الفصل الثالث 3

برمجة المعالج

Microprocessor Programming

3-1 مقدمة

بعد أن أخذنا فكرة عامة عن وظيفة ومهمة كل مسجل من مسجلات شريحة المعالج سنتعرض فى هذا الفصل وبعض الفصول القادمة إلى كيفية برمجة هذه الشريحة وهذا هو الشق الأول من دراسة شرائح المعالجات كما ذكرنا ، وأما الشق الآخر وهو مواجهة المعالج فإن ذلك سيكون فى فصول قادمة أخرى إن شاء الله . سنتعرض فى هذا الفصل لدراسة الأفكار العامة عن لغة التجميع (الأسمبلى) دون أن نخص بالذكر أى شريحة معينة حيث سيعقب هذا الفصل فصل خاص بلغة الأسمبلى والأوامر الخاصة بكل واحدة من شرائح المعالج Intel8085 و Z80 .

3-2 لغات الحاسب Computer languages

لغات الحاسب يمكن تقسيمها إلى قسمين أساسيين :

القسم الأول : وهو اللغات التى تعتمد على الماكينة machine dependent languages وكل لغة من هذا النوع تكون مصممة لماكينة معينة وما يتوافق منها مع ماكينة معينة ليس بالضرورة أن يتوافق مع الماكينات الأخرى . من أمثلة هذا النوع من اللغات ، لغة الماكينة machine language ولغة الأسمبلى assembly language حيث أن كل معالج له مجموعة الأوامر الخاصة به التى عادة لا تتوافق مع المعالجات الأخرى . فأنت مثلا إذا كتبت برنامجا بلغة الأسمبلى الخاصة بالشريحة MC6800 فإن هذا البرنامج لا يمكن أن ينفذ مع الشريحة Intel8085 أو الشريحة Z80 .

القسم الثانى : هو اللغات التى لا تعتمد على الماكينة machine independent language ومن أمثلة هذه اللغات جميع اللغات ذات المستوى العالى high level languages مثل الفورتران والبسكال و PL/1 و PL/C وغير ذلك من هذه اللغات التى تعد بالمئات الآن .

3-3 ما هو الأمر؟

الأمر معناه الكود أو الشفرة الثنائية التى تعطى للمعالج والتى على أثرها يقوم بعمل فعل معين . هذا الفعل قد يكون عملية جمع رقمين أو إحضار معلومة من الذاكرة أو غير ذلك من الأفعال التى يستطيع المعالج القيام بها . من أمثلة هذه الشفرات الثنائية الشفرة 10000000 والتى معناها اجمع محتويات المسجل B مع

مسجل التراكم A وضع النتيجة فى المسجل A . كما نرى فإن المعالج يتعرف فقط على الشفرات الثنائية ولا يعرف أى نوع آخر من الشفرات سواء كانت حرفية أو ثمانية أو ستعشرية . لقد سبق تعريف كل من الأمر والبرنامج فى الفصل الأول بصورة عامة ولكننا أعدنا تعريفهما فى هذا الفصل بشيء من التفصيل .

3-4 ما هو البرنامج ؟

```
00111010
01100000
00000000
01000111
00111010
01100001
00000000
10000000
00110010
01100010
00000000
```

شكل (3-1) برنامج يجمع محتويات العنوان 60H مع العنوان 61H ويضع النتيجة فى العنوان 62H

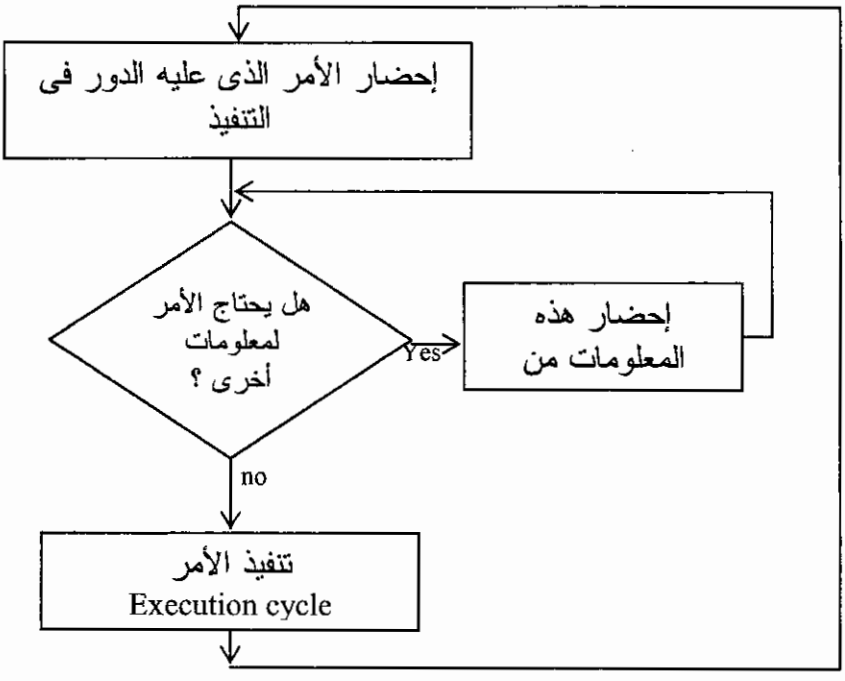
البرنامج عبارة عن مجموعة من الأوامر التى ينتج عن تنفيذها مجتمعة هدف أو عمل معين كإدارة موتور مثلا أو التحكم فى متغير معين أو التعرف على معلومة معينة من بين الكثير من المعلومات . يجب أن يحتوى البرنامج أو يحدد مصدر أى معلومة يتم استخدامها بواسطته ، فمثلا لإجراء عملية جمع لرقمين لابد وأن يحدد البرنامج أين يوجد الرقمان وأين ستوضع النتيجة . يمكن النظر لأى برنامج على أنه مجموعة من الشفرات الثنائية المخزنة فى الذاكرة فى انتظار أن يقوم المعالج بتنفيذها . كمثال على ذلك أنظر إلى البرنامج الموجود فى شكل (3-1) دون أن تبذل أى مجهود فى محاولة فهمه الآن . إن هذا البرنامج يقوم بجمع محتويات عنوان الذاكرة رقم 60H مع محتويات العنوان رقم 61H ويضع النتيجة فى العنوان 62H ، إن الحرف H يعنى أن هذه الأرقام مكتوبة بالنظام الستعشرى

. هذا البرنامج نقول أنه مكتوب بلغة الماكينة machine language وعادة يطلق عليه اسم برنامج الهدف object program .

5-3 كيف يقوم المعالج بتنفيذ البرنامج ؟

شكل (3-2) يبين الخطوات التى يقوم بها المعالج لكى يتم تنفيذ أى برنامج وهى كالتالى :

1. يقوم المعالج (وحدة التحكم بداخله) بقراءة الأمر الأول من الذاكرة وتخزينه فى مسجل الأوامر IR .



شكل (3-2) طريقة تنفيذ البرنامج بواسطة المعالج

2. يقوم المعالج بفك شفرة هذا الأمر أو بمعنى آخر يتم التعرف على هذا الأمر من بين قائمة أوامر المعالج ، وعلى ضوء هذا التعارف يقرر المعالج إذا كان هذا الأمر سيحتاج لمعلومات أخرى من الذاكرة لكى تتم عملية التنفيذ أم لا ؟ وإذا كان الأمر سيحتاج لمثل هذه المعلومات يقوم المعالج بإحضارها

أيضا من الذاكرة . بذلك تنتهى المرحلة الأولى من مـراحل تنفيذ الأمـر الأول وهى مرحلة الإحضار fetching cycle .

3. بمجرد الانتهاء من مرحلة أو دورة الإحضار تبدأ مرحلة التنفيذ execution cycle حيث يقوم مشفر الأوامر مع وحدة التحكم بإرسال الإشارات المناسبة إلى وحدة الحساب والمنطق التى تقوم بتنفيذ هذا الأمر .

4. بعد الانتهاء من مرحلة تنفيذ الأمر الأول يرجع المعالج إلى الخطوة الأولى حيث يبدأ فى عملية إحضار الأمر الثانى ثم يتم تنفيذه ثم يبدأ فى عملية إحضار الأمر الثالث وتنفيذه وهكذا حتى ينتهى البرنامج .

3-6 طريقة كتابة البرنامج للمعالج

3-6-1 الشفرات الثنائية Binary codes

إن الناظر لأول وهلة فى البرنامج المكتوب بلغة الماكينة فى شكل (3-1) سيصاب بالذهول وسيقول : هل من المعقول أن أكتب كل هذا العدد من الوحايد والأصفار فى برنامج لا يتعدى الأربع خطوات ؟ بالطبع إن ذلك حق لأن كتابة البرامج بلغة الماكينة تصاحبها بعض العيوب والتى نوردها فيما يلى :

1. هذه البرامج تأخذ وقتا طويلا فى إدخالها للذاكرة لأننا نكتبها بت بعد بت .
2. مثل هذه البرامج من الصعب فهمها أو متابعتها أو تصحيح أى خطأ فيها وذلك لأن الأعداد الثنائية عبارة عن نمازج من الوحايد والأصفار التى يصعب التفريق بينها خاصة بعد فترة عمل طويلة مع هذه الأرقام .
3. شكل هذا البرنامج لا يعطى أى دلالة على الغرض منه ، على العكس من برامج الباسيك أو حتى الأسمبلى كما سنرى بعد قليل فإنه بعد نظرة فاحصة على البرنامج تستطيع أن تخبر ما الغرض منه .
4. من السهل أن يقع المبرمج فى الكثير من الأخطاء أثناء كتابة هذه البرامج ومن الصعب عليه جدا استخراج هذه الأخطاء فيما بعد .

كمثال على ذلك سنكتب البرنامج السابق الموجود فى شكل (3-1) مرة أخرى فى شكل (3-3) وبجانبه صورة منه تحتوى على خطأ معين وحاول استخراج هذا الخطأ !! ما رأيك الآن لو كان البرنامج مكونا من عشرات أو حتى مئات من السطور هل تستطيع استخراج أخطائه كلها ؟ لاحظ أنك فى شكل (3-3) أمامك الصورة الصحيحة والصورة الخطأ وأنت فقط تقارن الاثنين ولكن عادة فى الوضع الحقيقى فإنه لن تكون أمامك الصورة الصحيحة للبرنامج ولكنك أخبرت بأن البرنامج به خطأ وعليك استخراجه ، بالطبع فإن هذه ستكون عملية شاقة .

3-6-2 الشفرات الستعشرية Hexadecimal codes

من الممكن تسهيل عملية كتابة البرامج بلغة الماكينة عن طريق استخدام نظام آخر غير النظام الثنائى وليكن مثلا النظام الستعشرى أو النظام الثمانى وسنعرض هنا للنظام الستعشرى فقط على أساس أنه الأكثر شيوعا وأنه الأسهل فى عملية الكتابة لأن عدد خانات العدد بالنظام الستعشرى تكون عادة أقل منها فى النظام الثمانى .

00111010	00111010
01100000	01100000
00000000	00000000
01000111	01000111
01110010	00111010
01100001	01100001
00000000	00000000
10000000	10000000
00110010	00110010
00110010	00110010
00000000	00000000

شكل (3-3) صعوبة استخراج الأخطاء فى ظل الكتابة بلغة الماكينة

شكل (3-4) يبين برنامج الجمع السابق وقد تمت كتابته هذه المرة بالنظام الستعشرى . من هذا الشكل نلاحظ أن عملية كتابة البرامج باستخدام النظام الستعشرى وكذلك عملية فحص البرنامج واستخراج الأخطاء منه ستكون أسهل بكثير من استخدام النظام الثنائى فى ذلك . المشكلة الآن هى أنه كما سبق وذكرنا أن المعالج لا يعرف سوى الإشارات المكتوبة بالنظام الثنائى فقط (وحايد وأصفار) فما هو الحل وقد كتبنا البرنامج بالنظام الستعشرى كما هو فى شكل (3-4) ؟ إن الحل لهذه المشكلة هو تحويل هذه الأوامر من الصورة الستعشرية إلى الصورة الثنائية قبل أن يتم إدخالها فى الذاكرة !! ولكن من سيقوم بعملية التحويل هذه ؟ بالطبع إذا قام بها المستخدم فقد رجعنا إلى المشكلة الأولى وذلك لصعوبَة عملية التحويل . إن هذه المهمة ، مهمة التحويل من الصورة الستعشرية إلى الصورة الثنائية ، هى مهمة سهلة جدا لأن يقوم بها المعالج نفسه عن طريق كتابة برنامج بلغة الماكينة (فى النظام الثنائى) يتلقى الأوامر من المستخدم بالنظام الستعشرى ثم يقوم هو (البرنامج) بتحويلها إلى النظام الثنائى وتحميلها

في الذاكرة . إن هذا البرنامج يسمى "محمل النظام الستعشرى" hexadecimal loader .

لقد حل النظام الستعشرى مشكلة صعوبة كتابة البرامج واستخراج الأخطاء منها إلى حد ما ، ولكن بقيت المشكلة الأخرى وهى أننا ما زلنا نتعامل مع أرقام صماء كشفرات للأوامر لا تحمل أى دلالة عن ماذا يفعل هذا الأمر أو ذاك . فمثلا الرقم 3A هو شفرة لأمر معين ولكننا لا نستطيع مثلا أن نميز ذلك الأمر فقد يكون هذا الرقم مثلا جزءا من عنوان كالأرقام 60, 61, 62 وغيرها ، وحتى إذا عرفنا أنه شفرة لأمر فلن نستطيع معرفة ماذا يفعل هذا الأمر إلا إذا رجعنا إلى كتالوج خاص بذلك .

3A
60
00
47
3A
61
00
80
32
62
00

شكل (3–4) نفس البرنامج الموجود فى شكل (3–3) ولكنه مكتوب بالنظام الستعشرى

3–6–3 الشفرات الحرفية Mnemonics codes

إنها لفكرة عظيمة لو أننا فهمنا المقصود من كل أمر من الأوامر وأعطينا كل واحدا منها كودا أو شفرة مكونة من ثلاثة أو أربعة أحرف على الأكثر على أن تكون هذه الأحرف من الأحرف الأبجدية التى تدل تقريبا على ما يقوم به المعالج عند تنفيذ هذا الأمر . فمثلا أمر الجمع يكون ADD التى هى اختصارا لكلمة Addition يعنى جمع ، وأمر الطرح يكون SUB وجاءت من Subtraction بمعنى طرح .... وهكذا مع باقى الأوامر كما سنرى فيما بعد . إن هذه الاختصارات هى ما يسمى بلغة الأسمبلى Assembly language أو أحيانا تسمى Mnemonics codes بمعنى الشفرات التى من السهل تذكرها حيث كلمة mnemonics تعنى المساعد لعملية التذكر ، وهى كذلك فى الحقيقة ، إذ الآن

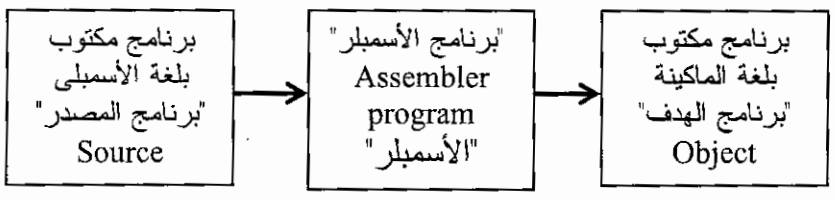
بوضع الأوامر فى هذه الصورة الحرفية أصبح من السهل تذكرها بل ومن السهل أن تخبر ماذا يفعل الأمر بمجرد النظر إليه . ما أروعها لو أن العرب قد سبقوا فى هذا المجال وفرضوا الكلمات طرح وجمع بدلا من SUB و ADD .....!
يقوم كل صانع لشريحة من شرائح المعالج بتزويدها بقائمة أو كتالوج يحتوى كل هذه الاختصارات الحرفية mnemonics ، ولذلك فإنك ستجد أن اختصارات كل شركة منتجة تختلف عن اختصارات الشركات الأخرى وسوف نرى ذلك فى الفصول القادمة إن شاء الله . إن أى برنامج مكتوب بهذه الاختصارات يقال عنه أنه مكتوب بلغة الأسمبلى . شكل (3-5) يبين البرنامج الموجود فى شكل (3-4) والذى سبق كتابته بالشفرات الستعشرية وقد كتبت جميع أوامره هذه المرة بلغة الأسمبلى لأحد المعالجات ، انظر لهذا البرنامج وحاول توقع ماذا يفعل كل أمر من هذه الأوامر قبل أن ندرسها بالتفصيل .
نحن هنا أيضا أمام مشكلة ترجمة هذه الشفرات الحرفية mnemonics التى لا يستطيع المعالج التعرف عليها إلى شفرات ثنائية يعرفها المعالج ، وكما هى الحال مع الشفرات الستعشرية فإننا سنترك أمر هذه الترجمة ليقوم بها المعالج نفسه عن طريق برنامج مكتوب لهذا الغرض يقوم المعالج بتنفيذه فيحول هذه الشفرات الحرفية إلى الشفرات الثنائية المطلوبة . هذا البرنامج اسمه الأسمبلر Assembler . على ذلك نستطيع القول أن الأسمبلر هو برنامج مكتوب بلغة الماكينة يقوم بتحويل البرنامج المكتوب بلغة الأسمبلى (الشفرات الحرفية) إلى برنامج مكتوب بلغة الماكينة . عادة يطلق على البرنامج المكتوب بلغة الأسمبلى "برنامج المصدر" source program والبرنامج المكتوب بلغة الماكينة "برنامج الهدف" object program . شكل (3-6) يبين رسما توضيحيا للدور الذى يقوم به الأسمبلر .
إن المقابل الذى يدفعه المستخدم نتيجة استخدامه للغة الأسمبلى يكون أولا فى كمية الذاكرة التى يشغلها برنامج الأسمبلر حيث أن هذا البرنامج لابد وأن يشغل كمية من الذاكرة الأساسية للميكروكومبيوتر وثانيا بعض التأخير الذى يحدث نتيجة الوقت الذى يأخذه الأسمبلر فى عملية الترجمة . وكما نعرف فإنه ليس هناك شىء كامل على الإطلاق ، لذلك فإننا نستطيع أن نلخص بعض عيوب لغة الأسمبلى فيما يلى :

1. مازالت هذه الاختصارات الحرفية غير كافية للدلالة على معنى الأوامر المختلفة حيث مازالت صيغ الأوامر بعيدة كل البعد عن اللغة العادية التى يستخدمها الإنسان وأيضا بالمقارنة بأوامر اللغات ذات المستوى العالى فإن لغة الأسمبلى تعتبر الأصعب فى التعلم .

2. لكى تستخدم هذه اللغة لابد من المعرفة الكاملة بمكونات المعالج ، المسجلات الموجودة بداخلة ، وطريقة المعالج فى التعامل مع الذاكرة وغير ذلك من الأمور الغير موجودة فى اللغات ذات المستوى العالى .
3. هذه اللغة كما ذكرنا من قبل تعتبر من اللغات التى تعتمد على الماكينة ، فأنت إذا كتبت برنامجا للمعالج Z80 فلن تستطيع استخدامه مع المعالج MC6800 مثلا .

```
LDA
60
00
MOV B,A
LDA
61
00
ADD B
STA
62
00
```

شكل (3-5) البرنامج الموجود فى شكل (3-4) وقد كتب هذه المرة بلغة الأسمبلى



شكل (3. 6) مهمة الأسمبلر

3-7 اللغات ذات المستوى العالى High level languages

لقد تم التغلب على الكثير من الصعوبات والعيوب المصاحبة للغة الأسمبلى باستخدام اللغات ذات المستوى العالى . إن كل أمر من أوامر أى لغة من هذه اللغات يدل أو يقوم بعملية مركبة على عكس لغة الأسمبلى فإن كل أمر فيها يقوم بعملية أولية . فمثلا برنامج الجمع السابق الذى يجمع رقمين موجودين فى الذاكرة والذى تمت كتابته فى أحد عشر سطرا باستخدام لغة الأسمبلى يمكن كتابته فى سطر واحد باستخدام لغة الباسيك كما يلى :

SUM = NUM1 + NUM2

لذلك فإن أى أمر من أوامر أى لغة من اللغات ذات المستوى العالى هو فى الحقيقة مجموعة من أوامر لغة الأسمبلى . إن ما يقوم به برنامج الأسمبلر فى حالة لغة الأسمبلى يقوم به برنامج آخر يسمى "برنامج المؤلف" أو الجامع أو المصنف compiler program فى حالة اللغات ذات المستوى العالى ، حيث يقوم هذا المؤلف بترجمة الأوامر المكتوبة باللغات ذات المستوى العالى إلى لغة الماكينة أو الشفرات الثنائية التي يقبلها المعالج . هذه اللغات ليست موضوع دراستنا فى هذا الكتاب لذلك سنكتفى بهذا القدر من الكلام عنها .

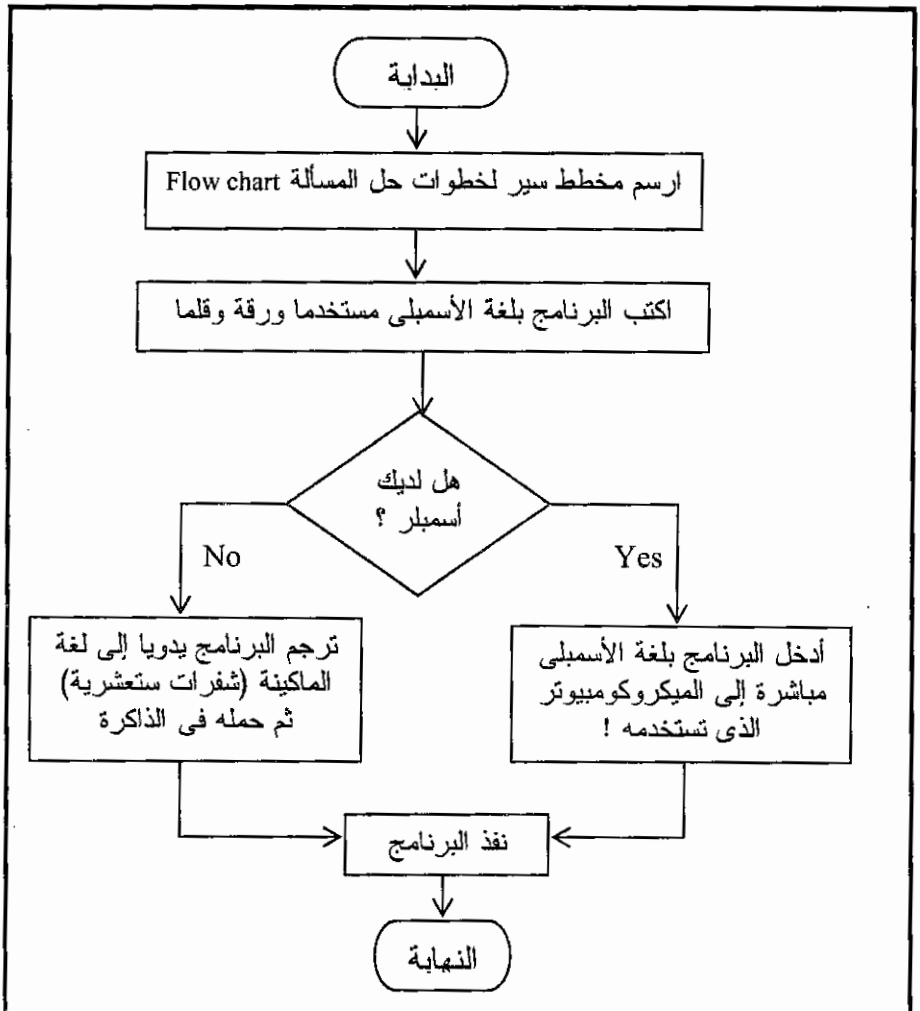
3-8 خطوات كتابة برنامج بلغة الأسمبلى

الآن وقد عرفنا الفرق بين لغة الماكينة ولغة الأسمبلى فبأى صورة سنكتب برامجنا ؟ الإجابة على ذلك ستتوقف على إمكانيات الميكروكومبيوتر الذى تتعامل معه سواء فى بيتك أو فى معملك . إذا كان لديك برنامج الأسمبلر فإنه بالطبع من الأفضل أن تكتب برامجك بلغة الأسمبلى على الحاسب بنفس الشكل ثم تطلب من الأسمبلر أن يقوم بعملية الترجمة وإدخال البرنامج إلى الذاكرة ، أما إذا لم يكن لديك أسمبلر فليس أمامك من خيار سوى الكتابة بلغة الماكينة أو الشفرات الستعشرية فى الذاكرة مباشرة . شكل (3-7) يبين خريطة تدفق أو مخطط سير flow chart لخطوات حل أى مشكلة باستخدام المعالج .

السؤال الآن فى أى مكان فى الذاكرة سنضع البرنامج ؟

للإجابة عن هذا السؤال يجب أن نتصور الذاكرة الخاصة بالجهاز أو الميكروكومبيوتر الذى نستخدمه وقد قسمت إلى ثلاثة أجزاء كالموضحة فى شكل (3-8) . الجزء الأول منها هو ROM أو ذاكرة القراءة فقط وهذه كما ذكرنا فى الفصل الأول لا يمكن للمستخدم أن يسجل فيها أى شىء . الجزء الثانى من الذاكرة هى RAM وهى الجزء الذى يمكن للمستخدم أن يتعامل معه ويجب أن نعلم أن نظام التشغيل الخاص بالجهاز الذى نستخدمه يحجز جزءا من

هذه RAM للاستعمال الخاص به والجزء المتبقى يمكن للمبرمج أن يستخدمه . لذلك يجب قبل أن تبدأ فى كتابة برنامجك بلغة الأسمبلى وإدخاله فى الذاكرة أن تعرف أين يقع جزء RAM الخاص بنظام التشغيل حتى لا يتداخل البرنامج الخاص بك معه . إن ذلك يتطلب إلقاء نظرة على ما يسمى بخريطة الذاكرة

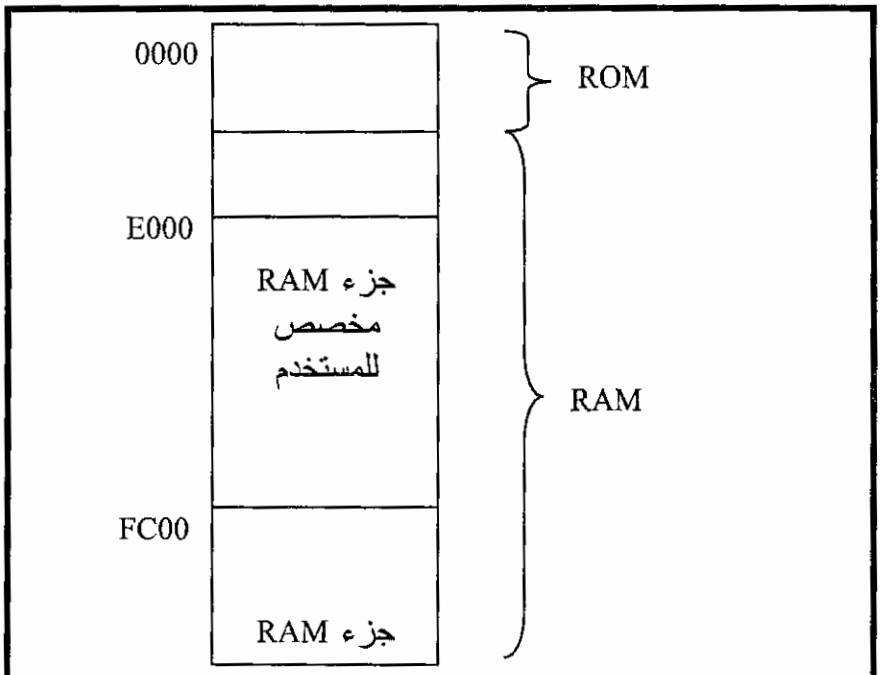


شكل (3– 7) خريطة تدفق لخطوات برمجة أى مسألة لأى معالج

الخاصة بالميكروكومبيوتر الذى تستخدمه . هذه الخريطة تعتبر شكلا توضيحيا يبين الذاكرة بأكملها من الأول حتى آخر بايت وقد قسمت إلى أجزاء مع التعريف بكل جزء فيما يستخدم وهل هو مشغول أم لا . الآن وقد عرفت الجزء من RAM الذى يمكنك أن تضع فيه برنامجك يجب عليك كمبرمج أن تقسم هذا

الجزء إلى جزأين أيضا ، أحدهما تكتب فيه البرنامج والآخر تخصصه للبيانات التى يحتاجها أو يخرجها البرنامج .

إن عملية تقسيم الذاكرة إلى جزء للبرنامج وآخر للبيانات عملية تعتمد على المبرمج بالدرجة الأولى وعلى البرنامج أيضا ، فقد يكون البرنامج لا يحتاج إلى بيانات أو لا يخرج بيانات على الإطلاق ، فى هذه الحالة فإن كل ذاكرة الكتابة ستكون للبرنامج ، وقد يكون البرنامج ينتج أو يحتاج للكثير من البيانات ، فى هذه الحالة يجب حجز جزء كاف لهذه البيانات .



شكل (3-8) خريطة الذاكرة للميكروكومبيوتر المستخدم خلال هذا الكتاب

مثال 3-1

لدينا مجموعة من الأرقام وليكن عددها 50H رقما ، والمطلوب استخراج أكبر عدد فى هذه المجموعة . أين سنكتب البرنامج ؟ وأين سنكتب البيانات (الخمسين رقما) ؟

كما نرى من شكل (3-8) فإن جزء ذاكرة الكتابة المخصص للمستخدم فى الميكروكومبيوتر الذى نستخدمه يبدأ من العنوان E000H وينتهى عند العنوان

FC00H . هذه المساحة من RAM يجب أن نقسمها إلى جزء للبرنامج وجزء للبيانات . البيانات المطلوبة هى 50H رقما ويمكن لنا أن نضعها ابتداء من العنوان E100H حتى العنوان E150H . أما البرنامج فيمكن لنا أن نكتبه مثلا ابتداء من العنوان E000H على أساس أنّ البايتات التى سيحتاجها البرنامج نفسه لن يصل عددها إلى 100H بايت بأى حال من الأحوال وإلا لو زاد عددها عن ذلك فسوف تتداخل مع البيانات عند العنوان E100H وفى هذه الحالة يجب أن نزيح البيانات بعيدا أو نكتب البرنامج بعد البيانات .

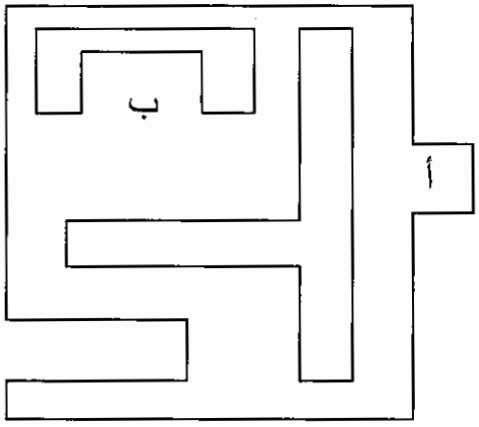
عملية تقسيم الذاكرة الموضحة فى شكل (3-8) يسمى فى مصطلحات الحاسبات بخريطة الذاكرة حيث يبين هذا الشكل جميع الذاكرة الملحقة بالجهاز الذى تستخدمه وفيما يستخدم كل جزء منها ، وهذه الخريطة يجب على أى مستخدم للغة الأسمبلى أن يتعرف عليها جيدا بالنسبة للحاسب الذى سيستخدمه ، والتقسيم الموضح فى شكل (3-8) سيكون هو التقسيم الذى سنتبعه خلال هذا الكتاب وبالطبع فإن هذه الخريطة تختلف باختلاف الجهاز المستخدم فيجب مراعاة ذلك .

3-9 تمارين

1. ما هو الأمر ؟
2. ما هو البرنامج ؟
3. تخيل أنك تركب سيارة آلية تريد الانتقال بها من المكان (أ) إلى المكان (ب) كما فى شكل (3-9) هذه السيارة عند كل تقاطع لابد أن تأخذ قرارا من أربعة تحدد اتجاه حركتها كما فى الجدول الموضح فى نفس الشكل . اكتب برنامجا لهذه السيارة بالقرارات ثم بالشفرات الثنائية بحيث عندما تنفذه تنتقل السيارة من المكان (أ) إلى المكان (ب) فى الشكل .
4. هل يمكنك إعطاء اسم للغة الشفرات الثنائية المكتوب بها البرنامج السابق ؟
5. ماذا يحدث لو جعلنا شفرة القرار تتكون من 3 بتات بدلا من 2 ؟ بالطبع فى هذه الحالة سيكون هناك إمكانية لعدد أكبر من القرارات يصل إلى 8 ، افترض هذه القرارات من عندك محاولا تطوير هذه السيارة ؟
6. اشرح باستخدام خريطة تدفق كيف يقوم المعالج بتنفيذ أى برنامج ؟
7. ما هى عيوب كتابة البرامج بالشفرات الثنائية ؟
8. أعد كتابة البرنامج الموجود فى شكل (3-1) مستخدما النظام الثمانى ؟ كم عدد ضربات المفاتيح التى ستنفذها لكى تكتب البرنامج مستخدما هذا النظام ؟ ما نوع المحمل loader الذى ستحتاج إليه فى هذه الحالة ؟ أيهما أفضل فى الكتابة وسهولة استخراج الأخطاء ، النظام الثمانى أم النظام الستعشرى ؟

9. اشرح فائدة استخدام الأسمبلر ؟
10. مطلوب كتابة برنامج يجمع 20 رقما مخزنة فى الذاكرة ، ارسم خريطة للذاكرة تبين عليها أين ستكتب البرنامج وأين ستوضع البيانات ؟
11. حاول الحصول على خريطة الذاكرة للميكروكومبيوتر الذى تتعامل معه وادرسها وطبق عليها المسألة السابقة ؟

القرار	الشفرة
أمام	00
يمين	01
شمال	10
توقف	11



شكل (3-9) تمرين 3