الباب السادس

البرامج الفرعية

مفهوم البرامج الفرعية

تداخل البرامج الفرعية

الباب السادس

البرامج الفرعيةSubroutines

مفهوم البرامج الفرعية:

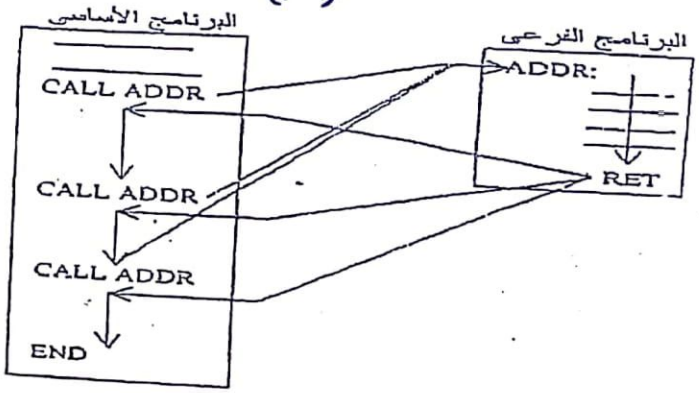
نستطيع القول عامة عندما بأنه تواجهك كمبرمج مشكلة كبيرة ومطلوب منك برمجتها فأن اسهل الطرق لذلك هى أن تقوم بتجزئة هذة المشكلة الكبيرة إلى مشاكل أو مسائل أصغر ثم تقوم ببرمجة هذه المسائل الصغيرة كل على حده ثم يكون هناك برنامج اساسى يقوم بربط أو تجميع وظائف هذه الأجزاء الصغيرة بالتتابع الذى يحل المسألة أو المشكلة الاساسية .

أحد طرق التجزىء أو التقسيم هى استخدام البرامج الفرعية Subroutines حيث أن استخدام البرامج الفرعية يعطي مميزات عديدة منها (1) تسهيل عملية البرمجة ,(2) اختصار كمية الذاكرة المستخدمة لكتابة شفرات البرنامج كما سنرى . هذا بجانب أن البرامج الفرعية من الموضوعان التي يجب أن نكون على دراية بها عند التعامل مع اى معالج او متحكم وسنتاولها هنا بنظرة عامة و اما تفصليا تعتمد على نوع المعالج او المتحكم.

** بناء البرامج الفرعية:

تبنى البرامج الفرعية كالبرامج العادية ولكن عند انتهائها تعود الى النقطة التى بعد النقطة التى تفرعت عندها . ويوضح الشكل رسما توضيحيا لعلاقة البرنامج الفرعى بالبرنامج الاساسى , نلاحظ من هذا الشكل أن البرنامج الفرعى عبارة عن جزء من البرنامج , أو برنامج صغير يتم النداء call عليه للتنفيذ من البرنامج الاساسى فينفذ , وبعد الانتهاء من تنفيذة تتم العودة الى البرنامج الاساسى وعند نفس المكان الذى تم الخروج منه للبرنامج الفرعى. اى ان طريقة تنفيذ البرنامج الفرعى شابة والى حد كبير طريق تنفيذ اوامر القفز jump instruction.

(٧٩)

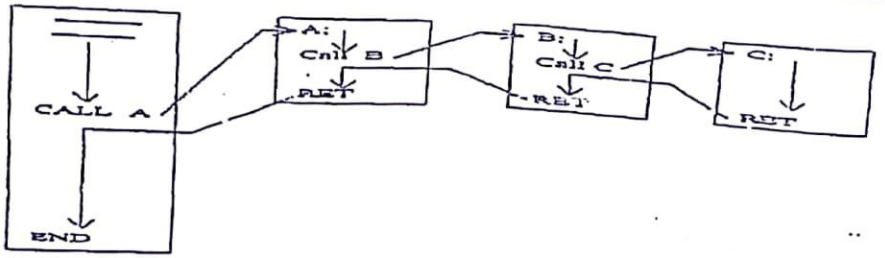


شكل (6- 1) رسما توضيحيا لعلاقة البرنامج الفرعي بالبرنامج الاساسي

الاختلاف فقط هو في عملية العودة الى نفس المكان الذي تم القفز منه في البرنامج الاساسي بعد الانتهاء من تنفيذ البرنامج الفرعي , ويرجع ذلك الى بعض الخطوات او الحتياطات التي تعملها CPU قبل القفز الى البرنامج الفرعي.

او بمعنى اخر هي إجراءات مكتوبة بشكل مستقل عن البرنامج الرئيسي. متى وجب على البرنامج الرئيسي أن ينجز الوظيفة المحددة بواسطة البرنامج الفرعي فإنه يستدعي البرنامج الفرعي إلى العمل و من أجل هذا يجب أن يتحول التحكم من البرنامج الرئيسي إلى نقطة البداية في البرنامج الفرعي، حيث يستمر تنفيذ البرنامج الفرعي، و عند اكتمال التنفيذ يعود التحكم إلى البرنامج الرئيسي بالتعليمة التالية لتعليمة مناداة البرنامج الفرعي:

نستطيع أن نتبين الفائدة العظيمة من استخدام البرامج الفرعية وهي توفير الذاكرة المستخدمة لكتابة البرنامج . في الكثير من التطبيقات يكون هناك جزء من البرنامج نحتاج إلى تكرار كتابته أكثر من مرة في البرنامج الأساسي, و كمثال على ذلك برامج أزمنة التأخير التي يمكن في هذه الحالة كتابتها مرة واحدة كبرنامج فرعي النداء عليه بالأمر CALL فينفذ وبعد الانتهاء من تنفيذه ترجع عملية التنفيذ الى حيث خرجت من البرنامج الاساسي.



(٨٠)

شكل (6-2) يبين خاصية أخرى في البرامج الفرعية

الشكل(6- 2)يبين خاصية أخرى في البرامج الفرعية وهي ان اى برنامج فرعي يمكنه النداء على برنامج فرعي أخر , فمثلا البرنامج الاساسي ينادى البرنامج الفرعي (ا) و البرنامج الفرعي (ا) ينادى البرنامج الفرعي (ب) و البرنامج الفرعي (ب) ينادى البرنامج الفرعي (ج) وهكذا لأى عدد من التداخلات. هذه العملية تسمى عملية التداخل Nesting للبرامج الفرعية. بعد الانتهاء من تنفيذ اخر برنامج فرعي في السلسلة وليكن البرنامج الفرعي (ج) فان المعالج يرجع الى البرنامج الفرعي (ب) من حيث تم النداء على البرنامج الفرعي (ج) وتتم تكملة البرنامج الفرعي(ب) حيث يرجع المعالج الى البرنامج الفرعي (ا) من حيث تم النداء على على البرنامج الفرعي (ب) بعد الانتهاء من تنفيذ البرنامج الفرع (ا) تتم العودة الى البرنامج الاساسي من حيث تم النداء على البرنامج الفرعي (ا) .

تعليمات المناداة و العودة

كلاً من هاتين التعليمتين معاً أزودان تقنية من أجل استدعاء البرنامج الفرعي إلى العمل و إعادة التحكم إلى البرنامج الأساسي لمتابعة تنفيذه. إن تعليمة المناداة مشروحة في الجدول:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد	يُتابع التنفيذ في البرنامج الفرعي من العنوان المحدد بواسطة المتحول operand الموجود في تعليمة المناداة. و المعلومات المطلوبة من أجل العودة مثل IP و CS تُحفظ في المكدس	CALL operand	مناداة برنامج فرعي	CALL

شكل(6-3)

إن كل برنامج فرعي يجب أن ينتهي بتنفيذ التعليمة التي تعيد التحكم إلى البرنامج الرئيسي و هذه التعليمة هي تعليمة العودة RET و هي مشروحة بالشكل التالي:

الأعلام المتأثرة	العملية	الصيغة	المعنى	الكلمة المختزلة
لا يوجد		RET/RET operand	العودة إلى البرنامج المُستدعي	RET

شكل6-4

العودة إلى البرنامج المستدعي عن طريق إعادة تخزين قيم IP فقط أو IP و CS معاً (حسب نوع تعليمة المناداة أي ضمن المقطع الجزئي أو خارجه) من أجل المتحول Far_pro . و إذا كان المتحول (operand) موجوداً في تعليمة العودة RET فيجب إضافته إلى محتويات SP . هذا و إن المتحول إذا وجد في تعليمة العودة فهو عبارة عن متحول إزاحة بـ 16 بت.

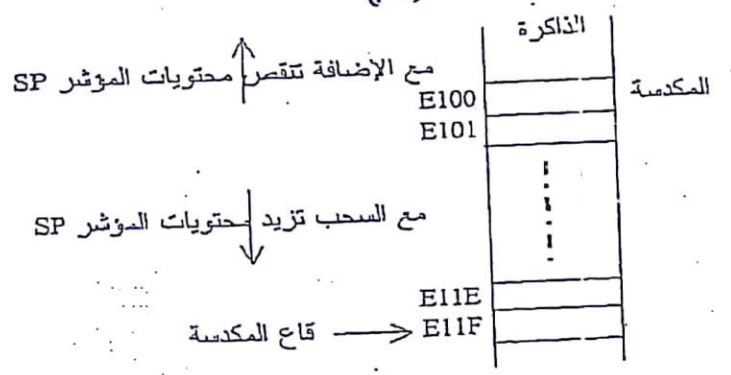
كيف يعود المعالج الى نفس المكان الذى خرج منه :

ان السر يكمن فى المخزن Stack ومؤشر المخزن Stack Pointer

و المخزن هو جزء مقتطع من الذاكرة RAM الملحق على المعالج أو حتى المسجلات العامة الموجودة داخل المعالج لخدمة أغراض النداء و العودة من البرامج الفرعية وايضا لخدمة اغراض المقاطعة interrupt وأغراض أخرى.

أن أقرب تشبيه للمخزن stack هو الصندوق الذي نضيف إلى محتوياته من فوقه وعندما نأخذ منه فأننا نأخذ من فوقه أيضا , أى أن أخر ماوضعنا فى الصندوق (المخزن) يكون أول ما نأخذ منه أو Last In First Out وتختصر LIFO .

ومسجل المخزن Stack Pointer SP هو مسجل مكون من 16 خانةأو على حسب نوع المعالج , ومحتوياته هي عنوان قمة أو أخر مكان تم التخزين فيه فى المخزن . عندما تكون المخزن فارغة فأن المؤشر SP يشير إلى قاعها و تكون محتويات مؤشر المخزن SP هي عنوان آخر مكان في المخزن عند الإضافة إلى المخزن فأن المؤشر ينقص محتوياته وعند السحب من المخزن فأن المؤشر تزيد محتوياته وهناك أختلاف بين معالج واخر ولكن بصورة عامة فانه عند النداء على البرنامج الفرعى فإن محتويات عداد البرنامج PC التى تحتوى عنوان الامر التالى فى التنفيذ فى البرنامج الاساسى يتم دفعها الى حيث يشير مؤشر المخزن و بالتالي تنقص محتويات المؤشرSP . بعد ذلك يتم تحميل عداد البرنامج بعنوان بداية البرنامج الفرعى وبذلك تنتقل عملية التنفيذ الى هناك كما بالشكل.



شكل(6-5)

فى نهاية البرنامج الفرعى يوجد الامر RET الذي بتنفيذه يتم سحب قمة المخزن stack التى يشير إليها المؤشر وتوضع في عداد البرنامج PC مرة اخرى , وبذلك تتم العودة الى البرنامج الاساسى وفى نفس المكان الذى تم الخروج منه.

فى حالة البرامج المتداخلة Nested Programs مهما كانت درجة تداخليا , وعند كل أمر نداء CALL يتم تخزين عنوان الأمر التالى للأمر CALL مباشرة (أى الامر الذى عليه الدور فى التنفيذ) وهكذا الى أن نصل إلى نداء آخر برنامج فرعى فى السلسلة حيث يكون الامر RET فى اخره هو أول أمر RET يتم تنفيذه ونتيجة له يحمل عداد البرنامج بأول عنوان فى قمة المخزن فيرجع التنفيذ إلى البرنامج الفرعي قبل الأخير وهكذا مع كل أمر RET يسحب عنوان قمة المخزن stack ويرجع إلى التنفيذ برنامج فرعى سابق الى ان يصل التنفيذ إلى حيث انتهى من البرنامج الاساسى .

أسئلة على الباب السادس

1. ماهو مفهوم البرامج الفرعية .
2. كيف تبنى البرامج الفرعية ؟
3. ماهى الفائدة من استخدام البرامج الفرعية ؟
4. ماهى عملية التداخل Nesting اثناء وجود اكثر من برنامج فرعى؟
5. اشرح بالتفصيل كيف يعود البرنامج الى المكان الذى خرج منه. مع توضيح دور المكدسة؟