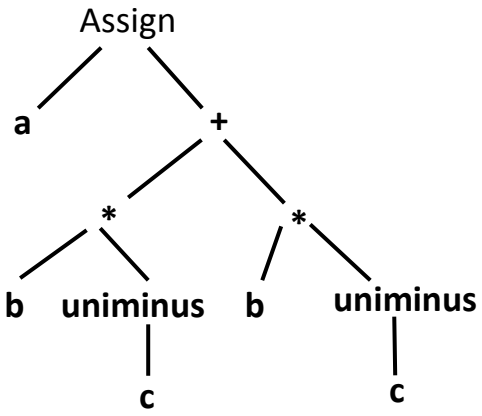
Intermediate Code Generation توليد الشفرة الوسطية

Intermediate representation are :

1. Syntax Tree
2. Postfix Notation
3. Three Address Code

1- Syntax Tree شجرة الأعراب :- هي وصف للهيكل الهرمي الطبيعي لبرنامج المصدر .
2-Postfix Notation التدوين اللاحق :-هو تمثيل خطي لشجرة الأعراب حيث يمثل قائمة من عقد الشجرة تظهر فيه كل عقدة مباشرة بعد أطفالها .

مثال :- ***a := b * - c + b * - c***



postfix notation : a b c uniminus * b c uniminus * + assign

نلاحظ اعلاه ان كل عقدة تظهر مباشرة بعد اطفالها

يتم أنشاء شجرة الأعراب بأستخدام ما يسمى SDD (Syntax Direction Definition) والذي يمثل مجموعة دوال البناء الخاصة بكل نوع من عقد الشجرة والجدول التالي يوضح ال SDD الخاص بالتعبيرات الحسابية البسيطة :-

Production	***Semantic Rule***
E → E1 + T	***E.node = New node('+', E1.node , T.node)***
E → E1 - T	***E.node = New node('-', E1.node , T.node)***
E → T	***E.node = T.node***
T→ (E)	***T.node = E.node***
T → id	***T.node = Newleaf (id,id.entry)***
T→ num	***T.node = New leaf (num , num.val)***

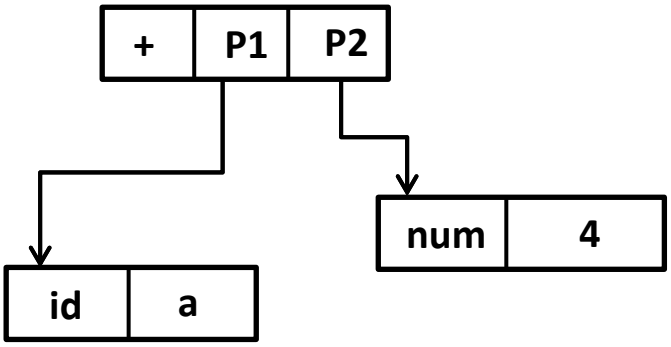
مثال :- ما هو SDD و شجرة الأعراب الخاص بالتعبير التالي ؟ أو أكتب خطوات SDD و شجرة الاعراب الخاصة بالتعبير التالي a + 4 ؟

الحل :-

P1 = New Leaf (id , a)

P2 = New Leaf (num , 4)

P3 = New Node(' + ' , p1 , p2)



س :- ما هي خطوات SDD و ما هي شجرة الأعراب الخاصة بالتعبير a -10*c

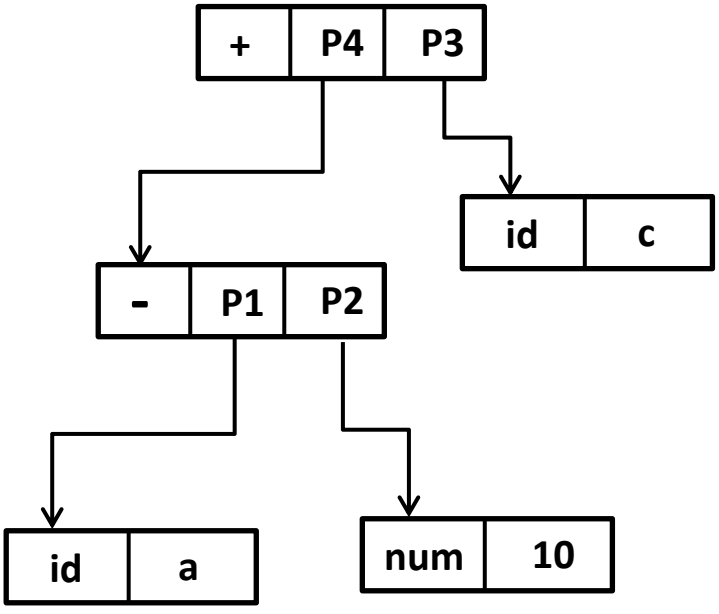
P1 = New Leaf (id , a)

P2 = New Leaf (num , 10)

P3 = New Leaf (id , c)

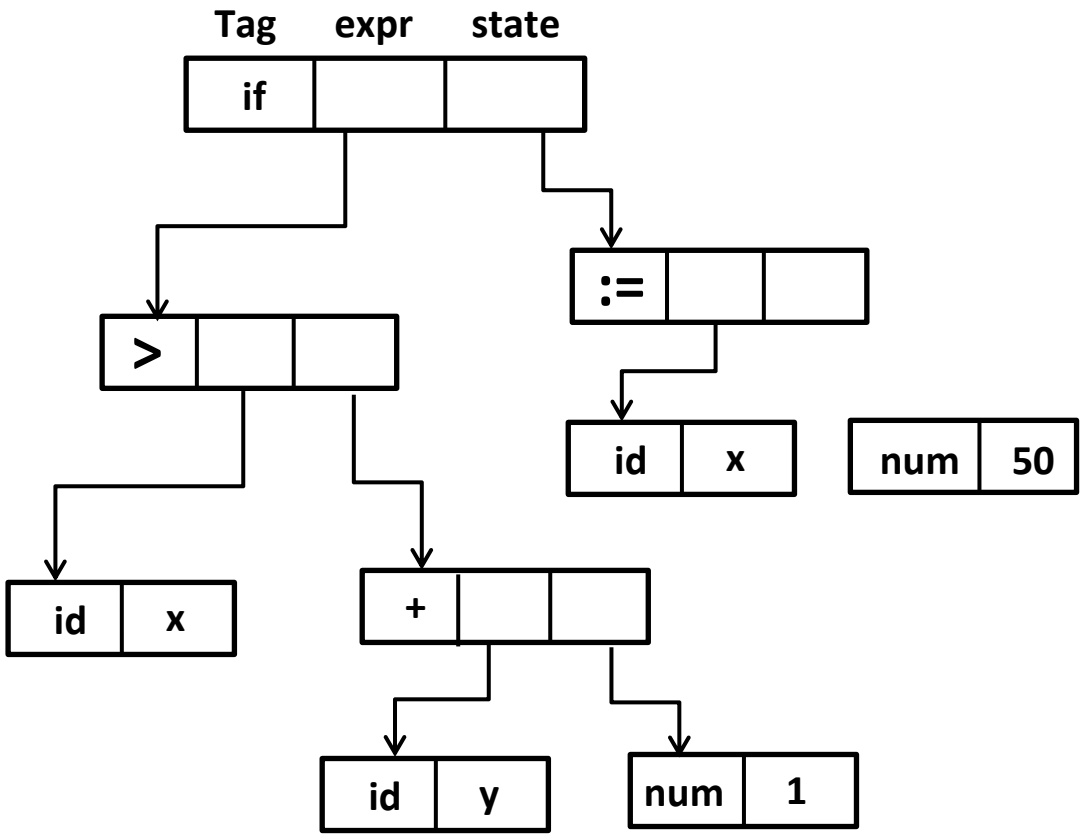
P4 = New Node (' – ' , P1 , P2)

P5 = New Node (' * ' , P4 , P3)



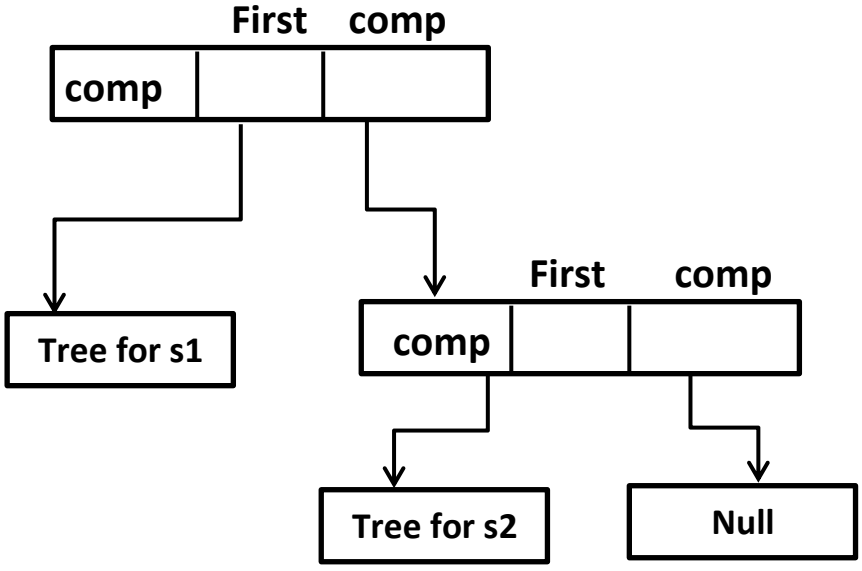
شجرة الأعراب الخاصة بالتعليمات

مثال :- *If x > y +1 Then x := 50*



مثال :- ارسم شجرة الأعراب الخاصة بالشفره التالية

```
Begin
    S1;
    S2;
end
```



شفرة ثلاثية العناوين Three Address Code

هي تتابع من التعليمات ذات الصيغة x := y op z حيث ان x,y,z هي أسماء او ثوابت مؤقتة يتم توليدها من قبل المترجم و ال op هي اي عملية حسابية أو منطقية .

مثال :- ما هي الشفرة ثلاثية العناوين الخاصة بالتعبير التالي **x + y * z** ؟

الحل :-

t1 := y * z

t2 := x + t1

أنواع التعليمات المستخدمة في الشفرة ثلاثية العناوين

1- تعليمة الاسناد **Assign Statement** و تأخذ احدى الصيغ أما **x:= y op z** أو **x:= op y**
2- تعليمة النسخ **Copy Statement** **x := y**
3- القفزة الغير شرطية **Unconditional Jump Statement** **goto L**
4- القفزة الشرطية **Conditional Jump Statement** **if (x relop y) goto L**

Relop (< , > , <= , >= , <> , ==)

Example 1 \find Three Address Code for the following expression a:= b * -c + b * -c

Solution :-

t1 := -c
t2 := b * t1
t3 := -c
t4 := b * t3
t5 := t2 + t4
a := t5

Example 2 \ find three address code for a or (b and not c)

Solution :-

```
100   if  a < b  goto  130
110   t1 := 0
120   goto  140
130   t1 := 1
140   ……..
```

Example 3 :- find three address code for a < b and c < d or e > f ?

Solution :-

```
100   if  a < b  goto  130
110   t1 := 0
120   goto  140
130   t1 := 1
140   if  c < d  goto  170
150   t2 := 0
160   goto  180
170   t2 :=  1
180   if  e > f  goto  210
190   t3 := 0
200   goto  220
210   t3 := 1
220   t4 := t1  and  t2
230   t5 := t4  or  t3
```

الثلاثيات و الرباعيات Quadruples and Triples

الشفرة ثلاثية العناوين هي صيغة مجردة في المترجم . ان هذه التعليمات قد تمثل قيود records تحتوي على حقول للعوامل operators و المعاملات operands و هذه التمثيلات تعرف بالثلاثيات و الرباعيات .

1. الرباعيات Quadruples :- هي عبارة عن هيكل قد يحتوي على اربعة حقول حقل للعمليات operator و حقل للمعامل الأول argument 1 و حقل للمعامل الثاني argument 2 و حقل اخر للنتيجة result.

أن محتوى الحقول arg 1 و arg 2 و result هي مؤشرات الى مداخل جدول الرموز الخاصة بالأسماء الممثلة لهذه الحقول كما أن الأسماء المؤقتة يجب ان تضاف الى جدول الرموز عند توليدها .

2. الثلاثيات Triples :- لتجنب اضافة الأسماء المؤقتة في جدول الرموز يجب ان نشير الى القيم المؤقتة بواسطة موقع التعليمة التي يتم احتسابها وبهذه الطريقة فان الشفرة ثلاثية العناوين يمكن ان تمثل بواسطة قيود ذات ثلاثة حقول بدلا من اربعة وهي حقل للعملية و حقل للمعامل الاول وحقل للمعامل الثاني فقط .

Example :-

a:= b * -c + b * -c

solution :-

t1 := -c
t2 := b * t1
t3 := -c
t4 := b * t3
t5 := t2 + t4
a := t5

1-Quadruples

	op	Arg 1	Arg 2	Result
(0)	Uniminus	c		t1
(1)	*	b	t1	t2
(2)	Uniminus	c		t3
(3)	*	b	t3	t4
(4)	+	t2	t4	t5
(5)	:=	t3		a

2-Triples

	op	Arg 1	Arg 2
(0)	Un minus	c	
(1)	*	b	(0)
(2)	Un minus	c	
(3)	*	b	(2)
(4)	+	(1)	(3)
(5)	assign	a	(4)