第一部分
基于 MIPS 体系结构

实验 1　MIPS 指令系统和 MIPS 体系结构

1.1　实验目的

(1) 了解和熟悉指令级模拟器。

(2) 熟练掌握 MIPSsim 模拟器的操作和使用方法。

(3) 熟悉 MIPS 指令系统及其特点,加深对 MIPS 指令操作语义的理解。

(4) 熟悉 MIPS 体系结构。

1.2　实验平台

实验平台采用指令级和流水线操作级模拟器 MIPSsim(随书光盘中提供)。

设计:张晨曦教授,版权所有。

开发:孙太一。

1.3　实验内容和步骤

首先要阅读 MIPSsim 模拟器的使用方法(见 1.4 节),然后了解 MIPSsim 的指令系统和汇编语言(见附录 A、附录 B 和附录 C)。

(1) 启动 MIPSsim(用鼠标双击 MIPSsim.exe)。

(2) 选择“配置”→“流水方式”选项,使模拟器工作在非流水方式下。

(3) 参照 1.4 节的使用说明,熟悉 MIPSsim 模拟器的操作和使用方法。

可以先载入一个样例程序(在本模拟器所在的文件夹下的“样例程序”文件夹中),然后分别以单步执行一条指令、执行多条指令、连续执行、设置断点等的方式运行程序,观察程序的执行情况,观察 CPU 中寄存器和存储器的内容的变化。

(4) 选择“文件”→“载入程序”选项,加载样例程序 alltest.asm,然后查看“代码”窗口,查看程序所在的位置(起始地址为 0x00000100)。

(5) 查看“寄存器”窗口 PC 寄存器的值:[PC]=0x_____。

(6) 执行 load 和 store 指令,步骤如下:

① 单步执行 1 条指令(F7)。

② 下一条指令地址为0x_____,是一条_____(有,无)符号载入_____(字节,半字,字)指令。

③ 单步执行 1 条指令(F5)。

④ 查看 R1 的值,[R1]=0x_____。

⑤ 下一条指令地址为0x_____，是一条_____(有，无)符号载入_____(字，半字，字)指令。

⑥ 单步执行1条指令。

⑦ 查看R1的值，[R1]=0x_____。

⑧ 下一条指令地址为0x_____，是一条_____(有，无)符号载入_____(字，半字，字)指令。

⑨ 单步执行1条指令。

⑩ 查看R1的值，[R1]= 0x_____。

⑪ 单步执行1条指令。

⑫ 下一条指令地址为0x_____，是一条保存_____(字，半字，字)指令。

⑬ 单步执行1条指令(F5)。

⑭ 查看内存BUFFER处字的值，值为0x_____。

(7) 执行算术运算类指令。步骤如下：

① 双击“寄存器”窗口中的R1，将其值修改为2。

② 双击“寄存器”窗口中的R2，将其值修改为3。

③ 单步执行1条指令。

④ 下一条指令地址为0x_____，是一条加法指令。

⑤ 单步执行1条指令。

⑥ 查看R3的值，[R3]=0x_____。

⑦ 下一条指令地址为0x_____，是一条乘法指令。

⑧ 单步执行1条指令。

⑨ 查看LO、HI的值，[LO]=0x_____，[HI]=0x_____。

(8) 执行逻辑运算类指令。步骤如下：

① 双击“寄存器”窗口中的R1，将其值修改为0xFFFF0000。

② 双击“寄存器”窗口中的R2，将其值修改为0xFF00FF00。

③ 单步执行1条指令。

④ 下一条指令地址为0x_____，是一条逻辑与运算指令，第二个操作数寻址方式是_____(寄存器直接寻址，立即数寻址)。

⑤ 单步执行1条指令。

⑥ 查看R3的值，[R3]=0x_____。

⑦ 下一条指令地址为0x_____，是一条逻辑或指令，第二个操作数寻址方式是_____(寄存器直接寻址，立即数寻址)。

⑧ 单步执行1条指令。

⑨ 查看R3的值，[R3]=0x_____。

(9) 执行控制转移类指令。步骤如下：

① 双击“寄存器”窗口中的R1，将其值修改为2。

② 双击“寄存器”窗口中的R2，将其值修改为2。

③ 单步执行1条指令。

④ 下一条指令地址为0x_____，是一条BEQ指令，其测试条件是_____，目

标地址为0x_____。

⑤ 单步执行1条指令。

⑥ 查看PC的值,[PC]=0x_____,表明分支_____(成功,失败)。

⑦ 下一条指令是一条BGEZ指令,其测试条件是_____,目标地址为0x_____。

⑧ 单步执行1条指令。

⑨ 查看PC的值,[PC]=0x_____,表明分支_____(成功,失败)。

⑩ 下一条指令是一条BGEZAL指令,其测试条件是_____,目标地址为0x_____。

⑪ 单步执行1条指令。

⑫ 查看PC的值,[PC]=0x_____,表明分支_____(成功,失败);查看R31的值,[R31]=0x_____。

⑬ 单步执行1条指令。

⑭ 查看R1的值,[R1]=0x_____。

⑮ 下一条指令地址为0x_____,是一条JALR指令,保存目标地址的寄存器为R____,保存返回地址的目标寄存器为R____。

⑯ 单步执行1条指令。

⑰ 查看PC和R3的值,[PC]=0x_____,[R3]=0x_____。

1.4 MIPSsim使用手册

1.4.1 启动模拟器

双击MIPSsim.exe即可启动该模拟器。MIPSsim是在Windows操作系统上运行的程序,它需要用.NET运行环境。如果你的计算机没有该环境,请先下载和安装“Microsoft .NET Framework 2.0版可再发行组件包”(详见Microsoft网站或www.Arch365.net上的信息)。

模拟器启动后,自动将自己初始化为默认状态,即:

- 所有通用寄存器和浮点寄存器为全0;
- 内存清零;
- 流水寄存器为全0;
- 清空时钟图、断点、统计数据;
- 内存容量为4096B;
- 载入起始地址为0;
- 浮点加法、乘法、除法部件的个数均为1;
- 浮点加法、乘法、除法运算延迟分别为6、7、10个时钟周期;
- 采用流水方式;
- 不采用定向机制;

- 不采用延迟槽；
- 采用符号地址；
- 采用绝对周期计数。

当模拟器工作在非流水方式下(配置菜单中的“流水方式”前没有√号)时，下面叙述中有关流水段的内容都没有意义，应该将其忽略。

1.4.2 MIPSsim 的窗口

在流水方式下，模拟器主界面共有 7 个子窗口，它们分别是“代码”窗口、“寄存器”窗口、“流水线”窗口、“时钟周期图”窗口、“内存”窗口、“统计”窗口和“断点”窗口。每一个窗口都可以被收起(变成小图标)、展开、拖动位置和放大/缩小。当要看窗口的全部内容时，可以将其放大到最大。

在非流水方式下，只有“代码”窗口、“寄存器”窗口、“内存”窗口和“断点”窗口。

1. “代码”窗口

“代码”窗口给出内存中代码的列表，每条指令占一行，按地址顺序排列。每行有 5 列(当全部显示时)，即地址、断点标记、指令的机器码、流水段标记和符号指令，如图 1.1 所示。

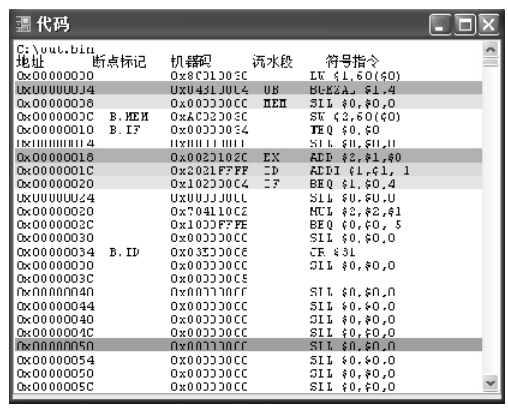

图 1.1 “代码”窗口

图中不同颜色的行代表相应的指令所处的执行段。黄色代表 IF 段，绿色代表 ID 段，红色代表 EX 段，青色代表 MEM 段，棕色代表 WB 段。

该窗口中各列的含义如下：

(1) 地址：以十六进制的形式给出。内存是按字节寻址的，每条指令占 4 个字节。当采用符号地址时，会在相应的位置给出汇编程序中出现的标号。

(2) 断点标记：如果在该指令处设有断点，则显示相应的标记。断点标记的形式为 B.X(X 为段名)，表示该断点是设置在该指令的 X 段。例如，若某行的断点标记为 B.EX，则表示在该指令的 EX 段设置了断点。

当模拟器工作在非流水方式下时，断点的标记为 B。

(3) 机器码：该行对应指令的十六进制机器码。若该行无指令，则仅仅显示 4 字节

数据。

(4) 流水段标记：当该指令正在执行时，它表示在当前周期该指令所处的流水段。当模拟器工作在非流水方式下时，它没有意义。

(5) 符号指令：机器代码所对应的符号指令。

在该窗口中选中某行(用鼠标左键单击)，然后再右击，在弹出的快捷菜单中选择“设置断点”和“清除断点”命令，它们分别用于在所选指令处设置断点和清除断点。

- 设置断点。选择要设断点的指令，右击，在弹出的快捷菜单中选择“设置断点”命令，弹出“设置断点”对话框，在“段”的下拉框中选择断点所在的流水段(在非流水方式下，不存在该下拉框)，单击“确定”按钮即可。
- 清除断点。选择指令，右击，在弹出的快捷菜单中选择“清除断点”命令，则设置在该指令处的断点被删除。

2. “寄存器”窗口

“寄存器”窗口显示 MIPSsim 模拟器中的寄存器的内容。共有 4 组寄存器，即通用寄存器、浮点寄存器、特殊寄存器和流水寄存器，分为 4 栏来显示。每一栏中分别有各自的数据格式选项，如图 1.2 所示。

图 1.2 “寄存器”窗口

1) 通用寄存器

MIPS64 有 32 个 64 位通用寄存器：R0，R1，…，R31。它们被简称为 GPRs(General-Purpose Registers)，有时也被称为整数寄存器。R0 的值永远是 0。

通过数据格式选项，可以选择显示的格式是十进制还是十六进制。

2) 浮点寄存器

共有 32 个 64 位浮点数寄存器：F0，F1，…，F31。它们被简称为 FPRs(Floating-Point Registers)。它们既可以用来存放 32 个单精度浮点数(32 位)，也可以用来存放 32 个双精度浮点数(64 位)。存储单精度浮点数(32 位)时，只用到 FPR 的一半，另一半没用。

3) 特殊寄存器

特殊寄存器有 4 个：

- PC(程序计数器,32 位);
- LO(乘法寄存器的低位);
- HI(乘法寄存器的高位);
- FCSR(浮点状态寄存器)。

4) 流水寄存器

- IF/ID. IR:流水段 IF 与 ID 之间的指令寄存器;
- IF/ID. NPC:流水段 IF 与 ID 之间的下一指令程序计数器;
- ID/EX. A:流水段 ID 与 EX 之间的第一操作数寄存器;
- ID/EX. B:流水段 ID 与 EX 之间的第二操作数寄存器;
- ID/EX. Imm:流水段 ID 与 EX 之间的立即数寄存器;
- ID/EX. IR:存放从 IF/ID. IR 传过来的指令;
- EX/MEM. ALUo:流水段 EX 与 MEM 之间的 ALU 计算结果寄存器;
- EX/MEM. IR:存放从 ID/EX. IR 传过来的指令;
- MEM/WB. LMD:流水段 MEM 与 WB 之间的数据寄存器,用于存放从存储器读出的数据;
- MEM/WB. ALUo:存放从 EX/MEM. ALUo 传过来的计算结果;
- MEM/WB. IR:存放从 EX/MEM. IR 传过来的指令。

除了流水寄存器外,其他寄存器都可以修改。只要双击某寄存器所在的行,系统就会弹出一个对话框。该对话框显示了该寄存器原来的值。在新值框中填入新的值,然后单击“保存”按钮,系统就会将新值写入该寄存器。

3. “流水线”窗口

“流水线”窗口显示流水线在当前配置下的组成,以及该流水线的各段在当前周期正在处理的指令,如图 1.3 所示。

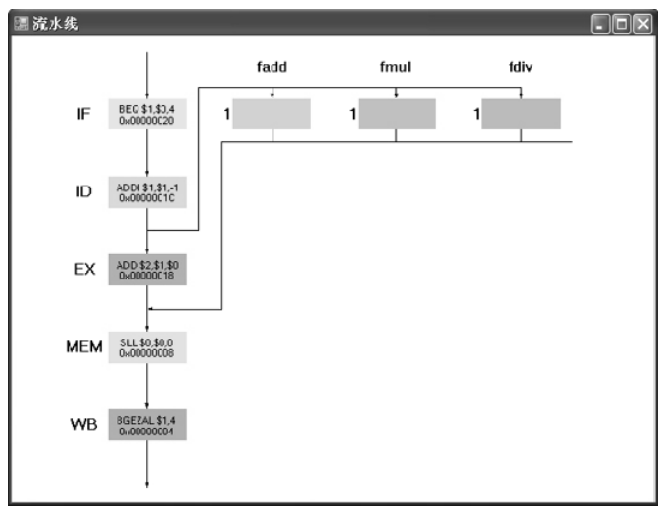

图 1.3 “流水线”窗口

非流水方式下，没有该窗口。

在该窗口中，每一个矩形方块代表一个流水段，它们用不同的颜色进行填充。在该窗口的左侧是IF段到WB段，其右边为浮点部件。浮点部件有浮点加法部件(fadd)、浮点乘法部件(fmul)和浮点除法部件(fdiv)3种。在菜单“配置”中的“常规配置”中修改浮点部件个数，可以看到该窗口中对应类型的浮点部件个数会发生相应的变化。

在运行过程中，各段的矩形方块中会显示该段正在处理的指令及其地址(十六进制)。当双击某矩形方块时，会弹出窗口显示该段出口处的流水寄存器的内容(十六进制)。

4.“时钟周期图”窗口

该窗口用于显示程序执行的时间关系，画出各条指令执行时所用的时钟周期。非流水方式下，没有该窗口。

以窗口左上为原点，横轴正方向指向右方，表示模拟器先后经过的各个周期(列)，纵轴正方向指向下方，表示模拟器中先后执行的各条指令(行)，如图1.4所示。

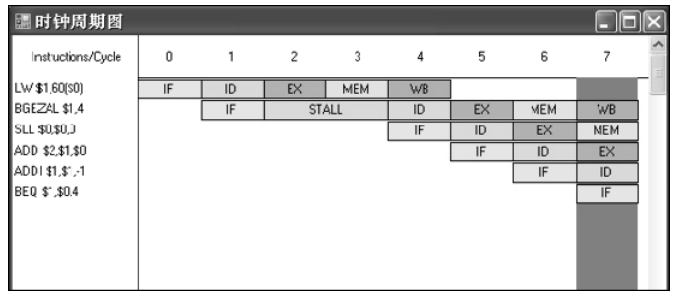

图1.4 “时钟周期图”窗口

横坐标有相对周期计数和绝对周期计数两种不同的表示形式。在默认的绝对周期计数下，按0,1,2,…依次递增的顺序计数。而在相对周期计数下，当前周期记为第0个周期，而其余周期(在左边)则按其相对于当前周期的位置，分别记为－1，－2，－3，…。

在由指令轴和周期轴组成的二维空间下，坐标(n，i)对应的矩形区域表示指令i在第n＋1周期时所经过的流水段(假设采用绝对周期计数)。

双击某行时，会弹出一个窗口，显示该指令在各流水段所进行的处理。

该窗口中还显示定向的情况。这是用箭头来表示的。若在第m周期和第m＋1周期间产生从指令i1到指令i2的定向，则在坐标(m，i1)和(m＋1，i2)表示的矩形区域之间会有一个箭头。

5.“内存”窗口

该窗口显示模拟器内存中的内容，左侧一栏为十六进制地址，右侧为数据，如图1.5所示。可以直接通过双击来修改其内容。这时会弹出“内存修改”对话框，如图1.6所示。对话框的上部区域为数据类型与格式选择区，通过选中其中的一项，就可以指定所采用的数据类型与格式。

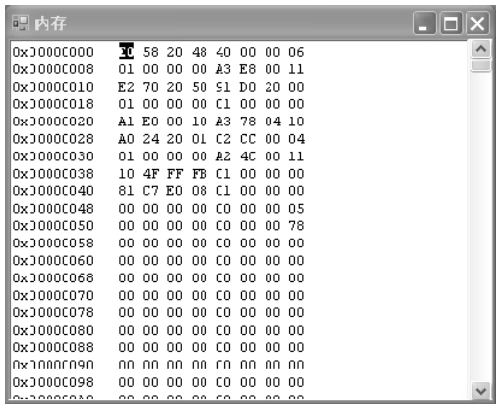

图 1.5 “内存”窗口

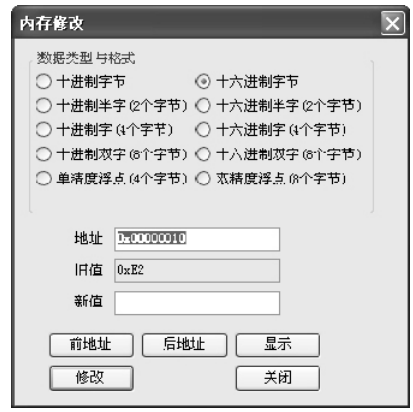

图 1.6 “内存修改”对话框

在“内存修改”对话框中,“地址”文本框最开始显示的是被双击的单元的地址,用户可以直接修改该地址。在“新值”文本框中输入新值,然后单击“修改”按钮,模拟器就会把新值写入内存中相应的单元。新值的格式必须与所选的数据类型和格式一致。

“前地址”与“后地址”按钮分别将当前地址减少和增加一个数据长度(字节数),并显示当前地址所指定单元的内容。“前地址”和“后地址”用于连续修改多个内存单元中的数据。“显示”按钮用于显示当前地址所指单元的内容。在修改地址后,单击该按钮就可以显示内存单元的内容。

6. “统计”窗口

该窗口显示模拟器统计的各项数据,在非流水方式下,没有该窗口。设计的数据如下。

1) 汇总

- 执行周期总数为 0。
- ID 段执行了 0 条指令。

2) 硬件配置

- 内存容量:4096 B。
- 加法器:1 个,执行时间(周期数)为 6。
- 乘法器:1 个,执行时间(周期数)为 7。
- 除法器:1 个,执行时间(周期数)为 10。
- 定向机制:不采用。

3) 停顿(周期数)

- RAW 停顿:0,占周期总数的百分比为 0。其中:
  - load 停顿:0,占所有 RAW 停顿的百分比为 0%;
  - 浮点停顿:0,占所有 RAW 停顿的百分比为 0%。
- WAW 停顿:0,占周期总数的百分比为 0%。
- 结构停顿:0,占周期总数的百分比为 0%。

◆ 控制停顿：0，占周期总数的百分比为 0%。

◆ 自陷停顿：0，占周期总数的百分比为 0%。

◆ 停顿周期总数：0，占周期总数的百分比为 0%。

4）分支指令

指令条数：0，占指令总数的百分比为 0%。其中：

• 分支成功：0，占分支指令数的百分比为 0%；

• 分支失败：0，占分支指令数的百分比为 0%。

5）load/store 指令

指令条数：0，占指令总数的百分比为 0%。其中：

• load：0，占 load/store 指令数的百分比为 0%；

• store：0，占 load/store 指令数的百分比为 0%。

6）浮点指令

指令条数：0，占指令总数的百分比为 0%。其中：

• 加法：0，占浮点指令数的百分比为 0%；

• 乘法：0，占浮点指令数的百分比为 0%；

• 除法：0，占浮点指令数的百分比为 0%。

7）自陷指令

指令条数：0，占指令总数的百分比为 0%。

7. “断点”窗口

断点一般是指指定的一条指令，当程序执行到该指令时，会中断执行，暂停在该指令上。在本模拟器中，断点可以设定在某条指令的某一个流水段上（如果是在流水方式下）。当该指令执行到相应的流水段时，会中断执行。

“断点”窗口列出当前已经设置的所有断点，每行一个。每行由 3 部分构成：地址（十六进制），流水段名称，符号指令，如图 1.7 所示。（在非流水方式下，“段”没有意义）

图 1.7 “断点”窗口

该窗口上方有 4 个按钮，即添加、删除、全部删除和修改。

（1）添加。单击“添加”按钮，会弹出“设置断点”对话框，在“地址”文本框中输入断点的十六进制地址，在“段”的下拉框中选择在哪个流水段中断（非流水方式下，不需要该操作），单击“确定”按钮即可。

（2）删除。选中某个断点（单击断点列表中相应的一项），单击“删除”按钮，则该断点被

清除。

(3) 全部删除。单击“全部删除”按钮,所有断点都将被清除。

(4) 修改。选中某个断点,单击“修改”按钮,会弹出“设置断点”对话框,在“地址”文本框中输入断点的地址,在“段”的下拉框中选择在哪个流水段中断,单击“确定”按钮即可将原断点修改为新设断点。

1.4.3 MIPSsim 的菜单

1. “文件”菜单

“文件”菜单如图 1.8 所示。

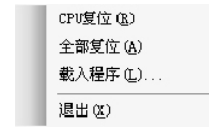

图 1.8 “文件”菜单

1) CPU 复位

将模拟器中 CPU 的状态复位为默认值。

2) 全部复位

将整个模拟器的状态复位为默认值。模拟器启动时,也是将状态设置为默认值。

3) 载入程序

将被模拟程序载入模拟器的内存。被模拟程序可以是汇编程序(.s 文件),也可以是汇编后的代码(.bin 文件)。单击该菜单后,系统将弹出“载入”对话框,选择要载入的文件,然后单击“打开”按钮。如果是.s 文件,系统会对该文件进行汇编。若汇编过程无错误,则将产生的二进制代码载入至模拟器的内存;若有错误,则报告错误信息。如果是.bin 文件,则直接将该文件的内容载入到模拟器内存。

被载入程序在内存中连续存放,其起始地址默认为 0。

4) 退出

单击“退出”菜单项退出模拟器。

2. “执行”菜单

该菜单提供了对模拟器执行程序进行控制的功能。在下面的执行方式中,除了单步执行,当遇到断点或者用户手动中止(用“中止”菜单项)时,模拟器将立即暂停执行。

在流水方式下,“执行”菜单如图 1.9 所示。

在非流水方式下,“执行”菜单如图 1.10 所示。

1) 单步执行一个周期

执行一个时钟周期,然后暂停,其快捷键为 F7。该菜单仅出现在流水方式下。

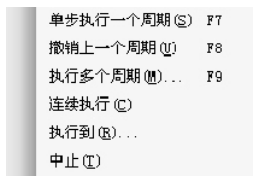

图 1.9 流水方式下的“执行”菜单

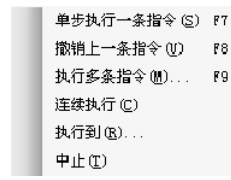

图 1.10 非流水方式下的“执行”菜单

2）撤销上一个周期

模拟器回退一个时钟周期，即恢复到执行该周期之前的状态，其快捷键为F8。该菜单仅出现在流水方式下。

3）执行多个周期

执行多个时钟周期，然后暂停。单击该菜单后，系统会弹出一个对话框，由用户指定要执行的周期的个数。该菜单仅出现在流水方式下。

4）连续执行

从当前状态开始连续执行程序，直到程序结束或者遇到断点或者用户手动中止。

5）执行到

单击“执行到”菜单项后，系统会弹出“设定终点”对话框，由用户指定此次执行的终点，即在哪条指令的哪个流水段暂停。单击“确定”按钮后，模拟器开始执行程序，直到到达终点位置或者遇到断点或者用户手动中止。所输入的终点地址将被归整为4的整数倍。

6）中止

单击该菜单项后，模拟器会立即暂停执行。当模拟器执行程序出现长期不结束的状况时，可以用该菜单强行使模拟器停止执行。

7）单步执行一条指令

执行一条指令，然后暂停。该指令的地址由当前的PC给出，其快捷键为F7。该菜单仅出现在非流水方式下。

8）撤销上一条指令

模拟器回退一条指令，即恢复到执行该指令之前的状态，其快捷键为F8。该菜单仅出现在非流水方式下。

9）执行多条指令

执行多条指令，然后暂停。单击该菜单项后，系统会弹出一个对话框，由用户指定要执行的指令的条数。

3. “内存”菜单

该菜单下有3项，即显示、修改和符号表。

1）显示

该菜单项用于设置显示内存的值的数据类型与格式。单击该菜单项后，系统会弹出“内存显示”对话框，如图1.11所示。在选定所要的数据类型与格式后，单击“确定”按钮，“内存”窗口中的数据就会按指定的数据类型与格式显示。

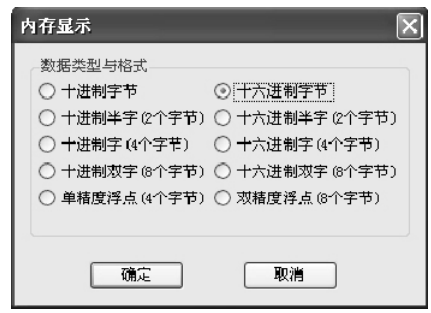

图1.11 “内存显示”窗口

2）修改

该菜单项用于对内存的值进行修改。单击该菜单项后，系统会弹出“内存修改”对话框，该对话框与图1.6所示的“内存修改”对话框相同。请参见相关的论述。

3）符号表

该菜单项用于显示符号表。符号表中列出了当前被执行的程序中各符号的地址（内存地址）。

4. “代码”菜单

1）设置断点

单击该菜单项，会弹出“设置断点”对话框，输入断点地址，并在“段”的下拉框中选择断点所在的流水段（在非流水方式下，不存在该下拉框），单击“确定”按钮即可。

2）取消断点

在“代码”窗口中选择某条指令，然后单击该菜单项，则设置在该指令处的断点被删除。

3）清除所有断点

选择此项，就会清除所有断点。

5. “配置”菜单

“配置”菜单用于修改模拟器的配置。需要注意的是，修改配置将使模拟器复位。

“配置”菜单如图 1.12 所示。

图 1.12 “配置”菜单

1）常规配置

该菜单项用于查看和设置以下参数：（其默认值见 1.4.1 节）

- 内存容量；
- 浮点加法运算部件个数；
- 浮点乘法运算部件个数；
- 浮点除法运算部件个数；
- 浮点加法运算延迟周期数；
- 浮点乘法运算延迟周期数；
- 浮点除法运算延迟周期数。

如果要修改这些参数，只要直接修改，然后单击“确定”按钮即可。

2）流水方式

“流水方式”开关。当该项被勾选（即其前面有个√号）时，模拟器按流水方式工作，能模拟流水线的工作过程。否则（即没有被勾选）就是按非流水方式执行程序。

该项的勾选和去选（即去掉其前面的√号）都是通过单击该项来实现的。

当从流水方式切换为非流水方式（或相反）时，系统将强行使模拟器的状态复位。

3）符号地址

“符号地址”开关。当该项被勾选时，“代码”窗口中的代码将显示原汇编程序中所采用的标号。否则（即没有被勾选）就不显示。

4）绝对周期计数

“绝对周期计数”开关。当该项被勾选时，“时钟周期图”窗口中的横坐标将显示为绝对周期，即时钟周期按 0，1，2，… 顺序递增的顺序计数。否则（即没有被勾选）就按相对周期计数，即把当前周期记为第 0 个周期，而其余周期（在左边）则按其相对于当前周期的位置，分别记为－1、－2、－3、…。

在非流水方式下，该菜单项不起作用（变灰）。

5）定向

“定向”开关，用于指定是否采用定向技术。当该项被勾选时，模拟器在执行程序时将采

用定向技术。否则就不采用。

在非流水方式下,该菜单项不起作用(变灰)。

6) 延迟槽

"延迟槽"开关,用于指定是否采用延迟分支技术。当该项被勾选时,模拟器在执行程序时将采用一个延迟槽。否则就不采用。

在非流水方式下,该菜单项不起作用(变灰)。

7) 载入配置

用于将一个配置文件(.cfg)载入模拟器,模拟器将按该文件进行配置并复位。单击该菜单项,系统将弹出"打开"对话框,让用户指定所要载入的配置文件。

8) 保存配置

用于将模拟器当前的配置保存到一个配置文件(.cfg)中,以便以后重用。单击该菜单项,系统将弹出"保存文件"对话框,让用户指定配置文件的名称和位置。

6. "窗口"菜单

1) 平铺

将当前已打开的子窗口平铺在主窗口中。

2) 层叠

将当前已打开的子窗口层叠在主窗口中。

3) 打开所有

将所有子窗口都打开。

4) 收起所有

将所有子窗口最小化。

7. "帮助"菜单

该菜单下有两项,即"帮助"和"关于 MIPSsim"。前者暂无信息,后者给出关于 MIPSsim 的版权信息和设计开发者信息。

1.5 相关知识:MIPS 指令系统

1981 年,Stanford 大学的 Hennessy 及其同事们发明了他们的 MIPS 计算机,后来,在此基础上形成了 MIPS 系列微处理器。到目前为止,已经出现了许多版本的 MIPS。下面介绍 MIPS64 的一个子集,并将它简称为 MIPS。

1.5.1 MIPS 的寄存器

MIPS64 有 32 个 64 位通用寄存器:R0,R1,…,R31。它们被简称为 GPRs(General-Purpose Registers),有时也被称为整数寄存器。R0 的值永远是 0。此外,还有 32 个 64 位浮点数寄存器:F0,F1,…,F31。它们被简称为 FPRs(Floating-Point Registers)。它们既可以用来存放 32 个单精度浮点数(32 位),也可以用来存放 32 个双精度浮点数(64 位)。存

储单精度浮点数(32 位)时,只用到 FPR 的一半,其另一半没用。MIPS 提供了单精度和双精度(32 位和 64 位)操作的指令,而且还提供了在 FPRs 和 GPRs 之间传送数据的指令。

另外,还有一些特殊寄存器,例如浮点状态寄存器。它们可以与通用寄存器交换数据。浮点状态寄存器用来保存有关浮点操作结果的信息。

1.5.2 MIPS 的数据表示

MIPS 的数据表示如下:

(1) 整数:字节(8 位)、半字(16 位)、字(32 位)和双字(64 位)。

(2) 浮点数:单精度浮点数(32 位)和双精度浮点数(64 位)。

之所以设置半字操作数类型,是因为在类似于 C 的高级语言中有这种数据类型,而且在操作系统等程序中也很常用,这些程序很重视数据所占的空间大小。设置单精度浮点操作数也是基于类似的原因。

MIPS64 的操作是针对 64 位整数以及 32 位或 64 位浮点数进行的。字节、半字或者字在装入 64 位寄存器时,用零扩展或者用符号位扩展来填充该寄存器的剩余部分。装入以后,对它们按照 64 位整数的方式进行运算。

1.5.3 MIPS 的数据寻址方式

MIPS 的数据寻址方式只有立即数寻址和偏移量寻址两种,立即数字段和偏移量字段都是 16 位的。寄存器间接寻址是通过把 0 作为偏移量来实现的,16 位绝对寻址是通过把 R0(其值永远为 0)作为基址寄存器来完成的。这样就有了 4 种寻址方式。

MIPS 的寻址方式是编码到操作码中的。

MIPS 的存储器是按字节寻址的,地址为 64 位。由于 MIPS 是 load-store 结构,GPRs 和 FPRs 与存储器之间的数据传送都是通过 load 指令和 store 指令来完成的。与 GPRs 有关的存储器访问可以是字节、半字、字或双字。与 FPRs 有关的存储器访问可以是单精度浮点数或双精度浮点数。所有存储器访问都必须边界对齐。

1.5.4 MIPS 的指令格式

为了使处理器更容易进行流水实现和译码,所有的指令都是 32 位的,其格式如图 1.13 所示。这些指令格式很简单,其中操作码占 6 位。MIPS 按不同类型的指令设置不同的格式,共有 3 种格式,它们分别对应于 I 类指令、R 类指令、J 类指令。在这 3 种格式中,同名字段的位置固定不变。

1. I 类指令

I 类指令包括所有的 load 和 store 指令、立即数指令、分支指令、寄存器跳转指令、寄存器链接跳转指令。其格式如图 1.13(a)所示。其中的立即数字段为 16 位,用于提供立即数或偏移量。

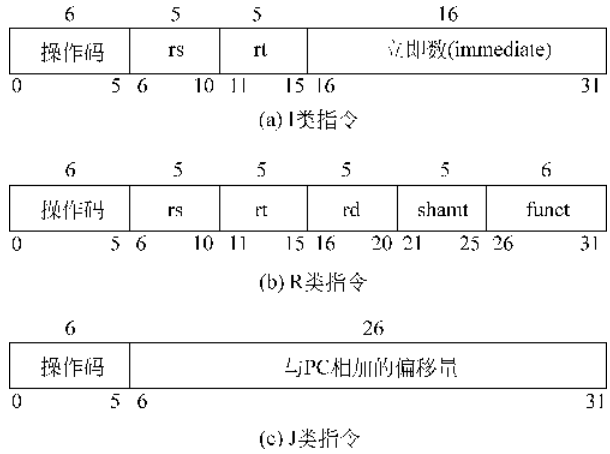

图 1.13　MIPS 的指令格式

1）load 指令

访存有效地址为 Regs[rs]+immediate，从存储器取来的数据放入寄存器 rt 中。

2）store 指令

访存有效地址为 Regs[rs]+immediate，需要存入存储器的数据放在寄存器 rt 中。

3）立即数指令

Regs[rt] ← Regs[rs] op immediate

4）分支指令

转移目标地址为 Regs[rs]+immediate，rt 无用。

5）寄存器跳转、寄存器跳转并链接

转移目标地址为 Regs[rs]。

2. R 类指令

R 类指令包括 ALU 指令、专用寄存器读/写指令、move 指令等。

ALU 指令：

Regs[rd] ← Regs[rs] funct Regs[rt]

funct 为具体的运算操作编码。

3. J 类指令

J 类指令包括跳转指令、跳转并链接指令、自陷指令、异常返回指令。在这类指令中，指令字的低 26 位是偏移量，它与 PC 值相加形成跳转的地址。

1.5.5　MIPS 的部分指令介绍

MIPS 指令可以分为 4 大类：load 和 store、ALU 操作、分支与跳转、浮点操作。

1. load 和 store 指令

除了 R0 外，所有通用寄存器与浮点寄存器都可以进行 load 或 store。表 1.1 给出了 load 和 store 指令的一些具体例子。单精度浮点数占用浮点寄存器的一半，单精度与双精度之间的转换必须显式地进行。浮点数的格式是 IEEE 754。

表 1.1　MIPS 的 load 和 store 指令的例子

指 令 举 例	指 令 名 称	含　　义
LD R2,20(R3)	装入双字	Regs[R2]←$_{64}$ Mem[20+Regs[R3]]
LW R2,40(R3)	装入字	Regs[R2]←$_{64}$ (Mem[40+Regs[R3]]$_0$)^{32} ## Mem[40+Regs[R3]]
LB R2,30(R3)	装入字节	Regs[R2]←$_{64}$ (Mem[30+Regs[R3]]$_0$)^{56} ## Mem[30+Regs[R3]]
LBU R2,40(R3)	装入无符号字节	Regs[R2]←$_{64}$ 0^{56} ## Mem[40+Regs[R3]]
LH R2,30(R3)	装入半字	Regs[R2]←$_{64}$ (Mem[30+Regs[R3]]$_0$)^{48} ## Mem[30+Regs[R3]] ## Mem[31+Regs[R3]]
L.S F2,60(R4)	装入单精度浮点数	Regs[F2]←$_{64}$ Mem[60+Regs[R4]] ## 0^{32}
L.D F2,40(R3)	装入双精度浮点数	Regs[F2]←$_{64}$ Mem[40+Regs[R3]]
SD R4,300(R5)	保存双字	Mem[300+Regs[R5]]←$_{64}$ Regs[R4]
SW R4,300(R5)	保存字	Mem[300+Regs[R5]]←$_{32}$ Regs[R4]
S.S F2,40(R2)	保存单精度浮点数	Mem[40+Regs[R2]]←$_{32}$ Regs[F2]$_{0\cdots31}$
SH R5,502(R4)	保存半字	Mem[502+Regs[R4]]←$_{16}$ Regs[R5]$_{48\cdots63}$

说明：要求内存的值必须边界对齐。

在下面解释指令的操作时，采用了类似于 C 语言的描述语言。符号的意义如下：

- Regs 表示寄存器组；
- Mem 表示主存，按字节寻址；
- 方括号表示内容，Mem[ ]表示存储器的内容，Regs[ ]表示寄存器的内容；
- x←$_n$ y 表示从 y 传送 n 位到 x。x,y←z 表示把 z 传送到 x 和 y；
- 用下标表示字段中具体的位。对于指令和数据，按从最高位到最低位(即从左到右)的顺序依次进行编号，最高位为第 0 位，次高位为第 1 位，依次类推。下标可以是一个数字，也可以是一个范围。例如，Regs[R4]$_0$ 表示寄存器 R4 的符号位，Regs[R4]$_{56\cdots63}$ 表示 R4 的最低字节；
- 上标用于表示对字段进行复制的次数。例如，0^{32} 表示一个 32 位长的全 0 字段；
- 符号 ## 用于两个字段的拼接，并且可以出现在数据传送的任何一边。

下面举个例子。假设 R8 和 R6 是 64 位的寄存器，则

Regs[R8]$_{32\cdots63}$←$_{32}$ (Mem [Regs[R6]]$_0$)^{24} ## Mem [Regs[R6]]

表示的意义是，以 R6 的内容作为地址访问主存，得到的字节按符号位扩展为 32 位后存入 R8 的低 32 位，R8 的高 32 位(即 Regs[R8]$_{0\cdots31}$)不变。

2. ALU 指令

MIPS 中所有的 ALU 指令都是寄存器-寄存器型(RR 型)或立即数型的。运算操作包

括算术和逻辑操作：加、减、乘、除、与、或、异或和移位等。表 1.2 中给出了一些例子。所有这些指令都支持立即数寻址模式，参与运算的立即数是由指令中的 immediate 字段（低 16 位）经符号位扩展后生成的。

表 1.2　MIPS 中 ALU 指令的例子

指 令 举 例	指 令 名 称	含　　义
DADDU R1,R2,R3	无符号加	Regs[R1] ← Regs[R2]+ Regs[R3]
DADDIU R4,R5,#6	无符号立即数加	Regs[R4] ← Regs[R5]+6
LUI　R1,#4	把立即数装入一个字的高 16 位	Regs[R1] ← 0^{32} ## 4 ## 0^{16}
DSLL　R1,R2,#5	逻辑左移	Regs[R1] ← Regs[R2]<<5
DSLT　R1,R2,R3	置小于	If(Regs[R2]< Regs[R3]) Regs[R1] ← 1 else Regs[R1] ← 0

R0 的值永远是 0，它可以用来合成一些常用的操作。例如

```
DADDIU R1,R0,#100                    // 给寄存器 R1 装入常数 100
```

又如

```
DADD R1,R0,R2                        // 把寄存器 R2 中的数据传送到寄存器 R1 中
```

3. 控制指令

表 1.3 给出了 MIPS 的几种典型的跳转和分支指令。跳转是无条件转移，而分支都是有条件转移。根据跳转指令确定目标地址的方式不同以及跳转时是否链接，可以把跳转指令分成 4 种。在 MIPS 中，确定转移目标地址的一种方法是把指令中的 26 位偏移量左移 2 位（因为指令字长都是 4 个字节）后，替换程序计数器的低 28 位；另外一种方法是由指令中指定的一个寄存器来给出转移目标地址，即间接跳转。简单跳转很简单，就是把目标地址送入程序计数器。而跳转并链接则要比简单跳转多一个操作：把返回地址（即顺序下一条指令的地址）放入寄存器 R31。跳转并链接适用于实现过程调用。

表 1.3　典型的 MIPS 控制指令

指 令 举 例	指 令 名 称	含　　义
J　name	跳转	$PC_{36\cdots63}$ ← name
JAL　name	跳转并链接	Regs[R31] ← PC+4; $PC_{36\cdots63}$ ← name; ((PC+4)−2^{27})≤name<((PC+4)+2^{27})
JALR　R3	寄存器跳转并链接	Regs[R31] ← PC+4; PC ← Regs[R3]
JR　R5	寄存器跳转	PC ← Regs[R5]
BEQZ　R4, name	等于零时分支	if(Regs[R4]== 0) PC ← name; ((PC+4)−2^{17})≤name<((PC+4)+2^{17})
BNE　R3, R4,name	不相等时分支	if(Regs[R3]!= Regs[R4]) PC ← name; ((PC+4)−2^{17})≤name<((PC+4)+2^{17})
MOVZ　R1,R2,R3	等于零时移动	if(Regs[R3]==0) Regs[R1] ← Regs[R2]

说明：除了以寄存器中的内容作为目标地址进行跳转以外，所有其他的控制指令的跳转地址都是相对于 PC 的。

所有的分支指令都是条件转移。分支条件由指令确定，例如，可能是测试某个寄存器的值是否为零。该寄存器可以是一个数据，也可以是前面一条比较指令的结果。MIPS 提供了一组比较指令，用于比较两个寄存器的值。例如，“置小于”指令，如果第一个寄存器中的值小于第二个寄存器的，则该比较指令在目的寄存器中放置一个 1(代表真)，否则将放置一个 0(代表假)。类似的指令还有“置等于”、“置不等于”等。这些比较指令还有一套与立即数进行比较的形式。

有的分支指令可以直接判断寄存器内容是否为负，或者比较两个寄存器是否相等。

分支的目标地址由 16 位带符号偏移量左移两位后和 PC 相加的结果来决定。另外，还有一条浮点条件分支指令，该指令通过测试浮点状态寄存器来决定是否进行分支。

4. 浮点指令

浮点指令对浮点寄存器中的数据进行操作，并由操作码指出操作数是单精度(SP)还是双精度(DP)的。在指令助记符中，用后缀 S 和 D 分别表示操作数是单精度还是双精度浮点数。例如 MOV. S 和 MOV. D 分别是把一个单精度浮点寄存器(MOV. S)或一个双精度浮点寄存器(MOV. D)中的值复制到另一个同类型的寄存器中。MFC1 和 MTC1 是在一个单精度浮点寄存器和一个整数寄存器之间传送数据的。另外，MIPS 还设置了在整数与浮点之间进行相互转换的指令。

浮点操作包括加、减、乘、除，分别有单精度和双精度指令。例如，加法指令 ADD. D(双精度)和 ADD. S(单精度)，减法指令 SUB. D 和 SUB. S 等。浮点数比较指令会根据比较结果设置浮点状态寄存器中的某一位，以便于后面的分支指令 BC1T(若真则分支)或 BC1F(若假则分支)测试该位，以决定是否进行分支。