计算机系统结构
实验指导书（试用版）

计算机学院 软件学院

2016 年 5 月

目录

说明：本实验指导书内容基于《计算机原理与系统结构模拟实验》（2015 年 6 月第一版，高等教育出版社）及其作者免费提供的配套模拟器软件改编。

第一部分 MIPSsim 使用手册

注意!! 本模拟器必须在 32 位系统下运行!!

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。如果你的机器没有该环境，请先下载和安装“Microsoft .NET Framework 2.0 版可再发行组件包”。

模拟器启动时，自动将自己初始化为默认状态。所设置的默认值为：

- 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；
- 内存大小为 4096 字节；
- 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，应该忽略之。

1.1MIPSsim 的窗口

在流水方式下，模拟器主界面中共有 7 个子窗口，它们是：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）：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。

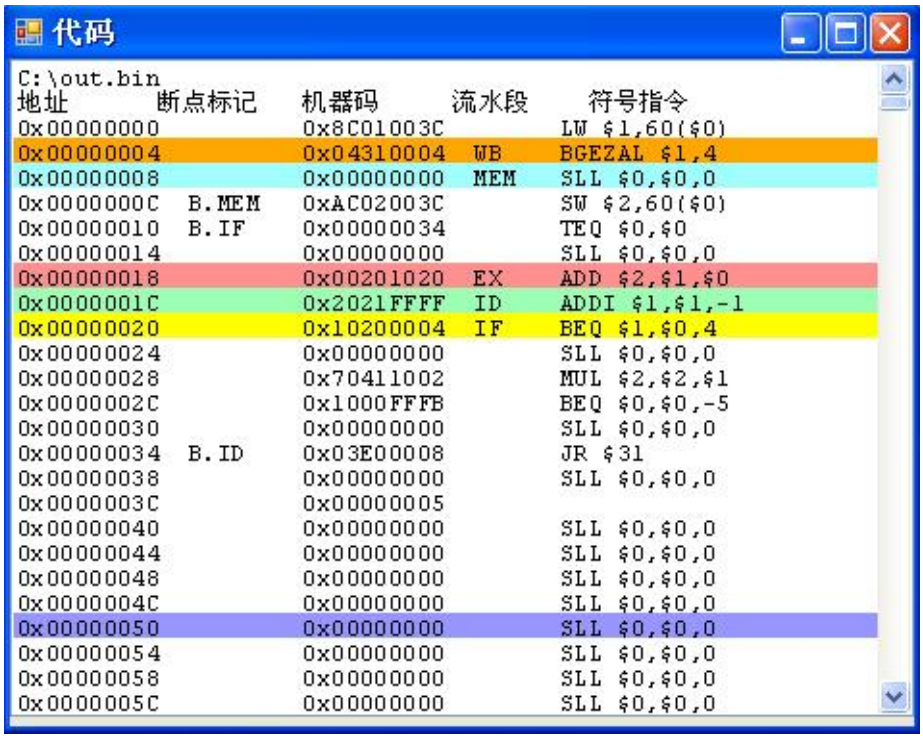


图 1.1 代码窗口

图中不同抹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：

- 地址：以 16 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。
- 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。
- 机器码：该行所对应的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；
- 流水段标记：表示当该指令正在执行时，它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。
- 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），然后再点击鼠标右键，就会弹出菜单：设置断点，清除断点，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（点击）要设断点的指令→点击右键→“设置断点”，弹出“设置断点”小对话框，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”即可。
- 清除断点

  选择（点击）指令→点击右键→“清除断点”，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：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。每一栏下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，其另一半没用。

(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，系统就会弹出一个小对话框。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，然后点击“保存”，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。

非流水方式下，没有该窗口。

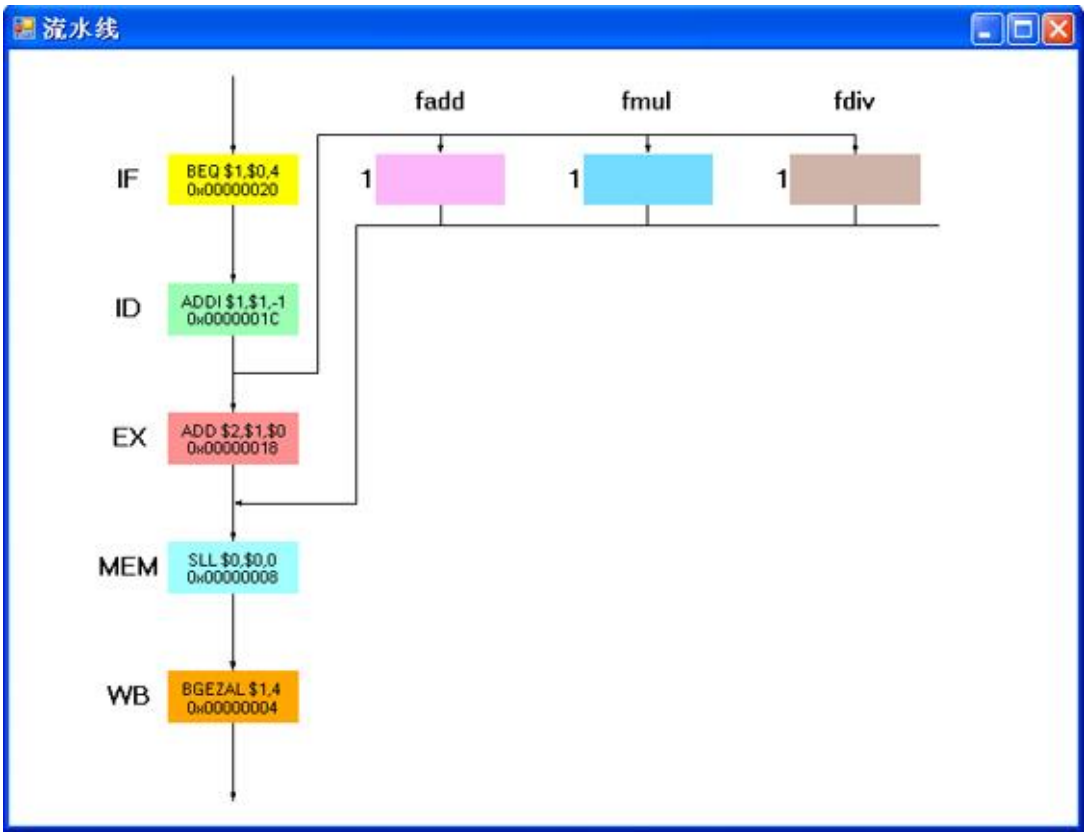


图 1.3 流水线窗口

在该窗口中，每一个矩形方块代表一个流水段，它们用不同的颜色填充。在该窗口的左侧是 IF 到 WB 段，其右边为浮点部件。浮点部件分有浮点加法部件（fadd）、浮点乘法部件（fmul）和浮点除法部件（fdiv）三种。在菜单“配置”→“常规配置”中修改浮点部件个数，可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（16 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（16 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。

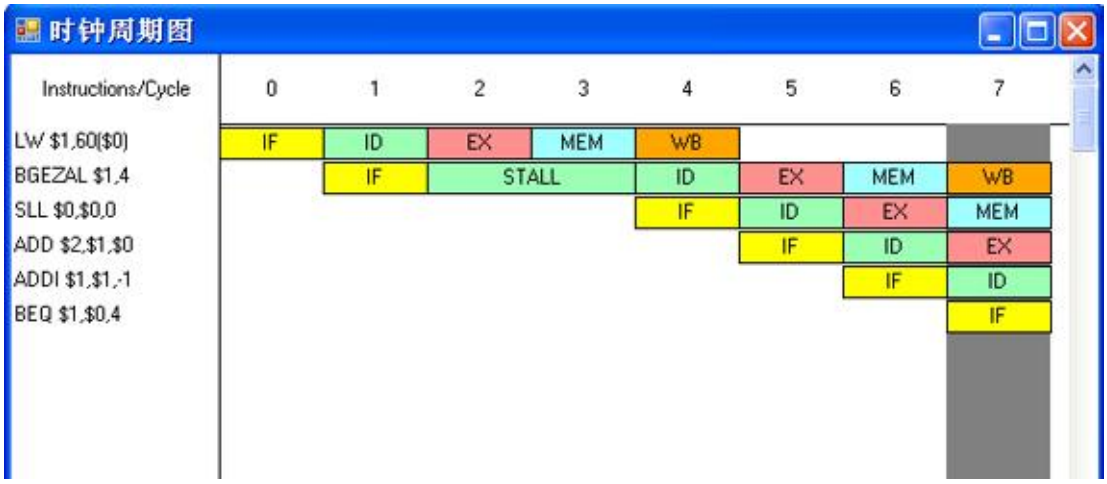


图 1.4 时钟周期图窗口

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，…等。

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，会弹出一个小窗口，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。这时会弹出一个“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，通过勾选其中的一项，就可以指定所采用的数据类型与格式。

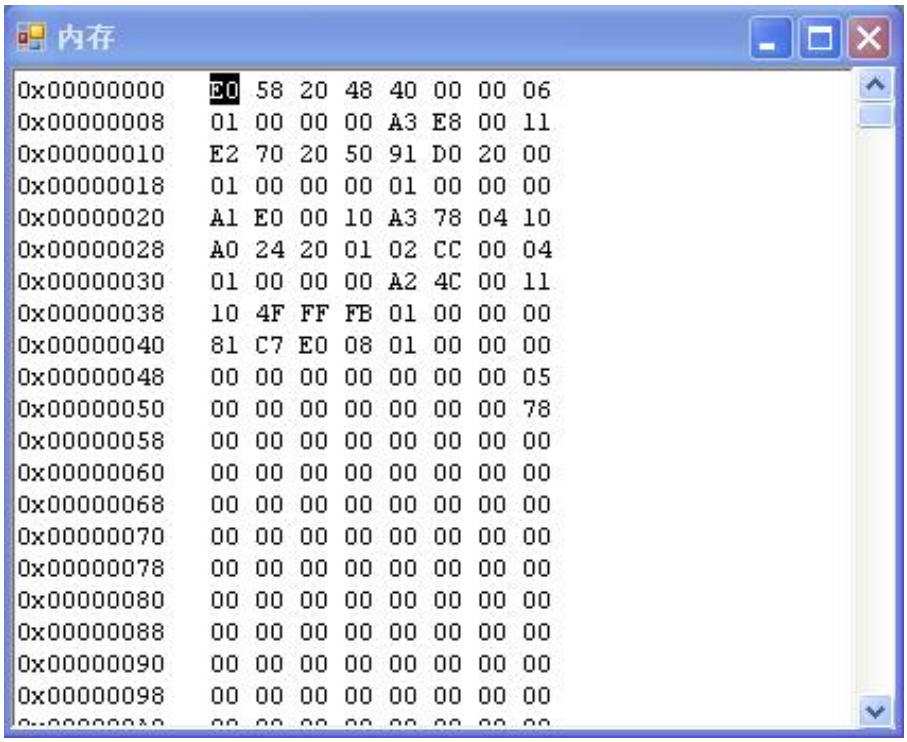


图 1.5 内存窗口

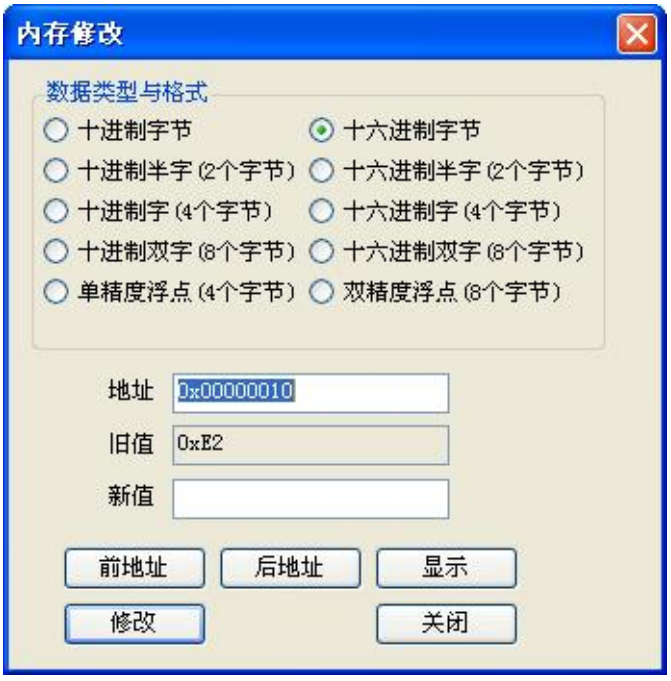


图 1.6 “内存修改”对话框

在该“内存修改”对话框中，地址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。在新值框中输入新值，然后点击按钮“修改”，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。

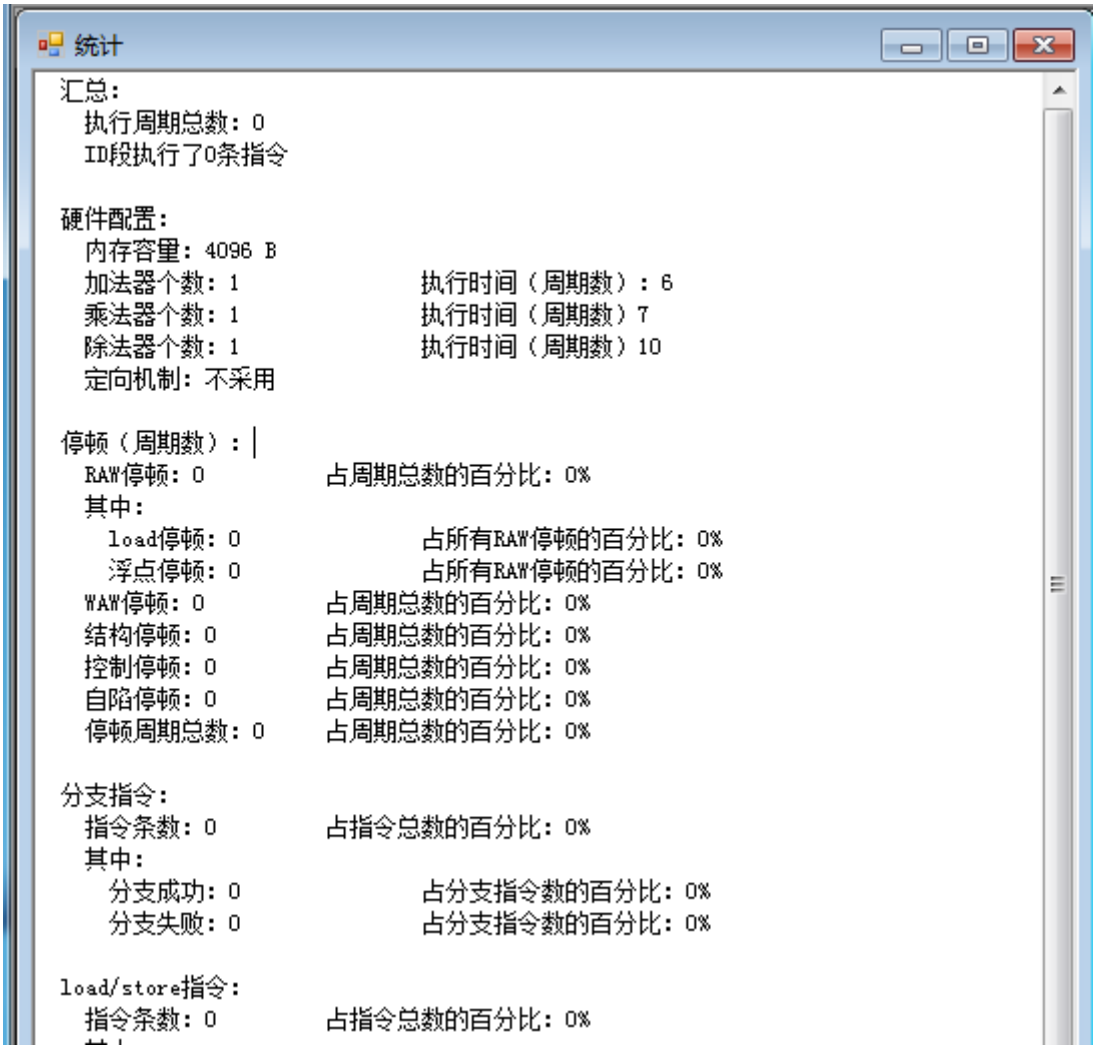
“前地址”与“后地址”按钮分别将当前地址减少和增加一个数据长度（字节数），并

显示当前地址所指定单元的内容。“前地址”和“后地址”用于连续修改一片的内存数据。“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，点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。如下所示。

（非流水方式下，没有该窗口）



汇总：

执行周期总数：0

ID 段执行了 0 条指令

硬件配置：

内存容量：4096 B

加法器个数：1　　执行时间（周期数）：6

乘法器个数：1　　执行时间（周期数）7

除法器个数：1　　执行时间（周期数）10

定向机制：不采用

停顿（周期数）：

RAW 停顿：0　　占周期总数的百分比：0%
其中：
load 停顿：0　　占所有 RAW 停顿的百分比：0%
分支/跳转停顿：0　　占所有 RAW 停顿的百分比：0%
浮点停顿：0　　占所有 RAW 停顿的百分比：0%
WAW 停顿：0　　占周期总数的百分比：0%
结构停顿：0　　占周期总数的百分比：0%
控制停顿：0　　占周期总数的百分比：0%
自陷停顿：0　　占周期总数的百分比：0%
停顿周期总数：0　占周期总数的百分比：0%

分支指令：
指令条数：0　　占指令总数的百分比：0%
其中：
分支成功：0　　占分支指令数的百分比：0%
分支失败：0　　占分支指令数的百分比：0%

load/store 指令：
指令条数：0　　占指令总数的百分比：0%
其中：
load：0　　占 load/store 指令数的百分比：0%
store：0　　占 load/store 指令数的百分比：0%

浮点指令：
指令条数：0　　占指令总数的百分比：0%
其中：
加法：0　　占浮点指令数的百分比：0%
乘法：0　　占浮点指令数的百分比：0%
除法：0　　占浮点指令数的百分比：0%

自陷指令：
指令条数：0　　占指令总数的百分比：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（16 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

该窗口上方有四个按钮：添加、删除、全部删除、修改。

◆ 添加

单击“添加”，会弹出小对话框“设置断点”，在“地址”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，下同），单击“确定”即可。

◆ 删除

选中某个断点（单击断点列表中相应的一项），单击“删除”，则该断点被清除。

◆ 全部删除

单击“全部删除”，所有断点都将被清除。

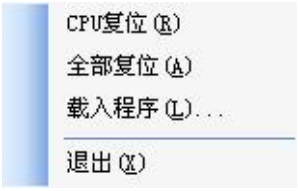
◆ 修改

选中某个断点，单击“修改”，会弹出小对话框“设置断点”，在“地址”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”即可将原断点修改为新设断点。

1.1 MIPSsim 的菜单

1. 文件菜单

文件菜单如下所示：



(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）。点击该菜单后，系统将弹出“载入”对话框，选择要载入的文件，然后点击“打开”。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。该起始地址可以从“代码”菜单中的“载入起始地址”来查看和修改。修改时，如果输入的地址不是 4 的整数倍，模拟器会自动将其归整为 4 的整数倍。

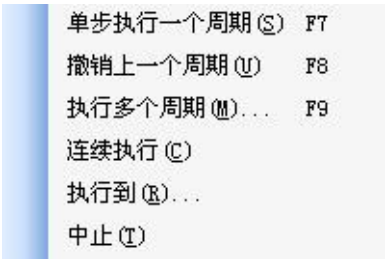
（4）退出

退出模拟器。

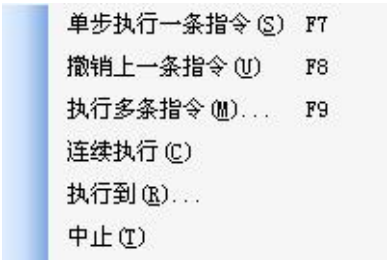
2. 执行菜单

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，执行菜单如下所示：



在非流水方式下，执行菜单如下所示：



（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。

（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。点击该菜单后，系统会弹出一个小对话框，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，直到程序结束或遇到断点或用户手动中止。

（5）执行到…

点击该菜单项后，系统会弹出一个“设定终点”小对话框，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。点击“确定”后，模拟器即开始执行程序，直到到达终点位置或遇到断点或用户手动中止。所输入的终点地址将被归整为 4 的整数倍。

（6） 中止

点击该菜单项后，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。点击该菜单项后，系统会弹出一个小对话框，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：显示，修改，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。点击该菜单项后，系统会弹出“内存显示”对话框，如图 1.8 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。

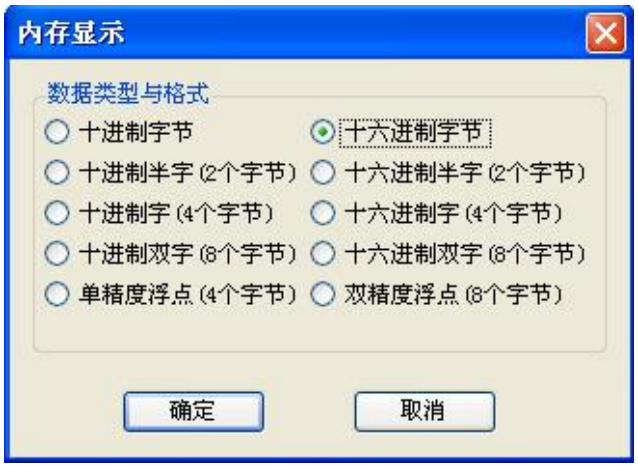


图 1.8 内存显示窗口

(2) 修改

该菜单项用于对内存的值进行修改。点击该菜单项后，系统会弹出“内存修改”对话框，该对话框与图 1.6 的“内存修改”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) 载入起始地址

该菜单项用于显示和修改代码载入内存的起始地址。如果要修改，只要在弹出的“载入起始地址”对话框中输入新的地址，然后点击“确认”即可。

(2) 设置断点

点击该菜单项，会弹出“设置断点”小对话框，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”即可。

(3) 取消断点

在代码窗口中选择某条指令，然后点击该菜单项，则设置在该指令处的断点被删除。

(4) 清除所有断点

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：修改配置将会使模拟器复位。

配置菜单如下所示：

(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 启动程序”）

- 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，然后点击“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）就是按串行方式执行程序。

该项的勾选和去选（即去掉其前面的“√”号）都是通过点击该项来实现的。下同。

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，…等。

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。点击该菜单项，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。点击该菜单项，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。

(5) 选择子窗口

在该菜单中还列出了所有的子窗口的名称，选择其中的任一个，将使该窗口被选中，并被提到最上层。

7. 帮助菜单

该菜单下有两项:“帮助”和“关于 MIPSsim”。前者提供用户手册,后者给出关于 MIPSsim 的版权信息和设计开发者信息。

第二部分 实验

实验 1 MIPS 指令系统和 MIPS 体系结构

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。

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。

3. 实验内容和步骤

首先要阅读 MIPSsim 模拟器的使用方法（见附录），然后了解 MIPSsim 的指令系统和汇编语言。

(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 MIPSsim 使用说明，熟悉 MIPSsim 模拟器的操作和使用方法。
可以先载人一个样例程序(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，
至看程序所在的位置(起始地址为 Ox00000100)。
(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(F7)。
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_____(字节．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_____(字节．半字，字)指令。
⑨单步执行 1 条指令。
⑩查看 R1 的值，[R1]=0x_____。
⑾单步执行 1 条指令。

⑿下一条指令地址为 0x_____，是一条保存_____(字节．半字，字)指令。
⒀单步执行 1 条指令(F7)。
⒁查看内存 BUFFER 处字的值，值为 0x_____。（内存—>符号表）
(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 L0、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）。
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_____
⑾单步执行 1 条指令。
⑿查看 PC 的值[PC]=0x_____，表明分支_____（成功，失败）；查看 R31 的值，[R31]=0x_____
⒀单步执行 1 条指令。
⒁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_____

实验 2 流水线及流水线中的冲突

1. 实验目的

（1）加深对计算机流水线基本概念的理解。
（2）理解 MIPS 结构如何用 5 段流水线来实现，理解各段的功能和基本操作。
（3）加深对数据冲突和资源冲突的理解，理解这两类冲突对 CPU 性能的影响。
（4）进一步理解解决数据冲突的方法，掌握如何应用定向技术来减少数据冲突引起的停顿。

2. 实验平台

指令级和流水线操作级模拟器 MIPSsim。

3. 实验内容和步骤

（1）启动 MIPSsim。
（2）进一步理解流水线窗口中各段的功能，掌握各流水寄存器的含义。（鼠标双击各段，即可看到各流水寄存器的内容）
（3）载入一个样例程序（在本模拟器所在文件夹下的“样例程序”文件夹中），然后分别以单步执行一个周期、执行多个周期、连续执行、设置断点等方式运行程序，观察程序的执行情况，观察 CPU 中寄存器和存储器内容的变化，特别是流水寄存器内容的变化。
（4）选择配置菜单中的“流水方式”选项，使模拟器工作于流水方式下。
（5）观察程序在流水方式下的执行情况。
（6）观察和分析结构冲突对 CPU 性能的影响，步骤如下：

1) 加载 structure_hz.s（在模拟器所在文件夹下的“样例程序”文件夹中）。
2) 执行该程序，找出存在结构冲突的指令对以及导致结构冲突的部件。
3) 记录由结构冲突引起的停顿周期数，计算停顿周期数占总执行周期数的百分比。
4) 把浮点加法器的个数改为 4 个。
5) 再重复 1-3 的步骤。
6) 分析结构冲突对 CPU 性能的影响，讨论解决结构冲突的方法。

（7）观察数据冲突并用定向技术来减少停顿，步骤如下：

1) 全部复位。
2) 加载 data_hz.s（在模拟器所在文件夹下的“样例程序”文件夹中）。
3) 关闭定向功能（在“配置”菜单下选择取消“定向”）。
4) 用单步执行一个周期的方式执行该程序，观察时钟周期图，列出什么时刻发生了 RAW 冲突。
5) 记录数据冲突引起的停顿周期数以及程序执行的总时钟周期数，计算停顿时钟周期数占总执行周期数的百分比。
6) 复位 CPU。
7) 打开定向功能。
8) 用单步执行一个周期的方式执行该程序，查看时钟周期图，列出什么时刻发生了 RAW 冲突，并与步骤 3）的结果比较。

时钟周期图为：

9) 记录数据冲突引起的停顿周期数以及程序执行的总周期数。计算采用定向以后性能比原来提高多少。

实验 3 使用 MIPS 指令实现求两个数组的点积

1. 实验目的

（1）通过实验熟悉实验 1 和实验 2 的内容

（2）增强汇编语言编程能力

（3）学会使用模拟器中的定向功能进行优化

（4）了解对代码进行优化的方法

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。

3. 实验内容和步骤：

（1）自行编写一个计算两个向量点积的汇编程序，该程序要求可以实现求两个向量点积计算后的结果。

向量的点积：假设有两个 n 维向量 a、b，则 a 与 b 的点积为：

$$\mathbf{a}\cdot\mathbf{b}=\sum_{i=1}^{n}a_ib_i=a_1b_1+a_2b_2+\cdots+a_nb_n$$

两个向量元素使用数组进行数据存储，**要求向量的维度不得小于 10**

（2）启动 MIPSsim。

（3）载入自己编写的程序，观察流水线输出结果。

（4）使用定向功能再次执行代码，与刚才执行结果进行比较，观察执行效率的不同。

（5）采用静态调度方法重排指令序列，减少相关，优化程序

（6）对优化后的程序使用定向功能执行，与刚才执行结果进行比较，观察执行效率的不同。

注意：不要使用浮点指令及浮点寄存器!!

使用 TEQ $r0 $r0 结束程序!!

4. 实验报告要求：

（1）实验目的。

（2）实验原理

（3）向量点积程序代码清单及注释说明

（4）优化后的程序代码清单

（5）未优化代码和优化代码性能分析比较结果（文字和截图说明），并给出优化的原因。

实验 4 使用 MIPS 指令实现冒泡排序法

1. 实验目的

（1）掌握静态调度方法

（2）增强汇编语言编程能力

（3）学会使用模拟器中的定向功能进行优化

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。

3. 实验内容和步骤：

（1） 自行编写一个实现冒泡排序的汇编程序，该程序要求可以实现对一维整数数组进行冒泡排序。

冒泡排序算法的运作如下：

①比较相邻的元素。如果第一个比第二个大，就交换他们两个。

②对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。

③针对所有的元素重复以上的步骤，除了最后一个。

④持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

要求数组长度不得小于 10

（2）启动 MIPSsim。

（3）载入自己编写的程序，观察流水线输出结果。

（4）使用定向功能再次执行代码，与刚才执行结果进行比较，观察执行效率的不同。

（5）采用静态调度方法重排指令序列，减少相关，优化程序

（6）对优化后的程序使用定向功能执行，与刚才执行结果进行比较，观察执行效率的不同。

注意：1 不要使用浮点指令及浮点寄存器!!

2 整数减勿使用 SUB 指令，请使用 DSUB 指令代替!!

4. 实验报告要求：

（1）实验目的。

（2）实验原理

（3）冒泡排序代码清单及注释说明

（4）优化后的程序代码清单

（5）未优化代码和优化代码性能分析比较结果（文字+截图说明），并给出优化的原因。

实验 5 指令调度与延迟分支

1. 实验目的

（1）加深对指令调度技术的理解。

（2）加深对延迟分支技术的理解。

（3）熟练账务用指令调度技术解决流水线中的数据冲突的方法。

（4）进一步理解指令调度技术对 CPU 性能的改进。

（5） 进一步理解延迟分支技术对 CPU 性能的改进。

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。

3. 实验内容和步骤

（1） 启动 MIPSsim。

（2） 根据实验 2 的相关知识中关于流水线各段操作的描述，进一步理解流水线窗口中各段的功能，掌握各流水线寄存器的含义（双击各段，就可以看到各流水线寄存器中的内容）。

（3） 选择“配置”→“流水方式”选项，使模拟器工作在流水方式下。

（4） 用指令调度技术解决流水线中的结构冲突与数据冲突：

1） 启动 MIPSsim。

2） 用 MIPSsim 的“文件”->“载入程序”选项来加载 schedule.s（在模拟器所在文件夹下的“样例程序”文件夹中）。

3） 关闭定向功能，这是通过“配置“->”定向“选项来实现的。

4） 执行所载入的程序，通过查看统计数据和时钟周期图，找出并记录程序执行过程中各种冲突发生的次数，发生冲突的指令组合以及程序执行的总时钟周期数。

5） 自己采用调度技术对程序进行指令调度，消除冲突（自己修改源程序）。将调度（修改）后的程序重新命名为 afer-schedule.s。（注意：调度方法灵活多样，在保证程序正确性的前提下自己随意调度，尽量减少冲突即可，不要求要达到最优。）

6） 载入 afer-schedule.s，执行该程序，观察程序在流水线中的执行情况，记录程序执行的总时钟周期数。

7） 比较调度前和调度后的性能，论述指令调度对提高 CPU 性能的作用。

（5） 用延迟分支技术减少分支指令对性能的影响：

1） 在 MIPSsim 中载入 branch.s 样例程序（在本模拟器目录的“样例程序”文件夹中。

2） 关闭延迟分支功能。这是通过在“配置”->“延迟槽”选项来实现的。

3） 执行该程序，观察并记录发生分支延迟的时刻，记录该程序执行的总时钟周期数。

4） 假设延迟槽为一个，自己对 branch.s 程序进行指令调度（自己修改源程序），将调度后的程序重新命名为 delayed-branch.s。

5） 载入 delayed-branch.s，打开延迟分支功能，执行该程序，观察其时钟周期图，记录程序执行的总时钟周期数。

4. 实验结论

对比不采用延迟分支和采用延迟分支两种情况下的时钟周期图，比较两种情况下的性能之间的不同，论述延迟分支对提高 CPU 性能的作用。

第三部分 附录