Burroughs

计算机系统结构

B 5700 / B 6700 系列

埃利奥特·艾·奥盖尼克 著

Computer System Organization

The B5700/B6700 Series

ELLIOTT I. ORGANICK

ACADEMIC PRESS

计算机系统结构
B5700/B6700系列

埃利奥特·艾·奥盖尼克 著

中国外贸咨询与技术
服务公司译

前　言

本书是美国著名电子计算机专家Dr.Elliott Organick教授所著。深入浅出地以Burroughs B5700/B6700型计算机为题材论述了应用堆栈原理的先进计算机结构。Burroughs 的B5700/B6700型计算机现在已发展及改进成新一代的B5900/B6900型。为了帮助中国的广大计算机工作者能了解堆栈结构的优越性，Burroughs 公司特别委托中国外贸咨询与技术服务公司翻译、印刷、並分送本书予各有关单位。希望能和中国的计算机工作者共享这个成就。

美国宝来计算机公司
中　国　办　事　处
北京友谊宾馆7222号

目　　录

第一章 概 述

过去十年中，Burroughs（宝来）公司已经设计和生产了不少计算机系统，这些系统的组织方式和硬件／软件的设计是大大地与众不同和大胆的。这些宝来系统表现了一种与用于程序执行的、有力的语义模型相一致的组织方式，这种组织方式对算法的执行来讲，反映自然的控制结构（控制流程和寻址关系），因此这种组织方式促进算法的执行。

由于Algol 60[44]诞生的推动，过去十年，在了解和开发用分程序结构的程序设计语言来表达包括大的和小的软件系统[18, 21, 27]在内的、复杂算法的潜力方面，取得了很大的进展。已经学习了许多有关算法，以及一些从人—程序接口的观点来看是表示这些算法的最优方式的方法。除了APL和有关语言方面的工作外，在程序设计语言的语法和语义设计方面，几乎所有近来的工作，都是基于Algol 60，以这算法的（静态的）分程序结构为前提，即，嵌套的说明对复杂算法的表达是自然的（假如不是必要的）形式。（PL／1[34], Euler[61], Algol68[80], Gedanken[51], PAL[62], …，宝来扩充Algol[12]）。除了孤立的几个计算机的设计（KDF9[22], Rice计2[52]，和SYMBOL（15, 20,53,55））以外，宝来之外几乎没有计算机系统，也严肃认真地为执行用这种语言*表达的算法提供必要的条件。

*语言APL[36]和紧密有关的诸如LCC（会话计算语言），呈现动态的而不是静态的分程序结构，因此在本书考虑范围之外。几个研究组报告，他们在努力建立为执行用这种语言表达的算法提供必要条件的计算机系统[1,7,31,57]，这些是使人感兴趣的。

足够有意义的是，上述宝来系统是首先提供有效的多道程序设计和多处理的系统之一，这些功能是所有现代大的和最小的计算机系统的目标。

宝来公司的设计者们认为，他们的操作系统的算法本来就是有分程序结构的，并选用一个能反映此结构的语言（Algol 60的扩充）来编这些算法的程序。而对一些其它的设计者们，这种选择看来可能是把不必要的限制强加在系统的研制所得的成果上，系统的研制是用吞吐量和灵活性（以及有关顾客的满足）来测量的。有人提出，对设计的“限制”实际上可能是设计的机会[17]。

以作者的观点，B5700／B6700进展中软件／硬件的研制，已经比我们在算法方法观点完善化方面的自然增长提前进行（或至少跟上），算法观点的增强是宝来公司和其它公司的成功努力的直接结果。他们对非常大的程序，特别是操作系统和各种信息实用程序进行设计，然后进一步理解，然后再设计，等等。

在二十世纪六十年代初，一般（在大学）讲授一个算法是一系列过程步骤，当用较高级的、类似Algol的语言编程序时，它被构造成为一组嵌套的分程序，这些分程序定义算法的标识符的作用域，和在计算机上执行时的动态资源要求。在七十年代，由于计算机科学的教育工作者和研究人员，以及操作系统和信息实用程序设计者们的努力[2,4,6,8,13,14,16,24—27,30,32,37,39,46—48,56,58]，我们一般可能是这样来教程序设计的：强调算法是独立的（常常是嵌套的）结构，这些结构通常是一些异步的任务，而这任务正是模仿了六十年代算法观点的本质。一个独立任务的程序展示这样一种结构，它在每个方面类似一个Algol（或PL／1）过程，但也可能有附加的产生、破坏其它任务或与其他任务同步

的步骤。

看来宝来B5700／B6700“系列”已经把这种算法结构的有效执行，作为一个主要的设计目标。对我们教育工作者和科学家来说，必须查明这个设计目标的重要性，以及宝来公司已经得到成功的程度，这不但使它的操作系统获益，而且通过简单的推论可看出，也使任何子系统级的系统用户获益。本书主要的目的是通过非正式的解释，希望揭示B6700（並往回参引到B5700）的崭新远景，提供它的一般设计和设计原理，以及它作为一个计算机系统的相应潜力。B6700在某种意义上表达出在B5700中对应概念的最好显示（即，一个改进的代表）。在其它方面，诸如存储器大小和原始速度，B5700只不过是B6700的一种更受限制的型式。由于这个原因，在这个相对地短的论述中，没有明显地讨论B5700及它与B6700的比较。然而，通过适当的增强求知欲，希望激励读者进入积极询问的状态，追求进一步研究B6700，它的前身B5700，和它的可能的后继者[2,3,8,10,12,13,14,16,21,32,47,48]。

B6700的硬件／软件的体系结构，从许多不同的观点来看都是使人感兴趣的——这完全不是指它的输入输出硬件子系统[48]和它在静态的（存储器）模块和活动的（处理器）部件之间的信息流程的连结（输入输出门闩矩阵）。为了再次简洁起见，不专门讨论这些体系结构方面，我们假定在此后的进行中，当读者还正在对B6700的中央控制结构、它的原理和情况获得一个清楚的了解和鉴别时，仅对这些体系结构方面的特征作些间接说明。

早在1964年，B5700的操作系统已是一个操作在仅有32000个字的存储器和一个或两个处理器上的生产性的多道程序设计和多

处理系统。从有重大不同的体系结构设计开始，其它公司经过了艰苦跋涉才达到或超过宝来公司的成就，有时在系统上有更大的速度〔18,42,43,46,56,〕。在此期间，已经研究了许多关于管理多道程序设计和多处理的技术和基本原理。其中重要的、或许是这些原理的最好的应用，是基于程序局部性的性质和由此得到的工作区的记法[28]。

表现出引用的高度局部性的程序是这样一种程序，它“偏爱”它的地址空间的一个相对地较小的子集，即，在任何给定的虚拟时间段里，执行一个程序像是在程序没的一个比较小的子集内引用。这个子集叫做工作区。〔局部性和工作区的更形式的定义已在〔23〕中给出，但是对于我们的目的而言，我们满足于上面非正式的表达〕。Denning[28]已表明，当工作区（和它们需要的空间）减少时，可以期望增加能有效地在一个固定大小的存储区内进行多道程序设计的、即没有过分翻来复去的独立程序的数目。经验已经表明〔46,59〕在执行时如果活动过程的工作区能被（连续地）监控和维持在磁心存储器里，那么多道程序设计就得到改进。关于B5700／B6700计算的工作区，有几点使人感兴趣的值得考虑（并且或许值得思索）。

1. 它们展示强的局部性。当任务是活动的时候，因为过程和计算的信息结构的数据成份是由程序员标识的逻辑上的子部分，因此，这种计算的工作区相对地容易维持在存储器里。
2. 工作区的有效的大小，趋向于接近理论上的大小〔即，为包括工作区所需要的实际磁心存储器的容量，趋向于接近（理论上的）需要的磁心存储器的最小容量〕。这是因为：

(a) 当计算是活动的时候，计算的局部数据结构需要生长和收缩时，常常保持在磁心存储器内；

(b) 当计算是活动的时候，仅仅算法的当前正由处理器访问的那些程序段，必须保持在存储器内；

(c) 当计算是活动的时候，除某些由所有活动的计算共享的系统例行程序和表格外，仅有的、必须保持在存储器里的其它部分是字典。它的各项分别指向所有计算的各个程序段。主要是由于这个字典的出现，才使得有效的工作区超过（理论上的）需要的磁心存储器的最小容量，不过这个字典对小程序通常是小的。

3. 要求做新项（段）的插入，或者代替其它不再在工作区者的处理代价，看来是接近最低可能的值。〔通过“最低可能的”，我们就只涉及逻辑寻址的步数，而不必涉及（物理的）处理速度〕。这是因为，当非当前的程序或数据对象必须从二级存储器里得到时，为了达到想要的目标地址，由硬件存取的所谓“描述符”包含（直接地）辅助存储器里目标的地址，而不是像为了一个想要的地址而要检索一个表的地址那样，涉及地址的地址。

如果作者在推测B6700／B5700的硬件结构方面是正确的，并且选择的存储器表示在工作区的维护方面提供这些有吸引力的性质，那么下列两点是小小的奇迹。其一是在这些系统中有效的资源利用是显著的；其二是相对于无结构的、可比较速度（和可比较输入输出功能，等等）的竞争系统来讲是有利的。换句话说，贯彻这样的设想是重要的，即基于其执行算法和记录，是思考和表示为嵌套结构的计算过程的关系（语义）模型的体系结构设

计，导致计算机系统具有更有效的资源利用。如果情况是与计算机的使用相联系的有关因素，诸如程序设计（和再程序设计）以及程序信息共享的代价、保护等等，在权衡时不仅不损害上述多道程序设计的好处，而且事实上这种体系结构设计大有好处，那么这种设想就更引起兴趣。

希望读者将让他的自然的好奇心驱使他读完这本专著的其余部分，然后去进一步研究 B5700／B6700，使得他能得出自己的结论。

这本专著的其余部分的纲要如下：我们给B6700控制结构一个有戏剧效果的（由顶向下逐段展开）讲解，以表明算法怎样被表示和执行。为此目的，先展示一系列情况，然后讨论。每个情况包括给算法在它的执行中的几个（感兴趣的）点的瞬态图。用近来被 Johnston[37]建议的暴露概念的模型（叫做轮廓线模型）来讨论瞬态图，它帮助我们把注意力集中在算法的结构（不变代码部分），算法执行的当前“记录”或状态的结构，处理机的状态，和它们之间的相互关系上。设计这一系列情况和讨论，是为了告诉你关于B6700的控制结构作为算法执行的一般模型的实现，使你有一个渐进的（虽然不完全）了解。

在进行时，我们将讨论几点有关这个硬件和支持的软件体系结构的细节，指出一些它的有关优点和限制。虽然没有给出与其他系统的明确的比较，希望研究完来的概念和虚拟存储器的结果形式——分段的实现，以及对某种类型的控制共享信息的结论的读者，然后将会发现，他们较容易自己去做与其它系统（例如 Multics [18,46]）中对应概念的有意义的比较。同样的意见能适用于与其他系统[4,30,42,45,46]的任务分配和程序间通信功能作

作比较。

上面说的情况研究被包括在第2至6章。第7章考虑B6700的一个潜在的限制，它是由以前各章所引起的一个讨论，而第8章考虑其他实在的和推断的限制，以及去掉这些限制的系统修改。第9章为已经发现他们自己准备好要较深研究者提供一个B6700硬件的更完全的（和更详细的）看法。

读者可能已觉察到，这个长的引言是处在没有提到“栈（Stack）”字的危险中。在这里，我们通过看到B5700／B6700的硬件栈结构已经是为达到硬件支持有效的和控制的算法执行的关键实现装置这一点，来补救这个情况。一点不知道存在于B5700／B6700里的栈机构的本书读者可能会有困难。然而，我们的许多读者可能不完全认识宝来公司已经能怎样有效地使用栈硬件，来反映一个算法的静态结构和它的执行记录，以及为有效执行、为递归的操作控制、和为最小的资源消耗、而建立使用这个存储表示支持的硬件。

第二章　分程序结构型进程和B6700作业

先从了解B6700作业的概念来开始我们的讨论，这个概念包括不随时间变化的算法*和随时间变化的数据结构两部分，而后者是那个算法执行的记录（见图2·1）。这种算法由一组可直接选址的（在虚拟存储的意义上）不变程序段构成的。

执行的记录是一个多用途的数据结构，它在任一给定的时间

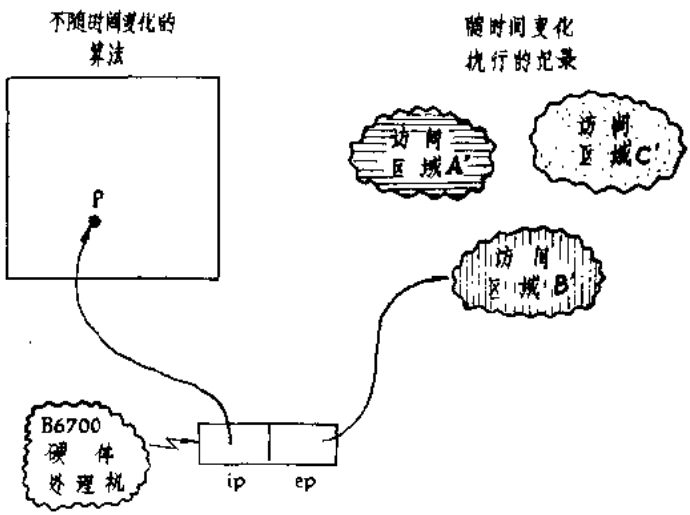

图2·1　B6700作业执行中的瞬态图

*从更高的作业观点，在程序执行过程中，新的程序成分可以加到该算法上，但是没有成分会随时间(内部地)变化。

定义。

(a) 作业的执行状态，包括全部变量的值（标量、数组型以及结构型）；

(b) 为这个作业服务的硬件处理机可能访问的选址环境（虚拟地址子空间），或者可能的几个（重叠的）选址环境，在适当的情况下，允许多于一个处理机同时执行该作业（多重活动）；

(c) 分程序间／过程间／任务间控制流程的历史（例如，调用链）。

最简单的形式下，硬件处理机用指针寄存器，一个指令指针寄存器ip和一个环境指针寄存器ep，组成一对指针寄存器，来完成访问记录可访问部分的功能。

示意图2·1表示，处理机的指令指针寄存器正要执行指令P，而这个时候环境指针寄存器指向记录的B′区域，当P执行时访问B′区域中的数据。图2·1暗示在以后的某时刻及／或前面某时刻ep可能指向记录的其他任何访问区域。

现在，B6700的全部程序设计都是用各种语言来写的，在分程序结构意义*下来编译的，明显的例子如Algol和PL／1。由于这个原因，图2·1示意地表示散开的访问区域的概念在一个重要意义上是牵强的。更真实的示意图在图2·2中给出，它表示B6700处理机的访问环境是一个统一型的访问区域，比如叫C′，B′和A′。这些区域的嵌套反映各分程序的嵌套，每个分程序定义各程序标识符的作用域（即有效域）。

*，象FORTRAN那样的语言，只能定义不嵌套的程序块。这种语言能看作分程序结构语言的退化的例子。

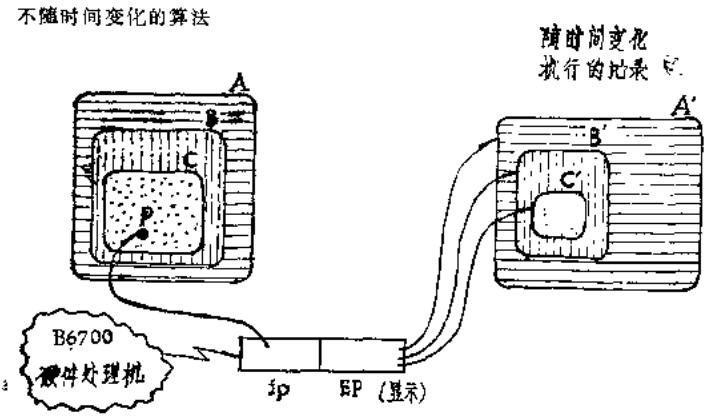

图2·2 B6700作业执行中的瞬态图

例如图2·3表示一个示范用的Algol程序的分程序结构和说明。程序右面的“轮廓线”图解性的表示另一种表达方式，它使说明的标识符的作用域概念变成比较醒目的鲜明形象。图2·2可看作是一个瞬态图，取自当B6700处理机正打算要执行图2·3程序中标号为P的语句的一条指令。这个瞬态图表示环境指针寄存器将指向记录区域C′，B′和A′。环境指针寄存器现在表示为一个显示EP，它是一个指针寄存器的向量（显示概念作为一个瞬时的概念处理）。这些访问区域自然对应于程序的相应的程序段（分程序），其相应的作用域（粗轮廓线）称为C，B和A。[采用条标号方法（例如A′和A）把记录轮廓线和它相应的分程序联系起来]。

与标识符c和d相应的单元，包含在区域C′之内；b相应的单元包含在区域B′之内；a相应的单元包含在区域A′之内。用名字可对这些单元有效地进行访问。

从概念方面讲，当执行标号为P的语句时，我们可以认为引

```
Line
No
  0   begin
  1     integer a, b;
            ) the real procedure, rnd
 10     begin
 11       integer b, c;

 25       begin
 26         integer c, d;

 29     P:  a ← a × b + c × rnd(d);

 34       end
 41     end

 52     begin
 53       integer b, c;

          Q:  ~~~~~

 64     end

 72     begin integer c, d;

 91     end

104   end
```

Algol text (文件)

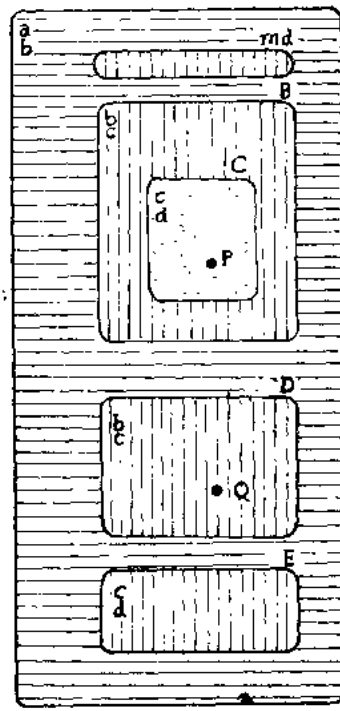

图2·3 Algol程序的分程序结构和说明及其相应的轮廓线结构

用a就是寻找分配给a的单元。这样一种查找可以用下面的办法来完成，顺C′，B′和A′记录区域向外扫描，碰到第一个名字叫a的单元即停止。同时地，当扫描B′时，完成b的查找。请注意，为什么在执行P的指令时，在A′中的名字为b的单元，因此对处理机来说，是完全“看不见的”（因而不可访问）。图2·4是图