ICL—2900系列介绍

（下册）

上海市计算技术研究所

ICL-2900系列介绍

（下册）

上海市计算技术研究所

目　录

（下　册）

第八章
语言和装入程序 ---------- 1
附录 8A
术语小词典 ---------- 39
第九章
外部质备工作介绍 ---------- 45
第十章
管理程序 K_1（第一部份）：事件管理 ---------- 89
第十一章
管理程序 K_1（第二部分）：总　论 ---------- 111
第十二章
管理程序 K_1（第三部分）：设　备 ---------- 147

第八章　　语言和装入程序

这章的目的是要介绍像 Sa，STAPLE，ALICE，ILF 等等这样的术语，建立它们彼此间的关系，并且要介绍用某些源语言书写的程序如何变成在新系列硬件的虚机器上运行的"进程"。

* 8.1　图 8.1 说明编译和装入的步骤。程序被装入虚机器，

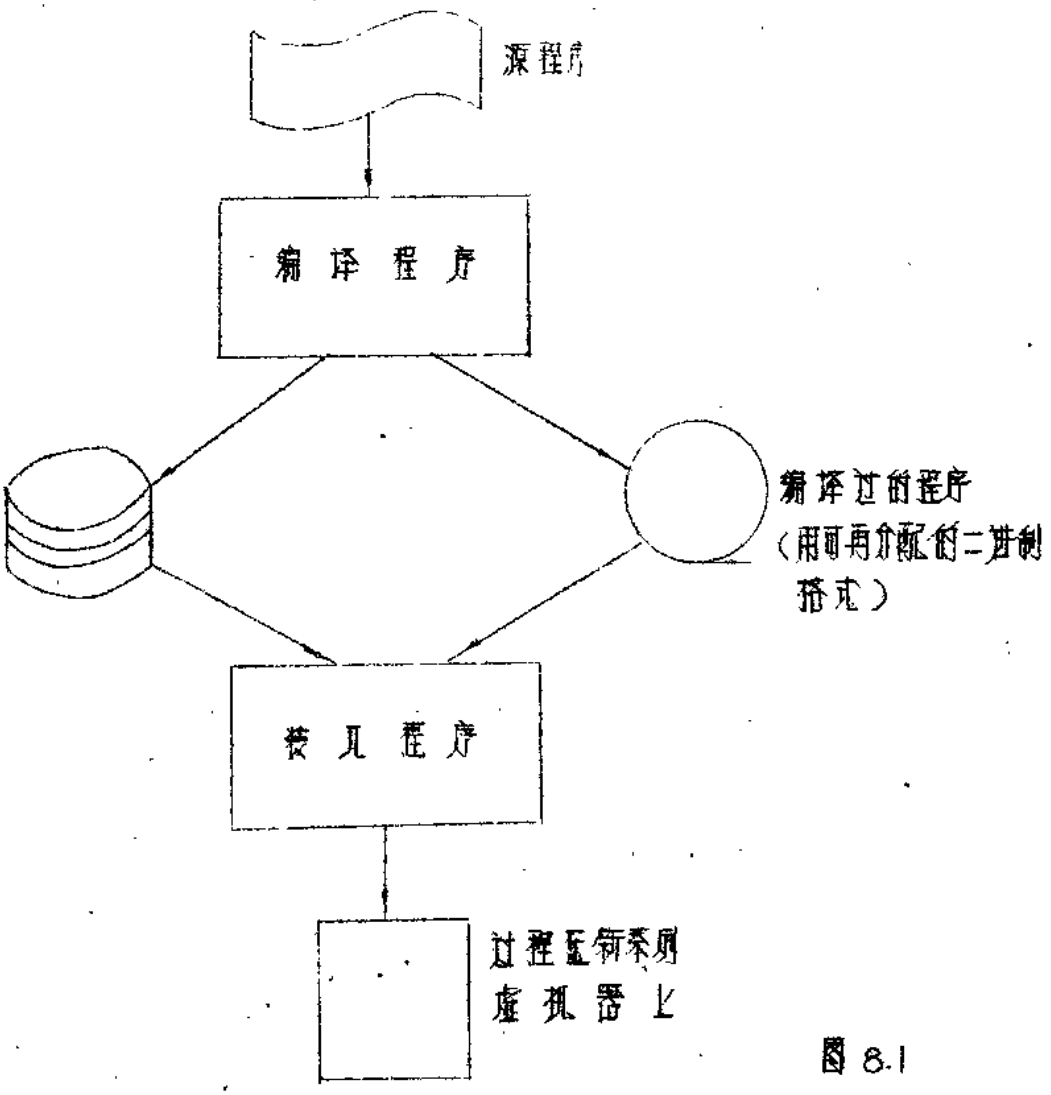


图 8.1

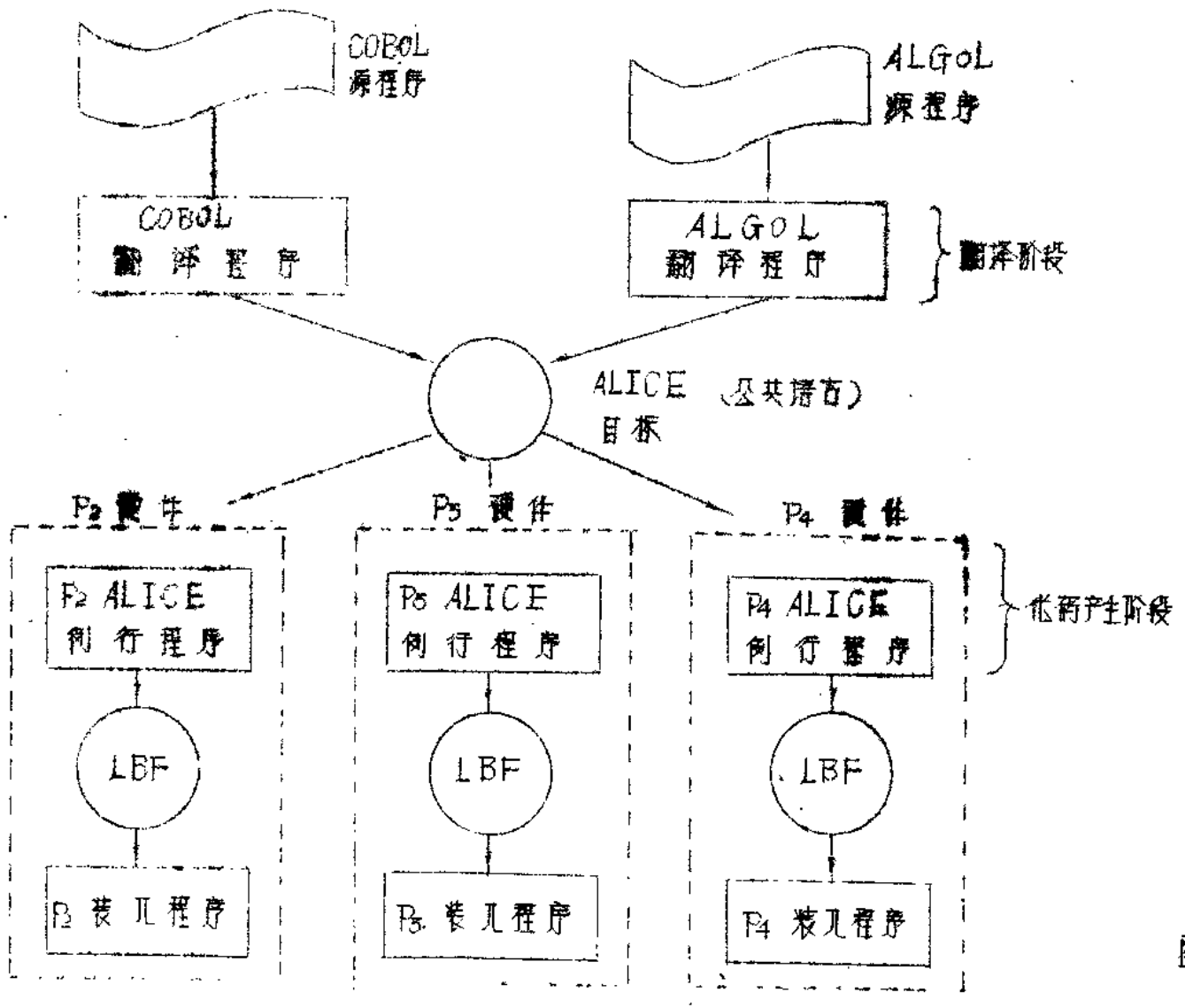

COBOL
源程序
ALGOL
源程序
COBOL
翻译程序
ALGOL
翻译程序
翻译阶段
ALICE (公共语言)
目标
P2 装配
P3 装配
P4 装配
P2 ALICE
例行程序
P3 ALICE
例行程序
P4 ALICE
例行程序
LBF
LBF
LBF
P2 装入程序
P3 装入程序
P4 装入程序
依存产生阶段

图 8.2

ALICE系统的优点

1.发展编译程序较易。

(a)需要的编译程序较少。如果有m种源语言及n种机器，那末，我们需要m个翻译器和n组ALICE的执行程序。否则，我们将需要m×n个编译程序。

(b)翻译程序较简单：它们全产生同样的高水平输出，因此排错较易。

2.为硬件的改进提供更广泛的范围。

ALICE能"隐含"某些初级的界面特征，设计工程师能改变这些特征，采取新技术发展的优点，并且仅需改变ALICE的执行程序而不是整个编译程序。

3.为软件版本提供了便利的形式。

新系列软件能以这个形式而被发布，它是系列无关的，因此可在任何新系列机器上运行。

图 8.3

格式发表本系列任何机器的软件，然后结合机器的一组ALICE执行程序把它转换为可装入格式。

我们曾想在新系列的编译程序上实现这些理想，但是它遇到了大量的困难，且到最后，整个理想才被置于实现。目前，情况被复杂化了，从今以后，它会变得简单些和清楚些。

还有二个问题——ALICE的定义和ICL以前的软件的转写。当然，这二个问题是彼此相关的。

ALIC 的定义

首先ALIC 的水平曾是一个问题。如果它太一般化，太高

级，那末编译过的程序会是低效的。另一方面，如果水平太接近硬件界面，那我们在硬件方面的灵动余地将太小。此外，如果 ALICE 以本系列的一个特定机器作标准，那末，在其他机器上就会产生低效的危险——请记住，本系列的各种机器相互之间的差别过去比现在还大。

另一个因素是曼彻斯特大学的一系列"会谈"。1970年夏，ICL公司和曼彻斯特大学讨论如何使 ICL 公司的系列机与曼彻斯特大学的 MU5 机兼容更接近——这将满足政府资助的研制经费的要求。这些会谈获得了一些进展（虽然最终被中断了），并且定义了 CTL——编译程序的低级语言或者说通用的目标语言。象图 8.4 所说明的，ICL 软件和曼彻斯特大学软件都编译为 CTL，CTL 能翻译为本系列的或 MU5 的指令，从那以后，两方面有某些分离的趋向，并且如何容易地达到这个概念也是可疑的。今天的 CTL 等价物称作翻译程序目标语言（TRans LatoR Target Language）

ICL 系统软件的书写

新系列设计的一个重要准则是，它应适合于用高级语言来书写软件，并且，如果它适用于用户软件，它也应将适用于 ICL 本身的软件——操作系统，编译程序，检查程序和应用性软件。布劳斯曼用 ALGOL 60 为 B 6500 系列书写他们的系统软件，IBM 公司用 PL1 书写他们的操作系统，这说明用高级语言书写软件是行得通的。

S3

S3 是为此目的而引进的语言。它基于 ALGOL 68*

* 8.5 图 8.5 说明了 S3 的一个实例。它是管理程序 B 的某些的一部分。它的功能是转储段（dump Segment）的内容，转储段是在磁头中断过程时存储的，这功能和事件捕获也都有关。（事件管理系统在第 10 章（TA 52）中说明）。图 8.5 也

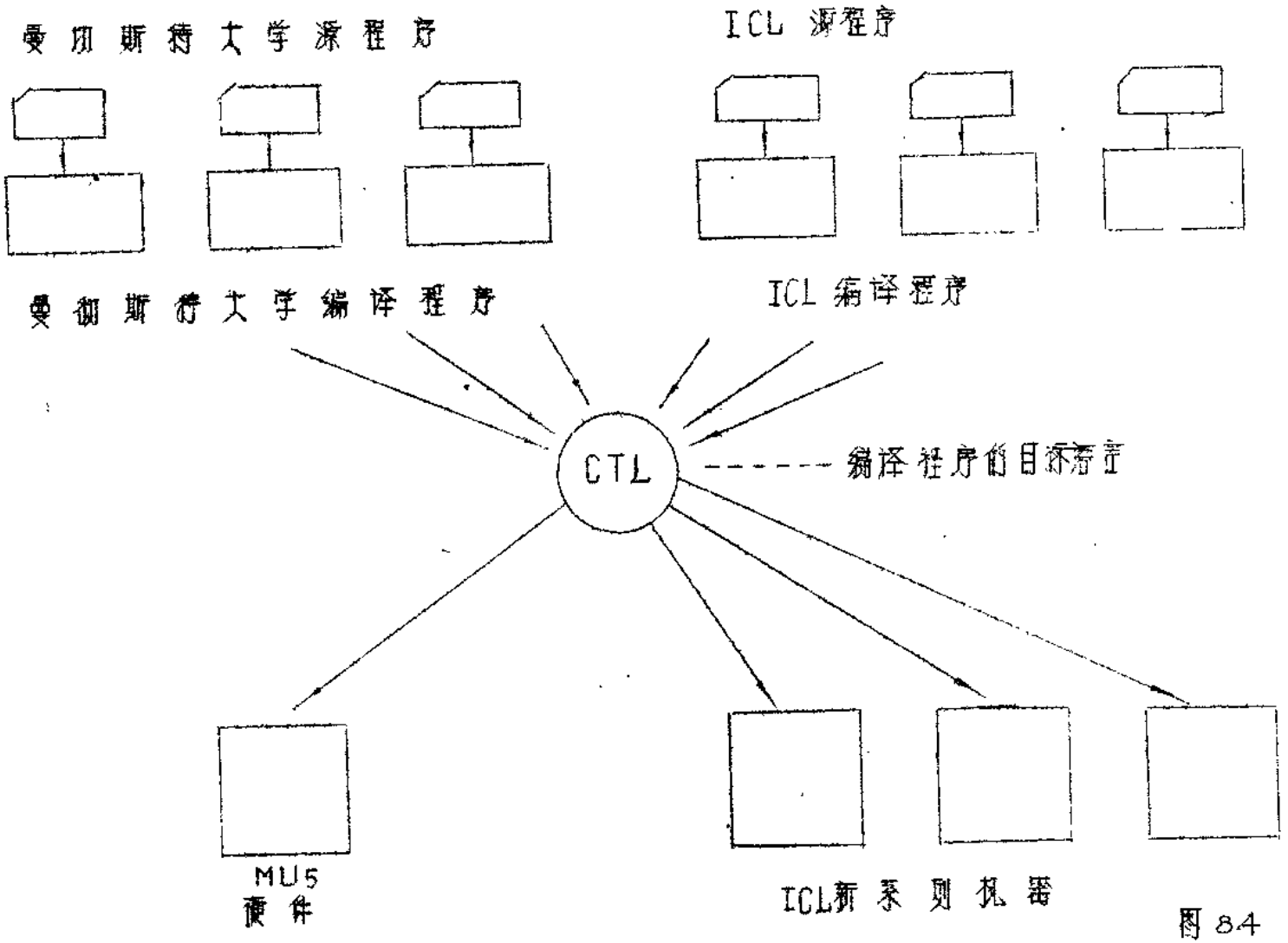

曼彻斯特大学源程序
ICL 源程序
曼彻斯特大学编译程序
ICL 编译程序
CTL
编译程序的目标语言
MU5
硬件
ICL新系列机器

图 8.4

HOLON 名：READ—INTERRUPT_DATA
层：E
MT 文件名：EB 5520 (A)
母体 HOLON 名：KEVM_INTERUPT_DATA_HANDLER
HOLON 注解：*.R

对于 MAL 的过程说明：

```
PROC (REF( ) WORD, REF INT, RESPONSE)
                    READ-INTERRUPT-DATA
@INTERRUPT_DATA, NO_OF_ITEMS, RESULT_CODE@
```

16 AUG 1973

图 8.5 (8)

包含了项目日志（pyogect Log）的副本，它详细说明了 S5 过程做什么事。

也许，你知道 S5，而不知道 S1 和 S2 是怎么回事。S1 是相当低级的语言，加上称为 BETA 的编译程序书写工具，（它是从系统 4 的编译程序书写工具继承过来的）我们有 S1—BETA，其用于书写 S2 的编译程序；S2 被用于书写 S5 的编译程序。实际上，S2 被删掉了。

S5 是高级语言，有良好的数据结构功能。

近来，已加上某些低级的功能。

S5 程序的研制和调试

研制新机器时，重要问题是如何使得软件在硬件调试好和工作之前就被调试好并投入工作。S5 小组的解决办法是研制实

读中断数据的说明

功能：

Action 为阅读装配代码本提供了便利，这个代码本是引进入运行于现行核的最近的中断过程的原因。把存储，如果需要的话，读装配读中断数据的子集，返回的中断数据字的个数，要报告调用者（实际返回的）是要求的个数与中断数据的字数之间的较小者。

如果参数在结构上是无效的或者不中断过程在当前核上运行的话，则把例外条件报告调用者。

参数和返回

类型	存取方式	值	意义
OUTPUT	INTERRUPT_DATA REF()WORD		KEVM 维持的装配代码的最后的有效代码本放在了参数返回（或者根据可以把这个参数复制到其中的代码本（的值））
OUTPUT	NO_OF_ITEMS (REF INT)		把第一个参数返回的中断数据字的个数（即装配代码本）放在了参数是第一个参数的长度和 KEVM 把存的装配代码字的个数中较小者（此数据存放于字 KEVM_SIZE_OF_DUMP_DATA 所说明的）
OUTPUT	RESULT CODE (RESPONSE)	SUCCESS	未遇到故障并且 INTERRUPT_DATA 包含中断数据的有效入口。

ARCH_INSUFFICIENT_PARAMETERS	这次调用中说明的是一个不完全的参数。
ARCH_INACCESSIBLE_PARAMETER	INTERRUPT_DATA或者NO_OF_ITEMS不能具有与回车的返回值。
KEVM_NO_INTERRUPT_DATA	这有中断数据供阅读,因为没有中断过程在当前状态运行。

图 8.5 (9)

操作数据

KEVM_IDT是一个局部表，当要过进入中断过程时，这个表格用于记录被迫转移的指令(9.V.EVENT-PENDING_INTERRUPT_ROUTINE.PLR1.B521)

上面记录在进入最后中断过程时所要的指令的表格的表目是通过 KEVM_CURRENT_NESTED_INTERRUPT_1 来检索的。

代码

halon 首先通过调用 COMMON halon MISSING_PARAMS 核对参数结构的有效性，核实在调用中缺的参数已被说明了，如果这样返回值能通过调用 COMMON halon INVALID_RANDW写到头二个参数中。

然后 holen 通过查对 KEVM_CURRENT_NESTED_INTERRUPTS 为非零来核对是否有中断数据可供阅读。

如果一切均有效，holen 将作为参数的中断数据的总数，INTERRUPT_DATA（中断数据）作为 INTERRUPT_DATA 的长度到任选几中断过程的由 KEVM 记存的被更长字的了数中的数个者，这个数字被写回到 NO_OF_ITEMS，在由 KEVM_CURRENT_NESTED_INTERRUPTS_I 所检索的 KEVM_IDT 中的表目的第一个 NO_OF_ITEMS 字被复制到 INTERRUPT_DATA，并且 holen 退出。

任何有效性故障都作为结果代码送回调用者。

16 AUG 1973

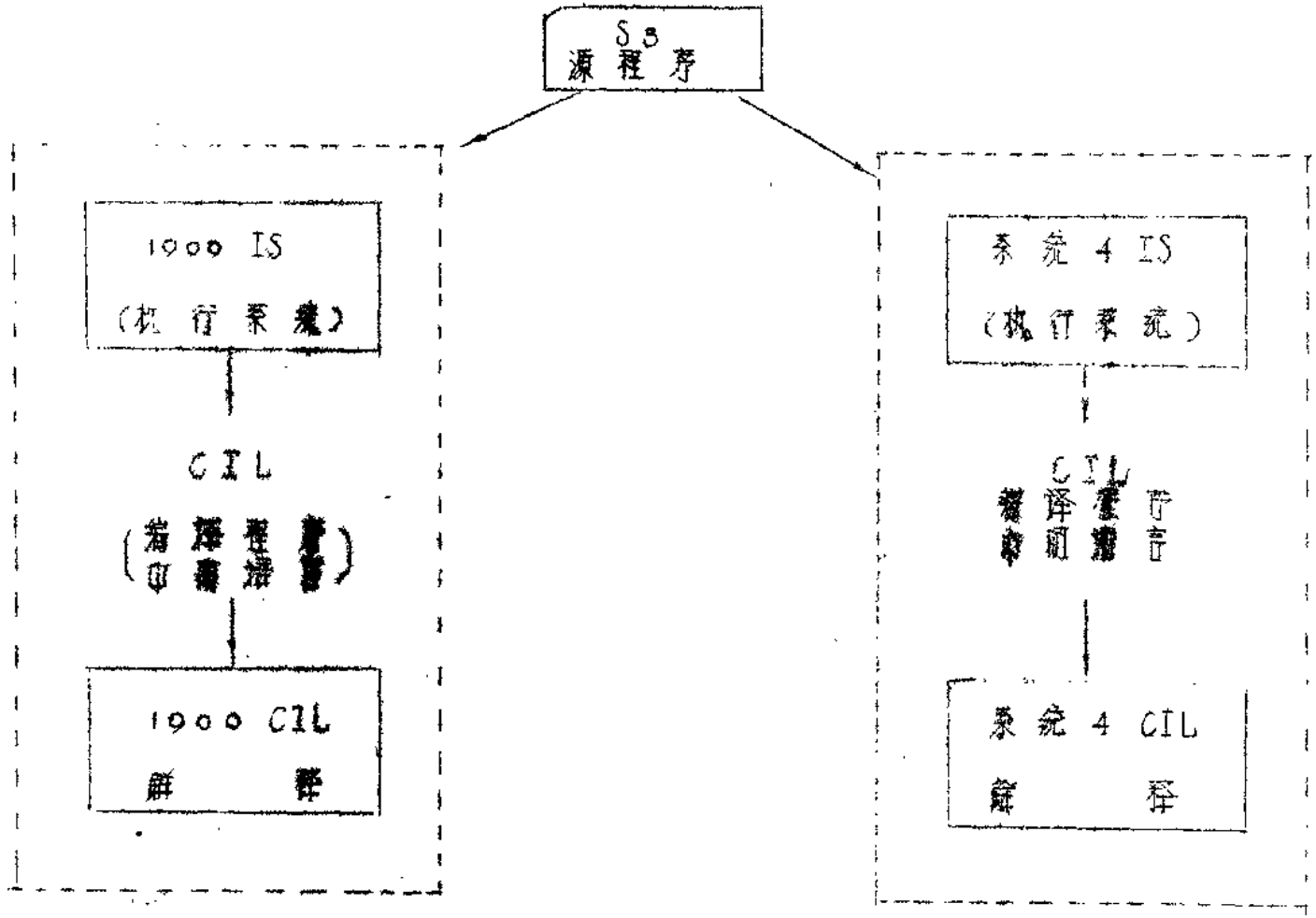
图 8.5 (10)

地系统（Inplemantation System）。

＊8.6 这个方案在图 8.6 中作了说明。

S5程序在 1900机上或系统 4 机上翻译成 CIL（编译程序中间语言的缩写）。然后从 1900或系统 4 的指令进行介释，这个方案本质上是为了使程序有效。

CIL ——编译程序中间语言——之所以在这方案中被使用，是因为 ALICE或者 CTL（编译程序目标语言）尚未足够清楚地


SS
源程序
1900 IS
(机 行 系 统)
CIL
(指 译 程 序)
1900 CIL
解 释
系 统 4 IS
(执 行 系 统)
CIL
(指 译 程 序)
系 统 4 CIL
解 释

早期SS系统

图 8.6

定义以致不能被使用。CIL 被选择为适当的高级语言，通过 ALICE。这就省下了以后从 CIL 翻译为 ALICE 的选择未定。于是，当 ALICE 例行程序可以使用时，CIL 的输出将被转换为可装入的二进制式（编写为 LBF）并被装入硬件。

（顺便指出，如果你发觉缩写——CIL、LBF、CIL——弄糊涂了，那末请参看[illegible] BA 的前言。在许多情形下，当你学懂了他们的全部内容时，无论是缩写或是全名都是非常有意义的。）

STAPLE

虽然 STAPLE 语言目前在 ICL 中广泛使用，但它是为在 West Gorton 使用而设计的。设计和实现 STAPLE 有二个主要理由。

一是 S3 中无低级功能。当时——1969 年下半年——没有什么要加进低级的特点。对于编译程序的编写者和大多数管理程序的要求来说，S3 似乎是很合适的，但对于测试程序，我们必须有低级功能，部分管理程序的接收也需要它们。测试程序的某些部分能用高级语言书写——事实上，是使用高级的 STAPLE。但是据如想象存贮器的寻址，有些器的测试都很难用高级的 S3 来做到，因为无法控制地址。

另一个原因是，在 West Gorton，测试程序的编制者必须使他们的程序在实际硬件上运行，而且比编译程序的编制者和管理程序的编制者早得多。这是因为最早的原型硬件是在 West Gorton 研制的，要[illegible]测试程序被编译好并准备好，一旦硬件组装完毕就装入。在通过运行测试程序证明硬件良好之前，没有人能使用硬件运行他们的程序。

但是 S3 不产生依赖程序编新系列机器的任何东西。这在 S3 小组的优先权表格中是非常低的，而在 West Gorton 表格上却非常高。

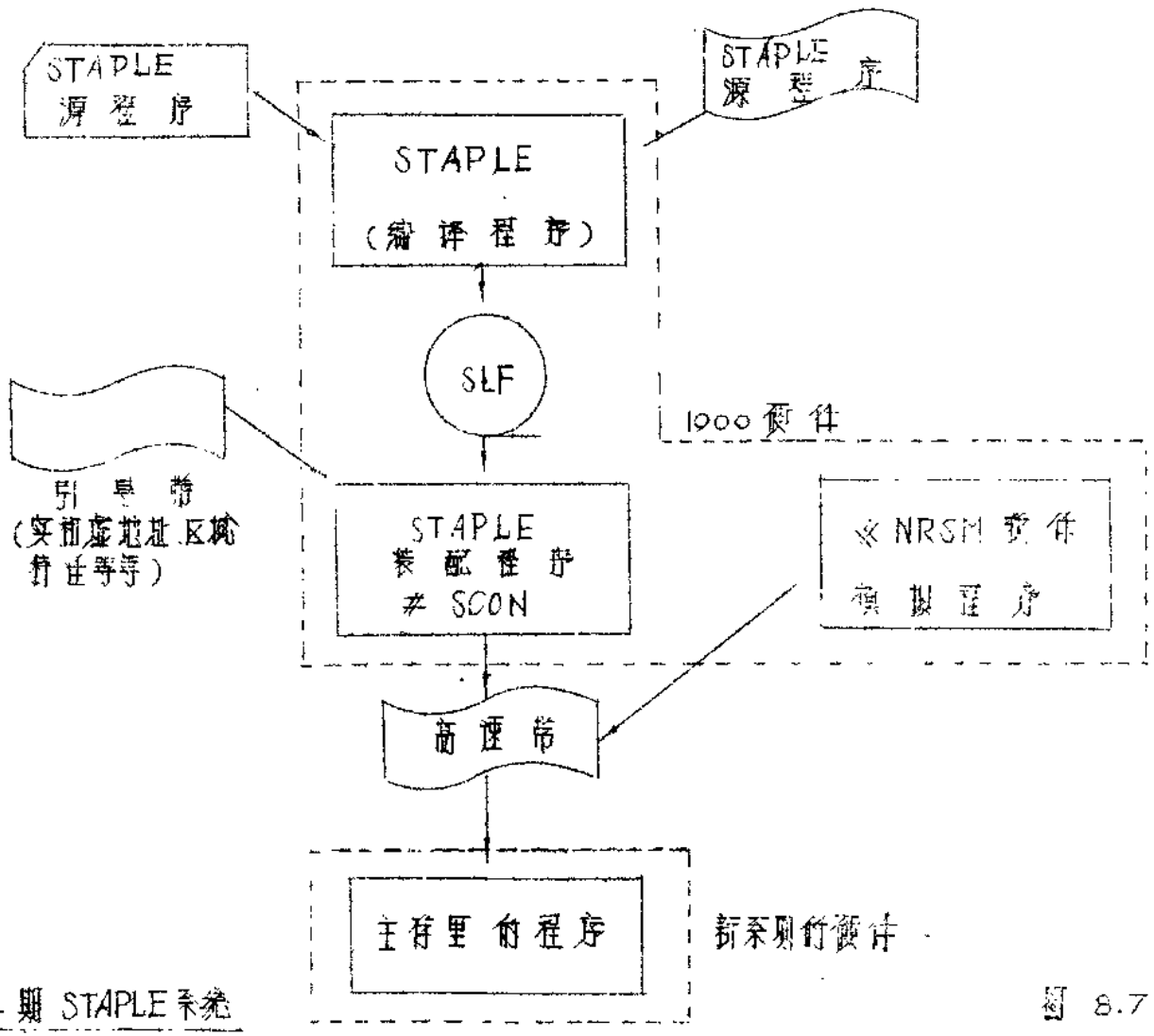
在 p_1 和 p_2 的设计和研制所在地 Stevenage 有一类似的问题，最困硬件的研制比 West Gorton 还都不显得十分严重。Stevenge 的程序员以为，如果 STAPLE 经修改能适合他们的要求的话，就同意使用 STAPLE，而不研制他们自己的语言和编译程序。

※ 8.7 早期的 STAPLE 系统在图 8.7 中说明。STAPLE 编译程序可输出叫作 SLF —— STAPLE 装几格式。在理论上，LBF —— 可装几的二进格式——应更好些，可是可装几的二进格式尚未定义好。跟着输出下就是 ALICE，因为 ALICE 也还没有定义。此外，ALICE 比机器代码稍高一点 —— 而低级的 STAPLE 是机器代码。这样，SLF 被设计为最简单的格式，和 NRSD 2.3.1 的设计思想一致（看图表式）。当它被定义好时，就为以后向下了把 STAPLE 10 装几格式翻译为可装几二进格式的选择余地。

早期的 STAPLE 装配程序（Consolidator），也在 1900 上运行，产生高速带（whoosh tape），用硬件的初始程序装几时能所接受的十六进制格式。

为了测试那些程序的工作，把它们提供给称为 NRSM 的软件模拟程序，这个模拟程序是在 BYackneeLL 设计和编写的，在 1900 硬件上运行，是最有价值的工具。West Goyton 的程序员把它作了修改和扩充，以适合自己的需要。它起二个重要的作用。一是在把这些程序提供给硬件之前就把它们调试好。假如你把一个未调试好的程序提供给一未调试好的硬件，而程序执行不下去，那末到底是硬件错还是软件错呢？软件模拟程序给出程序执行过程中的大量监督信息，这意味着，我们能充分相信产生出一了正确的程序。

软件模拟程序所起的另一作用是作为硬件的代用品，但有不同的侧重点。早期机器时间是非常有限的。一台机器必须由



早期 STAPLE系统

图 8.7