الفصل الثامن

توليد الشفرة

- توليد الشفرة
- المعلومات المستقاة من المحلل اللفظي
- المعلومات المستقاة من المحلل القواعدي
- توليد التعابير الرباعية لأجزاء اللغة الافتراضية
- تحويل التعابير الرباعية الى لغة الماكنة
- تمارين

الفصل الثامن: توليد الشفرة

8.1 توليد الشفرة (Code Generation)

إن عملية توليد الشفرة (Code Generation) هي المرحلة التي يتم فيها تحويل الرموز التي تم تدقيق قواعدها ومعانيها الى أوامر تفهمها المكونات المادية للحاسب حيث انه من المعلوم أن كل معالج يستطيع التعرف فقط على مجموعة من الأوامر تسمى (Instruction Set) ولا يستطيع التعرف على غيرها وتمر هذه العملية بمرحلتين :

- مرحلة توليد الشفرة الوسطية (Intermediate Code Generation) وان إجراءات هذه المرحلة لا تعتمد على نوع الحاسبة (المعالج).
- مرحلة توليد الشفرة الآلة (Machine Code) وان إجراءات هذه المرحلة تعتمد على نوع الحاسبة (المعالج).

في هذه المرحلة يتم الاستفادة من المعلومات التي يولّدها المحلل اللفظي (Lexical Analysis) ومحلل القواعد (Syntax Analysis) والتي يتم خزنها في جداول تسمى جداول الرموز (Symbol Tables) . لنفترض وجود لغة افتراضية لها قواعدها وكلماتها المحجوزة ...الخ

8.2 المعلومات المستقاة من المحلل اللفظي

يقوم المحلل اللفظي بتقطيع جمل البرنامج إلى مقاطع تسمى (tokens) كما يحدد نوع ذلك المقطع (نوع بياناته) كما يقوم بخزن تلك المقاطع في جداول مختلفة، والجداول الأساسية التي يقوم المحلل اللفظي بتوليدها للغة الافتراضية السابقة هي:

- جدول الكلمات المحجوزة (Reserved Words Table)
- جدول الإشارات (Delimiters Table)
- جدول الأرقام الصحيحة (Integer Table)
- جدول الأرقام الحقيقية (Real Table)

- جدول المتغيرات (Identifier Table)
- جدول المعلومات (Information Table)

وان جدول الكلمات المحجوزة وجدول الإشارات يعتمد في توليده على اللغة المستخدمة حيث أن لكل لغة هناك كلمات محجوزة وإشارات خاصة بها فلغة باسكال مثلا لها كلمات محجوزة وإشارات تختلف عن الكلمات المحجوزة وإشارات لغة C++ مثلا، أما الجداول الأخرى

الكلمة المحجوزة	موقعها	الكلمة المحجوزة	موقعها
AND	1	LT	17
BOOLEAN	2	NE	18
CALL	3	OR	19
DIMENTION	4	OUTPUT	20
ELSE	5	PROGRAM	21
ENP	6	REAL	22
ENS	7	SUBROUTINE	23
EQ	8	THEN	24
GE	9	VARIABLE	25
GT	10		
GTO	11		
IF	12		
INPUT	13		
INTEGER	14		
LABEL	15		
LE	16		

جدول (8.1) الكلمات المحجوزة للغة المقترحة

الاشارات	موقعها في الجدول
:	1
(	2
)	3
=	4
+	5
-	6
*	7
/	8
|	9
.	10
,	11
;	12

جدول (8.2) الإشارات للغة المفترضة

فتعتمد على البرامج الذي يقوم المحلل اللفظي بتحليلها والمكتوبة بلغة معينة، وبالنسبة إلى اللغة المفترضة فان جدول الكلمات المحجوزة موضح في الجدول (8.1) وجدول الإشارات موضح في الجدول (8.2).

ان بناء جدول خاص بالأعداد الصحيحة والحقيقية يعتبر مهما لعملية توليد شفرة البرنامج الذي سينفذ على المكونات المادية للحاسبة لأنه سيقوم بتوليد ثوابت مقابلة لها . أما بالنسبة إلى جدول المتغيرات فانه يضم أربعة حقول ، الحقل الأول يتعلق باسم المتغيرات الواردة في البرنامج والحقل الثاني تذكر فيه أرقام الإجراءات التي تستخدم ذلك المتغير في عملها ، أما الحقل الثالث فتذكر فيه أنواع (TYPE) تلك المتغيرات حيث ممكن أن يكون من نوع صحيح أو حقيقي أو منطقي ... الخ. أما الحقل الرابع والأخير فتوضع فيه مؤشرات (POINTERS) لجداول أخرى كما سيتم توضيحه لاحقا.

عندما يتم بناء جدول المتغيرات فانه يحتاج إلى بعض الخطوات العملية مثل:

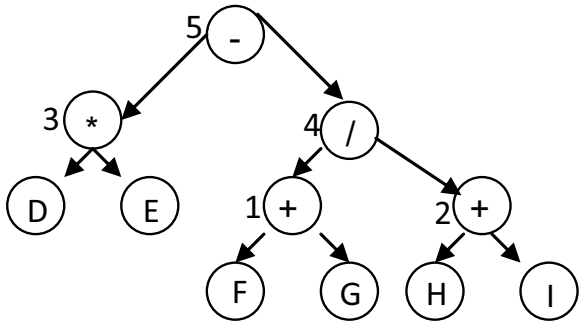
- أن يخصص لكل اسم من أسماء المتغيرات في البرنامج المطلوب ترجمته مكان معين في جدول المتغيرات وطريقة الوصول إليه منها عملية الوصول باستخدام دالة الهاش المشهورة في استرجاع البيانات (hash function).
- يتم إدخال اسم المتغير في الجدول لمرة واحدة على الرغم من استخدامه لمرات عديدة في البرنامج.
- ممكن أن يدخل اسم المتغير لأكثر من مرة إذا تم استخدام نفس اسم المتغير في أكثر من إجراء مختلف.

8.3 المعلومات المستقاة من المحلل القواعدي (Syntax Analysis)

يقوم المحلل القواعدي ببناء شجرة الإعراب لكل جملة في البرنامج فمثلا الجملة :

M: = D*E-(F+G)/(H+I)

يتم بناء شجرة الإعراب كما في الشكل (8.3):



شكل (8.3) شجرة الإعراب للجملة D*E-(F+G)/(H+I)

إن الأرقام الموضوعة على بعض دوائر الشجرة القواعدية أعلاه تمثل أسبقيات العمليات في الجملة أعلاه وبالتأكيد تشير إلى ترتيب تنفيذ العمليات في تلك الجملة ترتيبا تصاعديا.

هناك مرحلة تسمى مرحلة توليد الشفرة الوسطية (Intermediate Code Generation) وتنفذ هذه المرحلة قبل مرحلة توليد الشفرة والصيغ الوسطية في هذه المرحلة تأخذ إشكال مختلفة أهمها:

- **رموز بولش** (Polish Notations)

وهي صيغ تستخدم عادة للتعابير الرياضية كما يمكن استخدامها للتعابير غير الرياضية وهي على ثلاثة أشكال :

- **الصيغة الطبيعية** (Infix Notation)

مثال: (a+b)*(c+d)

- **صيغة قبل التحويل** (Prefix Notation)

وفيها يتم عزل العمليات الحسابية ووضعها قبل المعاملات لذا فان صيغة قبل التحويل على المثال أعلاه *+ab+cd

- **صيغة بعد التحويل** (Postfix Notation)

وفيها يتم عزل العمليات الحسابية ووضعها بعد المعاملات لذا فان صيغة بعد التحويل على المثال أعلاه ab+cd+*

- **صيغة التعابير الرباعية** (Quadruple) أو **الثلاثية** (Triple)

إن هذا النوع من الصيغ الوسطية يقوم بتحويل العبارات الى الشكل الرباعي الذي يأخذ الصيغة التالية: <**operator**>, <**operand1**>, <**operand2**>, <**result**>

المثال التالي يوضح كيفية تحويل التعبير الرياضي ((A*B+(Y+Z) إلى الصيغة الرباعية :

T1 = Y+Z ➔ + , Y , Z , T1

T2 = A*B ➔ * , A , B , T2

T3 = T1+T2 ➔ + , T1 , T2 , T3

كما هو ملاحظ من المثال أعلاه فان التعابير الرباعية تحتاج في عملها إلى متغيرات مؤقتة وان صيغة التعابير الرباعية تساعد في تطوير البرامج وتحسينها (Optimization).

أما بالنسبة للتعابير الثلاثية فهي حالة خاصة من التعابير الرباعية غير أنها لا تستخدم متغيرات مؤقتة بل تستخدم مؤشرات (pointers) فعلى سبيل المثال فان التعبير الرياضي A*B+(Y+Z) يمثل بطريقة التعبير الثلاثي على النحو الآتي :

(1) + , Y , Z

(2) * , A , B

(3) + , (1) , (2)

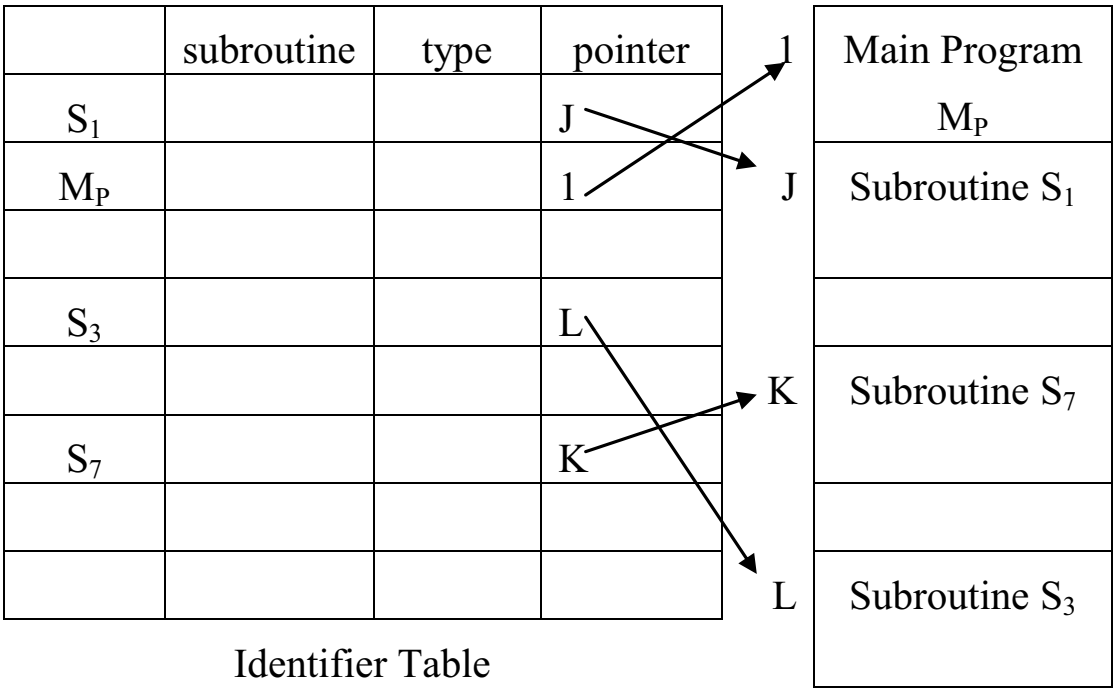
وعادة يتم استخدام أماكن في الذاكرة يؤشر عليها مؤشر معين لخزن النتائج فمثلا في الخطوة الأولى من المثال أعلاه فانه يتم جمع محتوى المتغير Y مع محتوى المتغير

Z ويوضع الناتج في مكان من الذاكرة يؤشر عليه المؤشر (1) ...وهكذا، لذا لا يفضل استخدام هذه الطريقة بسبب أي تغيير في البرنامج يتبعه تغيير كبير في ترقيمه (مؤشراته) وبالتالي تكون الصيغة الرباعية هي الأكثر شيوعا.

8.4 توليد التعابير الرباعية لأجزاء اللغة الافتراضية

8.4.1 توليد الصيغة الوسطية لبداية البرنامج

لو كانت بداية البرنامج في اللغة الافتراضية تبدأ بالجملة (program heading) أي بكلمة program وهي من الكلمات المحجوزة في تلك اللغة كما بينا ذلك من قبل ثم معرّف معين يدل على اسم البرنامج الذي يضاف إلى جدول المتغيرات على أن لا يوجد ما يربطه بأي برنامج فرعي (subroutine) وان ليس له أي نوع من أنواع البيانات ولكن لديه مؤشر يؤشر إلى الجدول الرباعي (quadruple table) وقيمته (1) للدلالة على بداية البرنامج كما ينطبق الكلام السابق على كل إجراء كما موضح في الشكل (8.4) التالي :



شكل (8.4) بداية ونهاية البرنامج الرئيسي وأي إجراء

8.4.2 توليد الصيغة الوسطية لمتغير عند التصريح

عند تعريف متغير في جزء تصريح المتغيرات كما لو كانت اللغة الافتراضية كذلك فان كل متغير سيستخدم إما في البرنامج الرئيسي أو احد إجراءاته لذا فانه يمثل في جدول التمثيل الرباعي، وهناك رقم يشير إلى استخدامه إما في البرنامج الرئيسي أو إحدى إجراءات البرنامج الرئيسي وهناك رقم يشير إلى نوع البيانات (type) لذلك المتغير ولنفترض أن الأرقام التالية تمثل أنواع البيانات (data types)المعرفة في اللغة الافتراضية:

Array = 1

Boolean = 2

Character = 3

Integer = 4

Label = 5

Real = 6

يقوم مولّد الشفرة بتكوين التعابير الرباعية لكل متغير x_i من متغيرات البرنامج إعتمادا على الطريقة التالية :لنفترض أن المتغير x_i كان يشغل الموقع L_i في جدول المتغيرات لذا سيكون التعبير الرباعي المقابل له هو :

$((5, L_i)$, , ,)

حيث يمثل الزوج المرتب $(5, L_i)$ الحد الأول في التعبير الرباعي ويمثل الرقم 5 أن هناك متغير وتمثل L_i موقع ذلك المتغير في جدول المتغيرات ، أما الحدود الثلاثة البقية تبقى فارغة والأرقام أدناه تمثل أنواع المتغيرات التي تذكر في الموقع الأول من الزوج المرتب في الحد الأول للتعبير الرباعي للمتغيرات.

(1) الإشارات (delimiters)

(2) الكلمات المحجوزة (reserved words)

(3) الأرقام الصحيحة (integers)

(4) الأرقام الحقيقية (reals)

(5) متغيرات مصرحة في البرنامج (identifiers)

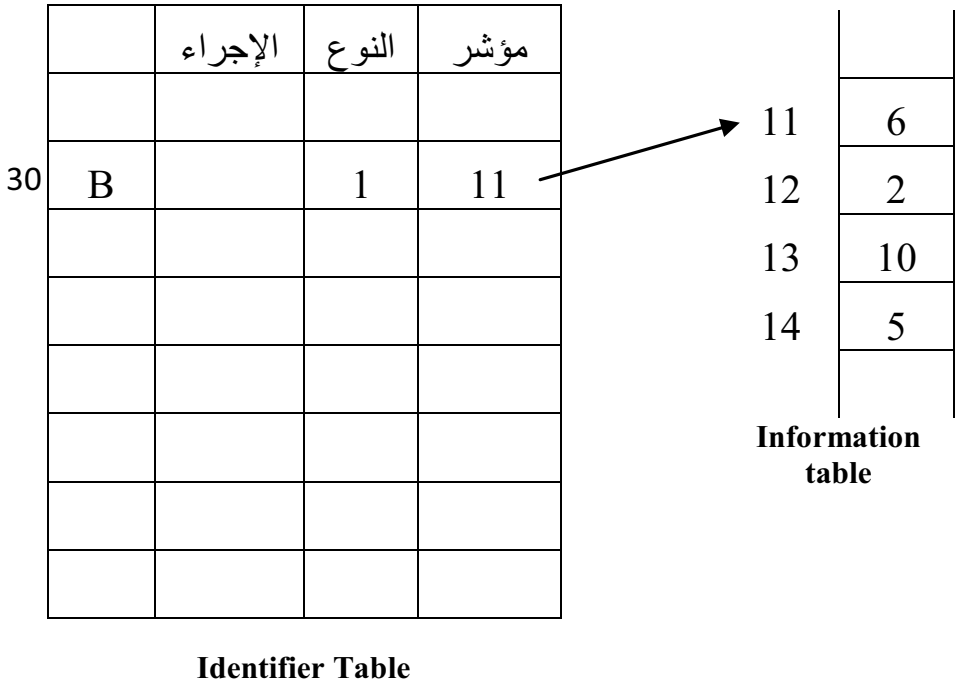
8.4.3 توليد الصيغة الوسطية للمصفوفات

تصرح المصفوفات (arrays) باستخدام عبارة (dimension) في اللغة المفترضة . لنفترض أن لدينا مصفوفة (B) وعناصرها أعداد حقيقية وذات بعدين الأول (10) والثاني (5). يقوم محلل القواعد (syntax analysis) بخزن معلومات عن مثل هذه المصفوفة في جدول بالذاكرة الرئيسية يسمى جدول المعلومات (information table) والذي يكون هيكله كالآتي:

نوع بيانات المصفوفة (حقيقي)	6
عدد الإحداثيات	2
طول الاحداثي الأول	10
طول الاحداثي الثاني	5

Information table

ويقوم جدول المتغيرات بالإشارة إلى جدول المعلومات المتعلق بالمصفوفة كالآتي:



في جدول المعرّفات لاحظ أن نوع المصفوفة B (type) يساوي (1) أي أن المتغير B من نوع مصفوفة. ويتم توليد تعبير رباعي للمصفوفة B على أنها معرّف (identifier) كالآتي على فرض أن المتغير B كان مصرح في الموقع (30) من جدول المتغيرات:

((5, 30) , , ,)

8.4.4 توليد المتغيرات الوسطية للعناوين (Labels)

لو كان لدينا العنوان في المثال الاتي :

```
LABEL   XX
        .
        .
        .
XX    Y = Z * W
```

فان العنوان XX يعامل معاملة أي متغير ويخزن في جدول المتغيرات وفي مثالنا لو خزن العنوان XX في الموقع 10 من جدول المتغيرات فان التعبير الرباعي المقابل له هو :

((5 , 10) , , ,)

8.4.5 توليد المتغيرات الوسطية لعبارة الإحلال (Assignment)

من المعلوم لدينا معنى الإحلال، مثال ذلك:

Z = X + Y

فانه يتم إحلال مجموع X مع Y محل القيمة الموجودة في Z.فان التعبير الرباعي لها حسب قاعدة التعابير الرباعية (operator operand1 operand2 result) هو :

(+, X, Y, Z)

على أن يمثل كل عنصر من عناصر التعبير الرباعي بزوج من الأرقام تمثل النوع والموقع لذا فان علامة + تمثل بالزوج المرتب (5 , 1) حيث يشير الرقم 1 إلى نوع المتغير (إشارة جمع) أما الرقم 5 فيشير الى موقع إشارة + في جدول الإشارات، أما

المتغيرات X,Y,Z فسبق شرح تمثيلهم في جدول المتغيرات ولو فرضنا ان مواقعهم في جدول المتغيرات كانت L_1, L_2, L_3 لأصبح التعبير الرباعي المقابل لجملة الإحلال أعلاه هي :

$((1 , 5) , (5 , L_1), (5 , L_2) , (5 , L_3))$

مثال آخر على جملة الإحلال:

$W = X + Y * Z$

إن جملة الإحلال أعلاه تحول إلى تعبيرين رباعيين كما يلي :

$(* , Y , Z , T1)$

$(+ , X , T1 , W)$

ولكن أيهما ينفذ أولا من قبل المكونات المادية للحاسوب (Hard Ware) ؟

باستخدام خوارزمية التحويل الى رموز بولش المعكوسة (Reverse Polish Notation) فان جملة الإحلال أعلاه تتحول إلى الصيغة التالية :

$(W , X , Y , Z , * , + , =)$

وبهذا تكون عملية التحويل إلى التعابير الرباعية امرأ سهلا بإتباع الخطوات الآتية :

1. اجعل المتغير i مساويا للـ 1.
2. خذ أول إشارة (من اليسار إلى اليمين) وارمز لها بالرمز O_i .
3. أبدا بالبحث بالاتجاه المعاكس وارمز للعنصرين اللذان يقعان مباشرة قبل O_i بالرمزين X_1 , X_2 وبالتالي نكون قد حصلنا على ما يلي :

$T_i = X_1 O_i X_2$

حيث يوضع الناتج في متغير مؤقت T_i.

4. إذا انتهى المدخل (انتهت رموز بولش المعكوسة لجملة الإحلال) توقف وبعكسه اجعل (i = i +1) وارجع للخطوة (2) أعلاه.

وبتطبيق الخطوات الأربعة أعلاه على جملة الإحلال W = X + Y * Z نحصل على ما يلي:

W, X, **Y, Z, ***, +, =	T1 = Y * Z	(* , Y , Z , T1)
W, **X , T1 , +** , =	T2 = X + T1	(+ , X , T1 , T2)
W , T2 , =	W = T2	(= , T2 , , W)

8.4.6 توليد المتغيرات الوسطية لعبارة إحلال المصفوفات

لو كانت لدينا جملة الإحلال التالية والتي تحتوي على مصفوفة:

Y = A(2,3) +5

وان المصفوفة A أعلاه يصرح عنها بالطريقة التالية في اللغة المفترضة :

DIMENTION INTEGER: A (2, 3)

وهناك أكثر من طريقة لخزن المصفوفات في الذاكرة والطريقة التقليدية هي أن تخزن بالطريقة التالية:

A(1,1) , A(2,1), A(1, 2) , A(2, 2) , A(1, 3) , A(2, 3)

أي أن الخلية A(I,J) تشغل الموقع (I + M (J-1)) حيث أن M تمثل عدد اسطر المصفوفة A (M=2) وعليه فان الخلية A(1,1) تشغل الموقع الأول (1+2*(1-1)) من المواقع التي ستخصص للمصفوفة في الذاكرة وكذلك فان A(1,3) ستشغل الموقع الخامس (1+2*(3-1)) من المواقع التي ستخصص للمصفوفة في الذاكرة وهكذا

بالنسبة للبقية . ولتمثيل المصفوفة أعلاه باستخدام التعابير الرباعية نستعين بصيغة بولش المعكوسة وبعدها تحول إلى تعابير رباعية كما يلي:

T1 = J – 1 ➔ (- , J , 1 , T1)

T2 = T1 * M ➔ (* , T1 , M , T2)

T3 = T2 + I ➔ (+ , T2 , I , T3)

كما نحتاج إلى تعبير رباعي ليقوم بعملية الإحلال :

T4 = A(T3) ➔ (= , A , T3 , T4)

لو رجعنا إلى مثالنا أعلاه Y = A(2,3) +5 فانه يمكن الحصول على التعابير الرباعية التالية :

T1 = 3 – 1 ➔ (- , 3 , 1 , T1)

T2 = T1 * M ➔ (* , T1 , M , T2)

T3 = T2 + 2 ➔ (+ , T2 , 2 , T3)

T4 = A(T3) ➔ (= , A , T3 , T4)

T5 = T4 + 5 ➔ (+ , T4 , 5 , T5)

Y = T5 ➔ (= , T5 , , Y)

أما إذا كانت المصفوفة تقع على جانب اليسار من جملة الإحلال مثل A(I) = B(I, J) فيعوض عن B(I, J) بمتغير مؤقت مثل T4 فتصبح جملة الإحلال بالشكل A(I) = T4 وتعالج كما في الطريقة أعلاه، وبهذا تكون لدينا ثلاثة أنواع من عبارات الإحلال كما يلي:

X = Y ➔ (= , Y , , X)

A(I) = X ➔ (= , X , A , I)

X = A(I) ➔ (= , A , I , X)

8.4.7 توليد الصيغة الوسطية للعبارة الشرطية

هناك أكثر من صيغة للعبارة الشرطية ومنها الصيغة القواعدية التي تتبعها اللغة الافتراضية وهي :

If <expression> then <statement> else <statement>

والتي يمكن أن تتحول إلى تعبير رباعي على النحو الأتي:

(IF , P , Q1 , Q2)

وهذا يعني انه إذا تحقق الشرط P فيتم تنفيذ الجملة Q1 وإلا تنفذ الجملة Q2.

لو كانت لدينا الجملة الشرطية التالية:

IF P THEN X=X+1 ELSE X=X+2

فان التعابير الرباعية المقابلة للجملة أعلاه هي (على فرض أن هذه التعابير تبدأ بالموقع 10 في جدول التعابير الرباعية) :

10 (IF , P , (6,11) , (6 , 14))

11 (+ , X , 1 , T1)

12 (= , T1 , , X)

13 ((2,11), , , (6, 16))

14 (+ , X , 2 , T2)

15 (= , T2 , , X)

8.4.8 توليد الصيغة الوسطية لعبارة التفرع

الشكل العام لعبارة التفرع للغة الافتراضية هي :

GTO XX;

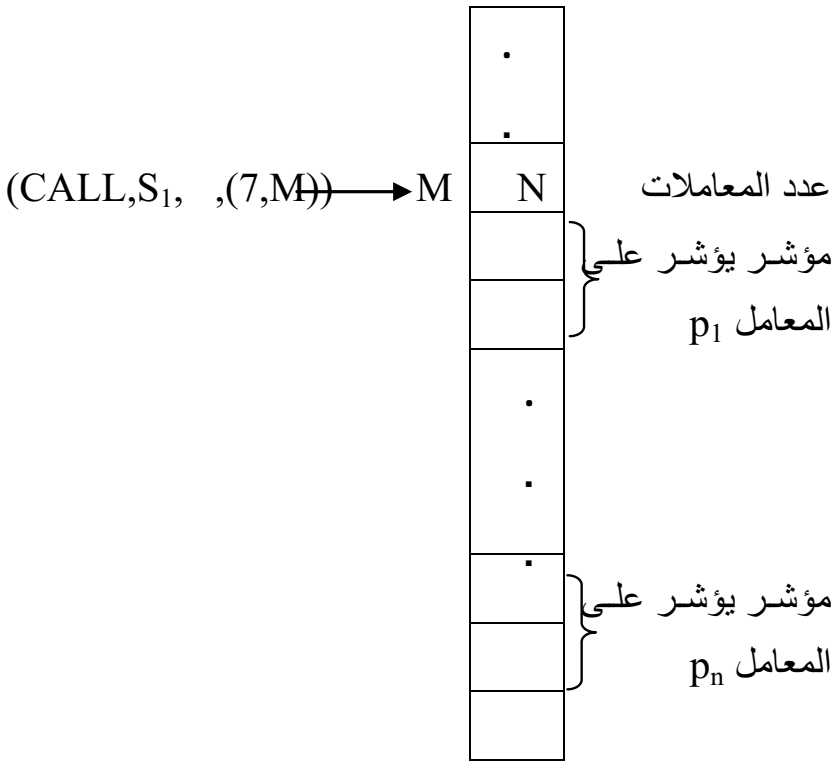
أما التعبير الرباعي المقابل لها هو (GTO , , ,Q1) حيث أن Q1 عبارة عن رقم أول تعبير رباعي من سلسلة التعابير الرباعية المناظرة للعبارة المعنونة بـ XX.

8.4.9 توليد الصيغة الوسطية لعبارة استدعاء الإجراءات

الشكل العام لعبارة الاستدعاء في اللغة الافتراضية هي :

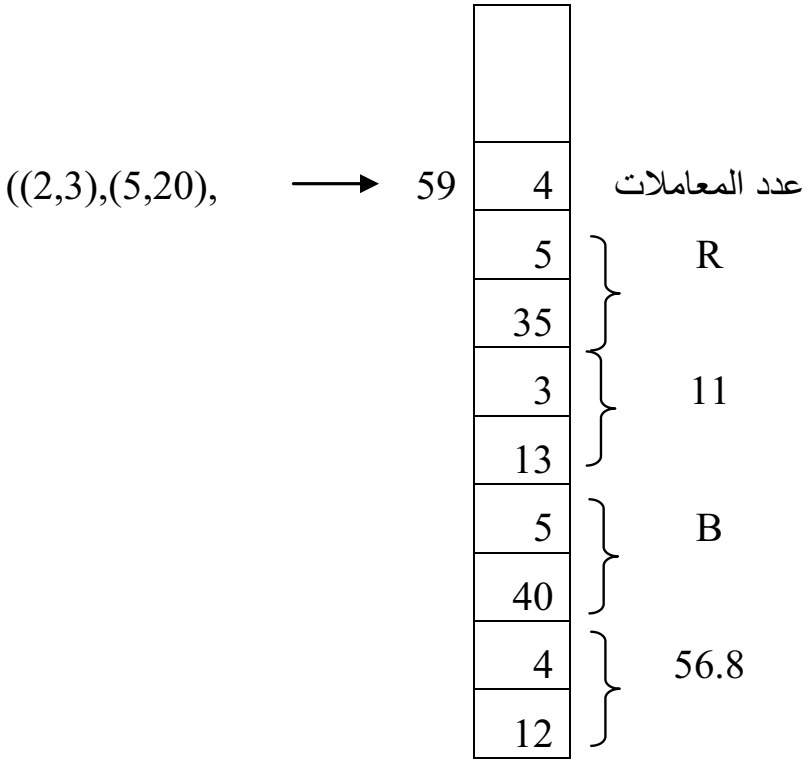
CALL S_1 (P_1, P_2 , … , P_n)

حيث أن S_1 تمثل اسم الإجراء الذي سيتم استدعاءه وان $P_1, P_2, \ldots, P_n$ تمثل معاملات ذلك الإجراء (parameters) وان المعلومات اللازمة لعملية تحويل عبارة استدعاء الإجراء تخزن في جدول يسمى بجدول المعلومات (Information Table) الذي يأخذ الهيكل التالي :



لنأخذ الإيعاز التالي : CALL T_1(R , 11, B , 56.8);

وإذا فرضنا أن المتغيرات T_1, R, B كانت مخزونة في المواقع 40 ,35 ,20 على التوالي في جدول المتغيرات والقيمة 11 تشغل الموقع 13 في جدول الأعداد الصحيحة والقيمة 56.8 تشغل الموقع 12 في جدول الأعداد الحقيقية ، عليه يكون التعبير الرباعي لعبارة الاستدعاء أعلاه كما في الشكل (8.5).



شكل (8.5) جدول المعلومات لعملية الاستدعاء

مثال8.1: ولّد التعابير الرباعية للبرنامج الآتي :

```
PROGRAM A1;
VARIABLE INTEGER: X, Y, I;
DIMENTION INTEGER A(12);
LABEL L91, L92;
I=1;
X=5;
Y=11;
L91  IF X GT Y THEN GTO L92 ELSE  X=X+2;
A(I) = X;
```

```
   I = I + 1;

   GTO L91;

L92 ENP;
```

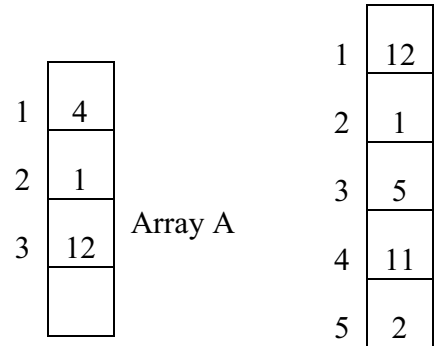
	الاسم	الإجراء	النوع	مؤشر	
1					
2	I	5	4		
3					
4					
5	A1			1 ➔	**Quadruple Table**
6					
7	A	5	1	1 ➔	**Information table**
8	X	5	4		
9					
10					
11	Y	5	4		
12					
13					
14	L91	5	5	10 ➔	**Quadruple Table**
15	L92	5	5	19 ➔	**Quadruple Table**

Identifier Table

أما جدول التمثيل الرباعي فهو:

1	((5, 8) , , ,)	X
2	((5, 11) , , ,)	Y
3	((5, 2), , ,)	I
4	((5, 7), , ,)	A
5	((5, 14), , ,)	L91
6	((5, 15) , , ,)	L92
7	((1, 4) , (3, 2) , , (5, 2))	I = 1
8	((1, 4) , (3, 3) , , (5, 8))	X = 5
9	((1, 4) , (3, 4) , , (5, 11))	Y = 11
10	((2, 10) , (5, 8) , (5, 11), (0, 1))	L91 T1 = X GT Y
11	((2, 12) , (0, 1) , (6, 12), (6, 13))	IF T1 GTO 12 ELSE GTO 13
12	((2, 11) , , , (6, 19))	GTO L92
13	((1, 5) , (5, 8) , (3, 5), (0, 2))	T2 = X+ 2
14	((1, 4) , (0, 2) , , (5, 8))	X = T2
15	((1, 4) , (5, 8) , (5, 7), (5, 2))	A(I) = X
16	((1, 5) , (5, 2) , (3, 2), (0, 3))	T3 = I + 1
17	((1, 4) , (0, 3) , , (5, 2))	I = T3
18	((2, 11) , , , (6, 10))	GTO L91
19	((2, 6) , , ,)	L92 ENP

الجدول الرباعي (Quadruple Table)



جدول المعلومات

جدول الإعداد الصحيحة

مثال8.2: ولّد التعابير الرباعية للبرنامج التالي :

```
PROGRAM AA;
VAIABLE REAL: L, M , N;
DIMENSION REAL: B(20) , C(4,5);
L = 12.5;
M= 13.7;
CALL BB(L, M, N);
B(N) = C(L, M) * 5.6;
ENP;
SUBROUTINE BB (REAL : X, Y, K)
VARIABLE REAL : Z;
Z = 8.9;
K = ( X – Z)*2.6+Y;
ENS;
```

	الاسم	الإجراء	النوع	مؤشر	
1					
2	X	11	6		
3	L	6	6		
4					
5	B	6	1	1 ➔	**Information table**
6	AA			1 ➔	**Quadruple Table**
7	N	6	6		
8	K	11	6		
9	C	6	1	4 ➔	**Information table**
10	M	6	6		
11	BB			16 ➔	**Quadruple Table**
12					
13	Y	11	6		
14					
15	Z	11	6		

Identifier Table

1	6	
2	1	
3	20	Array B
4	6	
5	2	
6	4	Array C
7	5	
8	3	عدد المعاملات
9	5	
10	3	L
11	5	
12	10	M
13	5	
14	7	N

جدول المعلومات

CALL BB(L, M, N)

1	20
2	4
3	5
4	1

جدول الأعداد الصحيحة

1	12.5
2	13.7
3	5.6
4	8.9
5	2.6

جدول الأعداد الحقيقية

أما جدول التمثيل الرباعي للبرنامج أعلاه فهو:

1	((5, 3) , , ,)	L
2	((5, 10) , , ,)	M
3	((5, 7), , ,)	N
4	((5, 5), , ,)	B
5	((5, 9), , ,)	C
6	((1, 4) , (4, 1) , , (5, 3))	L = 12.5
7	((1, 4) , (4, 2) , , (5, 10))	M= 13.7
8	((2, 3) , (5, 11) , , (7, 8))	CALL BB(L, M, N)
9	((1, 6) , (3, 3) , (3, 4), (0, 1))	T1 = 5-1
10	((1, 7) , (0, 1) , (3, 2), (0, 2))	T2 = T1 * 4
11	((1, 5) , (0, 2) , (3,4), (0, 3))	T3 = T2 + 1
12	((1, 4) , (5, 9) , (0,3), (0, 4))	T4 = C(T3)
13	((1, 7) , (0, 4) , (4, 3), (0, 5))	T5 = T4* 5.6
14	((1, 4) , (0, 5) , (5, 5), (5, 7))	B(N) = T5
15	((2, 6) , , ,)	ENP
16	((5, 2) , , ,)	X
17	((5, 13) , , ,)	Y
18	((5, 8) , , ,)	K
19	((5, 15) , , ,)	Z
20	((1, 4) , (4, 4) , , (5, 15))	Z = 8.9
21	((1, 6) , (5, 2) , (5, 15), (0, 6))	T6 = X - Z
22	((1, 7) , (0, 6) , (4, 5), (0, 7))	T7 = T6 *2.6
23	((1, 5) , (0, 7) , (5, 13), (0, 8))	T8 = T7 + Y
24	((1, 4) , (0, 8) , , (5, 8))	K = T8
25	((2,7) , , ,)	ENS

8.5 تحويل التعابير الرباعية الى لغة الماكنة

وهي المرحلة الثانية من عملية توليد الشفرة وتعتمد عملية التحويل هذه على نوع الحاسبة ومعماريتها ولنفترض أن هناك حاسبة تعمل بالأوامر التالية ، حيث كما ذكرنا من قبل أن لكل معالج مجموعة من الأوامر يستطيع فقط التعرف عليها :

الامر	المعنى
ADD	Add
ARG	Argument
AND	And Operation
DC	Define Constant
DS	Define Storage
END	End Operation
INC	Increment By 1
JSB	Jump to Subroutine
LDI	Load Indirect
LD	Load
ST	Store
STI	Store Indirect

مثال8.3: ولّد التعابير الرباعية للبرنامج التالي :

```
PROGRAM A2;

VAIABLE INTEGER: I, J , K;

DIMENSION INTEGER: A(20) , B(4,5);

I = 2;

J= 3;
```

```
CALL A3(I, J, K);

A(K) = B(I, J) + 13;

ENP;

SUBROUTINE A3 (INTEGER: X, Y, K)

VARIABLE INTEGER: Z;

Z = 6;

K = ( X – Z)^9+Y;

ENS;
```

	الاسم	الإجراء	النوع	مؤشر	
1					
2	X	11	4		
3	I	6	4		
4					
5	A	6	1	1 ➔	**Information table**
6	A2			1 ➔	**Quadruple Table**
7	K	6	4		
8	K	11	4		
9	A	6	1	4 ➔	**Information table**
10	J	6	4		
11	A3			16 ➔	**Quadruple Table**
12					
13	Y	11	4		
14					
15	Z	11	4		

1	4	
2	1	
3	20	Array A
4	4	
5	2	
6	4	Array B
7	5	
8	3	عدد المعاملات
9	5	
10	3	I
11	5	
12	10	J
13	5	
14	7	K

جدول المعلومات

CALL A3(I, J, K)

1	20
2	4
3	5
4	2
5	3
6	6
7	1
8	13
9	9

جدول الأعداد الصحيحة

1	((5, 3) , , ,)	I
2	((5, 10) , , ,)	J
3	((5, 7), , ,)	K
4	((5, 5), , ,)	A
5	((5, 9), , ,)	B
6	((1, 4) , (3, 1) , , (5, 3))	I = 2
7	((1, 4) , (3, 2) , , (5, 10))	J= 3
8	((2, 3) , (5, 11) , , (7, 8))	CALL A3(I, J, K)
9	((1, 6) , (3, 3) , (3, 7), (0, 1))	T1 = 5-1
10	((1, 7) , (0, 1) , (3, 2), (0, 2))	T2 = T1 * 4
11	((1, 5) , (0, 2) , (3,7), (0, 3))	T3 = T2 + 1
12	((1, 4) , (5, 9) , (0,3), (0, 4))	T4 = A(T3)
13	((1, 5) , (0, 4) , (3, 8), (0, 5))	T5 = T4+ 13
14	((1, 4) , (0, 5) , (5, 9), (5, 7))	A(K) = T5
15	((2, 6) , , ,)	ENP
16	((5, 2) , , ,)	X
17	((5, 13) , , ,)	Y
18	((5, 8) , , ,)	K
19	((5, 15) , , ,)	Z
20	((1, 4) , (3, 4) , , (5, 15))	Z = 6
21	((1, 6) , (5, 2) , (5, 15), (0, 6))	T6 = X - Z
22	((1, 7) , (0, 6) , (4, 5), (0, 7))	T7 = T6 ^2
23	((1, 5) , (0, 7) , (5, 13), (0, 8))	T8 = T7 + Y
24	((1, 4) , (0, 8) , , (5, 8))	K = T8
25	((2,7) , , ,)	ENS

لنأخذ المثال السابق ومن خلاله يتم توضيح عملية تحويل التعابير الرباعية إلى لغة الماكنة ، في هذه المرحلة يتم فحص جداول الأعداد الصحيحة والحقيقية لغرض حجز مواقع في الذاكرة للثوابت كما يلي :

I1	DC	20
I2	DC	4
I3	DC	5
I4	DC	2
I5	DC	3
I6	DC	6
I7	DC	1
I8	DC	9
I9	DC	13

حيث يشير المتغير Ii إلى تصريح الأعداد الصحيحة و Ri إلى تصريح الأعداد الحقيقية لو كان هناك أعداد حقيقية ،و (DC) مختصر لـ (Define Constant) بعدها يتم فحص جدول التعابير الرباعية كل تعبير على حدة وبالتسلسل فمثلا التعبير الرباعي (, , ,(5, 3)) فان مولّد الشفرة يفسر التعبير الرباعي أعلاه بان يفسر 5 على أنها متغير (معرّف) في جدول المتغيرات في الموقع 3 فان مولّد الشفرة يبحث في جدول المتغيرات في الموقع المقصود فيجد انه المتغير I وانه من نوع عدد صحيح وقد عرّف في البرنامج الرئيسي ، عليه يقوم بتخصيص خلية خزنيه واحدة أو اثنان حسب ذلك المترجم من خلال الإيعاز (I DS 1) على فرض أن المولّد هنا قد حجز خلية واحدة من خلال الإيعاز (Define Storage DS) أي حجز مساحة ولو انتقلنا إلى التعبير الرباعي (, , ,(5, 5)) لفسّر المولّد الرقم (5) الأول على انه معرّف (Identifier) والرقم الثاني (5) على انه موقع ذلك المعرّف في جدول المعرّفات (جدول أسماء

المتغيرات) حيث يتعرف المولّد على المعرّف A ومن خلال نوعه (1) يتبين أنها مصفوفة وقد عرّفت (صرّحت) في البرنامج الرئيسي وان هناك مؤشر يشير إلى الرقم (1) من جدول المعلومات (Information Table) وعند الذهاب إلى ذلك الجدول وفي الموقع (1) يجد مولّد الشفرة الرقم (4) الذي يدل على أن المصفوفة من نوع (integer) والرقم الذي بعده (1) يدل على أن المصفوفة ذات بعد واحد (one dimension) والرقم الذي يليه (20) الذي يدل على حجم المصفوفة ليتم حجز (20) خلية في الذاكرة الرئيسية لتلك المصفوفة من قبل المولّد بالتعاون مع نظام التشغيل الذي هو المسئول عن إدارة الذاكرة عن طريق الإيعاز (A DS 20) ولو أخذنا التعبير الرباعي ((1, 4), (3, 4), (5, 3)) , فان المولّد سيفسر الزوج الأول (1,4) على انه علامة (=) حيث ان الرقم (1) يفسر على انه جدول الإشارات ، أما الرقم (4) فيفسر على انه موقع تلك الإشارة في جدول الإشارات وهي (=) ، إما الزوج الثاني (4 ,3) فيفسر على انه العدد الصحيح (2) حيث أن الرقم (3) يفسر على انه جدول الأعداد الصحيحة ، أما الرقم (4) فيفسر على انه موقع ذلك العدد في الجدول وهو (2) وأخيرا الزوج المرتب (3 ,5) فيفسر على انه المتغير (I) حيث ان الرقم (5) يفسر على انه معرّف (متغير معرّف من قبل المستخدم) والرقم (3) موقعه في جدول المتغيرات على انه المتغير (I) أي أن التعبير الرباعي أعلاه يمثل الجملة (I = 2) وعليه فان مولّد الشفرة سيقوم بتوليد الأوامر التالية :

```
LD  I4

ST  I
```

حيث تم تخزين العدد الصحيح (2) في المتغير I4 كما بينا سابقا، أما الأمر LD فهو مختصر لكلمة (LOAD) ويعني تحميل العدد الصحيح (2) الموجود في I4، أما الأمر ST فهو مختصر لكلمة (STORE) ويعني تخزين العدد الصحيح (2) الذي حصلنا عليه في المتغير (I).

ولو أخذنا التعبير الرباعي ((0, 3) ,(3,7) , (0, 2) , (1, 5)) فان المولّد يفسره على انه T3 = T2 + 1 ويقوم بتوليد الأوامر التالية:

```
LD    I7

ADD   T2

ST    T3
```

فسيتم تحميل العدد الصحيح (1) المخزون في المتغير (I7) وجمعه مع المتغير المؤقت (T2) ويخزن الناتج في المتغير المؤقت (T3).ولو خذنا التعبير الرباعي التالي الذي يتضمن الاستدعاء ((7, 8) , , (5, 11) , (2, 3)) حيث يتم تفسير الزوج الأول على انه CALL لان الرقم الأول (2) تعني كلمة محجوزة والرقم الثاني (3) يعني موقع تلك الكلمة في جدول الكلمات المحجوزة وهي CALL ، أما الزوج الثاني (5,11) فيعني المتغير A3 وهو اسم الإجراء الذي سيتم استدعاؤه ، أما الزوج الأخير (7,8) فهو خزن معلومات عن معاملات الإجراء (المعاملات وعددها) والرقم 8 يشير إلى الموقع الثامن من جدول المعلومات الذي فيه المعلومات عن الإجراء عندها يتم توليد الأمر التالي لتنفيذ الاستدعاء .

JSB A_3

حيث أن (JSB) إيعاز مختصر لـ (Jump Subroutine) ويعني القفز الغير المشروط إلى الإجراء الذي اسمه A_3.

وهكذا لبقية التعابير الرباعية ، وأخيرا يقوم مولّد الشفرة التجميعية (Assembly Code) التي تمثل أوامر للمكونات المادية للحاسبة والتي تنفذ مباشرة باستخدام المجمع (Assembler) وهو برنامج يقوم بتحويل الأوامر المكتوبة بلغة التجميع إلى أوامر بلغة الماكنة (أي الصيغة الثنائية) .