做nor。
2. 你可以编写更多的模块，如零扩展和符号扩展，然后将其组合成一个通用的扩展器,这样就可以实现立即数的运算了。
3. 你可以加入LOAD及STORE逻辑。当然，这里需要加入数据内存了。
4. 不同mips指令的数据通路，大家可以参考mips仿真器MARS中的x ray功能观看，该仿真器可以动态演示指令的数据通路以及对应的控制信号生成。

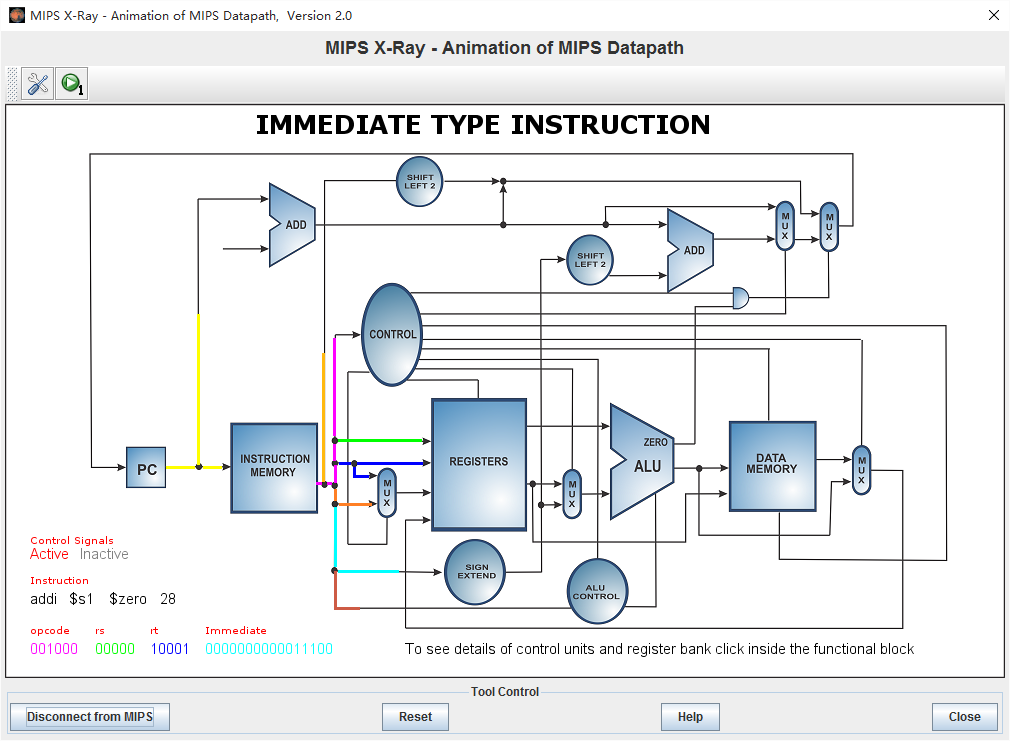
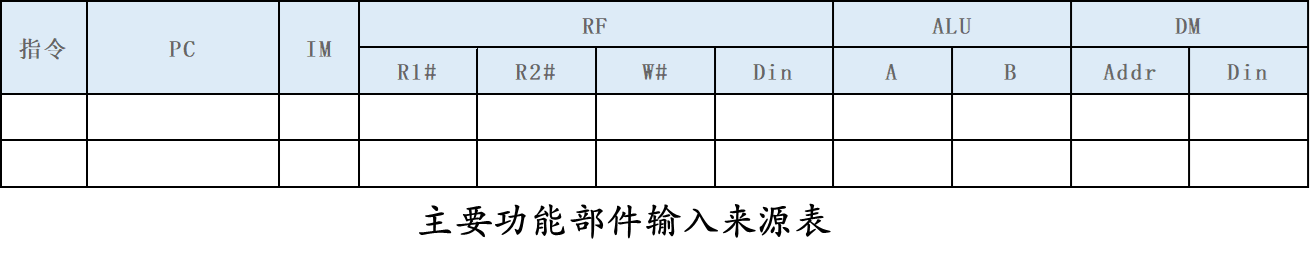


图2 显示部件示意图

1. 也许你可以把所有的工作集成在一起，完成DATAPATH（数据通道）了，注意在进行数据通路集成时会发现很多输入有多个来源，此时需要加入多路选择器，选择器的控制端成为新的控点或微命令。
2. 根据前述步骤完成的数据通路以及对应的控点，设计控制器，控制器应该进行封装，控制器封装完毕你的单周期CPU就完成了（**不建议使用真值表**生成控制器，最好将每个控点的逻辑表达式写出来，可以各处最小项之和，LOGISIM可以帮你化简）。

**2、工程化方法（推荐）**

1. **构建主要功能部件。**在logisim中选择相应部件构建PC寄存器，指令存储器IM，数据存储器DM，立即数扩展器，地址转移逻辑NPC，运算器ALU（运算器实验已经实现），寄存器文件(已提供标准库)。
2. **一次性构建所有数据通路。**绘制主要功能部件输入来源表，该表主要用于描述各部件之间的连接关系，记录各部件输入端数据来源，注意这里忽略控制类信号，仅保留数据类信号，具体如下表所示：



逐条分析每一条指令的功能，根据指令功能填写表格，如发现有新的需求，可以在表格中引入新的部件，并设置其输入来源。

1. **输入源合并。**将主要功能部件输入来源表中功能部件输入按列进行合并，如某个输入有多个输入来源，引入多路选择器，同时新增多路选择器选择控制信号，对于空输入如其他输入不影响指令执行，可直接忽略，否则需要增加额外的控制电路。根据合并后的输入来源在logisim中连接各主要功能部件，实现MIPS CPU所有数据通路。
2. **控制信号综合。**列出所有控制信号的产生条件，如下表，给出各控制信号逻辑表达式，利用译码电路生成各指令译码信号，然后根据相应逻辑表达式生成控制器，并封装。

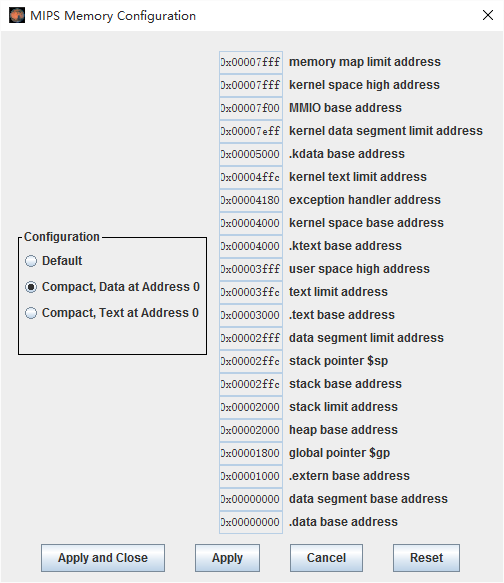


1. **控制信号综合。**列出所有控制信号的产生条件，如下表，给出各控制信号逻辑表达式，利用译码电路生成各指令译码信号，然后根据相应逻辑表达式生成控制器，并封装，最后实现其他新增的功能部件。
2. **系统调试。**撰写简单测试程序逐条对各指令进行测试，测试完毕后运行标准测试程序进行最终的验收测试。

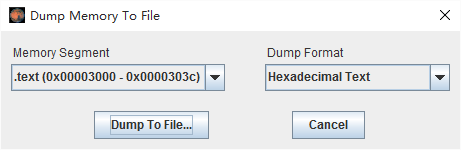
**2、检查要求**

1. 能自动运行如下排序测试程序，
2. 将测试文件“benchmark.hex”通过文件加载到指令存储器，该程序自动写入数据到存储器，并将存储器0-15号单元降序排序。程序演示效果
3. 能单步演示和自动运行
4. 编写一段小程序能演示CCMB指令的正确运行。

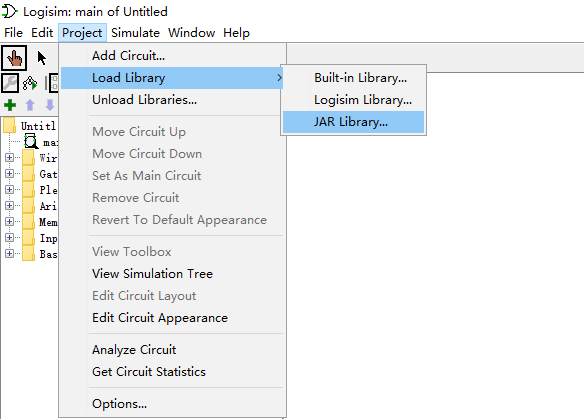
**3、汇编器说明**

汇编器采用附件包中的mars仿真器，该仿真器功能强大，请主动学习之。注意为了能让mars中汇编的机器码能在logisim中实用，需要设置mars界面中setting的Memory Configration，将内存模式设置为下图的模式，这样数据段起始位置就是0开始的位置。

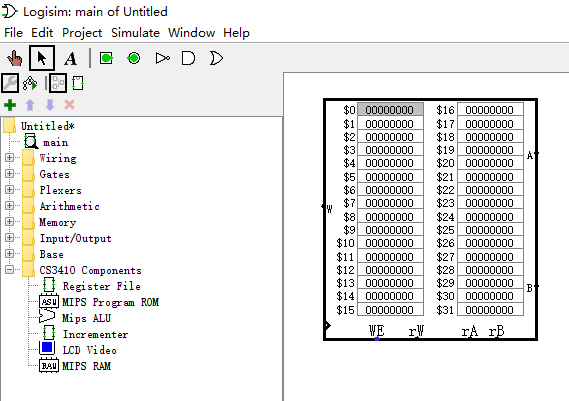
程序汇编后可以利用File菜单中的Dump Memory功能将代码段和数据段导出，采用十六进制文本的方式导出到某个文本文件，然后在文件第一行加入“v2.0 raw”即可在Logisim中加载到ROM或RAM。



**4、MIPS寄存器文件加载**

存储系统实验中我们仅仅设计了包含4个寄存器的寄存器文件，这里我们提供了一个包含32个寄存器的MIPS寄存器文件的库CS3410.jar,库加载方式如下图所示。

加载成功后logisim左下角的组件库会增加CS3410的选项，其中第一个组件就是寄存器文件，注意新增的库还包含一些其他的库，课酌情考虑使用，也可不用。



## 实验报告要求

1. 实验目的；
2. 各模块的设计电路和系统的整体电路,对设计要进行详细的分析与说明；
3. 列出操作步骤及顺序,标出重要的开关控制端；给出各控制信号逻辑表达式以及电路。
4. 实验结果的记录与分析；
5. 列出操作步骤及顺序,标出重要的开关控制端；
6. 实验收获和体会；
7. 实验中碰到的问题和解决的方法。

注：本文档有些的不全面、不完整，希望同学们修正。