الفصل الرابع عشر 14

التركيب الهيكلى للمعالج
intel8086/8088

14-1 مقدمة

نقدم فى هذا الفصل شرحا تفصيليا للتركيب الهيكلي للمعالج intel8086 وزميله المعالج intel8088 وذلك من الداخل حيث سندرس محتوياتهما من المسجلات والعدادات ومن الخارج حيث سنلقى نظرة سريعة على وظيفة كل طرف من أطرافهما . إن هذين المعالجين يعتبران أهم المعالجات ذات ال 16 بت وذلك لاستخدامهما فى الحاسب IBM الذى كان من أول الحاسبات الشخصية التى ظهرت فى السوق ثم سادت وفرضت نفسها على كل المستخدمين للحاسبات . من ضمن هذا الجيل من المعالجات ظهر أيضا المعالج MC68000 والمعالج Z8000 ولكن كما ذكرنا كان أكثرها شيوعا وأوفرها حظا فى الاستخدام هو المعالج intel8086 والذى نحن بصدد دراسته هنا .

المعالجين 8086/8088 كل منهما كما رأينا يتعامل مع بيانات مقدارها 16 بت ، ولكن هناك نقطة خلاف أساسية ووحيدة بين هذين المعالجين وهى كيفية التعامل مع هذه البيانات . إن المعالج 8086 يتعامل مع البيانات فى داخله وفى خارجه على أساس أنها 16 بت ، أى أنه له مسار بيانات خارجى مقداره 16 بت يستطيع من خلاله التعامل مع الأجهزة الخارجية مثل الشاشة والذاكرة ولوحة المفاتيح على أساس 16 بت ، فيرسل مثلا معلومة من 16 بت إلى الشاشة فى مرة واحدة ، ويستقبل معلومة من 16 بت من أى بوابة إدخال . هذا المعالج ، 8086 يتعامل داخليا أيضا بنفس الطريقة حيث تنتقل البيانات داخليا بين جميع المسجلات بعضها البعض أو مع وحدة الحساب والمنطق على مسار بيانات مقداره 16 بت أيضا . أما المعالج 8088 فإنه تماما مثل نظيره 8086 فى التعاملات الداخلية حيث ينقل البيانات داخله على مسار بيانات مقداره 16 بت ، بينما يختلف عن نظيره 8086 فى التعاملات الخارجية حيث أن مسار البيانات الخارجى له يتكون من 8 بت فقط ، لذلك فإنه يرسل أو يستقبل بيانات 8 بت فقط مع الأجهزة الخارجية ، وهذه كما ذكرنا هى نقطة الخلاف الوحيدة بين المعالجين كما سنرى . ونعتقد أن الرقم 6 فى المعالج 8086 يذكرنا بأنه يتعامل دائما من خلال مسار بيانات 16 بت ، بينما الرقم 8 الأخير فى المعالج 8088 فإنه يذكرنا بأن التعامل يكون من خلال مسار بيانات 8 بت فقط مع الأجهزة الخارجية . ويخطر ببالنا سؤال مهم هنا وهو: ما هو الداعى للمعالج 8088 إذا كان نظيره 8086 قد قام بالمهمة بكفاءة أحسن وبالتأكيد أسهل حيث يتعامل خارجيا وداخليا من خلال مسار بيانات 16 بت ؟ والإجابة على ذلك هى أن المعالجات 16 بت ظهرت فى منتصف الثمانينات وحلت محل المعالجات 8 بت فى جميع الحاسبات

وجميع التطبيقات وبعد أن كانت المعالجات 8 بت قد انتشرت فى السوق واستخدمت فى كثير من أجهزة الحاسبات وفى الكثير من التطبيقات أيضا مما تسبب فى أن كل هذه الأجهزة القديمة (8 بت) أصبحت عديمة الفائدة بعد أن فكر كل المستهلكين فى الأخذ بالمعالجات الجديدة (16 بت) . لذلك كان التفكير فى معالج وسيط يقلل من حجم الخسارة فى عملية الانتقال إلى المعالج 8086 ، فكان الحل هو معالج يتعامل داخليا على أساس 16 بت للاستفادة بمميزات ال16 بت ويتعامل خارجيا على أساس 8 بت لتسهيل عملية المواجهة مع الأجهزة الخارجية الموجودة أصلا فى الحاسبات والتطبيقات التى كانت تتعامل مع المعالجات 8 بت وبأقل تكلفة ممكنة ، فكان ذلك المعالج الوسيط هو المعالج 8088 ، ولقد نزلت فى هذا الوقت أيضا الكثير من البرامج التى تقوم بتحويل برمجيات المعالجات 8 بت إلى صورة تناسب المعالجات 16 بت كمرحلة انتقال وحتى لا يعاد صياغة هذه البرمجيات من جديد .

14-2 نظرة داخلية على محتويات المعالجين 8086/8088

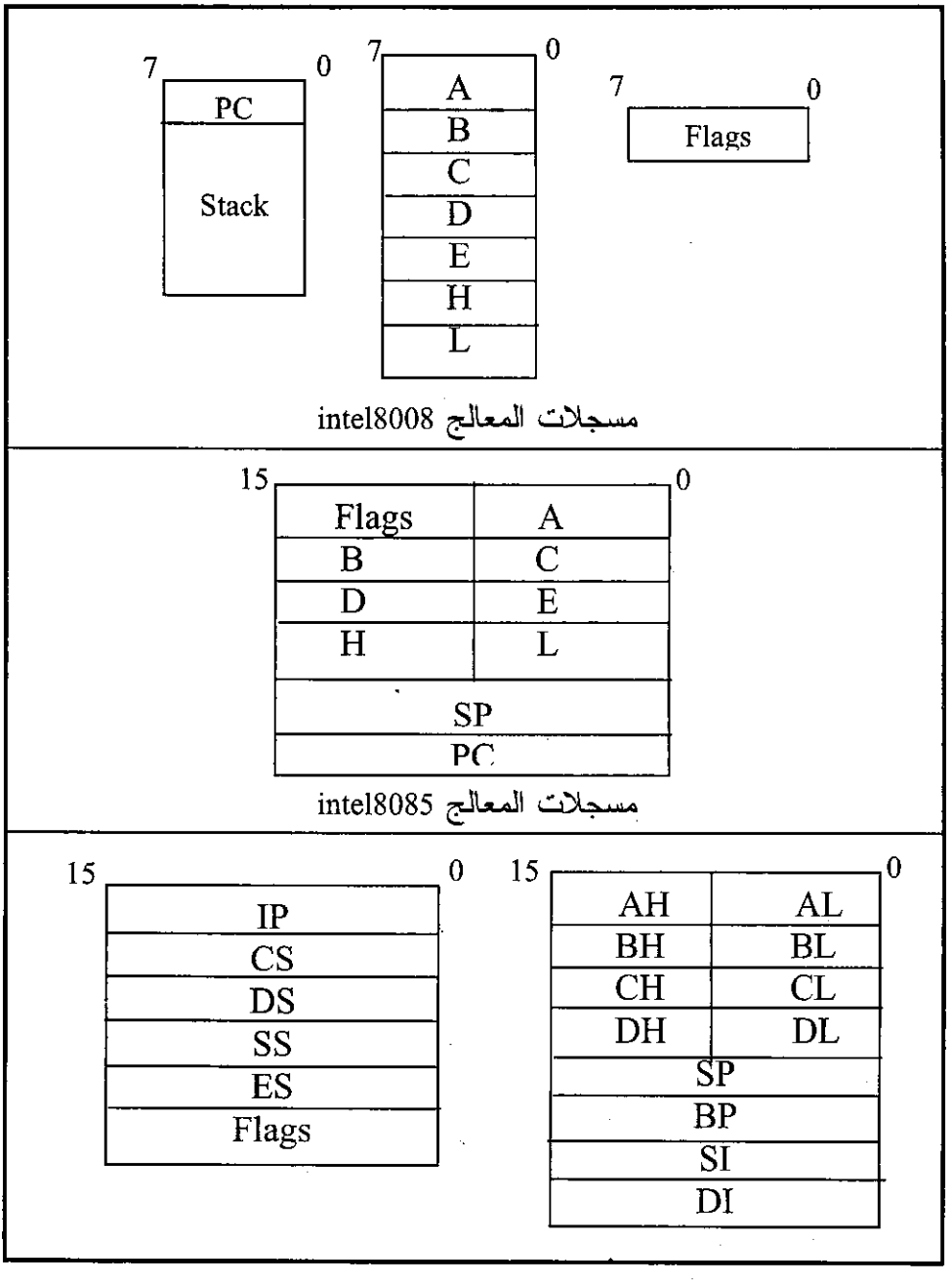
كما علمنا من قبل وعند دراستنا لشرائح المعالجات 8 بت أن أى معالج فى النهاية يمكن النظر إليه على أنه مجموعة من المسجلات والعدادات بجانب وحدة الحساب والمنطق حيث أنها أهم المكونات الداخلية فى المعالج ، أما فكرة عمل المعالج ، أى معالج ، فهى (كما سبق وشرحناها أيضا) إحضار شفرات الأوامر من الذاكرة وتنفيذها بنفس التتابع المسجلة به فى البرنامج . أى أن الفكرة ثابتة ولكن التطور يكون دائما فى المكونات حيث تتغير المسجلات ووحدة الحساب والمنطق من 8 بت إلى 16 بت إلى 32 بت إلى 64 بت وتتغير تكنولوجيا التصنيع نفسها مع الزمن فتزداد السرعة بدرجة كبيرة ولكن فكرة العمل تظل كما هى ثابتة . شكل (14-1) يبين تطور المسجلات فى المعالجات 8008 و 8085 و 8086 . نلاحظ من هذا الشكل أن عدد المسجلات كان 6 مسجلات كل منها 8 بت بخلاف المركم فى المعالج 8008 وأما المعالج 8085 فيحتوى نفس العدد من المسجلات ولكن الجديد هو أن هذه المسجلات يمكن فى بعض العمليات ازدواجها واستخدامها كمسجلات 16 بت كما رأينا سابقا ولكن المركم كما هو 8 بت ، وأما فى المعالج 8086 فإنه يحتوى أيضا نفس العدد من المسجلات العامة والتى اختلفت أسماؤها قليلا لتناسب استخدامها كمسجلات 16 بت أو 8 بت ، فمثلا المسجل B أصبح اسمه BX فى حالة استخدامه كمسجل 16 بت أو BL فى حالة استخدام النصف الأول منه كمسجل 8 بت و BH فى حالة استخدام النصف

الأعلى منه كمسجل 8 بت ، نفس الكلام مطبق على باقى المسجلات العامة وهى المسجلات C, D . الجديد أيضا أن المركم مطبق عليه نفس الكلام السابق فيمكن استخدامه كمسجل 16 بت (AX) أو مسجلين كل منهم 8 بت (AL و AH) . نلاحظ أيضا أن المكدسة stack كانت موجودة بداخل المعالج 8008 ثم انتقلت لتصبح جزءا من الذاكرة يشار إلى قمتها أو أول مكان فاضى فيها بمحتويات المسجل SP أو مؤشر المكدسة وذلك فى المعالجات 8085 و 8086 . أما عداد البرنامج PC فكان 8 بت فى المعالج 8008 وأصبح 16 بتا فى المعالجات 8085 و 8086 وأصبح اسمه مؤشر الأوامر IP فى المعالج 8086 وهناك فرق كبير بين الاسم "عداد البرنامج" و"مؤشر الأوامر" بالرغم من التماثل فى الوظيفة ولكن فى حالة المعالج 8085 فإنه يتعامل مع ذاكرة مقدارها 64 كيلو بايت وأما فى حالة المعالج 8086 فإنه يتعامل مع ذاكرة مقدارها 1 ميجابايت من خلال فكرة زكية وهى فكرة تجزىء الذاكرة التى سنشرحها بعد قليل إن شاء الله . نلاحظ أيضا من شكل (14-1) أن المعالج 8086 يحتوى على مسجلات أخرى لم تكن موجودة فى سابقيه وهى المسجلات BP, SI, DI, CS, DS, SS, ES وكلها مسجلات 16 بت سنتعرف على وظيفة كل منها بعد قليل . نلاحظ أيضا أن مسجل الأعلام أصبح 16 بت أيضا بدلا من ثمانية مما ينبئ بأنه سيكون هناك الكثير من الأعلام وبالتالى مقدرة أكثر على البرمجة وعدد أكثر من الأوامر . من الملاحظات المهمة أيضا فى شكل (14-1) هى احتواء المعالج 8008 على المكدسة كمجموعة من المسجلات موجودة بداخل المعالج نفسه فى حين أصبحت هذه المكدسة جزء من الذاكرة فى المعالجات التى تلت ذلك مما أمكن معه تكبير المكدسة لأى كمية مطلوبة .

14-3 نظرة تفصيلية على مسجلات المعالج 8086/8088

لقد استخدم مصممو المعالجين 8086/8088 فكرة زكية كان لها أكبر الأثر فى زيادة سرعة وكفاءة هذين المعالجين وهذه الفكرة هى انقسام هذه المعالجات إلى وحدتين أساسيتين لكل منهما وظيفة مختلفة تماما عن الوحدة الأخرى .
الوحدة الأولى هى وحدة التنفيذ Execution Unit (EU) وهى خاصة فقط بتنفيذ الأوامر ولا تتعداها لأى وظيفة أخرى ، والوحدة الثانية هى وحدة مواجهة المسارات Bus Interface Unit (BIU) وهذه أيضا لها وظيفة محددة وهى جلب الأوامر من الذاكرة ووضعها فى طابور أو قائمة انتظار queue فى انتظار التنفيذ عن طريق وحدة التنفيذ . هذا التقسيم فى الوظائف بين الوحدتين أتاح لوحدة التنفيذ أن تقوم فقط بتنفيذ الأوامر الموجودة فى قائمة الانتظار وفى أثناء انشغال

وحدة التنفيذ بتنفيذ الأوامر تقوم وحدة مواجهة المسارات بجلب أوامر أخرى من الذاكرة ووضعها فى القائمة والعمل على أن تكون القائمة مملوءة دائما بالأوامر التى فى انتظار التنفيذ .



مسجلات المعالج intel8086

شكل (14-1) تطور محتويات المعالجات 8008 و 8085 و 8086

بذلك تم توفير وقت كبير كانت وحدة التنفيذ تتوقف فيه لحين الذهاب إلى الذاكرة وإحضار الأوامر التى عليها الدور فى التنفيذ مما كان له أكبر الأثر فى زيادة سرعة هذه المعالجات بدرجة كبيرة . يمكن تمثيل مهمة كل من هاتين الوحدتين بمهمة المدير والسكرتارية حيث تكون السكرتارية هى المواجهة للعالم الخارجى حيث تستقبل هى جميع الطلبات والمشاكل من الجمهور واعدادها فى ملف كل على حسب ترتيب قدومه ثم تعرض هذا الملف على المدير الذى يقوم بحل هذه المشاكل فى حين تعمل السكرتارية على استقبال الطلبات الأخرى لحين أن ينتهى المدير من تنفيذ ما معه من طلبات . بالطبع فإن ذلك يكون له أكبر الأثر فى سرعة تنفيذ الطلبات عن ما لو كان المدير وحده يقوم باستقبال كمية من الطلبات ثم يجلس لتنفيذها وبعد أن ينتهى من تنفيذ هذه الكمية يقوم ليستقبل كمية أخرى وهكذا .

بعض المسجلات داخل المعالج 8086 تتبع وحدة التنفيذ والبعض الآخر يتبع وحدة المواجهة على حسب وظيفة كل مسجل من هذه المسجلات حيث يجب أن نتوقع أن جميع المسجلات العامة DX, CX, BX, AX ومسجل الأعلام والمسجلات SP DI, SI, BP, كلها تتبع وحدة التنفيذ وأما المسجلات ES, SS, DS, CS ومؤشر الأوامر IP فتتبع وحدة المواجهة تبعا لوظيفة كل منها كما سنرى .

على ضوء ما ذكرنا فى المقدمة عن الفرق بين المعالجين 8086 و 8088 فإننا يجب أن نتوقع أن وحدة التنفيذ EU فى كل من المعالجين ستكون نفسها تماما وأما وحدة المواجهة BIU فستختلف فى المعالج 8086 عنها فى المعالج 8088 حيث أنها فى الأول ستتعامل مع مسار بيانات 16 بت بينما ستتعامل مع مسار بيانات 8 بتات فى المعالج الثانى وهذا هو وجه الاختلاف الأساسى بينهما .

14-3-1 المسجلات عامة الأغراض

يحتوى المعالج 8086 على أربع مسجلات عامة الأغراض كل منها 16 بتا وهى المسجلات DX, CX, BX, AX . كل واحد من هذه المسجلات يمكن التعامل معها على أنها مسجلين كل منهم 8 بتات أو مسجل واحد 16 بت . فى حالة التعامل معها على أنها مسجلات 8 بتات فإن النصف الأدنى أو ذو القيمة الصغرى Low significant half يرمز له دائما بالرموز التالية DL, CL, BL,

AL أما النصف الأعلى أو ذو القيمة العليا High significant half فيرمز له دائما بالرموز التالية DH, CH, BH, AH . لذلك عند وضع معلومة من 16 بت مثل الرقم C351H فى المسجل BX مثلا فإن النصف الأعلى من المعلومة وهو C3 يوضع فى النصف الأعلى من المسجل وهو BH وأما النصف الأدنى من المعلومة وهو 51 فيوضع فى النصف الأدنى من المسجل وهو BL . فيما يلى سنلقى نظرة سريعة على وظيفة كل مسجل من هذه المسجلات :

المسجل AX

المسجل AX هو المركم accumulator وكما سنرى عند دراستنا للغة الأسمبلى للمعالج 8086 فإن المركم لن تكون له نفس الأهمية التى رأيناها عند دراستنا للمعالجات 8 بت ، حيث هنا سنرى أنه يمكن إجراء أى عملية حسابية أو منطقية على أى مسجلين مع بعضهما وليس من الضرورى أن يكون المركم واحدا منهما ، كما أن نتيجة هذه العملية تكون دائما فى المسجل الأول فى الأمر ، فمثلا الأوامر التالية كلها صحيحة :

ADD AX, BX

حيث سيجمع محتويات المسجل AX مع المسجل BX ويضع النتيجة فى المسجل AX الذى هو المركم .

ADD CX, BX

حيث سيجمع محتويات المسجل CX مع المسجل BX ويضع النتيجة فى المسجل CX ، وهكذا . هذا ولا زالت عمليات الإدخال والإخراج باستخدام الأمرين IN و OUT تتم عن طريق المركم كما هو الحال فى المعالجات 8 بت ولكن بإمكانيات أكثر وكفاءة أحسن كما سنرى عند الدراسة التفصيلية لهذه الأوامر .

المسجل BX

إن المسجل BX بجانب كونه أحد المسجلات العامة التى تستخدم فى كل أغراض البرمجة مثل العمليات الحسابية والمنطقية وعمليات الإزاحة والدوران وغيرها فإن له وظيفة أخرى محددة وخاصة به عند تنفيذ بعض الأوامر مثل الأمر XLAT والذى ينشئ جدولا فى الذاكرة أول عنوان فيه هو الموجود فى المسجل BX وتتكون عناصر هذا الجدول بتخزين محتويات النصف الأدنى من المركم AL فى عناوين متتالية فى الذاكرة تتكون بجمع محتويات المسجل AL مع محتويات المسجل BX كما سنرى عند شرح أوامر لغة الأسمبلى فيما بعد . لذلك فإن المسجل BX يحتوى عنوان البداية أو القاعدة Base للجدول الذى يتكون بالأمر XLAT . كما أن المسجل BX يستخدم فى أغراض العنونة غير المباشرة حيث يمكن وضع العنوان المراد التعامل معه فى ذاكرة البيانات فيه .

المسجل CX

المسجل CX أيضا بجانب كونه أحد المسجلات العامة مثل المسجلين BX, AX فإن له أيضا مهمة محددة خاصة به عند تنفيذ بعض الأوامر . فمثلا عند تنفيذ الأمر LOOP والذى ينفذ حلقة أو مجموعة من الأوامر عدة مرات فإن عدد المرات المراد تنفيذها لهذه الحلقة يوضع فى المسجل CX . أى أنه عداد أو Counter للحلقات عند تنفيذ الأمر LOOP .

المسجل DX

هذا المسجل أيضا بجانب كونه أحد المسجلات العامة فله أيضا وظيفة محددة عند تنفيذ بعض الأوامر ، فعند تنفيذ أمر ضرب رقمين كل منهما 16 بت فإن النصف الأعلى most significant part من النتيجة يوضع فى هذا المسجل (لاحظ أن النتيجة ستكون 32 بت) . كذلك عند تنفيذ بعض أوامر الإدخال والإخراج فإن المسجل DX يوضع به عنوان البوابة المراد الإخراج عليها أو الإدخال منها . أى أن هذا المسجل DX أحد وظائفه الخاصة هى أنه يحتوى جزء من البيانات عند تنفيذ أوامر ضرب أو قسمة رقمين كل منهما 16 بتا . وعلى ذلك فإن الأربع مسجلات السابقة لها أسماء كما رأينا تطابق الرموز التى أطلقت عليها والوظيفة الخاصة المنوطة بكل واحد من هذه المسجلات والتى سنعيدها كما يلى:

Accumulator	AX	المركم
Base	BX	القاعدة
Counter	CX	العداد
Data	DX	البيانات

هناك أيضا مجموعة من المسجلات التى يمكن أن تدخل ضمن مجموعة المسجلات العامة حيث أنها تكون تحت تصرف المبرمج ولكنها لها وظيفة محددة أيضا فى عمليات البرمجة فهى تستخدم إما للإشارة إلى أماكن محددة فى الذاكرة Pointer أو تستخدم فى الفهرسة Index عند التعامل مع الذاكرة بهذه الطريقة . هذه المسجلات هى كالتالى :

مؤشر المكدسة أو المسجل Stack Pointer, SP

المكدسة stack هى جزء مقتطع من الذاكرة يخزن فيه عادة البيانات المهمة قبل القفز من البرنامج الأساسى إلى برنامج فرعى أو برنامج مقاطعة والتى ستكون هناك حاجة إليها عند العودة مرة ثانية إلى البرنامج الأساسى بعد إنهاء البرنامج الفرعى (انظر فصل البرامج الفرعية) أو الانتهاء من خدمة المقاطعة (انظر فصل المقاطعة) . من أهم هذه البيانات مثلا محتويات مؤشر الأوامر IP حتى يتسنى لنا العودة لنفس المكان الذى خرجنا منه فى البرنامج الأساسى وكذلك

محتويات أى مسجل آخر قد نخاف من ضياعها أو تغيرها عند الخروج من البرنامج الأساسى مثل مسجل الأعلام والمركم . هذه البيانات تخزن فى المكدسة بالترتيب ويتم استدعاؤها بنفس الترتيب على أساس أن آخر ما تم تخزينه يكون أول ما يتم استدعاؤه Last In First Out (LIFO) ولكى نعرف حدود هذه المكدسة فإن مسجل مؤشر المكدسة Stack Pointer (SP) يحتوى عنوان آخر مكان تم التخزين فيه فى هذه المكدسة .

مسجل مؤشر القاعدة Base Pointer, BP

أحد المسجلات العامة التى تستخدم لعنونة أو للإشارة على بداية مجموعة بيانات أو طابور array بيانات موجود فى المكدسة stack .

مسجلى الفهرسة SI, DI

يستخدمان فى عملية العنونة غير المباشرة (المفهرسة) فى الذاكرة indirect addressing كما سنرى عند دراسة طرق العنونة المختلفة .

14-3-2 المسجلات الخاصة

المسجلات الخاصة الموجودة فى المعالج 8086/8088 هى مسجل مؤشر الأوامر Instruction Pointer (IP) وأربع مسجلات خاصة بتجزىء الذاكرة سنطلق عليها اسم مسجلات التجزىء memory segmentation registers . عدد هذه المسجلات أربعة وهى : ES, SS, DS, CS . لكى نأخذ فكرة عن وظيفة هذه المسجلات ، لابد أن نعرف أولا ما هو المقصود بتجزىء الذاكرة ؟ ولماذا يتم تجزىء الذاكرة؟

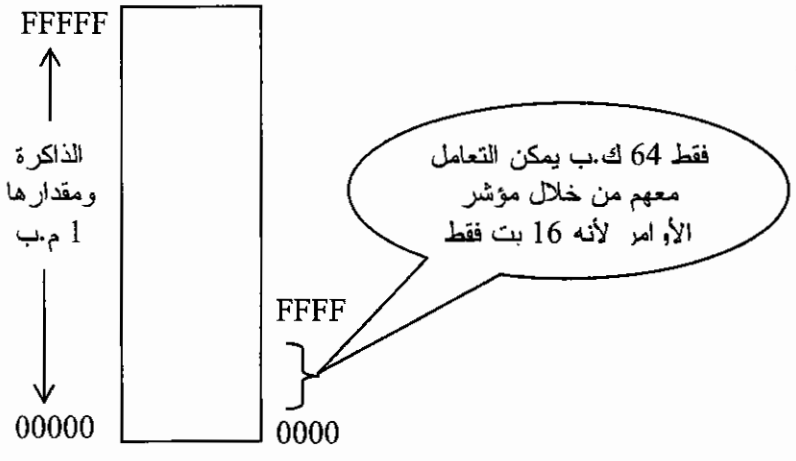
14-4 تجزىء الذاكرة Memory segmentation

مسار العناوين فى المعالج 8086/8088 يحتوى 20 بت أو بمعنى آخر يتكون من 20 خطا وهذا يعنى أن هذا المعالج يستطيع التعامل مع ذاكرة مقدارها 2^{20} أو 1048576 بايت أو اختصارا تكتب 1 ميجابايت (1 م.ب) . هذا يعنى أننا نستطيع مثلا أن نكتب أى برنامج فى خلال هذا المدى من الذاكرة والذى يبلغ 1 م.ب كما ذكرنا وعلى المعالج أن يحضر أوامر هذا البرنامج من الذاكرة وينفذها بالتتابع بلا أدنى مشاكل . إن هناك مشكلة صعبة تعوق المعالج من إمكانية إحضار الأوامر من الذاكرة بهذه السهولة وذلك لأن عداد البرنامج أو مؤشر الأوامر IP كما أسميناه هنا يحتوى على 16 بت فقط ، وكما نعلم أن مهمة المسجل IP هى أنه يحتوى عنوان الأمر الذى عليه الدور فى التنفيذ ، وهذا يعنى بالتالى أن أى أمر يقع فى الذاكرة خارج المدى العنوانى صفر إلى 64 ك.ب لن يستطيع المعالج إحضاره لأن مؤشر الأوامر IP يتكون من 16 بت فقط مما يعنى

أن المدى العنوانى الذى يستطيع المعالج التعامل معه من خلال هذا المسجل هو صفر إلى 2^{16} أى 65536 بايت فى الذاكرة . فما هو الحل لهذه المشكلة ونحن نريد كتابة البرامج فى أى مكان فى الذاكرة التى يبلغ مداها 1 م.ب وليس فقط فى أول 64 ك.ب ؟ شكل (14-2) يبين المدى العنوانى للمعالج 8086 على ضوء عدد خطوط مسار العناوين ، والمدى العنوانى الذى يمكن التعامل معه من خلال المسجل IP .

إن حل هذه المشكلة جاء من خلال استخدام فكرة زكية تمكنك كمبرمج من التعامل مع كل المدى العنوانى للذاكرة الذى يبلغ 1 م.ب بالرغم من استعمال مسجلات 16 بت فقط وكان ذلك من خلال استخدام أربع مسجلات سميت بمسجلات تجزىء الذاكرة memory segmentation registers وكل منها 16 بت ويرمز لها بالرموز التالية ES, SS, DS, CS وهذه الرموز لها دلالات تتطابق مع وظيفة كل مسجل سنعرفها بعد قليل .

كل واحد من هذه المسجلات ، مسجلات التجزيء ، يحتوى عنوان من 16 بت ولكن الظريف هنا أن هذه 16 بت تقابل أعلى 16 بت من مسار العناوين أى A4 إلى A19 وليس أول 16 بت A0 إلى A15 . أى أن محتويات أى واحد من هذه المسجلات لن تمثل عنوانا حقيقيا فى الذاكرة إلا بعد إزاحتها ناحية اليسار بمقدار 4بت أى بمقدار خانة ستعشرية أو بضربها فى الرقم 16 ضربا عشريا للحصول على عنوان من 20 بت .



شكل (14-2) الذاكرة التى يمكن أن يتعامل معها المعالج 8086 من خلال مسار عناوينه وتلك التى يمكن التعامل معها من خلال مؤشر الأوامر IP .

فمثلا بافتراض أن المسجل CS محتوياته كالتالى : CS=0800H فإن العنوان الفعلى المقابل لهذه المحتويات هو 08000H بإضافة 0H ناحية اليمين أى بإزاحة الرقم 4 بتات ناحية اليسار أو بضربه فى الرقم 16 ضربا عشريا . وكذلك إذا كانت محتويات المسجل DS كالتالى : DS=12F5H فإن العنوان الفعلى المقابل لهذه المحتويات هو 12F50H فى الذاكرة . هنا يظهر سؤال مهم وهو كيف يتم إحضار الأوامر من الذاكرة باستخدام مؤشر الأوامر IP الذى يتكون هو الآخر من 16 بت فقط ؟ وهل هذا المسجل له علاقة بمسجلات التجزىء ؟

بفرض أن محتويات مسجل التجزىء CS هى CS=12F0H ، وأن محتويات مؤشر الأوامر IP هى IP=001BH فما هو العنوان الحقيقى فى الذاكرة للأمر الذى عليه الدور فى التنفيذ ؟ يتحدد هذا العنوان بعد أن يقوم المعالج بإجراء الخطوتين التاليتين :

1- محتويات مسجل التجزىء CS تتم إزاحتها ناحية اليسار بمقدار 4 بت فتصبح المحتويات الجديدة هى :

CS=12F00H

2-تجمع محتويات مؤشر الأوامر مع محتويات المسجل CS بعد الإزاحة فيتكون لدينا العنوان الحقيقى كالتالى :

$$\begin{array}{r} CS = 12F00H \\ IP = 001BH \; + \\ \hline 12F1BH \end{array}$$

العنوان الحقيقى فى الذاكرة هو

أى أن الأمر الذى عليه الدور فى التنفيذ سيكون موجودا فى الذاكرة فى العنوان 12F1BH(20بت) كما رأينا فى المثال السابق . من ذلك نفهم حقيقة مهمة جدا وهى أن مؤشر الأوامر يشير أو يحدد عنوان في الذاكرة منسوبا أو محسوبا بمحتويات المسجل CS . ولنضرب لذلك المثال التوضيحى التالى : افترض أن لدينا سيارة هنا فى القاهرة وأقصى ما تستطيع أن تفعله هذه السيارة هو السير فى دائرة نصف قطرها 5 كيلومتر لتوزيع الحليب مثلا .. هذه هى مقدرتها ! ... فهل تستطيع هذه السيارة أن توزع الحليب فى لندن ؟ نعم تستطيع إذا نقلناها إلى لندن بالطائرة ! إن هذه السيارة تقابل مؤشر الأوامر الذى يحتوى فقط 16 بت ولا يستطيع التعامل إلا مع 64 كيلو بايت فقط ولكن هذه 64 كيلو بايت تتحدد بدايتها بمحتويات المسجل CS بعد إزاحتها ، وبذلك فإن مؤشر الأوامر يستطيع جلب أى أمر من أى مكان فى الذاكرة التى تبلغ 1 ميجابايت بعد جمع محتوياته مع محتويات المسجل CS التى تمت إزاحتها لليسار ، تماما مثل السيارة التى تستطيع أن توزع الحليب فى أى مكان فى العالم بعد نقلها بالطائرة للمكان المطلوب . لذلك فإنه فى بداية أى برنامج لابد من تحميل المسجل CS بالعنوان

الذى نرغب فى كتابة البرنامج ابتداء منه وهذا العنوان بالطبع يكون فى أى مكان خلال الذاكرة التى تبلغ 1 ميجابايت . هنا يظهر سؤال وهو: لماذا ارتبط مؤشر الأوامر IP بالمسجل CS بالذات ولم يرتبط بأى واحد آخر من مسجلات التجزىء مثل المسجل DS أو SS مثلا ؟ إن ذلك يرجع إلى الوظيفة المحددة لكل واحد من هذه المسجلات والتى نبينها فيما يلى :

14-4-1 مسجل مقطع الشفرات Code Segment register, CS

يحتوى هذا المسجل عنوان بداية جزء من الذاكرة يبلغ 64 كيلو بايت يخصص لكتابة شفرات البرامج فيه فقط ، ولذلك فإن مؤشر الأوامر يرتبط دائما بهذا المسجل لأن مؤشر الأوامر يشير على عنوان الأمر الذى عليه الدور فى التنفيذ ولابد أن الأمر يقع فى هذا الجزء من الذاكرة ، ويتحدد العنوان الحقيقى للأمر كما ذكرنا بإضافة محتويات مؤشر الأوامر مع محتويات المسجل CS بعد إزاحتها 4 بت ناحية اليسار . يمكن أن تتغير محتويات المسجل CS فى أثناء تنفيذ البرنامج مع أوامر القفز أو النداء على البرامج الفرعية وذلك فى حالات خاصة سيأتى شرحها بعد ذلك .

14-4-2 مسجل مقطع البيانات Data Segment register, DS

يحتوى هذا المسجل على عنوان بداية جزء من الذاكرة يبلغ 64 كيلو بايت أيضا وهذا الجزء يحتوى جميع البيانات التى يتعامل معها أو يحتاجها البرنامج ، تتم عنونة هذه البيانات بإضافة محتويات المسجل DS بعد إزاحتها لليسار 4 بت مع محتويات أى واحد من المسجلات DX أو BX أو SI أو DI كما سنرى عند دراستنا لطرق العنونة .

14-4-3 مسجل مقطع المكدسة Stack Segment register, SS

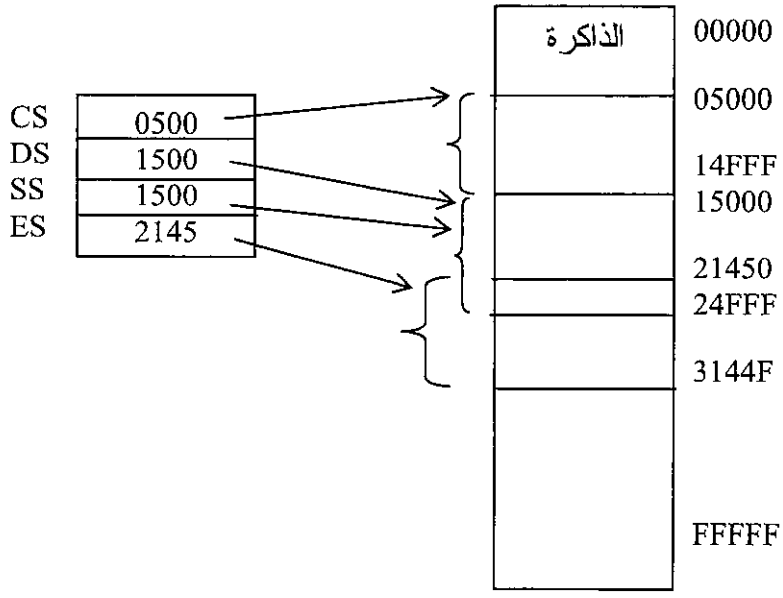
يحتوى هذا المسجل عنوان بداية جزء من الذاكرة يبلغ 64 كيلو بايت يستعملها المعالج كمكدسة . تستخدم المكدسة لتخزين البيانات والعناوين الضرورية عند القفز إلى البرامج الفرعية أو القفز إلى برنامج لخدمة مقاطعة حيث من شأن هذه البيانات والعناوين التى تخزن فى المكدسة أن تساعد المعالج على الرجوع إلى نفس المكان الذى تم القفز منه فى البرنامج الأساسى واسترجاع القيم الحقيقية لجميع المسجلات التى كانت موجودة قبل القفز بحيث يرجع المعالج إلى تنفيذ البرنامج الأساسى من حيث انتهى قبل القفز تماما دون خوف من تغير سير البرنامج بسبب فقد محتويات أحد المسجلات . يتم سحب البيانات من المكدسة على أساس أن آخر ما تم تسجيله يكون هو أول ما يتم سحبه أى أن آخر بايت تم تخزينها تكون أول بايت يتم سحبها Last In First Out, LIFO . تتم عملية السحب والإضافة فى المكدسة وبالتتابع الذى أشرنا إليه بمساعدة مسجل مؤشر

المكدسة Stack Pointer, SP حيث يحتوى هذا المسجل الذى يتكون من 16 بت على عنوان آخر مكان فى المكدسة تم التخزين فيه وبالطبع فإن مقدار المكدسة يتحدد ب 64 كيلو بايت كما ذكرنا . يتم تحديد العنوان الفعلى أو الحقيقى داخل هذا الجزء من الذاكرة (المكدسة) عن طريق إزاحة محتويات مسجل مقطع المكدسة SS ناحية اليسار بمقدار 4 بتات ثم إضافة محتويات مؤشر المكدسة إليها.

14-4-4 مسجل المقطع الإضافى Extra Segment register, ES

يحتوى هذا المسجل على عنوان بداية جزء من الذاكرة يبلغ 64 كيلو بايت تستخدم لتخزين البيانات أيضا وبالذات سلاسل الحروف strings . يتحدد العنوان الحقيقى أو الفعلى لأى معلومة داخل هذا الجزء بإزاحة محتويات مسجل المقطع ES ناحية اليسار بمقدار 4 بتات ثم يضاف إليه محتويات أى من المسجلين DI أو SI على حسب الأمر الذى يتم تنفيذه .

شكل (14-3) يبين محتويات مقترحة لكل واحد من مسجلات التجزىء والمساحة الفعلية التى يشغلها الجزء المقابل لكل مسجل على خريطة الذاكرة . نلاحظ من شكل (14-3) أن هذه الأجزاء يمكن أن تتداخل مع بعضها البعض ، بل ويمكن أن تشغل كلها نفس الجزء من خريطة الذاكرة .



شكل (14-3) محتويات مسجلات التجزىء والذاكرة الفعلية

14-4-5 مسجل الأعلام أو مسجل الحالة Status Register, SR

يحتوى هذا المسجل على 16 بت مستخدم منها 9 بتات فقط كأعلام وباقى بتات المسجل غير مستخدمه . إن كل بت أو علم من هذه الأعلام يعكس حالة معينة من حالات نتيجة آخر عملية أو منطقية قام المعالج بتنفيذها . وفيما يلى نقدم هذه الأعلام والجالة التى يعكسها أو يبينها ومتى يكون كل علم صفرا ومتى يكون واحدا على ضوء هذه النتيجة (سبق شرح معظم هذه الأعلام ولكن لا مانع من مراجعة سريعة لوظائفها) .

1- علم الحمل Carry flag, CF

إذا حصل حمل أو استلاف من أو إلى آخر بت نتيجة إجراء أى عملية حسابية أو منطقية فإن علم الحمل يصبح واحدا ، ويكون صفرا فيما عدا ذلك . يتأثر هذا العلم أيضا ببعض أوامر الإزاحة والدوران .

2- علم الباريتى Parity flag, PF

إذا احتوت نتيجة آخر عملية أو منطقية نفذها المعالج على عدد زوجى من الوحايد فإن علم الباريتى يصبح واحدا ، أما إذا احتوت النتيجة على عدد فردى من الوحايد فإن هذا العلم يصبح صفرا .

3-علم الحمل النصفى Half carry flag, HF

يكون هذا العلم واحدا إذا حصل هناك حمل أو استلاف من أو إلى البت الثالثة (منتصف البايت) عند إجراء أى عملية حسابية أو منطقية . لاحظ أننا نعد البتات فى أى بايت ابتداء من الصفر ، أى البت رقم صفر ورقم واحد ورقم اثنين وهكذا .

4-علم الصفر Zero flag, ZF

يكون هذا العلم واحدا إذا كانت نتيجة آخر عملية حسابية أو منطقية نفذها المعالج تساوى صفرا ، ويكون هذا العلم صفرا إذا كانت النتيجة تختلف عن الصفر .

5-علم الإشارة Sign flag, SF

يوضح هذا العلم إشارة نتيجة آخر عملية حسابية أو منطقية نفذها المعالج ، فإذا كانت هذه النتيجة سالبة يكون هذا العلم واحدا وإذا كانت هذه النتيجة موجبة فإن هذا العلم يكون صفرا . لاحظ أن المعالج يعتبر النتيجة سالبة إذا كانت آخر بت فيها تساوى واحدا ويعتبر النتيجة موجبة إذا كانت آخر بت تساوى صفرا . من ذلك نقول أن علم الإشارة يساوى دائما آخر بت فى النتيجة .

6-علم الفخ أو المصيدة Trap flag, TF

هذا العلم لا يعكس نتيجة عملية نفذها المعالج ، ولكن حينما يكون هذا العلم واحدا فإن المعالج ينفذ البرنامج خطوة بخطوة وحينما يكون صفرا فإنه ينفذها بالطريقة المعتادة .

7-علم تنشيط المقاطعة Interrupt enable flag, IF

لكى يمكن مقاطعة المعالج 8088/8086 فإنه يتم إعطاؤه إشارة على طرف طلب المقاطعة INTR (الطرف 18 فى شريحة المعالج) ولكن هذه المقاطعة لن يقبلها المعالج إلا إذا كان علم المقاطعة IF فعال أى يساوى واحدا هو الآخر .

8-علم الاتجاه Direction flag, DF

هناك بعض الأوامر الخاصة بالتعامل مع سلاسل الحروف character strings من خلال المسجلين SI, DI حيث يزاد واحد أو ينقص واحد من محتويات هذين المسجلين ليشير إلى مكان معين فى هذه السلسلة . يبين علم الاتجاه DF إذا كان سيكون هناك زيادة أم سيكون هناك نقص بمقدار واحد على محتويات هذين المسجلين . إذا كان DF=1 فإن ذلك يعنى أنه سيكون هناك زيادة بمقدار واحد على محتويات هذين المسجلين ، وبالطبع إذا كان DF=0 فإن ذلك يعنى أنه سيكون هناك إنقاص بمقدار واحد على هذه المحتويات .

9-علم الفيضان Overflow flag, OF

يبين هذا العلم إذا كان هناك فيضان حسابي في نتيجة أى عملية حسابية مثل الجمع والطرح أم لا . فمثلا فى حالة جمع الرقم 7FH الذى يساوى (+127) مع الرقم (1H) فإن النتيجة تكون 80H والتى تعتبر سالبة بعد أخذ المتمم الثنائى لها وتساوى (-128) ، ولأن الرقم (-128) يعتبر غير صحيح لأن أى رقم سالب يجب ألا يتعدى (-127) فإن علم الفيضان يكون واحد . وعلى ذلك فإن هذا العلم يكون صفرا طالما لم يكن هناك فيضان فى النتيجة .

127	7FH
001 +	+01H
128-	80H

14-5 طرق العنونة
Addressing modes

فى أثناء تنفيذ المعالج لأى برنامج فإنه ينقل بيانات من مسجل إلى مسجل آخر أو من مسجل إلى مكان ما فى الذاكرة أو من مكان ما فى الذاكرة إلى أى مسجل داخل المعالج نفسه . هناك طرق مختلفة يمكن استخدامها لكى يتم ذلك وهذه الطرق المختلفة يجب أن يلم بها أى مبرمج حتى يكون برنامجه ذو كفاءة عالية . سنقدم فى هذا الجزء شرحا مفصلا لهذه الطرق المختلفة من خلال استخدام الأمر MOV كمثال تطبيقى . يقوم الأمر MOV بنقل معلومة من مكان (المصدر) وهذا

المصدر يكون إما مسجل داخل المعالج نفسه أو بايت من بايتات الذاكرة إلى مكان آخر (الهدف) وهذا الهدف أيضا يكون إما مسجل أو مكان فى الذاكرة . الصورة العامة لهذا الأمر هى :

MOV destination, source

حيث تنتقل المعلومة من المصدر إلى الهدف وكمثال على ذلك الأمر التالى :

MOV AX,BX

حيث تنتقل محتويات المسجل BX إلى المسجل AX وهو المركم . نلاحظ أنه دائما يكتب مصدر المعلومة بعد الفاصلة من ناحية اليمين وأما الهدف الذى ستنتقل إليه المعلومة فيكتب بجانب الأمر MOV وقبل الفاصلة .

14-5-1 عنونة المسجل Register addressing mode

تستخدم هذه الطريقة لنقل معلومة (بايت أو كلمة ، والكلمة 2 بايت) من مسجل إلى مسجل آخر ، أى أن مصدر المعلومة يكون مسجلا وكذلك الهدف . وهذه الطريقة تعتبر أسرع الطرق لنقل معلومة من مكان إلى مكان حيث كل من مصدر وهدف المعلومة يكون مسجلا داخل المعالج نفسه ولا يتعامل المعالج مع الذاكرة على الإطلاق . كمثال على ذلك الأمران :

MOV CX,AX

الذى ينقل محتويات المسجل AX (16بت) إلى المسجل CX (16بت أيضا) .

MOV AH,AL

الذى ينقل محتويات النصف الأول AL من المسجل AX إلى النصف الثانى AH فى المسجل نفسه .

من المهم جدا هنا أن نلاحظ أحجام المسجلات التى نتعامل معها ، فلا يصح مثلا أن ننقل محتويات مسجل 8 بت إلى مسجل 16بت أو العكس حيث سيعطى الأسمبلر رسالة خطأ على ذلك لأن ذلك غير مسموح . الجدير بالذكر هنا أنه فى مثل هذه الأوامر فإن مسجل المصدر لا تتغير محتوياته ولكن يؤخذ منها نسخة أو صورة وتوضع فى المسجل الهدف . فالأمر MOV CX,AX مثلا يأخذ نسخة من محتويات المسجل AX ويضعها فى المسجل CX دون تغير فى محتويات المسجل المصدر AX والذى يتغير فقط هو المسجل الهدف CX .

14-5-2 العنونة الفورية Immediate addressing mode

تستخدم هذه الطريقة لنقل معلومة (بايت أو كلمة) موجودة فى الأمر نفسه إلى مسجل من المسجلات . أى أن مصدر المعلومة هنا ليس مسجلا داخل المعالج ولا بايت فى الذاكرة ولكن المعلومة تعتبر ثابت أو قيمة موجودة فى الأمر نفسه وبعد شفرة الأمر مباشرة ، كمثال على ذلك الأمر :

MOV AX, 34F6H

الذى يضع الثابت أو الرقم أو المعلومة 34F6H المكونة من 16بت والموجودة فى البرنامج بعد شفرة الأمر MOV فى المسجل AX . لاحظ أن H فى آخر أى رقم تعنى أن هذا الرقم مكتوبا فى النظام الستعشرى . بعض الأسمبلر تضع العلامة # أمام الثابت أو المعلومة الفورية ولكنها قليلة ونحن فى هذا الكتاب لن نتبع ذلك وسنضع أى ثابت بدون هذه العلامة ، فقط سنضع حرف H للدلالة على أن الرقم ستعشرى أو إذا كان الرقم فى النظام العشرى فلن نضع أى علامة .

14-5-3 العنونة المباشرة Direct addressing mode

هنا يتعامل المعالج مع الذاكرة حيث سيرسل لها أو يستقبل منها معلومة ، وعلى ذلك لابد من تحديد عنوان هذه المعلومة . فى العنونة المباشرة يحتوى الأمر نفسه على العنوان المباشر للمعلومة أو الثابت المراد جلبه أو إرساله من أو إلى الذاكرة . تذكر جيدا أن هذا العنوان يحدد نسبة إلى محتويات مسجل التجزىء DS ، فمثلا الأمر MOV AL,[1234H] معناه نقل نسخة من محتويات العنوان 1234H فى جزء البيانات إلى المسجل AL . لاحظ أنه بفرض أن محتويات المسجل DS=2000H فإن العنوان الفعلى للمعلومة السابقة سيكون 21234H بعد إزاحة محتويات المسجل DS لليسار 4بت كما رأينا مسبقا فى حسابات العناوين الفعلية . إذا كان العنوان الذى سيتم التعامل معه فى جزء البيانات سيتكرر كثيرا فى البرنامج فإنه يمكن فى أول البرنامج إعطاء رمزا لهذا العنوان ثم بعد ذلك يستخدم هذا الرمز للدلالة على هذا العنوان فى أى مكان فى البرنامج . فمثلا يمكن أن نرمز للعنوان 1234H فى المثال السابق بالرمز NUMBER باستخدام الأمر NUMBER EQU 1234H فى أول البرنامج ، ثم بعد ذلك نستخدم الرمز NUMBER للدلالة على هذا العنوان كما فى الأمر MOV AL,NUMBER .

14-5-4 العنونة غير المباشرة Indirect addressing

هذه الطريقة من العنونة تسمح بالتعامل مع بيانات موجودة فى الذاكرة حيث العنوان الذى سيتم التعامل معه فى هذه الحالة يكون موجودا فى أحد مسجلات المعالج التالية: BX, BP, SI, DI . كمثال على ذلك افترض أن المسجل BX يحتوى الرقم 1000H وطلبنا من المعالج تنفيذ الأمر التالى: MOV AX,[BX] . فى هذه الحالة سيقوم المعالج بإحضار نسخة من محتويات العنوان 1000H (والذى يليه) ويضعها فى المسجل AX . أى أن محتويات المسجل BX الموضوع بين قوسين مربعين كما رأينا تمثل عنوان المعلومة وليس المعلومة نفسها ، ففى عدم وجود القوسين سينسخ المعالج محتويات المسجل BX ويضعها فى المسجل AX كما رأينا فى أول طرق العنونة (عنونة المسجل) . يجب أن

نؤكد هنا أن العنوان الفعلى للمعلومة يحسب منسوبا لمحتويات مسجل التجزىء DS بعد إزاحته ناحية اليسار 4 بتات كما ذكرنا سلفا ، أى أنه إذا كانت محتويات المسجل DS=0100H فإن الأمر السابق سينسخ محتويات العنوان 01000+1000=02000H والذى يليه ويضعها فى المسجل AX . لاحظ أيضا أن المسجلات BX, SI, DI تعنون عناوين منسوبة إلى مسجل التجزىء DS بينما المسجل BP فيعنون عناوين منسوبة لمسجل التجزىء SS . كأمثلة على هذا النوع من العنونة انظر إلى الأوامر التالية :

```
MOV CX,[BX]
MOV [BP],BL
MOV [DI],AH
MOV [DI],[BX]  (خطأ)
```

حيث الأمر الأول سينقل محتويات عنوان (والذى يليه) فى جزء البيانات أو ذاكرة البيانات data segment المشار إليه بالمسجل BX إلى المسجل CX ، بينما الأمر الثانى سينقل محتويات النصف الأول من المسجل BX إلى عنوان مشار إليه بالمسجل BP ويقع فى جزء المكدسة . الأمر الثالث سينقل النصف العلوى من المسجل AX إلى عنوان مشار إليه بالمسجل DI ويقع فى جزء البيانات ، أما الأمر الرابع فغير مسموح به لأنه ينقل من ذاكرة إلى ذاكرة وهذا النوع من العنونة غير مسموح به إلا فى حالات خاصة جدا مع بعض أوامر سلاسل الحروف .

14-5-5 عنونة القاعدة زائد الفهرسة Base plus index addressing

تعتبر هذه الطريقة من العنونة بمثابة عنونة غير مباشرة ولكن طريقة تكوين أو الحصول على العنوان تختلف عن الطريقة السابقة . هنا المسجلين BX و BP يستخدمان كقاعدة base أو كبداية لمجموعة أو صف أو مرصوصة array من العناوين حيث BX تستخدم فى حالة وجود مرصوصة البيانات فى جزء البيانات و BP تستخدم فى حالة وجود مرصوصة البيانات فى جزء المكدسة . فى هذا النوع من العنونة يتكون العنوان بجمع محتويات واحدة من مسجلات القاعدة BX أو BP مع محتويات واحد من مسجلات الفهرسة SI أو DI . يوضح ذلك المثال التالى :

```
MOV DL,[BX+DI]
```

حيث سينسخ المعالج محتويات العنوان المكون من جمع محتويات المسجلين BX و DI ويضعها فى النصف الأول من المسجل DX .

14-5-6 العنونة النسبية Relative addressing mode

هذا النوع من العنونة يختلف اختلافا بسيطا عن عنونة القاعدة زائد الفهرسة الذى تم شرحه سابقا حيث هنا يتم تحديد عنوان الذاكرة المراد التعامل معه عن طريق جمع محتويات أحد المسجلات BX, BP, SI, DI مع إزاحة تعطى فى الأمر نفسه كما فى المثال التالى :

```
MOV AX,[BX+1000]
```

حيث هنا سيتم عنونة العنوان المحدد بجمع محتويات المسجل BX مع الرقم 1000H وهذا العنوان سيكون فى جزء البيانات من الذاكرة المحدد بمحتويات المسجل DS . جدول 14-1 يبين طرق العنونة السابقة كلها مع مثال لكل طريقة ، حيث يمكنك مراجعته على ضوء ما سبق وبتأنى حتى يمكنك فهم هذه الطرق .

14-6 تمارين

1. ما هى المسجلات ذات 8 بت التى يمكن التعامل معها من خلال الأوامر للمعالج 8086/8088 ؟
2. قارن بين المعالجات 4 و 8 و 16 و 32 من حيث سرعة التنفيذ إذا تساوت كل العوامل الأخرى ؟
3. ما هى وحدة التنفيذ ووحدة مواجهة المسارات فى المعالجين 8086/8088 ؟ وما أثرهما على أداء المعالج ؟ وما هو الفرق بين كل وحدة فى كل من المعالجين ؟
4. ما هو طابور الإحضار Prefetch Queue ؟ وما هو تأثيره على أداء المعالج؟ وكم عدد البايتات فيها فى كل من المعالج 8086 و 8088 ؟
5. ما هى المسجلات ذات 16بت ، والمسجلات ذات 8 بت التى يمكن التعامل معها من خلال الأوامر للمعالج 8086/8088 ؟
6. لماذا يطلق على المسجل CX مسجل العد Count register ؟ والمسجل DX مسجل البيانات Data register ؟
7. ما هو الخطأ فى أوامر الانتقال التالية :

```
MOV AL,BX
MOV CS,SS
MOV ES,F214H
MOV [BX],[DI]
```

الأمـــــــــــــــــــــــــــر	العنـــوان الفعلى للمعلومة فى الذاكـــرة
MOV AL,BL	عنونة مسجل
MOV AL, temp	(10xDS)+temp
MOV AL,55H	Immediate عنونة فورية
MOV AL,[BP]	(10xSS) + BP
MOV AL,[BX]	(10xDS) + BX
MOV AL,[DI]	(10xDS) + DI
MOV AL,[SI]	(10xDS) + SI
MOV AL,[BP+5]	(10xSS) + BP + 5
MOV Al,[BX+4H]	(10xDS) + BX + 4H
MOV AL,[DI-66H]	(10xDS) + DI - 66H
MOV AL,[SI-400H]	(10xDS) + SI - 400H
MOV AL, temp[BX]	(10xDS) + temp + BX
MOV AL, temp[BP]	(10xSS) + temp + BP
MOV AL, temp[SI]	(10xDS) + temp + SI
MOV AL, temp[DI]	(10xDS) + temp + DI
MOV AL, temp[BX+10H]	(10xDS) + temp + BX + 10H
MOV AL, temp[BP-23H]	(10xSS) + temp + BP - 23H
MOV AL, temp[SI+50H]	(10xDS) + temp + SI + 50H
MOV AL, temp[DI+80H]	(10xDS) + temp + DI + 80H
MOV AL,[BX+DI]	(10xDS) + DI + BX
MOV AL,[BP+DI]	(10xSS) + DI + BP
MOV AL,[BX+SI]	(10xDS) + SI + BX
MOV AL,[BP+SI]	(10xSS) + SI + BP
MOV AL,[BX+DI+8]	(10xDS) + DI + BX + 8
MOV AL,[BP+DI-10H]	(10xSS) + DI + BP - 10H
MOV AL,[BX+SI-7]	(10xDS) + SI + BX -7
MOV AL,[BP+SI+10H]	(10xSS) + SI + BP + 10H
MOV AL, temp[BX+SI]	(10xDS) + temp + BX + SI
MOV AL, temp[BP+SI]	(10xSS) + temp + BP + SI
MOV AL, temp[BX+DI]	(10xDS) + temp + BX +DI
MOV AL, temp[BP+DI]	(10xSS) + temp + BP + DI
MOV AL, temp[BX+SI+9]	(10xDS) + temp + BX +SI +9
MOV AL, temp[BP+SI-10H]	(10xSS) + temp + BP + SI - 10H
MOV AL,temp[BX+DI+200H]	(10xDS) + temp + BX +DI +200H
MOV AL, temp[BP+DI+1FH]	(10xSS) + temp + BP + DI +1FH

جدول 14-1

8. أكتب أوامر الانتقال التى تقوم بالآتى:
   - تحميل المسجل BX بالمعلومة 0F42H
   - تحميل العنوان 32000H بالمعلومة 0BH
   - تصفير بايتات الذاكرة (أى جعل محتوياتها تساوى صفرا) ابتداء من العنوان 32000H إلى 32050H بالتتابع
   - نقل محتويات الذاكرة 32000H حتى 32050H إلى 42000H حتى 42050H
   - تحميل المسجلات CS, SS, DS, ES بالعنوان 3200H
9. ما معنى وضع القوسين [ ] حول أى معامل من معاملات أى أمر ؟
10. ما هو عنوان الذاكرة الذى سيتم التعامل معه فى كل من الأوامر التالية إذا كانت DS=3200H, BX=0200H, DI=0300H, BP=1000H, SS=2000H و list=0250H :

- MOV AL,[3200H]
- MOV AL,[BX]
- MOV [DI],AL
- MOV AL,[BX+100]
- MOV AL,[BP+100H]
- MOV AL,[BP+DI]
- MOV AL, list[DI]
- MOV AL, list[100H]
- MOV AL,[BX+DI]