#	MIPS 指令	RTL 功能描述
C1	sllv $rd,$rt,$rs	R[$rd]←R[$rt]<<R[$rs] 可变左移
C2	srlv $rd,$rt,$rs	R[$rd]←R[$rt]>>R[$rs] 逻辑可变右移
C3	srav $rd,$rt,$rs	R[$rd]←R[$rt]>>R[$rs] 算术可变右移
C4	subu $rd,$rs,$rt	R[$rd]←R[$rs]-R[$rt] 无符号减
C5	xor $rd,$rs,$rt	R[$rd]←R[$rs]^R[$rt] 异或
C6	xori $rt,$rs,imm	R[$rt]←R[$rs]^{0×16,imm}
C7	lui $rt,imm	R[$rt]←{(imm)[15:0],0×16}
C8	sltiu $rt,$rs,imm	R[$rt]←R[$rs]<$SignExt_{16b}$(imm) 无符号比较
C9	multu $rs,$rt	{HI,LO}←R[$rs]*R[$rt] 无符号乘
CA	divu $rs,$rt	LO←R[$rs]/R[$rt] HI←R[$rs]%R[$rt] 无符号除法
CB	mflo $rd	R[$rd]←LO 取 LO 寄存器的值
M1	lb $rt,imm($rs)	R[$rt]←$SignExt_{8b}$($Mem_{1B}$(R[$rs]+$SignExt_{16b}$(imm)))
M2	lh $rt,imm($rs)	R[$rt]←$SignExt_{16b}$($Mem_{2B}$(R[$rs]+$SignExt_{16b}$(imm)))
M3	lbu $rt,imm($rs)	R[$rt]←{0×24,$Mem_{1B}$(R[$rs]+$SignExt_{16b}$(imm))}
M4	lhu $rt,imm($rs)	R[$rt]←{0×16,$Mem_{2B}$(R[$rs]+$SignExt_{16b}$(imm))}
M5	sb $rt,imm($rs)	Mem_{1B}(R[$rs]+$SignExt_{16b}$(imm))←(R[$rt])[7:0]
M6	sh $rt,imm($rs)	Mem_{2B}(R[$rs]+$SignExt_{16b}$(imm))←(R[$rt])[15:0]
B1	blez $rs,imm	if(R[$rs]<=0) PC←PC+$SignExt_{18b}$({imm, 00}) 有符号比较
B2	bgtz $rs,imm	if(R[$rs]>0) PC←PC+$SignExt_{18b}$({imm,00}) 有符号比较
B3	bltz $rs,imm	if(R[$rs]<0) PC←PC+$SignExt_{18b}$({imm, 00}) 有符号比较
B4	bgez $rs,imm	if(R[$rs]>=0)PC←PC+$SignExt_{18b}$({imm,00}) 有符号比较

每位同学应额外扩展 4 条指令，包括 2 条 C 类指令、1 条 M 类指令、1 条 B 类指令，简称 CCMB 系列指令。具体每位同学的指令要求见任务分配，最终实现的 MIPS 处理器能正确运行标准测试程序 benchmark.asm，要求修改 syscall 系统调用的功能，当系统调用号不等于 34 时，一律是暂停当前程序的执行，而不是停机，暂停后等待用户按下电路中的 Go 按钮才能继续运行。实验包中已经包括了所有 CCMB 系列指令的测试程序，请将对应的测试程序放置在 benchmark.asm 尾部以方便教师检查。

1.1.1 注意事项

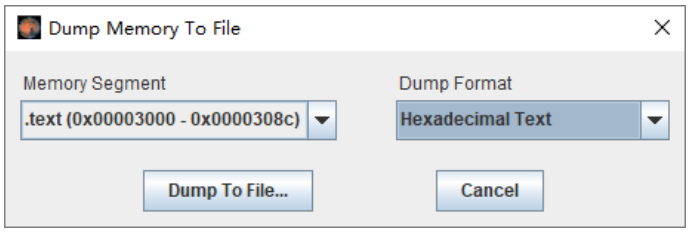
1）MIPS 虚存模式设置

由于实验中设计的单周期 MIPS 处理器并不包括内存管理单元 MMU 部件，所以程序运行时的访存地址均是物理地址，设计时采用指令存储器和数据存储器分离的哈佛架构。访问数据时应该访问数据存储器，数据存储器的地址起始地址是 0，实验包中的测试程序均直接使用了 0 号地址开始的数据单元，为了使测试程序在 MARS 和实验设计的处理器中的运行结果保持一致，必须配置 MARS 模拟器中的 MIPS 虚存模式，具体可以选择菜单 Setting 的 Memory Configration 选项，该选项可以设置 MIPS 虚拟地址空间的地址模式，这里应将虚存模式设置为 Compact 模式，这样数据段起始位置 0 开始的位置，如图错误**!文档中没有指定样式的文字。**.1 所示。注意，如果采用 Default 模式在 MARS 中运行 sort.asm 排序程序时访存指令会越界访问代码段或内核段，引起保护错。

图**错误!文档中没有指定样式的文字。**.1 MIPS 虚存模式配置

2）MIPS 机器指令导出

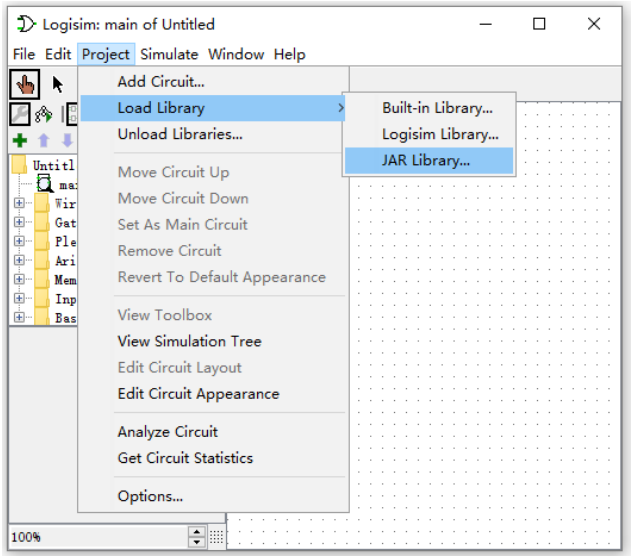
在 MARS 中可以利用 File 菜单中的 Dump Memory 功能将汇编程序的代码段的机器指令和数据段的数据导出，为了能直接在 Logisim 的 RAM 和 ROM 组件中宋使用，推荐采用十六进制文本的方式导出到某个文本文件，然后在文本文件第一行加入“v2.0 raw”，这样导出的文件即可直接加载到 Logisim 平台的 ROM 和 RAM 组件中，如图错误**!文档中没有指定样式的文字。**.2 所示。



图**错误!文档中没有指定样式的文字。**.2 MIPS 代码导出

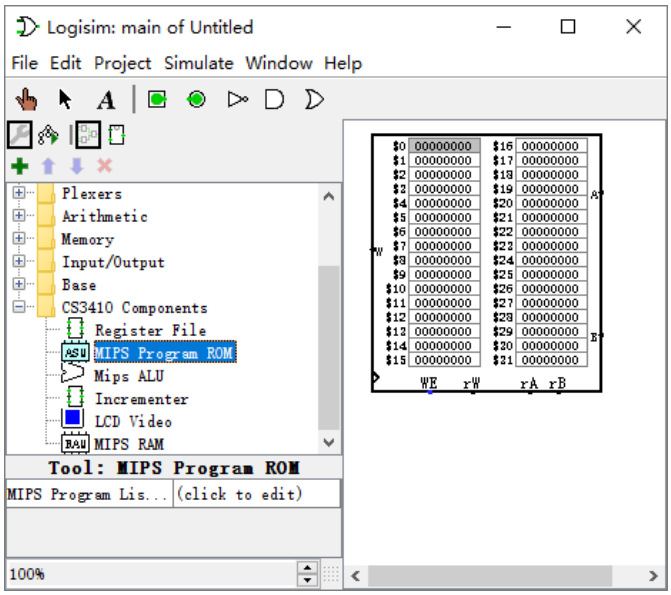
4、MIPS 寄存器文件库

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，为满足 MIPS 单周期处理器设计的需要，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，该库由美国康奈尔大学开发，在 Logisim 平台中可以通过加载 JAR 库的方式加载第三方 JAVA 库，如图错误**!文档中没有指定样式的文字。**.3 所示。



图**错误!文档中没有指定样式的文字。**.3 JAR 库加载

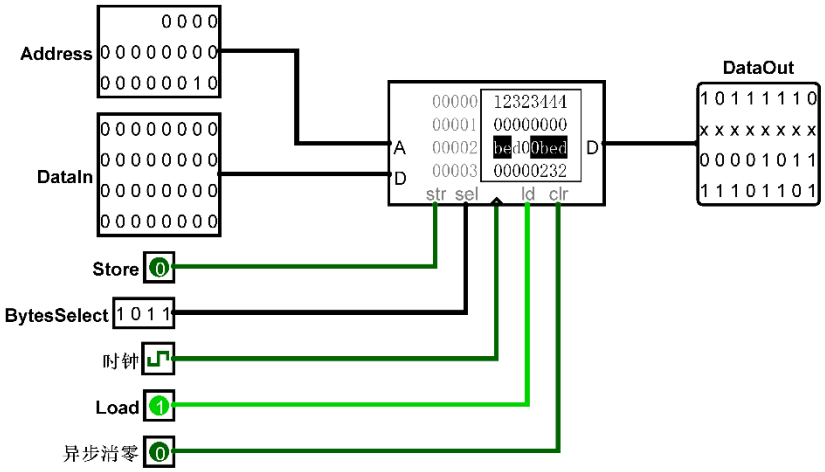
选择实验包中的 CS3410.jar 库，加载成功后 logisim 左下角的组件库会增加 CS3410 组件的选项，其中第一个组件就是寄存器文件，添加该组件到画布中，如图错误**!文档中没有指定样式的文字。**.4 所示，这个寄存器文件的引脚 rA，rB 为读寄存器编号，rW 为写入寄存器编号，WE 为写使能，W 为写入数据，A、B 为 rA，rB 寄存器的输出值，该组件直接提供了 32 个寄存器观察窗口，非常直观，用户还可以使用手型的戳工具直接修改寄存器的值。需要注意的该组件缩小显示时组建上数字的显示可能不正常。



图**错误!文档中没有指定样式的文字。**.4 MIPS 寄存器文件库

4、MIPS RAM 库

在存储系统实验中我们设计过 MIPS RAM 电路，CS3410 库中也提供了一个标准的 MIPS RAM 组件抽象，在实现 LB、SB、LH、SH 等字节访问或半字访问指令时非常有用。该组件的访问接口如图**错误!文档中没有指定样式的文字。**.5 所示，各引脚功能说明如下：



图错误!文档中没有指定样式的文字。.5 MIPS RAM 组件

- A 是 20 位字地址输入，MIPS RAM 组件中的每一行代表内存中的 4 个字节，图中 A 端地址为 0x00002，MIPS RAM 中第二行用黑底白字显示，0xbed00bed 为 2 号字单元的值，如果需要访问当前地址中具体某个字节或半字，需要设置 MIPS RAM 组件底部的 Sel 输入端口。缺省地址为 20 位，可以访问 1MWord=4MB。
- 左侧 D 为数据输入端，位宽 32 位， 包含即将写入 RAM 的数据，具体写入时会根据 sel 端的设置决定数据输入端的哪些字节被写入。
- str 是存储控制位，类似于写使能，当 str=1 时，RAM 会根据 sel 的值写入 D 中的某些字节。
- sel 是选择控制字段，线宽 4 位，当 sel=0001 时，D 端口的 0~7 位被选中，当 sel = 0010, D 端口的 8~15 位被选中，当 sel = 0100 时, D 端口 16~23 位被选中，当 Sel=1000 时，D 端口 24~31 位被选中。另外 SEL=0011，表示 0~15 位被选中，SEL=1100，表示 16~31 位被选中，SEL=1111 时，4 个字节同时被选中。
- ld 是载入位，ld 为 1 时组件右侧的数据输出端口 D 输出当前字中被选择的字节，如果 ld=0，输出为高阻态。
- clr 为 1 时会清空 RAM 组件的所有内容。