الفصل الثانى عشر 12

البرامج الفرعية
Subroutines

12-1 مـقدمـة

نستطيع القول بأنه عامة عندما تواجهك كمبرمج مشكلة كبيرة ومطلوب منك برمجتها فإن أسهل الطرق لذلك هى أن تقوم بتجزىء أو تكسير هذه المشكلة الكبيرة إلى مشاكل أو مسائل أصغر ثم تقوم ببرمجة هذه المسائل الصغيرة كل على حدة ثم يكون هناك برنامج أساسى يقوم بتجميع أو تنفيذ هذه الأجزاء الصغيرة بالتتابع الذى يحل المسألة أو المشكلة الأساسية . أحد طرق التجزىء هذه هى البرامج الفرعية subroutines . إن استخدام البرامج الفرعية من مميزاتها تسهيل عملية البرمجة واختصار كمية الذاكرة المستخدمة لكتابة شفرات البرنامج كما سنرى فى هذا الفصل .

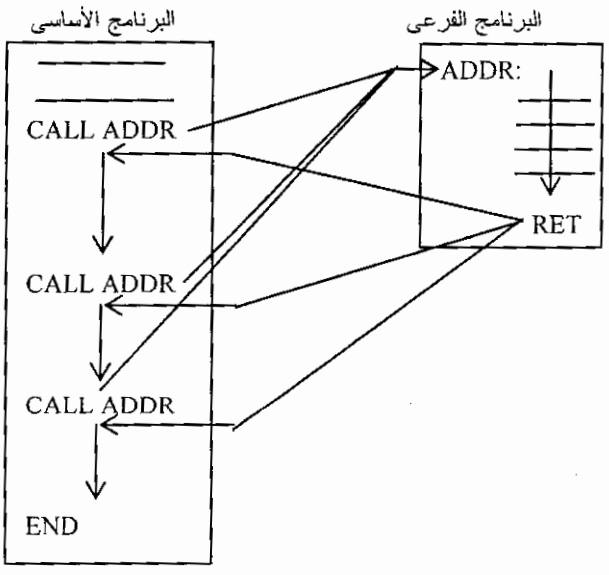
12-2 ما هو البرنامج الفرعى ؟

شكل (12-1) يبين رسما توضيحيا لعلاقة البرنامج الفرعى بالبرنامج الأساسى. نلاحظ من هذا الشكل أن البرنامج الفرعى عبارة عن جزء من برنامج ، أو برنامج صغير ، يتم النداء عليه للتنفيذ من البرنامج الأساسى فينفذ ، وبعد الانتهاء من تنفيذه تتم العودة إلى البرنامج الأساسى وعند نفس المكان الذى تم الخروج منه للبرنامج الفرعى . أى أن طريقة تنفيذ البرنامج الفرعى تشابه وإلى حد كبير طريقة تنفيذ أوامر القفز ، الاختلاف فقط هو فى عملية العودة إلى نفس المكان الذى تم القفز منه فى البرنامج الأساسى بعد الانتهاء من تنفيذ البرنامج الفرعى ، ويرجع ذلك إلى بعض الخطوات أو الاحتياطات التى يعملها المعالج قبل القفز إلى البرنامج الفرعى .

من شكل (12-1) نستطيع أن نتبين الفائدة العظيمة من استخدام البرامج الفرعية وهى توفير الذاكرة المستخدمة لكتابة البرنامج . فى الكثير من التطبيقات يكون هناك جزء من البرنامج تكون مضطرا لكتابته أكثر من مرة وكمثال على ذلك جزء البرنامج الذى يعمل زمن التأخير فى برنامج إشارات المرور فى الفصل السابق . إن مثل هذا الجزء باستخدام البرامج الفرعية يمكن كتابته مرة واحدة فقط وفى كل مرة تكون هناك الحاجة إلى تنفيذه يتم النداء عليه بالأمر CALL فينفذ وبعد الانتهاء من تنفيذه ترجع عملية التنفيذ إلى حيث خرجت من البرنامج الأساسى .

شكل (12-2) يبين خاصية أخرى فى البرامج الفرعية وهى أن أى برنامج فرعى يمكنه النداء على برنامج فرعى آخر ، فمثلا البرنامج الأساسى ينادى البرنامج الفرعى (أ) والبرنامج الفرعى (أ) ينادى البرنامج الفرعى (ب)

والبرنامج الفرعى (ب) ينادى البرنامج الفرعى (ج) وهكذا لأى عدد من التداخلات . هذه العملية تسمى عملية تعشيش nesting للبرامج الفرعية . بعد الانتهاء من تنفيذ آخر برنامج فرعى فى السلسلة وليكن البرنامج الفرعى (ج) فإن المعالج يرجع إلى البرنامج الفرعى (ب) من حيث تم النداء على البرنامج الفرعى (ج) وتتم تكملة البرنامج الفرعى (ب) حيث يرجع المعالج إلى البرنامج الفرعى (أ) من حيث تم النداء على البرنامج الفرعي (ب) ، بعد الانتهاء من تنفيذ البرنامج الفرعى (أ) تتم العودة إلى البرنامج الأساسى من حيث تم النداء على البرنامج الفرعى (أ) .

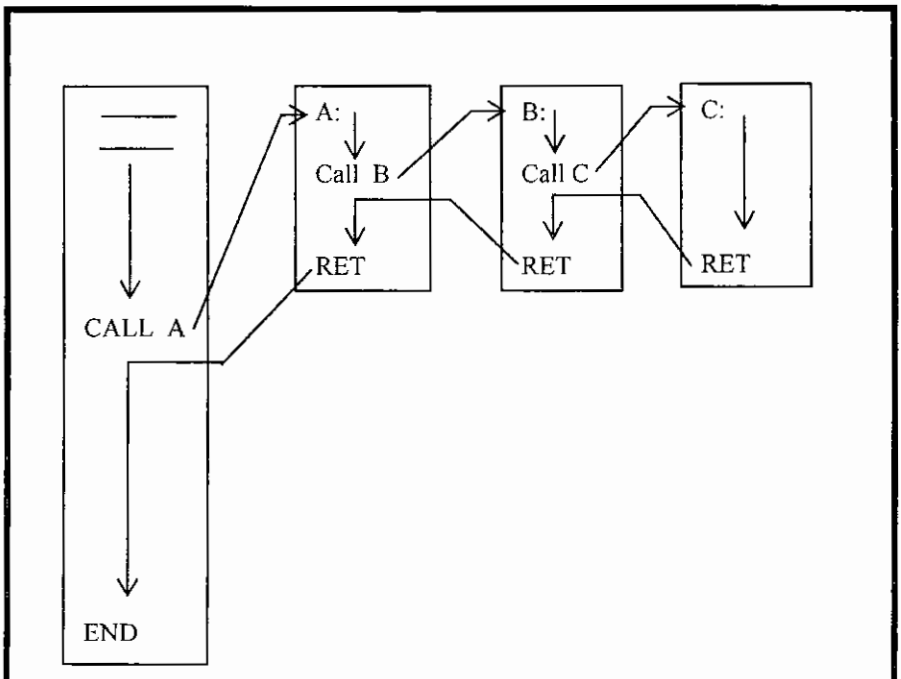


شكل (12-1) البرنامج الفرعى والبرنامج الأساسى

يجب أن نحذر هنا من خطأ أو فخ يمكن أن نقع فيه وهو أن ينادى واحد من البرامج الفرعية اللاحقة أحد البرامج الفرعية السابقة كأن ينادى مثلا البرنامج (ج) البرنامج (ب) أو (أ) . فى هذه الحالة سيدور المعالج فى حلقة لانهائية لا مخرج منها ولن يرجع المعالج أبدا إلى البرنامج الأساسى الذى خرج منه حيث سيظل البرنامج (ج) ينادى على (ب) والبرنامج (ب) ينادى على (ج) إلى ما لا نهاية .

هناك الكثير من أوامر النداء على البرامج الفرعية والعودة منها . فمنها ما هو غير مشروط مثل الأمر CALL addr للشريحتين 8085 و Z80 بحيث ينتقل

التنفيذ إلى العنوان addr دون أى شرط مثله فى ذلك مثل الأمر JMP . وفى المقابل هناك أمر العودة RET للمعالج 8085 و Z80 الذى يكون آخر أمر فى البرنامج الفرعى والذى عند تنفيذه تتم العودة دون أى شرط إلى المكان الذى تم النداء منه . هناك أيضا النداء المشروط على البرامج الفرعية والعودة المشروطة من البرامج الفرعية مثل الأمر CZ addr للمعالج 8085 والذى يعنى نداء على برنامج فرعى مشروط بعلم الصفر يساوى واحدا . كما أن هناك أيضا العودة المشروطة بعلم الصفر يساوى واحدا وهو الأمر RZ للمعالج 8085 . راجع الفصل الخاص ببرمجة المعالج الذى تستخدمه للتعرف على جميع أوامر النداء والعودة من البرامج الفرعية .



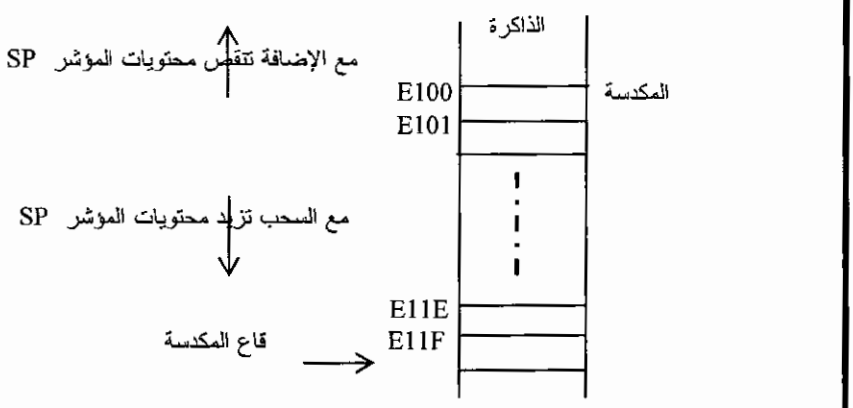
شكل (12-2) عملية تعشيش البرامج الفرعية

12-3 كيف يعود المعالج إلى نفس المكان الذى خرج منه ؟

إن السر يكمن فى المكدسة stack ومؤشر المكدسة stack pointer . المكدسة هى جزء مقتطع من الذاكر RAM الملحقة على المعالج لخدمة أغراض النداء والعودة

من البرامج الفرعية وأيضا لخدمة أغراض المقاطعة وأغراض أخرى كما سنرى فى فصل آخر . إن أقرب تشبيه للمكدسة هو الجوال (الشوال) الذى نضيف إليه من فوهته وعندما نأخذ منه فإننا نأخذ من فوهته أيضا ، أى أن آخر ما وضعنا فى الشوال (المكدسة) يكون أول ما نأخذ منها أو Last In First Out وتختصر LIFO . تشبيه آخر للمكدسة هو رص الأطباق ، فأنت حينما ترص الأطباق رأسيا تضيف إلى قمة المكدسة وعندما تريد أخذ طبق فإنك تأخذ آخر طبق وضعته على القمة .

وأما مؤشر المكدسة stack pointer, SP فإنه مسجل مكون من 16 بت محتوياته هى عنوان قمة أو آخر مكان تم التخزين فيه فى المكدسة . عندما تكون المكدسة فارغة فإن مؤشر المكدسة يشير إلى قاعها وتكون محتويات ال SP هى عنوان آخر مكان فى المكدسة . عند الإضافة إلى المكدسة فإن المؤشر ينقص محتوياته وعند السحب من المكدسة فإن المؤشر تزيد محتوياته بحيث أن الزيادة أو النقص تكون دائما بمقدار 2 لكل عملية سحب أو إضافة كما فى شكل (12-3) .



شكل (12-3) المكدسة ، السحب منها والإضافة إليها

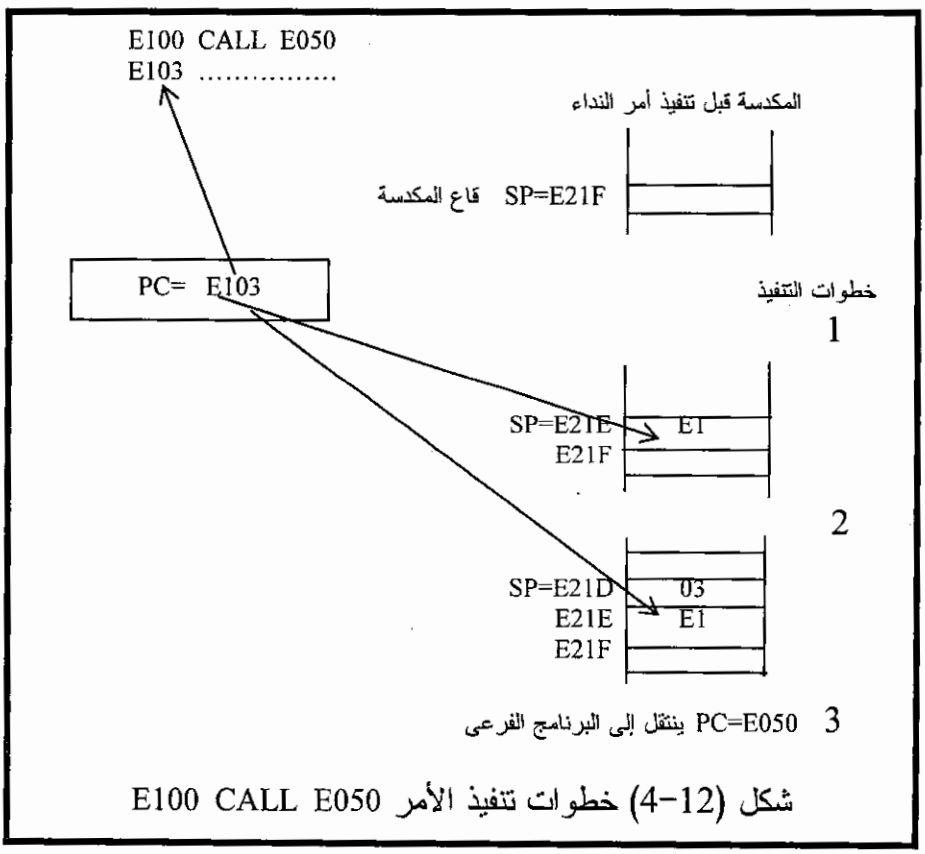
إن الإضافة والسحب من المكدسة تكون دائما على أزواج المسجلات ولا يمكنك بأى حال أن تضيف أو تسحب مسجلا واحدا أو بايت واحدة من أو إلى المكدسة. لذلك فإن مؤشر المكدسة حينما يزيد أو ينقص فإنه يزيد أو ينقص بمقدار اثنين ، أى اثنين بايت ، ولا يمكن أن يزيد أو ينقص بمقدار واحد على الإطلاق . شكل (12-4) يبين الخطوات التى يقوم بها المعالج عند تنفيذ أمر النداء على أى برنامج فرعى وهى كالتالى :

1. مؤشر المكدسة SP ينقص decrement بمقدار واحد ، ويخزن البايت ذات

القيمة العظمى من عداد البرنامج PC فى هذا العنوان .

2. مؤشر المكدسة SP ينقص بمقدار واحد آخر وتخزن البايت ذات القيمة الصغرى من عداد البرنامج فى العنوان الجديد . بذلك يكون قد تم الحفاظ على محتويات عداد البرنامج وهى عنوان الأمر الذى عليه الدور فى التنفيذ بعد الأمر CALL فى المكدسة .
3. يحمل عداد البرنامج بعنوان أول بايت فى البرنامج الفرعى وهو العنوان الموجود فى أمر النداء CALL addr . بذلك ينتقل التنفيذ إلى البرنامج الفرعى .

لاحظ من شكل (12-4) أن قاع المكدسة وهى العنوان E21F تكون فارغة دائما وذلك لأن عملية إنقاص مؤشر المكدسة تكون دائما قبل التخزين فيها . فى نهاية تنفيذ البرنامج الفرعى يكون آخر أمر ينفذ هو الأمر RET وعند تنفيذ هذا الأمر تتم العمليات العكسية للعمليات الموضحة فى شكل (12-4) وهى كالتالى :



شكل (12-4) خطوات تنفيذ الأمر E100 CALL E050

1. محتويات قمة المكدسة فى العنوان E21D وهى الرقم 03 تدفع إلى البايت ذات القيمة الصغرى من عداد البرنامج وتزداد قيمة محتويات مؤشر المكدسة بمقدار واحد فيصبح E21E .
2. محتويات القمة الجديدة للمكدسة فى العنوان E21E وهى الرقم E1 تدفع إلى البايت ذات القيمة العظمى من عداد البرنامج ، وتزداد قيمة محتويات مؤشر المكدسة بمقدار واحد آخر فتصبح E21F وهى قمة المكدسة (أو قاعها) التى كانت موجودة فى بداية تنفيذ الأمر CALL E050 .
3. بذلك تصبح محتويات عداد البرنامج هى عنوان الأمر الموجود عند المكان E103 وهو الأمر الذى عليه الدور فى التنفيذ فى البرنامج الأساسى بعد أمر النداء مباشرة . إن هذه الخطوات قد تختلف اختلافا بسيطا من معالج لآخر ولكن يظل المعنى الأصلى كما هو .

فى حالة البرامج الفرعية المعششة مع بعضها nested ، مهما كانت درجة تعشيشها ، فإنه عند كل أمر نداء CALL يتم تخزين عنوان الأمر التالى للأمر CALL مباشرة (أى الأمر الذى عليه الدور فى التنفيذ) وهكذا إلى أن نصل إلى آخر برنامج فرعى فى السلسلة حيث سيكون الأمر RET فى آخره هو أول أمر RET يتم تنفيذه ونتيجة له يحمل عداد البرنامج بأول اثنين بايت من قمة المكدسة فيرجع التنفيذ إلى البرنامج الفرعى قبل الأخير وهكذا مع كل أمر RET يسحب عنوان (2 بايت) من قمة المكدسة ويرجع التنفيذ إلى برنامج فرعى سابق إلى أن يصل التنفيذ إلى حيث انتهى من البرنامج الأساسى .
هناك أمر يمكن به تخزين أى زوج مسجلات فى المكدسة وهو الأمر :

PUSH rp

الذى يقوم بتنفيذ أول خطوتين فى شكل (12-4) ، مع ملاحظة أن محتويات عداد البرنامج لا تتغير هنا على الإطلاق لأنه ليس هناك أى قفز . كذلك فإن الأمر :

POP rp

يقوم بالعملية العكسية للأمر PUSH ، أى يسحب اثنين بايت من المكدسة ويضعهم فى زوج المسجلات المذكور فى الأمر POP ويزيد محتويات مؤشر المكدسة بمقدار اثنين . هذه العملية تكون مهمة جدا عند خدمة المقاطعة كما سنرى فى فصل قادم وهذان الأمران موجودان لجميع المعالجات التى ندرسها ويمكن مراجعتهما فى قوائم الأوامر .
إن محتويات مؤشر المكدسة التى تشير إلى قاعها أى أول مكان فيها يتم تحديدها عن طريق المبرمج فى البرنامج باستخدام الأمر LXI SP,addr حيث يقوم هذا الأمر بتحميل المسجل SP بالعنوان addr الذى يختاره المبرمج ليكون عنوان قاع

المكدسة وذلك فى المعالج 8085 ، أو LD SP,addr فى حالة المعالج Z80 (لاحظ أنه قبل أى تعامل مع المكدسة تكون قمتها هى قاعها أى وهى فارغة) .
بذلك نكون قد أنهينا أهم ما يتعلق بالمكدسة ومؤشر المكدسة والبرامج الفرعية وفوائدها وننتقل الآن إلى بعض الأمثلة كتطبيقات على ذلك . لاحظ أن كل ما تم شرحه فى هذا الباب يمكن تطبيقه على أى معالج من المعالجات التى درسناها (والتى سندرسها بخلاف طفيف) ، فقط نكون حذرين عند كتابة شكل الأمر بالأسمبلى أو شفرته الستعشرية ، ولكن الأساس أو الهيكل العام لفكرة المكدسة والبرامج الفرعية التى شرحناها هى نفسها دائما .

12-4 حساب أزمنة التأخير

مثال 12-1

لدينا بوابة الإخراج رقم 00 وموصلا عليها ثمانية دايودات أو موحدات ضوئية كل دايود موصل على بت من بتات البوابة . المطلوب إنارة هذه الدايودات بالتتابع بحيث أن الدايود الأول يضيء وبعده بنصف ثانية يضيء الدايود الثانى وبعده بنصف ثانية أخرى يضيء الثالث ، وهكذا حتى تصبح كل الدايودات مضيئة ، ثم بعد آخر دايود بنصف ثانية تطفأ جميع الدايودات ثم يبدأ فى الإضاءة من جديد بنفس الطريقة السابقة . اكتب برنامجا يقوم بهذه المهمة مستخدما البرامج الفرعية .
تعتمد فكرة البرنامج على عمل برنامج فرعى للتأخير بزمن مقداره نصف ثانية ثم ننادى على هذا البرنامج من البرنامج الأساسى بعد إنارة كل دايود . قبل أن ندخل فى تفاصيل البرنامج نريد أن نوضح هنا كيفية الحصول على أى زمن تأخير بأى قيمة . تقوم الفكرة أساسا على عمل حلقة ينفذها المعالج عددا من المرات دون التأثير على أى شيء فى البرنامج وبمعلومية عدد مرات تنفيذ هذه الحلقة وعدد الأوامر فيها وزمن تنفيذ كل أمر نستطيع حساب الزمن الكلى لتنفيذ الحلقة . انظر مثلا لهذا البرنامج البسيط :

```
      MVI C,FF;  7
LOOP: DCR C;     4
      JNZ LOOP;  10
```

الرقم الذى على يمين كل أمر هو عدد نبضات التزامن clock التى يأخذها الأمر حتى يتم تنفيذه وهذا العدد موضح فى قوائم أوامر كل واحد من المعالجات التى شرحنا طريقة برمجتها فى الفصول السابقة . فإذا كان تردد التزامن clock المستخدم هو 2 ميجاهرتز فإن ذلك يعنى أن زمن كل نبضة هو نصف ميكروثانية . لذلك فإن زمن تنفيذ الأمر الأول مثلا سيكون 3,5 ميكروثانية والأمر الثانى سينفذ فى 2 ميكروثانية والثالث سينفذ فى 5 ميكروثانية وهكذا .

فى البرنامج السابق نلاحظ أن زمن تنفيذ الحلقة مرة واحدة سيكون (4 + 10)×0،5=7 ميكروثانية ، وهذه الحلقة ستنفذ عددا من المرات مقداره 256 (FF) مرة لذلك فإن مثل هذه الحلقة ستعطى زمن تأخير مقداره 256×7=1792 ميكروثانية ، وأما البرنامج السابق كله فسيعطى 1792 + 3,5 = 1795,5 ميكروثانية . إذا كان هذا الزمن يكفى كتأخير لما تحتاج فقد انتهيت وإلا فعليك البحث عن طريقة تطيل بها هذا الزمن ، ومن ذلك استخدام زوج من المسجلات بدلا من مسجل واحد كما فى البرنامج التالى :

```
      LXI D,FFFF;  10
LOOP: DCX D;       6
      MOV A,D;     4
      ORA E;       4
      JNZ LOOP;    10
```

هذا البرنامج يقوم بتحميل الزوج DE بالقيمة FFFF(65536) ثم يدخل فى حلقة إنقاص بمقدار واحد إلى أن تصل محتويات زوج المسجلات إلى أصفار وعندها تنتهى الحلقة . لاحظ أن الأمر DCX ليس له تأثير على الأعلام لذلك فقد تم نقل محتويات مسجل من الاثنين إلى المركم ثم نفذت عملية OR على المسجلين والتى لن تكون نتيجتها صفرا إلا إذا كانت محتويات كل من المسجلين أصفارا . زمن التأخير الذى سيعطيه هذا البرنامج سيكون :

0،5×24×0،5×10+65535=786437 ميكروثانية .

=0،79 ثانية تقريبا .

إذا كان هذا الزمن يفى بما تحتاج إليه من زمن تأخير فقد انتهيت ، وإلا فعليك استخدام الحلقات المعشقة كما فى البرنامج التالى :

```
      MVI B,FF;   7
LOOP1: MVI C,FF;  7
LOOP2: DCR C;     4
       JNZ LOOP2; 10
       DCR B;     4
       JNZ LOOP1; 10
```

زمن تأخير الحلقة الداخلية = 0،5×14×256

= 1792 ميكروثانية تقريبا .

زمن تأخير الحلقة الخارجية= 0،5×(10+4+7)+1792)×256

= 461440 ميكروثانية تقريبا .

زمن التأخير الكلى للبرنامج= 3,5 + 461440

= 461443,5 ميكروثانية .

= 0،46 ثانية تقريبا .

لاحظ أنه وإن كان زمن التأخير الناتج من حلقتين معشقتين أقل من زمن التأخير الناتج من زوج من المسجلات إلا أن الحلقات المعشقة يمكن تعشيقها لأى درجة فمثلا فى داخل الحلقة LOOP2 يمكن عمل حلقة ثالثة LOOP3 وهكذا للحصول على أزمنة تأخير كبيرة . أحيانا يكون المطلوب أزمنة محددة بقدر الإمكان كما فى المثال الذى نحن بصدده الآن حيث المطلوب هو زمن تأخير مقداره 0,5 ثانية . فى هذه الحالة يمكن استخدام الأمر NOP فى أماكن معينة فى حلقات التأخير للضبط الدقيق للأزمنة المطلوبة . مثلا ماذا سيكون الوضع لو أضفنا الأمر NOP داخل الحلقة الداخلية LOOP2 فى البرنامج السابق ؟ وإجابة على ذلك فمن المعروف أن الأمر NOP يأخذ 4 نبضات تزامن لذلك سيصبح زمن التأخير الجديد هو 59،. ثانية تقريبا ، وهذه القيمة أبعد من القيمة المطلوبة !! فى هذه الحالة يمكن وضع الأمر NOP فى الحلقة الخارجية ، وإذا لم يف (يفى) بالغرض المطلوب فيمكن تركه فى الحلقة الداخلية مع التغيير فى القيمة الابتدائية فى أى من المسجلين B أو C ، ولقد وجدنا أنه بوضع القيمة الابتدائية D8 فى المسجل B بدلا من القيمة FF فإن زمن التأخير فى هذه الحالة يساوى499932 ميكروثانية وهى أقرب شىء إلى نصف الثانية . أى أن التعديل البسيط فى القيمة النهائية لزمن التأخير ممكن بعدة طرق .

بعد أن رأينا كيفية الحصول على زمن التأخير المطلوب فإن البرنامج التالى يبين عملية الإضاءة التتابعية ، وهذا البرنامج مكتوبا بلغة الأسمبلى للشريحة 8085 . هذا البرنامج كتب بصورة مبسطة جدا ويستطيع كل قارئ أن يأتى ببرنامج آخر يؤدى نفس الهدف وقد يكون أبسط من ذلك . ولقد تعمدنا كتابة البرنامج باستخدام العلامات LABELS حتى نترك للقارئ حرية كتابة البرنامج فى أى مكان فى الذاكرة يريده . بذلك نكون قد انتهينا من إعطاء القارئ فكرة كافية عن البرامج الفرعية وما يتعلق بها من المكدسة ومؤشر المكدسة .

```
START:  MVI A,01
        OUT 00
        CALL DELAY
        MVI A,03
        OUT 00
        CALL DELAY
        MVI A,07
        OUT 00
        CALL DELAY
        MVI A,0F
        OUT 00
        CALL DELAY
        MVI A,1F
```

```
        OUT 00
        CALL DELAY
        MVI A,3F
        OUT 00
        CALL DELAY
        MVI A,7F
        OUT 00
        CALL DELAY
        MVI A,FF
        OUT 00
        CALL DELAY
        MVI A,00
        OUT 00
        CALL DELAY
        JMP START
DELAY:  MVI B,DB
LOOP1:  MVI C,FF
LOOP2:  DCR C
        NOP
        JNZ LOOP2
        DCR B
        INZ LOOP1
        RET
```

12-5 تمارين

1. إشرح دور المكدسة مع البرامج الفرعية ؟
2. ما هو الفرق بين القفز العادى فى أى برنامج والقفز إلى برنامج فرعى ؟
3. اشرح دور الأمر RET (أمر العودة من البرامج الفرعية) فى ضمان عودة المعالج إلى نفس المكان الذى خرج منه فى البرنامج الأساسى ؟
4. لماذا يكون من الضرورى عادة استخدام الأمر PUSH فى أول البرنامج الفرعى والأمر POP فى آخره ؟
5. يحتوى هذا الفصل على بعض البرامج الفرعية للحصول على أزمنة التأخير ، فهل يتغير مقدار هذه الأزمنة بتغير التزامن clock المستخدمة ؟
6. اكتب برنامجا فرعيا للحصول على زمن تأخير مقداره 100ميللثانية مع العلم أن التزامن يساوى 2 ميجاهرتز؟
7. اكتب برنامجا فرعيا آخر يستخدم البرنامج الفرعى السابق للحصول على زمن تأخير مقداره ثانية واحدة ؟
8. اكتب برنامجا فرعيا آخر يستخدم البرنامجين السابقين للحصول على زمن تأخير مقداره دقيقة واحدة ؟
9. استخدم البرامج الفرعية السابقة فى عمل ساعة رقمية تظهر الثوانى والدقائق والساعات على مظهرات السبع قطع 7 segment ؟
10. اكتب برنامج التحكم فى إشارات المرور فى الفصل السابق مستخدما البرامج الفرعية ، ولاحظ الفرق ؟
11. اكتب برنامج يحسب قيمة X التالية :

$$X= 5! + 8! + 9!$$

حيث ! تمثل مضروب الرقم .

12. اكتب برنامج يحسب قيمة X التالية :

$$X= 5^5 + 8^4 + 9^3$$