الجلسة الرابعة عشر

توليد الشيفرة الوسيطية (2)

Intermediate Code Generation (2)

الهدف من الجلسة

- تعريف الطالب بكيفية توليد الشيفرة الوسيطية مكتوبة بلغة التجميع (Assembly).

مستلزمات الجلسة

- حاسب بمواصفات دنيا RAM: 1 GB, CPU: 1.6 GHz, Windows 7 OS 32 bit
- Turbo c++
- LEX & BISON tools
- مراجعة لعمليات التحميل والتخزين والإزاحة والعمليات الحسابية والمنطقية للغة التجميع.

خطوات العمل

- توليد الشيفرة الوسيطية (بلغة التجميع)
- كيفية توليد الشيفرة المبدئية
- توليد شيفرة لعمليات التصريح
- توليد الشيفرة الوسيطية لعمليات الإسناد

الخلاصة والنتائج:

يفترض عند نهاية الجلسة:

- تمكن الطالب من تطبيق مثال عملي عن كيفية توليد الشيفرة الوسيطية لعمليات التصريح.

تمكن الطالب من تطبيق مثال عملي عن كيفية توليد الشيفرة الوسيطية لعمليات الإسناد.

1.1 توليد الشيفرة الوسيطية (تتمة):

1.1.1 توليد شيفرة لعمليات الإسناد:

عند إسناد قيمة من النوع الصحيح لرمز ما يتم نسخ هذه القيمة إلى قسم البيانات في الذاكرة إلى موقع وسيط ثم يتم نسخ القيمة من الموقع الوسيط إلى الموقع الهدف (الرمز الهدف).

```
ID EQ NUM SEMICOLON{
.....
//
تعليمات خاصة بتحليل المعاني
sprintf (strtemp1, "%d", $3);// strtemp1 تخزين قيمة الرقم الصحيح في المتغير
Add_Code("mov ");
Add_Code("al ,");
Add_Code(strtemp1);
Add_Code("\r\n");
Add_Code("mov ");
Add_Code( $1);
Add_Code(" ,al ");
Add_Code("\r\n");}
```

mov al,الرقم الصحيح

mov الرمز,al

1.1.2 حالة إسناد محرف لرمز ما:

في حالة نسخ محرف إلى رمز ما يتم نسخ قيمة المحرف إلى مسجل وسيط al مثلا، ثم يتم نسخ قيمة المسجل إلى موقع الذاكرة الذي يعنون الرمز المطلوب وذلك وفقاً للتعليمات التالية التي ستقوم بتوليد شيفرة التجميع لهكذا حالة.

```
ID EQ CHAR SEMICOLON{.....
Add_Code("mov al, ");
Add_Code($3);
Add_Code("\r\n");
Add_Code("mov ");
Add_Code( $1);
Add_Code(" ,al ");
Add_Code("\r\n");}
```

حالة إسناد سلسلة لرمز ما:

عند إسناد سلسلة مؤلفة من عدد من المحارف إلى رمز ما، فإن عملية النسخ ستتم بشكل متتال بحيث ينسخ كل محرف إلى موقع واحد من المواقع المخصصة للرمز المعرف على أنه من نوع السلسلة.

فلو أردنا أن ننسخ العبارة "abc" إلى الرمز k الذي تم التصريح عنه مسبقاً والذي حجزنا له 50 بايت في الذاكرة معنونة بالعنوان k، حيث يتم نسخ المحرف a إلى الموقع الأول، ثم المحرف b إلى الموقع الثاني، ثم المحرف c إلى الموقع الثالث.

إذا الفكرة هي استخدام حلقة متكررة تقوم كل مرة بنسخ محرف واحد إلى موقع واحد، ثم زيادة المؤشر مرة واحدة كل تكرار حتى انتهاء محارف السلسلة.

```
|ID EQ CHAIN SEMICOLON{.....
strtemp = GenStrIdent();//توليد رمز سلسلة جديد
Add_Data(strtemp);
Add_Data("\tdb\t");
Add_Data($3);//إضافة الرمز إلى قسم البيانات من ملف الخرج
Add_Data("\r\n");//إضافة سطر جديد
Add_Code("mov di,offset ");
Add_Code($1);
Add_Code("\r\n");
Add_Code("mov si,offset ");
Add_Code(strtemp);
Add_Code("\r\n");
Add_Code("mov cx,[si+1]\r\n");
Add_Code(GenLabel("next"));
Add_Code(":\r\n");
Add_Code("\tmov al,[si]\r\n");
Add_Code("\tcmp al,0\r\n");
Add_Code("\tjz ");
Add_Code(GenLabel("end"));
Add_Code("\r\n");
Add_Code("\tmov byte ptr [di],al\r\n");
Add_Code("\tinc si\r\n");
Add_Code("\tinc di\r\n");
Add_Code("\tdec cx\r\n");
Add_Code("\tcmp cx,0\r\n");
Add_Code("\tjz ");
Add_Code(GenLabel("end"));
Add_Code("\r\n");
```

```
mov di,offset new
mov si,offset _msg0_
mov cx,[si+1]
next0:
        mov al,[si]
        cmp al,0
        jz end0
        mov byte ptr [di],al
        inc si
        inc di
        dec cx
        cmp cx,0
        jz end0
```

```
Add_Code("\tloop ");
Add_Code(GenLabel("next"));
Add_Code("\r\n");
Add_Code(GenLabel("end"));
Add_Code(":\r\n");GenLabelIncCounter();}
```

تقوم التعليمات السابقة بتحويل عملية إسناد سلسلة لرمز من جدول الرموز إلى لغة التجميع، حيث يستخدم المتغير strtemp لحفظ اسم الرمز الوسيط الجديد وفقاً للتعليمة:

```
strtemp = GenStrIdent();
```

حيث يتم حجز موقع لهذا المتغير في ذاكرة الحاسب مماثل لحجم السلسلة، ثم يتم نسخ هذه السلسلة إلى الهدف. فمثلاً لو كانت السلسلة المصدرية المراد إسنادها إلى رمز ما هي abc وفق التعليمة المصدرية h="abc" سيتم إنشاء رمز جديد لدى مناداة التابع GenStrIdent وليكن _msg0_ ويتم إعطاؤه القيمة "abc" والتي تمثل السلسلة المصدرية، وسيصبح لدينا الإضافة التالية في قسم البيانات وهي:

```
"abc"       db          msg0_
```

بعد ذلك تجري عملية نسخ كل محرف من محارف السلسلة _msg0_ إلى الهدف والذي هو الرمز المطلوب h.

1.1.3 حالة إسناد رمز إلى رمز آخر:

تمثل هذه التعليمة إسناد قيمة رمز لرمز آخر مثل التعليمة g=k، ولتحويل هذه التعليمة للغة التجميع يجب نسخ قيمة الرمز المصدري إلى مسجل ما ثم نسخ قيمة هذا المسجل إلى موقع الذاكرة الذي يحتوي قيمة الرمز الهدف. لإنجاز ذلك يتم تعديل القاعدة الخاصة بإسناد رمز إلى رمز آخر بإضافة جزء الشيفرة التالي:

```
expression:
exp
|exp expression;
exp:
CON | LOOP
ID EQ ID SEMICOLON
{.....
//تعليمات خاصة بتحليل المعاني
Add_Code("mov al, ");
Add_Code($3);
Add_Code("\r\n");
Add_Code("mov ");
Add_Code( $1);
```

```
mov al, k
mov g ,al
```

Add_Code(" ,al ");
Add_Code("\r\n");}

تبقى هناك حالة واحدة وهي عند إسناد رمز من نوع السلسلة إلى رمز آخر من نوع السلسلة أيضاً مثل التعليمة o=new; حيث أن كل من o وَ new سلسلتان.

هنا يجب نسخ كل عنصر من السلسلة الأولى إلى العنصر العنصر المقابل له من الثانية، لذا يجب إضافة جزء الشيفرة التالي قبل الجزء السابق:

```
exp:ID EQ ID SEMICOLON {
.....// تعليمات خاصة بتحليل المعاني
if(get_sym_type($1)==4)
// التحقق من أن عملية الإسناد هي إسناد رمز من نوع سلسلة لرمز آخر
{Add_Code("mov di,offset ");
Add_Code($1);
Add_Code("\r\n");
Add_Code("mov si,offset ");
Add_Code($3);
Add_Code("\r\n");
Add_Code("mov cx,[si+1]\r\n");
Add_Code(GenLabel("next"));
Add_Code(":\r\n");
Add_Code("\tmov al,[si]\r\n");
Add_Code("\tcmp al,0\r\n");
Add_Code("\tjz ");
Add_Code(GenLabel("end"));
Add_Code("\r\n");
Add_Code("\tmov byte ptr [di],al\r\n");
Add_Code("\tinc si\r\n");
Add_Code("\tinc di\r\n");
Add_Code("\tdec cx\r\n")
;Add_Code("\tcmp cx,0\r\n");
Add_Code("\tjz ");
Add_Code(GenLabel("end"));
Add_Code("\r\n");
```

```
mov di,offset o
mov si,offset new
mov cx,[si+1]
```

```
...
next1:
        mov al,[si]
        cmp al,0
        jz end1
        mov byte ptr [di],al
        inc si
        inc di
        dec cx
        cmp cx,0
        jz end1
        loop next1
end1:
```

```
Add_Code("\tloop ");
Add_Code(GenLabel("next"));
Add_Code("\r\n");
Add_Code(GenLabel("end"));
Add_Code(":\r\n");
GenLabelIncCounter();}
```

وهذا المقطع يقوم بنسخ كل عنصر من السلسلة المصدرية إلى مقابله في الهدف.

بعد إجراء كل هذه التعديلات التي تم توضيحها يتم إعادة توليد المترجم من جديد لتوليد الشيفرة الوسيطية.

1.1.4 توليد شيفرة التجميع للعمليات الحسابية والمنطقية:

قد يتضمن الملف المصدري عبارات مثل ;a=b+c أو a=b/c أو a=b and c وغيرها الكثير من العبارات المماثلة، وللتعامل مع هذه العبارات يجب تعديل القواعد لتتلاءم مع هذه العمليات الحسابية والمنطقية والقواعد الواجب إضافتها هي:

```
exp:ID EQ ID PLUS ID SEMICOLON {
.....//تعليمات خاصة بتحليل المعاني
else{
//إضافة شيفرة التجميع (الأسمبلي) إلى ملف الخرج
Add_Code("\r\n");
Add_Code("\tmov al ");
Add_Code($3);
Add_Code("\r\n");
Add_Code("\tadd al ");
Add_Code($5);
Add_Code("\r\n");
Add_Code("\tadc ah,0 ");
Add_Code("\r\n");
Add_Code("\tmov ");
Add_Code($1);
Add_Code("\t ,ax ");}}
```

```
// x=x+f;
mov al, x
add al, f
adc ah,0
mov x ,ax
```

يتم في شيفرة الأسمبلي المضافة وضع قيمة الرمز الأول في مسجل al ثم يجري جمع قيمته مع الرمز الثاني، كما يتم الأخذ بعين الاعتبار وجود الحمل والذي سيتوضع في المسجل ah، ثم يتم وضع النتيجة في الرمز الهدف.

يعد الكود السابق صحيحاً في حال أن الرمزين من النوع الصحيح أو الحقيقي.

تكرر التعليمات السابق نفسها مع تعديل بنوع التعليمة في حالة الطرح والقسمة والضرب.

أما في حالة الضرب يتم تعديل التعليمات التالية:

```
exp:ID EQ ID PLUS ID SEMICOLON {
.....تعليمات خاصة بتحليل المعاني//
else
{
Add_Code("\tmov al, ");
Add_Code($3);
Add_Code("\r\n");
Add_Code("\tmov al, ");
Add_Code($5);
Add_Code("\r\n");
Add_Code("\tmul bl");
Add_Code("\r\n");
Add_Code("\tmov ");
Add_Code($1);
Add_Code("\t ,ax ");
```

```
// x=x*f;
mov al, x
mov bl,f
mul bl
mov x ,ax
```

توليد شيفرة التجميع لبنى التحكم if then else:

يتطلب توليد شيفرة التجميع لبنى التحكم مثل if then else إضافة التعليمات التالية إلى القواعد الخاصة ببنى التحكم:

```
CON: IF LPAR ID ISEQ NUM RPAR THEN ID EQ NUM ELSE ID EQ NUM {.....
else {
Add_Code("\r\n");
Add_Code("mov al, ");
Add_Code($3);
Add_Code("\r\n");
Add_Code("cmp al, ");
sprintf(strtemp1, "%d", $5);

Add_Code(strtemp1);
Add_Code("\r\n");
Add_Code("jne ");
Add_Code(GenLabel("end"));
```

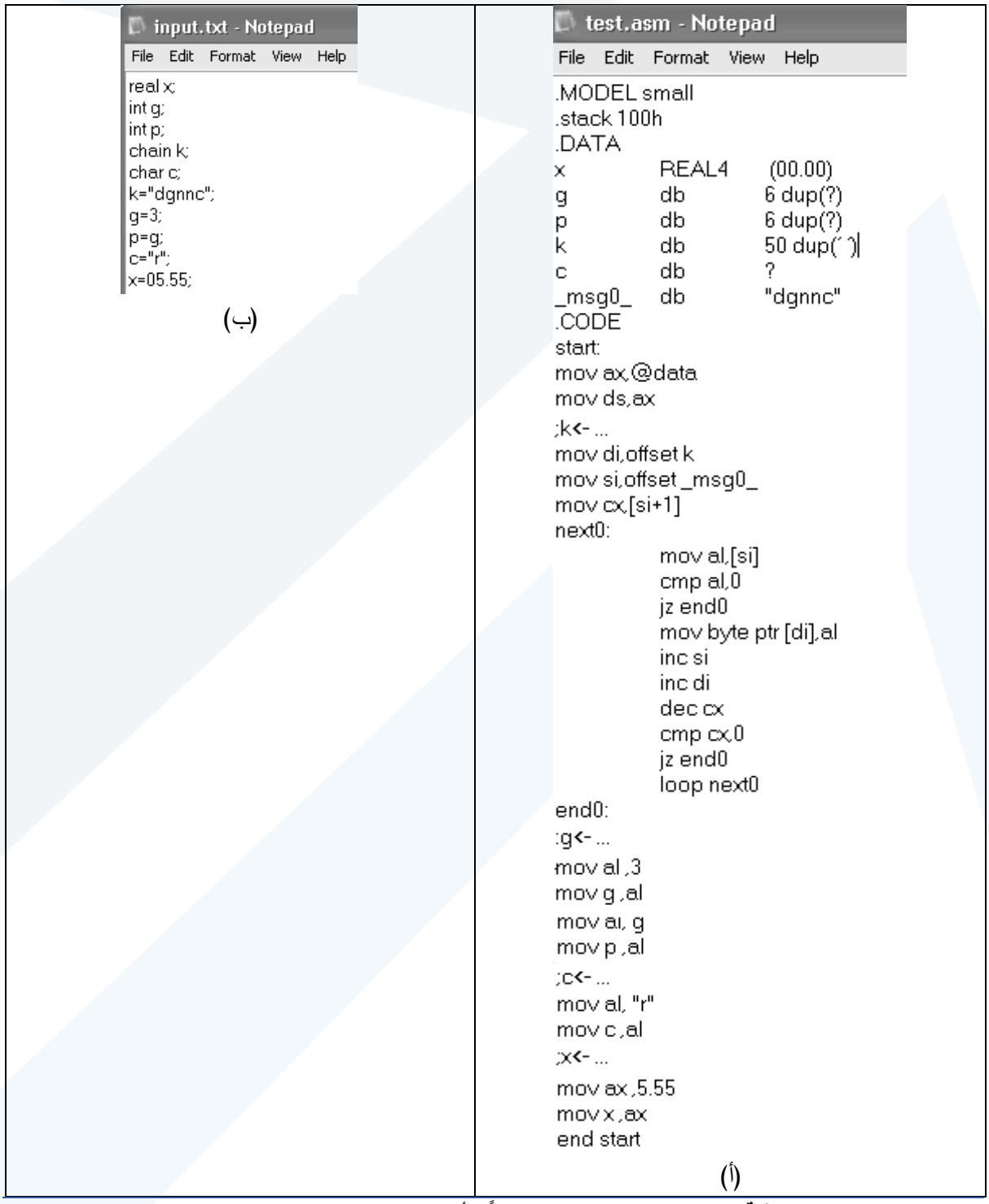
```
//if (g==3) then k=100; else k=200;
// شيفرة الأسمبلي الموافقة
mov al, g
cmp al, 3
jne end2
mov al ,100
mov k ,al
jmp Break0
end2:
mov al ,200
mov k ,al
Break0:
end start
```

```
Add_Code("\r\n");
sprintf(strtemp1, "%d", $10);

Add_Code("mov ");
Add_Code("al ,");
Add_Code(strtemp1);
Add_Code("\r\n");
Add_Code("mov ");
Add_Code( $8);
Add_Code(" ,al ");
Add_Code("\r\n");
Add_Code(GenLabel("end"));

GenLabelIncCounter();
Add_Code("\r\n");}};
```

بعد تعديل الملفات ننفذ المترجم من جديد لتوليد الشيفرة الجديدة المعدلة.

1.2 اختبار المترجم بعد تزويده بعملية توليد الشيفرة الوسيطية:

يوضح الشكل الآتي شيفرة الأسمبلي المكافئة لملف المدخلات المتضمن عبارات تصريح وإسناد فقط:



توليد الشيفرة الوسيطية لعمليات التصريح والإسناد معاً: (أ) الملف المصدري (ب) ملف شيفرة التجميع الناتج

انتهت الجلسة – د. علي ميا ، م. رشا شباني