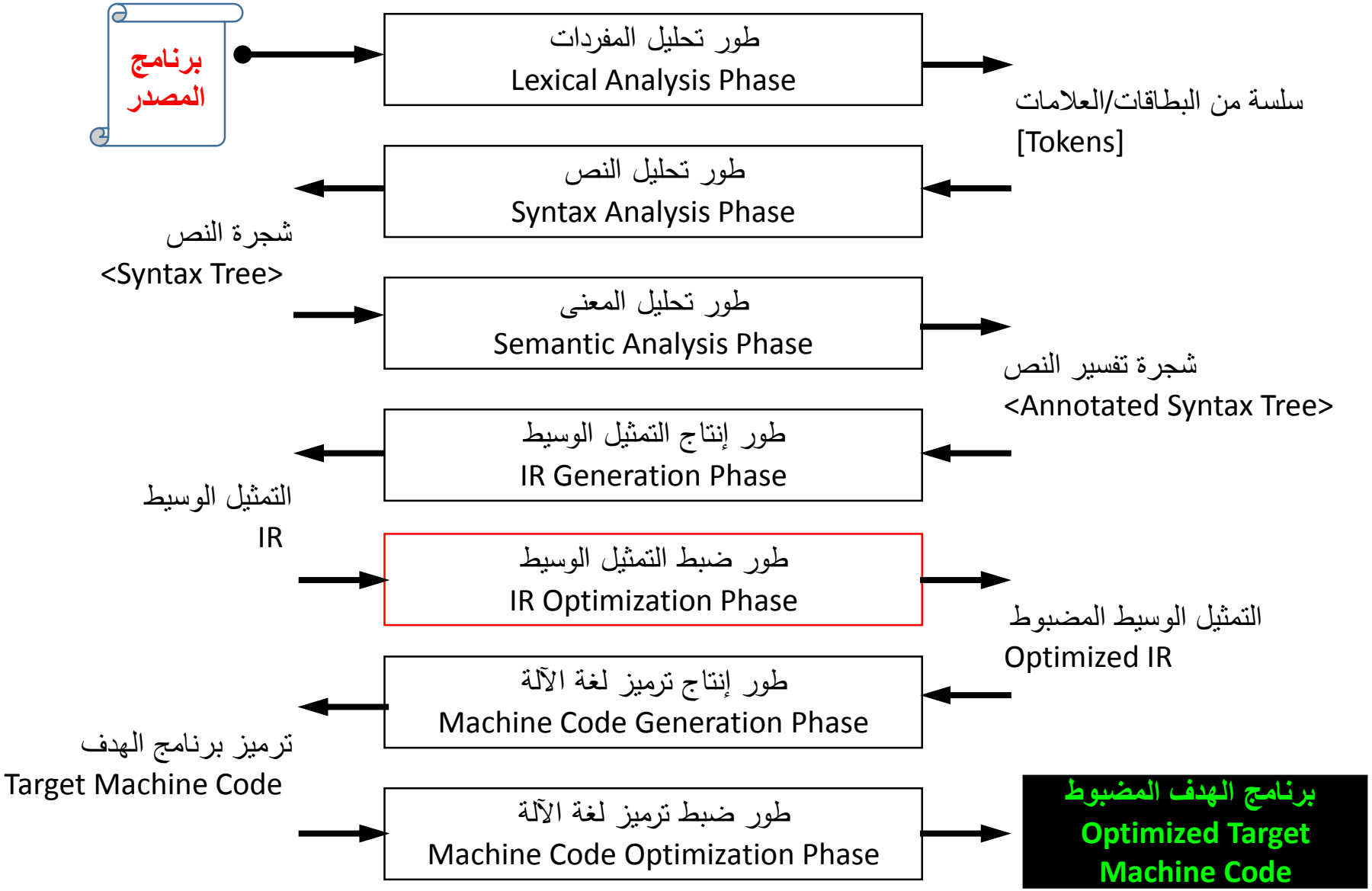
عملية الترجمة

Compilation Process

Continue

أطوار عملية الترجمة



طور ضبط التمثيل الوسيط

- هذا الطور هو بداية الطرف الخلفي وهو مرحلة التجميع/التأليف Synthesis.
- يستقبل تمثيل وسيط وينتج أيضاً تمثيل وسيط ولكن محسن/مضبوط
- هنا يقوم المترجم بتحويرات مختلفة على التمثيل الوسيط لغرض تحسينه وبالتالي نتحصل في النهاية على ترميز لغة الآلة سريع التنفيذ.
- **يهدف هذا الطور إلى:**
  - أبسط قدر من التعليمات
  - الأسرع تنفيذاً
  - تؤدي إلى أدق النتائج

بعض عمليات ضبط التمثيل الوسيط

- خمد/إلغاء إنتاج ترميز/تعليمات للأوامر التي لن تنفذ كالتي تكون في جمل شرطية زائدة أو لا يمكن أن تتحقق.

```
    x = 0;
    if x>0 goto L8;
    Y = 2 * x;
L8: M = Y * x;
```

- حذف المتغيرات غير المستخدمة
- التخلي عن عمليات كالضرب في 1 أو الإضافة إلى 0.

بعض عمليات ضبط التمثيل الوسيط

- التخلي عن عمليات كالضرب في 1 أو الإضافة إلى 0.
- ضبط التكرارات Looping, مثل استثناء التعليمات التي لا تتأثر من التكرار.

```
int value;
for (int i = 0; i < 5; i++) {
        value = 500;
        System.out.println(i, "Value: ", value);
}
```

- الحد من التعبيرات/التعليمات المتكررة

```
a = b + 1;
a = b + 1;
```

مثال ضبط التمثيل الوسيط

```
_t1 = id2 * id3
_t2 = _t1 +  0
_t3 = id2 * id3
_t4 = _t3 +  _t2
id1 = _t4
```

بعد الضبط

- اختصار خمس تعبيرات إلى ثلاث
  - حذف الإضافة للصفر
  - إلغاء التعليمات المتكررة

```
_t1 = id2 * id3
_t2 = _t1 +  _t1
id1 = _t2
```

مثال ضبط التمثيل الوسيط

- مراجعة التمثيل:

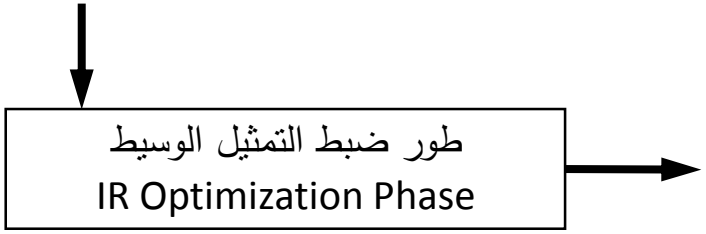
يمكن لضابط التمثيل أن يستبدل عملية تغيير العدد الصحيح إلى حقيقي inttofloat بأن يغير الرقم 60 إلى 60.0 حقيقي لمرة واحدة وبصورة نهائية بدلاً من استدعاء وظيفة inttofloat كلما استخدم الرقم 60

```
t1 = i2r(60)
t2 = id3 * t1
t3 = id2 +  t2
id1 = t3
```

```
t1 = id3 * 60.0
t2 = id2 +  t1
id1 = t2
```

مثال ضبط IR

```
t1 = i2r(60)
t2 = id3 * t1
t3 = id2 +  t2
id1 = t3
```




```
t1 = id3 * 60.0
t2 = id2 +  t1
id1 = t2
```

مقايضة السرعة بالكفاءة

- قد ينجم عن عمليات هذا الطور بطؤ عملية الترجمة
- توفر المترجمات إمكانية استثناء/إيقاف عملية ضبط التمثيل الوسيط.
- بعض المترجمات توفر إمكانية اختيار دقة/درجة الجودة في ضبط تمثيل IR.
- برنامج بطئ الترجمة ولكن سريع التنفيذ وقليل التعليمات
  - أقل استنزاف لوقت المعالج
  - أقل استهلاك للذاكرة
- **كم من مرة سيترجم البرنامج وكم مرة سينفذ؟**

طور إنتاج ترميز لغة الآلة

- العملية النهائية للحصول على البرنامج الهدف هو طور إنتاج ترميز لغة الآلة.
- تعبيرات ترميز العنوان الثلاثي تترجم إلى تعليمات لغة تجميع أو لغة الآلة.
- في هذه العملية يقوم المترجم بـ:
  - تحديد مكان في الذاكرة لكل متغير Variable
- كل تعبير في التمثيل الوسيط يحول إلى تعبير بلغة الآلة يؤدي نفس الوظيفة.

مثال طور إنتاج برنامج الهدف

- تمثيل وسيط بترميز العنوان الثلاثي.

```
_t1 = id3 * id2
_t2 = _t1 +  _t1
id5 = _t2
```

- ترميز لغة التجميع
  - العامل إلى اليسار هو محطة حفظ القيمة
  - الحرف f للعدد الحقيقي floating point

```
Ldf R1, id3        # load
Ldf R2, id2        # load
Mulf R1, R1, R2    # mult
Addf R2, R1, R1    # add
Stf id5, R2        # store
```

مثال طور إنتاج برنامج الهدف

t1 = id3 * 60.0

id1 = id2 + t1


```
Ldf R2, id3
Mulf R2, R2, #60.0
Ldf R1, id2
Addf R1, R1, R2
Stf id1, R1
```

طور إنتاج ترميز برنامج الهدف

- التعبيرات في المثال السابق تقوم بتحميل المسجل R2 بمحتويات العنوان id3 الموجودة في جدول الرموز.
- يلي ذلك ضربها في الثابت 60.0 الذي يشار إليه بالعلامة # ليعرف كثابت.
- بعدها ينقل محتويات id2 إلى المسجل R1.
- ثم يجمع محتويات R1 مع R2 ويكون الناتج في المسجل R1.
- وفي النهاية يحفظ محتوى R1 في المُعرَّف id1.

طور ضبط ترميز برنامج الهدف

- يمكن أن يكون هنالك عملية ضبط أخرى تتم على ترميز برنامج الهدف.
- هنا يكون الاهتمام بنوعية الكيان المادي لتحقيق الاستخدام الأمثل للمعالج والمسجلات.
- أحياناً يوجد أكثر من معالج بالحاسوب
- بعض المعالجات خاصة بوظائف معينة كالعمليات الرياضية Math-coprocessor.
- بعض المسجلات خاصة بالعمليات الرياضية مثل Accumulator.
- صيغة التعليمات أو طريقة العنونة تختلف من معالج إلى آخر.

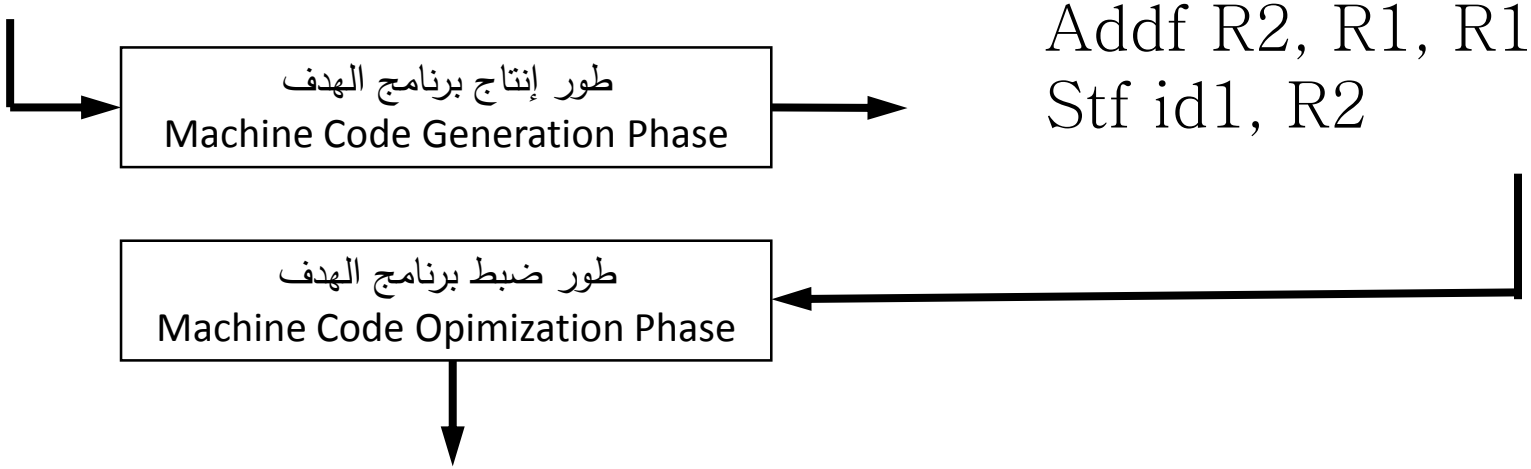
طور ضبط ترميز برنامج الهدف

- مثل طور ظبط IR قد توفر المترجمات إمكانية استثناء/إيقاف عملية ضبط برنامج الهدف.
- كذلك بعض المترجمات توفر إمكانية اختيار دقة/درجة الجودة في ضبط تمثيل البرنامج الهدف.

مثال ضبط برنامج الهدف

```
_t1 = id2 * id3
_t2 = _t1 + _t1
id1 = _t2
```

```
Ldf R1, id2          # load
Ldf R2, id3          # load
Mulf R1, R1, R2      # mult
Addf R2, R1, R1      # add
Stf id1, R2          # store
```



Intel Assembly Code

```
Ldf R1, id2
Ldf R2, id3
Mulf AC, R1, R2
Addf AC, R1    # Accumulator وحفظ النتيجة في المسجل المجمع 2 الاستغناء عن استدعاء المسجل
Stf id1, AC
```

موضوعنا التالي:

تفصيل طور تحليل المفردات