عملية الترجمة

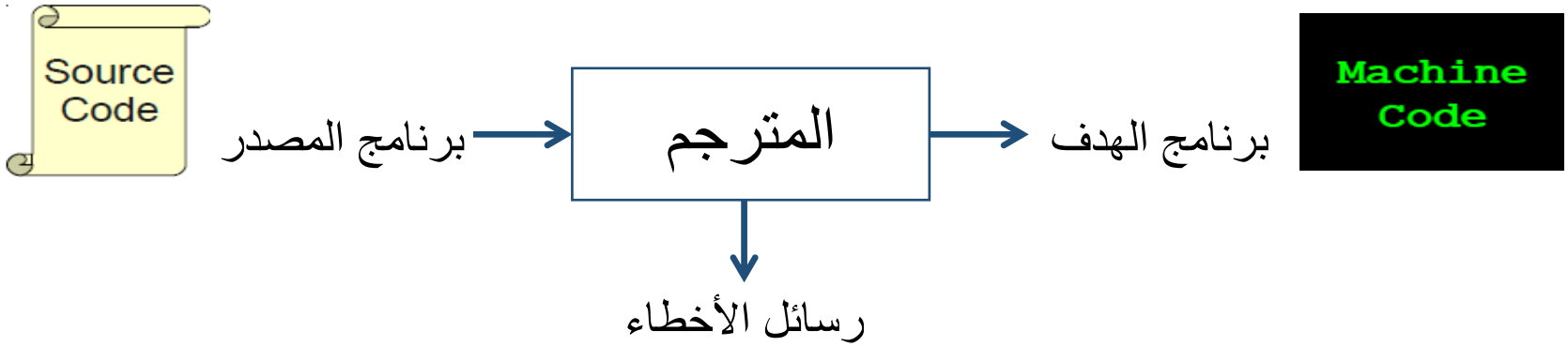
Compilation Process

المواضيع

- هيكلية المترجم Compiler Architecture
- أطوار عملية الترجمة Compilation Phases

هيكلية المترجم

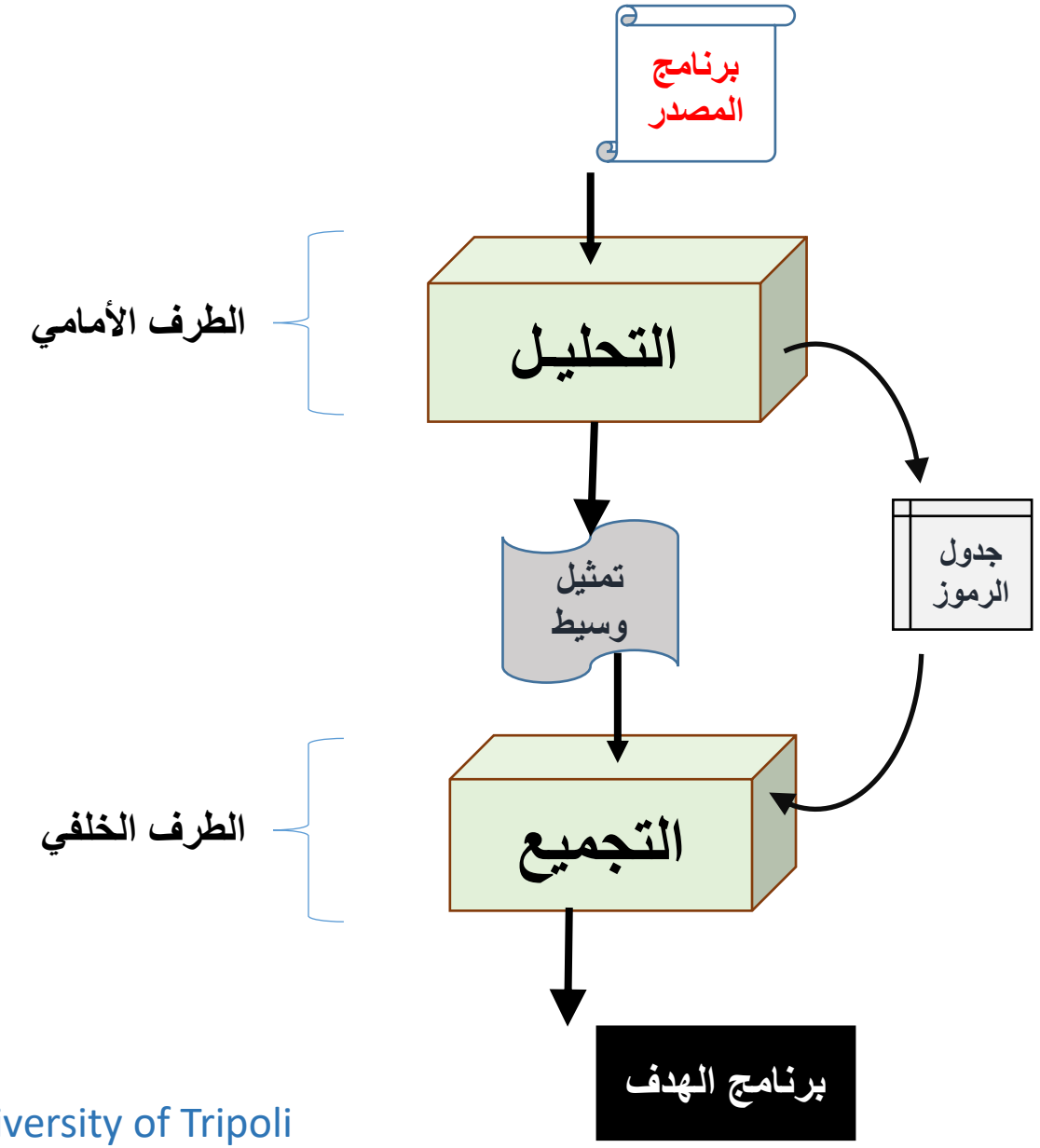
- إلى هنا عاملنا المترجم على أنه عبارة عن صندوق يقوم بتحويل برنامج المصدر إلى برنامج هدفي مكافئ له في المعنى.



- **المترجم عادة يتوصل إلى الترجمة في عدة خطوات, فبالتالي هو يحتوي على عدة مركبات تضمن من خلالها صحة قواعد صياغة المفردات والنصوص**
- إذا نظرنا داخل المترجم سنجد أن عملية التحويل تلك تتم عبر مرحلتين مترادفتين:

1. التحليل/التفكيك (Analysis) و 2. التأليف/التجميع (Synthesis)

هيكلية المترجم



1- التحليل (Analysis)

- مرحلة التحليل تختص بتفكيك برنامج المصدر إلى قطع متجانسة/متوافقة.
- ثم فرض/صياغة بنية نحوية على هذه القطع.
- بعدها تقوم مرحلة التحليل باستخدام البنية النحوية لتخلق/لتكون منها صيغة جديدة وسطية لبرنامج المصدر الأصلي وتعرف بالتمثيل الوسيط

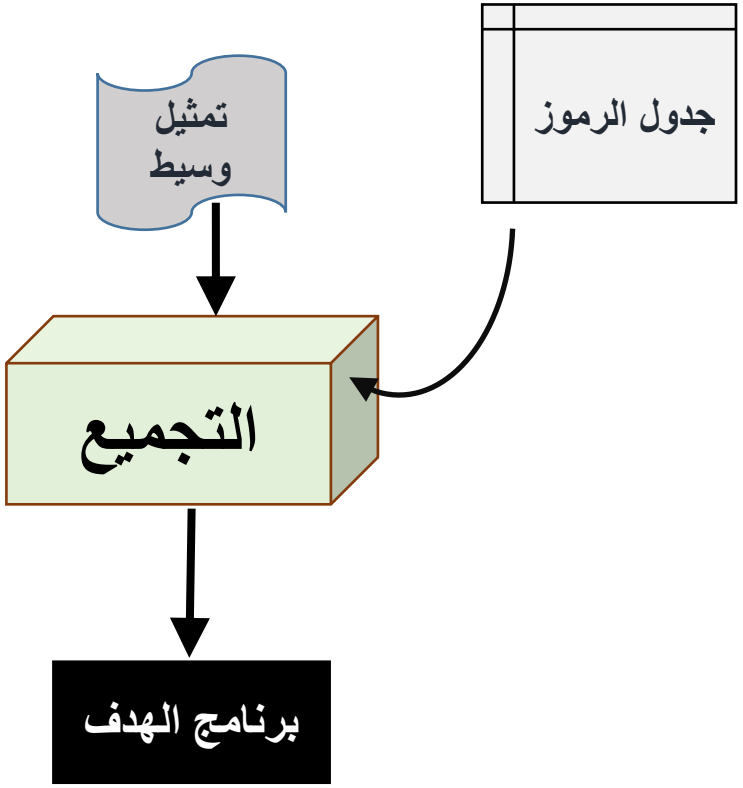
(Intermediate Representation ***IR***).

1- التحليل (Analysis)

- في حالة تكتشف مرحلة التحليل أن برنامج المصدر مركب/مصاغ بطريقة غير سليمة أو لا تعطي معناً منطقي/مفهوم, فإنه على هذه المرحلة توفير رسائل/تقارير معلوماتية توضح تلك الأخطاء ليستفيذ منها المستخدم/المبرمج لإجراء التصحيح اللازم.
- خلال مرحلة التحليل يتم تحصيل بعض المعلومات عن برنامج المصدر تخزن في جدول يسمى "جدول الرموز" (Symbol Table) الذي بدوره يمرر مع الصيغة الوسطية إلى المرحلة التالية وهى التجميع (Synthesis).

2- التجميع (Synthesis)

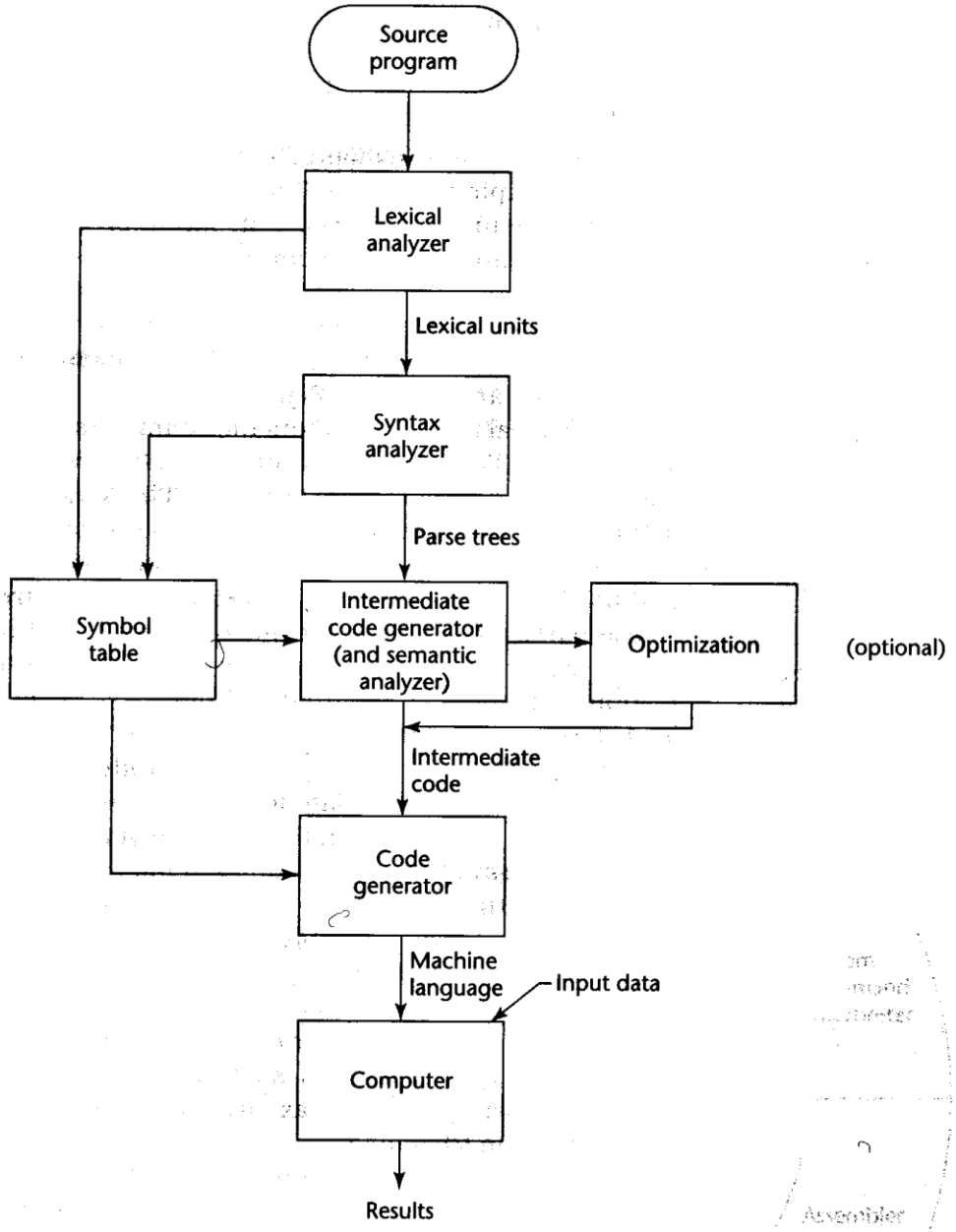
- مرحلة التجميع تقوم ببناء برنامج الهدف المطلوب من خلال التمثيل الوسيط وجدول الرموز.



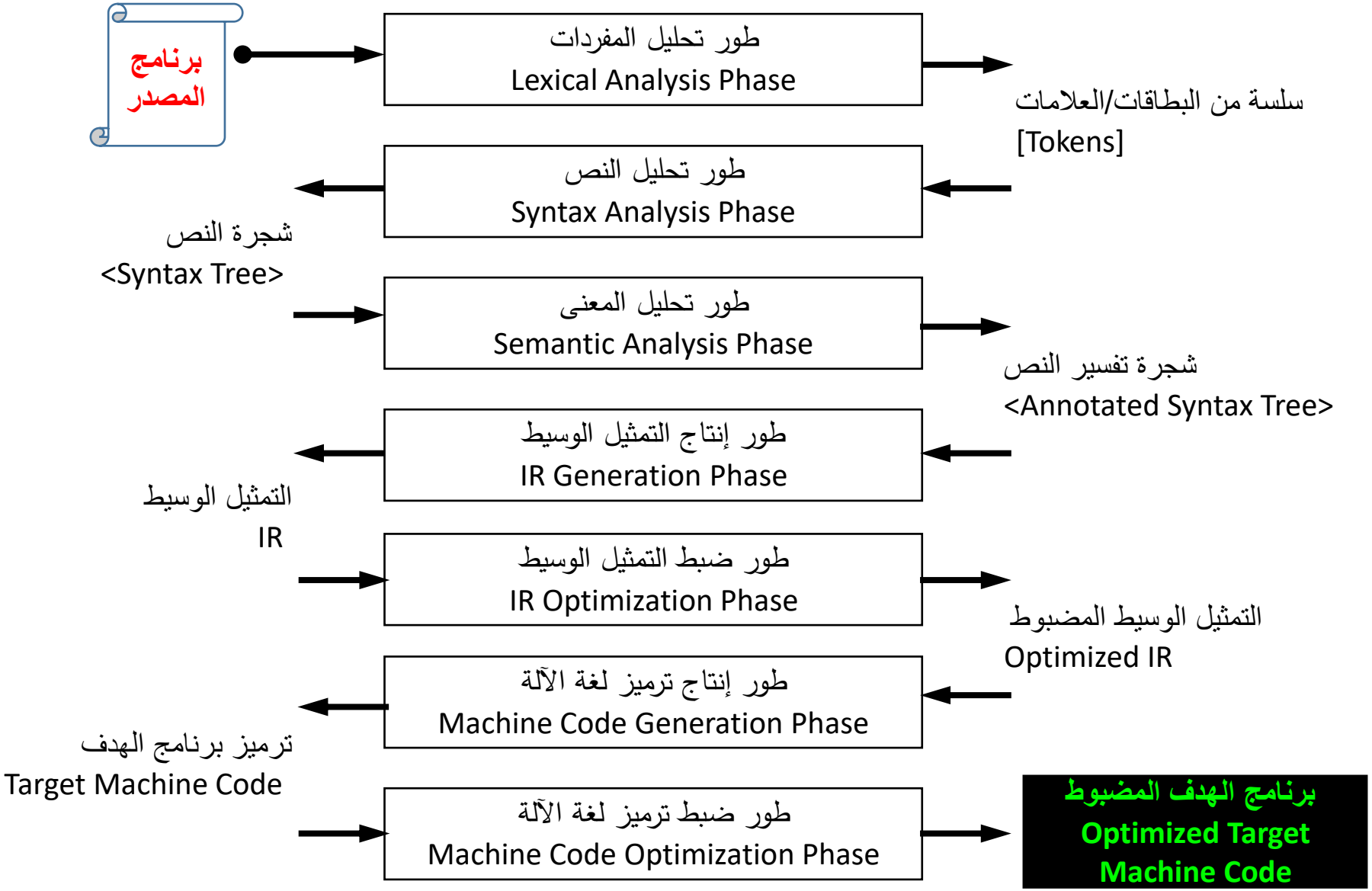
أطوار عملية الترجمة

- عملية ترجمة البرنامج المصدري إلى البرنامج المستهدف تمر بعدد من الأطوار خلال كل مرحلة من مرحلتي التحليل والتجميع. أي أنه كل من المرحلتين تقسم إلى مراحل داخلية تعرف بالأطوار (Phases).
- كل طور يعمل على نقل/تغيير برنامج المصدر من تمثيل إلى آخر (من صيغة إلى أخرى).
- عملياً, بعض الأطوار يمكن أن تدمج مع بعض في طور واحد.
- جدول الرموز الذي يحتوي على معلومات عن البرنامج المصدري يمكن أن يستخدم من قبل أي طور أثناء عملية الترجمة.

Figure 1.3
The compilation process



أطوار عملية الترجمة



طور تحليل المفردات (التمشيط Scanning)

- يقوم محلل المفردات بقراءة سيل الرموز Characters (الحروف, العلامات, والأرقام) التي تشكل التعليمات المكونة لبرنامج المصدر بالتسلسل من اليسار إلى اليمين.
- يتم تجميع الرموز على هيئة بطاقات/علامات (Tokens) ذات معنى مشترك.
- **مثال لهذه البطاقات:**
  - المعرفات/المتغيرات التي يعرفها المبرمج (*identifiers*)
  - الكلمات المحجوزة: integer, float, double, cin, scanf, …
  - المشغلات (*operators*): +, -, *, /, =, <, >, AND, OR, XOR, …
  - الرموز الخاصة: (,), ;, “, {, }, &, $, …

عملية تحليل المفردات

- الشكل العام للبطاقات/للعلامات يكون كاتالي:

<قيمة الصفة, اسم العلامة>

<token-name, attribute-value>

- اسم العلامة: رمز يبين نوع العلامة ويستفاذ منه في طور تحليل النص.
- قيمة الصفة: يشير إلى قيمة بجدول الرموز تخص هذه العلامة Token.
- معلومات القيم بجدول الرموز تستخدم في مرحلة تحليل المعنى وإنتاج ترميز لغة الآلة.

مثال لعملية تحليل المفردات

a = a + 2; → محلل المفردات Lexical Analyser →

<T_IDENTIFIER, 1> (اسم متغير)

<T_OPERATOR, => (مشغل بقيمة "=")

<T_IDENTIFIER, 1> (اسم متغير)

<T_OPERATOR, + > (مشغل بقيمة " + ")

<T_INTCONSTANT, 2> (ثابت صحيح بقيمة "2")

<T_SPECIAL, ;> (رمز خاص بقيمة ";")

ويمكن تبسيطه إلى: <id, 1> <=> <id, 1> <+ > <2> <;>

مثال لعملية تحليل المفردات

مدخلات تحليل المفردات:

```
position = initial + rate * 60 ;
```

مخرجات تحليل المفردات؟

<id, 1> <=> <id, 2> <+ > <id, 3> <*> <60> <;>

جدول الرموز:

ID	name	Type	value
1	position	float	…
2	initial	float	…
3	rate	float	…

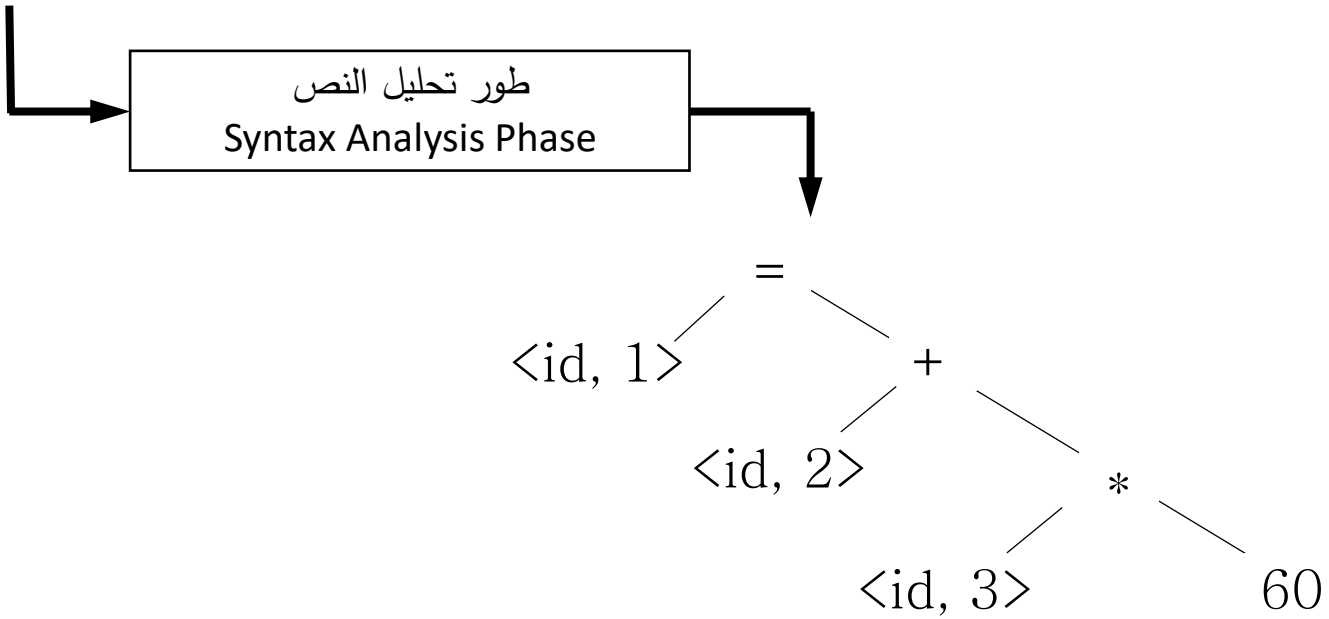
تُهمل الفراغات التي تفصل بين الكلمات والمتغيرات.

طور تحليل النص (الإعراب Parsing)

- العلامات/الأدلة/البطاقات (Tokens) التي تم تجميعها من عملية تمشيط/تحليل المفردات لم تخضع لعملية مراجعة القواعد النحوية للغة البرمجة المستعملة.
- في عملية التحليل النصي يتم ربط كل دليل (Token) بقاعدة لغوية محددة وتوضع المحصلة في مجموعة واحدة. تسمى هذه العملية بالإعراب (Parsing).
- مخرجات هذا الطور تسمى بشجرة النص أو شجرة الإعراب, أي سجل القواعد النحوية التي استخدمت لتكوين برنامج المصدر.

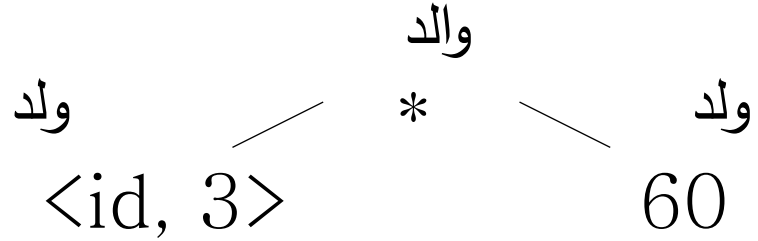
مثال لعملية تحليل النص/الإعراب

<id, 1> <=> <id, 2> <+ > <id, 3> <*> <60> <;>



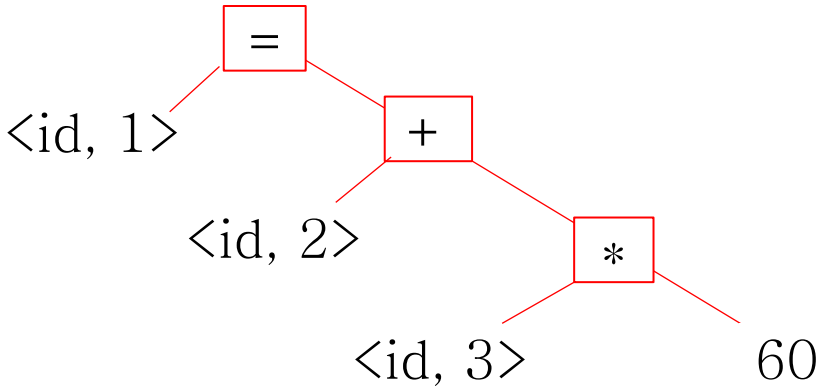
طور تحليل النص (الإعراب Parsing)

- شجرة النص السابقة تعرض الترتيب الذي يجب أن تنفذ به العمليات (* ,+ ,=).
- تتكون الشجرة من عُقل/عقد (nodes), تبدأ بالعقلة الجدر root ثم تتفرع إلى العقل الداخلية, وكل عقلة تسمى والد parent يتفرع منها ولدان children واحد عن اليمين والآخر عن اليسار.



طور تحليل النص (الإعراب Parsing)

- أقصى عقلة داخلية <*> تفيد بأن عملية الضرب ستنفذ أولاً بضرب المتغير rate في الثابت 60. وتنتقل العمليات حتى تصل إلى عقلة الجدر <=> التي تفيد بأن ناتج عملية الجمع الناجمة من عقلة الولد الأيمن <+> ستخزن في المتغير position الممثل بالولد الأيسر <id, 1>.



- هذا الترتيب في تنفيذ العمليات يتوافق مع الترتيب التقليدي للعمليات الحسابية الذي يعطي مشغل الضرب أولوية عن مشغل الجمع.
- الأطوار المتتالية للمترجم تستخدم التركيب النحوي في تحليل برنامج المصدر وفي إنتاج برنامج الهدف.

طور تحليل المعنى Semantic Analysis

- محلل المعنى يستخدم شجرة النص/الإعراب مع المعلومات بجدول الرموز لاختبار ترابط/صحة برنامج المصدر بالنسبة لقواعد لغة البرمجة.
- يتم كشف الأخطاء اللغوية كالتعليمات التي تتكون من مفردات صحيحة ولكن لا تتقيد بقواعد صياغة التعليمات الخاصة بلغة البرمجة.
  - ✓ كشف المتغيرات المستخدمة ولم يتم تعريفها.
  - ✓ دالة/وظيفة تم استدعائها مع خطأ في عدد أو نوع مدخلاتها أومخرجاتها.
  - ✓ عدم تجانس المتغيرات Type mismatches.

```
int  arr[2], c;
c = arr * 10;        // arr اسم مصفوفة و10 عدد صحيح
```

طور تحليل المعنى Semantic Analysis

- تجميع معلومات عن أنواع المتغيرات وحفظها إما في شجرة النص أو جدول الرموز للاستفاذة منها أثناء بناء التمثيل الوسيط IR.
- من أهم الوظائف هو اختبار أنواع المتغيرات المستعملة في بناء التعليمات.
- يعمل المترجم على اختبار كل مشغل Operator ومدى تجانس عوامله Operands.

إظهار رسالة خطأ لتصحيح دليل المصفوفة `char name[10.5];`

من عدد حقيقي إلى عدد صحيح.

طور تحليل المعنى Semantic Analysis

- قد تفرض إعدادات/قواعد اللغة تغيير بعض أنواع المتغيرات في حالة عدم تجانسها وتسمى هذه العملية بالإجبار Coercion.
- مثلاً

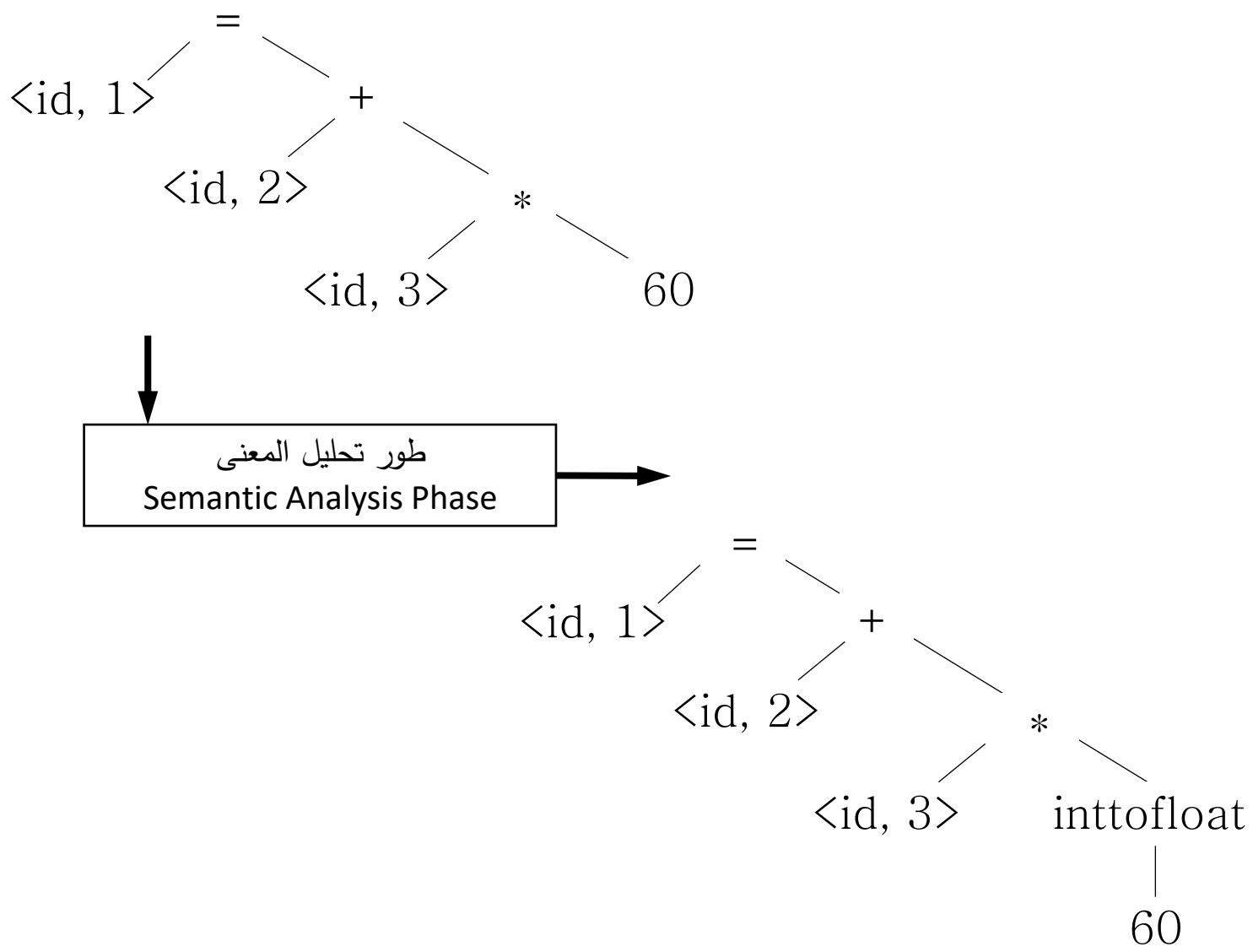
```
float  x, y;
y = x * 90;     // 90 عدد صحيح
```

يلاحظ المترجم أن المشغل * تم استخدامه مع عدد حقيقي وآخر صحيح فيقوم بتبديل 90 إلى عدد حقيقي 90.0 باستخدم أمر مثل inttofloat. **لمـــــــــاذا هذا؟**

طور تحليل النص السابق اهتم بتركيب مفردات الجملة وليس بمعناها/منطقها.

- متغير = متغير + ثابت لايهتم بنوع العناصر

مثال لعملية تحليل المعنى



طور إنتاج التمثيل الوسيط IR

- هذا الطور يستخدم شجرة المعنى لإنتاج التمثيل الوسيط لبرنامج المصدر.
- يفترض بهذا التمثيل أن يكون:
  1. سهل الإنتاج
  2. وسهل التحويل إلى برنامج الهدف.
- يعتبر قريب من تمثيل لغة الآلة.
- هناك صيغ عديدة للتمثيل الوسيط ولكن أكثرها استخداماً يعرف بـ

ترميز العنوان الثلاثي

Three – Address Code (TAC)

ترميز العنوان الثلاثي TAC

- وهي تشبه إلى حد كبير لغة التجميع.
- تتألف من تسلسل التعليمات وكل أمر/تعليمة لايتجاوز ثلاث عوامل Operands

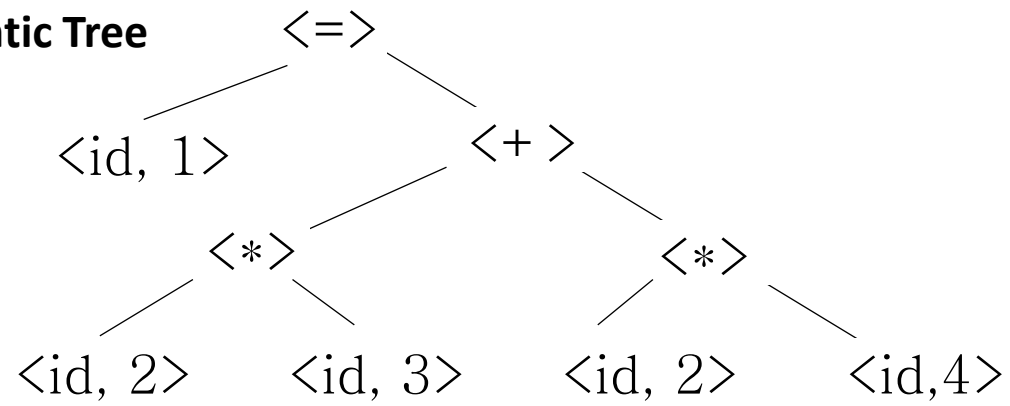
مثال:

```
a = b * c + b * d    //جملة من خمس عوامل
```

Tokens:

<id, 1> <=> <id, 2> <*> <id, 3> <+> <id, 2> <*> <id, 4>

Semantic Tree



ترميز العنوان الثلاثي TAC

- وهي تشبه إلى حد كبير لغة التجميع.
- تتألف من تسلسل التعليمات وكل أمر/تعليمة لايتجاوز ثلاث عوامل Operands

مثال:

```
a = b * c + b * d    //جملة من خمس عوامل
```

ترميز العنوان الثلاثي

```
_t1 = <id,2> * <id,3>
_t2 = <id,2> * <id,4>
_t3 = _t1 + _t2
<id,1> = _t3
```

- يعتبر كل عامل كأنه أحد المسجلات Registers
- كما تستخدم متغيرات variables مؤقتة temporary توصف بالحرف t أو _t تستخدم لخفظ عدد العوامل إلى ثلاثة.

ترميز العنوان الثلاثي TAC

- هنالك تفاصيل أكثر: كالتي تتعلق بأوامر الشرط (التفرع) والتكرار واستدعاء الوظائف/الدوال.

مثال:

```
if (a <= b)
    a = a - c;
b = b * c;
```

ترميز العنوان الثلاثي: حلين

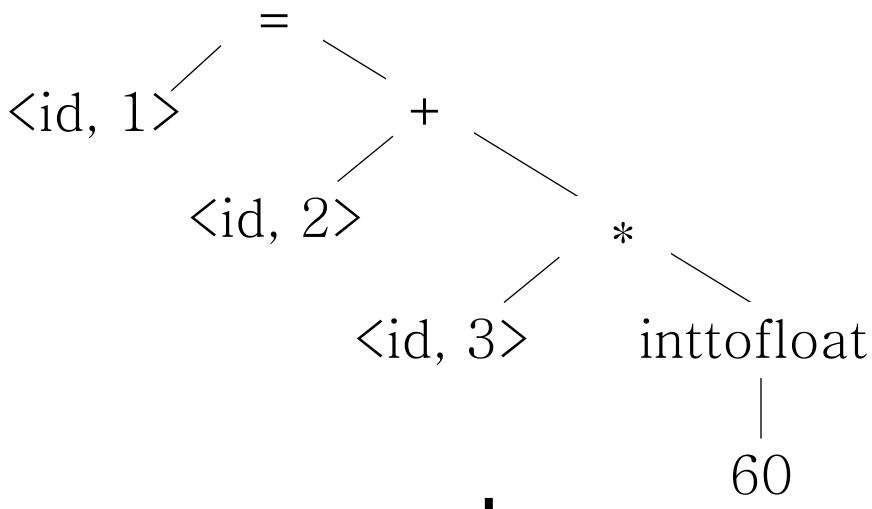
```
    _t1 = a > b
    if _t1 goto L
    _t2 = a - c
    a = _t2
L0: _t3 = b * c
    b = _t3
```

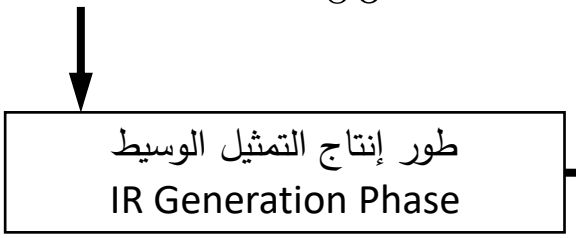
```
    _t1 = a < b
    _t2 = a == b
    _t3 = _t1 OR _t2
    if !_t3 goto L0
    _t4 = a - c
    a = _t4
L0: _t5 = b * c
    b = _t5
```

صفات ترميز العنوان الثلاثي

1. كل تعليمة لا تحتوي على أكثر من مشغل واحد Operator في الطرف الأيمن.
2. لا تحتوي التعليمة على أكثر من ثلاث عوامل Operands.
3. ترتيب التلعيمات يكون حسب أولوية التنفيذ (مثلاً: الضرب يسبق الجمع).
4. على المترجم استخدام أسماء مؤقتة لحفظ القيم الحسابية.
5. قد تحتوي تعليمات العنوان الثلاثي على أقل من ثلاث عوامل.

مثال لطور إنتاج IR





?

t1 = i2r(60)

t2 = id3 * t1

t3 = id2 + t2

id1 = t3