北 京 邮 电 大 学
计 算 机 学 院

计算机系统结构实验指导书

王春露 邝坚 编著

2007.3 – 2013.4

目　录

计算机系统结构实验简介

*DLX*是一个虚拟处理器。该处理器是加州大学伯克利分校计算机系*JohnL .H ennessy*教授和斯坦福大学计算机系*David A. Patterson*教授在其《计算机体系结构:一种定量的方法》一书中提出的。该处理器反映了新一代处理器的特点。通过了解DLX处理器的结构和工作原理，并利用DLX模拟器进行实验，可以帮助学生综合地了解和运用有关处理器指令系统的设计、流水线的设计与实现等方面的知识，有助于计算机系统结构课程内容的理解。

DLX处理器简介

第一节 *DLX*基本结构

DLX是一种典型的*Load/Store*型指令集结构。它不仅体现了当今多种机器的指令集结构的共同特点，而且它还体现出未来一些机器的指令集结构的特点。这些机器的指令集结构设计思想都和DLX指令集结构的设计思想十分相似，它们都强调：

（1） 具有一套简单的*Load/Store*指令集；

（2） 注重指令流水效率；

（3） 简化指令的译码；

（4） 高效支持编译器。

DLX是一种易于学习和研究的处理器结构模型。这种类型的机器正在日趋流行，而且其结构非常易于理解。

1．DLX 中的寄存器

DLX 中有 32 个通用寄存器（GPRs），分别将其命名为 R0,R1…R31。每个通用寄存器长度为 32 位。

另外，DLX 中有 32 个浮点寄存器（FPRs），分别将其命名为 F0,F1…F31。每个浮点寄存器长度为 32 位。这些浮点寄存器可以用来保存 32 位的单精度浮点数，或者通过相邻两个浮点寄存器奇偶对 FiFi+1（i=0,2,4…,30）来保存双精度浮点数，这种组合而成的 64 位双精度浮点寄存器在 DLX 中分别被命名为 F0,F2…F28,F30.

2. DLX 数据类型

DLX 提供了多种长度的整型数据和浮点数据。对整型数据而言，有 8 位，16 位，32 位多种长度;对浮点而言，有 32 位单精度浮点数和 64 位双精度浮点数。浮点数据表示采用的是 IEEE754 标准。DLX 操作都是对 32 位整型数据及 32 或 64 位浮点数据进行的。

3．DLX 的寻址方式和数据传送

DLX 提供了寄存器寻址，立即寻址，偏移寻址和寄存器间接寻址四种寻址方式。寄存器寻址字段的大小为 5 位，用来标识 32 个通用寄存器或浮点寄存器。

4. DLX 的指令格式

因为 DLX 只有四种寻址方式，所以将其寻址方式编码在操作码中。为了简化指令译码，并充分发挥流水线的效率，所有 DLX 指令的字长均是 32 位，其中用 6 位表示操作码。DLX 中各种类型指令的格式如图 1 所示：

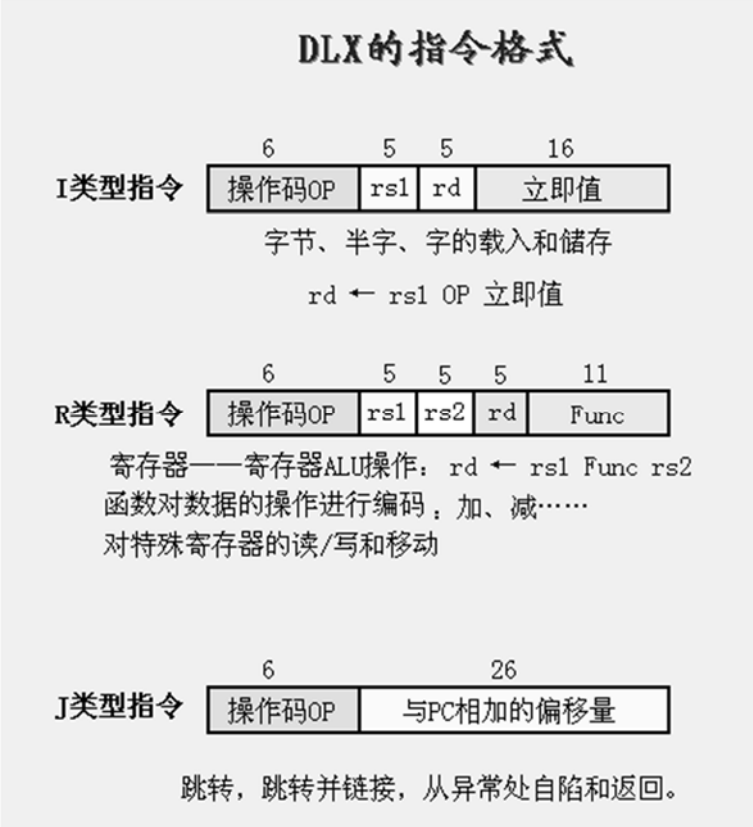

图 1

5. DLX 中的操作

DLX 指令中的操作可以分为四种类型，即：Load 和 Store 操作、ALU 操作、分支和跳转操作、浮点操作。

(1)Load 和 Store 操作

可以对 DLX 的所有通用寄存器和浮点寄存器进行 Load（载入）和 Store（储存）操作，但是通用寄存器 R0 的 Load 操作没有任何效果。

(2) ALU 操作

在 DLX 中，所有的 ALU 指令都是寄存器－寄存器型指令，其运算包含了简单的算术和逻辑运算，如加、减、AND、OR、XOR 和移位。另外，DLX 还允许所有这些指令对立即值进行操作，立即值以 16 位符号扩展形式出现。LHI（Load 高位立即值）操作将立即值载入到一个寄存器的高半部分，而该寄存器的低半部分则设置为 0。这样就可以通过两条 Load 指令构造一个 32 位的常数。

正如上面所提到的，R0 主要用来合成一些有用的操作。比如，Load 一个常数就可以看作是

一次简单的立即值加操作，其中一个源操作数是 R0；寄存器－寄存器间的数据移动也可以看作是一次简单的加，其中一个源操作数是R0。这两个操作可以分别用LI和MOV表示。

在DLX指令集中，还有一些寄存器比较指令（=，≠，<，>，≤，≥），如果比较结果为真，这些指令就在目标寄存器中填入1（表示真），否则填入0（表示假）。因为这些比较操作指令要对目标寄存器进行“设置”，所以也称它们为设置相等、设置不等、设置小于等指令。

(3) 分支和跳转操作

在 DLX 中，对程序流程的控制是通过一些跳转和分支指令来实现的。根据描述目标地址的方法和是否链接可以将跳转操作指令分为四种类型。其中两种类型的跳转指令用带符号位的 26 位偏移量加上程序计数器的值来确定跳转的目标地址,另外两种类型的跳转指令则指定一个寄存器，由寄存器中的内容决定跳转的目标地址。跳转有两种类型，一种是简单跳转，另一种是跳转并链接（用于过程调用），后者将返回一个地址，即将下一条顺序指令地址（返回地址）保存在寄存器R31中。

DLX中的所有分支指令均是条件分支指令，其源操作数寄存器中包含了一个数值或某个比较结果。分支指令测试该源操作数寄存器中的值是0还是非0，决定分支是否成功。分支目标地址由一个带符号的26位偏移量加上程序计数器的值来确定，分支目的地址指向下一条要执行的指令

(4) 浮点操作

在DLX中，浮点指令的操作数来源于浮点寄存器，同时该浮点指令还指明了相应的操作是单精度浮点操作还是双精度浮点操作。

DLX的浮点操作有：加、减、乘、除。后缀D代表双精度浮点操作，而后缀F代表单精度浮点操作（如：ADDD、ADDF、SUBD、SUBF、MULTD、MULTF、DIVD、DIVF）。值得提出的是，DLX 的浮点比较操作设置浮点状态寄存器中的位，如果比较结果为真，则将该位设置为1；如果比较结果为假，则将该位设置为0。浮点分支指令BFPT和BFTF则测试该寄存器的值来决定分支是否成功。

另外，操作 MOVF 将一个单精度浮点寄存器的内容拷贝至另一个单精度浮点寄存器； MOVD 则将一个双精度浮点寄存器的内容拷贝至另一双精度寄存器；MOVFP2I和MOVI2FP操作则是在一个浮点寄存器和通用寄存器之间移动数据,如果要将一个双精度浮点数移入两个通用寄存器则需要两条指令，另外DLX还提供了在32位浮点寄存器中进行整数乘除操作的指令。

第二节　DLX流水线结构

一．一个简单实现

为了说明指令的流水执行方式，先论述在不流水的情况下，DLX 指令是如何执行的。图 2 给出了实现 DLX 指令的一种简单数据通路，下面可以看出在五个时钟周期内可以完成一条 DLX

指令。

1．取指令周期（IF）：

IR ← Mem[PC]NPC ← PC＋4

其操作为：根据PC值从存贮器中取出指令，并将指令送入指令寄存器IR;PC值增加4,指向顺序的下一条指令，并将下一条指令的地址放入临时寄存器NPC中.

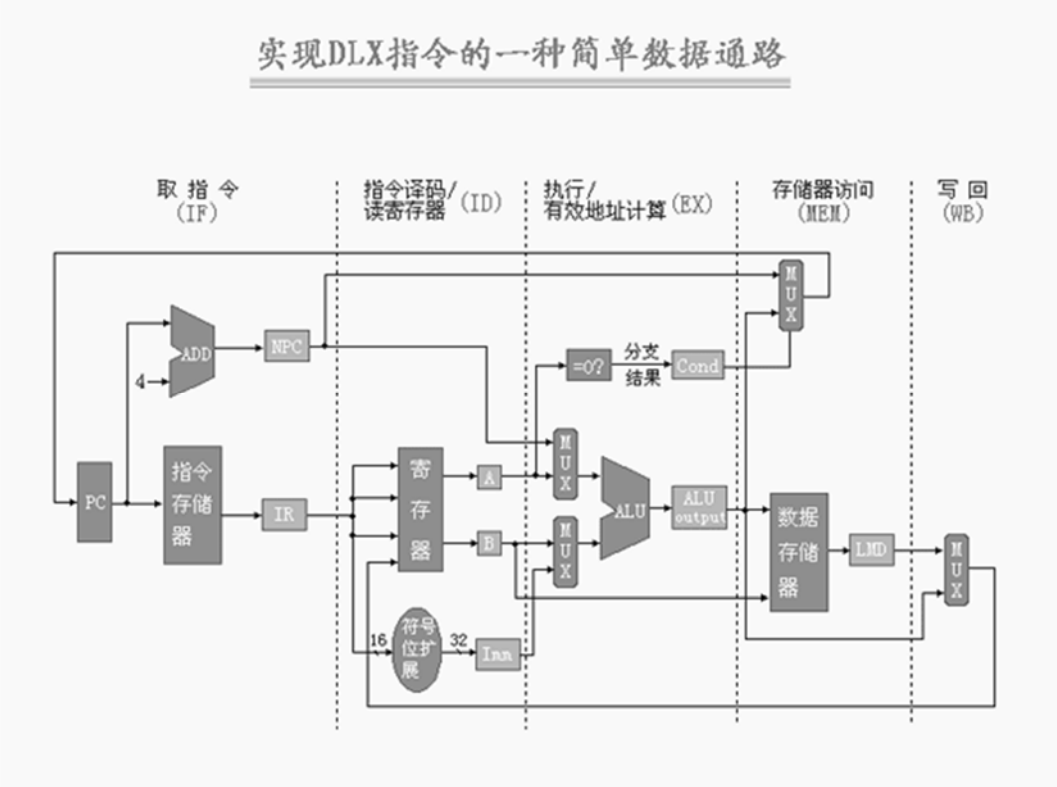

图2

2．指令译码/读寄存器周期（ID）：

A ←Regs[IR6..10]

B ←Regs[IR11..15]

Imm←(IR16)16 ## IR16..31

其操作为：对指令操作码进行译码，按照给定的寻址方式和地址段中的内容形成操作数的地址，并用这个地址读操作数。操作数可能在驻村中，也可能在通用寄存器中。[8]

指令的译码操作和读寄存器操作是并行进行的。之所以能做到这一点，是因为DLX指令格式中，操作码在固定位置。这种技术也称为固定字段译码技术。值得注意的是，在上述过程中，可能读出了一些在后面周期中并不会使用到的寄存器内容，但是这并不会影响指令执行的正确性。相反，却可以有效地降低问题的复杂性。

另外，由于立即值在DLX指令格式中处于固定位置，因此这里也对其进行符号扩展，以便在下一个周期能使用它。当然由于指令的不同，也许在后面的周期中并不会用到这个立即值，但无论如何，提前形成立即值总是有益无害的。

3．执行/有效地址计算周期（EX）在这个周期，不同的指令有不同的操作。

（1）存储器访问：

ALUOutput←A＋Imm

当指令为存贮器访问指令时，该周期的操作为：ALU将操作数相加形成有效地址,并将结果放入临时寄存器ALUoutput中。

（2）寄存器－寄存器ALU 操作：

ALUOutput←A opB

当指令为寄存器---寄存器ALU操作指令时，该周期的操作为：ALU根据操作码指出的功能对临时寄存器A和B中的值进行处理，并将结果送入临时寄存器ALUoutput中.

（3）寄存器－立即值ALU 操作：

ALUOutput←A opImm

当指令为寄存器---寄存器ALU操作指令时，该周期的操作为：ALU根据操作码指出的功能对临时寄存器A和Imm中的值进行处理，并将结果送入临时寄存器ALUoutput中.

（4）分支操作

ALUOutput←NPC＋Imm

Cond←(A op 0)

当指令为分支指令时，该周期的操作为：ALU 将临时寄存器NPC和Imm中的值相加，得到分支的目标地址。同时，对在前一个周期读入到寄存器A的值进行检查,决定分支是否成功。OP由分支操作码决定

这里，将有效地址计算周期和执行周期合并为一个时钟周期，这是由 DLX 指令集结构本身的特点所允许的，因为在 DLX 指令集结构中，没有任何指令需要同时计算数据的存储器地址、计算分支指令的目标地址和对数据进行处理。另外，上面四种操作类型中没有包含各种形式的跳转操作，它们和分支操作十分相似，这里就不再赘述

4．存储器访问/分支完成周期（MEM）

在该周期处理的DLX指令只有Load、Store和分支指令。

存储器访问：LMD ←Mem[ALUOutput]或Mem[ALUOutput] ←B

存贮器的访问操作包含了Load和Store两种类型的操作。如果指令是Load指令，就将临时寄存器ALUOutput中的值作为访存地址，从存贮器中读出相应的数据，并放入临时寄存器LMD中；如果指令是Store指令，就将临时寄存器中的值按照临时寄存器ALUOutput所知名的地址写入存贮器。

(1) 分支操作：if（cond）PC ←ALUOutput elsePC ←NPC

如果分支条件寄存器中的内容为真，表明分支转移成功，选择临时寄存ALUOutput中的值作为分支目标地址，并将其放入PC中。否则，他将临时寄存器NPC中的值送入PC中，作为下一条指令地址。

5．写回周期（WB）：

不同指令在该周期完成的工作也不一样。这里按如下指令类型对写回周期所要完成的工作进行说明。

（1）寄存器－寄存器型ALU 指令：Regs[IR16..20] ←ALUOutput

（2）寄存器－立即值型ALU 指令：Regs[IR11..15] ←ALUOutput

（3）Load指令：　　Regs[IR11..15] ← LMD

上述指令均是将结果写入寄存器文件。无论结果是来自于存储器系统（临时寄存器 LMD 中的内容），还是来自于 ALU 的计算结果（临时寄存器 ALUoutput 中的内容），都由操作码决定将其送入目标寄存器相应的域中

从图1--2中可以看出，分支指令需要4个时钟周期完成，其它指令需要5个时钟周期完成。假设分支指令数占指令总数（也称混合指令数）的12%，则其 CPI 是4.88个时钟周期。由此可见，上述实现无论是从性能方面，还是从硬件开销方面，都称不上是一种优化实现。

由于 ALU 指令在 MEM 周期不做任何工作，所以可以在 MEM 周期就完成 ALU 指令，这并不影响执行部件的时钟速度，但是可以降低 CPI。假设ALU指令数占混合指令条数的44%，则CPI 可达4.44，这种改进所带来的加速比为4.88/4.44=1.1。

如果还要降低 CPI，就有可能会延长时钟周期时间，因为在每个时钟周期中可能需要完成更多的工作，因此，时钟周期时间和 CPI 之间也存在着折衷关系，这种折衷取舍需建立在对系统仔细分析的基础之上。

二．DLX 基本流水线

我们可以将每一个时钟周期看作是流水线的一个时钟周期，使图 2 中的数据通路成为一条指令流水线。硬件每个时钟周期启动一条新的指令，并执行 5 条不同指令中的某一部分。我们可以把该流水线描述成为图 3 所示的流水过程。

一种简单DLX流水线的流水过程

	时钟周期								
指令编号	1	2	3	4	5	6	7	8	9
指令i	IF	ID	EX	MEM	WB				
指令i+1		IF	ID	EX	MEM	WB			
指令i+2			IF	ID	EX	MEM	WB		
指令i+3				IF	ID	EX	MEM	WB	
指令i+4					IF	ID	EX	MEM	WB

图 3 简单的 DLX 流水线

从图 3 可以看出，每条指令仍然需要 5 个时钟周期完成，但是指令执行的吞吐量却有很大提高，虽然DLX流水线十分简单，但是要使DLX指令的各种组合能够在上述流水线中真正流水起来，充分发挥流水线的效率并不是容易的事情，它必须解决各种相关问题。

实验一 WINDLX 模拟器安装及使用

一、 实验类别： 综合实验
二、 实验目的：建立实验环境，了解 WINDLX 模拟器的结构及使用
三、 实验学时：2
四、 人组数：1/1
五、 实验设备环境：WinDLX 要求的硬件平台是 IBM PC 兼容机，WinDLX 是一个 Windows 应用程序，运行于 Windows 3.0 以上的操作系统。
六、 实验原理：WinDLX 软件包中带有说明文件，供安装程序时候使用。
七、 教学要点与学习难点：软件包中还有 WinDLX 教程和联机帮助，可以通过它们进
八、 一步了解模拟器的使用方法和 DLX 处理器的原理。大家在进行模拟实验以前应该
九、 仔细阅读这些文档。
十、 实验内容和要求：阅读模拟器 Help 文档和相关资料，利用 Fact.s 及 Input.s 代
十一、 码熟悉模拟器的配置、各项工具使用、寄存器设置及指令系统。

十二、 实验步骤：
（1） WINDLX 模拟器安装。
（2） 熟悉模拟器的配置。
（3） 熟悉各工具的使用。

十三、 可研究与探索的问题：

十四、 实验报告要求：简要介绍 WINDLX 模拟器结构和功能

实验二　指令流水线相关性分析

一、 实验类别：验证实验

二、 实验目的：通过使用 WINDLX 模拟器，对程序中的三种相关现象进行观察，并对使用专用通路，增加运算部件等技术对性能的影响进行考察，加深对流水线和 RISC 处理器的特点的理解。

三、 实验学时：4

四、 实验组人数：1/1

五、 实验设备环境：WinDLX 模拟器可以装入 DLX 汇编语言程序，然后单步、设置断点或者连续执行该程序。CPU 的寄存器、流水线、I/O 和存储器都可以使用图形的方式表示出来。模拟器还提供了对流水线操作的统计功能。该模拟器对理解流水线和 RISC 处理器的特点很有帮助。

六、 实验原理：指令流水线中主要有结构相关、数据相关、控制相关。相关影响流水线性能。

七、 教学要点与学习难点：三种相关及其解决办法

八、 实验内容和要求：使用 WinDLX 模拟器，对求阶乘程序 Fact.s 做分析

九、 实验步骤：

（1） 观察程序中出现的数据/控制/结构相关。指出程序中出现上述现象的指令组合。

（2） 考察增加浮点运算部件对性能的影响。

（3） 考察增加 forward 部件对性能的影响。

（4） 观察转移指令在转移成功和转移不成功时候的流水线开销。

注意：除（2）以外，浮点加、乘、除部件都只有一个；
本问题中所有浮点运算部件的延时都请设定为 4 个周期。

十、 可研究与探索的问题：

十一、 实验报告要求：

（1） 实验目的。

（2） 实验原理。

（3） 针对上面的实验内容，记录实验过程，给出分析结果，相关理论依据。

（4） 给出实验总结：根据实验，总结采用流水线技术会遇到的问题和为解决这些问题所采用的各种技术的作用。同时简单谈谈自己对流水线技术的认识。

实验三　　DLX 处理器程序设计

一、　实验类别：综合型
二、　实验目的：学习使用 DLX 汇编语言编程，进一步分析相关现象
三、　实验学时：4
四、　实验组人数：1/1
五、　实验设备环境：
　　DLX 汇编语言环境
六、　实验原理：掌握向量运算算法和编程方法。
七、　教学要点与学习难点：DLX 汇编语言
八、　实验内容和要求：
　　自编一段汇编代码，完成两双精度浮点一维向量的加法(或乘除法)运算，并输出结果。向量长度>=16。观察程序中出现的数据/控制/结构相关
九、　实验步骤：
　　（5）　熟悉 DLX 汇编语言。
　　（6）　编写两双精度浮点一维向量的加法运算程序。
　　（7）　对此程序完成上面实验二中 1)、2)、3)、4)方面的分析。

十、　可研究与探索的问题：
十一、　实验报告要求：
　　（1）　实验目的。
　　（2）　代码清单及注释说明。
　　（3）　程序相关性分析结果。

实验四　　代码优化

一、　实验类别：综合实验
二、　实验目的：学习简单编译优化方法，观察采用编译优化方法所带来的性能的提高。
三、　实验学时：4
四、　实验组人数：1/1
五、　实验设备环境: DLX 汇编语言环境
六、　实验原理：采用静态调度方法重排指令序列，减少相关，优化程序
七、　教学要点与学习难点：指令静态调度方法。
八、　实验内容和要求：对实验二或实验三的代码进行优化，给出性能改进的量化值，同时给出采取优化手段的理论依据。
九、　实验步骤：
（1）　使用静态调度方法手工优化实验 2 或实验 3 的代码
（2）　对优化程序，重复实验二中（1）、（2）、（3）、（4）工作。

十、　可研究与探索的问题：
十一、　实验报告要求：
（1）　实验目的
（2）　实验原理
（3）　优化程序代码清单及注释说明
（4）　参照实验要求记录你实验分析结果
（5）　你解决的困难和解决方法
（6）　你的实习体会（如有）
（7）　你对该实习的建议

实验五　　循环展开（选作）

一、　实验类别：综合实验

二、　实验目的：进一步学习 DLX 汇编语言编程方法，学习循环展开编译优化方法，观察采用循环展开编译优化方法所带来的性能的提高。

三、　实验学时：2

四、　实验组人数：1/1

五、　实验设备环境：DLX 汇编语言环境

六、　实验原理：对循环程序采用循环展开（loop unrolling）方法进行优化，减少相关。

七、　教学要点与学习难点：

（1）　矩阵乘算法

（2）　循环展开优化编译方法

八、　实验内容和要求：使用 DLX 汇编语言编写二维矩阵相乘程序，观察相关性；再用循环展开方法手工该优化程序，分析比较性能的改进

九、　实验步骤：

（3）　编写矩阵相乘程序

（4）　重复实验二中（1）（2）（3）（4）工作。

（5）　使用循环展开手工优化程序

（6）　对优化程序，重复实验二中（2）工作。

（7）　对优化程序，将浮点部件的延迟改为 8 个时钟周期，再重复实验二中（2）工作。观察实验现象并分析原因。

注意：为了简单起见，可以固定矩阵的大小，例如 10×10，可以不赋初值，不输出计算结果，目的仅仅是为了考察矩阵相乘的指令序列。

十、　可研究与探索的问题：

比较 Intel X86 汇编语言程序与 DLX 汇编语言程序的风格的不同。

十一、　实验报告要求：

（1）　实验目的

（2）　实验原理

（3）　矩阵乘程序代码清单及注释说明

（4）　优化程序代码清单

（5）　未优化代码和优化代码性能分析比较结果

（6）　你解决的困难和解决方法

（7）　你没有解决的困难（如有）以及你做过的努力

（8）　你的实习体会（如有）

（9）　你对该实习的建议