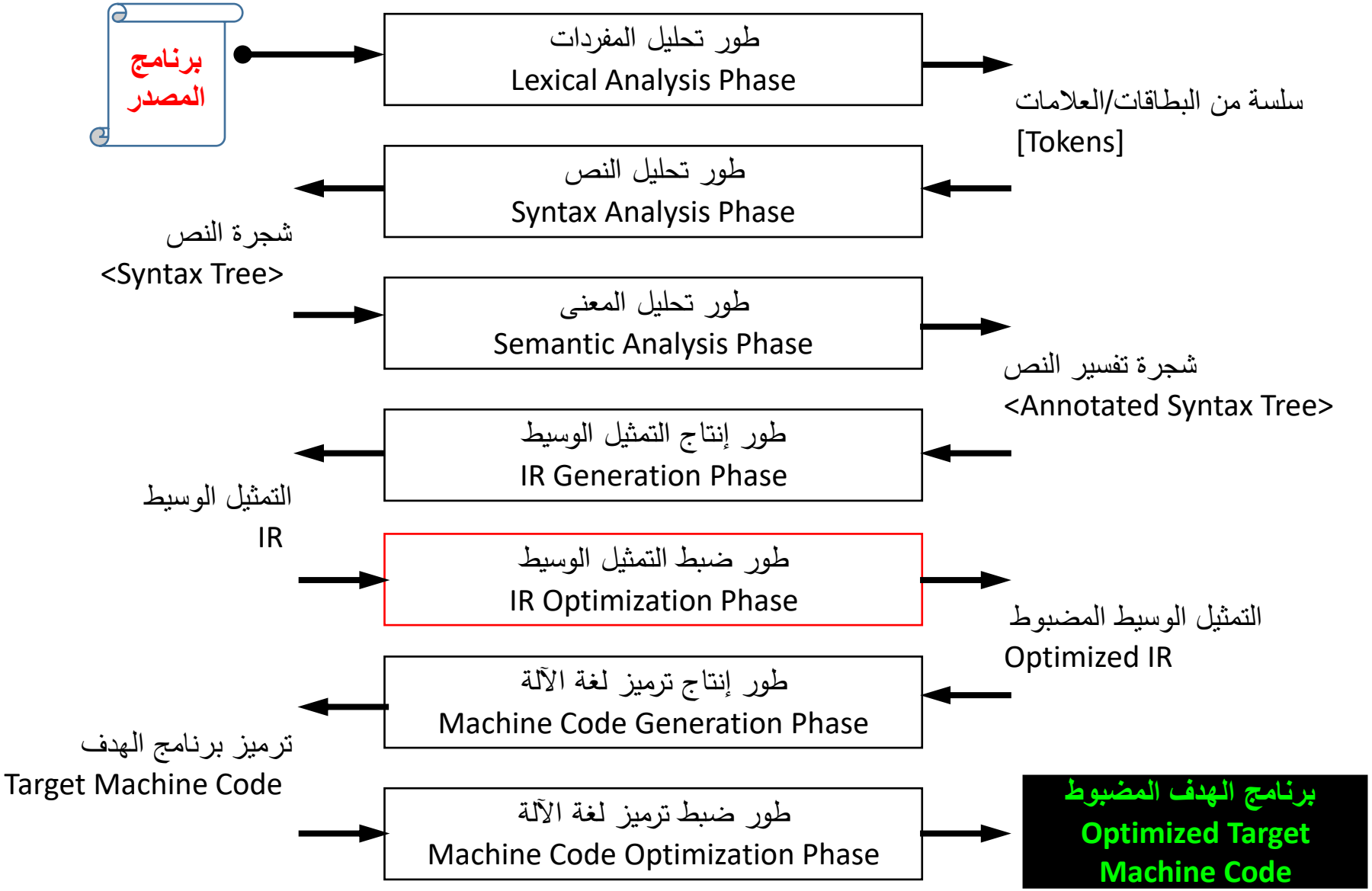
عملية الترجمة

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;
```

- حذف المتغيرات غير المستخدمة
- التخلي عن عمليات كالضرب في 1 أو الإضافة إلى 0.
- ضبط التكرارات Looping, مثل استثناء التعليمات التي لا تتأثر من التكرار.
- الحد من التعبيرات/التعليمات المتكررة

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

مثال ضبط التمثيل الوسيط

```
_t1 = b * c
_t2 = _t1 + 0
_t3 = b * c
_t4 = _t3 + _t2
a = _t4
```

بعد الضبط

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

```
_t1 = b * c
_t2 = _t1 + _t1
a = _t2
```

مثال ضبط التمثيل الوسيط

- مراجعة التمثيل:

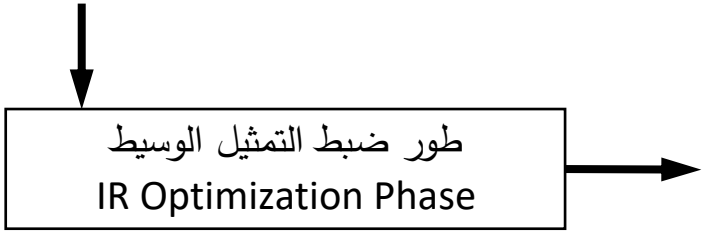
يمكن لضابط التمثيل أن يستبدل عملية تغيير العدد الصحيح إلى حقيقي inttofloat بأن يغير الرقم 60 إلى 60.0 حقيقي لمرة واحدة وبصورة نهائية بدلاً من استدعاء وظيفة inttofloat كلما استخدم الرقم 60

```
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
```

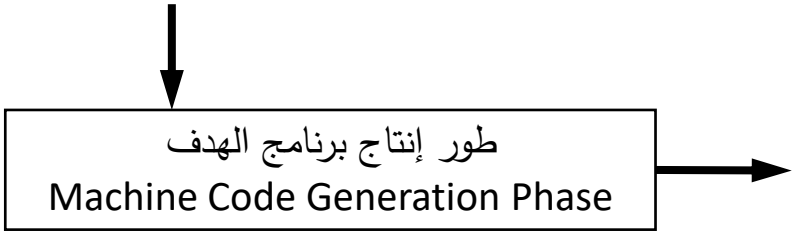
- ترميز لغة التجميع
  - العامل إلى اليسار هو محطة حفظ القيمة
  - الحرف f للعدد الحقيقي floating point

```
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

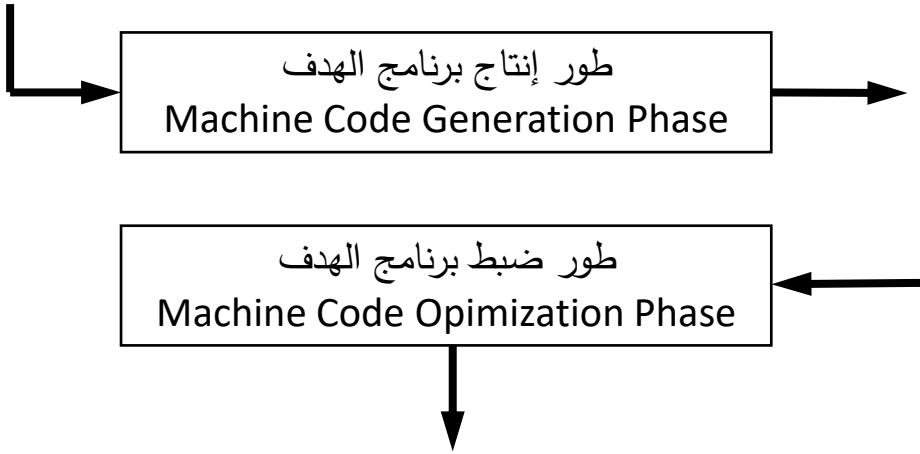
- يمكن أن يكون هنالك عملية ضبط أخرى تتم على ترميز برنامج الهدف.
- هنا يكون الاهتمام بنوعية الكيان المادي لتحقيق الاستخدام الأمثل للمعالج والمسجلات.
- أحياناً يوجد أكثر من معالج بالحاسوب
- بعض المعالجات خاصة بوظائف معينة كالعمليات الرياضية Math-coprocessor.
- بعض المسجلات خاصة بالعمليات الرياضية مثل Accumulator.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

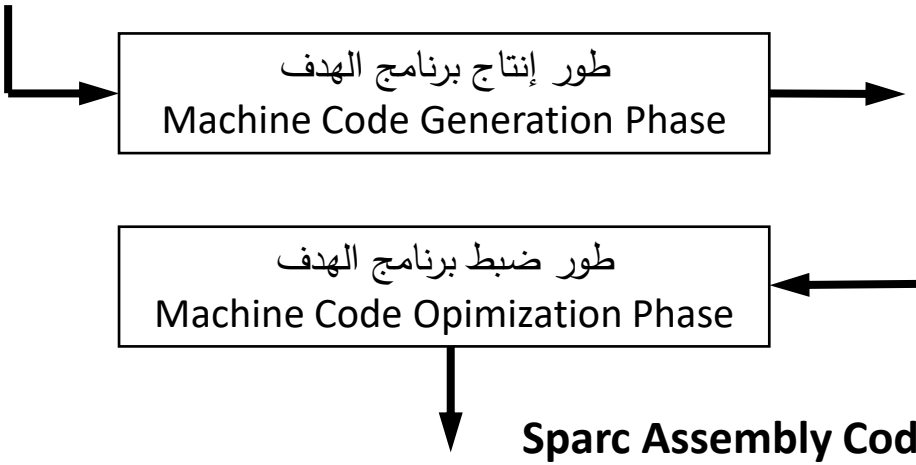
Intel Assembly Code

```
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
```

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
```



Sparc Assembly Code

```
ld    [%fp-16], %l1    تحميل عدد حقيقي من العنوان 16 إلى المسجل 1
ld    [%fp-20], %l2
mul %l1, %l2, %l3    ضرب المسجل 1 في المسجل2 وحفظ النتيجة في المسجل 3
add  %l3, %l3, %l0
st    %l0, [%fp-24]    حفظ محتوى المسجل 10 في موقع الذاكرة 24 لعدد حقيقي
```

مثال للغة الآلة من لغة التجميع

XOR CL, [12H]

- العملية هي **أو الاستثنائية** XOR, و CL عبارة عن مسجل في المعالج, و12H هو عنوان مكان متغير في الذاكرة.
- رمز العملية XOR بلغة الآلة هو "001100dw"
- d تشير إلى مكان حفظ نتيجة العملية, 1 تعني حفظ النتجية في مكان المعامل الأول وهو المسجل CL, و 0 حفظ النتيجة في مكان المعامل الثاني لهذه العملية.
- w تشير لحجم البيانات: 0 تعني بايت واحد, و 1 تعني كلمة كاملة (طول الكلمة يعتمد على المعالج).

مثال للغة الآلة من لغة التجميع

XOR CL, [12H]

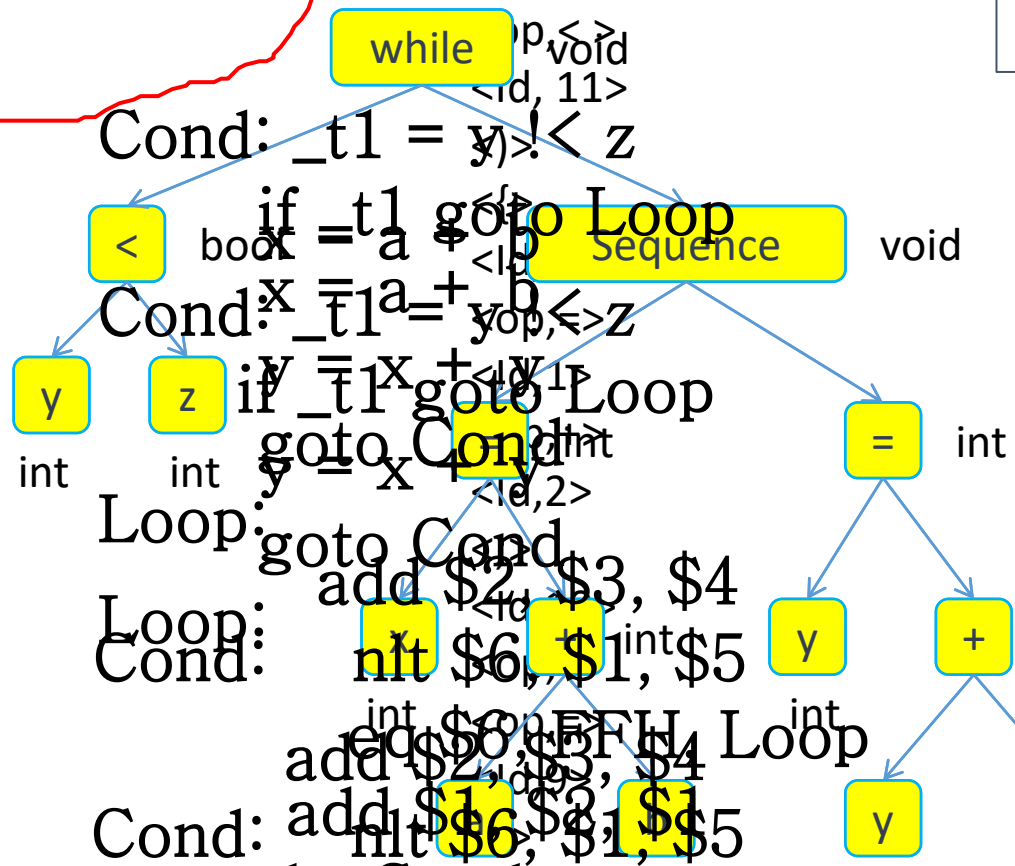
- العملية هي **أو الاستثنائية** XOR, و CL عبارة عن مسجل في المعالج, و12H هو عنوان مكان متغير في الذاكرة.
- رمز العملية XOR بلغة الآلة هو "001100dw"
- إذاً البايت الأول XOR بلغة الآلة هو "00110010"
- رموز المسجل CL هو 00001110
- عنوان مكان الذاكرة 12H هو00010010
- **هناك تفاصيل كثيرة في تركيبة التعليمات والعوامل وتختلف باختلاف المعالجات**

XOR CL, [12H] = 00110010 00001110 00010010

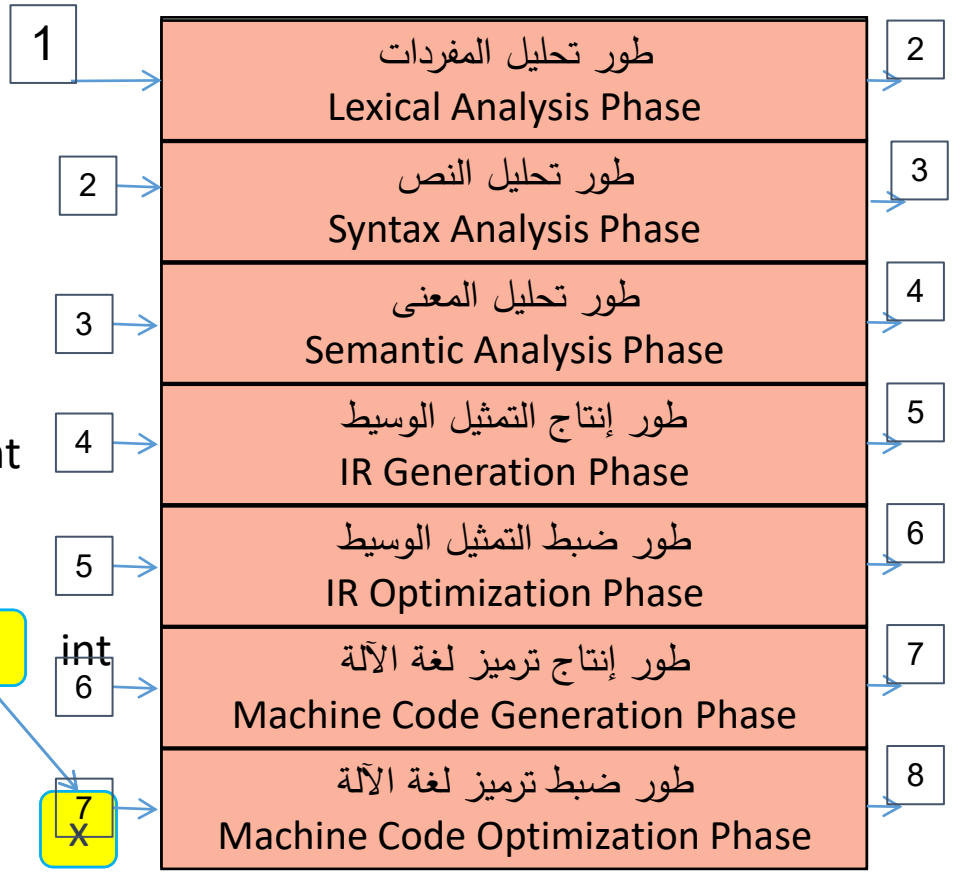
32H 0EH 12H = Hexadecimal 16 ويكتب بنظام

مراجعة

```
while(y < z) {
    x = a + b;
    y += x;
}
```




```
Cond: _t1 = y < z
      if _t1 goto Loop
      x = a + b
      y = x + y
Cond: _t1 = y < z
      if _t1 goto Loop
      goto Cond
Loop: goto Cond
Loop: add $2, $3, $4
Cond: nlt $6, $1, $5
      eq $6, FFH, Loop
      add $2, $3, $4
      add $1, $2, $1
      nlt $6, $1, $5
      br Cond, FFH, Loop
Loop: add $1, $2, Cond
Loop:
```



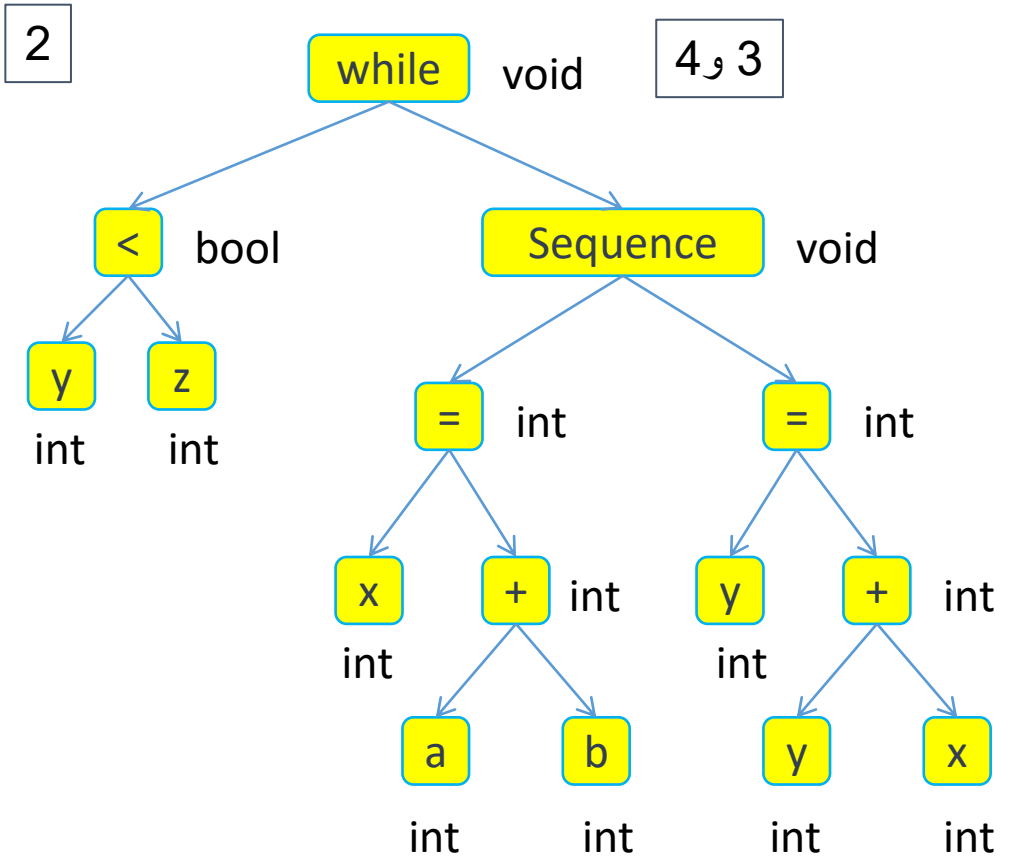
1

```
while(y < z) {
      x = a +  b;
       y + = x;
}
```

2

```
<While>
<(>
<Id, 10>
<op,< >
<Id, 11>
<)>
<{>
<Id,9>
<op,=>
<Id,1>
<op,+>
<Id,2>
<;>
<Id,10>
<op,+>
<op,=>
<Id,9>
<;>
<}>
```

3 و 4



5

```
Cond: _t1 = y !< z
        if _t1 goto Loop
        x = a +  b
        y = x +  y
        goto Cond
Loop:
```

6

```
        x = a +  b
Cond: _t1 = y !< z
       if _t1 goto Loop
        y = x +  y
        goto Cond
Loop:
```

7

```
        add $2, $3, $4
Cond:   nlt $6, $1, $5
        eq $6, FFH, Loop
        add $1, $2, $1
        br Cond
Loop:
```

8

```
        add $2, $3, $4
Cond:   nlt $6, $1, $5
        eq $6, FFH, Loop
        add $1, $2, Cond
Loop:
```