DJS－8计算机

FORTRAN IV 编译系统设计说明

第二分册

优　化

中国科学院高能物理研究所七室
FORTRAN IV编译系统设计组

1981年12月

目　录

§1 引言

1.1 优化的目的和必要性

编译程序目标的优化，是为了使目标能更加有效地执行。这里的效率是指运算时间和占用空间两个方面，即希望在执行时间和占用内存空间上都能尽可能节省，但这往往是矛盾的。一般情况下，主要考虑运算时间的节省，必要时采取折衷方案。

随着数学模型更加复杂和使用计算机的人数更多，其中多数人写程序主要集中精力实现他们的模型，以及程序的易读，而不注意怎样整理表达式而节省计算时间，这样给编译提出了非常重要的优化任务。

1.2 局部优化与全局优化

编译程序优化一般分为全局和局部两大类。

局部优化主要是指依赖于具体机器的优化措施，往往与个别的计算机有关，或者限于各个“局部”之内优化，各“局部”之间不发生任何关系，这个“局部”可能是一个语句，一个分段点之内或一个基本块等。

全局优化，主要是指与机器无关的更加一段地优化算法，它不限于某个“局部”之内，要考虑各个“局部”之间的关系。诸如为进行优化而收集必要的信息方法，相同式的消除理论等。从而以全局角度进行优化。

1.3 优化的功能和举例

优化还能更详细地分类，把优化按项目不同列出如下种类：

1·3·1 常数运算合并

为了节省运行时间，将诸常数项的运算在编译时合并成为一个常数项。

如表达式

```
2·0*R*3·14*H**2/3
```

合并成

```
C*R*H**2
```

其中 C═2·0*3·14/3

C在编译时算好。

1·3·2 公共子表达式节省

有表达式	经过节省后
L 2·0*H*R	h ═ H * R
⋮	L 2·0 * h
	⋮
L 1 H * R * S	L 1 h * S

这里的L，L 1用来标识其后面的表达式。

能够这样做是有条件的。

(1) 要求L和L 1之间的任何可能执行的路径上H和R没有再定义。

(2) L必须在L 1的必经路径上，即从段入口到L 1必须经过L，否则节省就不能充分进行。

1·3·3 循环不变量运算外移

所谓不变量运算就是相对于循环内其值不变的算术表达式。

例：


```
      K═S
1     R═D*K
      X═A*B
      ⋮
      G═R/X
      ⋮
      K═K+1
      IF(K·LS·SI)GθTθ1
```

在例句中构成一个循环其中A * B运算与循环无关，即A、B在循环中没有再定义，因此可以将A * B外提成为：

```
      K═S
      X═A*B
1     R═D*K
      ⋮
      G═R/X
      ⋮
      K═K+1
      IF(K·LS·SI)GθTθ1
```

必须指出，这种外移也要求处于循环的必经路径上，否则是危险的。

例如

```
      Dθ  1  I═1，N
          ⋮
      IF(X·EQ·0·)GθTθ2
          y═1·/X
```

```
      2 …
      1   CθNTINUE
```

上例中即使X在循环内没有再定义，1。／X也不能外移，因它处于循环中可能不经过的路径上（当X＝0时），如果移出将导致溢出（1。0／0。）。

1。3。4　强度消减

这里的“强度”是指运算强度，比如运算＊＊强度大于＊、／，而后者又大于十、一，如果我们能够将运算强度消减，即由高的变为低的，当然是很好的优化。

比如运算X＊＊2、X＊＊3、X＊＊4等等可以变为X＊X，X＊X＊X，（X＊X）＊（X＊X）显然使运算时间大大节省，因为幂运算一般地要引用相应函数过程。这里讲的强度削减主要指与循环控制变量有关的运算强度削减，举例说明如下：

```
例    DO   1   I＝2，N
                ⋮
           K＝4＊I＋5
                ⋮
      1   CθNTINUN
```

如果变量K在循环中没有其它再定义，经削减后变成

```
           K＝13
     DO    1   I＝1，N
                ⋮
           K＝K＋4
                ⋮
      1   CθNTINUE
```


循环中所有引用K的地方不变。这样使循环内用一次加法（K＋4）代替原来的一次乘法和加法（4＊I＋5），在循环外多了一次传送（K＝13），在编译时计算4＊2＋5。这显然是很大的节省，这种削减特别地对于隐含在下标表达式中的运算更为有效。

在上面例子中的削减运算的类型是整型的，对于实型运算或其它类型，考虑用循环中逐次累加办法代替乘法运算，可能造成很大的含入误差，因而不采用这种优化。

1。3。5　消去简单存贮

下面的赋值语句

```
y＝C
```

其中C是绝对常数或相对常数（相对于循环）称之为简单存贮，如果y在此处定义后。在允许的范围内用C值代替y的引用，必然增加优化的可能性，而且当前y值不再被引用的情况下（即所谓不再是“忙”的），这个赋值语句可以节省。

1。3。6　数组元素地址计算优化

在数组元素地址计算中在可能情况下使用变址，会优化的更好。例如有循环

```
           I＝1
      1    ⋮
           M＝A＊B（2＊I）
                ⋮
           I＝I＋3
           IF（I。LE。N）GθTθ1
```

在运行时，必须计算数组元素的下标表达式及其地址值，上

面的数组元素B（2*I）在计算其地址时需要计算

$$B_Z(2*I) = 2*I+W_B$$

其中B_Z表示数组元素B（2*I）的地址，W_B表示数组B的头地址减1（对一般维数来讲是减去一个跨度）。

这个表达式通过外移，削减变成如下形式

```
      I=1
      J=2*I+WB(=2+WB)
1     ⋮
      M=A*>J<(>J<表示运算对象是以J
      ⋮          的内容为地址的对象）
      I=I+3
      J=J+6
      IE(I.LE.N)GθTθ1
```

经过上述优化，循环内数组元素地址计算由一次乘法、一次加法变成了一次加法，循环外是一次传送。

如果上面的数组元素的下标表达式只是一个绝对常数时，其地址值的计算可以在编译时算出，特别地如果循环内再有一个数组元素B_1（2*I+4），B_1（2*I+4）的地址计算可以完全省略而代之使用B_Z加上位移量d，其中

$$d = W_{B1} - W_B + 4$$

这在多维数组元素的情况下，优化效果更为明显。

1·3·7 内部 基本外部函数优化

考虑这些函数在源程序中出现的比较频繁，而且优化后会有比较大的效果。例如：


```
      Dθ  1  I=1, N
          ⋮
      y=SIN(X)*cos(FLθAT(I)*X)
      Z=TAN(3.1416/2.)*cos(FLθAT(I)*X)
          ⋮
1     CθNTINUE
```

其中常数TAN(3.1416/2.)在编译时算好，而cos(FLθAT(I)*X)，可节省为计算一次，SIN(X)可外提到循环外去计算（要求X与循环无关），这样优化后变成下面的形式

```
      SINX=SIN(X)
      Dθ  1  I=1, N
          ⋮
      CθNFLAX=cos(FLθAT(I)*X)
      y=SINX*CθSFLAX
      Z=C*CθSFLAX
          ⋮
1     CθNTINUE
```

其中C=TAN(3.1416/2.)

节省这样的函数运算就等于节省一次过程引用．效果是十分明显的，然而由于上述各种优化，可能产生大量的中间结果需要存贮．为此中间结果（指临时存贮，内部变量）的内存空间分配上必须采取适当的优化措施。

§2 优化设计

2·1 分级优化

由于源程序所表示的数学模型是各种各样的，它们对于计算

时间和所占内存空间要求也是多种多样的，对于那些运算时间短，又是多次重复（仅仅输入数据的改变）的题目，采取高级优化是不合适的，因为它要牺牲编译时间为其代价来换取运行速度。而对于那些循环多计算时间不是十分长的题目，进行循环的局部优化可以缩短运行时间。但对于那些表达式繁长其重复执行，计算时间长的题目，进行全局性的优化编译是极为必要的，它会带来很大的好处，即使同一个程序不同的程序段，也可能具有上述不同的特点，对它们的编译采取分级优化处理就更为合适，因此优化可以给出几种级别以供程序员针对不同的情况加以选择。

本编译系统优化选择三种级别：

θPT＝2　全局优化

θPT＝0　不优化

θPT＝1　中间级优化

中间级优化是为了那些不满足全局优化的程序而设立，它是全局优化功能的进一步缩小。

2·2　结构的确定

2·2·1　程序基本块

程序的基本块就是优化的基本单位，因而称之为基本块。

一个可执行语句的集合{ S_i　i=1，2，…，N. N>0 }称为一个基本块，如果满足下面条件：

(1)　S_j 执行完以后，必须紧接着执行 S_{j+1}，0＜j＜N。

(2)　S_{j+1} 的执行只能是 S_j 执行完毕直接后继 0＜j＜N。

其中 S_1 称为基本块的初始语句或块入口，S_N 称为基本块的终结语句或块出口。

优化首先将源程序分成块，下列语句是优化基本块的终结语

句。

(1)　Dθ语句。

(2)　各类转语句（GθTθ、GθIK、GθKI）。

(3)　算术条件语句。

(4)　停语句（STθP）。

(5)　返回语句（RETURN）。

(6)　在转语句或算术条件语句中引用的标号所标识的语句之前的第一个可执行语句（如果它存在）。

(7)　逻辑条件语句（不计内嵌语句）。

(8)　逻辑条件语句的内嵌语句。

下列语句是优化基本块的开始语句：

(1)　程序段的第一个可执行语句。

(2)　在一个基本块终结语句之后第一个可执行语句。

可见，一个块可能只有一个语句，如逻辑条件语句的内嵌语句，它是一个基本块。

2·2·2　块间信息联络

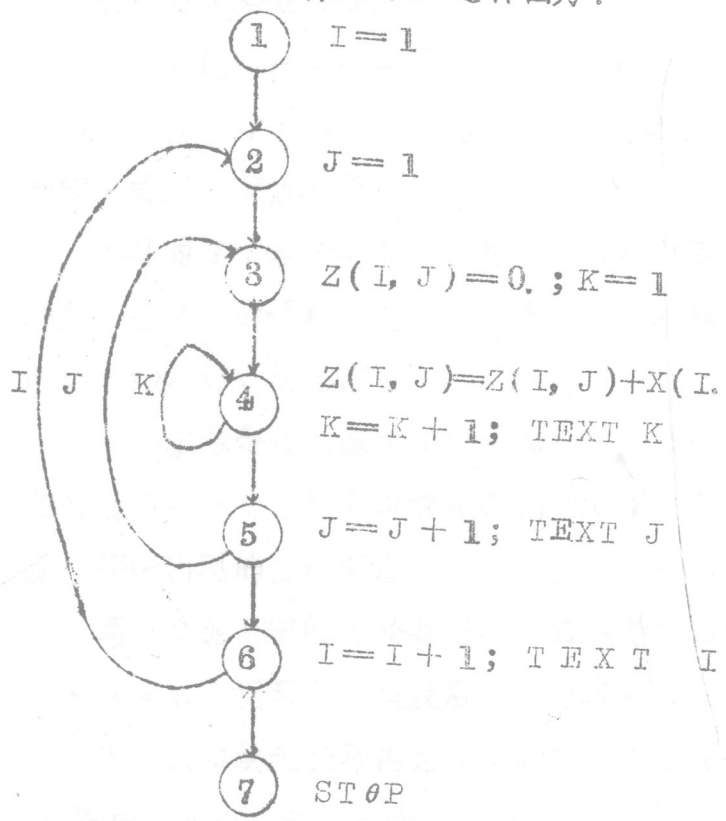
程序块确定之后，就要解决块间通讯的方式，为此必须确定块与块之间的若干关系。首先引进有向图的概念。

我们将程序的基本块作为节点，而块与块之间的关系作为弧，那么一个有向图就是一个有序对 D＝＜A，R＞的集合。这样一个程序段就可以由一个有向图的有限集合来表示，这个有向图将程序段内各块间的若干关系表现的很清晰。如程序段：

```
DIMENSIONX(10 , 10), Y(10, 10), Z(10, 10)
  Dθ3   I＝1, 10
  Dθ3   J＝1, 10
```

```
      Z(I, J)=0
      DθZ  K=1, 10
3     Z(I, J) = Z(I, J)+X(I, K)*Y(K, J)
      STθP
```

上面的程序段的控制流程构成有向图，在平面上用圆圈表示节点，用带箭头的线表示弧，这样图为：



一个基本块a在执行时，可能导至后继块b，那么我们称b块是a块的向前联络．反之a块是b块的向后联络，在新考虑的集合A中，所有向前联络集合作为关系R．那么有序对D=<A，R>是集合A中的向前联络的有向图，反之定义R－1是R的逆，即向后联络的集合，也就是有序对D=<A，R－1>是A集合中的向后联络有向图。所以这种联络是块间信息通讯的脉络，有

了它就可以将诸块间的关系搞的清清楚楚，以达到良好的全局优化之目的。

2·2·3　基本块级次的确定

基本块的级次是反映一个方向图控制流程的诸块的层次关系，如有有序对$<a_0, a_1>$，$<a_1, a_2>$，…，$<a_{n-1}, a_n>$或简写为关系$(a_0, a_1, \cdots, a_n)$，我们称由a_0到a_n为一个路径，其路径长度为N，对由a_0到a_n有向前路径$(a_0 a_1 \cdots a_n)$，那么所有由a_0到a_n的向前路径中其长度最短者定为a_0到a_n的距离。

一个以a_0为入口的流程图D=<A，R>，其任一节点的级次就是由a_0到它的距离。

2·2·4　基本块的向后直接优先块的确定

一个以a_0为入口的流程图D=<A，R>，其任意节点a_i的向后优先节点的集合是指由a_0到a_i的任何路径上所包含的节点（不包括a_i本身）。

节点a的直接优先是其向后优先节点集合中与节点a的距离最短者。如果a_0入口节点的向后直接优先块定义为它自己，那流程图D=<A，R>的任意一节点的向后直接优先存在且唯一（流程图中的节点所包括的语句如果满足FORTRAN规定，这个结论成立）。

2·2·5　循环编号和循环层次

流程图中一个路径$(a_i, a_{i+1}, \cdots, a_n)$，如果$a_i=a_n$，则称之为循环路径。

一个流程图中的循环区域$D_i=<A_i, R_i>$是D的子集，$A_i \subseteq A$是其子集$R_i=<A_i, A_i> \cap R$，A_i定义为$a \in A_i$

包括a的任一循环路径上的节点都属于Ai．且Ai的元素都在a的每个循环路径上。

一个循环区域当入口只有一个节点时，称为一个循环。

循环区域的层次定义如下：D=＜A，R＞的循环层次为0，基于A的循环区域Di＝＜Ai，Ri＞，如果其入口节点唯一，则有层次为1，否则为0。

一般地说，一个循环区域$Di_1, i_2, \cdots i_j$的层次是循环区域$Di_1, i_2, \cdots i_{j-1}$的层次加1．如果$Di_1, i_2, \cdots i_j$有唯一的入口节点，否则层次不增。

循环的层次就是与其等同的循环区域的层次。循环层次是循环的嵌套关系的反映，也是执行循环优化的顺序，即由嵌套最深的层开始，到0层结束。

循环编号是按照源程序段开始定义循环编号为0，然后顺序扫描当出现一个循环入口块就将编号加1，即循环编号是循环在源程序中出现的自然顺序。

同一个循环中的诸块具有相同的编号和相同的层次。

2·2·6　循环入口块及后目标块的确定

循环入口块是循环内诸块中，第一次接受控制的块，并且该块的后继块之一就是其本身的块。

一个循环D=＜A，R_A）它是基于B集合的．集合B的关系表示为R_B，设a_0为循环的入口，若在a_0的向后优先集合中所有只有一个向前联络的块中到a_0的距离最短的块b存在且唯一，$b \in B$，$b \notin A$，有$\langle a_0, b\rangle \in R_B^{-1}$则b是循环A的后目标。

为了我们的目的，如果一个循环没有向后目标，我们可以制

造一个节点，使其成为后目标，以本节的符号为例，设A没有向后目标，则造一个节点$b \in B$，与节点a_0构成有序对＜b，a_0＞$\in R_B$，以及对任意A，有$C \in B-A$，若有＜c，a_0＞$\in R_B$，则以＜C，b＞代之．于是b是循环，D=＜A，R_A＞后目标。

§3　优化的要求

3·1　源程序的中间文本表示

3·1·1　二元式及二元式片

二元式是由运算符θP和运算对象P两部分构成，当P是数组元素的地址时，记为＞P＜。

二元式片是由具有相同运算符优先数的一系列二元式组成。一个运算片表明这个片的初值与其包含的一系列二元式的运算产生的结果，如果运算片的运算符是十，一，运算片的初值为0．如果片的运算符是*、／，**运算片的初值为1。

考虑到关系运算和逻辑运算实现比较简单，本方案没有考虑它们参与优化，实际上加入这部份优化内容对本设计方案没有什么困难。

比如表达式X=A*B*（C－D＋E）*G／F以二元式表示

h_1　＋C
　　－D
　　＋E
h_2　*A
　　*B
　　*h_1

* G

／E

X＝＝h ₂

在片头指示的 h ₁，h ₂分别表示所在运算片的结果。

二元式片应尽可能的大，以便优化能更充分进行，为此规定如下：

一个表达式，如果由多个项组成，那么先生成那些非简单对象的项的二元式片，将其结果代替对应的项，再生成该表达式的运算片，一项的运算片如果由多个因式组成，那么先生成那些非简单对象的因式的二元式片．将其结果代替对应的的因式，再生成该项的运算片。类似同样的叙述去产生因式的运算片，一个初等量如果不是简单对象，则生成它的运算片。

这里所谓简单对象是指变量、常数。

这样做的目的是使运算片之间如有相同运算，便能找到公共子表达式和与循环无关的不变运算外移，例如有两个运算片

h ₁		h ₂	
	* X		* X
	* A		* A
	／y		* Z
	* Z		／C
	* D		

我们可以找到 h ₁和 h ₂两个片的公共部份

h ₃ * X

* A

* Z

另外如果在 h ₁中 A 和 D 与循环无关，那么运算

h ₄ * A

* D

便可以外移，同时，其中的常数运算可充分合并。

3·1·2 内部变量

在由源程序生成中间文本时，一些中间结果不用内部变量标识。

内部变量分为临时的和非临时的，为简单起见分别称为临时变量和内部变量。在优化之前所有中间结果，需要存贮时，都用临时变量标识。优化中产生的中间结果为内部变量。规定，临时变量的存贮分配将在编译生成目标阶段做，而内部变量的存贮分配由优化阶段去做。

3·2 优化的要求及对源程序的限制

3·2·1 优化的要求

编译的优化要求第一阶段 编译做好必要的准备工作，即进行了充分的语法检查，生成必要的中间文本项及字典项，详见附录。这里我们强调一些约定。

(1) 关于公用（包括共名）块和等价字典的约定。

对于公用块的各个元素，要求在公用字典中和外部符号字典中能够查到它们，并且要求一个块的元素能与其它块的元素区别开来。

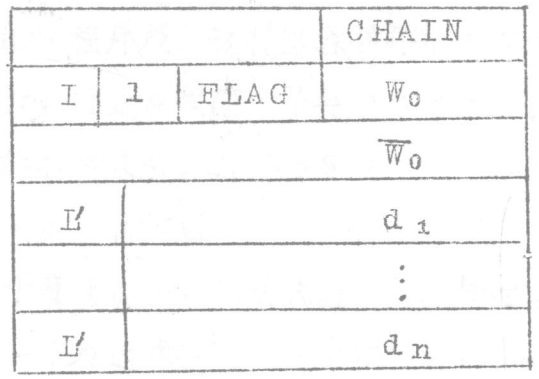
对于等价要求能从等价字典中查出各自独立的等价片。

这项要求之目的，是为了在优化中建立变量，数组有关引用定义字位信息时，能够全面的考虑到由于等价，公用所造成的结合给引用定义所造成的影响。

(2) 关于数组的约定

对于一个数组说明符，产生一个数组信息字，它在编译时表示成为如下形式：

数组说明符A（d_1，d_a，…d_n）产生的数组字是：



$\overline{W}_0$称为数组假头，$\overline{W}_0$称为数组头，1是元素占单元长度特征

$$\overline{W}_0 = W_0 - SP$$

$$SP = 1 \sum_{i=1}^{ni-1} \prod_{j=0} d_j \quad d_0 = 1$$

使用假头是对数组元素地址计算会带来好处。

此外对于维界说明di可在栏中L说明di是常数还是变量，如果是变量，那么它肯定是本程序段中的不变量，不许有定义处。当di参于运算时，不必在优化中考虑它的定义问题。

(3) 关于内部标号的约定：

一个基本优化块的中间文本项是以一个标号文本项开始的，这个标号标识了这个块。对于那些第一个语句没有标号的块，插入内部标号，以此标识这个块，这种情况发生在逻辑条件语句的内嵌语句，Dθ循环语句的终结语句下面的第一个可执行语句（如果存在且没有标号）以及Dθ循环语句下面的第一个可执行

语句（如果它没有标号）。

(4) 关于Dθ循环的约定

举例说明：

```
DθL   I=m1, m2, m3
L1 ·····
      ⋮
L   CθNTINUE
```

内部文本项为

```
I=m1
L2 ·····
      ⋮
L   CθNTINUE
I=I+m3
IF(I·LE·m2)GθTθL1
```

其中L_2可能是内部标号，L标识的语句可能是允许的语句，这个循环及其判断是循环满足 否的中间文本项按等价源程序生成。

这样做的目的是为了在优化中把Dθ句和一般循环统一处理，而Dθ句文本项的目的是因在循环内m_1，m_2，m_3不能再定义，而循环参量I只能在循环终结时再定义，因此它们实际上是循环体内不变量，这对优化中考虑它们的引用时，而不必考虑在循环内是否会有另外的定义。

3·2·2 全局优化对源程序的限制

本优化方案是按段编译，并与最后的装配的结构相适应的，考虑到内部函数及基本外部函数出现的频率高，因此对它们的优

化处理会产生更好的效果。但是在优化时，只是当它们被引用的要求与其内部、基本外部函数表中的规定一致时，就会被认为是那个函数的引用。当然，这时并不知道是否会有一个定义的FθRTRAN程序段，这样当源程序中有这个函数的定义段时，此定义无效，因此源程序中不要用内部函数、基本外部函数名作为函数段的名字来用，这样会导致计算结果的错误。

3·2·3 付作用

由于高级优化，表达式的计算次序与源程序有了很大变化，这样当一个表达式在运行发生溢出时，当时所确定的页、行可能不是固有的源程序中的位置，因此它可能是从循环中外移的运算。

全局优化在编译时要花费较多时间，一个程序在没有经过编译检查和试算，最好不要使用全局优化。

3·3 符号记法说明

(1) 以下面记法标识某个表A(字典项或栈)的第a项的P字段 $P_A(a)$ (A(a)表示A的第a项)

(2) nbit表示字位n，nbit(A)是A的第n个字位

(3) 以add(P)表示P单元(24位)的地址部分(后16位)

其中P，A，a有可能是数字组合或者是同样结构的组合。

§4 控制流程

本优化阶段从编译第一阶段接受数据，并受总控程序FSD控制。当程序员指定的优化级别为θPT=0时，跳过优化阶段由FSD直接调用第三阶段BY3。当程序员指定的优化级别为θPT=1，2时由FSD调用本阶段的总控程序BY2。

本阶段由一个控制程序BY_2和八组程序组成。BY_2控制

本阶段的执行。所有控制路径皆以此控制程序为始终。

第一组程序是优化的准备阶段，由BY_2调用，首先将程序段划分基本块；制造一个入口块以段头开始；跳过语句函数(如果有)；把块表按标号定义出现顺序链接。遇到各种转向(GθTθ；GθIK，GθKI)制造一个新的称号定义文本项及块表项；填写向前、向后联络信息；给段出口块加以标志。这由 BLθCKING子程序来完成。然后给程序块确定级次，由DEF—STAGES子程序完成，第三步建立直接优先块，给循环入口块加标志，DBDB子程序完成。第四步，查找循环，建立循环表LθθPT，给各个循环中的块填写循环编号和层次，由 BLθθP子程序完成。最后建立后目标，当没有后目标块时，建立一个新的后目标块嵌入到方向图中。

第二组程序是给数组和变量分以座标数。由BθN子程序完成。当由BθN子程序返回BY_2后，BY_2将从循环表LθθPT中挑选出循环层数最大者，然后调用优化处理阶段，程序名字为θPTPRθCESS。

优化处理阶段由六组程序组成。这六组程序始终受θPTPR—θCESS程序控制。

第三组程序是根据BY_2所选择的循环的最深层数，建立循环内的块排列顺序表，LBT表。由ALBT子程序完成。

第四组程序是完成常数运算的合并；运算对象的排序；收集引用定义信息；嵌入语句函数；给每个数组元素计算文本项后面嵌入一个内部赋值文本项为分配内存空间及变址做好准备；改造数组元素计算文本项(如果可以时)并建立W区。这组程序是在DEFBIT子程序控制下完成的。

第五组程序是公共子表达式节省，它是按着一个循环内排列于表ＬＢＴ中的次序逐个进行的，首先是块内节省，然后当θPT＝２时进行块外节省。子表达式的算法是采用杂凑函数法。本组程序始终在子程序ＣＳＥＥ控制下进行。

进行完公共子表达式节省后，ＣＳＥＥ子程序返回到θPTPRθＣＥＳＳ子程序，然后，它将调用ＢＡＣＫＭθＶＥ子程序进行不变量运算的外移，这就是第六组程序。

第六组程序进行不变量运算的外移是将不变量运算移至后目标中，后移是有条件的，它是在一个运算片中至少有两个以上的运算对象在所处理的循环中没有定义。在外移的同时，本组程序还进行了简单存贮的传递。

第七组程序是强度的削减，它只限于对递归定义的量，递归定义变量即在循环内只有此次定义。削减条件①限于Ｃ＊Ｖ，Ｖ±Ｋ运算，其中Ｃ，Ｋ为常数，Ｖ是递归变量，即是循环控制变量。②限于块内必须是下标运算。

第八组程序是内存和变址的分配。根据上述节省，削减和不变量运算的外移的原则，可知，块内节省，所产生的内部变量其命名片同在一个块内，而不变量外移，强度削减均是和该循环的后目标块打交道，块外节省的算法指出，一个块内的片节省，总是沿着其向后优先块直至循环入口块进行，所以决定下述分配原则：①循环的嵌套层之间连续分配，而并列层重叠分配。②内部变量，递归内部变量分配内存单元，下标变量分以变址单元，当变址超过５４个时，分以假变址单元（即内存单元）。③当下变为ＣＡＬＬ语句的实参时，不分变址，而分内存单元。

执行完第八组程序后返回到θPTPRθＣＥＳＳ子程序，然后由

θPTPRθＣＥＳＳ返回到Ｂｙ２，Ｂｙ２检查循环表Ｌθθ ＰＴ是否有与上面处理的同层循环，如有，重复上述过程，当查到Ｌθθ ＰＴ表的结束，如仍没有同层循环，就将嵌套层数减１，处理次内层循环直至处理到最外层为至，最后Ｂｙ２调用ＲＥＳＴθＲＥ恢复必要的信息后，把控制返回到总控程序ＦＳＤ。完成了一段的优化任务。见流程图：

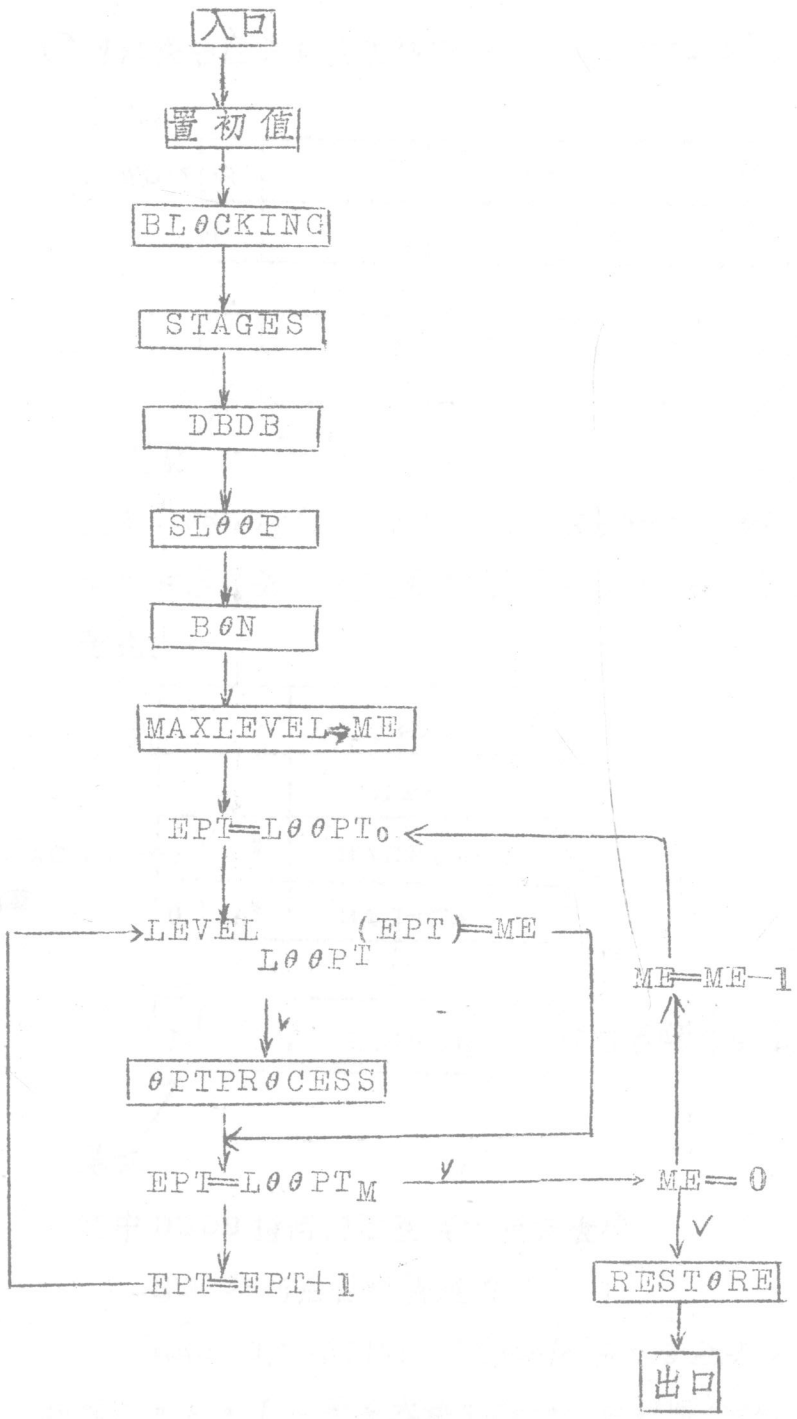
图中ＥＰＴ为 Ｌθθ ＰＴ 地址指示字。

ＭＥ为循环层数。

Ｌθθ ＰＴ₀为循环表始地址。

Ｌθθ ＰＴＭ为循环表末地址。

By₂ ：由 FSD调用



E

§ 5 优化的准备

5·1 程序分块、有向图的内部表示

(1) 定义性标号文本项结构

LAB 8	CHAIN 16
	L
	BTEND

LAB：文本项操作码——标号。

CHAIN：链下一个文本项。

L：块表项指示，指向标号字典项。

BTEND：本块文本项尾项。

其中只有BTEND由本阶段填写。

(2) 块表结构：BLT（第一阶段为LD）

每项四个半字，划分如下：

A 7	RDBF 17
B	FθRWC
L	BACKC
C	TEXT

RDBF 为该块引用，定义单位信息头。

FθRWC 向前联络区头。

BACKC 向后联络区头。

TEXT 文本项头。

A字段划分

0位 处理过否标誌。

1位 后目标块标誌

2位 含有Dθ语句标誌。

3位 循环入口块标誌。

4位

5位 循环出口块标誌。

6位 带有GθTθ语句块。

7位 程序单元的出口块。

B字段：循环编号

C字段： 级次

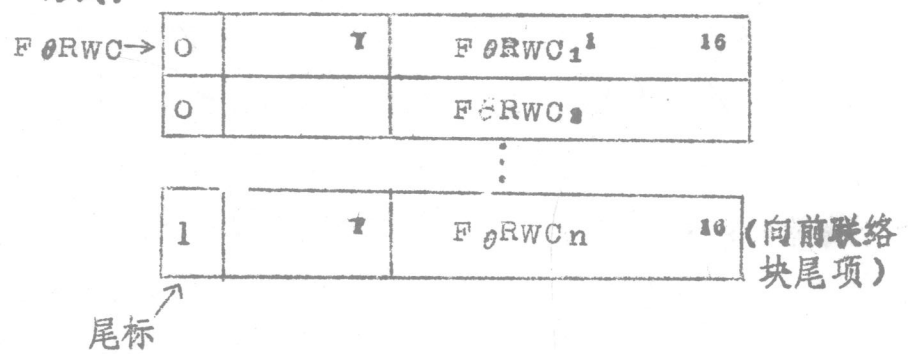
L字段：循环层次

该块表项除TEXT，B₀，B₁，B₂外都由本阶段填写。

(3)、联络信息区结构：

①向前联络区头（块表项中）F θRWC指示本块的向前联络块

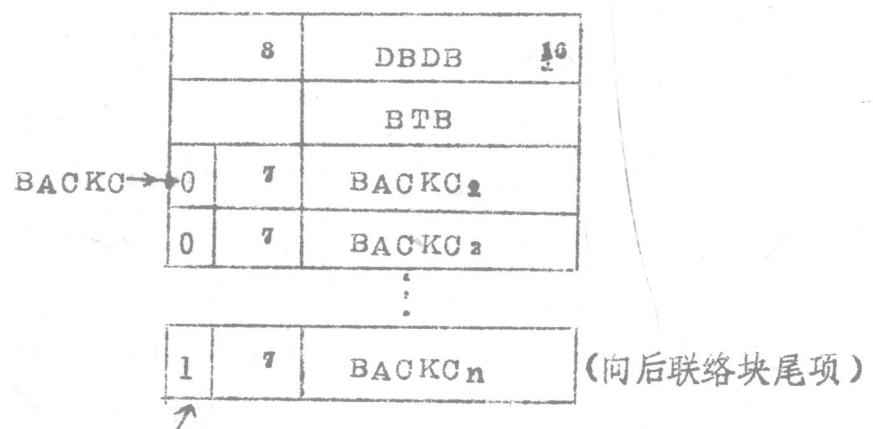
形式：



其中 F θRWC$_i$（$i \geq 1$）指向向前联络块的块表项。

②向后联络区头：包括直接优先块，后目标，向后联络块。

形式：



其中 DBDB 指向向后直接优先块表项

BTB 指向后目标块表项

BACKC$_i$ 指向向后联络块表项（$i \geq 1$）

根据2·2·1关于基本块的概念，将程序分块。即标号文本项是一个基本块的第一个文本项，而这个标号文本项的前一个

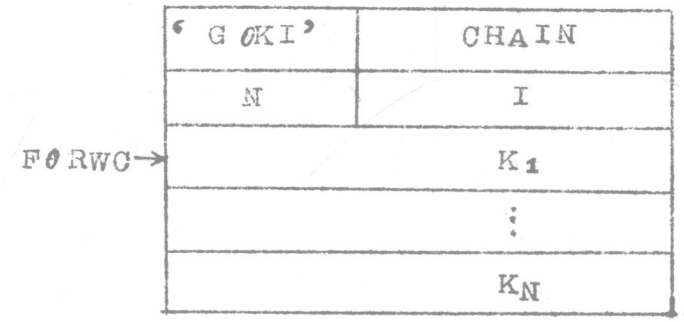
文本项就是前一个基本块的末文本项。按上面指出的原则首先以段头开始，制造一个入口块块表项和标号文本项，然后按第一阶段给出的文本项链扫描文本项，当迁到标号文本项 T 时，改造第一阶段按标号出现的顺序链结块表项为按标号定义的顺序链结，从文本项 T 的 L 字栏得到块表项指示字 L_1，将 L_1 送到前一块 L 的 BLT 的链字拦中，以达到把前一块的 BLT 链当前块的 BLT，再用 L 得到它的标号文本项指示字，把文本项 T 的前一文本项 T_F 填写在该标号文本项的 BTEND 字拦中，即填写 L 块的末文本项；按照 T_F 填写 L 的向前联络信息。

(1)、当文本项 T_F 的操作码 $\theta P_{TEXT}(T_F)$ 是各种转移如 GθTθ 计算转 GθKL，赋值转 GθIK，算术 IF，逻辑 IF 中所引用的标号项指示字（块表项）是 L 的向前联络块

(2)、当 $\theta P_{TEXT}(T_F)$='STθP' 或 'RETURN' 标识 L 块是出口块。

(3)、其他情况，当 $L_1 \neq 0$ 时，L 的向前联络块是 L_1；当 $L_1 = 0$ 时，L 没有向前联络，这发生在程序段的不合法的最后一个可执行语句的情况。

为了节省内存空间，向前联络区指示字 FθRWC 指向 GθTθ GθKL、GθIK、AIF、LIF 文本项中标号字典项指示字的首拦如计算转 GθTθ(K_1K_2, …, K_N)，I 的文本项为



並在K_N所在位置的0位上置尾项标誌。然后根据FORWC$_2$求向后联络数，即将FORWC$_2$所指向的块表项BLT的C字段加1（C字段初值为0）。

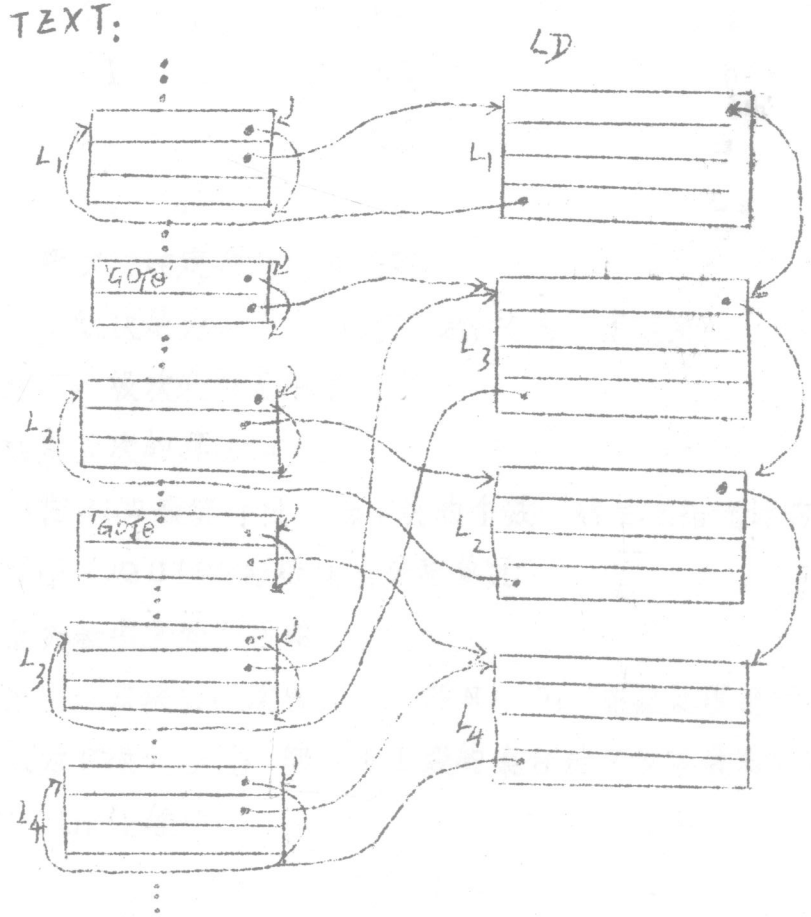
填向前联络块由ENT-FORWC子程序完成，它由BLOCKING子程序调用。计算后联络数由GBACKCMAG子程序完成，它被ENT-FORWC子程序调用。

继续扫描，对于不是标号的文本项及GOTO GOKI GOIK的文本项除了继续扫描外，不做任何工作。当迂到GOTO GOKI GOIK文本项而下一个文本项又不是标号时，制造一个新的标号定义文本项及块表项后，重复上述过程。当扫描到最后一个文本项即T＝0时，控制转向扫描BLT，从入口块开始，查字段C，根据字段C给向后联络区分以内存空间，置向后联络尾标，至BLT的链CHAIN=0结束。然后重新扫描BLT，查FORWC区每当FORWC$_2$所指向的块L_2的向后联络块就是当前扫描的块，填写向后联络，並将$C_{BLT}(L_1)$减1，直到所有的BLT都扫描到为止。

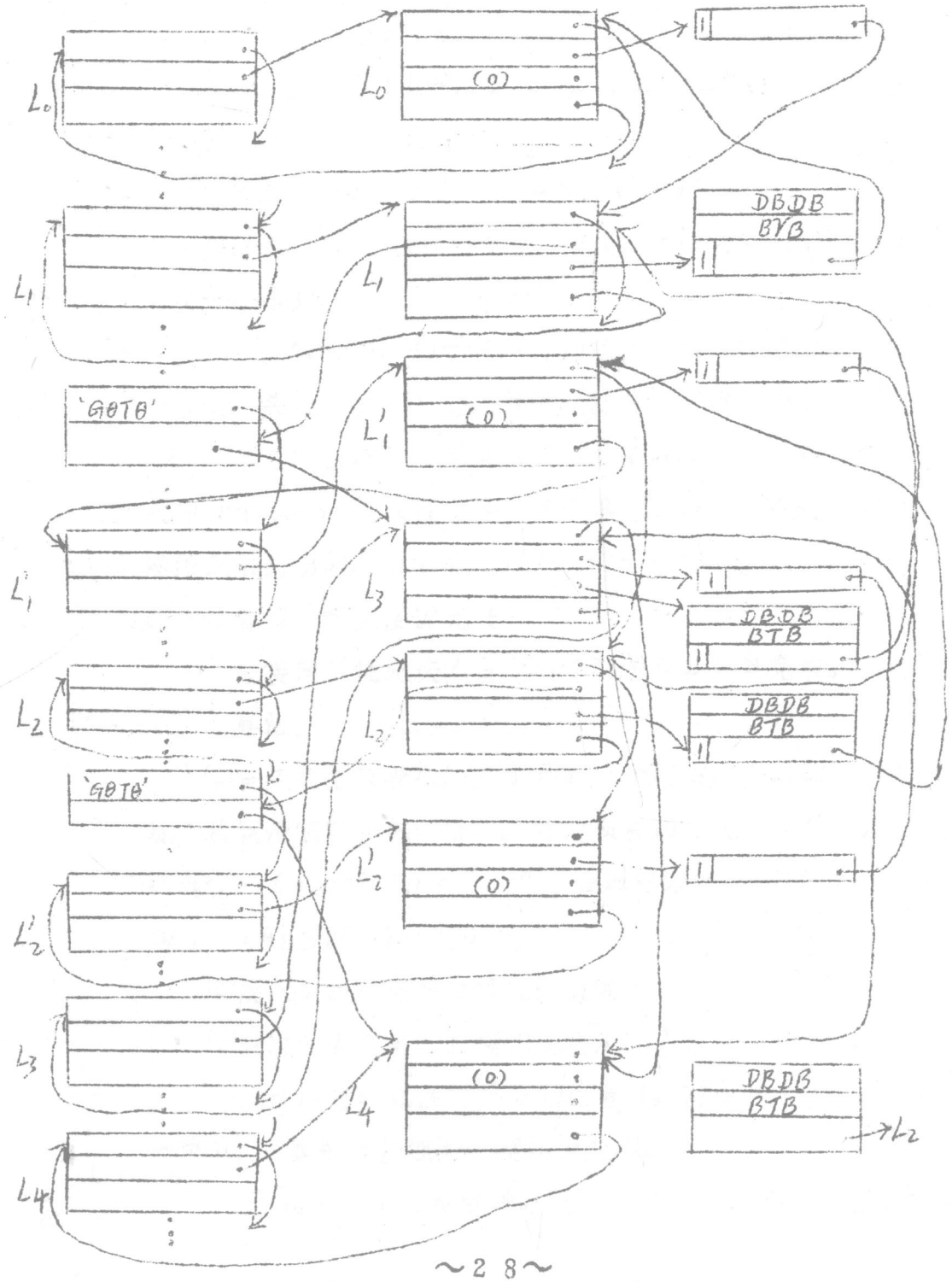
经过BLOCKING子程序就完成了程序划块，填写各块联络信息之任务。

例：源程序

第一阶段中间文本项及块表表示



经过优化阶段的BLOCKING子程序以后的情况：



列表如下

块名	FORWC	BACKC
L_0	L_1	0
L_1	L_3	L_0
L'_1	L_2	0
L_2	L_4	L'_1
L'_2	L_3	0
L_3	L_4	L_1
L_4	0	L_2

5．2块级次的确定及级次表说明：

确定级次是为了下面确定直接优先块和查找循环，一旦这些工作结束，级次表就无用了。

确定级次的算法如下：

(1)扫描BLT表统计段内基本块的个数，然后根据统计的数字调用子程序INQUIRE给级次表分配单元

(2)置块表项末处理标誌

(3)把段入口块的级次置为0，即N＝0，並置处理过标誌。

(4)认为N级处理过，则N＋1级的集合是N级集合的向前联络块集合且未处理过

即 $\{y_i^{N+1}\}=\{x \mid \in \langle x_0, x\rangle\ x_0 \in \{y_i^N\}, \langle x_0, x\rangle \in R\}$

其中R为向前联络关系集合 $\{y_i^0\}=\{$段入口块$\}$，当 $\{y_i^{N+1}\}=\{\phi\}$ 时级次确定结束。