

# 微程序设计及其仿真技术

复旦大学计算机科学系

徐公权 吴时霖 编著  
王德新 顾宝发

# **微程序设计及其仿真技术**

复旦大学计算机科学系

徐公权 吴时霖 编著  
王德新 顾宝发

\*

上海科学技术文献出版社出版发行  
(上海市武康路2号)

新华书店经销

昆山亭林印刷厂印刷

\*

开本 787×1092 1/32 印张 11.5 字数 278,000

1989年2月第1版 1989年2月第1次印刷

印数: 1—3,200

ISBN 7-80513-175-9/T·53

定价: 5.90元

《科技新书目》17.-241

# 前 言

一个好的计算机系统是软硬件技术紧密结合的产物。介于硬件技术和软件技术之间的微程序技术(或称固件技术)是整个计算机系统的核心,它已越来越广泛地、深入地应用于计算机的设计、使用和维护等领域。作为微程序设计的最重要应用之一——计算机系统仿真(或称微程序仿真)已普遍引起人们的关注。面对各种丰富而又宝贵的软件资源,解决新老计算机系统(或不同计算机系列)之间的“程序兼容性”问题是计算机系统设计师的一个重要任务,近20年来的实践证明:微程序仿真是解决这种程序兼容性的最有效手段。

本书主要以 VAX-11/780 计算机系统为背景,以 MICRO2 微语言为工具,介绍了微程序仿真技术的一般原理、方法及其实现。全书共七章,第一章是概论。第二章描述了 DEC 公司近期推出的 MICRO2 通用微汇编语言。第三章论述了微程序仿真的一般方法和原理。第四章介绍了 VAX-11/780 计算机的数据通路。第五章是在二、三、四章基础上,实现了 VAX-11/780 仿真 Sample 机,给出了仿真的流程及其微程序。第六章以初等函数统一算法和快速富里埃变换为例说明了用户微程序设计。第七章讲述了仿真微程序的开发、调试和维护及其在 VAX-11/780 上的操作过程。最后,附录 A、B 和 C 分别给出了 VAX-11/780 的定义文件,Sample 机的仿真微程序的列表文件和它的目标文件。

本书的第一、二、三、七章由徐公权编写,第四章由顾宝发编

写,第五章由王德新编写,第六章由吴时霖编写,黄德利也参加了部分工作。本书是我们长期从事微程序方面教学工作和科研工作的产物,书中不少内容取材于有关科研成果与论文。限于作者的水平和经验,书中难免存在一些缺点和错误,敬请广大读者批评指出。

作 者 1987年8月

# 目 录

|                                 |           |
|---------------------------------|-----------|
| <b>第一章 概论 .....</b>             | <b>1</b>  |
| 1.1 微程序设计技术的发展阶段.....           | 1         |
| 1.1.1 概念确立阶段(1951~1964年) .....  | 1         |
| 1.1.2 普及应用阶段(1964~1970年) .....  | 2         |
| 1.1.3 飞速发展阶段(1970~现在) .....     | 6         |
| 1.2 微程序设计技术的评价.....             | 8         |
| 1.3 微程序迁移 .....                 | 12        |
| <b>第二章 MICRO2 通用微汇编语言 .....</b> | <b>16</b> |
| 2.1 引言 .....                    | 16        |
| 2.2 标识微程序 .....                 | 18        |
| 2.2.1 位计数方向.....                | 18        |
| 2.2.2 程序基数.....                 | 19        |
| 2.2.3 其它源微程序标识关键字.....          | 19        |
| 2.3 定义字段及其值 .....               | 22        |
| 2.3.1 MICRO2 名字.....            | 22        |
| 2.3.2 定义段名.....                 | 24        |
| 2.3.3 限定词.....                  | 25        |
| 2.3.4 定义值名.....                 | 31        |
| 2.4 表达式和有效性检验 .....             | 33        |
| 2.4.1 表达式(expression) .....     | 33        |
| 2.4.2 有效性表达式.....               | 39        |
| 2.5 宏定义 .....                   | 41        |
| 2.5.1 引言.....                   | 41        |
| 2.5.2 宏名.....                   | 43        |

|        |                    |     |
|--------|--------------------|-----|
| 2.5.3  | 宏体(macro-body)     | 43  |
| 2.5.4  | 参数(parameter)      | 45  |
| 2.6    | 微指令                | 47  |
| 2.6.1  | 微指令                | 47  |
| 2.6.2  | 微字                 | 48  |
| 2.7    | 微地址分配              | 50  |
| 2.7.1  | 定义微地址空间            | 51  |
| 2.7.2  | 规定微地址分配的方式         | 52  |
| 2.8    | 条件汇编               | 59  |
| 2.9    | MICRO2 输出文件和列表控制   | 65  |
| 2.9.1  | MICRO2 输出文件        | 65  |
| 2.9.2  | 列表控制               | 67  |
| 2.10   | 对 MICRO2 语言的评价     | 70  |
| 2.11   | 实例                 | 73  |
| 2.11.1 | 模型机的数据通路           | 73  |
| 2.11.2 | 标识和字段定义            | 74  |
| 2.11.3 | 实例宏定义              | 76  |
| 2.11.4 | 一个微子程序             | 76  |
| 2.11.5 | 另一个微子程序            | 78  |
| 第三章    | 仿真方法               | 80  |
| 3.1    | 引言                 | 80  |
| 3.2    | 仿真器(emulator)的设计方法 | 83  |
| 3.2.1  | 仿真器的作用             | 83  |
| 3.2.2  | 仿真器的类型             | 85  |
| 3.2.3  | 一级微程序设计和二级微程序设计    | 87  |
| 3.2.4  | 仿真器的主要功能部件         | 94  |
| 3.3    | 计算机系统仿真微程序的总体设计策略  | 97  |
| 3.3.1  | 兼容方案               | 97  |
| 3.3.2  | 独立方案               | 111 |

|                                       |            |
|---------------------------------------|------------|
| 3.3.3 混合方案 .....                      | 112        |
| 3.4 环境映象 .....                        | 113        |
| 3.5 目标机指令系统和中断系统的仿真 .....             | 117        |
| 3.5.1 指令系统的仿真 .....                   | 117        |
| 3.5.2 中断处理 .....                      | 119        |
| 3.6 专用外设的仿真 .....                     | 125        |
| 3.6.1 适配器的设计 .....                    | 126        |
| 3.6.2 专用外设的输入/输出中断处理 .....            | 133        |
| 3.7 仿真微程序的进入和退出 .....                 | 135        |
| 3.7.1 进入 .....                        | 136        |
| 3.7.2 退出 .....                        | 138        |
| <b>第四章 VAX-11/780 数据通路与定义文件 .....</b> | <b>140</b> |
| 4.1 VAX-11/780 数据通路 .....             | 140        |
| 4.1.1 运算部分 .....                      | 140        |
| 4.1.2 地址部分 .....                      | 149        |
| 4.1.3 数据部分 .....                      | 152        |
| 4.1.4 阶码部分 .....                      | 159        |
| 4.1.5 VAX-11/780 数据通路评述 .....         | 161        |
| 4.2 VAX-11/780 定义文件简介 .....           | 163        |
| <b>第五章 计算机系统仿真微程序实例——VAX-11/780</b>   |            |
| <b>仿真 Sample 机 .....</b>              | <b>169</b> |
| 5.1 Sample 机的特点 .....                 | 169        |
| 5.2 Sample 机的映象 .....                 | 175        |
| 5.3 Sample 机系统仿真的实现 .....             | 177        |
| 5.3.1 仿真微程序的接口程序功能 .....              | 178        |
| 5.3.2 Sample 机系统仿真微程序的功能及流程 .....     | 181        |
| <b>第六章 用户微程序设计实例——初等函数统一算法及其</b>      |            |
| <b>实现 .....</b>                       | <b>204</b> |
| 6.1 坐标旋转与叠代方程 .....                   | 204        |

|       |                            |     |
|-------|----------------------------|-----|
| 6.2   | 收敛方案                       | 212 |
| 6.3   | CORDIC 算法实现的框图             | 215 |
| 6.4   | 离散的富里埃变换 DFT               | 218 |
| 第七章   | 仿真微程序的开发、调试和维护             | 220 |
| 7.1   | 固件工程概述                     | 220 |
| 7.2   | 模块结构微程序设计                  | 225 |
| 7.2.1 | 基本控制结构及其在 VAX-11/780 上的实现  | 225 |
| 7.2.2 | 模块结构微程序设计实例                | 232 |
| 7.3   | 真微程序的开发环境                  | 237 |
| 7.3.1 | WCS 的测试方法                  | 237 |
| 7.3.2 | 微语言                        | 241 |
| 7.3.3 | WCS 的装入程序                  | 242 |
| 7.3.4 | 调试命令                       | 242 |
| 7.3.5 | 微程序模拟系统                    | 244 |
| 7.4   | 在 VAX-11/780 上开发用户仿真微程序的过程 | 247 |
| 附录 A  | VAX-11/780 定义文件和宏定义        | 253 |
| A.1   | VAX-11/780 定义文件            | 253 |
| A.2   | VAX-11/780 宏定义             | 275 |
| 附录 B  | SAMPLE 仿真微程序列表文件           | 317 |
| B.1   | 目录表                        | 317 |
| B.2   | 源程序列表文件                    | 318 |
| B.3   | 符号交叉引用表                    | 342 |
| B.4   | 控存映象表                      | 346 |
| B.5   | 控存占用情况及编译错误信息汇总表           | 349 |
| 附录 C  | SAMPLE 仿真微程序目标文件 (.ULD)    | 350 |

# 第一章 概 论

## 1.1 微程序设计技术的发展阶段

四十多年来,微程序设计技术经历了确立概念、普及应用以及飞速发展等三个阶段,现简要叙述如下。

### 1.1.1 概念确立阶段(1951~1964 年)

在第一台电子计算机 ENIAC 诞生五年后的 1951 年,英国剑桥大学数学教研室教授威尔克斯(M. V. Wilks),在曼彻斯特大学计算机会议上,第一次提出了微程序的概念和原理。威尔克斯在题为“设计自动化计算机的最好方法”的报告中指出,一条机器指令可以分割为许多更基本的操作序列,计算机的操作可以归结为信息传送,而信息传送的关键是控制门,这些门可以用存放在某种存贮阵列中的信息位来控制。因此他认为,可以用一种有规则的、类似于程序设计的方法,来设计计算机繁杂的控制逻辑,这种方法就是属于存贮控制逻辑概念的微程序设计方法。

威尔克斯还亲自指导了剑桥大学的微程序设计实践,他的方案首先在剑桥大学的微程序设计的 EDSAC<sub>2</sub> 系统中实现。威尔克斯方案后来被称为“威尔克斯模型”。

1951 年到 1964 年,正是计算机经历第一代和第二代的发展阶段。在这个阶段中,计算机设计者发明了变址器、堆栈结构、程序中断系统,同时,出现了计算机族,开始研究分时系统;初步确立了系统软件;…。这个时期的存贮器,从水银延迟线、光屏管、磁鼓(作主存用),发展到全面采用磁芯存贮器。本来,这个时期计算机技术的发展是需要微程序设计技术的,但是,这个时期并

没有为实现微程序设计技术创造客观条件。这是因为,上述存贮器价格贵、速度慢。如果把上述存贮器用作微程序控制存贮器,就体现不出微程序设计的优越性。然而,其它类型的存贮器这时还处在研究阶段。所以,这个时期的微程序设计技术,由于缺乏廉价、可行的控制存贮器而发展缓慢。只有极少数单位做了一些很原始的微程序计算机模型。比较有名的有:如上述的EDSAC<sub>2</sub>(用1K容量的只读型磁芯矩阵做控制存贮器);美国麻省理工学院林肯实验室的CQ<sub>24</sub>(用1K容量的只读型二极管矩阵做控制存贮器)等。美国IBM公司是最早重视微程序设计技术的大公司之一,在1961年的ACM全国会议上,曾经提出过有关IBM7950机微程序设计的论文。但是,在这个时期,IBM公司也还不能解决控制存贮器的问题。

尽管在这个时期微程序设计技术发展缓慢,但是,威尔克斯的概念已经引起人们的重视,英、美、日、意、法和苏联等国,已经逐渐有人开始从理论上对微程序设计技术进行探讨。值得指出的是,威尔克斯一开始只是注重于用规整的存贮控制逻辑,来代替繁杂的组合控制逻辑,微程序设计也只是被看作为实现硬件设计的一种手段。格兰茨、默塞尔、范德普尔等人,曾先后著文,发表了通过修改或改编微程序来改变指令系统,或者给用户提供一个更大更好的指令系统的见解,这种见解颇有远见。

### 1.1.2 普及应用阶段(1964~1970年)

1964年到1970年间,正是计算机经历以采用中、小规模集成电路为主要标志的第三代发展阶段。在这个阶段中,计算机技术取得了重大的进展,从而使系统硬件达到了阶段性的相对稳定,使系统软件付诸实用。在系统硬件方面的主要进展有:确立了“计算机结构”的概念;采用了通道、虚拟存贮器或虚拟机等概念和技术;开始研究并初步实践计算机网;采用了先行控制、

高速缓冲存储器、流水线、阵列式结构等技术。这样一来,便使计算机结构突破了串行式的冯·诺依曼结构,大大地提高了计算机的速度和效率等等。在这个阶段中,计算机技术的另一方面的重大进展就是:产生了系列机;发展了仿真和模拟技术;提高了机器的可靠性、可利用率和可维护性(即RAS技术)。这些方面的重大进展,是导致普及应用微程序设计技术的直接因素。而这个阶段的存储技术的进展,则是普及应用微程序设计技术的客观基础。下面具体分析一下促使微程序设计技术进入普及应用阶段的主要原因。

### 1. 研制系列机的需要

系列机的主要标志之一,就是在系列内的各个型号的计算机都具有程序兼容性。这就要求各个型号都要遵循统一的结构格式,特别是要求实现统一的指令系统。由于系列机往往要兼顾科学计算、数据处理和过程控制等多方面的用途,所以,这个统一的指令系统也就必然设计得很庞大和复杂。面对这个统一的结构格式和指令系统,系列机中的各个型号都必须有适当的性能/价格比,这样才能确保各种型号都有生存力。例如IBM360系统,一开始共分10个型号,从小型计算机到大型计算机的性能比是1比300,价格比则是1比100,对于这种系统,若用常规硬件组合逻辑的设计方法,则难以确保既定的性能/价格比,若采用微程序设计技术,则能确保这个性能/价格比。这是由实践所证实了的。另一方面,微程序设计还使系列机具有如下潜力:为了适应各种用途,可以灵活地扩充性能。

### 2. 仿真和模拟技术的需要

计算机越来越多,其设计过程也越来越复杂,这就促使人们去寻求新途径,以便解决如下两个问题:

#### 1) 希望过去的程序能在新一代的机器中运行;

2) 希望借助计算机来分析、评价系统的设计方案, 判断设计的正确性, 以缩短研制周期。

要解决上述两方面的问题, 其主要办法就是采用仿真和模拟技术。在微程序设计的计算机中, 由于微程序把指令操作分割得更独立、更细小, 所以用微程序来仿真就比用程序模拟更容易、更准确、更快速。若微程序仿真的对象是研制中的机器, 则通过仿真便可以评价设计方案、判断设计的正确性, 指出修改的途径, 从而大大缩短研制周期。若仿真的对象是已有的计算机, 则已有的计算机程序就可以在进行仿真的机器(宿主机)中运行, 即便是第二代的程序, 也可以很容易地在第三代的微程序仿真机中执行。

### 3. RAS 技术的需要

微程序设计的计算机具有如下优点: 结构规整, 更加可靠, 设计灵活, 修改方便, 学习容易, 便于诊断。例如, 在微程序设计的计算机中, 无需增加太多的硬件就能进行微诊断。这种微诊断, 能够把故障定位在很小的范围内。这些优点正是 RAS 技术所要求的。

正是由于上述三方面的需要, 才导致了微程序设计技术迅速进入普及应用阶段。从1964年起陆续投产的IBM360系统的各种型号, 除超大型机外, 都普遍成功地采用了微程序设计技术。这是微程序设计技术进入普及应用阶段的主要标志。IBM370系统则全部采用微程序设计。1970年, 哈森在他的著作“微程序设计——原理和实践”中, 曾经比较全面地介绍了IBM360系统的微程序设计。

必须指出, 威尔克斯本人和在IBM等公司工作的许多人, 都对“威尔克斯模型”进行了许多重大的改进和发展, 从而才使得微程序设计技术走上了实用的轨道, 迎来了普及应用的新阶

段。这些改进和发展,已经反映在哈森的著作中,主要有以下几方面:

### 1. 确立了微指令的基本编译法

在 IBM 系统中,除 20 型、25 型等小型机采用垂直微指令以外,一般都采用水平微指令。在水平微指令中,采用了既能大大缩短微指令字长,又能达到一定执行速率的字段直接和间接编译法。此外,IBM360 系统还采用了许多技巧(如 40 型的间接操作控制等等)。

### 2. 确立了微程序的顺序控制方式

一方面,IBM360 系统通过采用增量方式和分段断定方式,来确定后继微地址,从而缩短了微指令的顺序控制字段的长度。另一方面,IBM360 系统的微指令,普遍设置了灵活的“转移条件选择测试字段”和“转移控制字段”等,使微程序具有如下功能:能够很灵活地顺序执行;可以进行无条件转移;可以进行 2 路到 64 路的条件转移;可以方便地构成循环和微子程序等等。这样一来,就可以大大提高微程序的执行速度。

### 3. 初步实现微程序设计自动化

IBM360 系统的微程序设计,采用微汇编语言和框图语言,建立了“控制自动化系统”CAS,初步实现了微程序设计自动化,并开始研究高级微程序设计语言。

### 4. 初步实践了仿真和模拟技术

### 5. 解决了微程序控制存贮器问题

IBM360 系统普遍采用廉价可行的变压器和静电耦合型只读存贮器,以作为微程序控制存贮器。换句话说,IBM 公司解决了微程序设计技术赖以进入普及应用阶段的控制存贮器问题。

从六十年代中期起,微程序设计的计算机已经成批地投入

了使用。这个阶段有代表性的微程序设计的计算机,除 IBM360 系统外,还有 RCA Spectra70/45, H4200/8200 以及 MELCOM 1530 等等。

### 1.1.3 飞速发展阶段(1970~现在)

从七十年代开始,电子计算机跨入了第四代发展阶段。这一代计算机的重要标志之一就是采用大规模集成电路 LSI。LSI 不仅成本低、体积小、功耗小,以及可靠性高等多方面优点,而且它开拓了计算机结构进一步创新的前景。

这个阶段的 LSI 存贮器,淘汰了变压器型和静电耦合型等只读存贮器 ROM,只读型的 LSI 和 VLSI 存贮器,由于它的廉价、快速、可靠和灵活等许多优点,所以它大大地促进了微程序设计技术的发展和运用。而可写可读型的 LSI、VLSI 存贮器,还为这门技术的进一步创新和应用,提供了物质基础。采用可写可读型控制存贮器的机器,具有可扩展性和能由用户选择机器性能的结构。未来的这种机器,甚至可望具有动态的结构。

微程序设计技术飞速发展的另一个重要方面,就是研究学术活动的蓬勃开展。从 1968 年起,“IEEE 计算机协会微程序设计技术委员会”(IEEE Computer Society Technical Committee on Microprogramming)和“ACM 微程序设计特设专业组”(ACM Special Interest Group on Microprogramming)每年联合举办一次微程序专业会议,并出版会刊“微会记录”(Proceeding of Micro)。1970 年以来,ACM 还创立季刊“SIG 微新消息”(SIG Micro Newsletter)。此外,许多期刊和会议录,例如“IEEE 计算机会报”(IEEE Transaction on Computer),“计算机设计”(Computer Design),“AFIPS 会议录”(AFIPS Conference Proceedings)等等,也都不定期地刊登了微程序设计专集或者文章。在涉及微程序设计技术的领域内,迄今为止

发表过的文献已有几千篇。在这些文献中,更引人注目的课题是固件工程(包括高级微程序设计语言,微码压缩与优化,微程序正确性验证,微程序模拟等);微程序的垂直迁移(Vertical Migration),或称软件固化;微程序设计技术和大规模集成电路对未来计算机结构的影响等等。

微程序设计技术飞速发展的又一个重要方面,就是开始使用高级微程序设计语言。例如:Burroughs公司的MIL是为其B1700/1800系列设计的微语言;Microdata公司的MPL是为其栈结构机器Microdata32/S设计的类似于PL/I的语言,MPL的每种语法结构都与32/S的微结构部件相对应;D. A. Patterson的结构微语言STRUM虽然是专为Burroughs D机器设计的一种与PASCAL类似的语言,但却是目前公认的最成功的高级微语言之一;此外,最近发表的Burroughs B5900专用的EMPL和面向Nonstop微型机的Micro TAL都属于与机器有关的高级微语言范畴。R. Eckhouse提出的MPL是第一种企图设计得与机器无关的高级微程序语言,迈出了确有意义的第一步,但是它的最大弱点在于没有提供强有力的微码压缩手段,因此难以生成高效率的目标微码;O. V. Ramamoorthy等提出了一种SIMPL语言,这种语言与ALGOL60类似,它的意义在于第一次将一种有效的优化方法与高级微语言结合起来,为对高级微语言产生有效的目标微码开辟了道路,但SIMPL所采用的优化方法却远非是理想的,在CDC 6400上编译一小段SIMPL微程序大约需7秒钟,而且SIMPL语言基本上是按CDC6400的特点设计的;因此把SIMPL微程序移植到别的主机上去也是困难的。所以迄今为止真正称得上通用的能产生高效率的微代码,并接近自然语言的实用高级微语言仍处在研究实验阶段。

由于 LSI 和 VLSI 控制存储器的迅速发展,微程序设计技术已在从微型到巨型的整个计算机谱系上起着举足轻重的作用,随着时间的推移和计算技术的发展,作为开发系统低级平行度,提高性能/价格比的重要手段,微程序设计早已突破计算机控制器设计的领域而在高级语言的直接执行、功能迁移、信号处理、图象系统以及诊断容错等日益广泛的范围内获到越来越多的应用。

## 1.2 微程序设计技术的评价

微程序设计技术之所以能够得到迅速发展和广泛应用,是因为这门技术具有很多优越性。这门技术的主要优点如下:

### 1. 经济性

经济性是衡量计算机系统的主要尺度之一。微程序设计的计算机比常规设计的计算机更经济,表现在以下几方面:

#### 1) 有更好的性能/价格比

计算机的价格随着控制逻辑复杂性的增加而增加。由于计算机往往通过增加系统复杂性的方式来提高性能,所以又可以说,计算机的价格随着性能的提高而提高。人们希望:大幅度提高性能的结果,最好不导致价格也大幅度地提高。但是,在常规设计的计算机中,若大幅度提高性能,则几乎必然会导致价格也大幅度地提高,而在微程序设计的计算机中,这种提高却是小幅度的。早在六十年代中后期,哈森等人就已比较具体地论证了这个问题(见图 1-1)。

我们还可以用下列表达式对两种控制方式的控制部件的价格进行比较分析。

设:  $C_m$  为每个元件(Component)的平均制造价格

$C_d$  为每个元件的平均设计价格(指由它所组成的计算

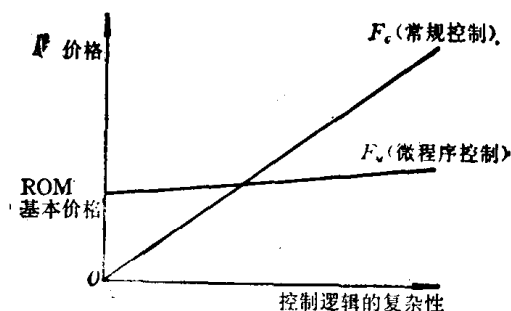


图 1-1 控制部件(常规控制与微程序控制)的价格比较

机硬件设计价格)

$C_w$  为控制存储器中每个微字价格(随微字长度不同而不同)

$C_o$  为每条微指令的平均设计价格

$F_c$  为常规控制部件的总价格

$F_M$  为微程序控制部件的总价格

$n_h$  为常规控制部件的元件数

$n_1$  为微程序控制部件的元件数

$n_i$  为微指令总数

显然有下列关系式:

$$F_c = (C_m + C_d)n_h$$

$$F_M = (C_m + C_d)n_1 + (C_w + C_o)n_i$$

由于  $n_1 \ll n_h$ , 再加上 VLSI 的  $C_w$  值急剧下降, 所以一般总成立下列不等式(而且控制器越复杂, 则  $F_M$  越小于  $F_c$ ):

$$F_M < F_c$$

必须着重指出, 在第三代发展阶段诞生的系列机往往兼顾多用途, 所以指令系统十分庞大和复杂, 从而使控制逻辑也十分复杂。在采用微程序设计后, 系列机就能在价格幅度增加不大的