الفصل الأول ١

مقدمة عامة عن المتحكمات والمعالجات

General Introduction About Microcontrollers and Microprocessors

١-١ مقدمة

قبل أن نتعمق فى شرح تفصيلى للمتحكمات microcontrollers وتطبيقاتها المختلفة لابد أن نلقى نظرة سريعة على المكونات الأساسية لأى متحكم وأى معالج microprocessor والفرق بين المتحكم والمعالج والفرق بينهما وبين الحاسب أو الميكروكمبيوتر ومتى نختار أى منهما .

١-٢ ما هو الحاسب؟ وكيف نشأت فكرته؟

إن الفكرة التى يقوم عليها الحاسب أساسا ما هى إلا تقليدا لطريقة الإنسان فى حل أى مسألة . أنت مثلا حينما تريد أن تحل مسألة فى الطبيعة أو الإلكترونيات ، ماذا تحتاج ؟ إنك لكى تحل هذه المسألة ستحتاج للآتى :

١. القوانين المهمة لحل هذه المسألة وبالطبع فإنك ستستعين بأحد الكتب التى تحتوى على هذه القوانين . أليست هذه تكافىء الذاكرة ROM التى نسجل عليها البرنامج وكل ما نحتاج إليه من ثوابت .

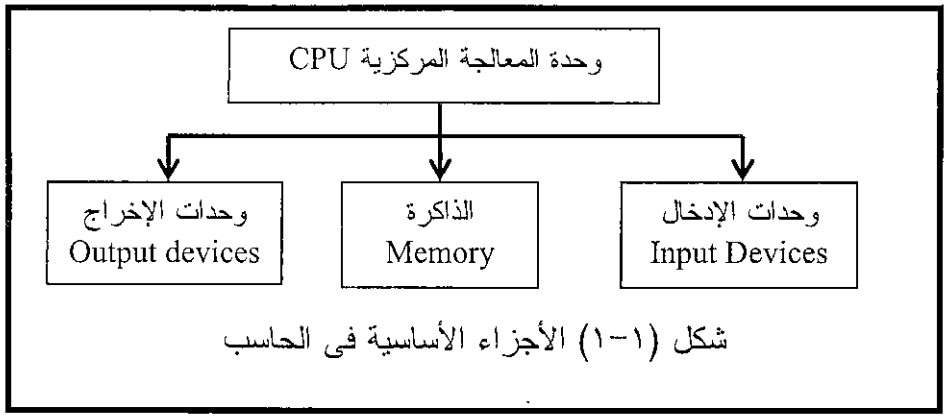
٢. ستحتاج كراسة أو بعض الأوراق لتدوين بعض النتائج الحسابية المرحلية فى أثناء عملية الحل . أليست هذه تكافىء الذاكرة RAM التى نسجل عليها كل البيانات التى نحتاج إليها أو التى تنتج من البرنامج .

٣. ستحتاج أيضا إلى آلة حاسبة لمساعدتك فى إجراء بعض العمليات الحسابية . أليست هذه هى وحدة الحساب والمنطق ALU التى تقوم بأى عملية حسابية أو منطقية موجودة فى البرنامج .

٤. أخيرا ربما تحتاج إلى طابعة لكتابة تقرير أو وضع الحل فى صورة نهائية لائقة للعرض . إليست هذه وحدة إخراج للبيانات لإظهارها بصورة مناسبة للمستخدم .

هذه الأشياء الأربعة لو تم توفيرها مجتمعة لن تحل المسألة من تلقاء نفسها ولكن لابد من وجود منظم لعملية الحل وهو الشخص نفسه ، وهو ما يقابل وحدة المعالجة المركزية Central Processing Unit, CPU ، ولابد من وجود خطة للحل أيضا ، وهى ما يقابل البرنامج الذى تنفذه وحدة المعالجة المركزية . هذه الأجزاء الأربعة السابقة هى تقريبا ما يتركب منه الحاسب كما سنرى سوى أن خطة الحل وهى البرنامج يتم وضعها للحاسب عن طريق الإنسان بحيث إذا جاء الحل خطأ فلابد أن هناك خطأ فى البرنامج الموضوع للحاسب بواسطة الشخص المبرمج . أى أن الحاسب لا يحل المسألة من تلقاء نفسه ولكنه يسير على خطة الحل التى وضعتها أنت له مستفيدا فقط بالسرعة الهائلة التى ينفذ بها العمليات أو أوامر البرنامج . لذلك سنعرض الآن للأجزاء الرئيسية فى الحاسب والتى تتكون كما هو موضح فى شكل (١-١) من ثلاثة أجزاء رئيسية هى :

١.وحدات الإدخال والإخراج Input/output units .
٢.الذاكرة Memory .
٣.وحدة المعالجة المركزية Central Processing Unit .



شكل (١-١) الأجزاء الأساسية فى الحاسب

١-٣ وحدات الإدخال والإخراج Input/output units

وحدات الإدخال هى الوسائل التى يتم بها تكييف المعلومات لتكون فى صورة مناسبة يستطيع المتحكم أو المعالج التعامل معها ، ومثال ذلك لوحة المفاتيح التى تحول أى زرار تقوم بضغطه إلى إشارات كهربية وشفرات يقبلها المعالج أو المتحكم . يجب أن نفرق هنا بين بوابة الإدخال ووحدة الإدخال حيث بوابة الإدخال يتم من خلالها إدخال المعلومات التى تم تجهيزها بواسطة وحدة الإدخال إلى المتحكم . وأما وحدات الإخراج فهى الوسائل التى يتم بها إظهار المعلومات الخارجة من المعالج ، ومثال ذلك الشاشة التى ما هى إلا وسيلة ضوئية لإظهار المعلومات التى تخرج من المتحكم أو المعالج ، بالطبع فإن هذه الشاشة تكون متصلة بأحد بوابات الإخراج ، ولذلك يجب أن نفرق هنا بين بوابات الإخراج ووحدات الإخراج حيث بوابة الإخراج يتم من خلالها إخراج المعلومة من المتحكم أو المعالج إلى وحدة الإخراج التى تتعامل مع هذه المعلومات بوسائل مختلفة . سنكتفى بذلك الحديث السريع عن هذه الوحدات حيث أنها ليست الموضوع الأساسى للكتاب ولن نتعرض لها أكثر من ذلك .

١-٤ الذاكرة Memory

الذاكرة ما هى إلا وعاء لحفظ المعلومات لحين الحاجة إليها وهذه المعلومات إما أن تكون بيانات ستكون هناك حاجة إليها فيما بعد أو تكون برنامجا مخزنا فى الذاكرة فى انتظار التنفيذ . إن أى برنامج تكتبه على الحاسب وبأى لغة ولتكن لغة الباسيك BASIC مثلا لابد وأن يوضع أولا فى الذاكرة الأساسية للحاسب ثم يتم استدعاؤه

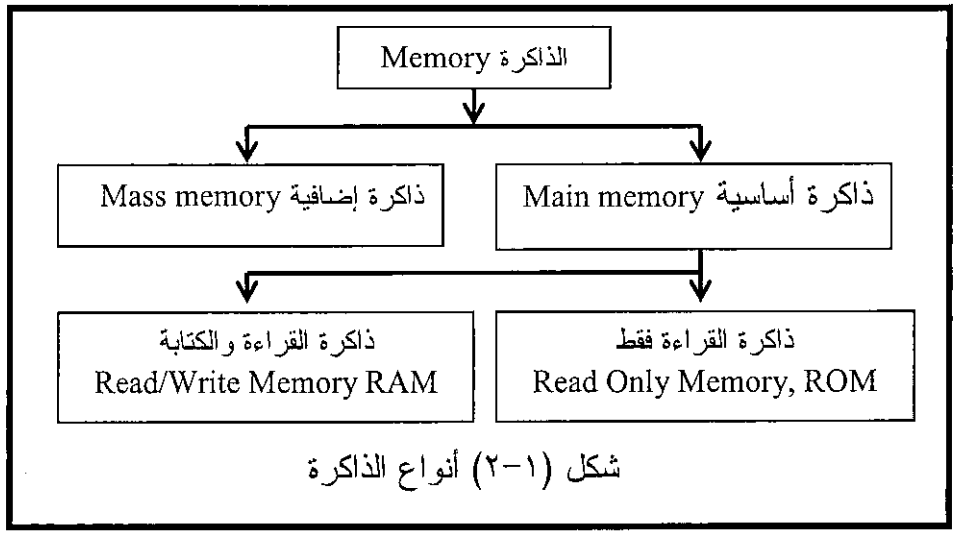
من هناك للتنفيذ عند الأمر بذلك ، أى أن الحاسب لا ينفذ إلا برامج موجودة فى الذاكرة الأساسية فقط . تنقسم الذاكرة عامة إلى قسمين :

القسم الأول وهو الذاكرة الأساسية للحاسب Main memory وهى التى تخزن فيها البرامج التي تنتظر التنفيذ ، وهذا النوع من الذاكرة يكون عادة من أشباه الموصلات Semiconductors وعلى شرائح تكاملية ويحدد مقدارها على حسب نوع المعالج المستخدم فى الحاسب كما سنرى فيما بعد . الذاكرة الأساسية للحاسب تنقسم بدورها إلى جزأين :

الأول : وهو ما يسمى بذاكرة القراءة فقط Read Only Memory , ROM وهذه الذاكرة أيضا تكون عبارة عن شرائح من أشباه الموصلات التى تم تسجيل محتوياتها بواسطة الصانع نفسه وعادة تحتوى الثوابت والبرامج المهمة لتشغيل نظام الحاسب والتى لا تضيع بانقطاع مصدر الطاقة عنها .

الثانى : وهو ذاكرة القراءة والكتابة Read/Write Memory ولقد تم العرف على تسمية هذا النوع من الذاكرة بذاكرة الاتصال العشوائى Random Access Memory, RAM وهى الذاكرة التى تحتوى البيانات والبرامج التى فى انتظار التنفيذ كما ذكرنا من قبل وهذه الذاكرة تفقد محتوياتها بانقطاع مصدر الطاقة .

القسم الثانى من أقسام الذاكرة هو الذاكرة الإضافية أو Mass (secondary) Memory وهي الذاكرة التي تستخدم لتخزين البيانات أو البرامج لفترات طويلة وعادة فإن هذه الذاكرة تكون مغناطيسية مثل الأقراص الممغنطة Floppy disks والشرائط الممغنطة Tapes وهناك أيضا الأقراص الصلبة Hard Disks . هذا القسم من الذاكرة لا دخل للمعالج فى تحديد كميته ولكن كميته تحدد على حسب رغبة المستخدم وما وصلت إليه التكنولوجيا فى هذا المجال . شكل (١-٢) يبين رسما توضيحيا لأقسام الذاكرة فى الحاسب التى سبق الحديث عنها .



شكل (١-٢) أنواع الذاكرة

عند التعامل مع المعالجات لابد من إضافة أو توصيل شرائح ذاكرة خارجية على المعالج ، كذلك هناك بعض المتحكمات التى من الممكن ، وعند الحاجة لذلك ، أن نوصل عليها شرائح ذاكرة خارجية ولا نكتفى بالذاكرة الموجودة بداخلها مثل المتحكم 8051 . لذلك فإنه فيما يلى سنلقى نظرة سريعة على خواص وأنواع شرائح الذاكرة الموجودة فى السوق والتى سنستخدمها فى الكثير من المشاريع أو التطبيقات التى يمكن أن نتعرض لها .

توجد الذاكرة فى صورة شرائح تحدد سعة كل منها عادة بالبايت . فالشريحة التى سعتها ٢ كيلوبايت مثلا تحتوى ٢٠٤٨ مكانا تخزينيا كل مكان عبارة عن بايت (٨ بت) . أحيانا تكتب سعة هذه الشريحة كما يلى: 2Kx8 بت . على حسب هذا العدد من الأماكن التخزينية (بايتات) لابد أن يخرج من شريحة الذاكرة عدد من الأطراف يطلق عليها مسار العناوين address bus لهذه الشريحة ، بحيث أنه ٢ أس عدد أطراف مسار العناوين يساوى سعة هذه الشريحة أو عدد الأماكن التخزينية فى هذه الشريحة . يتم تخزين البيانات داخل الشريحة ، أو قراءة البيانات منها من خلال عدد من الأطراف تسمى مسار البيانات data bus . فى العادة يتم تخزين بايت أو قراءة بايت من معظم شرائح الذاكرة ، لذلك فإن مسار البيانات لهذه الشرائح يكون ٨ أطراف أو ٨ أرجل . عند التعامل مع أى شريحة ذاكرة لابد من تنشيطها أولا أو اختيارها ويتم ذلك عن طريق طرف تنشيط خاص بذلك يسمى طرف اختيار الشريحة أو طرف تنشيطها Chip Select, CS أو Chip Enable, CE . بعض الشرائح يتم تنشيطها بوضع هذا الطرف بواحد ، ومعظم الشرائح يتم تنشيطها بوضع هذا الطرف بصفر وفى هذه الحالة فإن هذا الطرف يكتب وفوقه خط صغير كما يلى: $\overline{CS}$. عند التعامل مع أى شريحة ذاكرة بغرض التخزين أو الكتابة فيها فإنه يتم تنشيط طرف خاص بذلك يسمى طرف الكتابة Write, WR بحيث أنه عند تنشيط هذا الطرف فإن البيانات الموجودة على مسار البيانات يتم تخزينها فى المكان الذى عنوانه موجود على مسار العناوين . هناك أيضا فى كل شريحة طرف خاص بالقراءة من الشريحة يسمى طرف القراءة Read, RD بحيث أنه عند تنشيط هذا الطرف فإن البيانات الموجودة فى البايت التى يوجد عنوانها على مسار العناوين تخرج على مسار البيانات . فى الغالب تكون هذه الخطوط (RD و WR) منخفضة الفعالية ، أى أنها تنشط عندما تكون صفر . من الخواص المهمة لأى شريحة ذاكرة ما يسمى بزمن الوصول Access time وهذا هو الزمن الذى يمضى من لحظة وضع العنوان المراد التعامل معه على مسار العنوان للشريحة ثم تنشيطها حتى لحظة خروج هذه البيانات على مسار البيانات للشريحة . هذا الزمن كلما كان صغيرا كلما كان ذلك أفضل وكلما كان سعر الشريحة أغلى . هذا الزمن يتراوح بين ١٠ نانوثانية أو أقل قليلا حتى ٢٠٠ نانوثانية ويتوقف ذلك على تكنولوجيا أو طريقة تصنيع هذه الشرائح .

ذاكرة القراءة فقط Read Only Memory, ROM

هذا النوع من الذاكرة لا تفقد محتوياتها عند انقطاع القدرة nonvolatile ، ولذلك فإنها غالبا ما يتم برمجتها عن طريق المصنع وتسمى Mask ROM ، ويتم برمجتها مرة واحدة فقط ولا يمكن مسحها أو إعادة التسجيل عليها . غالبا ما يرمز لها بالرمز One Time Programmable, OTP ، أى البرمجة لمرة واحدة . يمتاز هذا النوع من الذاكرة برخص ثمنه جدا بالمقارنة بالأنواع التالية ولذلك فإنها تستخدم فى حالة المنتج الكبير فى أخر مرحلة من تصميم المشروع وبعد التأكد من أن البرنامج خالى من أى خطأ . هناك أنواع عديدة من هذه الذاكرة نذكرها باختصار فيما يلى :

الذاكرة Programmable ROM, PROM

هذا النوع يبرمج أيضا لمرة واحدة ، OTP ، ولكنه يمكن برمجته عن طريق المستخدم لأن بتاتها تكون فى الغالب عبارة عن فيوزات يتم حرقها بتمرير تيار كهربى خلالها . إذا حصل أى خطأ فى التسجيل عليها فإن الشريحة تهمل ويبرمج غيرها .

الذاكرة Erasable Programmable ROM, EPROM

هذه الذاكرة يمكن برمجتها بتمرير تيار كهربى خلالها ، كما يمكن مسح محتوياتها تمهيدا لإعادة التسجيل عليها عن طريق تعريضها لأشعة فوق بنفسجية لمدة حوالى عشرون دقيقة ، وهذا يعتبر عيب فى هذا النوع حيث أن هذا الوقت يعتبر طويلا نسبيا . لذلك فإن هذا النوع من الشرائح يكون لها شباك زجاجى يمكن تعريضها للأشعة من خلاله ، وبالطبع فإن ذلك يستلزم نزع الشريحة من مكانها فى المشروع ووضعها فى جهاز المسح . أيضا فإنه لكى يتم برمجتها فإنها توضع فى مبرمج خاص يقوم بحرق البرنامج على الشريحة . أى أنه لكى يتم برمجة الشريحة أو مسحها لابد من نزعها من مكانها فى المشروع أو فى الدائرة .

الذاكرة Electrically Erasable Programmable ROM, EEPROM

هذه الذاكرة يمكن مسحها وبرمجتها عن طريق تيار كهربى ولذلك فإنها لا تحتاج للزمن ٢٠ دقيقة للتعريض لضوء معين لمسحها . وعلى ذلك فإنه بتجهيزات معينة على الدائرة التى تستخدم مثل هذا النوع فإنه يمكن برمجتها ومسحها وهى فى مكانها ولا يحتاج الأمر لنزعها ووضعها فى جهاز معين . كما أنه يمكن فى هذا النوع مسح البت الخطأ فقط وإعادة تسجيلها دون الحاجة إلى مسح كل محتويات الشريحة . بالطبع فإن سعر هذا النوع من الشرائح يكون أغلى بكثير من النوع السابق .

الذاكرة اللحظية Flash Memory EPROM

تشبه تماما الذاكرة EEPROM من حيث أنه يتم مسحها وتسجيلها كهربيا وتمتاز عنها فى أن سرعة المسح أكبر بكثير من النوع السابق . هذه الذاكرة هى المستخدمة لتخزين ال BIOS فى أجهزة الحاسب وكثر استخدامها الآن حتى أنه يتوقع أن تحل محل الاسطوانة الصلبة فى جهاز الحاسب قريبا .

الذاكرة القراءة والكتابة Random Access Memory, RAM

هذه الذاكرة هى ذاكرة قراءة وكتابة بمعنى أنه يمكن التسجيل فيها ثم مسح هذه المحتويات دون أى مشاكل ولكن عيبها أن محتوياتها تفقد بمجرد انقطاع القدرة عن الشريحة volatile . هناك منها أكثر من نوع كالتالى :

الذاكرة الاستاتيكية Static RAM, SRAM

وحدة التخزين هنا هى القلاب flip flop أو الماسك latch الذى يتكون فى الأصل من أكثر من ترانزستور ولذلك فإن سعرها يكون مرتفعا بالنسبة إلى ال DRAM التى سنراها بعد قليل .

الذاكرة الديناميكية Dynamic RAM, DRAM

وحدة التخزين هنا هى مكثف بدلا من قلاب لذلك فإنها تكون أرخص بكثير من الذاكرة السابقة ولكن من عيوبها أن المكثف يفقد شحنته بعد زمن معين (حوالى 3 ميللى ثانية) لذلك لابد من إعادة شحنه قبل هذا الوقت . من مميزاتها أيضا أنها لا تستهلك قدرة power بشكل كبير مثل الذاكرة السابقة لذلك فإنها استخدمت بكثرة كذاكرة قراءة وكتابة فى الحاسبات .

ذاكرة القراءة والكتابة التى لا تفقد محتوياتها Nonvolatile RAM, NVRAM

هذه الذاكرة هى نوع خاص جدا من الذاكرة SRAM المصنعة بتكنولوجيا خاصة جدا تستهلك القدرة بكفاءة عالية جدا بجانب أنها تحتوى بداخلها بطارية تستخدم لتغذية الشريحة فى حالة انقطاع القدرة . لذلك فإنها تحتوى أيضا دائرة تحكم تنقل تغذية الشريحة إلى المصدر الخارجى فى حالة وجوده وفى حالة انقطاع القدرة تنتقل التغذية إلى البطارية الداخلية وكل ذلك بغرض الاستخدام الأمثل للبطارية الداخلية . هذه الشريحة مرتفعة الثمن جدا ويمكنها الاحتفاظ بمحتوياتها حتى 10 سنوات .

١-٥ وحدة المعالجة المركزية Central Processing Unit, CPU

الوظيفة الأساسية لوحدة المعالجة المركزية هى تنفيذ البرامج عن طريق إحضار الأوامر من الذاكرة الواحد بعد الآخر ثم تنفيذها بنفس التتابع ، فمثلا يتم إحضار الأمر الأول ثم ينفذ وبعد ذلك يحضر الأمر الثانى وينفذ فالأمر الثالث وينفذ وهكذا إلى أن تصل إلى نهاية البرنامج . بعض هذه الأوامر تحتاج لبيانات من أماكن أخرى فى الذاكرة يتم إحضارها ، وبعضها يحتاج لبيانات من بوابات إدخال يتم إحضارها أيضا ، والبعض الآخر من الأوامر يتطلب كتابة أو تسجيل بعض البيانات إما فى الذاكرة أو فى وحدات إخراج ، كل ذلك وأكثر تقوم به وحدة المعالجة المركزية . هذه الوحدة هى المعالج أو الميكروبروسيسور بعينه ، وهى مكون أساسى من مكونات أى متحكم ، لذلك لابد من شرح مكوناتها بالتفصيل هنا . هذه الوحدة تتكون من المكونات التالية :

١. مجموعة مسجلات وعدادات .

٢. وحدة الحساب والمنطق ALU .
٣. وحدة التزامن Clock .

وحدة الحساب والمنطق لا يمكن الحديث عنها بالتفصيل فى مثل هذا المكان الضيق لأنها تحتاج لإفراد فصل خاص بها وهذا خارج نطاق هذا الكتاب . لذلك سنكتفى بمعرفة الوظيفة الأساسية لوحدة الحساب والمنطق وهى إجراء العمليات الحسابية والمنطقية التى تطلبها منها وحدة المعالجة المركزية حيث تقوم وحدة المعالجة بإرسال أوامر الحساب والمنطق إلى وحدة الحساب والمنطق التى تقوم بدورها بتنفيذ هذه الأوامر وإرجاع النتيجة لوحدة المعالجة المركزية . وأما وحدة التزامن فهى الوحدة المسئولة عن إجراء أى فعل يقوم به المعالج بالتزامن أو التوافق مع نبضة من نبضات الساعة الخاصة به ، تماما مثلما يتوافق المتدربون فى برنامج تمارين الصباح مع المدرب وهو يقول ، واحد ، اثنين ، واحد ، اثنين ، وهكذا حتى يتوافق الفريق معه فى كل حركة يقوم بها . أما مجموعة المسجلات والعدادات ووظيفة كل منها فسوف تكون الموضوع الأساسى فى الجزء التالى الذى سنعرض فيه بشكل عام لوظيفة كل مسجل من المسجلات الرئيسية فى وحدة المعالجة المركزية التى هى مكون أساسى من مكونات أى متحكم كما ذكرنا سابقا .

١-٥-١ مسجل التراكم Accumulator, A

يعتبر مسجل التراكم Accumulator ، وعادة يرمز له بالرمز A ، من أكثر مسجلات وحدة المعالجة المركزية عملا ولذلك فإنه يمكننا النظر إليه على أنه سكرتيرا لهذه الوحدة . إن أى عملية حسابية أو منطقية تحتاجها وحدة المعالجة المركزية لابد وأن يكون مسجل التراكم طرفا فيها ، فمثلا لو أردت أن تجمع أى رقمين فإن واحدا منهما لابد أن يوضع فى مسجل التراكم وأما الرقم الآخر فيوضع فى أى مسجل آخر أو حتى فى الذاكرة ، ليس هذا فقط بل إن نتيجة أى عملية حسابية أو منطقية لا توضع إلا في مسجل التراكم ومنه يمكن نقلها لأى مكان آخر وذلك في المعالجات ٨ بت وفي الكثير من المتحكمات . هناك مهمة أخرى أيضا لهذا المسجل وهى أن أى عملية إدخال أو إخراج للبيانات من خلال بوابات الإدخال أو الإخراج عادة تكون من خلال هذا المسجل . أى أن المعلومة توضع فى مسجل التراكم أولا ثم يتم إخراجها إلى بوابة الإخراج ، أو إذا كانت المعلومة قادمة من بوابة إدخال فإنها توضع أولا فى مسجل التراكم ثم يتم نقلها منه لأى مكان آخر . إن عدد البتات (الخانات) bits الموجودة فى مسجل التراكم عادة يساوى عدد خطوط مسار البيانات data bus .

١-٥-٢ عداد البرنامج Program Counter, PC

كما علمنا فإن مهمة وحدة المعالجة المركزية cpu الأساسية هى إحضار الأوامر من الذاكرة الواحد بعد الأخر ثم تنفيذها ، ولذلك فإنه لابد لهذه المهمة من تحديد للأماكن التى تحتوى هذه الأوامر فى الذاكرة . يحتوى عداد البرنامج دائما على عنوان البايت

أو المكان في الذاكرة الذى يحتوى الأمر الذى عليه الدور فى التنفيذ ، وكلما تم إحضار أى أمر من الذاكرة وقبل أن يتم تنفيذه فإن عداد البرنامج تتغير محتوياته بحيث تشير دائما إلى عنوان الأمر القادم فى التنفيذ . تذكر أيضا أنه حتى لو حدث قفز من مكان فى البرنامج إلى مكان آخر فإن وحدة التحكم تضع عنوان الأمر الذى سيتم القفز إليه فى عداد البرنامج حتى يصبح هو الأمر الذى عليه الدور فى التنفيذ وتنتقل عملية تنفيذ البرنامج إلى هناك . عدد بتات هذا العداد عادة تساوى عدد بتات مسار العناوين address bus وهذا منطقى جدا حتى يمكن إحضار الأوامر مهما كانت فى أى مكان فى الذاكرة سواء كانت فى أولها أو فى آخرها ، لاحظ أن كمية الذاكرة التى يمكن أن يتعامل معها المعالج تتوقف على عدد البتات أو الخطوط فى مسار العناوين كما ذكرنا من قبل . إذا نظرنا إلى عداد البرنامج على أنه مسجل يحتوى عنوان الأمر الذى عليه الدور فى التنفيذ فإننا سنصنفه على أنه من المسجلات ذات الأغراض الخاصة dedicated register الغير متاحة للمبرمج لاستخدامها فى عمليات البرمجة .

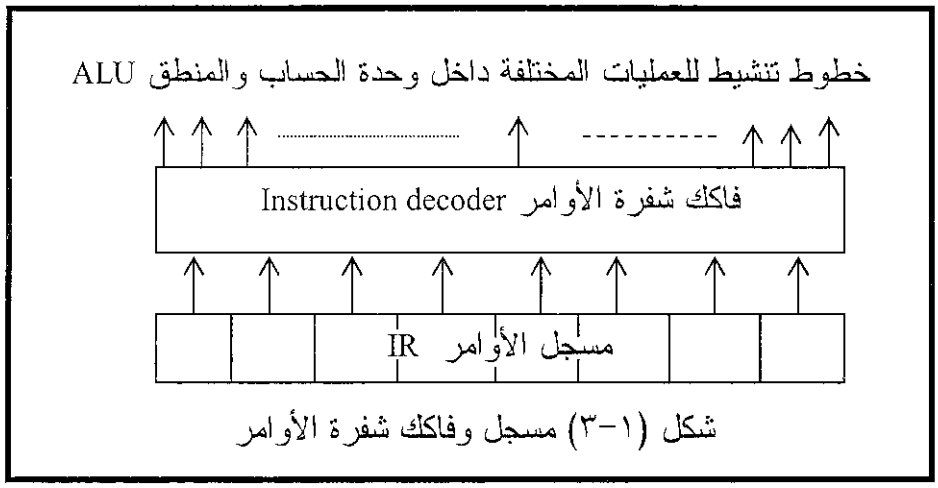
١-٥-٣ مسجل وفاكك شفرة الأوامر

Instruction Register And Decoder

بعد أن يتم إحضار الأمر من الذاكرة إلى وحدة المعالجة المركزية لابد أن يسجل أو يوضع فى أحد الأماكن فى انتظار تنفيذه ، هذا المكان هو مسجل الأوامر . أى أن مسجل الأوامر يحتوى شفرة الأمر الذى يتم تنفيذه الآن . لاحظ أن عدد بتات مسجل الأوامر عادة يساوى عدد بتات البايت فى الذاكرة التى تساوى بدورها عدد بتات مسار البيانات . أول خطوات تنفيذ أى أمر تبدأ من فاكك شفرة الأوامر الذى يفك شفرة الأمر أو يتعرف عليه والذى يتصل دخله بخرج مسجل الأوامر كما فى شكل (١-٣) بحيث أنه على حسب شفرة الأمر الموجودة فى مسجل الأوامر فإن عملية واحدة فقط سيتم تنفيذها على حسب الشفرة الموجودة على دخل فاكك الشفرة ويتم ذلك بالطبع بمساعدة وحدة التحكم ووحدة الحساب والمنطق .

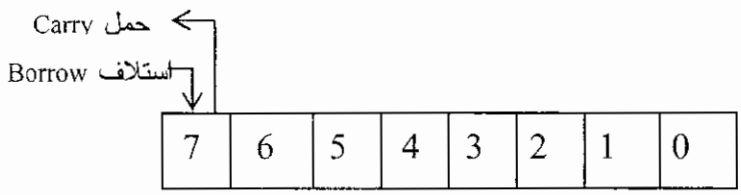
١-٥-٤ مسجل الحالة Status Register, SR

أحيانا يطلق على هذا المسجل اسم مسجل الأعلام Flag Register, FR . يعتبر هذا المسجل نشرة إخبارية تعكس حالة نتيجة آخر عملية حسابية أو منطقية قامت وحدة المعالجة المركزية بتنفيذها ، فمن هذا المسجل نستطيع أن نعرف مثلا إذا كانت هذه النتيجة سالبة أم موجبة أم تساوى صفرا وغير ذلك من الأخبار المفيدة . هذا المسجل يحتوى على عدد من البتات وكل واحدة منها تعتبر علما flag يعكس أو يدل على حالة معينة من العملية الحسابية أو المنطقية التى تم تنفيذها ، من هذه الأعلام أو هذه البتات ما يلى :



شكل (١-٣) مسجل وفاكك شفرة الأوامر

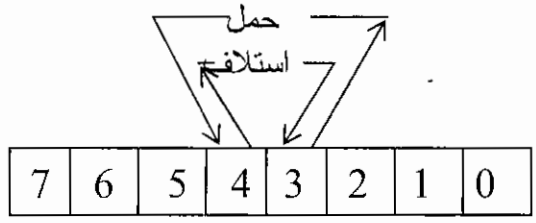
- **علم الصفر** Zero flag, ZF هذه البت تكون واحدا إذا كانت نتيجة آخر عملية حسابية أو منطقية تم تنفيذها تساوى صفرا ، وتكون هذه البت صفرا إذا كانت النتيجة مختلفة عن الصفر سواء موجبة أو سالبة .
- **علم الإشارة Sign flag, SF** هذه البت تكون واحدا إذا كانت نتيجة آخر عملية حسابية أو منطقية تم تنفيذها سالبة ، أما إذا كانت هذه النتيجة موجبة فإن هذا العلم يكون صفرا ، لذلك فإنه أحيانا يسمى بعلم السالبية Negative Flag, NF . لاحظ أن آخر بت فى النتيجة توضح إشارتها فإذا كانت آخر بت تساوى صفرا فإن ذلك يعنى أن النتيجة موجبة أما إذا كانت هذه البت واحدا فإن ذلك يعنى أن النتيجة سالبة لذلك فإنه دائما تكون محتويات علم الإشارة تساوى محتويات آخر بت فى النتيجة .
- **علم الحمل Carry flag , CF** هذا العلم يكون واحدا إذا حصل حمل carry من آخر بت فى أى عملية جمع أو حصل استلاف Borrow لآخر بت فى أى عملية طرح ، ويكون صفرا إذا لم يكن هناك حمل أو استلاف فى آخر عملية حسابية . شكل (١-٤) يبين عملية الحمل والاستلاف من وإلى البت الأخيرة ، رقم ٧ .
- **علم الباريتى Parity flag, PF** هذا العلم يكون واحدا إذا كانت نتيجة آخر عملية حسابية أو منطقية تحتوى على عدد زوجى من الوحايد ، أما إذا كانت هذه النتيجة تحتوى على عدد فردى من الوحايد فإن هذا العلم يكون صفرا .
- **علم الحمل النصفى أو البينى Half carry flag , HC** هذا العلم يكون واحدا إذا كان هناك حمل من الخانة أو البت الثالثة إلى البت الرابعة نتيجة أى عملية جمع أو هناك استلاف من البت الرابعة إلى البت الثالثة نتيجة أى عملية طرح ، ويكون صفرا فيما عدا ذلك أى إذا لم يحدث استلاف أو حمل من أو إلى البت الرابعة ، لاحظ أننا هنا نبدأ عملية عد البتات بالرقم صفر ، أى أن أول بت هى البت رقم صفر . شكل (١-٥) يبين كيفية تأثر علم الحمل النصفى .



شكل (١-٤) الحمل والاستلاف من وإلى البت رقم ٧

هذه الأعلام ستستخدم فى أوامر القفز المشروط والنداء على البرامج الفرعية المشروطة كما سنرى فيما بعد ، لذلك يطلق على هذا المسجل أحيانا مسجل الشروط condition code register ، كما أن كل من مسجل التراكم ومسجل الحالة يطلق عليهما كلمة حالة المعالج Processor Status Word, PSW .

التطبيق على جميع هذه الأعلام واستخدامها سيأتى عند الشرح التفصيلى لأوامر المتحكم ، مع العلم أن عدد الأعلام سيختلف من متحكم لآخر كما سنرى عند دراستنا للتركيب التفصيلى لكل متحكم سندرسه فى هذا الكتاب ولكن دعنا الآن ننظر للمثال التالى كتطبيق سريع على هذه الأعلام .



شكل (١-٥) الحمل والاستلاف النصفى (من وإلى الخانة الثالثة)

مثال ١-١:

اكتب محتويات الأعلام السابقة بعد إجراء عملية جمع الرقمين 77H و A5H . لاحظ أن الرقمين مكتوبين فى الصورة الستعشرية hexadecimal ، وهذا هو المقصود من وضع الحرف H بعد كل رقم .

الجمع الثنائى للرقمين السابقين سيتم كما يلى :

الرقم الأول	0111 0111
الرقم الثانى	1010 0101
النتيجة	0001 1100 حمل 1

نلاحظ الآتى من النتيجة السابقة :

١. النتيجة لا تساوى الصفر ، إذن فعلم الصفر يساوى صفر ZF=0 .
٢. آخر بت فى النتيجة صفر فالنتيجة موجبة وعلم الإشارة يساوى صفر SF=0 .
٣. هناك حمل من البت السابعة (الأخيرة) فعلم الحمل يساوى واحد CF=1 .
٤. النتيجة تحتوى ثلاثة وحايد (عدد فردى) فعلم الباريتى يساوى صفر PF=0 .
٥. ليس هناك حمل من الخانة الثالثة للرابعة فعلم الحمل النصفى يساوى صفر HCF=0 .

١-٥-٥ مسجل مؤشر المكدسة Stack Pointer register, SP

سيأتى إن شاء الله شرحا تفصيليا للمكدسة stack فيما بعد فى معرض الكلام عن البرامج الفرعية والمقاطعة ، ولكن الآن بإمكانك أن تعرف أن المكدسة هى جزء من الذاكرة يتم فيه تخزين بعض العناوين أو البيانات المهمة والتى لابد من الحاجة إليها واسترجاعها مرة ثانية وبنفس الترتيب الذى تم تخزينها به . مسجل مؤشر المكدسة يحتوى عنوان آخر مكان تم التسجيل فيه فى هذا الجزء من الذاكرة . لاحظ أن المبرمج عادة تكون لديه الحرية فى اختيار الجزء من الذاكرة الذى سيستعمل كمكدسة عن طريق تحميل المؤشر بقيمة قاع أو بداية المكدسة .

١-٥-٦ المسجلات عامة الأغراض General Purpose Registers

فى الكثير من الأحوال عندما نجمع أكثر من رقم ، نحتاج لحفظ نتيجة معينة لحين استخدامها فى عمليات أخرى لاحقة ، ولذلك فإنه بدلا من إرسال هذه النتيجة إلى الذاكرة ثم استدعائها ثانية مما يأخذ الكثير من الوقت ، لذلك فقد تم تجهيز المتحكم ببعض المسجلات العامة التى تستخدم لتخزين مثل هذه النتائج المرحلية لحين الحاجة إليها . عدد البتات فى هذه المسجلات يكون عادة مساويا لعدد بتات مسار البيانات . عدد هذه المسجلات يختلف من متحكم لآخر ومن شركة لأخرى كما سنرى فى فصول هذا الكتاب .

الآن وبعد كل ما سبق هل نستطيع أن نحدد ما هو المعالج؟ وما هو المتحكم؟ وما هو الفرق بينهما؟

١-٦ ما هو المعالج؟ وما هو المتحكم؟ وما هو الفرق بينهما؟

المعالج microprocessor هو وحدة معالجة مركزية تكون مهمته هى إحضار الأوامر من الذاكرة وتنفيذها ولا يحتوى أى ذاكرة لتخزين البرامج أو البيانات سوى عدد قليل من المسجلات العامة التى تساعد فى تنفيذ الأوامر . ولذلك فإن المعالج كشريحة منفردة يعتبر عديم الفائدة ، ولذلك فإنه لكى تتم الاستفادة من المعالج لابد من توصيل ذاكرة معه حتى يمكن أن توضع بها البرامج التى سينفذها المعالج . أيضا لابد من

توصيل عدد من بوابات الإدخال والإخراج لاستخدامها فى التعامل مع الأجهزة الخارجية . لذلك فإن أى مقرر فى المعالجات يكون جزء كبير منه خاص بكيفية توصيل الذاكرة وبوابات الإدخال والإخراج على هذا المعالج . من ذلك نرى أن الحاسب هو وحدة حساب ومنطق (معالج) مع كمية من الذاكرة وعدد من بوابات الإدخال والإخراج .

من خواص المعالجات أن عدد المسجلات العامة فيها يكون قليلا جدا لا يتعدى العشرة مسجلات بالمقارنة بالعديد من المسجلات العامة فى المتحكمات والتى تصل إلى العشرات . كما أن عدد أوامر المعالجات يكون كبيرا جدا ، فى العادة يكون بالمئات . من أمثلة المعالجات المعالج Z80 و Intel8085 و MC6800 وكلها أمثلة على المعالجات ذات ٨ بت (أى أن مسار البيانات لها ٨ خطوط) . مثل هذه المعالجات تستخدم بكثرة فى الكثير من التطبيقات والمشاريع ودوائر التحكم ومازالت تنتج حتى الآن . سارت المعالجات فى سلسلة من التطورات التى لا يمكن تقديمها هنا حتى وصلت إلى المعالج Pentium4 الشهير والمستخدم فى كل حاسباتنا الشخصية هذه الأيام . هذه المعالجات ذات الإمكانيات المتقدمة تستخدم فقط فى بناء الحاسبات العامة ولا تستخدم فى بناء دوائر تحكم أو مشاريع بسيطة نظرا لارتفاع ثمنها وتعقيدها .

المتحكم microcontroller هو جهاز حاسب (ميكروكومبيوتر) كامل على شريحة واحدة . كما سنرى فى فصول هذا الكتاب فإن كل المتحكمات بلا استثناء تحتوى داخليا وعلى نفس الشريحة كل مكونات الميكروكومبيتر التالية :

- وحدة معالجة مركزية CPU
- كمية من ذاكرة الكتابة والقراءة RAM حيث يختلف مقدارها من متحكم لآخر .
- كمية من ذاكرة القراءة فقط (EPROM أو EEPROM أو Flash) يختلف مقدارها ونوعها من متحكم لآخر .
- عدد من بوابات الإدخال والإخراج يختلف من متحكم لآخر .
- إمكانية للإرسال والاستقبال على التوالى ، قد توجد أو لا توجد من متحكم لآخر .
- بعض المتحكمات قد تحتوى محول انسيابى/رقمى A/D أو محول رقمى/انسيابى D/A .

كل هذه المكونات تكون موجودة على شريحة واحدة هى شريحة المتحكم ، أو قل شريحة الحاسب . بعض المتحكمات يكون بها إمكانية توصيل ذاكرة خارجية ، والمعظم ليس به هذه الإمكانية . تتصف المتحكمات دائما بأن مجموعة الأوامر لها تكون قليلة مقارنة بالمعالجات فهى لا تتعدى المائة أمر وفى الغالب تكون فى حدود العشرات . كل المتحكمات التى سندرسها هنا يكون مسار البيانات بها ٨ بت ، مع العلم أن هناك متحكمات ١٦ و ٣٢ بت ولكنها قليلة الاستخدام وتحتاج لكتاب خاص بها .

الفرق بين المعالج والمتحكم :

- كما لمسنا فيما سبق فإن الهدف الأساسى من المتحكم هو التحكم فى الوسط الموجود به عن طريق قراءة بيانات وإخراج بيانات يتحكم بها فى الوسط المحيط به . لذلك فإن أطراف المتحكم يكون أغلبها ، إن لَم يكن كلها ، تكون خطوط إدخال وإخراج للبيانات ولا يبالى بخطوط لمسار العناوين أو خطوط لمسار بيانات خارجى أو خطوط تحكم مثل RD أو WR لأن المتحكم فى الغالب لا يتعامل مع ذاكرة خارجية ، فما الفائدة من هذه الأطراف . فى المعالج تكون كل أطراف المعالج إما أطراف لمسار عناوين أو أطراف لمسار بيانات خارجى ، أو أطراف لمسار تحكم لأن المعالج لابد أن توصل عليه ذاكرة خارجية وبوابات إدخال وإخراج ، لذلك لابد من وجود هذه الأطراف ليتمكن مستخدم المعالج من توصيل هذه الأشياء خارجيا .
- ظاهرة شائعة فى أطراف المتحكمات أن كل طرف تقريبا يؤدى أكثر من وظيفة واحدة وهذا ما نسميه function multiplexing ، فى حين أن هذه الظاهرة تكون موجودة فى حدود ضيقة جدا فى المعالجات . توقع مثلا أن تجد أحد خطوط المتحكم يعمل كطرف إدخال/إخراج ويمكنه أن يستخدم كطرف مقاطعة مثلا .
- عدد أوامر المعالجات تكون أكثر بكثير جدا من عدد أوامر المتحكمات . إن ذلك يعنى أن برمجة أى مشكلة باستخدام المعالجات تكون فى الغالب أسهل من المتحكمات نظرا لكثرة الأوامر مما يعنى أنك لا تكون مضطرا لعمل مناورة بالأوامر لتصل إلى هدف معين ، فى المعالجات فى الغالب ستجد أمر يؤدى لك ما تريد مباشرة .
- طبعا كما رأينا فإنه لتصميم أى نظام باستخدام المعالجات فإنه لابد من توصيل ذاكرة وبوابات إدخال/إخراج بالكم والكيفية التى يتطلبها النظام أو التطبيق . أما فى حالة استخدام المتحكمات فإن الأمر سيكون أبسط بكثير فى حالة وجود متحكم يفى بالأغراض التى يطلبها منك التطبيق المقترح .

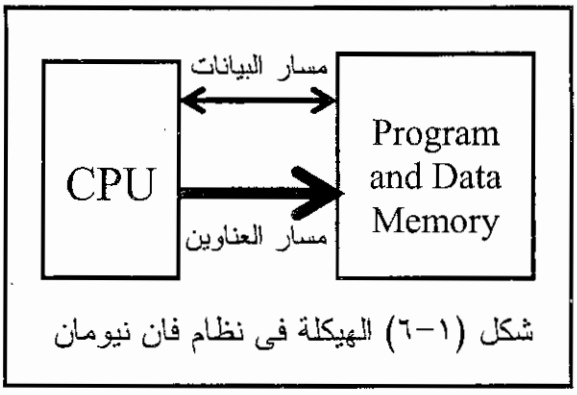
بعد أن رأينا هذه الفروق ، دائما يسأل البعض ، **وهل يفضل استخدام المعالج أم المتحكم فى التطبيق المطلوب منى ؟**

الإجابة على هذا السؤال بالطبع لن تكون قاطعة ولن تحصل على إجابة عليها إلا إذا أجبت على كل هذه الأسئلة لأنها تتوقف على العديد من العوامل أولها هو التطبيق نفسه ، كم يحتاج من ذاكرة؟ وهل الذاكرة الداخلية فى المتحكم إذا استخدمته ستكفى أم لا؟ كم عدد بوابات الإدخال/الإخراج التى تحتاج إليها؟ وهل يفى المتحكم بهذا الغرض؟ تذكر أن أى تطبيق يمكن تنفيذه باستخدام المعالجات ، كما أنه يمكن تنفيذه أيضا باستخدام المتحكمات فأيها تفضل أنت؟

١-٧ هيكلة الحاسب Computer Architecture

كما رأينا فإن الحاسب والمتحكم يتكون أساسا من وحدة المعالجة المركزية cpu والذاكرة وبعض وحدات الإدخال/الإخراج . نقصد بالهيكلة هنا هو كيفية اتصال الذاكرة مع وحدة المعالجة المركزية ، وديناميكية إحضار الأوامر من الذاكرة وتنفيذها . هناك طريقتان لتوصيل الذاكرة مع وحدة المعالجة المركزية وهما كالتالى:

١- طريقة فان نيومان Van neumann



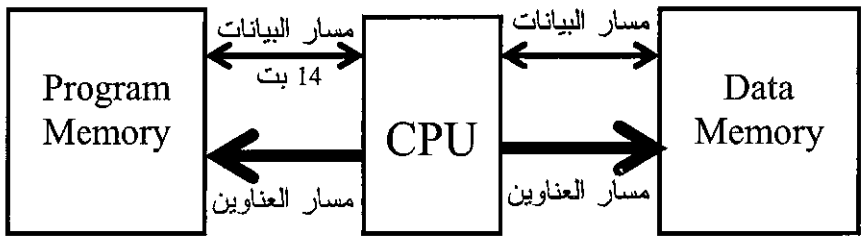
شكل (١-٦) الهيكلة فى نظام فان نيومان

شكل (١-٦) يبين طريقة توصيل الذاكرة مع وحدة المعالجة المركزية فى ظل هذه الطريقة . نلاحظ هنا أن هناك مسار للعناوين يتحدد عليه عنوان المكان أو البايت فى الذاكرة التى سيتم التعامل معها . بعد تحديد عنوان البايت التى سيتم التعامل معها فإن محتويات هذه البايت تنتقل على مسار البيانات . من خواص هذه الطريقة أيضا أن نفس الذاكرة تحمل الأوامر (البرنامج) والبيانات . فى هذا النظام يتم إحضار الأمر أولا ثم تنفيذه بعد ذلك بحيث أنه فى أثناء الإحضار تتوقف عملية التنفيذ ، وفى أثناء التنفيذ لا يتم إحضار أوامر . لذلك تتصف هذه الطريقة ببطئها عن طريقة هارفارد التى سنراها بعد قليل . عرض بايت الذاكرة هنا يكون ٨ بت . من الممكن أن تزيد شفرة الأمر هنا عن بايت واحدة وفى هذه الحالة فإن شفرة الأمر تكتب فى أكثر من بايت ويتم إحضار شفرة الأمر فى أكثر من دورة إحضار . كل المعالجات ذات ٨ بت تعمل بهذه الطريقة ويعمل بها أيضا بعض المتحكمات مثل المتحكم 8051 ، كما يعمل بها أيضا بعض المعالجات المتقدمة . تذكر أن فان نيومان هو نسبة لمكتشف هذه الطريقة .

٢- طريقة هارفارد Harvard

شكل (١-٧) يبين طريقة توصيل الذاكرة مع وحدة المعالجة المركزية فى ظل هذه الطريقة . نلاحظ هنا أن هناك ذاكرة خاصة بالبيانات وأخرى خاصة بالأوامر (البرنامج) . عرض البايت فى ذاكرة البيانات هو ٨ بت . فى هذا النظام تكون شفرة جميع الأوامر لها نفس الطول ، ولذلك تبنى ذاكرة البرامج من بايتات يكون عرضها هو عرض شفرة الأمر . فإذا كانت شفرة الأمر تتكون من ١٤ بت مثلا كما فى بعض إصدارات المتحكمات PIC فإن ذاكرة البرمجة يكون عرضها هو ١٤ بت . لذلك فإن جميع الأوامر يتم إحضارها فى دورة أمر واحدة فقط . فى هذا النظام بما أن كل ذاكرة تكون مستقلة تماما عن الذاكرة الأخرى فإن عملية إحضار أمر وتنفيذ آخر تسير على التوازى مما يجعل هذه الطريقة أسرع بكثير من الطريقة السابقة

حيث يمكن معها استخدام طريقة الانسيابية pipelining فى إحضار وتنفيذ الأوامـــر والتى سنشرحها فى معرض شرح متحكمات PIC .



شكل (١-٧) الهيكلة فى نظام هارفارد

مسار العناوين Address bus

أى مكان تتعامل معه وحدة المعالجة المركزية cpu سواء كان ذاكرة أو غيرها ، داخل المتحكم أو خارجه ، لابد وأن تحدد عنوانا لهذا المكان . هذا العنوان يتم وضعه فى صورة شفرات كهربية من الوحايد والأصفار بواسطة وحدة المعالجة المركزية على هذه الأطراف التى تسمى مسار العناوين . لذلك فإنه على حسب عدد هذه الأطراف المخصصة لحمل شفرة العناوين يتحدد عدد الأماكن التى يمكن للوحدة أن تتعامل معها ، ودائما يكون عدد هذه الأماكن يساوى ٢ مرفوعا لأس عدد هذه الخطوط أو الأطراف . فى جميع شرائح المعالجات ٨ بت يكون عدد أطراف مسار العناوين يساوى ١٦ طرفا لذلك فإن مقدار الذاكرة التى يتعامل معها مثل هذا المعالج يساوى 2^{16} بايت = 64 كيلوبايت باعتبار أن كل واحد كيلو بايت يساوى ١٠٢٤ بايت. لاحظ أن الإشارة الموجودة على مسار العناوين تكون دائما خارجة من المعالج إلى الأجهزة الخارجية وليس العكس لأن المعالج هو فقط الذى يحدد العنوان الذى يريد التعامل معه .

مسار البيانات Data bus

بمجرد أن يحدد المعالج المكان الذى يريد التعامل، معه عن طريق العنوان الذى وضعه على مسار العناوين يقوم المعالج بإخراج أو استقبال المعلومة نفسها على مسار آخر وهو مسار البيانات . هذا المسار أيضا عبارة عن عدد من الخطوط تصل بين المعالج cpu والأجهزة المحيطة حيث تسير عليها البيانات المطلوب تداولها بين المعالج والأجهزة خارجه . إن عدد البتات التى تعرف بها أى شريحة معالج يكون على حسب عدد بتات أو أطراف مسار البيانات ، فالشرائح ٨ بت سميت كذلك لأن لها مسار بيانات مقداره ٨ بت والشرائح ١٦ بت سميت كذلك لأن لها مسار بيانات ١٦ بت وهكذا . فى عالم الحاسبات توضع أى معلومة دائما فى صورة عدد من

الخانات أو البتات وكل بت أو خانة من هذه الخانات يوضع بها واحد أو صفر حيث يمثل الواحد بجهد معين ويمثل الصفر بجهد آخر . التركيبة المكونة من هذه الوحايد والأصفار تسمى بالشفرة الثنائية للمعلومة . إن ثمانية من هذه البتات أو الخانات تسمى بايت واثنين بايت تسمى كلمة Word . فى حالة الشرائح ٨ بت فإن أى معلومة تنقل من أو إلى المعالج لابد وأن تكون مكونة من ثمانية بتات ، إذا كانت هذه المعلومة مكونة من عدد من البتات أكبر من ثمانية فإنها تنقل على أكثر من مرة وعلى حسب عدد بتاتها . فى حالة الشرائح ١٦ بت تكون وحدة التعامل فى نقل المعلومات هى ١٦ بت ، لذلك فإنه من البديهى أن نتوقع أن الشرائح ١٦ بتا تكون أسرع من الشرائح ٨ بت لهذا السبب أساسا وأسباب أخرى ، فما بالك الآن بالشرائح ٣٢ بت والشرائح ٦٤ بت . لاحظ أن زيادة عدد بتات مسار البيانات لن ينعكس فقط على سرعة نقل البيانات إلى الذاكرة ولكنه ينعكس أيضا على سرعة تنفيذ العمليات الحسابية . كلمة أخيرة عن مسار البيانات وهى أن الإشارة عليه يمكن أن تكون خارجة من المعالج إلى الأجهزة المحيطة أو داخلة إلى المعالج من الأجهزة المحيطة ولذلك فإنه يطلق عليه أنه ثنائى الاتجاه .

خطوط التحكم Control lines

هذه الخطوط يختلف عددها من معالج لآخر وعن طريق هذه الخطوط يخبر المعالج أى جهاز من الأجهزة المحيطة (الذاكرة مثلا) الذى تم تحديد عنوانه على مسار العناوين عن الغرض من هذا التعامل ، فقد يكون الغرض من التعامل مع الذاكرة مثلا هو القراءة منها ، أى استقبال معلومة منها ، فى هذه الحالة فإن المعالج يرسل إشارة إلى الذاكرة على خط التحكم $\overline{MEMR}$ Memory Read, تعرف منها الذاكرة أن الغرض من التعامل هو القراءة فتقوم بإرسال المعلومة المطلوبة على مسار البيانات فيتلقاها المعالج . أما إذا كان الغرض من التعامل هو الكتابة أو إرسال معلومة إلى الذاكرة فإن المعالج يقوم بوضع إشارة على الخط $\overline{MEMW}$ Memory Write, تفهم منها الذاكرة الغرض من التعامل فتتلقى المعلومة من على مسار البيانات . هناك خطان للتحكم بنفس الطريقة للتعامل مع بوابات الإخراج والإدخال . هناك أيضا خطوط المقاطعة Interrupt التى بها تتم مقاطعة أى برنامج يجرى تنفيذه وخطوط المسك HOLD التى بها يتم فصل المعالج عن المسارات لأغراض معينة .

٨-١ البرامج الفرعية Subroutines

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 أو التقسيم هى البرامج الفرعية subroutines . إن استخدام البرامج الفرعية من مميزاتها تسهيل عملية البرمجة

واختصار كمية الذاكرة المستخدمة لكتابة شفرات البرنامج كما سنرى . هذا بجانب أن البرامج الفرعية والمكدسة stack من الموضوعات التى لابد أن نكون على دراية بها عند التعامل مع أى متحكم أو معالج ، وسنتناولها هنا بنظرة عامة وأما تفصيليا فسوف يعتمد ذلك على نوع المتحكم الذى سنتكلم عنه .

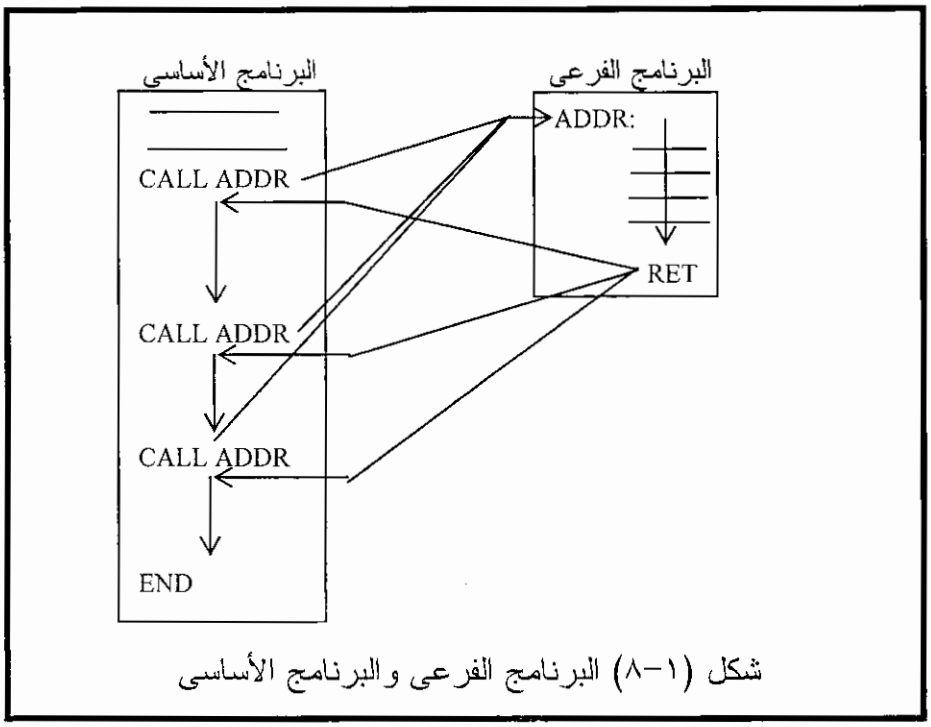
١-٨-١ ما هو البرنامج الفرعى ؟

شكل (١-٨) 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 cpu قبل القفز إلى البرنامج الفرعى .

من شكل (١-٨) 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 فى البرنامج الأساسى ، وكمثال على ذلك برامج أزمنة التأخير التى يمكن فى هذه الحالة كتابتها مرة واحدة كبرنامج فرعى يتم النداء عليه بالأمر CALL فينفذ وبعد الانتهاء من تنفيذه ترجع عملية التنفيذ إلى حيث خرجت من البرنامج الأساسى .

شكل (١-٩) 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 الفرعى (ب) ، بعد الانتهاء من تنفيذ البرنامج الفرعى (أ) تتم العودة إلى البرنامج الأساسى من حيث تم النداء على البرنامج الفرعى (أ) .

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 المتحكم فى حلقة لانهائية لا مخرج منها ولن يرجع أبدا إلى البرنامج الأساسى الذى خرج منه حيث سيظل البرنامج (ج) ينادى على (ب) والبرنامج (ب) ينادى على (ج) إلى ما لا نهاية .



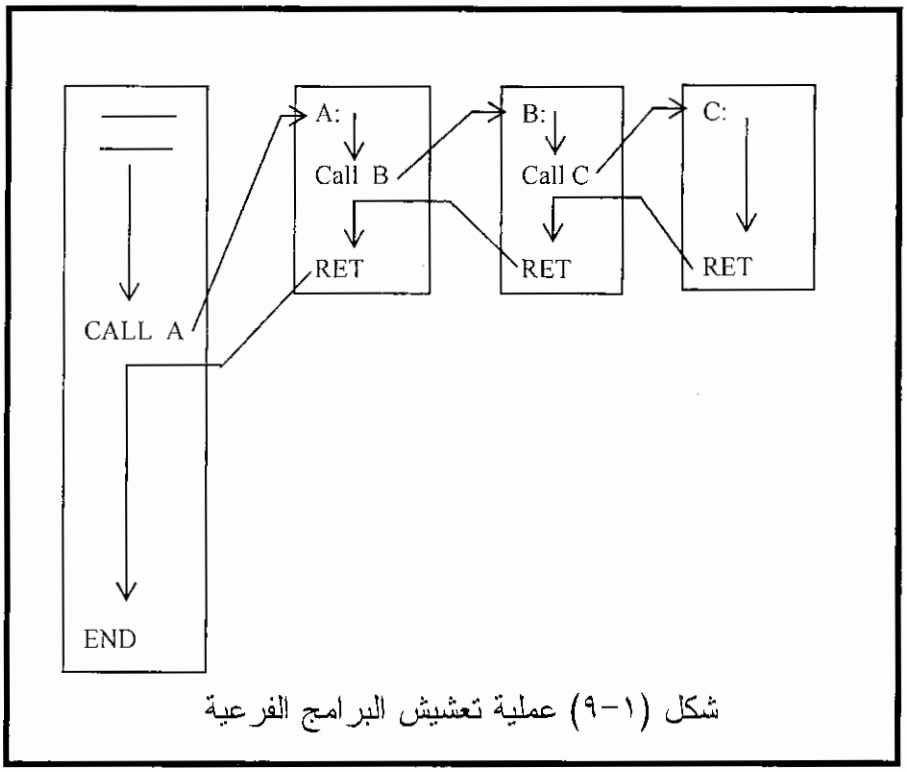
شكل (١-٨) البرنامج الفرعى والبرنامج الأساسى

هناك الكثير من أوامر النداء على البرامج الفرعية والعودة منها وسنترك الحديث عن هذه الأوامر بالتفصيل لأنها تختلف من متحكم لآخر . كل برنامج فرعى لابد أن ينتهى بأمر عودة RET تختلف صورته من متحكم لآخر كما سنرى ولكن هذا الأمر مهم جدا لأنه يضمن عودة التنفيذ إلى البرنامج الأساسى كما سنرى عند الحديث عن المكدسة .

١-٨-٢ 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 أو حتى المسجلات العامة الموجودة داخل المتحكم لخدمة أغراض النداء والعودة من البرامج الفرعية وأيضا لخدمة أغراض المقاطعة وأغراض أخرى . 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 (أو على حسب نوع المتحكم) 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 وهناك اختلاف أيضا من متحكم لآخر ولكن بصورة عامة فإنه عند النداء على البرنامج الفرعى فإن محتويات عداد البرنامج PC التى هى عنوان الأمر التالى فى التنفيذ فى البرنامج الأساسى يتم دفعها فى المكدسة إلى حيث يشير مؤشر المكدسة وبالتالى تنقص محتويات ال SP . بعد ذلك يتم تحميل عداد البرنامج بعنوان بداية البرنامج الفرعى وبذلك تنتقل عملية التنفيذ إلى هناك . فى نهاية البرنامج الفرعى يوجد الأمر RET الذى بتنفيذه يتم سحب قمة المكدسة التى يشير إليها المؤشر وتوضع فى عداد البرنامج PC مرة أخرى ، وبذلك تتم العودة إلى البرنامج الأساسى وفى نفس المكان الذى تم الخروج منه .



شكل (١-٩) عملية تعشيش البرامج الفرعية

فى حالة البرامج الفرعية المعششة 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 عنوان فى قمة المكدسة فيرجع التنفيذ إلى البرنامج الفرعى قبل

الأخير وهكذا مع كل أمر RET يسحب عنوان من قمة المكدسة ويرجع التنفيذ إلى برنامج فرعى سابق إلى أن يصل التنفيذ إلى حيث انتهى من البرنامج الأساسى .

١-٨-٣ المقاطعة

تلعب المكدسة دورا مهما أيضا فى المقاطعة . فى أثناء انشغال المتحكم أو المعالج فى تنفيذ أى برنامج يمكن لأى جهاز خارجى ، أو حتى مؤقت وصل إلى نهايته ، أو تم الانتهاء من استقبال أو إرسال معلومة ، يمكن فى كل هذه الأحوال مقاطعة المتحكم وإجباره على ترك البرنامج الأساسى والذهاب إلى برنامج خاص بخدمة أى مقاطعة من المقاطعات السابقة وتنفيذه ثم العودة إلى البرنامج الأساسى وفى نفس المكان الذى تمت عنده المقاطعة . بنفس طريقة البرامج الفرعية عند حدوث المقاطعة وقبولها فإن المتحكم يدفع بعداد البرنامج فى المكدسة ويحمل عداد البرنامج بعنوان برنامج خدمة المقاطعة حيث يتم القفز إلى هناك ويبدأ المتحكم فى تنفيذ برنامج خدمة المقاطعة . ينتهى برنامج خدمة المقاطعة بأمر عودة RTI بحيث عند تنفيذه يقوم المتحكم بتحميل عداد البرنامج بقمة المكدسة وذلك بمساعدة المؤشر SP ويعود إلى البرنامج الأساسى .

بذلك نكون قد أخذنا فكرة سريعة عن البرامج الفرعية والمقاطعة ودور المكدسة مع كل منهما وكما قلنا أن هناك تفاصيل فى كل من الموضوعين تختلف من متحكم لآخر .