四、與機器無關的最佳化階段 (Machine-independent Optimization phase)

一 功能：

輸入中間矩陣，運用與機器無關的最佳化技巧，產生與機器無關的最佳化矩陣，並且更新識別字表及文字表的相關部分。

二. 資料基底：

1. 矩陣 -- 供輸入及輸出，為了能在矩陣中加入或刪除某一項目，矩陣向內包含了向前及向後指標；矩陣項目的格式如下：

運算符號	運算元1	運算元2	向前指標	向後指標

其中 向前指標存放前面一個矩陣項目的註標 (index)。

向後指標存放後面一個矩陣項目的註標。

2. 識別字表 -- 供參考及執行更新作業。

說明：本階段將參考識別字的訊息，同時執行最佳化時可能將某些暫時儲存符號刪除。

3. 文字表 --供參考及執行更新作業。

說明：本階段將慘考文字的訊息，同時執行最佳化時可能會加入一些新的文字 (literal) 到本表中。

三. 演算法：

討論四個與機器無關的最佳化之技巧如下：

1. 刪除共同的副式子 (common subexpression)。

(1) 限制：

限同一條指述 (statement) 的共同副式子才能夠刪除；換言之，若一條指述中有一個以上的共同副式子，則只需要保留第一個共同副式子對應的中間矩陣項目即可，而其他共同副式子對應的矩陣項目予以刪除；但對於不同一條指述而言，雖然具有相同的副式子，為了避免錯誤發生，不得刪除相同副式子對應的矩陣項目；

例如：下列程式片段裡，B的值經第二條指令運算後改變，使的第一及第三條指令中 (A + B) 的運算值不同，故不得刪除這兩條指令中 (A + B) 對應的矩陣項目。

D = (A + B) * C ;

B = B + 2 ;

F = (A + B) * E ;

(1)例子：(刪除共同的副式子之例子)

C = B * A

A = A + 2

D = A * B / (B * A + C)

a. 原始程式片段例子

M1	*	B	A	0	2
M2	=	C	M1	1	3
M3	+	A	2	2	4
M4	=	A	M3	3	5
M5	*	A	B	4	6
M6	*	B	A	5	7
M7	+	M6	C	6	8
M8	/	M5	M7	7	9
M9	=	D	M8	8	?

指述 C = B * A

指述 A = A + 2

指述 D = A * B / (B * A + C)

b. 最佳化前中間矩陣

M1	*	A	B	0	2
M2	=	C	M1	1	3
M3	+	A	2	2	4
M4	=	A	M3	3	5
M5	*	A	B	4	6
M6	*	A	B	5	7
M7	+	C	M6	6	8
M8	/	M5	M7	7	9
M9	=	D	M8	8	?

指述 C = B * A

指述 A = A + 2

指述 D = A * B / (B * A + C)

c. 將運算符號為＋或＊之矩陣項目的兩運算元按字母順序排列後矩陣

M1	*	A	B	0	2
M2	=	C	M1	1	3
M3	+	A	2	2	4
M4	=	A	M3	3	5
M5	*	A	B	4	7
M6	*	A	B	5	7
M7	+	C	M5	5	8
M8	/	M5	M7	7	9
M9	=	D	M8	8	?

指述 C = B * A

指述 A = A + 2

指述 D = A * B / (B * A + C)

(d)刪除共同副式子後中間矩陣

2. 在編譯時計算 (compile time compute)。

(1) 說明：

在編譯時將常數 (constant) 間的運算先予計算，變成節省目的程式之儲存空間，而且縮短程式之執行時間。

(2) 例子：(編譯時計算的例子)

A = B – 5 * 12 / 10

(a)原始程式片段

M1	*	5	12	0	2
M2	/	M1	10	1	3
M3	-	B	M2	2	4
M4	=	A	M3	3	?

(b)最佳化前中間矩陣

M1					
M2	/	60	10	0	3
M3	-	B	M2	2	4
M4	=	A	M3	3	?

(c)第一個矩陣項目予以計算，並且經刪除及代換後中間矩陣

M1					
M2					
M3	-	B	6	0	4
M4	=	A	M3	3	?

(d)第二個矩陣項目予以計算，並且經刪除及代換後最佳化矩陣

3.布林式子的最佳化 (Boolean expression optimization)。

(1) 說明：

利用布林式子的特性來縮短計算。例如：一指述 IF L1 OR L2 OR L3 THEN F1 其中 L1，L2及L3均為邏輯式子；F1是由一個或多個指述所形成；將採用布林代數的功能與否分別說明執行的情形如下：

1. 不採用布林代數的功能時：L1，L2及3全部都要計算及測試計算結果是否為真，當三者中任一為真時才執行F1。

2. 採用布林代數的功能時：首先計算及測試L1，當L1為真則L2及L3都不必計算及測試便執行F1，否則計算及測試L2，當L2為真則L3不必計算及測試便執行F1，否則才計算及測試L3，當L3為真時執行F1；如此這般便可縮短計算。

(2) 演算法：

將含布林代數的指述展開以達成最佳化；例如將上述的例子用鍊狀的IF指令取代如下：

```
IF  L1  THEN  F1
ELSE  IF  L2  THEN  F1
ELSE  IF  L3  THEN  F1
```

雖然取代後程式本身較不精簡，卻能縮短程式的執行間。

4.將迴路中不變的計算式子移出迴路外 (move invariant computations outside of loops)。

(1)說明：

對於迴路內任一式子而言，若一式子的所有變數在迴路中計算時變數的值都不改變，則能將此式子移出迴路外，以便節省程式的執行時間。

(2)演算法：

1. 辨認不變的計算式子 -- 例如，下面的程式片段中，那些式子是不變的計算式子呢？

```
    ⋮
DO  I = 1 TO 10；
A = 5；
B = D + 2；
C = C + 2；
END；
    ⋮
```

從上述程式片段可看出A = 5；及B = D + 2；等兩條指述內的變數A、B及D等的值在迴路中始終不改變，得知這兩條指述屬於不變的計算式子，故可移出迴路外；而指述 C = C + 2；內變數C的值在迴路中一直在改變，故不得移出迴路外。

2. 找出放置不變的計算式子之適當位置 -- 通常是迴路起始的前面一個位置。

3. 移動不變的計算式子到適當的位置 -- 只需更改矩陣項目中小部分的向前及向後指標即可。

五、儲存位置之分配階段 (Storage Assignment phase)

(一)功能：

1. 分配儲存位置給程式用到的所有變數。
2. 分配儲存位置給存放中間結果的所有暫時儲存位置，例如：於解釋階段中預留用來存放矩陣行結果的暫時儲存位置。

3. 分配儲存位置給所有的文字 (literal)。
4. 對於文字及某些變數被分配的儲存位置予以設定起始值。

(二)資料基底：

1. 識別字表。
2. 暫時儲存位置表 (可視為識別字表的一部分)。
3. 文字表。
4. 矩陣。

六、數碼產生階段 (Code Generation phase)

(一) 功能：

產生適當的組合數碼；作法是輸入矩陣，以每一個矩陣項目的運算符號欄為巨集呼叫的名稱，運算元欄為引數，將對應的數碼產生規則 (code production) (結構為巨集定義方式)展開，並且執行與機器有關的最佳化，同時參考識別字表及文字表以便產生數碼的對應位址，然後產生適當的數碼。

(二) 資料基底：

1. 輸入 -- 矩陣。

   說明：是矩陣項目的運算符號欄為巨集呼叫的名稱，運算元欄為引數，供展開對應的數碼產生規則，產生組合數碼。

2. 參考到的中間暫存表 -- 識別字表及文字表。

   說明：用來決定變數的資料型態及位置，以便產生數碼的正確位址。

3. 參考到的永久表 -- 數碼產生規則 (code production)。

   說明：數碼產生規則的內容是以巨集定義的格式存放，它以運算符號為巨集定義的名稱，以運算元及矩陣的行數為引數 (argument)。

六、數碼產生階段 (Code Generation phase)

(一) 功能：

產生適當的組合數碼；作法是輸入矩陣，以每一個矩陣項目的運算符號欄為巨集呼叫的名稱，運算元欄為引數，將對應的數碼產生規則 (code production) (結構為巨集定義方式)展開，並且執行與機器有關的最佳化，同時參考識別字表及文字表以便產生數碼的對應位址，然後產生適當的數碼。

(二) 資料基底：

1. 輸入 -- 矩陣。

   說明：是矩陣項目的運算符號欄為巨集呼叫的名稱，運算元欄為引數，供展開對應的數碼產生規則，產生組合數碼。

2. 參考到的中間暫存表 -- 識別字表及文字表。

   說明：用來決定變數的資料型態及位置，以便產生數碼的正確位址。

3. 參考到的永久表 -- 數碼產生規則 (code production)。

   說明：數碼產生規則的內容是以巨集定義的格式存放，它以運算符號為巨集定義的名稱，以運算元及矩陣的行數為引數 (argument)。

矩陣	最初的組合數碼	較佳的數碼
＋ A B	(1)L 1 , A	(1)L 1 , A
	(2)A 1 , B	(2)A 1 , B
	(3)ST 1 , M1	- -
- M1 C	(4)L 1 , M1	- -
	(5)S 1 , C	(3)S 1 , C
	(6)ST 1 , M2	- -
= D M2	(7)L 1 , M2	- -
	(8)ST 1 , D	(4)ST 1 , D

圖a 指述 D ＝ A ＋ B – C 的數碼

(三) 演算法：

敘述刪除多餘的load及store指令之演算法如下：

1. 若執行時一運算元Mi已儲存於暫存器中，則不需要再將它載入 (reload)。
2. 由於矩陣轉換成數碼組合的過程中，無法預知下一矩陣行是否用到本矩陣行的暫存值，因此言到產生下一個矩陣行的對應數碼時，才決定是否要儲存本矩陣行的暫存值，如此便可刪除不必要的store指令。

-執行最佳化時盡量利用尚未使用到的暫存器；例如：IBM370系統有16個一般用途暫存器可供使用。

-利用執行速度較快的指令取代執行速度較慢的指令；例如：以執行速度較快的RR態指令取代執行速度較慢的RX指令。

七、組合及輸出階段 (Assembly and Output phase)

(一) 功能：

產生可重新定位之目的碼 (relocatable object code)，並且予以輸出。

(二) 資料基底：

1. 輸入有三：
   (1) 組合數碼。
   (2) 識別字表。
   (3) 文字表。
2. 輸出 -- 可重新定位之目的碼。

八、N次處理之編譯程式

從另一個角度來看，編譯程式對其資料基底做N次處理，請參閱下頁圖b，分述如下：

(一) 第1次處理 (語彙分析階段)：

掃瞄整個原始程式，建立識別字表、文字表及齊一符號表。

(二) 第2次處理 (語法分析及解釋階段)：

掃瞄齊一符號表，產生中間矩陣，並且將有關資訊加入識別字表中。

(三) 第3次至第N-3次處理 (最佳化階段)：

各種不同形式的最佳化，可能需對矩陣做數次處理。

(四) 第N-2次處理 (位置分配階段)：

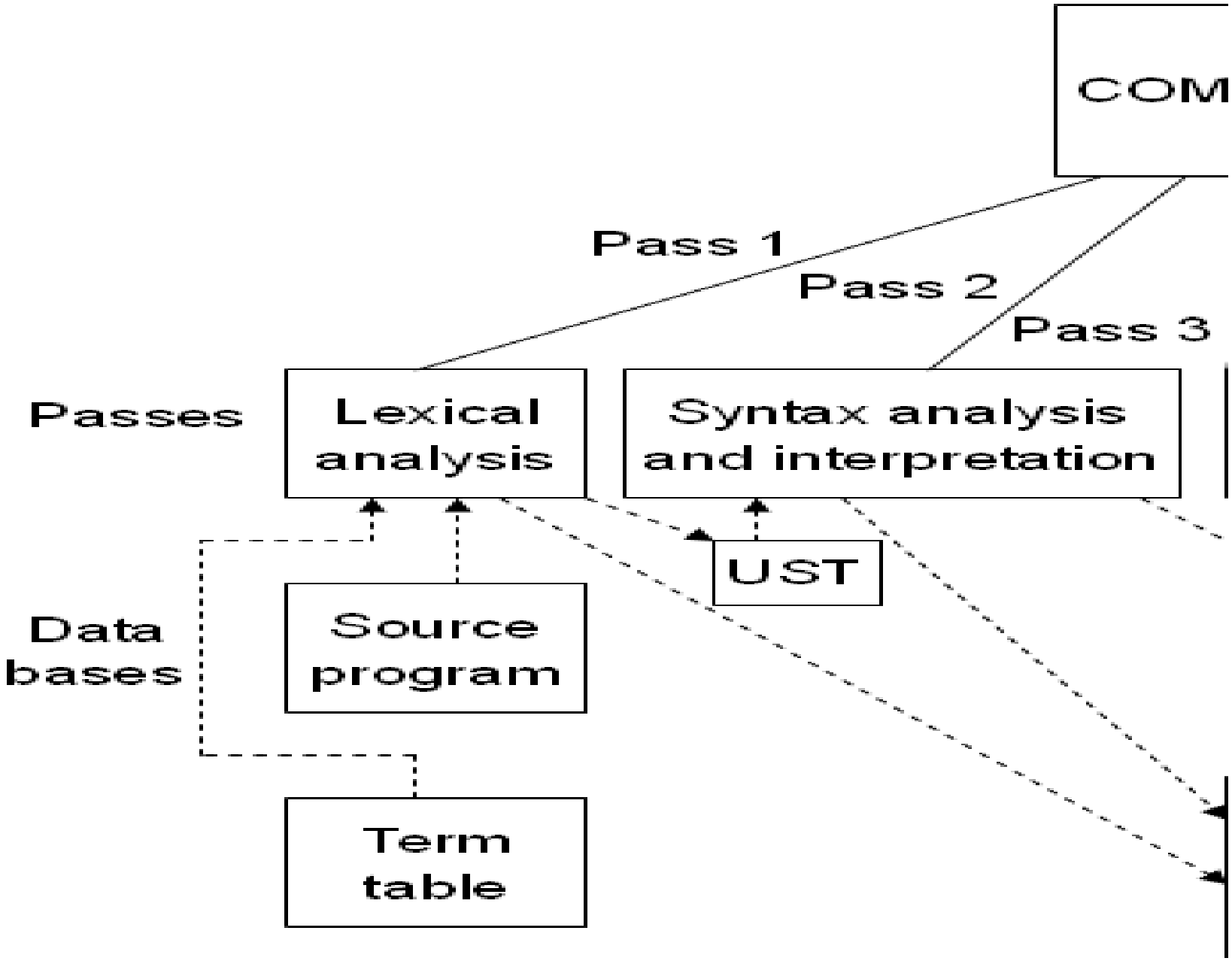
指定位置給識別字表 (含暫時儲存表)、文字表及中間矩陣，然後對具有起始值之文字及某些變數所對應的位置設定起始值。

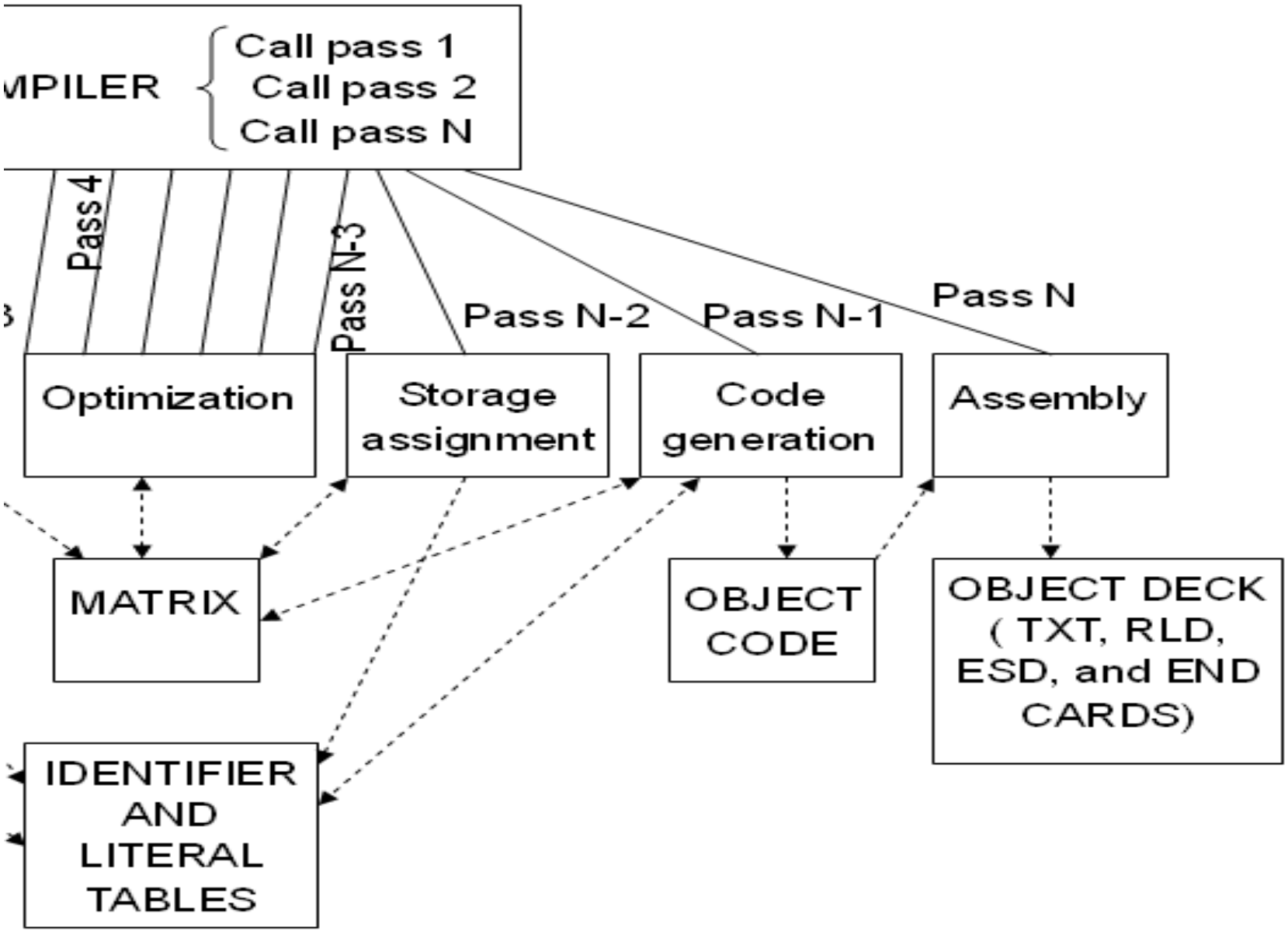
(五) 第N-1次處理 (數碼產生階段)：

輸入中間矩陣，產生最初組合數碼。

(六) 第N次處理 (組合及輸出階段)：

解決了符號的參考，產生供載入程式輸入之可重定位目的碼。


COM
Pass 1
Pass 2
Pass 3
Passes
Lexical analysis
Syntax analysis and interpretation
UST
Data bases
Source program
Term table



圖b：N次處理之編譯程式