الجامعة الافتراضية السورية
SYRIAN VIRTUAL UNIVERSITY

المعالجات والمتحكمات الصغرية

الدكتور عبد الناصر العاسمي

ISSN: 2617-989X

Books & Refrences

المعالجات والمتحكمات الصغرية

الدكتور عبد الناصر العاسمي

من منشورات الجامعة الافتراضية السورية

الجمهورية العربية السورية 2020

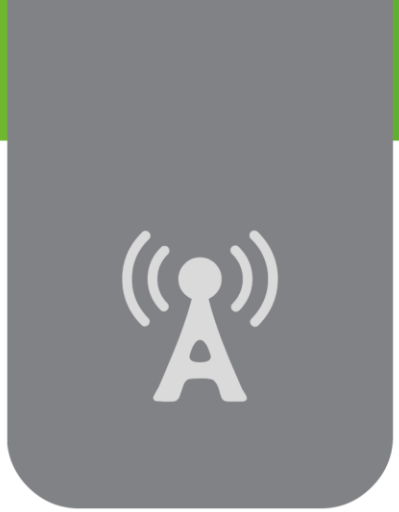
Microprocessors and Microcontrollers

Dr. Abdelnasser Assimi

Publications of the Syrian Virtual University (SVU)

Syrian Arab Republic, 2020

الفهرس



الفصل الأول: مدخل إلى المعالجات الصغرية

الكلمات المفتاحية:

معالجات صغرية، نظم المعالجة الحاسوبية، مسرى.

ملخص:

نُبيّن في هذا الفصل مقدمة عامة عن نظم المعالجة الحاسوبية ومكوناتها الرئيسية. نهتم بشكل خاص بوحدة الحساب المركزية.ودورها الأساسي في هذه النظم لذلك نستعرض البنية الداخلية العامة للمعالجات وطريقة ربطه مع باقي أجزاء النظام. كما نستعرض لمحة تاريخية عامة عن ظهور المعالجات وتطورها ومعايير اختيار المعالج وفقاً للتطبيق المرغوب.

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

- تذكرة عن بنية النظم الحاسوبية ومكوناتها
- تعريف المعالج ومكوناته الرئيسية
- لمحة تاريخية عن المعالجات
- المعايير المؤثرة في اختيار المعالج

نُبيّن في هذا الفصل مقدمة عامة عن نظم المعالجة الحاسوبية ومكوناتها الرئيسية.
نهتم بشكل خاص بوحدة الحساب المركزية.ودورها الأساسي في هذه النظم لذلك نستعرض البنية الداخلية العامة للمعالجات وطريقة ربطه مع باقي أجزاء النظام.
كما نستعرض لمحة تاريخية عامة عن ظهور المعالجات وتطورها ومعايير اختيار المعالج وفقاً للتطبيق المرغوب.

1. ما هو المعالج الصغري:

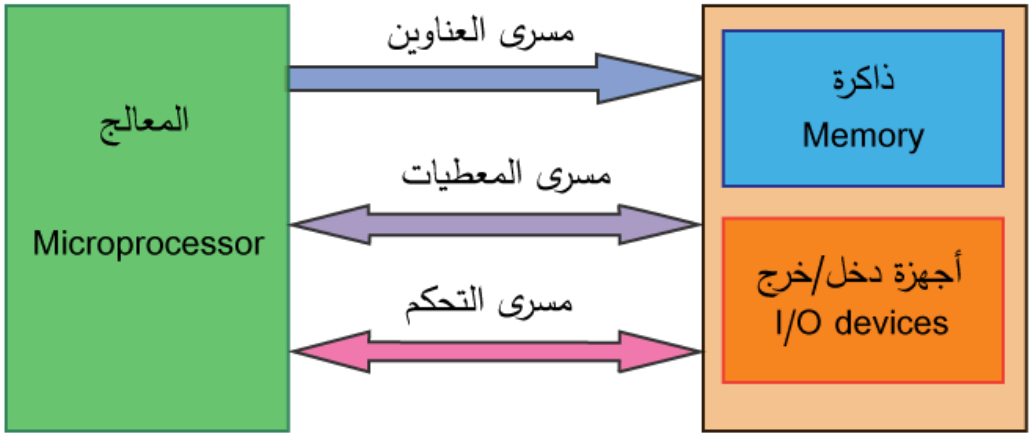
المعالج الصغري هو دارة الكترونية متكاملة قابلة للبرمجة تستقبل المعطيات من وحدة إدخال وتقوم بمعالجة هذه المعطيات وفق مجموعة من التعليمات الحسابية والمنطقية المخزنة في ذاكرة وتخرج النتائج من خلال وحدة إخراج. وبعبارة أخرى يقوم المعالج بتنفيذ البرنامج المخزن في الذاكرة ويتبادل المعطيات مع الوسط الخارجي من خلال بوابات دخل وخرج مختصة.

2. نظم المعالجة الحاسوبية:

قبل دراسة المعالجات سوف نبدأ بما هو مألوف أكثر وهو نظم المعالجة الحاسوبية مثل الحواسيب الشخصية واللوحية وغيرها من الأجهزة التي يشكل المعالج الجزء الأساسي في بنيتها.

يتألف أي نظام معالجة حاسوبي بشكل أساسي من ثلاثة أجزاء:

- معالج
- ذاكرة
- تجهيزات دخل/خرج



نموذج مبسط لنظام معالجة حاسوبي

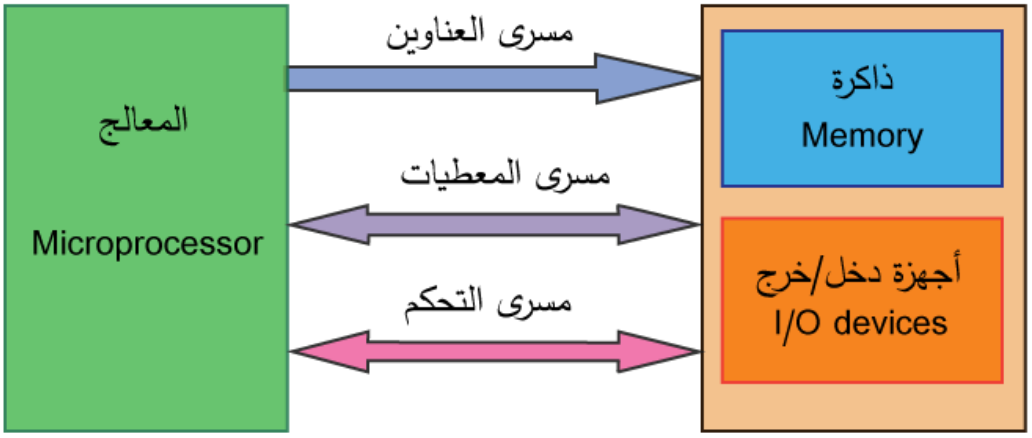
1.2. المعالج:

يشكل المعالج قلب النظام ويقوم بكافة العمليات بما فيها التحكم بباقي الأجزاء. حيث يقوم بتنفيذ سلسلة تعليمات البرنامج المخزن بالذاكرة الرئيسية ويستخدم أجهزة الدخل والخرج للتعامل مع الطرفيات الخارجية.

تتصل الأجزاء بعضها ببعض من خلال ثلاث مساري وهي:

1. مسرى المعطيات.
2. مسرى العناوين.
3. مسرى التحكم.

كما يبين الشكل التالي.



نموذج مبسط لنظام معالجة حاسوبي

2.2. الذاكرة:

تستخدم لتخزين البرامج والمعطيات التي يتعامل معها البرنامج.

- ذاكرة حية RAM: قابلة للقراءة والكتابة وتزول محتوياتها بانقطاع التغذية (volatile)
- ذاكرة ميتة ROM: قابلة للقراءة فقط ولا تزول محتوياتها بانقطاع التغذية (nonvolatile)
- ذواكر أخرى: أقراص صلبة، أقراص ليزرية أو مغناطيسية

الذاكرة الحية: RAM (Random Access Memory/ Read-Write memory)	الذاكرة الميتة: ROM (Read Only Memory)
وتستخدم كذاكرة مؤقتة لتخزين المعطيات والبرنامج الذي يتم تنفيذه. وهي ذاكرة قابلة للقراءة والكتابة في أي موقع. يضيع محتوى هذه الذاكرة عند فصل التغذية الكهربائية عنها. لذلك تسمى بالذاكرة الطيارة (volatile).	وهي ذاكرة قابلة للقراءة فقط من قبل المعالج ولا يمكن الكتابة فيها. تحتفظ الذاكرة الميتة بمحتواها بشكل دائم حتى بعد فصل التغذية عنها. لذلك تسمى هذه الذاكرة بالذاكرة الغير طياره (nonvolatile).

تصنف الذواكر إلى ذواكر أساسية وذواكر ثانوية.

تعتبر الذاكرة الحية (RAM) والذاكرة الميتة (ROM) مثالين عن الذواكر الأساسية والتي تستخدم لتنفيذ وتخزين البرامج والمعطيات. أما القرص الصلب والأقراص المرنة فإنها تشكل مثالين عن الذواكر الثانوية والتي تستخدم أيضاً لتخزين البرامج والمعطيات والنتائج التي نحصل عيها بعد تنفيذ البرامج.

ولكن الفارق أن المعالجات لا تقوم بتنفيذ البرامج من الذاكرة الثانوية مباشرة بل يقوم بنسخها أولاً إلى الذاكرة الأساسية ومن ثم يقوم بتنفيذها.

3.2. تجهيزات الإدخال والإخراج (تجهيزات العبور):

تشكل تجهيزات الإدخال والإخراج (input output devices)، والتي سنسميها بتجهيزات العبور، الوسيلة التي يتفاعل فيها المعالج مع العالم الخارجي.

من تجهيزات الإدخال المألوفة نذكر لوحة المفاتيح والفأرة ولواقط الصوت والكاميرات والماسحات الضوئية.

أما من تجهيزات الإخراج فنذكر الشاشات والطابعات ومكبرات الصوت.

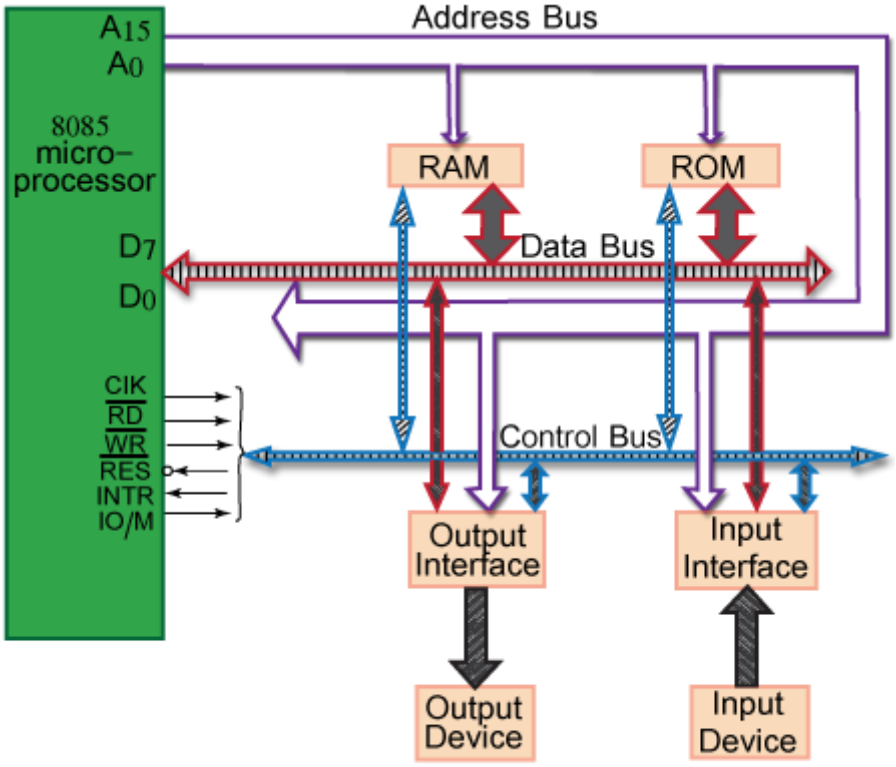
4.2. المسرى:

يعتبر المسرى (bus) بشكل أساسي وسيلة الاتصال بين وحدة المعالجة المركزية (CPU) والتجهيزات الطرفية وهو عبارة عن مجموعة من الأسلاك التي تحمل المعلومات على شكل بتات.

تتصل أجزاء نظم المعالجة الحاسوبية بعضها ببعض من خلال ثلاث مساري وهي:

1. مسرى العناوين: أحادي الاتجاه و يحدد عنوان الذاكرة أو بوابة الطرفية المراد النفاذ إليها.
2. مسرى المعطيات: ثنائي الاتجاه و يحمل المعطيات التي يكتبها المعالج أو تلك التي يقرؤها.
3. مسرى التحكم: مجموعة الإشارات الضرورية لسير عملية النفاذ بشكل صحيح.

يوضح الشكل التالي مثالاً عن نظام معالجة وطريقة ربط مكوناته ، يبين الشكل مساري الربط بين المعالج والطرفيات حيث نعتبر في هذا لمخطط معالجاً على 8 بتات مثل المعالج Intel 8085.



مساري الربط بين المعالج والطرفيات

1. مسرى العناوين:

وهو مسرى أحادي الاتجاه ويستخدم من قبل CPU لإرسال عنوان الذاكرة المراد النفاذ إليها أو لانتقاء بوابة دخل خرج معينة يمكن أن يتكون المسرى من 8 أو 16 أو أكثر من الخطوط المتوازية حيث يحدد عدد البتات في المسرى القيمة العظمى لعدد مواقع لذاكرة التي يمكن النفاذ إليها. مثلاً إن مسرى عناوين بعرض 16 بت يسمح بالنفاذ إلى 2^16 موقع مختلف من المعطيات.

نرمز إلى خطوط مسرى العناوين بالرموز A1،A0 ،... ،An حيث يمثل n عرض مسرى العناوين (بوحدة البت).

2. مسرى المعطيات:

هو مسرى ثنائي الاتجاه تمر عبره المعطيات بين المعالج والطرفيات. إن لعرض مسرى المعطيات تأثيراً كبيراَ على بنية المعالج حيث يحدد عرض هذا المسرى عدد بتات المعطيات التي يمكن التعامل معها ومعالجتها في وقت واحد.

يمكن أن يختلف عرض مسرى المعطيات الداخلي الذي يربط وحدة المعالجة لمركزية مع بقية المكونات داخل المعالج عن عرض مسرى المعطيات الخارجي الذي يربط المعالج بالذواكر والطرفيات الخارجية.

يحدد عرض مسرى المعطيات أكبر رقم يمكن أن بتعامل معه المعالج خلال عملية واحدة. فمثلاً، إن أكبر رقم يمكن معالجته دفعة واحدة في معالج ذو مسرى معطيات بعرض 8 بت هو 255.

نرمز إلى خطوط مسرى العناوين بالرموز D1،D0 ،... ،Dn حيث يمثل n عرض مسرى المطيات (بوحدة البت).

3. مسرى التحكم:

يحتوي مسرى التحكم على مجموعة من الخطوط المنفردة التي تحمل إشارات التزامن. يصل مسرى التحكم إشارات التحكم إلى الذواكر تجهيزات الدخل خرج والطرفيات الأخرى لضمان صحة تنفيذ العمليات التي يقوم بها المعالج. ويحتوي مسرى التحكم على إشارات التحكم مثل إشارة القراءة (read) وإشارة الكتابة (write) وإشارة المقاطعة (interrupt) وإشارة المسك أو الحجز (hold).

تخبر إشارة المقاطعة (interrupt) المعالج بأن طرفية خارجية تحتاج إلى أن تخديم.

أما إشارة المسك (hold) فإنها تسمح لطرفية ما مثل متحكم النفاذ المباشر للذاكرة DMA (Direct Memory Access) بإخبار المعالج بأنها تريد استخدام مسرى العناوين ومسرى المعطيات.

مثال:

إذا كان المطلوب قراءة محتوى موقع ما في الذاكرة فسوف:

1. يرسل المعالج أولاً عنوان ذلك الموقع على مسرى العناوين.

2. يرسل المعالج إشارة القراءة على مسرى التحكم.
3. تجيب الذاكرة بوضع محتوى الموقع المطلوب على مسرى المعطيات ليقوم المعالج بأخذ هذه المعطيات.

5.2. أنواع النظم الحاسوبية:

يمكن تقسيم نظم المعالجة إلى قسمين:

1. نظم المعالجة ذات الأهداف العامة القابلة للبرمجة:

وهي نظم يمكن للمستثمر النهائي أن يقوم بإعادة برمجتها مثل أجهزة الحاسوب الشخصية.

2. نظم المعالجة المدمجة:

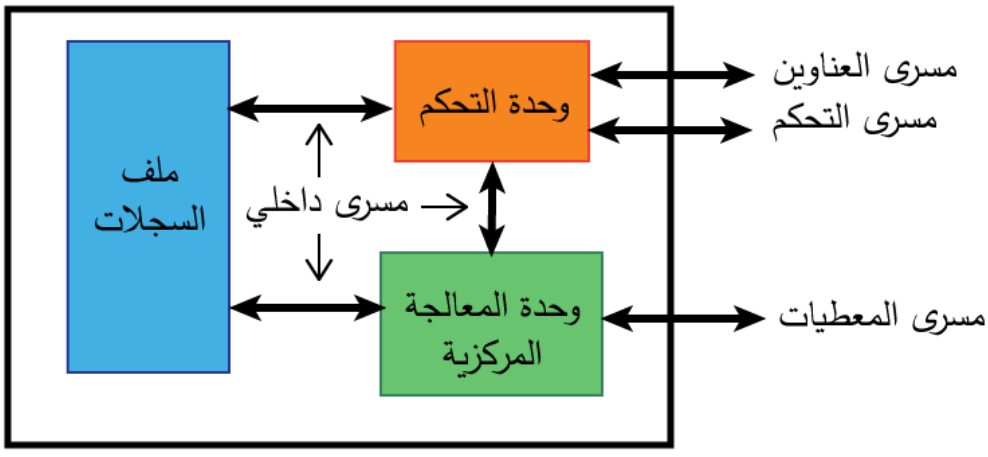
نظم المعالجة المدمجة (Embedded Systems) وهي نظم تقوم بمهمة محددة وتبرمج من قبل المصنع ولا يمكن للمستخدم أن يقوم بإعادة برمجتها مثل أجهزة الهاتف النقال والغسالات الآلية وأفران الميكروويف ...الخ.

في معظم هذه النظم فإن المعالج والذاكرة وبوابات الدخل والخرج تكون موجودة ضمن نفس الدارة التكاملية التي تعرف باسم المتحكم الصغري (Microcontroller) والتي سنتعرض لدراستها في فصل لاحق.

3. ماذا يوجد داخل المعالج:

يبين الشكل التالي مخططاً مبسطاً لمكونات المعالج، حيث يتكون من الكتل الرئيسية التالية:

1. وحدة الحساب والمنطق ALU (Arithmetic Logic Unit): تقوم بالعمليات الحسابية والمنطقية.
2. ملف السجلات (Register File): وهو مجموعة من السجلات التي تستخدم لتخزين العناوين والمعطيات التي يتعامل معها المعالج في كل لحظة من تنفيذ البرنامج.
3. وحدة التحكم: إدارة عملية تنفيذ البرنامج من جلب للتعليمات وتفكيكها وتنفيذها بمساعدة وحدة الحساب والمنطق وملف السجلات.



نموذج مبسط لبنية المعالج

1.3. وحدة الحساب والمنطق:

تشكل وحدة الحساب والمنطق ALU (Arithmetic Logic Unit) قلب أي معالج. تقوم هذه الوحدة بجميع العمليات الرياضية والمنطقية في المعالج. وهي مكونة من دارات منطقية ولها مدخلين للمعطيات ومخرج للمعطيات ومخرج حالة. تأخذ هذه الوحدة معطيات الدخل من سجلات المعالج وتعالجها حسب التعليمة المرغوب تنفيذها في وحدة التحكم ومن ثم تخزن النتيجة في سجلات الخرج. تستخدم كل المعالجات الحديثة المعطيات الثنائية بصيغة المتمم الثنائي.

تشمل العمليات التي تقوم بها وحدة الحساب والمنطق الجمع والطرح وكذلك العمليات المنطقية AND وOR وNOT وXOR. كما أن بعض المعالجات تقوم أيضاً بعمليات الضرب والقسمة كما تشمل العمليات التي تقوم بها وحدة الحساب والمنطق عمليات الإزاحة والمقارنة.

2.3. ملف السجلات:

يتكون ملف السجلات (Register File) من العديد من السجلات المستخدمة بشكل أساسي لتخزين المعطيات والعناوين ومعلومات الحالة خلال تنفيذ البرنامج. يتم بناء السجلات باستخدام دارت منطقية وقلابات (flip-flop). ومن السجلات الموجودة في معظم المعالجات:

1. عداد البرنامج:

عداد البرنامج (program counter) هو سجل يخزن عنوان التعليمة التالية التي سيتم تنفيذها وبالتالي يلعب هذا السجل دوراً مركزياً في التحكم بتسلسل التعليمات التي ينفذها المعالج. بعد قراءة التعليمة من الذاكرة يتم زيادة عداد البرنامج بشكل آلي بمقدر واحد. هذا بالطبع بفرض أن التعليمات تنفذ تباعاً.

ويتأثر محتوى هذا السجل بتعليمات القفز وتعليمات استدعاء التوابع الجزئية. ففي حالة تعليمة القفز فإنه يتم أولاً شحن عداد البرنامج بعنوان التعليمة لتي سوف يتم القفز إليها وبعد ذلك يتم زيادة العداد بشكل عادي حتى الوصول إلى تعليمة قفز ثانية. وعندما يصل المعالج إلى تعليمة استدعاء تابع جزئي فيتم أولاً زيادة عداد البرنامج بمقدر واحد ويتم تخزين النتيجة في ذاكرة خاصة تدعى ذاكرة المكدس (stack). ومن ثم يتم شحن عداد البرنامج بعنوان أول تعليمة في البرنامج الجزئي. يتم بعد ذلك زيادة العداد تلقائياً حتى الوصول إلى تعليمة الرجوع في نهاية التابع وعندها يتم شحن عداد البرنامج بالقيمة المخزنة في المكدس ويستمر البرنامج خطوة فخطوة حتى الوصول إلى تعليمة قفز جديدة واستدعاء تابع جزئي. من الجدير بالذكر بأن المقاطعات تؤثر على قيمة عداد البرنامج بطريقة مشابهة لعملية استدعاء تابع جزئي.

2. سجلات التعليمة:

سجلات التعليمة (instruction registers) هي سجلات تخزن رمز التعليمة التي تنفذ حالياً. تستخرج وحدة التحكم رمز العملية من سجل التعليمة وتقوم بتوليد الإشارات الضرورية لتنفيذ لتعليمة.

3. سجلات العزل:

تمثل سجلات العزل (buffer register) الواجهة البينية بين المعالج والذواكر المرتبطة به.

وسجلات العزل النظامية هي:

- سجل عنوان الذاكرة memory address register (MAR)
- سجل عزل الذاكرة memory buffer register (MBR)

يتم وصل السجل MAR مع مرابط العناوين الخارجية للمعالج ويحتوي هذا السجل على العنوان المطلق لعنوان للمعطيات أو التعليمة التي نريد النفاذ إليها. أما السجل MBR والمعروف أيضاً بسجل معطيات الذاكرة فيكون موصولاً مع مرابط المعطيات للمعالج ويخزن كل المعطيات التي يتم قراءتها أو كتابتها في الذاكرة.

4. سجل الحالة:

يخزن سجل الحالة (status register) حالة الخرج كنتيجة لأي عملية تتم في وحدة المعالجة المركزية وتعطي معلومات إضافية حول النتيجة. ويدل كل بت من بتات سجل الحالة عن حصول أو عدم حصول شرط معين. ويتم تحديث قيم هذه البتات بعد نهاية كل عملية. ومن بتات الحالة الشائعة: بت الحمل (carry)، بت الفائض (overflow)، بت الصفر (zero)، بت السالب (negative) وغيرها.

5. مؤشر المكدس:

يستخدم سجل مؤشر المكدس (stack pointer) لتخزين عنوان موقع الذاكرة التي تحتوي على آخر قيمة مخزنة في ذاكرة المكدس. في الواقع فإن المكدس هو كتلة من الذاكرة مخصصة لتخزين المعطيات بشكل مؤقت. حيث تستخدم لتخزين قيمة أي سجل عام خلل تنفيذ إجرائية جزئية أو عند تخديم مقاطعة ما. يتم نقل المعطيات من السجل العام إلى المكدس باستخدام تعليمة PUSH في بداية الإجرائية. وفي نهاية التابع يتم استعادة المعطيات من المكدس إلى السجل باستخدام التعليمة POP. يستخدم المعالج المكدس كذاكرة مؤقتة لأن عملية تخزين واستعادة المعطيات باستخدام التعليمتين PUSH و POP يتم بشكل أسرع من نقل المعطيات من و إلى الذاكرة باستخدام التعليمة MOVE.

6. سجلات عامة:

السجلات العامة (general-purpose registers) هي سجلات لاستخدام العام. وتستخدم بشكل صريح تخزين معومات المعطيات والعناوين. تستخدم سجلات المعطيات للعمليات الحسابية والمنطقية. أما سجلات العناوين فتستخدم للعنونة الغير مباشرة كمؤشرات ذاكرة. وبفضل هذه السجلات يتم تسريع تنفيذ الخوارزميات عن طريق النتائج المرحلية وتجنب النفاذ إلى الذاكرة الخارجية لهذا الغرض. وذلك لأن النفاذ إلى الذاكرة الخارجية يكون أبطأ من النفاذ إلى سجلات المعالج الداخلية.

في المعالجات التي ظهرت باكراً كان هناك سجل عام واحد يسمى المراكم (accumulator) وكان يلزم أربعة تعليمات من أجل تنفيذ تعليمة جمع بسيطة ويتضمن ذلك جلب معطيات القيمة الأولى من الذاكرة الخارجية ومن ثم جمعها مع القيمة الثانية في الذاكرة ومن ثم وضع النتيجة في المراكم وأخيرة تخزين محتوى المراكم في الذاكرة الخارجية. مع توافر للعديد من السجلات العامة في المعالجات الحديثة صار ممكناً تنفيذ العديد من العمليات الحسابية دون الحاجة إلى النفاّذ إلى الذاكرة من أجل تخزين النتائج المرحلية.

7. سجلات مؤقتة:

تستخدم السجلات المؤقتة (temporary registers) لتخزين المعطيات إذا لزم الأمر خلال تنفيذ التعليمة. وهي سجلات مخفية بشكل كامل عن مستخدم المعالج.

3.3. وحدة التحكم:

تقوم وحدة التحكم بتنظيم وتنسيق عمل كل أقسام المعالج والتجهيزات الطرفية.

- قراءة التعليمة من الذاكرة
- فك ترميز هذه التعليمة
- إرسال الإشارات المناسبة لبقية أجزاء النظام لتنفيذ التعليمة

تقوم وحدة التحكم بتنظيم وتنسيق عمل كل أقسام المعالج والتجهيزات الطرفية. تعد هذه الوحدة مسؤولة عن التحكم بحلقة البحث عن التعليمة من الذاكرة وتنفيذها. كما تقوم بتنسيق عمل جميع تجهيزات الدخل والخرج. ولذلك هي بلا شك الكتلة الأعقد داخل لمعالج وتشغل معظم مساحة الدارة التكاملية للمعالج.

تتكون هذه الوحدة من دارات منطقية تقوم بتشغيل المعالج من خلال سلسلة من الخطوات المتزامنة. وترسل مجموعة إشارات التحكم ونبضات التزامن لمختلف مكونات المعالج ومرابطه الخارجية.

وتوضيحاً لذلك، من أجل تنفيذ تعليمة ما من الذاكرة، ترسل وحدة التحكم أمر قراءة إلى الذاكرة لقراءة التعليمة وتستقبل التعليمة من الذاكرة من خلال مسرى المعطيات. من ثم تقوم وحدة التحكم بفك ترميز هذه التعليمة وتقوم بإرسال الإشارات المناسبة لوحدة الحساب والمنطق وسجلات المعالج وغيرها من المكونات لكي يتم تنفيذ هذه التعليمة بشكل صحيح. فإذا كانت التعليمة مثلاً تطلب تخزين معطيات في موقع من الذاكرة فإن وحدة التحكم

ستقوم بإرسال عنوان موقع الذاكرة على مسرى العناوين والمعطيات المراد تخزينها على مسرى المعطيات وإشارة الكتابة في مسرى التحكم.

يمكن تصنيف وحدات التحكم وفقاً لطريقة بنائها إلى نمطين:

1. وحدات تحكم عتادية: RISC(Reduced Instruction Set Computer):

- بنية هذه الوحدة صغيرة وسريعة
- لكنها صعبة التصميم

في وحدات التحكم العتادية (hard-wired control units) تتكون وحدة التحكم في هذا النمط من دارت منطقية تقوم بتوليد تسلسل الإشارات المناسبة لكل تعليمة من التعليمات. وتكون بنية هذه الوحدة صغيرة وسريعة ولكنها صعبة التصميم. يكون عدد التعليمات الممكنة في هذا التصميم قليلاً ولذلك يعرف هذا التصميم أيضاً باسم حاسب ذو مجموعة تعليمات محدودة RISC (Reduced Instruction Set Computer).

2. وحدات تحكم مرمزة CISC(Complex Instruction Set Computer):

- تصميم وحدة التحكم من هذا النمط سهلاً
- مرونة أكثر في تنفيذ التعليمات ولكن على حساب السرعة

في وحدات التحكم المرمزة (microcoded control units)، يكون تصميم وحدة التحكم من هذا النمط سهلاً ويتم تنفيذ تعليمة ما في هذه لحالة من خلال تنفيذ سلسلة من التعليمات الجزئية الصغيرة. يقدم هذا النمط من وحدات التحكم مرونة أكثر في تنفيذ التعليمات من وحدات التحكم العتادية ولكن على حساب السرعة. ولذلك فإن عدد التعليمات الممكنة في هذا التصميم يكون كبيراً ويعرف هذا التصميم باسم معالج ذو مجموعة تعليمات معقدة CISC (Complex Instruction Set Computer).

4. لمحة موجزة عن تاريخ المعالجات وتطورها:

1.4. لمحة موجزة عن تاريخ المعالجات:

- عام 1946: أول جيل من حواسيب ENIAC مصمم باستخدام أنابيب الأشعة المهبطية
- عام 1958: ظهور أول حاسب باستخدام الترانزستور TRADIC من قبل شركة IBM.
- عام 1959: اختراع الدارة التكاملية.
- عام 1960: بداية استخدام الدارات التكاملية في بطاقات وحدة المعالجة المركزية.
- عام 1971: اختراع أول معالج ضمن دارة تكاملية واحدة. المعالج Intel 4004 على 4 بتات ومكون من 2250 ترانزستور من شركة انتل.
- أواخر السبعينات: ظهور المعالجين Intel 8080، Intel 8085 على 8 بتات ومسرى عناوين على 16 بت.

- عام 1981: ظهور أول حاسب شخصي من قبل شركة IBM باستخدام المعالج Intel 8088.
- الثمانينات: انتشار المعالج MC6800 من شركة Motorola وبداية ظهور حواسيب ماكنتوش Apple Macintosh باستخدام المعالج MC68000.

الشرح:

لقد تم اختراع أول معالج كدارة متكاملة واحدة في عام 1971 من قبل شركة انتل (Intel) وهو المعالج Intel 4004 والذي كان معالجاً يستخدم مسرى معطيات بعرض 4 بتات فقط وكان يتكون من 2250 ترانزستور.

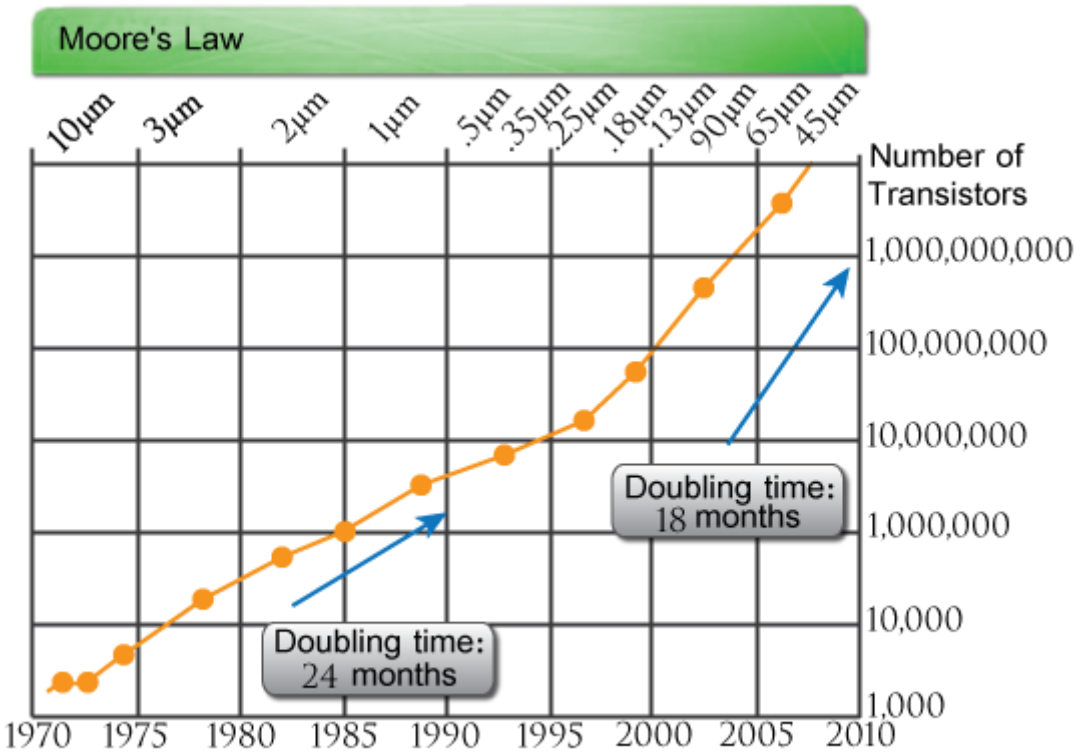
أما قبل هذا التاريخ فقد كان يتم تصميم نظم المعالجة الرقمية والحاسبات الالكترونية باستخدام عناصر الكترونية منفصلة. حيث ظهر أول جهاز حاسوب الكتروني والمسمى ENIAC في عام 1946 باستخدام تقنية مصابيح الأشعة المهبطية (vacuum tube).

وبعد ظهور الترانزستور تم تصنيع أول جهاز حاسوب والمسمى TRADIC باستخدام الترانزستورات من قبل شركة IBM في عام 1958. وفي عام 1959 تم تصنيع أول دارة متكاملة والذي سمح بتطوير أجهزة حاسوب باستخدام بطاقات تحتوي على دارات متكاملة.

2.4. تطور المعالجات:

قانون مور Moore في تطور الدارات المتكاملة:

إن درجة مكاملة الدارات المتكاملة وفقاً للكلفة الدنيا يتضاعف تقريباً كل سنتين.



قانون مور في تطور الدارات المتكاملة

تطور معالجات عائلة 80x86:

Processor	Year	Transistors	Clock Rate (MHz)	External Data Bus	Internal Data Bus	Address Bus
4004	1971	2.25K	0.108	4	8	12
4008	1972	3.5K	0.200	8	8	14
8080	1974	6K	3	8	8	16
8085	1976	6K	6	8	8	16
8086	1978	29K	10	16	16	20
8088	1979	29K	10	8	16	20
80286	1982	134K	12.5	16	16	25
80386DX	1985	275K	33	32	32	32
80386SX	1988	275K	33	16	32	24
Pentium C	1993	3.1M	66–200	64	32	32
Pentium MMX	1997	4.5M	300	64	32	32
Pentium Pro	1995	5.5M	200	64	32	36
Pentium II	1997	7.5M	233–450	64	32	36
Pentium III	1999	9.5M	550–733	64	32	36
Itanium	2001	30 M	800–…	128	64	64

جدول يبين تطور معالجات عائلة 80x86

وبعد إنتاج أول معالج في عام 1971 تطور تعقيد المعالجات ودرجة مكاملتها بشكل متسارع وفقاً للقانون المعروف بقانون مور Moore والذي يقول بأن درجة مكاملة الدارات المتكاملة وفقاً للكلفة الدنيا يتضاعف تقريباً كل سنتين.

بعد إنتاج أول معالج تم إنتاج معالجات أخرى بأربعة بتات مثل المعالج Intel 4040 من شركة Intel والمعالج PPS-4 من شركة Rockwell والمعالج T3472 من شركة Toshiba وغيرها.

تم تطوير أول معالج على 8 بتات والمسمى Intel 8008 من قبل شركة Intel في عام 1972 ومن ثم المعالج Intel 8080 في عام 1973 ومن ثم المعالج Intel 8085 في عام 1975 الذي أصبح شائع الاستخدام بشكل كبير.

ومن المعالجات الأخرى بثمانية بتات نذكر المعالج Z80 من شركة Zilog عام 1976 والمعالج MC6800 من شركة Motorola عام 1974 والمعالج MC6809 لنفس الشركة عام 1978. وغيرها من المعالجات من شركات أخرى.

ظهر أول معالج على 16 بت متعدد ولكن مكون من عدة دارات متكاملة. أما أول معالج 16 بت على دارة واحدة فقد كان المعالج TMS9900 من شركة Texas Instrument. وكان أول معالج 16 بت من شركة انتل فقد كان المعالج Intel 8086 في عام 1978. وبعده أنتجت هذه الشركة العديد من المعالجات من Intel 80186 إلى Pentium و Itanium كما هو مبين في الجدول التالي.

5. معايير اختيار المعالج:

يوجد في السوق الآلاف من المعالجات. إن اختيار المعالج المناسب لتطبيق ما ليس بالمهمة السهلة. حيث يجب الأخذ بعين الاعتبار عند اختيار معالج لتطبيق ما العديد من العوامل.

1. السعر
2. استهلاك الطاقة
3. الأداء
4. التوفر
5. الدعم البرمجي
6. كثافة الرماز

1. السعر:

إن من أهم عوامل اختيار المعالج هو سعر المعالج إذ يجب أن يكون اقتصادياً ولا سيما من أجل تطبيقات النظم المدمجة المحدودة السعر.

2. استهلاك الطاقة:

يعد استهلاك الطاقة من العوامل المهمة من أجل النظم التي تعمل على البطارية. وتتغير الاستطاعة المستهلكة من قبل المعالج تبعاً لجهد التغذية وسرعة التشغيل. وكذلك يجب الأخذ بعين الاعتبار بنية المسرى الذي يربط المعالج بالذواكر.

3. الأداء:

إن كان معالجاً ما جيداً بالنسبة لتطبيق ما فليس بالضرورة أن يكون جيداً بالنسبة لتطبيق آخر. وبالتالي يجب معرفة درجة تعقيد البرامج التي سيشغلها المعالج ومتطلباتها من حيث السرعة وحجم الذاكرة ومتطلبات حجم الذاكرة اللازمة للتطبيقات المعتبرة.

4. التوفر:

قبل اختيار معالج ما، يجب التأكد أولاً من أنه متوفر ومن الممكن الحصول عليه بسهولة.

5. الدعم البرمجي:

من أجل تطوير البرامج اللازمة لعمل المعالج فمن الضروري الحصول أيضاً على الأدوات البرمجية الضرورية وذلك فقد يكون هذا الأمر العامل الحاسم في اختيار معالج ما دون آخر بسبب توفر البرمجيات اللازمة للتطوير مثل المترجمات والمحاكيات.

6. كثافة الرماز:

تعرف كثافة الرماز(code) بالنسبة بين حجم رماز المصدر إلى حجم النسخة التنفيذية. فكلما كانت النسخة التنفيذية أصغر كان ذلك أفضل وذلك لأنها تتطلب حجم ذاكرة أقل. وبشكل عام تعتبر المعالجات من نوع RISC أقل كفاءة من حيث كثافة الترميز مقارنة بالمعالجات من نوع CISC.

مع ذلك فإنه لا يوجد فقط معالج واحد يناسب تطبيق ما حيث يمكن تؤدي عدة معالجات الوظيفة المرجوة بنفس الكفاءة ومتقاربة في مختلف الاعتبارات السابقة، وهنا يمكن أن يلعب دور مدى انتشار استخدام كل معالج والتجارب السابقة مع مختلف المعالجات المنتقاة.

أسئلة الفصل الأول

أسئلة عامة

1. ما هو المعالج الصغري؟

المعالج الصغري هو دارة الكترونية متكاملة قابلة للبرمجة.

2. ما هي المكونات الأساسية في النظام الحاسوبي؟

يتألف أي نظام معالجة حاسوبي من معالج وذاكرة وتجهيزات دخل وخرج.

3. هناك ثلاث أنواع من المساري التي تربط المعالج مع الذواكر والطرفيات الخارجية، فما هي؟

مسرى المعطيات ومسرى العناوين ومسرى التحكم.

4. ما الفرق بين الذاكرة الحية والذاكرة الميتة؟

الذاكرة الحية هي الذاكرة التي تفقد المعلومات المخزنة فيها بعد زوال التغذية الكهربائية عنها.
أما الذاكرة الميتة فتحافظ على المعلومات المخزنة فيها بعد زوال التغذية الكهربائية عنها.

5. ما هي الأقسام الرئيسية في المعالج؟

يتكون المعالج من وحدة الحساب والمنطق ALU، و ملف السجلات، ووحدة التحكم.

6. ما هي وظيفة سجل عداد البرنامج في المعالج؟

هو سجل يخزن عنوان التعليمة التالية التي سيتم تنفيذها.

7. ما هي وظيفة سجل الحالة في المعالج.

يحتوي هذا السجل على بتات كل منها يعبر عن تحقق شرط معين بعد تنفيذ كل تعليمة مثل بت الحمل وبت الفائض وبت الصفر وغيرها.

8. ما هو المكدس وكيف يتم النفاذ إلى هذه الذاكرة وما هو دور سجل مؤشر المكدس؟

المكدس هو كتلة من الذاكرة مخصصة لتخزين المعطيات بشكل مؤقت. يستخدم سجل مؤشر المكدس (stack pointer) لتخزين عنوان موقع الذاكرة التي تحتوي على آخر قيمة مخزنة في ذاكرة المكدس.

9. ما هو دور وحدة التحكم في المعالج؟

تقوم وحدة التحكم بتنظيم وتنسيق عمل كل أقسام المعالج والتجهيزات الطرفية.

10. قارن بين وحدات التحكم العتادية RISC ووحدات التحكم المرمزة CISC من حيث سهولة التصميم والسرعة وعدد التعليمات التي تدعمها.

وحدات التحكم المرمزة أبسط من حيث التصميم وتدعم عدد تعليمات أكثر ولكنها أقل سرعة من وحدات التحكم العتادية.

11. على ماذا ينص قانون "مور" في الدارات المتكاملة؟

ينص قانون مور على أن درجة مكاملة الدارات المتكاملة وفقاً للكلفة الدنيا يتضاعف تقريباً كل سنتين.

أسئلة خيارات متعددة

1. في أي عام تم إنتاج أول معالج صغري على دارة تكاملية واحدة؟

A. 1946
B. 1969
C. 1971
D. 1973

2. ما هو عدد مواقع الذاكرة التي يمكن للمعالج النفاذ إليها باستخدام مسرى عناوين على 8 بت.

A. 8
B. 64
C. 256
D. 512

3. ما هو أكبر عدد يتعامل معه معالج له مسرى معطيات على 8 بت؟

A. 8

B. 255

C. 256

D. 511

4. تستخدم السجلات العامة في المعالج بهدف:

A. تخزين تعليمات المعالج قبل إرسالها إلى وحدة الحساب والمنطق.

B. تخزين حالة وحدة الحساب والمنطق بعد إجراء كل عملية.

C. تنظيم عمل وحدة التحكم في المعالج.

D. تخزين المعطيات والعناوين التي تتعامل معها وحدة الحساب والمنطق بشكل مؤقت.

5. إن المعيار الأهم في اختيار المعالج بالنسبة للنظم المحمولة هو:

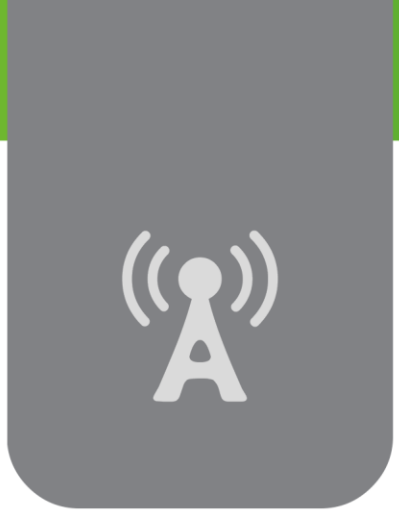
A. السعر

B. الدعم البرمجي

C. استهلاك الطاقة

D. الأداء من حيث السرعة

رقم التمرين	الإجابة الصحيحة
1	C
2	C
3	B
4	D
5	C



الفصل الثاني: البنية الداخلية للمعالجIntel 8086

الكلمات المفتاحية:

المعالج 8086، بنية المعالج، وحدة المعالجة المركزية، فضاء العنونة، السجلات، وحدة الحساب والمنطق.

ملخص:

نُبيّن في هذا الفصل البنية الداخلية للمعالج Intel 8086 ومواصفاته العامة وآلية عمله في تنفيذ البرنامج ودور السجلات الداخلية في هذا العمل. كما نبين مفهوم فضاء العنونة وكيفية النفاذ إلى الذاكرة واللتجهيزات الخارجية من خلال مساري العناوين والمعطيات والتحكم.

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

- المواصفات العامة للمعالج Intel 8086
- البنية العامة للمعالج والوحدات الوظيفية فيه
- تقسيم السجلات الداخلية ودورها
- كيفية عمل المعالج مفهوم فضاء العنونة
- إشارات المسرى وكيفية النفاذ للذواكر والطرفيات الدخل والخرج

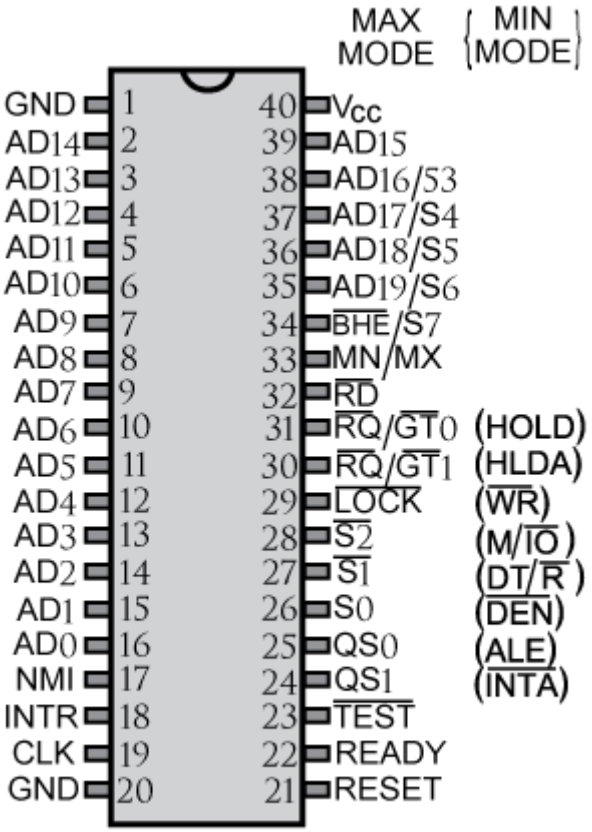
نُبيّن في هذا الفصل البنية الداخلية للمعالج Intel 8086 ومواصفاته العامة وآلية عمله في تنفيذ البرنامج ودور السجلات الداخلية في هذا العمل. كما نبين مفهوم فضاء العنونة وكيفية النفاذ إلى الذاكرة والتجهيزات الخارجية من خلال مساري العناوين والمعطيات والتحكم.

المعالج Intel 8086 هو أول معالج من معالجات عائلة 80x86 تم إصداره في عام 1978 كأول معالج يعمل على 16 بت. وهو مصمم باستخدام 29000 ترانزستور.

1. المواصفات الأساسية للمعالج:

يمكن تلخيص المواصفات الأساسية للمعالج بما يلي:

1. يعمل على معطيات 16 بت.
2. له مسرى عناوين بعرض 20 بت أي أنه يمكنه النفاذ إلى ذاكرة بسعة 1 ميغا بايت.
3. يستطيع النفاذ إلى 64K.
4. يعمل باستخدام ساعة تصل إلى 10 MHz.
5. يعمل باستخدام تغذية وحيدة 5V.
6. معلب ضمن دارة تكاملية ذات 40 مربط.



مرابط المعالج ٨٠٨٦

ينتمي المعالج 8086 إلى سلسلة المعالجات الصغرية ذات 16 بت، ويقصد بذلك أن وحدة المعالجة المركزية، وسجلاته الداخلية، ومعظم تعليماته صممت لتتعامل مع الكلمات الثنائية المرمزة على 16 بت. فالمعالج 8086 له 16 خطاً لنقل المعطيات، فيمكنه إذاً قراءة أو كتابة كلمات مرمزة على 8 بتات أو 16 بت في الذاكرة أو في أحد المعابر.

ويحتوي مسرى المعالج على 20 خط عنونة، وهذا ما يجعل عدد المواقع التي يستطع المعالج الوصول إليها هو 2^{20} موقعاً (اي 1048576 بايت). وتخزن الكلمات المرمزة على 16 بت في الذاكرة باستخدام موقعين متتاليين (كل موقع يحتوي على كلمة من 8 بتات). فإذا كان البايت الأول موجود في موقع ذي عنوان زوجي، فإن المعالج يستطيع الوصول إلى الكلمة بتمامها بعملية واحدة. أما إذا كان عنوان البايت الأول فردياً، فإن المعالج سيقرؤها في دور أول، ثم سيقرأ البايت الآخر في دور آخر.

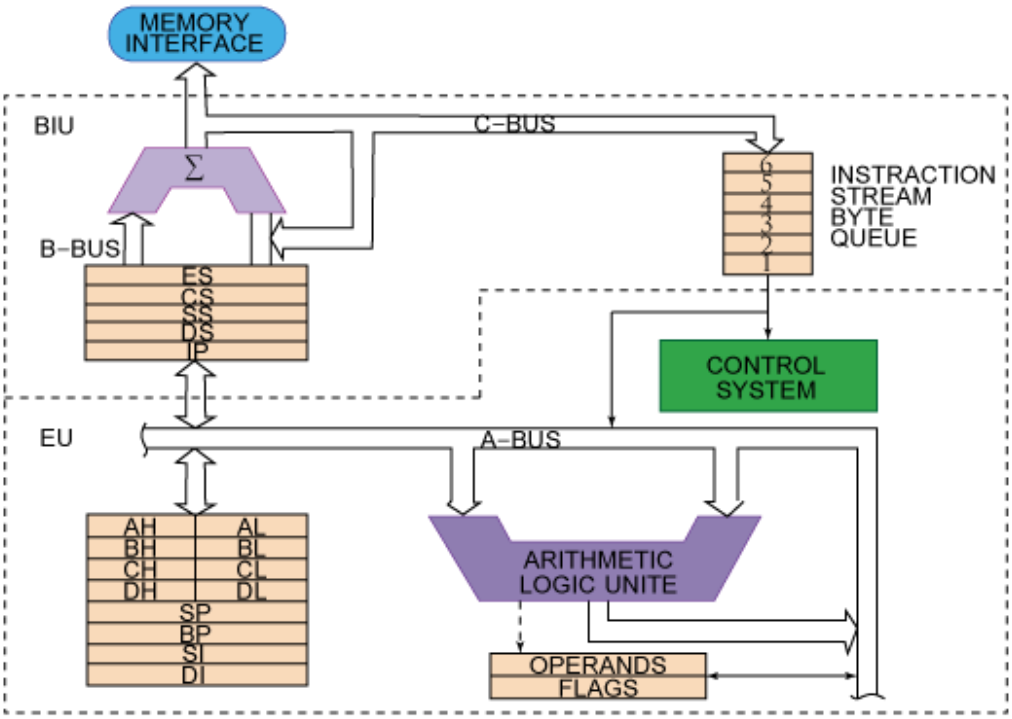
ومن الضروري، قبل كتابة برامج للمعالج المصغري 8086، فهم بنيته الداخلية ومعرفة سجلاته الداخلية.

2. البنية الداخلية:

تتألف وحدة المعالجة المركزية من جزأين أساسيين كما هو مبين في الشكل وهما:

- وحدة واجهة المسرى BIU (Bus Interface Unit): تضع هذه الوحدة العناوين على المسرى، وتجلب التعليمات من الذاكرة، وتقرأ المعطيات من الذاكرة والمعابر، وتكتب النتائج في الذاكرة والمعابر.
- وحدة التنفيذ EU (Execution Unit): تخبر وحدة التنفيذ في المعالج 8086 وحدة واجهة المسرى عن مكان جلب التعليمات أو المعطيات، وتقوم بتفكيك التعليمات وتنفيذها.

ويسمح تقسيم العمل بين هاتين الوحدتين بتسريع المعالجة. حيث تستطيع هاتان الوحدتين العمل على التوازي. في الوقت الذي تقوم فيه وحدة واجهة المسرى بجلب تعليمات من الذاكرة، تستطيع وحدة التنفيذ إجراء التعليمات التي جلبت سابقاً بشرط أن لا تكون هذه التعليمات بحاجة للنفاذ إلى المسرى.



البنية الداخلية للمعالج ٨٠٨٦

1.2. وحدة واجهة المسرى:

تضع هذه الوحدة العناوين على المسرى، وتجلب التعليمات من الذاكرة، وتقرأ المعطيات من الذاكرة والمعابر، وتكتب النتائج في الذاكرة والمعابر. بمعنى آخر، تضطلع الوحدة BIU بكافة عمليات نقل المعطيات والعناوين على المسرى. وسنشرح فيما يلي عمل الأجزاء الوظيفية المكونة لهذه الوحدة.

1.1.2. الرتل:

لزيادة سرعة تنفيذ البرنامج تجلب الوحدة BIU سلفاً ست بايتات من الذاكرة وتحفظ هذه البايتات الست داخل مجموعة سجلات تعمل بطريقة «الداخل أولاً يخرج أولاً» FIFO. وهذا يعني أن البايت الأول سيقرأ أولاً ليرسل إلى وحدة التنفيذ، يليه البايت الثاني وهكذا. تسمى تلك السجلات الرتل Queue.

من جهة أخرى، تستطيع وحدة واجهة المسرى أن تجلب التعليمات وتخزنها في الرتل أثناء قيام وحدة التنفيذ بتفكيك التعليمة أوتنفيذها، والذي لا يتطلب استخدام المسرى.

عندما تصبح وحدة التنفيذ جاهزة للتعليمة التالية، فإن عليها أن تقرأ فقط التعليمة الجديدة من رتل وحدة واجهة المسرى، وهذا أسرع بكثير من جلبها من الذاكرة. فالجلب يتطلب وضع العناوين على المسرى وإرسال إشارة القراءة، ثم قراءة الكلمة من خطوط المعطيات.

وهذه الطريقة تماثل إلى حد بعيد عمل البَنّاء. فبدلاً من أن يبحث البَنّاء كل مرة عن قطعة الآجر ويرصفها، يستطيع أن يضع بالقرب مته مخزوناً صغيراً من الآجر، يصل إليه بسرعة، ويقوم مساعده بتعبئة هذا المخزون باستمرار.

تنفذ جميع تعليمات المعالج 8086 وفق الطريقة المذكورة،باستثناء تعليمتي القفز والاستدعاء CALL & JUMP. ففي تلك الحالة، ينبغي افراغ الرتل من التعليمات، ثم شحنه ثانيةً بدءاً من العنوان الجديد. وتسمى آلية جلب التعليمة الجديدة أثناء تنفيذ التعليمة الحالية بالتواردية Pipelining.

2.1.2. سجلات القطاعات:

يقسم فضاء عنونة الذاكرة إلى أربعة قطاعات، كل منها بطول 64 كيلوبايت كحد أقصى، وهي:

- قطاع ذاكرة البرنامج: يستخدم لتخزين البرنامج.
- قطاع ذاكرة المكدس: يستخدم لتخزين العناوين والمعطيات المرحلية بشكل مؤقت خلال تنفيذ البرنامج.
- قطاع ذاكرة المعطيات: يستخدم لتخزين المعطيات التي يتعامل معها البرنامج.
- قطاع ذاكرة إضافي: قطاع ذاكرة يمكن أن يستخدم لأغراض أخرى.

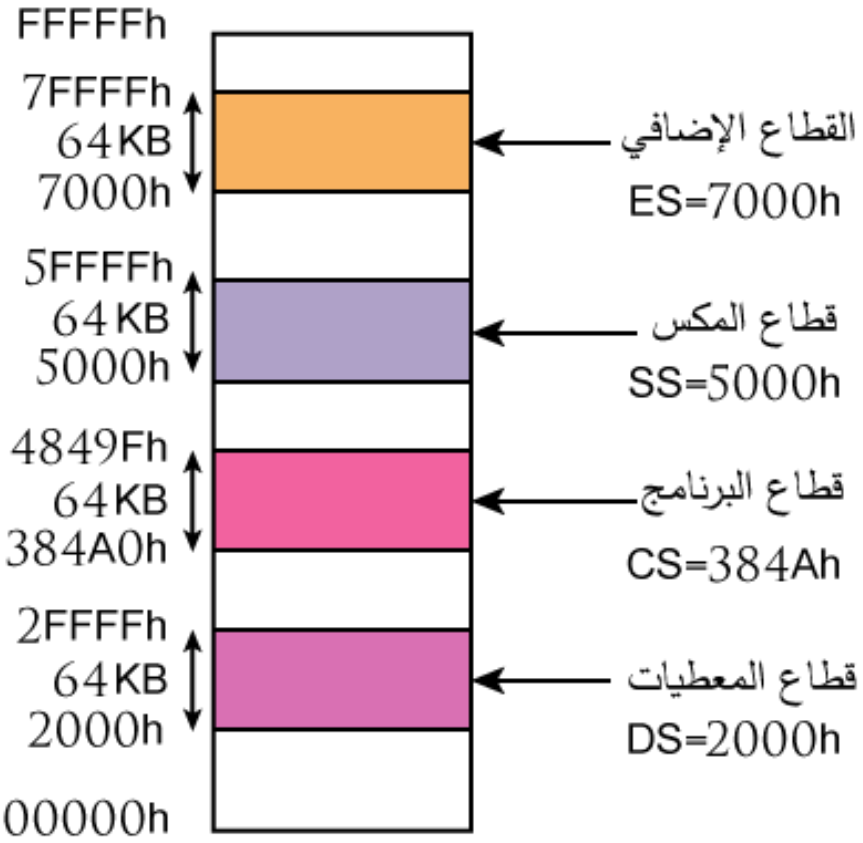
ويمكن أن تكون هذه القطاعات مستقلة عن بعضها البعض وهذه هي الحالة الطبيعية، أو أن تتراكب فيما بينها في بعض الحالات الخاصة.

يوجد أربعة سجلات كل منه على 16 بت تستخدم لتخزين الجزء العلوي (16 بت) من عنوان بداية القطاع الموافق. وهذه السجلات هي:

1. سجل قطاع البرنامج CS (Code Segment)
2. سجل قطاع المكدس SS (Stack Segment)
3. سجل قطاع المعطيات DS (Data Segment)
4. سجل القطاع الإضافي ES (Extra Segment)

لما كان فضاء العنونة في المعالج على 20 بت، فيكون عنوان بداية كل قطاع بوضع جميع البتات الأربعة السفلية المتبقية من العنوان أصفاراً. فمثلاً: اذا كانت قيمة السجل CS=348Ah فيكون عنوان بدية قطاع البرنامج هو 348A0h.

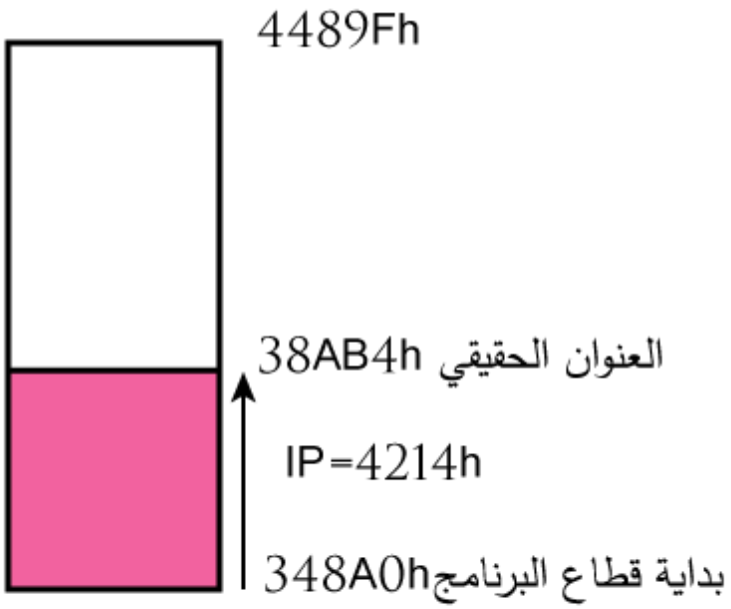
يتم عنونة كل قطاع ابتداءاً من الصفر. فمثلاً يكون عنوان البايت الخامس من قطاع البرنامج هو ناتج جمع عنوان بداية القطاع مع دليل الموقع وهو هنا 348A0h+00005h=348A5h.



توزع القطاعات الأربعة في فضاء عناوين المعالج

3.1.2. مؤشر التعليمات:

وهو سجل على 16 بت يستخدم لتخزين عنوان الموقع الذي ستجب منه وحدة واجهة المسرى التعليمة المقبلة من قطاع البرنامج. أي يمثل الانزياح عن عنوان بداية قطاع البرنامج. وبالتالي للحصول على عنوان الذاكرة الموافق تقوم وحدة واجهة المسرى بجمع محتوى هذا السجل مع عنوان بداية قطاع البرنامج على 20 بت وهو كما ذكرنا قيمة سجل قطاع البرنامج CS بعد إضافة أربعة أصفار في البتات الدنيا. ويبين الشكل آلية جمع االنزياح المخزن في السجل IP إلى العنوان القاعدي CS.



إضافة محتوى السجل IP إلى السجل CS لتوليد العنوان الحقيقي للتعليمة المقبلة

2.2. وحدة التنفيذ:

تخبر وحدة التنفيذ في المعالج 8086 وحدة واجهة المسرى عن مكان جلب التعليمات أو المعطيات، وتقوم بتفكيك التعليمات وتنفيذها. وهي تحتوي على الأجزاء الوظيفية التالية:

1. دارات التفكيك
2. وحدة الحساب والمنطق
3. سجل الرايات
4. السجلات ذات الاستخدام العام
5. سجل مؤشر المكدس SP
6. سجلات الدليل

1.2.2. دارات التفكيك:

تقوم دارات التفكيك بترجمة التعليمات المقروءة من الذاكرة إلى سلسلة من الأفعال التي تنفذها وحدة التنفيذ بواسطة وحدة الحساب والمنطق.

2.2.2. وحدة الحساب والمنطق:

توكل إى هذه الوحدة مهمة تنفيذ العمليات الحسابية الأساسية، مثل الجمع والطرح، والعمليات المنطقية البسيطة. وهي تنفذ هذه العمليات على حدين، كل منهما مرمز على 16 بت. وللدلالة على نتائج عملها، تتصل هذه الوحدة بسجل خاص يسمى سجل الرايات أو سجل الحالة.

3.2.2. سجل الرايات:

يطلق اسم راية على البت الذي يدل على تحقق شرط معين من جراء تنفيذ تعليمة ما، أو ايعازاً محدداً لوحدة التنفيذ. وتحتوي وحدة التنفيذ في المعالج عى سجل رايات طوله 16 بت، منها 9 بتات ذات دلالة وسبعة غير مستخدمة.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

سجل الرايات في المعالج 8086.

- راية الحمل CF (Carry Flag): وتعبر عن وجود حمل أو استعارة بعد عملية الجمع أو الطرح.
- راية الزوجية PF (Parity Flag): تكون مساوية للواحد إذا كان عدد البتات الواحدية في القيمة الناتجة زوجياً.
- راية الحمل المساعد AF (Auxiliary Flag): وتعبر عن وجود حمل ناتج بعد جمع البايت الأدنى من معاملات الجمع.
- راية الإشارة SF (Sign Flag): تكون مساوية للواحد إذا كان ناتج العملية سالباً.
- راية الفائض OF (Overflow Flag): ويكون مساوياً للواحد عند وجود طفحان في نتيجة العملية الحسابية ويحدث مثلاُ عندما لا يمكن تمثيل ناتج الضرب على 16 بت.

أما الرايات الثلاث الباقية، وتسمى رايات التحكم، فهي تستخدم في قيادة بعض العمليات في المعالج، ويمكن لمبرمج الكتابة في هذه الرايات لى خلاف الرايات الست الأولى التي تكتبها وحدة التنفيذ

- راية التنفيذ الخطوي TF (Trap Flag): وتسمح، عندما تصبح قيمتها مساوية للواحد، بتنفيذ البرنامج خطوة فخطوة

- راية المقاطعة IF (Interrupt Flag): وهي تفيد في سماح أو منع المقاطعات في المعالج
- راية الاتجاه DF (Direction Flag): وهي تستخدم أثناء تنفيذ التعليمات الخاصة بسلاسل المحارف

4.2.2. السجلات ذات الاستخدام العام:

يوجد لوحدة التنفيذ ثمانية سجلات عامة كل منها على 8 بت وهي:

.AL, AH, BL, BH, CL, CH, DL, DH

ويمكن استخدام كل منها بشكل منفصل. ويسمى السجل AL المراكم (Accumulator) لأنه يتمتع بصفات خاصة.

من جهة أخرى، يمكن دمج كل سجلين معاً لتخزين كلمة على 16 بت، فيدمج السجلن AL وAH لحصول على زوج يسمى AX. وكذلك يدمج BL وBH للحصول على BX, ونفس الشيئ بالنسبة للبقية.

5.2.2. سجل مؤشر المكدس SP:

يستخدم هذا السجل لتخزين الانزياح عن بداية قطاع المكدس لموقع آخر قيمة تم وضعها في المكدس.

6.2.2. سجلات الدليل:

تحتوي وحدة التنفيذ، إضافة إى مؤشر المكدس، ثلاثة سجلات طول كل منها 16 بت، وهي: مؤشر القاعدة BP (Base Pointer)، ودليل المصدر SI (Source Index)، ودليل الوجهة DI (Destination Index). يمكن استخدام هذه السجلات في تخزين المعطيات الثانوية كي سجل عام. إلا أن استخدامها ارئيسي هو تخزين انزياح معلومة ما في أحد قطاعات الذاكرة. فعلى سبيل المثال، يستخدم السجل SI لتخزين انزياح كلمة في قطاع المعطيات، وهذا يعني أن عنوان تلك الكلمة الحقيقي ينتج من إزاحة محتوى السجل DS أربعة بتات نحو اليسار وإضافة محتوى السجل SI إلى النتيجة.

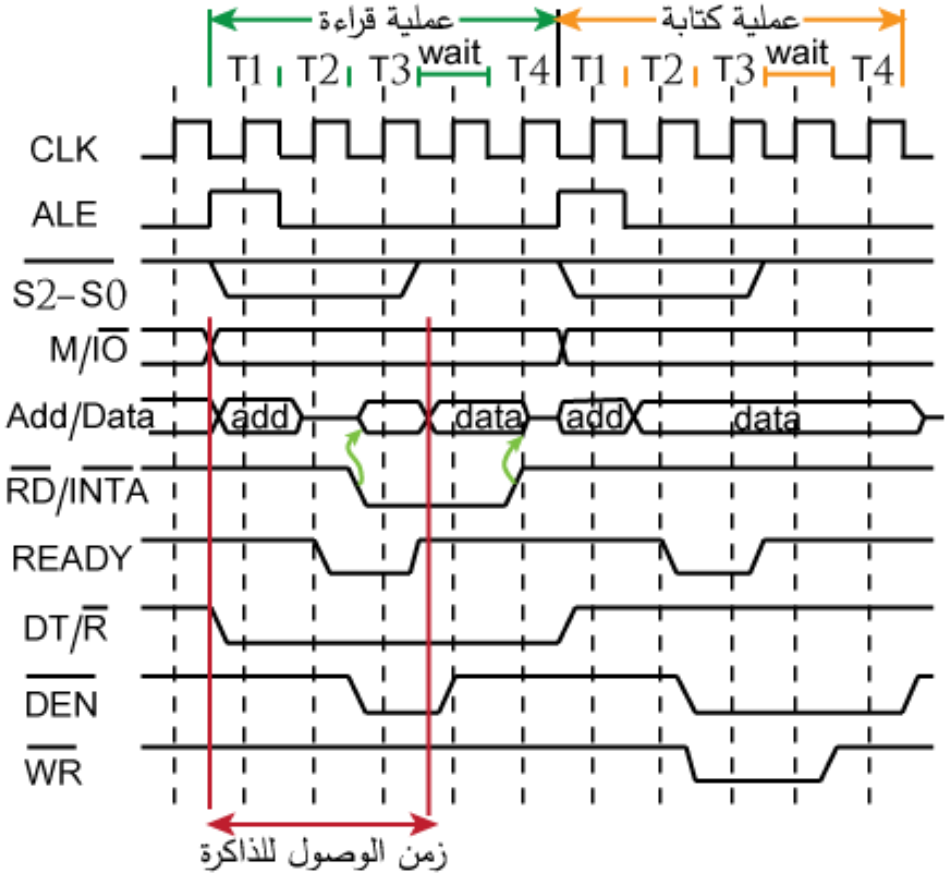
3. إشارات المسرى:

ذكرنا سابقاً أن المعالج هو وحدة تقوم بتنفيذ التعليمات المخزنة في الذاكرة. ولكي يستطيع المعالج تنفيذ تعليمة ما ينبغي أن يقرأ هذه التعليمة من الذاكرة.

بعد التحليل يقوم المعالج بتنفيذ التعليمة (سواء تضمنت اتعليمة عملية حسابية أو منطقية). وقد يضطر المعالج، تبعاً لنوع التعليمة، إلى تخزين النتيجة في الذاكرة.

بمعنى آخر، لا بد للمعالج من القيام بعمليات قراءة وكتابة في الذاكرة أو في إحدى الطرفيات الأخرى. ويستطيع ذلك بواسطة الإشارات الالكترونية التي يتبادلها مع الذاكرة أو الطرفيات.

نصف في هذه الفقرة تسلسل إشارت المتبادلة على مسرى المعالج أثناء عمليتي القراءة والكتابة.



إشارات المسرى أثناء عمليتي القراءة والكتابة

1.3. عملية القراءة من الذاكرة:

عندما يقرأ المعالج قيمة ما من الذاكرة فإنه يستخدم خطوط العنونة والمعطيات AD0−15 وخطوط العنونة العليا A16−19 وخطوط التحكم التالية:

$$\overline{RD}, ALE, M/\overline{IO}, S0-S2, READY, \overline{DEN}, DT/\overline{R}$$

يتضمن تنفيذ تعليمة ما في المعالج عدة عمليات أساسية (وهي جلب التعليمة وتحليلها وتنفيذها وتخزين النتائج). تحتاج كل عملية أساية إى عدد من أدوار الساعة CLK للتنفيذ، وتسمى كل عملية منها دور الآلة Machine Cycle. كما يطلق اسم دور التعليمة Instruction Cycle على مجموعة أدوار الساعة اللازمة لتنفيذ تعليمة ما تنفيذاً كاملاً. فيحتوي إذاً دور التعليمة على دور أو أكثر من أدوار الآلة، وهذا يحتوي بدوره على مجموعة أدوار من الساعة.

أثناء القراءة يؤهل المعالج أولاً الإشارة $M/\overline{IO}$ فيضعها على المستوى العالي (5V) إذا أراد مخاطبة الذاكرة، أو على المستوى المنخفض (0V) في حال مخاطبة معبر دخل/خرج.
ولتحليل المخطط ينبغي تتبع الإشارات من اليسار نحو اليمين.

بعد وضع قيمة $M/\overline{IO}$ المناسبة، يضع المعالج عنوان الذاكرة (أو المعبر) المراد قراءته مستخدماً مسرى اعناوين وفي الوقت ذاته، يولد نبضة على الخرج ALE للدلالة على جاهزية العنوان الموجود على الخطوط AD0−15. كما يحدد المعالج طول الكلمة المطلوبة بواسطة الإشارة $\overline{BHE}$.

بعد وضع العناوين، يستعد المعالج لقراءة القيمة من الذاكرة (أو من المعبر) فيؤهل إشارة القراءة $\overline{RD}$، وهذا ما يدعو الطرفية أو الذاكرة إى وضع المعطيات على الخطوط.
أما إشارة الجاهزية READY فهي تدل على أن الذاكرة أو الطرفية جاهزة لتبادل المعلومات مع المعالج. فإذا وجد المعالج على هذا الدخل القيمة 0 (أي أن إشارة الجاهزية غير فعالة) فإنه سيدخل في حالة انتظار wait State إلى أن تصبح الطرفية جاهزة، وعندئذٍ يستأنف عمله.

هناك إشارتان يخرجهما المعالج لقيادة دارات مساعدة خارجية وهما $DEN, DT/\overline{R}$ تدل الإشارة الأولى $DT/\bar{R}$ على اتجاه المعطيات. فمثلاً في حالة القراءة، تأخذ هذه الإشارة القيمة 0 للدلالة على أن المعلومات تتجه من اذاكرة نحو المعالج. أما الإشارة $\overline{DEN}$، فهي تتأهل عندما تظهر امعطيات على خطوط العنونة والمعطيات AD0−15.

2.3. عملية الكتابة في الذاكرة:

يظهر الشكل أيضاً مخططاً لإشارات المسرى أثناء عملية الكتابة. ويبدو، بالمقارنة بحالة القراءة، أن هناك تشابهاً واضحاً في تسلسل العمل.

يضع امعالج أولاً القيمة امناسبة عى الخرج $M/\overline{IO}$ لتحديد هوية الوجهة (ذاكرة أم معبر)، ثم يخرج عنوان اموقع المطلوب الكتابة فيه على خطوط العنونة والمعطيات. ويؤهل الإشارة $\overline{BHE}$ عند نعامله مع كلمة مرمزة على 16 بت. بعد ذلك، يولد نبضة على الخرج ALE للدلالة على جاهزية العناوين على المسرى.

أما المرحلة التالية، فهي مختلفة عن حالة القراءة. إذ يخرج المعالج القيمة المراد كتابتها على الخطوط AD0– 15، ويؤهل إشارة الكتابة $\overline{WR}$ للإيعاز للذكرة أو المعبر بالكتابة. يدخل بعدئذٍ المعالج في حالة انتظار لإشارة الجاهزية READY، وحين وصولها ينتقل إى تنفيذ التعليمة التالية.

يضع المعالج، أثناء عملية الكتابة، القيمة المناسبة على الخط $DT/\overline{R}$ لتحديد اتجاه المعلومات. ففي هذه الحالة، تكون قيمة الإشارة 1، لأن المعلومات تتجه من المعالج نحو الذاكرة أو الطرفية. ويؤهل الخط $\overline{DEN}$ فور خروج امعطيات على خطوط المسرى تنبيه الدارات الخارجية إى وجود المعطيات.

أسئلة الفصل الثاني

أسئلة عامة

1. ما هي الأقسام الأساسية في البنية الداخلية للمعالج الصغري 8086؟

يتكون المعالج الصغري 8086 من جزأين أساسيين وهما: وحدة واجهة المسرى BIU ووحدة التنفيذ EU.

2. ما هي قطاعات الذاكرة المستخدمة في المعالج 8086 وما هي السجلات المستخدمة للدلالة عليها؟

1. قطاع ذاكرة البرنامج ويعرف باستخدام السجل CS.
2. قطاع ذاكرة المكدس، ويعرف باستخدام السجل SS.
3. قطاع ذاكرة المعطيات ويعرف باستخدام السجل DS.
4. قطاع ذاكرة إضافي، ويعرف باستخدام السجل ES.

3. ماهي السجلات العامة على 16 بت في المعالج 8086؟

السجلات هي : AX و BX و CX و DX.

4. ما هي سجلات الدليل في المعالج 8086 وما هو الاستخدام الأساسي لها؟

السجلات هي: سجل مؤشر القاعة BP وسجل دليل المصدر SI وسجل دليل الوجهة DI. استخدامها الرئيسي هو تخزين انزياح معلومة ما في أحد قطاعات الذاكرة.

5. ما هي وظيفة المربط $M/\overline{IO}$ في المعالج 8086؟

يستخدم هذا المربط للتميز بين عمليات النفاذ إلى الذاكرة وبين عمليات النفاذ إلى معابر الدخل والخرج. حيث يضع المعالج هذا المربط على المستوى العالي (5V) إذا أراد مخاطبة الذاكرة، أو على المستوى المنخفض (0V) في حال مخاطبة معبر دخل/خرج.

أسئلة خيارات متعددة

1. ما هو عرض مسرى المعطيات في المعالج 8086؟
   A. 8
   B. 16
   C. 20
   D. 32

2. ما هو عرض مسرى العناوين في المعالج 8086؟
   A. 12
   B. 16
   C. 20
   D. 32

3. ما هو أكبر عدد يتعامل معه معالج له مسرى معطيات على 8 بت؟
   A. 8
   B. 255
   C. 256
   D. 511

4. كم يبلغ عدد سجلات الرتل في واجهة المسرى BIU في المعالج 8086؟
   A. 2
   B. 4
   C. 6
   D. 8

5. ما هي وظيفة سجلات الرتل في واجهة المسرى BIU في المعالج 8086؟
   A. تخزين التعليمات التي تنتظر التنفيذ في وحدة التنفيذ EU.
   B. تخزين معطيات وحدة التنفيذ EU بشكل مؤقت.
   C. تخزين عناوين التعليمات التي سيتم جلبها من الذاكرة الخارجية عبر واجهة المسرى BIU.
   D. حفظ عناوين بداية مختلف قطاعات الذاكرة المستخدمة في البرنامج.

6. ما هو العنوان الحقيقي للتعليمة عندما يكون سجل مؤشر المعطيات IP=0050h وسجل قطاع التعليمات CS=1000h؟

A. 00050h

B. 01000h

C. 01050h

D. 10050h

7. عن ماذا تعبر الراية ZF في سجل الحالة للمعالج 8086؟

A. راية الحمل

B. راية الصفر

C. راية الزوجية

D. راية الإشارة

8. ما هو دور السجل SP في المعالج 8086؟

A. مؤشر القاعدة.

B. مؤشر البرنامج

C. سجل الحالة

D. مؤشر المكدس

رقم التمرين	الإجابة الصحيحة
1	B
2	C
3	B
4	C
5	A
6	D
7	B
8	D

الفصل الثالث: الوحدات المحيطية والدخل والخرج

الكلمات المفتاحية:

تجهيزات الدخل والخرج، بوابات العبور، النفاذ إلى بوابات العبور، دارة بوابات العبور، دارة المتحكم بالمقاطعات، دارة المؤقت. المخطط الزمني للنفاذ.

ملخص:

نستعرض في هذا استراتيجيات النفاذ بين الأجهزة المحيطية والذاكرة الأساسية عن طريق المعالج. كما ندرس أهم تجهيزات العبور المستخدمة مثل دارة بوابات العبور ودارة التحكم بالمقاطعات ودارة المؤقت وغيرها. وكذلك كيفية إدارة عمليات التخاطب مع الدارات المحيطية حسب طبيعة استخدامها من قبل المعالج.

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

- استراتيجيات نفاذ المعالج إلى طرفيات العبور
- طرق وصل المعالج مع الطرفيات
- التعرف على دارة بوابات العبور
- التعرف على دارة التحكم بالمقاطعات
- التعرف على دارة المؤقت

نستعرض في هذا الفصل استراتيجيات النفاذ بين الأجهزة المحيطية والذاكرة الأساسية عن طريق المعالج.
كما ندرس أهم تجهيزات العبور المستخدمة مثل دارة بوابات العبور ودارة التحكم بالمقاطعات ودارة المؤقت وغيرها. وكذلك كيفية إدارة عمليات التخاطب مع الدارات المحيطية حسب طبيعة استخدامها من قبل المعالج.

1. استراتيجيات العبور:

من أهم استراتيجيات النفاذ بين تجهيزات العبور والذاكرة الأساسية:

1. طرق العبور المبرمجة (Programmed IO):

- تقوم وحدة المعالجة المركزية بتنفيذ عمليات العبور بشكل كامل.
- لا تحتاج عمليات العبور المبرمجة إلى تجهيزات وعناصر خاصة.
- تأخذ الكثير من وقت المعالج في إجراء عمليات روتينية.
- يمكن استخدام المقاطعات لإعلام المعالج عن توفر معطيات جديدة لدى جهاز العبور لتوفير وقت المعالج.
- تستخدم للعمليات التي تتطلب معدل نقل معطيات منخفض.

2. طرق العبور بالنفاذ المباشر إلى الذاكرة DMA (Direct Memory Access):

- عمليات العبور التي تقوم دارة خارجية بتنفيذها لتخفيف العبء عن المعالج.
- يتطلب تنفيذها وجود دارة خارجية قادرة على توليد عناوين الذاكرة ونقل المعلومات من أو إلى الذاكرة، إضافة إلى إمكان طلب استحواذ المسرى و آلية انتخاب.
- تجري على التوازي مع عمل المعالج ولكن تبقى تحت إشرافه الكامل.
- تستخدم للعمليات التي تتطلب معدل نقل معطيات مرتفع.

ذكرنا في الفصل السابق إن معظم الوظائف الحسابية الرئيسية للنظم الحاسوبية تحتاج إلى مكونين رئيسين هما وحدة المعالجة المركزية والذاكرة، ولكن النظام الحاسوبي يحتاج أيضاً إلى أجهزة عبور بغرض نقل المعطيات بين المعالج والذاكرة من جهة، والعالم الخارجي من جهة ثانية.

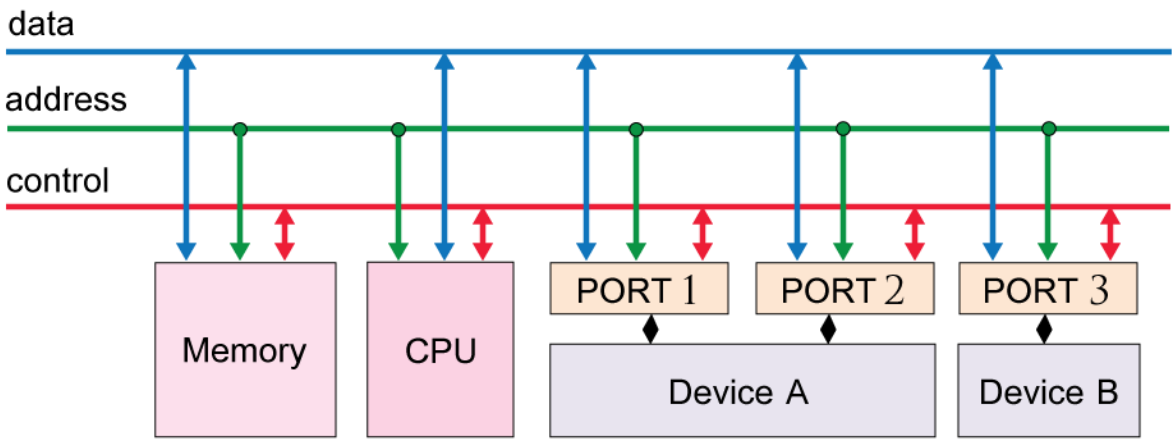
يمكن تسمية عمليات العبور التي تقوم وحدة المعالجة المركزية بالتحكم التام بها بعمليات العبور المبرمجة (Programmed IO). حيث لا تحتاج عمليات العبور المبرمجة إلى تجهيزات وعناصر خاصة، ولكنها تأخذ الكثير من وقت المعالج في إجراء عمليات روتينية.

لتوفير هذا الوقت، يمكن أيضاً القيام بنقل كتلة من المعطيات بين جهاز عبور والذاكرة دون تدخل وحدة المعالجة المركزية، وذلك بزيادة تعقيد بنية التحكم في العبور بدرجة طفيفة نسبياً. ويتطلب ذلك أن يكون جهاز العبور قادراً على توليد عناوين الذاكرة ونقل المعلومات من أو إلى الذاكرة، إضافة إلى إمكان طلب استحواذ المسرى و آلية انتخاب. في هذا النمط من العمل ، تبقى وحدة المعالجة المركزية هي المسؤولة عن إطلاق عملية نقل كل كتلة معطيات، ومن ثم يقوم جهاز العبور بإجراء عملية النقل دون الحاجة إلى أن تقوم وحدة المعالجة المركزية بتنفيذ أي برنامج. نسمي هذا النمط من العبور بالنفاذ المباشر إلى الذاكرة DMA (Direct Memory Access).

كذلك قد تُزَوَّد أجهزة العبور بدارات خاصة تمكنها من طلب الخدمة من وحدة المعالجة المركزية، أي طلب تنفيذ برنامج محدد لتخديم جهاز العبور، يسمى هذا الطلب بالمقاطعة. تسمح هذه الإمكانية بتحرير وحدة المعالجة المركزية من عبء اختبار حالة دارة العبور دورياً. تسبب المقاطعة انتقال وحدة المعالجة المركزية إلى تنفيذ برنامج معالجة المقاطعة، ومن ثم يعود إلى متابعة تنفيذ البرنامج الأساسي عند الانتهاء من خدمة المقاطعة.

2.عمليات العبور المبرمجة:

يبين الشكل مخطط وصل الأجهزة الطرفية مع المعالج عن طريق البوابات.



وصل أجهزة الطرفية مع المعالج عن طريق البوابات

سنبحث أولاً في طريقة التحكم بالعبور على وجهٍ مبرمج، أي أن كل عملية نقل معطيات من جهاز العبور تتم عن طريق تنفيذ تعليمات معينة من قبل المعالج. فمثلاً تحتاج عملية نقل كلمة معطيات من جهاز العبور إلى الذاكرة إلى تنفيذ تعليمتين من قبل المعالج هما: إدخال كلمة المعطيات من جهاز العبور إلى وحدة المعالجة المركزية، وتعليمة تخزين لنقل الكلمة من وحدة المعالجة المركزية إلى الذاكرة.

1.2. عنونة الدخل والخرج:

- يتخاطب المعالج مع كل من أجهزة العبور والذاكرة الأساسية بواسطة مسرى وحيد مشترك باستخدام نفس خطوط العنونة
- بوابة عبور (I/O port): هي كل عقدة وصل بين المسرى الأساسي وجهاز العبور ويسند إليها عنوان فريد.
- طرق عنونة أجهزة العبور:

- العبور المحجوز من الذاكرة (memory-mapped I/O): تعامل بوابات العبور معاملة الذاكرة تماماً.
- العبور عن طريق فضاء عنونة مستقل: وذلك باستخدام إشارات تحكم مستقلة عن عمليات النفاذ للذاكرة.

يتخاطب المعالج مع كل من أجهزة العبور والذاكرة الأساسية بواسطة مسرى وحيد مشترك باستخدام نفس خطوط العنونة. تسمى كل عقدة وصل بين المسرى الأساسي وجهاز العبور بوابة عبور (I/O port) ويسند إليها عنوان فريد. قد يتضمن جهاز العبور سجل تخزين مؤقت للمعطيات وهذا ما يجعله يبدو من وجهة نظر المعالج مشابهاً للذاكرة الأساسية. وبالتالي يمكن لمعالج النفاذ إلى جهاز العبور بنفس الطريقة التي يستخدمها للنفاذ إلى الذاكرة. نسمي هذه الطريقة في التعامل مع أجهزة العبور بالعبور المحجوز من الذاكرة (memory-mapped I/O).

هناك طريقة أخرى شائعة الاستخدام في النفاذ إلى أجهزة العبور يتم فيها فصل فضاء عنونة الذواكر عن فضاء عنونة أجهزة العبور كما هي الحال في المعالج 8086. في هذا الطريقة يتم تخصيص خط تحكمي مثل الخط $M/\overline{IO}$ في مسرى التحكم للتمييز بين الذواكر وأجهزة العبور. وذلك ينعكس إلى وجود تعليمات للنفاذ إلى أجهزة العبور مختلفة عن تلك المستخدمة للنفاذ إلى الذاكرة. من الجدير بالذكر هنا، كما هو مبين في المواصفات الأساسية للمعالج، بأن فضاء عنونة أجهزة العبور محدود على 16 بت فقط (فضاء بطول 64KB) وليس على 20 بت كما هي الحال بالنسبة للذواكر.

2.2. تعليمات العبور الأساسية:

يمكن تحقيق العبور المبرمج باستخدام تعليمتين فقط:

- تعليمة الكتابة: تقوم التعليمة OUT X بنقل كلمة من 8 بتات من المراكم إلى بوابة العبور ذات العنوان X.
- تعليمة القراءة: تقوم التعليمة IN X بنقل كلمة من 8 بتات من بوابة العبور ذات العنوان X إلى المراكم.

يفترض المعالج، عند استخدام هاتين التعليمتين، بأن بوابة العبور المعنونة جاهزة للاستجابة للتعليمة المنفذة وإلا فلن تتم العملية بشكل صحيح كما هو متوقع.

3. طرق نقل المعطيات :

من أهم طرق نقل المعطيات المبرمجة:

1. طريقة التقصي (polling).
2. طريقة القدح أو المقاطعة.
3. عمليات العبور بالمصافحة الوحيدة.
4. عمليات العبور بالمصافحة المزدوجة.

عندما لا تكون بوابة العبور جاهزة بشكل دائم، فإن على المعالج أن يتأكد أولاً من جاهزية بوابة العبور قبل تنفيذ تعليمة النفاذ ويتم ذلك بطريقتين:

1.3. طريقة التقصي (polling):

- يجب أن يوفر جهاز العبور بوابة إضافية تسمى بوابة الحالة تعبر عن جاهزية بوابة المعطيات للنفاذ من قبل المعالج.
- يفحص المعالج بوابة الحالة بشكل دوري ويقوم بالنفاذ عندما يكتشف المعالج أن جهاز العبور جاهز للنفاذ.

في طريقة التقصي (polling) يجب أن يوفر جهاز العبور بوابة إضافية تسمى بوابة الحالة وهي جاهزة بشكل دائم تحتوي على معلومات تعبر عن حالة جهاز العبور بما فيها جاهزية بوابة المعطيات للنفاذ من قبل المعالج. يقوم المعالج قبل النفاذ إلى بوابة المعطيات بقراءة بوابة الحالة بشكل مستمر، وعندما يتبين بأن بوابة المعطيات جاهزة، يقوم بتنفيذ عملية النفاذ سواء كانت قراءة أم كتابة. فمثلاً، في حال كان جهاز العبور عبارة عن لوحة المفاتيح فإن على المعالج أن يتحرى حالة لوحة المفاتيح بشكل منتظم ليعرف إذا ما تم الضغط على مفتاح ما قبل أن يقوم بقراءة ما هو المفتاح الذي تم ضغطه.

2.3. طريقة القدح أو المقاطعة:

1. يقوم جهاز العبور يقوم بتوليد إشارة مقاطعة عندما تكون جاهزة للنفاذ.
2. يقوم المعالج بالنفاذ إلى بوابة جهاز العبور ضمن إجرائية الاستجابة للمقاطعة.

في الطريقة السابقة، يصرف المعالج الكثير من الوقت في تقصي جاهزية جهاز العبور قبل أن يتمكن من النفاذ إليه. بينما في طريقة المقاطعة فإن جهاز العبور يقوم بتوليد إشارة مقاطعة عندما تكون جاهزة للنفاذ وعندها فقط يقوم المعالج بالنفاذ إلى بوابة جهاز العبور ضمن إجرائية الاستجابة للمقاطعة. في مثال لوحة المفاتيح، فإن المعالج لا يقرأ بوابة المعطيات للوحة المفاتيح إلا عندما تصله إشارة مقاطعة تعلمه بأنه تم الضغط على مفتاح ما.

تطبق هاتين الطريقتين في حالة العبور البطيء للمعطيات كما هي الحال في مثال لوحة المفاتيح. ولكنها لا تصلح لوحدها لنقل المعطيات بمعدل سريع بسبب عدم الفاعلية بالنسبة لوثوقية النفاذ. فإذا أخذنا مثال الموديم كجهاز عبور يقوم بإرسال المعطيات إلى المعالج، فعلى الموديم أن يتأكد من المعالج انتهى قراءة كلمة المعطيات الحالية من بوابة العبور قبل أن يضع الكلمة التالية على البوابة. وإلا فمن الممكن ضياع بعض المعطيات نتيجة لانشغال المعالج بعمل آخر مثلاً. للتغلب على هذه المشكلة فإنه يتم تبادل المعطيات وفق مراسيم المصافحة (handshaking).

3.3. عمليات العبور بالمصافحة الوحيدة:

1. يضع الجهاز المحيطي المعطيات على مخارجه ويرسل إشارة قدح $\overline{STB}$ للمعالج الصغري.
2. يكشف المعالج إشارة القدح (إما بالتقصي أو المقاطعة) ويقرأ المعطيات. ثم يقوم المعالج بإرسال إشارة إشعار ACK إلى الجهاز المحيطي

في هذا النمط من العمليات، يضع الجهاز المحيطي المعطيات على مخارجه ويرسل إشارة قدح $\overline{STB}$ للمعالج الصغري الذي يكشف الإشارة (إما بالتقصي أو المقاطعة) ويقرأ المعطيات. ثم يقوم المعالج بإرسال إشارة إشعار ACK إلى الجهاز المحيطي لإعلامه بإمكانية إرسال الكلمة التالية.

4.3. عمليات العبور بالمصافحة المزدوجة:

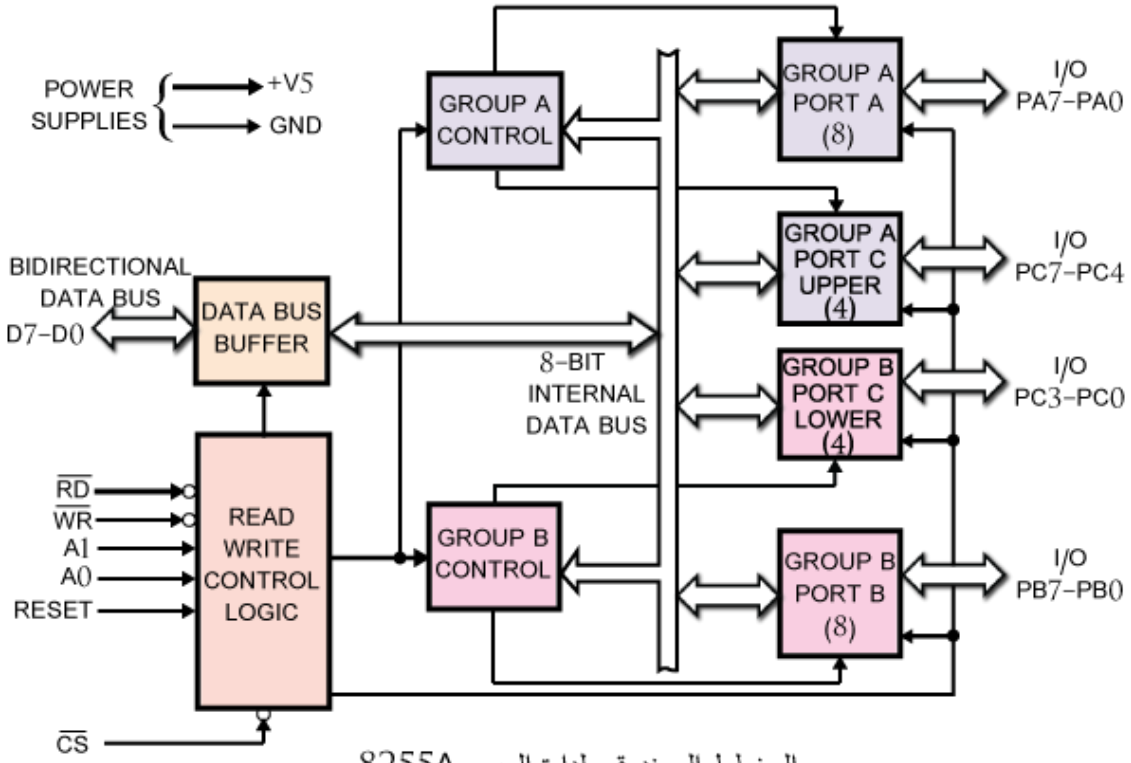
1. يستعلم الجهاز المرسل عن جاهزية المستقبل بوضع خط القدح $\overline{STB}$ على المستوى المنخفض.
2. إذا كان المستقبل جاهزاً، يقوم بإعلام المرسل عن جاهزيته بوضع خط الإشعار ACK على المستوى المنطقي العالي.
3. يضع المرسل المعطيات الجديدة على خطوط المعطيات ويجعل المستوى المنطقي على خط القدح عالياً.
4. يقوم المستقبل بقراءة المعطيات ثم يضع منطقاً منخفضاً على خط الإشعار ACK.

في هذا النمط من المصافحة، يستعلم الجهاز المرسل عن جاهزية المستقبل بوضع خط القدح على المستوى المنخفض. فإذا كان المستقبل جاهزاً، يقوم بإعلام المرسل عن جاهزيته بوضع خط الإشعار على المستوى المنطقي العالي. عندها يضع المرسل المعطيات الجديدة على خطوط المعطيات ويجعل المستوى المنطقي على خط القدح عالياً ليُعلم المستقبل بأن المعطيات الحديثة متوفرة، يقوم المستقبل بقراءة المعطيات ثم يضع منطقاً منخفضاً على خط الإشعار ACK لإعلام المرسل بأن المعطيات قد تمت قراءتها وأن المستقبل ينتظر معطيات جديدة.

4. داراة العبور المبرمجة:

تستخدم الدارة 8255A كبوابة عبور لإدخال أو إخراج معطيات ثنائية يمكن وصلها مع أجهزة طرفية خارجية كالطابعة مثلاً.

يبين الشكل التالي المخطط الصندوقي لدارة العبور 8255A من شركة Intel



المخطط الصندوقي لدارة العبور 8255A

تتمتع الدارة بالمواصفات التالية:

- مسرى معطيات على 8 بت.
- مسرى عناوين على 2 بت.
- تملك ثلاث بوابات دخل/خرج A و B و C كل منها على 8 بت.
- تملك بوابة تحكم داخلية C وذلك لبرمجة اتجاه البوابات الثلاث.
- مسرى التحكم يتكون من 4 إشارات: قراءة $\overline{RD}$، كتابة $\overline{WR}$، تصفير RESET، تأهيل $\overline{CS}$.

تستخدم الدارة 8255A من شركة Intel لربط أجهزة العبور مع المعالج الصغري. تملك الدارة 8 مرابط للربط مع مسرى المعطيات ومربطين A0, A1 للربط مع مسرى العناوين. وبالتالي فهي تملك 3 بوابات عبور (A, B, C) كل منها على 8 بت وبوابة رابعة للكتابة في سجل تحكم داخل الدارة لاختيار نمط عمل الدارة.

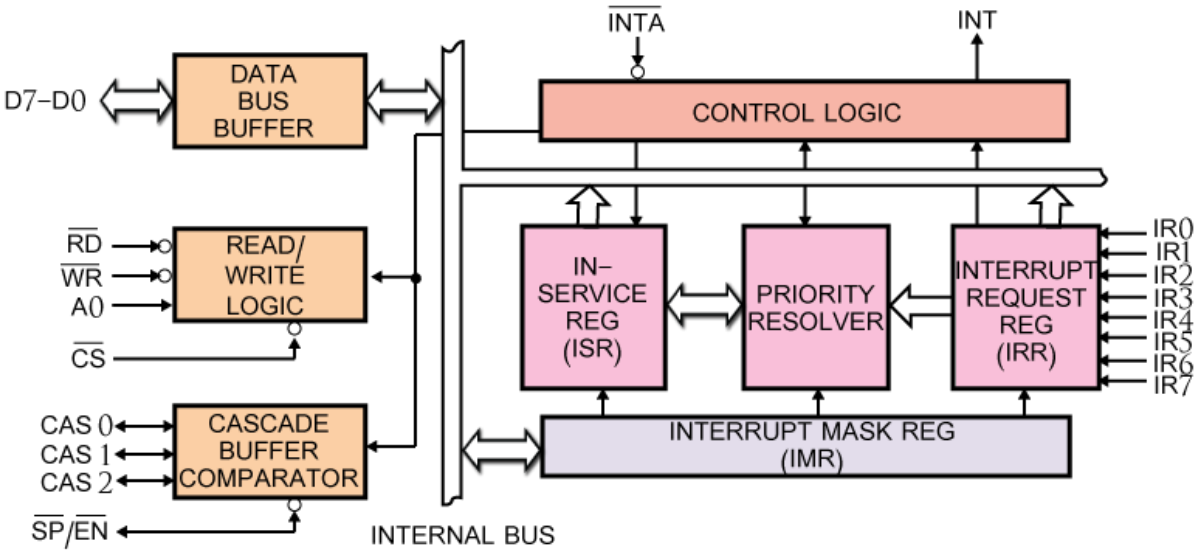
تقسم البوابة C إلى قسمين متساويين CA وCB، كل قسم على 4 بت وغالباً ما تستخدم كخطوط تحكم، مثلاً كخطوط حالة لكل من البوابتين A وB.

نستخدم كلمة التحكم لنحدد عمل البوابات A, B, C كبوابات دخل أو خرج أو كبوابات عبور ثنائية الاتجاه (وهذا متاح للمعبرين A وB فقط). كذلك يمكن برمجة بعض خطوط المعبر C لتقوم بتوليد إشارات المصافحة والمقاطعة آلياً، لورود تراكيب معينة للدخل.

يؤهل المدخل $\overline{CS}$ دارة العبور للكتابة أو القراءة، ويوصل عادة إلى خرج دارة تفكيك العنونة والانتخاب. تملك الدارة أيضاً مربطاً للتصفير (الاستهلال)، يوصل عادة إلى خط تصفير المعالج الصغري Reset، بحيث يقوم بإعادة تهيئة البوابات كافة إلى وضع ابتدائي محدد تكون فيه كافة البوابات معدة كبوابات دخل، وذلك منعاً لإخراج أية قيمة على بوابة قد تكون مربوطة مع جهاز محيطي يستخدمه كمعبر دخل، لتفادي أي تصادم بين الدارة والجهاز المحيطي.

5. المتحكم المبرمج بالمقاطعة:

- للمعالج 8086 مربطين للمقاطعات هما مقاطعة غير قابلة للحجب NMI فعالة على الجبهة الصاعدة و مقاطعة قابلة للحجب INTR. فعالة على المستوى العالي.
- تستخدم الدارة 8259 من أجل زيادة عدد المقاطعات حتى 8 مقاطعات. يبين الشكل المخطط الصندوقي لدارة العبور 8259A من شركة Intel.



المخطط الصندوقي لدارة المتحكم بالمقاطعة 8259A

- يتم ربط الدارة مع مرابط مقاطعة المعالج عن طريق المربطين INT و $\overline{INTA}$.
- عندما تحدث مقاطعة على أحد المداخل الثمانية للدارة، تولد الدارة مقاطعة على المخرج INT.
- يجيب المعالج بإشارة إشعار على المربط $\overline{INTA}$ فتقوم الدارة بوضع رقم المقاطعة التي حدثت على مسرى المعطيات.
- يقرأ المعالجة رقم المقاطعة التي حدثت وينفذ البرنامج الموافق لرقم المقاطعة التي حدثت.

للمعالج 8086 مدخلين للمقاطعة الصلبة (Hardware Interrupt) هما المدخل INTR للمقاطعة القابلة للحجب و NMI للمقاطعة غير القابلة للحجب اللذان يسمحان لإشارة خارجية بمقاطعة تنفيذ البرنامج.

يملك المعالج في داخله قلاباً يسمح بتأهيل أو حجب المقاطعة INTR، فإذا كتبت فيه القيمة 0 تحجب المقاطعة ولا يستجيب المعالج لأية إشارة على الدخل INTR، في حين يكون المعالج مؤهلاً لأن يقاطع في حال كتابة القيمة 1 في القلاب المسمى براية المقاطعة IF. وهناك تعليمات خاصة بشحن القيمة 1 أو 0 في هذا القلاب، كما سنرى في فصل البرمجة بلغة التجميع. عند الإقلاع تكون قيمة هذا القلاب صفراً، وكذلك يجري إعادة قيمته إلى الصفر أليا عند استجابة المعالج لمقاطعة خارجية، لمنع مقاطعة إجرائية تخديم المقاطعة. ولكن إذا كانت المقاطعة المخدمة ذات أولوية دنيا، يمكن تأهيل المقاطعة في بداية إجرائية تخديم المقاطعة، وذلك لسماح لمقاطعة ذات أولوية أعلى بالاستحواذ على تخديم المعالج. تقوم تعليمة IRET في نهاية إجرائية تخديم المقاطعة بإعادة تأهيل المقاطعة آلياً، أما المقاطعة NMI فهي مؤهلة دوماً.

تسبب جبهة صاعدة على المدخل NMI حدوث المقاطعة، على حين تقدح المقاطعة على المدخل NMI، عندما تكون مؤهلة، عند تحسس مستوى منطقي عالٍ على هذا المدخل.
توضع إجرائية خدمة المقاطعة للمقاطعة غير القابلة للحجب في العنوان 0008–000Bh وتملك بعض المعالجات عدداً من مداخل المقاطعة لها أولويات مختلفة وطرق معالجة متباينة. أما في المعالج 8086 فلا يوجد سوى مدخلي INTR و NMI المذكورين آنفاً. ولكن يستطيع أن ينفذ 256 نوعاً من المقاطعات (لكل منها أولوية وإجرائية تخديم المقاطعة الخاصة بها)، ولكي يتمكن المعالج من تمييز أنواع المقاطعة المختلفة نحتاج إلى دارة تسمى بالمتحكم المبرمج في المقاطعة، كالدارة 8259A.

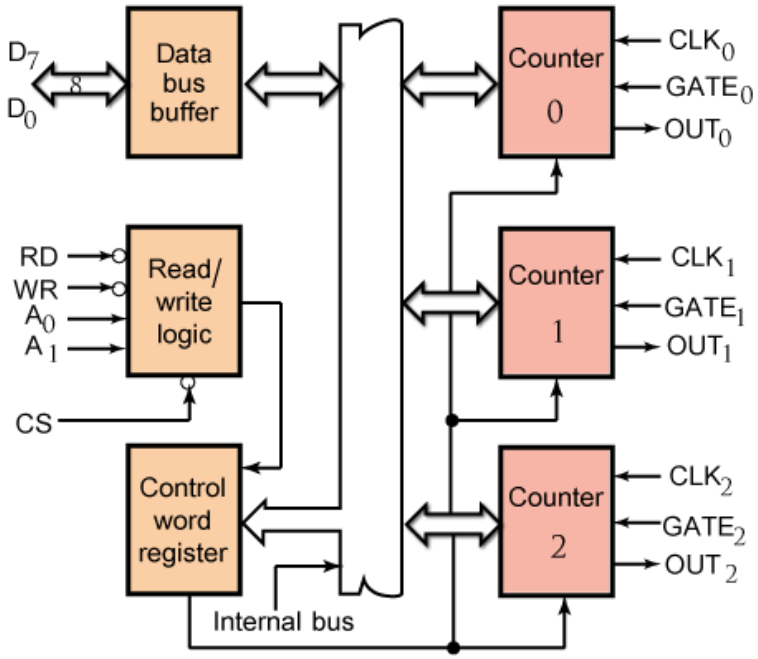
يقوم المعالج 8086، استجابة لطلب المقاطعة بتحرير المسرى، ثم يرسل نبضة إشعار باستلام المقاطعة على الخط INTA (ليعلم الجهاز الخارجي باستلام المقاطعة). يعود المعالج بعدها ليرسل نبضة ثانية على نفس خط إشعار المقاطعة ليطب من الجهاز الخارجي أن يضع رقماً يعبر عن نوع المقاطعة المطلوبة (من 0 إلى 255) على خطوط المعطيات الثمانية الأدنى لكي يقرأها المعالج.

فور قراءة المعالج لنوع المقاطعة، يستخدم هذا الرقم كموجه إلى إجرائية تخديم المقاطعة، وبذلك يمكن استخدام مدخل المقاطعة لتخديم أكثر من جهاز خارجي، لكل منها نوع من أنواع المقاطعة وعنوان إجرائية تخديم المقاطعة خاص به.

6. المؤقت المبرمج:

يستخدم المؤقت لقياس الزمن من قبل المعالج وذلك لتنفيذ مهمات معينة ضمن أزمنة محددة. مثل تشغيل وإطفاء مشير ضوئي(LED) كل واحد ثانية.

المخطط الصندوقي لدارة المتحكم بالمقاطعة 8254A.



المخطط الصندوقي لدارة المؤقت 8254A

تحتوي هذه الدارة على ثلاث مقسمات يمكن برمجة ثابت التقسيم لكل منها بشكل مستقل.

يمكن وصل خرج المؤقت لتوليد مقاطعة منتظمة للمعالج بفوارق زمنية متساوية.

من أبسط أمثلة استخدام المقاطعة الاستفادة من مدخل المقاطعة الاستفادة من مدخل المقاطعة في العد وقياس الزمن والتوقيت. بالطبع يستطيع المستثمر كتابة برنامج يحوي حلقة تأخير لتحقيق التواقت في بعض التطبيقات، إلا أن هذا يعني أن المعالج الصغري لا يقوم بأي عمل مفيد ما دام في حلقة التأخير البرمجية. لذا فمن المفيد أكثر، وصل دارة مؤقت خارجية على أحد مداخل المقاطعة للمعالج الصغري، مثلاً إذا وضعنا المعالج الصغري في منظومة لقياس درجة الحموضة لسائل ما، بحيث يجب قراءة درجة الحموضة من أحد البوابات كل أربع دقائق، في هذه الحالة يمكن وضع دارة توقيت بسيطة مكونة من مهتز يولد موجة مربعة بدور أربع دقائق توصل إلى مدخل المقاطعة NMI على سبيل المثال. تتم عملياً هذه العملية عن طريق تقسيم تردد ساعة المعالج باستخدام عناصر خارجية (عداد) للحصول على التردد المناسب للتوقيت.

تحوي معظم نظم المعالجات عدادات مثل 8253 أو 8254 من شركة Intel قابلة للبرمجة من قبل المعالج، لكي يقوم بتقسيم ساعة المعالج على أي رقم بحيث نحصل على التردد المطلوب.

أسئلة الفصل الثالث

أسئلة عامة

1. ما هي عمليات العبور المبرمجة؟

هي عمليات العبور التي تقوم وحدة المعالجة المركزية بالتحكم التام بها.

2. ما هو نمط العبور بالنفاذ المباشر إلى الذاكرة؟

هي عمليات العبور التي يقوم بها جهاز أو دارة خارج المعالج لإجراء عمليات النقل دون الحاجة إلى أن تقوم وحدة المعالجة المركزية بتنفيذ أي برنامج.

3. اشرح عملية العبور بطريقة التقصي.

يقوم المعالج قبل النفاذ إلى بوابة المعطيات بقراءة بوابة الحالة بشكل مستمر، وعندما يتبين بأن بوابة المعطيات جاهزة، يقوم بتنفيذ عملية النفاذ سواء كانت قراءة أم كتابة.

4. اشرح عملية العبور بطريقة المقاطعة.

يقوم جهاز العبور بتوليد إشارة مقاطعة عندما تكون جاهزة للنفاذ وعندها فقط يقوم المعالج بالنفاذ إلى بوابة جهاز العبور ضمن إجرائية الاستجابة للمقاطعة.

5. اشرح عملية العبور بالمصافحة.

في هذا النمط من العمليات، يضع الجهاز المحيطي المعطيات على مخارجه ويرسل إشارة قدح $\overline{STB}$ للمعالج الصغري الذي يكشف الإشارة (إما بالتقصي أو المقاطعة) ويقرأ المعطيات. ثم يقوم المعالج بإرسال إشارة إشعار ACK إلى الجهاز المحيطي لإعلامه بإمكانية إرسال الكلمة التالية.

6. اشرح عملية العبور بالمصافحة المزدوجة.

في هذا النمط من المصافحة، يستعلم الجهاز المرسل عن جاهزية المستقبل بوضع خط القدح على المستوى المنخفض. فإذا كان المستقبل جاهزاً، يقوم بإعلام المرسل عن جاهزيته بوضع خط الإشعار على المستوى المنطقي العالي. عندها يضع المرسل المعطيات الجديدة على خطوط المعطيات ويجعل المستوى المنطقي على خط القدح عالياً ليُعلم المستقبل بأن المعطيات الحديثة متوفرة، يقوم المستقبل بقراءة المعطيات ثم يضع منطقاً

منخفضاً على خط الإشعار ACK لإعلام المرسل بأن المعطيات قد تمت قراءتها وأن المستقبل ينتظر معطيات جديدة.

7. ما هي طريقة العبور المحجوز من الذاكرة (memory-mapped I/O)؟

هي عملية العبور إلى بوابات الدخل والخرج باستخدام نفس تعليمات النفاذ إلى الذاكرة، وتكون عناوين هذه البوابات جزء من فضاء عنونة الذاكرة.

أسئلة خيارات متعددة

1. ما هو عدد بتات عنونة بوابات الدخل والخرج في المعالج 8086؟

A. 8

B. 16

C. 20

D. 32

2. تتم عمليات الدخل والخرج في المعالج 8086 على معطيات بعرض:

A. 8 بت

B. 16 بت

C. 8 أو 16 بت

D. 32 بت

3. ما هي طريقة العبور المناسبة من أجل توفير وقت المعالج في عملية النفاذ لبوابات الدخل والخرج؟

A. طريقة العبور بالتقصي.

B. طريقة العبور بالمقاطعة.

C. طريقة العبور بالمصافحة.

D. طريقة العبور بالمصافحة المزدوجة.

4. ما هي طريقة العبور الأنسب من حيث الوثوقية والسرعة لنقل المعطيات بين المعالج والطرفيات الخارجية من أجل معدلات النقل السريعة؟

A. طريقة العبور بالتقصي

B. طريقة العبور بالمقاطعة.

C. طريقة العبور بالمصافحة.

D. طريقة العبور بالمصافحة المزدوجة.

5. ما هي وظيفة الدارة التكاملية 8255A ؟

A. دارة عبور مبرمجة.

B. دارة تحكم بالمقاطعات.

C. دارة مؤقت.

D. دارة عزل معطيات.

6. ما هي وظيفة الدارة التكاملية 8259A ؟

A. دارة عبور مبرمجة.

B. دارة تحكم بالمقاطعات.

C. دارة مؤقت.

D. دارة عزل معطيات.

7. ما هي وظيفة الدارة التكاملية 8254A ؟

A. دارة عبور مبرمجة.

B. دارة تحكم بالمقاطعات.

C. دارة مؤقت.

D. دارة عزل معطيات.

8. ما هو عدد الإشارات الخارجية التي يمكن تقاطع المعالج 8086؟

A. 1

B. 2

C. 3

D. 256

رقم التمرين	الإجابة الصحيحة
1	B
2	B
3	B
4	C
5	A
6	B
7	C
8	D

الفصل الرابع: مدخل إلى لغة التجميع

الكلمات المفتاحية:

لغة التجميع، أنماط العنونة، قطاعات الذاكرة، قوائم المهمات التتابعية، المخطط التدفقي، الصيغة القياسية للبرنامج.

ملخص:

يهدف هذا الفصل إلى عرض أساسيات البرمجة بلغة التجميع Assembly. نعرض أولاً أنماط العنونة المختلفة، التي تسمح للمعالج الوصول إلى المعطيات في الذاكرة، ثم نتطرق بعد ذلك إلى المراحل الأساسية لتطوير برنامج بلغة التجميع، بدءاً من تعريف المسألة وحتى كتابة البرنامج.

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

- كيفية إدارة الذاكرة ومبدأ التجزئة إلى قطاعات
- أنماط العنونة المختلفة
- التعرف على مراحل كتابة برنامج
- التعرف على مفهوم قوائم المهمات التتابعية والمخططات الانسيابية
- التعرف على الصيغة القياسية لبرنامج مكتوب بلغة التجميع

يهدف هذا الفصل إلى عرض أساسيات البرمجة بلغة التجميع Assembly.
نعرض أولاً أنماط العنونة المختلفة، التي تسمح للمعالج الوصول إلى المعطيات في الذاكرة، ثم نتطرق بعد ذلك إلى المراحل الأساسية لتطوير برنامج بلغة التجميع، بدءاً من تعريف المسألة وحتى كتابة البرنامج.

1. لغات البرمجة:

يقوم أي معالج صغري بتنفيذ برنامج مخزن في الذاكرة، ولذا فإن المعالج ينفذ برنامجاً مكتوباً باللغة الثنائية فقط. توجد ثلاث مستويات لكتابة برنامج لمعالج صغري، نعرضها باختصار تباعاً.

1.1. لغة الآلة:

يمكن كتابة البرامج ببساطة كمتتالية أرقام ثنائية، تمثل تعليمات المعالج الواجب تنفيذها.

عنوان الذاكرة	المحتوى الست عشري	المحتوى الثنائي	العملية
00100h	E4h	11100100	INPUT From
00101h	05h	00000101	Port 05h
00102h	04h	00000100	ADD
00103h	07h	00000111	07h
001004h	E6h	11100110	OUTPUT To
001005h	02h	00000010	Port 02

تسمى الصيغة الثنائية للبرنامج، بلغة الآلة Machine Language لأنها الشكل الوحيد الذي تفهمه الآلة. ولكن من الصعب كتابة برنامج بهذه الصيغة لصعوبة حفظ تراميز جميع التعليمات الثنائية.

2.1. لغة التجميع:

لجعل البرمجة أيسر، يكتب المبرمج بلغة المجمع، ثم يقومون بترجمة البرامج إلى لغة الآلة بحيث يمكن شحنها في الذاكرة وتنفيذها. تستخدم لغة التجميع مصطلحات "تذكرية" (mnemonics) ذات ثلاث و أربعة أحرف لتمثيل كل نوع التعليمات. والمصطلح هو مجرد وسيلة لمساعدة المبرمج في التذكر، والحروف المستخدمة فيها تدل عادة على كلمة معينة في اللغة الانكليزية وتشير إلى التعليمة التي تقوم بها التعليمة, فمثلاً يُرمز إلى عملية الطرح بالرمز SUB (المنسق من فعل Substract)، وتدل تعليمة OR على العملية المنطقية الموافقة، أما تعليمة نقل المعلومة من مكان إلى آخر فهي تعليمة MOV وهكذا.

يتم تحويل البرنامج المكتوب بلغة التجميع إلى لغة الآلة باستخدام برنامج على الحاسب يسمى مجمع (Assembler). وغالباً ما يترافق مع أدوات مساعدة لتطوير البرامج: مثل المحاكي (simulator) والمنقح (Debugger). ومن الجدير بالذكر أن لكل معالج صغري لغة مجمع خاصة به، لأن هذه اللغة تعكس مباشرة بنيته الداخلية وإمكانات البرمجة المتاحة لهذا المعالج.

3.1. اللغات العالية المستوى:

يمكن كتابة برنامج لمعالج صغري باستخدام إحدى اللغات التي تسمى اللغات العالية المستوى (High-Level Languages)، مثل Basic أو Pascal أو C وغيرها. ولتحويل برنامج مكتوب بلغة عالية المستوى إلى برنامج بلغة الآلة، ينبغي استخدام أداة تسمى المترجم (Compiler). وعند البرمجة بلغة عالية المستوى، يكون الزمن اللازم لإنجاز البرنامج أقصر منه بلغة التجميع، لأن اللغة العالية المستوى تستخدم لبنات بنيوية أغنى لكتابة البرنامج ولكن تنفيذها يكون عادة أطول من الكتابة بلغة التجميع، لأن المترجم المستخدم للتحويل لا يكون أمثل.

في العادة يتم البدء بكتابة البرامج بلغة عالية المستوى ومن ثم يتم إعادة كتابة أجزاء منه فقط وهي الأجزاء التي تتطلب وقت تنفيذ أصغري، بلغة التجميع.

2. مبدأ التجزئة:

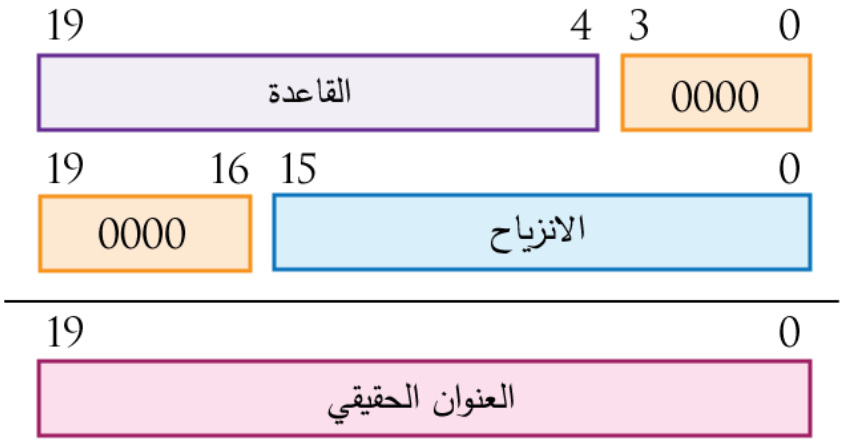
قبل التطرق إلى تقنيات البرمجة بلغة المجمع، لا بد من التذكير بأن نفاذ المعالج إلى الذاكرة يتم باستخدام مفهوم القطاعات وذلك لأن مسرى العناوين يتكون من 20 بت بينما السجلات الداخلية في المعالج تحتوي على 16 بت فقط.

لتوليد العنوان الحقيقي (Physical Address) على 20 بت يقوم المعالج بجمع قيمة قاعدية (Base Address)، مرمزة على 16 بت ومخزنة في مؤشر من مؤشراته الداخلية، إلى قيمة انزياح (Offset) مرمزة أيضاً على 16 بت ومخزنة في سجل من سجلات المعالج الداخلية، ولكن بعد أن يتم إزاحة عنوان القاعدة أربع بتات لليسار.

مثال: نريد الوصول إلى المعلومة المخزنة في العنوان الحقيقي 2437Ah في قطاع المعطيات، بفرض أن قيمة السجل DS=2000h، فما هو الانزياح الواجب جمعه للوصول إلى تلك المعلومة.

لتوليد العنوان الحقيقي ينبغي أولاً إزاحة السجل DS أربعة بتات إلى اليسار، ثم جمع محتواه إلى الانزياح المناسب. وبالتالي نحصل بالطرح على قيمة الانزياح المرمز على 16 بت، أي:

.Offset=2437Ah – 20000h = 0437Ah



آلية حساب العنوان الحقيقي

هناك طرق عديدة لتقديم هذا الانزياح لوحدة واجهة المسرى، فقد تحتوي التعليمة ذاتها على قيمة الانزياح، أو قد يخزن الانزياح في سجل ما في المعالج، أو في موقع من الذاكرة. تسمى هذه الطرائق المختلفة أنماط العنونة (Addressing Modes).

3. أنماط العنونة:

تعبر أنماط العنونة عن الطرق المختلفة لتقديم قيمة الانزياح لحساب العنوان الحقيقي.

1. العنونة الفورية (Immediate Addressing).
2. العنونة بالسجل (Register Addressing).
3. نمط العنونة المباشرة (Direct Addressing).
4. العنونة غير المباشرة بالسجل (Register Indirect Addressing).
5. العنونة بالدليل (Index Addressing).

1. العنونة الفورية (Immediate Addressing): نقل قيمة ثابتة إلى سجل

```
MOV  CX, 437Bh
MOV  CL,48h
```

2. العنونة بالسجل (Register Addressing): نقل قيمة سجل إلى سجل

```
MOV  CX, AX
MOV  CL, AL
```

3. العنونة المباشرة (Direct Addressing): نقل قيمة ذاكرة ذات انزياح معطى إلى سجل

```
MOV  CL, [437Ah]
MOV  BX, [437Ah]
```

4. العنونة غير المباشرة بالسجل (Register Indirect Addressing)

وهي العمليات التي يعطى فيها قيمة الانزياح ضمن سجل ولها عدة أشكال:

- العنونة غير المباشرة بالسجل بدون انزياح

```
MOV [BX], AX
```

- العنونة غير المباشرة بالسجل مع انزياح ثابت

```
MOV SI,[BP+4]
```

- العنونة غير المباشرة باستخدام سجل الدليل

```
OR [BX+DI], 0100h
```

- العنونة غير المباشرة باستخدام سجل الدليل وانزياح ثابت

```
AND DL, [BX+SI+02h]
```

5. نمط العنونة بالدليل (Index Addressing)

هذا النمط مشابه لنمط العنونة غير المباشر بالسجل، ولكن الاختلاف يكمن في السجل المستخدم. فهنا لا نستخدم إلا السجلين SI و DI، أما في ذاك النمط، فيمكن استخدام سجلات المعالج AX, BX, CX, DX للعنونة

ADD AX, [SI]

1.3. العنونة الفورية:

في نمط العنونة الفورية (Immediate Addressing)، تحتوي تعليمة المعالج على القيمة "الفورية" الواجب استخدامها أثناء التنفيذ.

مثال 1:

MOV CX, 437Bh

نقل القيمة 437Bh إلى السجل CX.

ملاحظة: في تعليمة MOV، يدل الحد الأول دوماً على الوجهة، والحد الثاني على المصدر.

مثال 2:

MOV CL,48h

نقل القيمة 48h إلى السجل CL (المرمز على 8 بت).

يمكن إذاً نقل قيم على 8 بت أو 16 بت إلى أحد سجلات المعالج أو أحد مواقع الذاكرة نقلاً فورياً.

2.3. العنونة بالسجل:

في نمط العنونة بالسجل (Register Addressing)، تحتاج تعليمة المعالج إلى القيمة المخزنة في أحد سجلاته الداخلية.

مثال 1:

MOV CX, AX

عند تنفيذ هذه التعليمة، يتم نسخ القيمة المخزنة في السجل AX والمرمزة على 16 بت إلى السجل CX.

مثال 2:

MOV CL, AL

تنقل هذه التعليمة محتوى السجل AL ذا البتات الثمانية إلى السجل CL.

3.3. العنونة المباشرة:

في نمط العنونة المباشرة (Direct Addressing)، فحد التعليمة هو انزياح عنوان موقع في الذاكرة، يحتوي على القيمة الواجب استخدامها.

مثال 1:

```
MOV CL, [437Ah]
```

عند تنفيذ هذه التعليمة، يتم نقل القيمة الموجودة في موقع الذاكرة الموجود في قطاع المعطيات بانزياح قدره 437Ah عن بداية القطاع.

مثال 2:

```
MOV BX, [437Ah]
```

في هذا المثال يتم شحن السجل BX القيمة الموجودة في موقعين متتابعين من الذاكرة كل منهما على 8 بت بحيث يكون انزياح الموقع الأول هو 437Ah وهو الذي سينقل محتواه إلى السجل BL والموقع التالي إلى السجل BH.

ملاحظة: عند تنفيذ تعليمة نقل معطيات لقيم على 16 بت في النمط المباشر يجب التأكد من أن الانزياح هو قيمة زوجية.

4.3. العنونة الغير مباشرة بالسجل:

فى نمط العنونة غير المباشرة بالسجل (Register Indirect Addressing)، يعطى في التعليمة السجل الذي يحتوي على انزياح عنوان المعلومة في الذاكرة. وبكلمة أخرى، فإن حد التعليمة المرمزة في النمط غير المباشر سجل محتواه انزياح عنوان القيمة المطلوبة. ويوجد لهذا النمط من العنونة أربعة أشكال مختلفة:

1. العنونة غير المباشرة بالسجل بدون انزياح.
2. العنونة غير المباشرة بالسجل مع انزياح ثابت.
3. العنونة غير المباشرة باستخدام سجل الدليل.
4. العنونة غير المباشرة باستخدام سجل الدليل وانزياح ثابت.

العنونة غير المباشرة بالسجل بدون انزياح:

وهو الشكل الأبسط، إذ يمثل محتوى السجل انزياح عنوان القيمة في قطاع المعطيات.

مثال:

```
MOV [BX], AX
```

تقوم هذه التعليمة بنقل محتوى السجل AX إلى موقعين متتاليين في الذاكرة. بانزياح مقداره BX عن بداية قطع المعطيات المحدد بالسجل.

العنونة غير المباشرة بالسجل مع انزياح ثابت:

في هذا الصنف من العنونة تضاف قيمة ثابتة إلى محتوى السجل المذكور في التعليمة للحصول على انزياح عنوان المعلومة في الذاكرة.

مثال:

MOV SI,[BP+4]

تنقل هذه التعليمة إلى السجل SI، القيمة (على 16 بت) ذات الانزياح المساوي لناتج جمع محتوى السجل BP وقيمة 4.

العنونة غير المباشرة باستخدام سجل الدليل:

بدلاً من إضافة قيمة ثابتة إلى محتوى السجل المستخدم في العنونة غير المباشرة، يمكن إضافة محتوى احد سجلات الدليل (وهي SI, DI, BI). فيصبح انزياح عنوان القيمة المطلوبة مساوياً لجمع محتوى السجلين معاً: سجل الدليل وسجل العنونة.

مثال:

OR [BX+DI], 0100h

تجري هذه التعليمة العملية المنطقية OR بين القيمة 0100h وموقع في الذاكرة يُعطى انزياحه بجمع محتوى السجلين DI وBX معاً.

العنونة غير المباشرة باستخدام سجل الدليل وانزياح ثابت:

يمكن أيضاً إضافة قيمة ثابتة إلى ناتج جمع محتوى سجل الدليل وسجل العنونة لتكوين انزياح عنوان القيمة المطلوبة.

مثال:

AND DL, [BX+SI+02h]

تجري هذه التعليمة، وهي تتعامل مع قيم مرمزة على 8 بت، عملية AND بين السجل DL وموقع محدد في الذاكرة بانزياح يساوي جمع BX وSI والقيمة الثابتة 02h.

5.3. العنونة بالدليل:

تعتمد التعليمات التي تستخدم نمط العنونة بالدليل (Index Addressing) على أحد سجلي الدليل SI و DI لحصول على انزياح عنوان القيمة المطلوبة في الذاكرة. هذا النمط مشابه لنمط العنونة غير المباشر بالسجل، ولكن الاختلاف يكمن في السجل المستخدم. فهنا لا نستخدم إلا السجلين SI و DI، أما في ذاك النمط، فيمكن استخدام سجلات المعالج AX, BX, CX, DX للعنونة.

مثال:

```
ADD AX,[SI]
```

تقوم هذه التعليمة بجمع محتوى AX مع كلمة على 16 بت في الموقع ذو الانزياح SI في قطاع المعطيات ويتم تخزين الناتج في السجل AX نفسه.

4. مراحل تطوير برنامج بلغة التجميع:

عندما يطلب كتابة برنامج بلغة التجميع، ينبغي إتباع المراحل التالية لضمان بناء برنامج صحيح البنية:

1. تعريف المسألة: بداية صياغة المسألة بكتابة المهمات Tasks التي يجب أن يقوم بها البرنامج.
2. تمثيل عمليات البرنامج: التعبير عن خطوات عمل البرنامج. وهناك عدة طرق لتمثيل خوارزمية البرنامج
   - قوائم المهمات التتابعية: قائمة بالمهمات المطلوبة وذلك بغية توضيح خوارزمية البرنامج.
   - المخططات التدفقية: هي أشكال بيانية تمثل مختلف عمليات البرنامج.
   - الترميز الكاذب (Pseudo-code): يستخدم كطريقة لصياغة البرنامج بلغة عالية المستوى تحتوي على العديد من البنى البرمجية.
3. إيجاد التعليمات المناسبة: معرفة أنواع التعليمات الممكن استخدامها مع المعالج، ثم العودة إلى لائحة تعليمات المعالج لتحديد طريقة الاستخدام السليم.
4. كتابة البرنامج: كتابة البرنامج بلغة التجميع مع تعليمات الاستهلال الضرورية في بداية البرنامج.
5. التوثيق: إضافة تعليقات وشرح ضمن البرنامج لتوضيح عمله.

وسنأتي فيما يلي على شرح كل مرحلة من هذه المراحل.

عندما يطلب كتابة برنامج بلغة التجميع، ينبغي إتباع المراحل التالية لضمان بناء برنامج صحيح البنية، قادر على أداء المهام المطلوبة.

1. تعريف المسألة.
2. تمثيل عمليات البرنامج.
3. إيجاد التعليمات المناسبة.
4. كتابة البرنامج.
5. التوثيق.

1.4. تعريف المسألة:

لا بد في بداية صياغة المسألة بكتابة المهمات Tasks التي يجب أن يقوم بها البرنامج مع تعريف المداخل والمخارج. فمثلاً، قد تصاغ مسألة برمجة على النحو الآتي:

نريد بناء برنامج يقوم بقياس درجة الحرارة من حساس مربوط مع المعالج على بوابة دخل معينة ومن ثم نريد تخزين نتائج القياس في الذاكرة بعد إضافة معامل تصحيح للقياسات بمقدار 7 درجات مئوية. أي نريد تنفيذ مايلي:

1. قراءة درجة الحرارة من حساس معين.
2. إضافة معامل تصحيح قدره 7.
3. تخزين النتيجة في الذاكرة.

ففي هذا البرنامج، هناك ثلاث مهمات مطلوبة، وهي مصوغة باللغة "الاعتيادية". أما في المسائل التي هي أكثر تعقيداً، فيحسن بنا تجزئة المهمات الصعبة إلى مهمات فرعية أسهل، وكتابة مراحل تنفيذ كل من هذه المهمات الفرعية.

2.4. تمثيل عمليات البرنامج:

تسمى سلسلة العمليات المستخدمة في حل مسألة برمجة بخوارزمية البرنامج Algorithm. ومن أهم هذه الطرق:

1. قوائم المهمات التتابعية.
2. المخططات التدفقية (Flowcharts).
3. الترميز الكاذب (Pseudo-code).

سنتعرف على هذه الطرق تباعاً.

1.2.4. قوائم المهمات التتابعية:

على نحو مشابه لما عرض في الفقرة لسابقة، يضع بعض المبرمجين قائمة بالمهمات المطلوبة، تسمى بقائمة المهمات التتابعية، وذلك بغية توضيح خوارزمية البرنامج. لنفترض على سبيل الإيضاح، أننا نريد بدلاً من أخذ عينة واحدة من حساس الحرارة، أن نأخذ عشر عينات بواقع عينة كل ساعة وتصحيح القيمة المقروءة بإضافة 7 وتخزين النتيجة في الذاكرة. يمكن صياغة مهمات هذا البرنامج كما يلي:

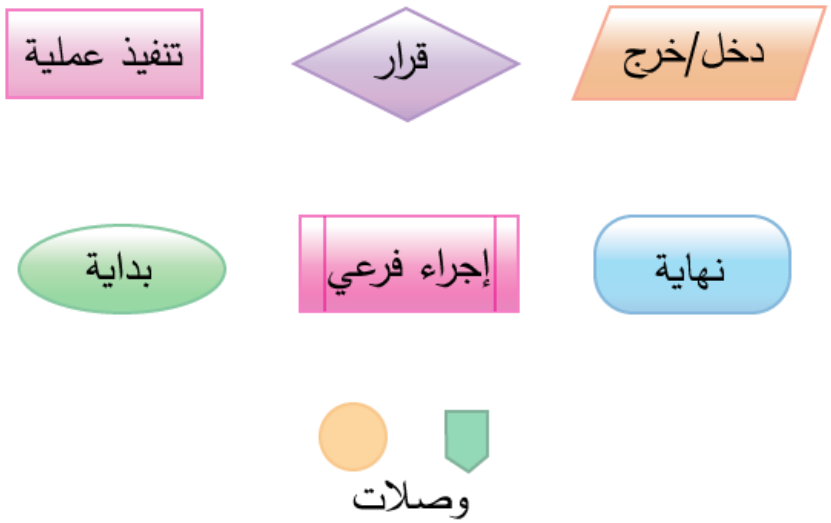
1. قراءة عينة من حساس الحرارة.
2. إضافة 7 إلى القيمة المقروءة.
3. تخزين القيمة المصححة في موقع من الذاكرة.

4. انتظار ساعة كاملة.
5. هل أصبح لدينا 10 عينات؟
6. إذا كان الجواب كلا: عد إلى المهمة الأولى.
7. إذا كان الجواب نعم: توقف عن العمل.

من الجدير بالذكر، أن كتابة قوائم المهمات جيداً يسهل أمر تحويلها إلى لغة التجميع. إذ يكفي لإجراء ذلك، تحديد بعض التفاصيل المتعلقة بالمكونات المادية، مثل عنوان معبر القراءة، ومواقع التخزين في الذاكرة.

2.2.4. المخططات التدفقية:

المخططات التدفقية (Flowcharts) هي أشكال بيانية تمثل مختلف عمليات البرنامج، فيرمز إلى كل عملية برمز بياني. ويظهر في الشكل التالي بعض الرموز المستخدمة في بناء المخططات التدفقية.



بعض الرموز المستخدمة في المخططات التدفقية

يمكن رسم المخطط التدفقي للمثال السابق كما في الشكل التالي:

المخطط التدفقي للمثال المدروس

يدل الشكل الإهليلجي على بداية البرنامج أو نهايته. كما يُرمز إلى عمليات القراءة أو الكتابة من المداخل أو المخارج بمتوازي أضلاع. أما العمليات الحسابية والمنطقية فيرمز إليها برمز مستطيل. يرمز إلى البرامج الفرعية بمستطيل مخطط ويحتوي على مجموعة من التعليمات التي تقوم بوظيفة محددة. أما عمليات الاختبار والقرار فيرمز إليها بمعين يتفرع عنه عدة وصلات حسب نتيجة الاختبار. وهناك أيضاً شكلان يدلان على الوصلة. فإذا اضطررنا إلى كتابة مخطط تدفقي طويل ووصلنا إلى نهاية الصفحة، فيمكننا إنهاء المخطط على الصفحة برمز الوصلة الخماسية الأضلاع، ومتابعة المخطط على الصفحة التالية بالرمز ذاته. أما في حالة متابعة المخطط على الصفحة ذاتها ولكن في عمود مختلف، فيمكن استخدام رمز الدائرة.لإنهاء المخطط في عمود من الصفحة، ثم متابعته في عمود آخر من الصفحة.

إن المخططات التدفقية وسيلة بيانية لعرض تسلسل لعرض تسلسل البرنامج، إلا أنها تعاني بعض المساوئ. إذ لا يمكن كتابة إلا القدر اليسير من المعلومات داخل كل رمز. وقد تصبح المخططات معقدة إذا كان البرنامج طويلاً.

3.2.4. الترميز الكاذب:

الترميز الكاذب (Pseudo-code): هو طريقة لصياغة البرنامج بلغة عالية المستوى عند تصميم البرنامج. ولكن لا تستخدم فعلياً لكتابة البرامج وترجمتها.

يتم باستخدام البنى البرمجية التالية:

- "إذا – افعل – وإلا": تستخدم للاختبار وتنفيذ مهمة معينة حسب نتيجة الاختبار مثل التعليمة if-else
- "كرر – إلى أن": تستخدم لتكرر عملية حتى يتحقق شرط معين مثل تعليمة Repeat-until
- "من أجل – افعل": وتستخدم أيضاً للتكرار باستخدام عداد مثل تعليمة For
- "مادام – افعل": تستخدم للتكرار حسب نتيجة اختبار مثل تعليمة While
- "في حال – افعل": تستخدم للانتقاء المتعدد مثل تعليمة Case

3.4. إيجاد التعليمات المناسبة:

بعد وضع البنية العامة للبرنامج، تأتي مرحلة تحديد التعليمات المطلوبة لأداء كل جزء من البرنامج. وكما هو الحال عند تعلم لغة ما، فإن إتقان المعجم كاملاً ل يؤدي إلى الطلاقة في اللغة. فمن الأفضل حفظ بعض الكلمات ووضعها في جمل مفيدة، ثم الانتقال إلى تعلم المزيد من الكلمات، واستخدامها في صيغ أكثر تعقيداً. وكذلك الأمر فيما يخص لغة المجمع. فمن المفيد معرفة أنواع التعليمات الممكن استخدامها، ثم العودة إلى لائحة تعليمات المعالج لتحديد طريقة الاستخدام السليم.

لنبحث على سبيل المثال عن التعليمات المناسبة لصياغة برنامجنا البسيط (قراءة الحساس، والتصحيح، والتخزين). نجد من استعراض تعليمات الدخل والخرج أن تعليمة IN تسمح بقراءة قيمة من بوابة دخل، ويمكن استخدام تعليمة ADD لجمع معامل التصحيح 7 إلى القيمة المقروءة، كما يمكن استخدام التعليمة MOV لنسخ ناتج الجمع إلى مكان ما في الذاكرة.

4.4. كتابة البرنامج:

1.4.4. تعليمات الاستهلال:

بعد إيجاد التعليمات اللازمة لأداء المهمات المطلوبة، لا بد من البحث عن بعض التعليمات الضرورية قبل كتابة البرنامج. إن الهدف من هذا التعليمات الإضافية وضع قيمة ابتدائية في الأجزاء المختلفة من النظام، مثل سجلات القطاعات، والرايات، والطرفيات القابلة للبرمجة. تسمى هذه التعليمات بتعليمات الاستهلال (Initialisation).

وعلى سبيل المثال يجب أولاً تقسيم الذاكرة إلى قطاعات. وبعد تحديد عنوان بداية كل قطاع نقوم بوضع القيم المناسبة في سجلات القطاعات.

من جهة أخرى، تضم معظم النظم الحاسوبية طرفيات متعددة قابلة للبرمجة، مثل المؤقتات والتحكمات. لذا ينبغي إدراج تعليمات استهلال هذه الطرفيات في بداية البرنامج لتحديد أنماط عملها. وإضافة إلى ذلك، ينبغي استعمال تعليمات الاستهلال لوضع رايات التحكم في المعالج مثل راية تأهيل المقاطعات عند استخدام المقاطعات.

أنسب طريقة لإعداد الاستهلال هي وضع قائمة بكافة السجلات والطرفيات المبرمجة، والرايات التي يضمها النظام الحاسوبي المراد برمجته. بعد ذلك، يمكن إضافة تعليمات وضع القيم الابتدائية لجميع هذه السجلات.

2.4.4. الصيغة القياسية للبرنامج:

يتضمن كل سطر من البرنامج المكتوب بلغة التجميع الحقول التالية:

اللصاقة (Label): وهي اسم يرمز إلى نقطة معينة في البرنامج. وهو يغني عن استخدام العنوان الحقيقي لذلك السطر.

رمز التعليمة: ويضم هذه الحقل اسم التعليمة المراد استخدامها مثل ADD، MOV وغيرها.

الحدود أو المعاملات (Operands): تسمح هذه الحدود بتحديد القيم اللازمة لتنفيذ التعليمة.

التعليق (Comment): يسمح هذا الحقل، الذي يبدأ بعد علامة ";" بإعطاء توضيح حول عمل كل تعليمة في البرنامج بهدف جعل البرنامج مفهوماً بشكل أفضل للآخرين أو عند العودة إليه لاحقاً لتعديله.

مثال:

```
SUM: MOV  AL, 5          ; load AL value
MOV  BL,20          ; load BL value
ADD  AL, BL;AL=AL+BL
```

نقوم هنا بشحن السجل AL بالقيمة 5، والسجل BL بالقيمة 20، ثم نجمع السجلين ويخزن الناتج في المعامل الأول لتعليمة الجمع AL.

ومن الجدير بالذكر بأن ترجمة كل تعليمة من تعليمات لغة التجميع إلى لغة الآلة يمكن أن ينتج عنه بايت واحد أو أكثر حسب التعليمة وحدود التعليمة. كما أن زمن تنفيذ التعليمات في المعالج يتفاوت من تعليمة لأخرى.

يمكن أن نحصل على عد بايتات ترميز التعليمة في لغة الآلة وزمن تنفيذها مقدراً بعدد أدوار ساعة المهتز التي تشغل المعالج من النشرة الفنية للمعالج التي يزودها المصنع. سوف نبين في الفصل القادم فائدة هذه المعلومات عند برمجة حلقات التأخير الزمني. فمثلاً إن تعليمة NOP التي لا تقوم بأي عملية هي مرمزة بلغة الآلة على بايت واحد (بقيمة 90h) ويستغرق تنفيذها في المعالج 3 أدوار ساعة.

5.4. التوثيق:

يشمل التوثيق تحديد اسم البرنامج واسم الملف المحتوى عليه وتاريخ كتابته ورقم الإصدار ووصف موجز له، وتحديد الإجراءات المستخدمة والبوابات الخارجية. توضع هذه المعلومات في بداية البرنامج. ويشمل التوثيق أيضاً وضع التعليقات المناسبة للأسطر التي تحتاج إلى توضيح.

إن تأكيد ضرورة التوثيق أمر هام للغاية. إذ تدل التجارب أن برنامجاً صغيرً كتب منذ بضعة أشهر بلا تعليقات أو شرح، يكون صعب الفهم اليوم ولو عاد إليه المبرمج ذاته!

أسئلة الفصل الرابع

1. نريد الوصول إلى المعلومة المخزنة في العنوان الحقيقي 2437Ah في قطاع المعطيات، بفرض أن قيمة السجل DS=2000h، فما هو الانزياح الواجب جمعه للوصول إلى تلك المعلومة ؟

Offset=2437Ah – 20000h = 0437Ah.

2. لماذا نستخدم برنامج المجمع Assembler ؟

يستخدم برنامج المجمع لتحويل البرنامج المكتوب بلغة المجمع إلى لغة الآلة.

3. ما العمل الذي تقوم به التعليمة التالية MOV BX, [437Ah]؟

شحن السجل BX بالقيمة الموجودة في موقعين متتابعين من الذاكرة كل منهما على 8 بت بحيث يكون انزياح الموقع الأول هو 437Ah وهو الذي سينقل محتواه إلى السجل BL والموقع التالي إلى السجل BH.

4. بفرض أن AX=1200h و BX=4000h، و DS=1000h، ما هو عمل التعليمة التالية MOV [BX], AX؟

نقل محتوى السجل AX أي القيمة 1200h إلى موقعي الذاكرة 14000h و 14001h.

5. هناك 5 مراحل تطوير برنامج للمعالج. فما هي هذه المراحل؟

1. تعريف المسألة.
2. تمثيل عمليات البرنامج.
3. إيجاد التعليمات المناسبة.
4. كتابة البرنامج.
5. التوثيق.

6. ما هو المخطط التدفقي (Flowcharts) للبرنامج؟

المخطط التدفقي هو أشكال بيانية تمثل مختلف عمليات البرنامج بهدف توضيح خوارزمية البرنامج، حيث يرمز إلى كل عملية برمز بياني.

7. نريد بناء برنامج يقوم بقياس درجة الحرارة من حساس مربوط مع المعالج على بوابة دخل معينة ومن ثم نريد تخزين نتائج القياس في الذاكرة بعد إضافة معامل تصحيح للقياسات بمقدار 7 درجات مئوية. اكتب المهمات التتابعية لهذا البرنامج.

1. قائمة المهمات التتابعية لهذا البرنامج هي:

2. قراءة درجة الحرارة من الحساس.

3. إضافة معامل تصحيح قدره 7.

4. تخزين النتيجة في الذاكرة.

أسئلة خيارات متعددة

1. ماهي اللغة التي يفهمها المعالج ويقوم بتنفيذها؟

A. لغة الألة

B. لغة التجميع

C. لغة C

D. جميع ما سبق.

2. ما هي العملية التي يدل عليها الرمز التالي في المخططات التدفقية : 

A. دخل/خرج

B. قرار.

C. إجراء فرعي.

D. تنفيذ عملية.

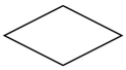
3. ما هي العملية التي يدل عليها الرمز التالي في المخططات التدفقية : 

A. دخل/خرج

B. وصلة مخطط في نفس الصفحة.

C. وصلة مخطط في صفحة أخرى.

D. تنفيذ عملية.

4. ما هي العملية التي يدل عليها الرمز التالي في المخططات التدفقية : 

A. قرار.

B. وصلة مخطط في نفس الصفحة.

C. وصلة مخطط في صفحة أخرى.

D. بداية البرنامج أو نهايته.

5. يستخدم مصطلح الترميز الكاذب (Pseudo-code) من أجل:

A. وصف تعليمة لم تتم صياغتها بشكل صحيح وفق قواعد لغة التجميع.

B. مجموعة تعليمات مكتوبة بلغة الآلة.

C. وصف لخوارزمية البرنامج بلغة عالية المستوى.

D. إحدى اللغات العالية المستوى المستخدمة في كتابة البرامج يتم نقلها إلى لغة الآلة باستخدام المترجم.

6. إن الطريقة الأنسب لتمثيل البرامج الطويلة والمعقدة هي:

A. استخدام لغة عالية المستوى مثل لغة C.

B. استخدام لغة الترميز الكاذب.

C. استخدام المخططات التدفقية.

D. استخدام المهمات التتابعية.

7. تقوم التعليمة التالية ADD AX,[SI] بما يلي:

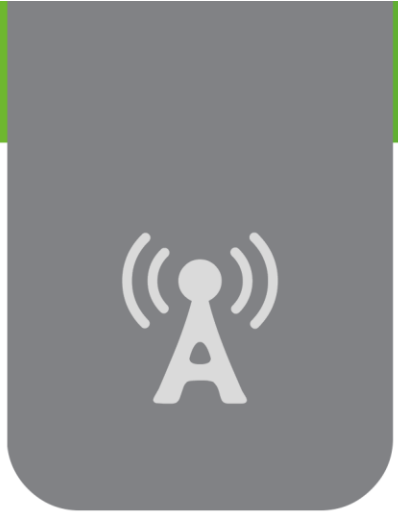
A. جمع السجلين AX و SI.

B. جمع السجل AX مع البايت ذو الدلالة الأدنى من السجل SI.

C. جمع السجل AX مع محتوى الكلمة في موقع الذاكرة التي يشير إليها السجل SI.

D. جمع السجل AX مع محتوى الكلمة في موقع الذاكرة ذو الانزياح SI.

رقم التمرين	الإجابة الصحيحة
1	A
2	B
3	B
4	A
5	C
6	B
7	D



الفصل الخامس: تقنيات البرمجة بلغة التجميع

الكلمات المفتاحية:

التعليمات الحسابية والمنطقية، عمليات القفز، الحلقات.

ملخص:

يهدف هذا الفصل إلى منح الطالب القدرة على برمجة المعالج 8086، وذلك بتعريف الطالب بمنهجية كتابة البرنامج انطلاقاً من تعريف المسألة مروراً بالطرق المستخدمة لتمثيل خوارزمية البرنامج وانتهاءً بتقنيات البرمجة الأساسية بلغة التجميع. نعرض تباعاً العمليات الحسابية والمنطقية وعمليات القفز والحلقات والتنفيذ المشروط مع توضيح هذه التقنيات بالأمثلة المناسبة.

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

- التعليمات الحسابية والمنطقية
- تعليمات القفز بأنواعه
- كيفية برمجة الحلقات
- كيفية برمجة بعض البنى الأساسية بلغة التجميع

يهدف هذا الفصل إلى منح الطالب القدرة على برمجة المعالج 8086، وذلك بتعريف الطالب بمنهجية كتابة البرنامج انطلاقاً من تعريف المسألة مروراً بالطرق المستخدمة لتمثيل خوارزمية البرنامج وانتهاءً بتقنيات البرمجة الأساسية بلغة التجميع. نعرض تباعاً العمليات الحسابية والمنطقية وعمليات القفز والحلقات والتنفيذ المشروط مع توضيح هذه التقنيات بالأمثلة المناسبة.

1. مقدمة:

إن أفضل طريقة لتعلم تقنيات البرمجة هي دراسة هذه التقنيات من خلال أمثلة بسيطة لتوضيح كيفية ترجمة خطوات البرنامج التي تمت صياغتها بلغة عالية المستوى إلى لغة التجميع. في البداية سوف نأخذ مثالاً بسيطاً عن كيفية كتابة برنامج بسيط بلغة التجميع وفق الخطوات التي تم عرضها في الفصل السابق ومن ثم سوف نقوم بشرح بعض التقنيات المستخدمة في البرمجة بلغة التجميع.

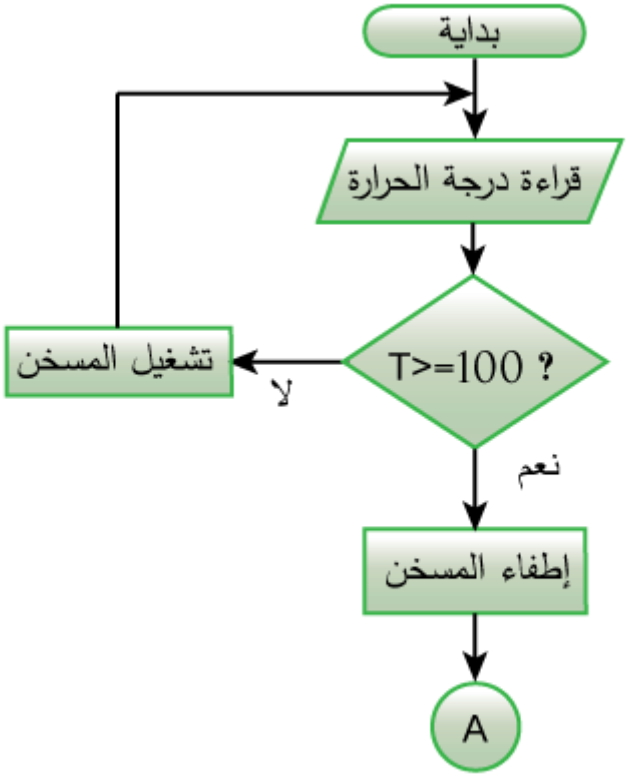
2. برنامج التحكم بدرجة حرارة محلول:

1.2.تعريف المسألة:

لنفترض أننا، في مسألة تحكم في عملية كيميائية، نريد أن نجعل درجة حرارة المحلول مساوية 100 °C قبل البدء بالعمل. فإذا كنت درجة حرارة المحلول أقل من 100 °C، ينبغي تشغيل المسخن، وذلك حتى تصبح درجة حرارة المحلول أكبر أو تساوي 100 °C، وعندها نوقف المسخن وننتقل إلى خطوة أخرى.

2.2. خوارزمية البرنامج:

بعد تعريف المسألة، نقوم بوضع خوارزمية البرنامج. وهنا سوف نستخدم طريقة المخطط التدفقي لوصف هذه الخوارزمية. يبين الشكل التالي المخطط التدفقي لبرنامج التحكم بدرجة حرارة محلول.



المخطط التدفقي لخوارزمية التحكم بدرجة حرارة محلول

في البداية نقوم بقراءة درجة الحرارة، ثم نقارن القيمة المقروءة مع 100. فإذا كانت درجة الحرارة أقل من100 °C سوف نقوم بتشغيل المسخن والعودة إلى البداية وقراءة درجة الحرارة من جديد. أما إذا تحقق الشرط فهذا يعني أننا وصلنا إلى درجة الحرارة المناسبة لذلك سيتم إطفاء المسخن ومتابعة العمل.

3.2. البحث عن التعليمات:

نعلم أننا نستطيع استخدام تعليمتي IN وOUT للقراءة والكتابة من بوابة خارجية. أما للمقارنة فسيتم استخدام تعليمة المقارنة CMP (Compare) وسنرى كيفية استخدامها في شرح البرنامج.

4.2. كتابة البرنامج:

لكتابة البرنامج بلغة التجميع نفترض أن حساس الحرارة موصول إلى البوابة ذي العنوان FFF8h، وأن المسخن موصول إلى البوابة FFFAh، ونفترض أن تشغيل المسخن يقتضي كتابة 1 إلى هذه البوابة.
يكتب البرنامج كما يلي:

```
Temp_In:    MOV     DX,0FFF8h       ; point to input port
            IN      AL,DX           ; read temperature
```

```
                CMP     AL,100          ; compare with 100 °C
                JAE     Heater_off      ; if above or equal go to
                MOV     AL,01h          ; load code to turn heater on
                MOV     DX,FFFAh        ; point to output port
                OUT     DX,AL           ; turn heater on
                JMP     Temp_In         ; go to Temp_In
Heater_off:     MOV     AL,0            ; load code to turn heater off
                MOV     DX,FFFAh        ; point to output port
                OUT     DX,AL           ; turn heater off
```

في البداية يجب قراءة درجة الحرارة من البوابة ذي العنوان FFF8h ونفترض أن قيمة درجة الحرارة ممثلة على 8 بتات. لذلك نقوم باستخدام التعليمة IN بعد شحن عنوان البوابة الممثل على 16 بت في السجل DX. نخزن النتيجة في السجل AL.

ثم تأتي مرحلة مقارنة درجة الحرارة المقروءة في السجل AL مع 100 °C وذلك عن طريق تعليمة CMP. حيث تقوم هذه التعليمة بطرح 100 من AL دون أن تخزن نتيجة الطرح ولكن تضع في سجل الحالة قيم الرايات الستة (الحمل، الصفر، الزوجية، الفائض، الإشارة، الحمل المساعد) بالقيم المناسبة حسب نتيجة الطرح. في الجدول التالي نجد قيم رايات الحمل والصفر والإشارة حسب نتيجة المقارنة:

	CF	ZF	SF
AL<100	1	0	1
AL=100	0	1	0
AL>100	0	0	0

فإذا كانت درجة الحرارة أكبر أو تساوي 100 °C، يجب الذهاب إلى مرحلة إطفاء المسخن. من الجدول، يمكن أن نجد أن قيمة راية الحمل تكون مساوية للصفر عندما يتحقق هذا الشرط. تفحص التعليمة JAE (Jump Above or Equal) قيمة هذه الراية وتقفز عند تحقق الشرط إلى اللصاقة Heater_off.

إذا لم يتحقق الشرط فإن البرنامج يستمر ويقوم بتنفيذ التعليمة التالية وهنا علينا تشغيل المسخن ثم الذهاب إلى بداية البرنامج لقراءة درجة الحرارة من جديد.

تتم عملية تشغيل المسخن من خلال كتابة القيمة 01h في بوابة الخرج ذي العنوان FFFAh باستخدام التعليمة OUT بعد شحن عنوان بوابة الخرج في السجل DX. ثم نقفز إلى بداية البرنامج (اللصاقة Temp_In) باستخدام تعليمة القفز JMP.

أما عندما يتحقق الشرط فإن علينا إطفاء المسخن ومتابعة المراحل الأخرى من البرنامج انطلاقا من نقطة الوصل A. ويتم ذلك من خلال كتابة القيمة 00h في بوابة الخرج.

نجد في هذا المثال أننا قمنا بعمليات دخل وخرج باستخدام العنونة غير المباشر بالسجل وذلك عن طريق التعليمتين IN وOUT. كما قمنا باستخدام العملية الحسابية CMP للمقارنة. بالإضافة إلى عملية قفز مشروط من خلال التعليمة JAE وعملية قفز غير مشروط JMP.

سوف نقوم أولاً بالاطلاع على التعليمات الحسابية والمنطقية التي يدعمها المعالج ثم ننتقل إلى عمليات القفز بأنواعها المختلفة.

3. التعليمات الحسابية والمنطقية:

ليس الهدف من هذه الفقرة شرح كل تعليمة من تعليمات المعالج، ولكن نورد هنا فقط قائمة بالعمليات التي يدعمها المعالج. وسوف نتعلم كيفية استخدام بعضها من خلال الأمثلة المطروحة. كما يمكن الرجوع إلى قائمة تعليمات المعالج في النشرة الفنية أو أحد المراجع من أجل فهم أفضل لكيفية استخدام أي تعليمة من التعليمات.

يدعم المعالج عمليات الجمع والطرح والضرب والقسمة على 8 بت أو على 16 بت. ويبين الجدول التالي قائمة بالعمليات الحسابية التي يدعمها المعالج.

Addition	Subtraction	Multiplicatio	Division
ADD	SUB	MUL	DIV
ADC	SBB	IMUL	IDIV
INC	DEC	AAM	AAD
AAA	NEG		CBW
DAA	CMP		CWD
	AAS		
	DAS		

كما يدعم المعالج العمليات المنطقية المعروفة بالإضافة إلى عمليات الإزاحة المنطقية والحسابية والدوران من خلال راية الحمل أو من دونها. يبين الجدول التالي قائمة بالعمليات المنطقية التي يدعمها المعالج:

Logical	Shift	Rotate
NOT	SHL	ROL
AND	SHR	ROR
OR	SAL	RCL
XOR	SAR	RCR
TEST		

إن تنفيذ التعليمات الحسابية قد يؤثر في سجل الرايات وتغير بعض الرايات وذلك حسب التعليمة المنفذة ونتيجة العملية كما رأينا من أجل تعليمة المقارنة CMP في مثال التحكم بدرجة حرارة محلول.

4. تعليمات القفز:

تستخدم تعليمات القفز لإجبار المعالج على الانتقال إلى موقع جديد في البرنامج وتنفيذ التعليمات الموجودة في هذا الموقع الجديد. ويتم ذلك عن طريق تغيير محتوى مؤشر التعليمات وأحياناً تغيير محتوى سجل قطاع البرنامج CS أيضاً.

1.4. تعليمة القفز اللامشروط:

يقصد بالقفز اللامشروط أن المعالج سيقفز حتماً، عند تنفيذ هذه التعليمة إلى المكان المحدد دون أن يفحص أي شرط. لنأخذ المثال التالي:

```
Back: ADD   AL, 03h          ;add 3 to Total        0000
      NOP                                          0002
      NOP                                          0003
      NOP                                          0004
      NOP                                          0005
      JMP   Back                                   0006
      NOP                                          0008
      NOP                                          0009
```

في هذا المثال، تكرر مجموعة التعليمات NOP عدداً لانهائياً من المرات. وتشير اللصاقة Back إلى العنوان الذي نريد العودة إليه في كل مرة. فعندما يصادف المجمع تعليمة القفز، يبحث عن السطر ذي اللصاقة Back،

ويحسب انزياح ذلك السطر عن تعليمة القفز. ونلاحظ أيضاً أن التعليمات المكتوبة بعد تعليمة القفز لن تنفذ أبداً، لأن القفز إلى السطر Back قفز غير مشروط، وعندما يدخل المعالج في هذه الحلقة (Back-JMP) فلن يستطيع الخروج منها.

لنحسب في حالة مثالنا السابق قيمة الانزياح اللازمة: عنوان السطر Back هو 0h، وعنوان تعليمة القفز هو 6h، ولكن بعد أن ينفذ المعالج تعليمة القفز، فإن مؤشر التعليمات IP يكون مساوياً 8h. فلكي يعود المؤشر إلى Back يجب إضافة انزياح قدره 8h-، ومن ثم يكتب الانزياح السالب بصيغة المتمم الثنائي على 8 بتات كما يلي [1111 1000]=F8h.

ملاحظة: ينبغي لحساب الانزياح، الأخذ في الحسبان العنوان الذي يؤشر السجل IP عليه بعد تنفيذ تعليمة JMP وهو عنوان التعليمة التالية، لا عنوان تلك التعليمة فحسب.

لتعليمة JMP خمسة أنواع مختلفة وهي:

1. القفز القريب

يؤدي القفز القريب (Near Jump) إلى جلب تعليمة من موقع في قطاع البرنامج الحالي بانزياح مرمز على 16 بت بصيغة المتمم الثنائي. أي أن قيمة الانزياح عن الموقع الحالي تقع في المجال من 32767- إلى 32768+. ويدل الانزياح الموجب على القفز الأمامي في البرنامج، أما الانزياح السالب فهو يدل على القفز الخلفي.

2. القفز القريب القصير

وهي حالة من القفز القريب، ولكن الانزياح فيها مرمز على 8 بتات أي يقع الانزياح في المجال [127 ,128-].

3. القفز القريب غير المباشر

يتم استبدال سجل مؤشر التعليمات بقيمة مخزنة في سجل ذي 16 بت أو من موقعين متتاليين من الذاكرة.

4. القفز البعيد

تسمح تعليمة القفز البعيد(Far Jump) بالقفز إلى قطاع برنامج آخر، ولذلك تقوم هذه التعليمة محتوى مؤشر التعليمات وقطاع البرنامج معاً. وتحتوي التعليمة في البايت الثاني والثالث قيمة الانزياح الجديد الذي يشحن في مؤشر التعليمات، وتحتوي في البايت الرابع والخامس على قيمة قطاع البرنامج الجديد.

5. القفز البعيد غير المباشر

في هذا النوع من القفز يتم شحن مؤشر التعليمات وقطاع البرنامج من أربعة مواقع متتالية من الذاكرة. الموقعين الأولين لشحن قيمة مؤشر البرنامج والموقعين الأخيرين لشحن قيمة سجل قطاع البرنامج.

ملاحظة: من الجدير بالذكر أنه عندما نستخدم تعليمة القفز المباشر باستخدام لصاقة مثل (JMP Label) فإن برنامج المجمع هو الذي يقوم بحساب الانزياح اللازم للوصول إلى اللصاقة المجددة (Label) وتحديد عملية القفز إذا كانت من النوع القصير أو القريب أو البعيد وترجمة ذلك بشكل مناسب إلى لغة الآلة.

2.4. تعليمات القفز المشروط:

وهي التعليمات التي تقوم بالقفز فقط عند تحقق شرط معين. وهي تعتمد على رايات سجل الحالة الستة التي تقوم وحدة الحساب والمنطق بوضع قيمها بعد كل عملية.

مثال:

```
JC      Save
```

إذا كانت راية الحمل مرفوعة، يقفز المعالج إلى السطر ذي اللصاقة Save، وإلا ينفذ التعليمة التالية.

نجد في الجدول التالي جميع تعليمات القفز المشروط في المعالج 8086.

Inst.	Description	Inst	Description
JA	Jump if above	**JNE**	Jump if not equal
JAE	Jump if above or equal	**JNG**	Jump if not greater
JB	Jump if below	**JNGE**	Jump if not greater or equal
JBE	Jump if below or equal	**JNL**	Jump if not less
JC	Jump if carry	**JNLE**	Jump if not less or equal
JCXZ	Jump if CX is zero	**JNZ**	Jump if not zero
JE	Jump if equal	**JNO**	Jump if not overflow
JG	Jump if greater	**JNP**	Jump if not parity
JGE	Jump if greater or equal	**JNS**	Jump if not sign
JL	Jump if less	**JO**	Jump if overflow
JLE	Jump if less or equal	**JPO**	Jump if parity odd
JNA	Jump if not above	**JP**	Jump if parity
JNAE	Jump if not above or equal	**JPE**	Jump if parity even
JNB	Jump if not below	**JS**	Jump if sign
JNBE	Jump if below or equal	**JZ**	Jump if zero
JNC	Jump if no carry		

تعد تعليمات القفز المشروط من نمط القفز القريب والقصير. ولذا يجب أن تقع الوجهة ضمن قطاع البرنامج ذاته، ,أن تكون مسافة القفز محصورة بين 128- و127 بايت. أما عندما تكون مسافة القفز أكبر من ذلك فهنا

لا بد من القفز على مرحلتين: في المرحلة الأولى إلى مكان قريب وقصير باستخدام تعليمة القفز الشرطية إلى موقع نضع فيه تعليمة قفز لاشرطية إلى الموقع المرغوب لإتمام المرحلة الثانية.

لتوضيح ذلك سوف نعيد صياغة مثال التحكم بدرجة حرارة محلول كما هو مبين في البرنامج التالي. حيث يوضح هذا البرنامج كيفية استخدام القفز البعيد نحو Heater_off في حال كان هذه اللصاقة أبعد من 127 بايت عن مكان القفز

```
Temp_In:    MOV  DX, 0FFF8h
            IN   AL, DX
            CMP  AL, 100
            JB   Heater_on
            JMP  Heater_off
Heater_on:  MOV  AL, 80h
            MOV  DX, FFFAh
            OUT  DX, AL
            JMP  Temp_In
Heater_off: MOV  AL,0h
            MOV  DX, FFFAh
            OUT  DX, AL
```

لاحظ أننا استخدمنا هنا تعليمة القفز الشرطية JB بدلاً من JAE سابقاً.

يمكن استخدام تعليمات القفز الشرطية لكتابة البنى البرمجية المختلفة. في المثال السابق قمنا ببناء البنية البرمجية المصاغة بالترميز الكاذب "مادام- افعل" حيث نقوم باختبار الشرط أولاً وبناءً على نتيجة الاختبار نقوم بالقفز أو لا.

يبين المثال التالي كيفية برمجة البنية "من أجل- افعل" (For-Do). حيث تنفذ هذه البنية غالباً بشحن قيمة العداد n داخل سجل من سجلات المعالج، ثم إنقاصه في كل مرة إلى أن يبلغ الصفر، فنخرج عندئذٍ من الحلقة.

مثال توضيحي

نريد قراءة ثمان قيم مخزنة في الذاكرة، وهي تمثل أسعار الكلفة لبعض المنتجات. ونريد الحصول على أسعار مبيعها التي نحصل عليها بإضافة ثابت قدره 7. ولهذا، يجب إضافة هذا العدد إلى القيمة المقروءة ثم تخزين القيم الناتجة في الذاكرة.

تصاغ الخوارزمية السابقة بلغة المجمع كما يلي:

```
            MOV  Ax, Source_data
            MOV  DS, AX
            LEA  BX, Prices    ; Initialize Data Segment and pointer
            MOV  CX, 08h       ; Initialize Counter
```

```
Do_next:    MOV  AL, [BX]
            ADD  AL, 07h
            MOV  [BX], AL
            INC  BX
            DEC  CX
            JNZ  Do_next
```

في هذا البرنامج، نهيئ أولاً سجل مؤشر قطاع المعطيات DS، ثم نشحن انزياح العنوان Prices في السجل BX، كما نشحن السجل CX بالقيمة 08h لأنه سيؤدي دور العداد في حلقة "من أجل–افعل". يُشحن بعد ذلك السجل BX. وأخيراً يُنقص العداد بمقدار 1 ثم تُفحص راية الصفر، فإن لم تكن مرفوعة فهذا يعني أننا لم ننتهي من معالجة جميع القيم.

هناك صيغة أكثر فاعلية في برمجة الحلقات ذات العداد وذلك باستخدام تعليمة LOOP كما سنرى في الفقرة التالية.

5. الحلقات:

يمكن استخدام تعليمات الحلقة LOOP في تكوين حلقات في البرنامج. فهي تتميز بإنقاص السجل CX بشكل آلي في كل مرة وفحص قيمته (وفي بعض الأحيان تفحص راية الصفر) فإذا كانت قيمته غير معدومة، يقرر المعالج القفز إلى اللصاقة المحددة (وهنا نستغني عن تعليمة الإنقاص والمقارنة). يمكن أيضاً استخدام تعليمة JCXZ، التي تختبر قيمة السجل CX فإذا كانت معدومة قفز المعالج إلى اللصاقة المحددة بالتعليمة. ولفهم كيفية استخدام الحلقات سوف ندرس مثالين شهيرين وهما حلقات التأخير ونسخ سلسلة محارف.

1.5. حلقات التأخير:

يراد في بعض التطبيقات أن يقوم المعالج بالانتظار مدة معينة من الزمن. أي أن المبرمج يرغب في تنفيذ حلقة تأخير مدة محدودة. يمكن إجراء ذلك عن طريق إدراج حلقات تأخير (Delay Loops) في البرنامج. ولذلك لا بد من معرفة تردد الهزاز الخارجي الذي يحكم عمل المعالج (أو ساعة العمل)، وحساب زمن تنفيذ التعليمات المعطى من النشرة الفنية للمعالج. فلكل تعليمة زمن تنفيذ محدد يقدر بأدوار الساعة. فمثلاً، يحتاج تنفيذ تعليمة MOV لنقل محتوى سجل إلى آخر لدوري ساعة. أما تعليمة القفز الشرطي JNZ فتحتاج إلى 16 دور ساعة في حال تحقق الشرط وتم القفز، وإلا فهي تحتاج إلى أربعة أدوار ساعة.
لنأخذ مثلاً البرنامج التالي:

		زمن التنفيذ (دورة ساعة)
	MOV CX, N	T0 4
Loop_Time:	NOP	T1 3
	NOP	T2 3
	LOOP Loop_Time	T3 17/5

لنحسب زمن تنفيذ هذه الحلقة بدلالة عدد مرات التكرار N:

$T= T0+N(T1+T2+T3)- =4+N(3+3+17)-12=23N-8$

يُنفذ السطر الأول مرة واحدة, في حين تنفذ بقية الأسطر N مرة. ولكن في المرة الأخيرة لا يتم القفز إلى بداية الحلقة ولكن يتم الخروج من الحلقة وهنا يتم استهلاك 5 أدوار ساعة فقط بدل من 17 عند إجراء عملية القفز، ولهذا تم تنقيص 12(5-17) من الجواب النهائي.

فإذا علمنا أن تردد ساعة المعالج هي 5MHz يكون دور الساعة مساوياً 0.2µs وبالتالي من أجل N=217 فإننا نحصل على تأخير قدره:

$T=(23N-8)x0.2=996.6\ \mu s.$

ويمكن، إن لم نستطع الحصول على زمن التأخير الكافي من حلقة واحدة، وضع حلقتين متداخلتين (Nested Loops) كما يلي:

```
        MOV  BX, count1
LOOP1:  MOV  CX, count2
LOOP2:  LOOP LOOP2
        DEC  BX
        JNZ  LOOP1
```

فالمبدأ هو تكرار عملية شحن القيمة count2 في السجل CX، ثم إنقاص هذا السجل حتى الصفر عدداً من المرات قدره count1. يكون هنا زمن التأخير الكلي هو:

$T=4+count1[4+17count2-12 +3 +17]-12=count1(17count2+12)-8$

2.5. نسخ سلسلة محارف:

تستخدم عملية نسخ سلسلة محارف في خوارزميات معالجة النصوص أو في نسخ جزء من الذاكرة من مكان إلى آخر. نحتاج في هذه العملية إلى استخدام المؤشرات. مؤشر على السلسلة المصدر ومؤشر آخر على سلسلة الوجهة. ولذلك يستخدم السجل SI للتأشير على المصدر والسجل DI للتأشير على الوجهة. كما نحتاج إلى عداد للمحارف التي تم نسخها ونستخدم لذلك السجل CX.

يمكن ترميز هذا البرنامج باستخدام قائمة المهام التالي:

- ضع في SI عنوان بداية المصدر
- ضع في DI عنوان بداية المصدر

كرر

- انقل بايت من المصدر إلى الوجهة
- زد مؤشر المصدر SI.
- زد مؤشر الوجهة DI.
- أنقص العداد CX

إلى أن يصبح العداد = 0.

ويكون نص البرنامج بلغة التجميع كالتالي:

```
MOV   AX,0h
MOV   DS,AX
MOV   ES,AX          ; Initialize ES
MOV   SI,2000h       ; Initialize SI
MOV   DI,2400h       ; Initialize DI
MOV   CX,80h         ; number of bytes to copy
CLD
REP   MOVSB
```

تقوم التعليمة MOVSB بنقل محرف من السلسلة المصدر المؤشر عليها من SI إلى السلسلة الوجهة المؤشر عليها من DI. وتزيد هذه التعليمة آلياً قيمة المؤشرين SI وDI وهذا ما يفسر وجود التعليمة CLD محو راية الاتجاه (التي تقوم بتأهيل الزيادة الآلية لسجلات الدليل). وتدل التعليمة REP على تكرار العملية حتى يصبح محتوى السجل CX معدوماً.

6. التعليمات الموسعة:

عندما يتكرر استخدام مجموعة من التعليمات عدداً كبيراً من المرات في البرنامج ذاته فإن هناك طريقتان لمعالجة ذلك: في الطريقة الأولى، يمكن وضع هذه التعليمات في إجرائية مستقلة (والتي سنتعرض لها في الفصل القادم) ومن ثم طلب هذه الإجرائية بواسطة تعليمة الاستدعاء CALL عند الحاجة في لبرنامج الرئيسي. تمكن هذه الطريقة من تقليص حجم البرنامج حيث أن مجموعة التعليمات ستكتب لمرة واحدة فقط في ذاكرة البرنامج ضمن الإجرائية، ولكن تعاني هذه الطريقة من البطء في التنفيذ واستهلاك في ذاكرة المكدس. أما الطريقة الثانية فهي لا تقلص في حجم البرنامج ولكن تمكن اختصار نص البرنامج حيث يتم وضع هذه التعليمات في وحدات برمجية تسمى التعليمات الموسعة (Macro). وهذه الطريقة مفيدة عندما يكون عدد التعليمات المكررة قليلاً.

مثال:

```
PUSH_ALL MACRO
        PUSHF
        PUSH AX
        PUSH BX
        PUSH CX
        PUSH DX
        PUSH BP
        PUSH SI
        PUSH DI
        PUSH DS
        PUSH ES
        PUSH SS
ENDM
```

قمنا هنا بتعريف تعليمة موسعة تدعى PUSH_ALL وهي مجموعة تعليمات تخزين سجلات المعالج في المكدس. وعندما يجد المجمع هذه التعليمة في مكان ما في البرنامج فإنه يقوم باستبدالها بمجموعة التعليمات التي يحتويها وذلك قبل ترجمة البرنامج إلى لغة الآلة.
بالإضافة إلى ذلك يمكن تعريف تعليمات موسعة ذات معاملات. مثل التعليمة الموسعة التالية:

```
MOV_ASCII MACRO Number, Source, Destination
        MOV   CX, Number
```

```
        LEA   CX, Source
        LEA   DI, Destination
        REP   MOVSB
ENDM
```

تقوم هذه التعليمة الموسعة بنسخ سلسلة محارف بطول Number من عنوان المصدر Source إلى عنوان الوجهة Destination. وعند طلب هذه التعليمة في البرنامج، يُستعاض عن المعاملات بالقيم الفعلية على النحو الآتي مثلاً:

```
MOV_ASCII  03Dh, Str1, Str2
```

فالتعليمة الموسعة اذاً هي مجموعة من تعليمات المعالج التي تعطى اسماً في بداية البرنامج ويمكن باستخدامها جعل البرنامج سهل القراءة.

أسئلة الفصل الخامس

1. نريد الوصول إلى المعلومة المخزنة في العنوان الحقيقي 2437Ah في قطاع المعطيات، بفرض أن قيمة السجل DS=2000h، فما هو الانزياح الواجب جمعه للوصول إلى تلك المعلومة ؟

Offset=2437Ah – 20000h = 0437Ah.

2. ما هو عمل التعليمتين التاليتين في المعالج 8086؟

```
MOV     DX,0FFF8h
IN      AL,DX
```

قراءة محتوى بوابة الدخل على 8 بت ذات العنوان 0FFF8 ووضع الناتج في السجل AL.

3. ما هو عمل التعليمتين التاليتين في المعالج 8086؟

```
CMP     AL,100
JAE     Heater_off
```

مقارنة السجل AL مع القيمة 100 والقفز إلى اللصاقة Heater_off إذا كان AL أكبر أو يساوي القيمة 100.

4. ما هو عمل التعليمات التالية في المعالج 8086؟

```
MOV     AL,01h
MOV     DX,FFFAh
OUT     DX,AL
```

كتابة القيمة 01H على 8 بت في بوابة الخرج ذات العنوان FFFAh.

5. هناك خمسة أنواع للقفز اللاشرطي، فما هي؟

1. القفز القريب
2. القفز القريب والقصير
3. القفز القريب الغير مباشر
4. القفز البعيد
5. القفز البعيد غير المباشر.

6. ما الفرق بين القفز اللاشرطي القريب والقفز اللاشرطي القريب والقصير.

في القفز اللاشرطي القريب يكون مقدار انزياح اللصاقة مرمزاً على 16 بت بصيغة المتمم الثنائي، بينما في القفز اللاشرطي القريب والقصير يكون مقدار الانزياح مرمزاً على 8 بت فقط بصيغة المتمم الثنائي.

7. ما الفرق بين القفز اللاشرطي المباشر والقفز اللاشرطي غير المباشر؟

في القفز اللاشرطي غير المباشر يتم الحصول على معلومات مكان القفز من رمز التعليمة بلغة الآلة. أما في القفز اللاشرطي غير لمباشر فإن معلومات مكان القفز تكون مخزنة في الذاكرة.

8. اشرح كيف يقوم المعالج بعملية القفز البعيد الغير مباشر.

في هذا النوع من القفز يتم شحن مؤشر التعليمات وقطاع البرنامج من أربعة مواقع متتالية من الذاكرة. الموقعين الأولين لشحن قيمة مؤشر البرنامج والموقعين الأخيرين لشحن قيمة سجل قطاع البرنامج.

أسئلة خيارات متعددة

1. ماهي اللغة التي يفهمها المعالج ويقوم بتنفيذها؟
   A. لغة الألة
   B. لغة التجميع
   C. لغة C
   D. جميع ما سبق.

2. عند تنفيذ تعليمة ما في المعالج 8086، فإن السجل IP يحتوي على:
   A. رمز الآلة للتعليمة الحالية.
   B. رمز الآلة للتعليمة التالية.
   C. انزياح عنوان التعليمة الحالية عن بداية قطاع البرنامج.
   D. انزياح عنوان التعليمة التالية عن بداية قطاع البرنامج.

3. نستطيع باستخدام تعليمة القفز اللاشرطي في المعالج 8086 القفز إلى:
   A. إلى لصاقة ذات انزياح يمكن ترميزه على 8 بت.
   B. إلى لصاقة ذات انزياح يمكن ترميزه على 16 بت.
   C. إلى أي موقع في الذاكرة.
   D. كل ما سبق.

4. نستطيع باستخدام تعليمات القفز الشرطي في المعالج 8086 القفز إلى:

A. إلى لصاقة ذات انزياح يمكن ترميزه على 8 بت.

B. إلى لصاقة ذات انزياح يمكن ترميزه على 16 بت.

C. إلى أي موقع في الذاكرة.

D. كل ما سبق.

5. عند القفز اللاشرطي إلى تعليمة تبعد عن قيمة IP الحالية بمقدار 200 بايت، ما هو نوع القفز الأنسب؟

A. القفز القريب والقصير.

B. القفز القصير.

C. القفز البعيد المباشر.

D. القفز البعيد غير المباشر.

6. تقوم التعليمة JCXZ Label بالعمل التالي:

A. القفز الشرطي إلى اللصاقة Label إذا كانت راية الصفر في سجل الحالة مرفوعة.

B. القفز الشرطي إلى اللصاقة Label إذا كان محتوى السجل CX صفراً.

C. القفز إلى العنوان ذو الانزياح CX في قطاع البرنامج في كانت راية الصفر في سجل الحالة مرفوعة.

D. إنقاص السجل CX ثم القفز الشرطي إلى اللصاقة Label إذا كان محتوى السجل CX صفراً.

7. بعد استخدام تعليمة المقارنة CMP AL,20 نريد أن نقوم بالقفز إلى اللصاقة Do_Next إذا كان AL أقل تماماً من 20. ما هي اتعليمة المناسبة لذلك:

A. JNE

B. JE

C. JNC

D. JBE

8. التعليمات الموسعة (Macro) هي:

A. هي إجرائية جزئية تحتوي على مجموعة من التعليمات بلغة المجمع.

B. هي تعليمات إضافية بلغة المجمع موجودة في النسخة المتطورة من المعالج.

C. وحدة برمجية تضم عدة تعليمات بلغة المجمع يتم إدراجها في البرنامج عند استدعاء التعليمة الموسعة.

D. هي تعليمات خاصة للتعامل مع الذواكر الخارجية الموسعة والتي تحتوي على المعطيات ذات الحجم الكبير.

مسائل

1. احسب زمن تنفيذ الحلقة التالية بدلالة قيمة N مقدراً بعدد دورات الساعة، ثم بوحدة الثانية إذا علمت أن تردد ساعة العمل هو Fosc=5 MHz و N=100.

		زمن التنفيذ (دورة ساعة)
	MOV CX, N	T0 4
Loop_Time:	NOP	T1 3
	NOP	T2 3
	LOOP Loop_Time	T3 17/5

الحل

T= T0+N(T1+T2+T3)− =4+N(3+3+17)−12=23N−8 clock period

N=100: T=23x100−8=2292 Clock Period=2292/5x106=458.4 µs

2. اشرح عمل البرنامج التالي ثم اذكر ماهو العمل الذي يقوم به:

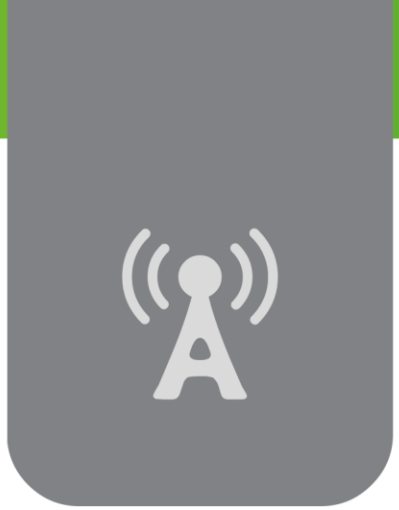
```
MOV  AX,0h
MOV  DS,AX
MOV  ES,AX
MOV  SI,2000h
MOV  DI,2400h
MOV  CX,80h
CLD
REP  MOVSB
```

الحل:

في البداية يتم وضع قيمة الصفر في كل من سجل قطاع المعطيات وسجل قطاع الذاكرة الإضافي. ثم يتم وضع القيمة البدائية لمؤشر على معطيات المصدر SI=2000h والقيمة البدائية مؤشر معطيات الوجهة DI=2400h. ثم يتم إنشاء حلقة نسخ باستخدام التعليمة REP MOVSB تتكرر بمقدار 128 مرة (CX=80H) لنسخ المعطيات من عنوان المصدر إلى عنوان الوجهة.

بالنتيجة تقوم هذه التعليمات بنسخ 128 بايت من العنوان 2000H من الذاكرة إلى العنوان 2400h.

رقم التمرين	الإجابة الصحيحة
1	A
2	D
3	D
4	A
5	B
6	B
7	C
8	C



الفصل السادس: البرامج الفرعية والإجرائيات

الكلمات المفتاحية:

الإجرائيات القريبة والبعيدة، المكدس، تمرير المعاملات للإجرائية.

ملخص:

يهدف هذا الفصل لشرح مفهوم البرامج الفرعية وأهميتها في بناء البرامج ذات التعقيد المتزايد بشكل يسهل العمل ضمن النهج التنازلي في حل المسائل المعقدة وتوزيع المهام ضمن فريق عمل. نشرح هنا آلية عمل البرامج الفرعية وطرق تمرير المعاملات ودور المكدس في هذا العمل

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

- مفهوم الإجرائيات في لغة التجميع
- آلية عمل الإجرائيات
- عمل المكدس في طلب الإجرائية
- طرق تمرير المعاملات للإجرائيات

الفصل السادس: البرامج الفرعية والإجرائيات

يهدف هذا الفصل لشرح مفهوم البرامج الفرعية وأهميتها في بناء البرامج ذات التعقيد المتزايد بشكل يسهل العمل ضمن النهج التنازلي في حل المسائل المعقدة وتوزيع المهام ضمن فريق عمل. نشرح هنا آلية عمل البرامج الفرعية وطرق تمرير المعاملات ودور المكدس في هذا العمل.

1. مقدمة:

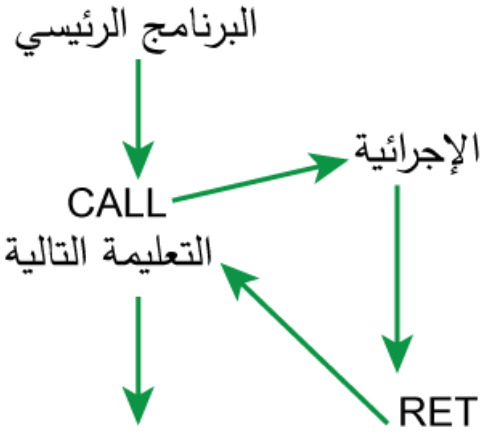
عندما يكثر استخدام نفس المجموعة من التعليمات بشكل متكرر في برنامج ما يصبح من الأفضل جمعها في وحدة واحدة تسمى برنامجاً جزئياً (Subprogram). ومن ثم استدعاء هذه الإجرائية في أي مكان من البرنامج.
توفر البرامج الفرعية للمبرمج عدة ميزات منها:

- الكتابة المضغوطة للبرنامج: أي تقليص حجم الملف المصدري، اذ يستغنى عن تكرار مجموعة التعليمات بتعليمة واحدة وهي طلب للبرنامج الفرعي
- تنظيم البرنامج: يغدو البرنامج المكتوب أسهل للقراءة والفهم
- سهولة التطوير: يمكن عند كتابة البرنامج اختبار كل برنامج فرعي على حدة، والتوثق من أدائه، قبل مكاملته في البرنامج النهائي
- تسهيل اعتماد النهج التنازلي في حل مسألة البرمجة:. ففي هذا النهج، تعرف المسألة جيداً، ثم يجزأ العمل الكلي في وحدات أصغر، وتقسم بدورها إلى أجزاء أصغر فأصغر، إلى أن تصبح الخوارزمية جلية تماماً

يتم طلب البرنامج الفرعي بتعليمة CALL لتنفيذ مجموعة التعليمات التي يتضمنها. وتسمى البرامج الفرعية إجرائيات (Procedures).

2. عمل الإجرائيات:

تعمل الإجرائيات على النحو التالي: عندما يصادف المعالج تعليمة CALL في البرنامج الرئيسي، فإنه يخزن أولاً قيمة مؤشر التعليمات في ذاكرة المكدس ومن ثم يشحن عنوان بداية الإجرائية في مؤشر التعليمات IP. وعندئذٍ، يبدأ المعالج بتنفيذ الإجرائية انطلاقاً من أول تعليمة في الإجرائية. يستمر تنفيذ التعليمات إلى أن يصادف المعالج RET التي تشير إلى نهاية الإجرائية، فيقوم المعالج بإعادة شحن مؤشر التعليمات بالقيمة التي خزنها في ذاكرة المكدس وبالتالي الذهاب إلى تنفيذ التعليمة التي تلي تعليمة الاستدعاء CALL في البرنامج الرئيسي.
وفي بعض الأحيان تقوم تعليمة الاستدعاء بتخزين قيمة سجل قطاع التعليمات في المكدس أيضاً وذلك عندما تكون الإجرائية موجودة خارج القطاع الحالي.



عملية استدعاء إجرائية

3. تعليمة الاستدعاء:

يمكن أن نميز بين أربعة أنواع لتعليمة CALL، وذلك تبعاً لطريقة الحصول عى عنوان بداية الإجرائية:

1. تعليمة الاستدعاء القريبة لمباشرة

نحصل على عنوان بداية الإجراء بجمع الانزياح المحدد بالتعليمة (قيمة مع إشارة على 16 بت) إلى مؤشر التعليمات IP. يمكن بهذه التعليمة طلب إجرائية تبعد مسافة تقع بين 32768– و 32767+ بايت عن الموقع الحالي.

2. تعليمة الاستدعاء القريبة غير المباشرة

يبقى المعالج في هذه الحالة أيضاً ضمن القطاع ذاته. ولكنه يحصل على مؤشر التعليمات IP من سجل أو من موقعين متتاليين في الذاكرة,

3. تعليمة الاستدعاء البعيدة المباشرة

يخرج المعالج بهذه التعليمة إلى قطاع آخر بحثاً عن بداية الإجرائية. ولذا ينبغي تغيير مؤشر التعليمات IP وسجل قطاع البرنامج CS معاً. ويحصل المعالج على القيم الجديدة من التعليمة التي ستُرمز على 5 بايتات في لغة الآلة. فيدل البايت الثاني والثالث على مؤشر التعليمات، ويحتوي البايت الرابع والخامس على قيمة سجل قطاع البرنامج الجديد.

4. تعليمة الاستدعاء البعيدة غير المباشرة

في هذا النمط يغيّر المعالج القطاع الحالي. ولكنه يحصل على القيمة الجديدة من الذاكرة. فيدل أول موقعين على مؤشر التعليمات IP، في حين يحتوي الموقعان التاليان على قيمة سجل قطاع البرنامج الجديد.

4. تعليمة العودة:

كما ذكرنا سابقاً يعود المعالج من برنامج فرعي (أو إجرائية) بالتعليمة RET. تسترجع هذه التعليمة القيم المخزنة في المكدس وتشحنها في مؤشر التعليمات IP وفي قطاع البرنامج CS. فإذا استخدمت تعليمة CALL من النوع القريب (داخل القطاع)، فإن تعليمة RET تسترجع كلمة واحدة من المكدس (16 بت) وتشحنها في مؤشر التعليمات IP. أما إذا كانت من النوع البعيد، فإن تعليمة RET تسترجع كلمتين من المكدس، إذ تشحن الكلمة الأولى في مؤشر التعليمات IP والثانية في قطاع البرنامج CS.

توجد أربعة أنواع من تعليمة العودة RET:

1. تعليمة العودة القريبة: وهي تنسخ كلمة واحدة من المكدس إلى مؤشر التعليمات IP.
2. تعليمة العودة القريبة مع انزياح: وهي مثل سابقتها، ولكنها تجمع إلى مؤشر التعليمات قيمة ثابتة محددة بالتعليمة.
3. تعليمة العودة البعيدة: تنسخ كلمتين من المكدس إلى مؤشر التعليمات وسجل قطاع البرنامج.
4. تعليمة العودة البعيدة مع انزياح: وهي مثل سابقتها ولكنها تضيف إلى مؤشر التعليمات قيمة ثابتة محددة بالتعليمة.

5. استخدام المكدس في البرامج الفرعية:

ذكرنا أن المكدس هو عبارة عن جزء من الذاكرة يستخدم لتخزين عناوين العودة. كما يمكن الاستفادة منه، أثناء تنفيذ إجرائية ما، في تخزين محتوى السجلات المستخدمة في البرنامج الرئيسي. أما العمل الثالث للمكدس فهو حفظ المعطيات أو العناوين المعالجة داخل إجرائية ما.

يعمل المكدس على مبدأ الداخل آخراً يخرج أولاً LIFO (Last In First Out). يدير المعالج المكدس بواسطة سجلين: سجل قطاع المكدس SS وسجل مؤشر المكدس SP. يستخدم هذان السجلان لتوليد العناوين الحقيقية لمواقع موجودة ضمن المكدس.

يتم التخزين والقراءة من المكدس بطريقتين: بشكل آلي من قبل المعالج عند حفظ عناوين العودة قبل استدعاء إجرائية (أو مقاطعة) وعند العودة إلى مكان الاستدعاء بعد انتهاء الإجرائية. أما الطريقة الأخرى فهي تتم بشكل برمجي لحفظ المعطيات من خلال التعليمتين PUSH للكتابة و POP للقراءة.

عند التخزين في المكدس، نبدأ دوماً بالموقع ذي العنوان الأعلى ثم ننزل متجهين نحو الموقع الأدنى. وهذا يعاكس طريقة التخزين في الذاكرة من الأسفل للأعلى.

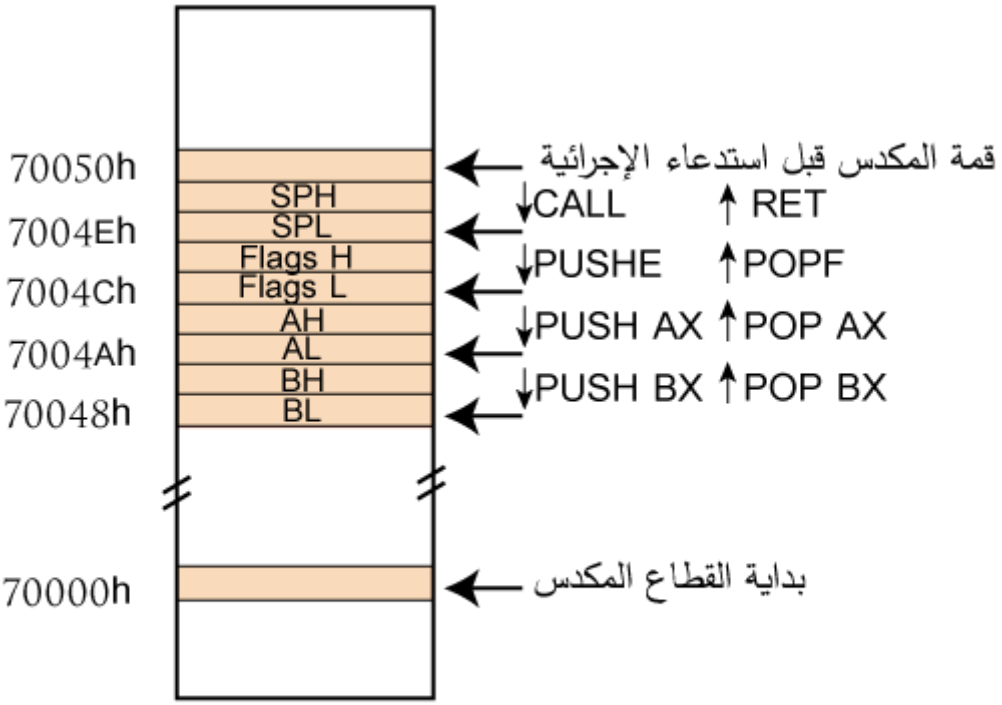
لنفرض أن عنوان بداية المكدس هو 70000h ونفرض أننا نريد أن يكون حجم ذاكرة المكدس هو 80 بايت (50h). ولذلك فإن مؤشر المكدس سوف يشير في بداية البرنامج إلى قمة المكدس في الموقع 70050h. ويتم وضع القيم البدائية للسجلين SS وSP بالشكل التالي:

```
MOV     AX,7000h
MOV     SS,AX
LEA     SP, 0050h
```

لنفرض أننا نريد استدعاء إجرائية تدعى Proc حيث تقع هذه الإجرائية ضمن القطاع نفسه (استدعاء قريب). بالإضافة إلى ذلك، لا نريد أن تقوم هذه الإجرائية بتغيير محتوى سجل الحالة ومحتوى السجلين AX وBX لأننا نحتاج إلى القيم المخزنة فيها بعد تنفيذ الإجرائية. لذلك نستخدم تعليمة PUSHF لتخزين سجل الحالة في المكدس وتعليمة PUSH لتخزين السجلين AX وBX في بداية الإجرائية واستعادة هذه القيم (بالترتيب العكسي) في نهاية الإجرائية.

```
        CALL    Proc
        …
        …
Proc:   PUSHF
        PUSH    AX
        PUSH    BX
        …
        …
        POP     BX
        POP     AX
        POPF
        RET
```

عند استدعاء الإجرائية يقوم المعالج بإنقاص مؤشر المكدس SP بمقدار 2 ثم يقوم بتخزين سجل مؤشر البرنامج IP بشكل آلي في موقعين متتاليين لأن IP على بايتين. وعند تنفيذ تعليمة PUSHF يقوم المعالج بإنقاص مؤشر المكدس بمقدر 2 ويخزن قيمة سجل الحالة في المكدس في موقعين متتاليين، وعند تعليمة PUSH الأولى ينقص SP بمقدار 2 ويخزن AX في موقعين متتاليين، ثم بنفس الطريقة يخزن BX.
يُبيّن الشكل التالي محتوى ذاكرة المكدس وقيمة مؤشر المكدس بعد تنفيذ كل عملية.



محتوى ذاكرة المكدس وقيمة مؤشر المكدس بعد تنفيذ كل عملية

وعندما يصل المعالج إلى تعليمة POP الأولى يقوم بقراءة الكلمة (2 بايت) التي يؤشر عليها SP ويخزنها في BX ويزيد SP بمقدار 2. وبنفس الطريقة يتم استعادة قيمة السجل AX، ثم تتم استعادة قيمة سجل الحالة باستخدام التعليمة POPF. وعندما يصل المعالج إلى تعليمة العودة RET في نهائية الإجرائية يقوم بقراءة آخر موقعين وتخزينهما في مؤشر البرنامج بطريقة معاكسة لعملية الكتابة ثم زيادة مؤشر المكدس بمقدار 2 ليعود إلى قيمته الأصلية قبل استدعاء الإجرائية.

6. الطلب القريب للإجرائيات:

سنعرض هنا مثالاً يوضح طريقة طلب إجرائية طلباً قريباً.

تعريف المسألة وكتابة الخوارزمية

يمكن كتابة حقت التأخير– كالمثال ذي عرض سابقاً– في صيغة إجرائية تُطلب من أي موقع في البرنامج. فلنكتب على سبيل لمثال برنامجاً يقرأ 100 كلمة من بوابة بفاصل زمني قدره 1ms بين الكلمة والأخرى، ثم يخزن الأوزان الدنيا فقط من هذه القيم في جدول في الذاكرة.

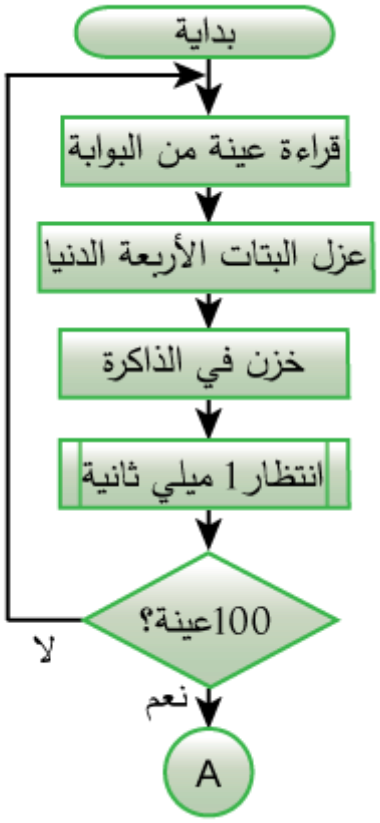
الخوارزمية:

كرر

- احصل على عينة من البوابة
- اعزل الأوزان الدنيا
- خزن في الجدول
- انتظر فترة 1ms

إلى أن يصبح عدد العينات يساوي 100.

يظهر الشكل التالي المخطط التدفقي المعتمد.



لتنفيذ هذه الخوارزمية، فمن الضروري وضع مؤشر على جدول التخزين، ووضع عداد للعينات المقروءة. نزيد بعد معالجة كل عينة مؤشر الجدول بمقدار واحد وننقص العداد بمقدار واحد أيضاً.

تكتب خوارزمية الانتظار مستقلةً عن البرنامج الرئيسي، وهذا ما يجعل تسلسل البرنامج أكثر وضوحاً. إذ لا داعي للغوص في التفاصيل عند قراءة البرنامج الرئيسي.

كتابة البرنامج

نعرض فيما يلي البرنامج كاملاً مع التصريح عن قطاعات الذاكرة المستخدمة. لذلك نلاحظ أن نص البرنامج أطول من بقية الأمثلة، ومع ذلك على القارئ أن يحاول فهم رموز هذا البرنامج للتدرب على تحليل البرامج.

```
SAMPLE_PORT EQU 0FFF8h

DATA_HERE SEGMENT
SAMPLES DW 100 DUP(0)                    ;set up  array of 100 words
DATA_HERE ENDS

STACK_HERE SEGMENT STACK
DW 40 DUP(0)                             ; set stack length of 40 words
STACK_TOP LABEL WORD
STACK_HERE ENDS

CODE_HERE SEGMENT
ASSUME CS:CODE_HERE, DS:DATA_HERE, SS:STACK_HERE

START:
        MOV   AX, DATA_HERE              ;initialize data segment register
        MOV   DS, AX
        MOV   AX, STACK_HERE             ; initialize stack segment register
        MOV   SS, AX
        MOV   SP, OFFSET STACK_TOP; initialize stack pointer to stack top
        LEA   SI, SAMPLES                        ;point SI to the start of array
        MOV   BX, 100                    ;load BX with number of samples
        MOV   DX, SAMPLE_PORT                    ;point DX to input port
NEXT_VALUE:
        IN      AX, DX                   ; read data from port
        AND   AX, 0FFFh                  ; mask upper 4 bits
        MOV   [SI], AX                   ; store data word in array
        CALL  WAIT_1MS                   ; delay of 1ms
```

```
        INC    SI                      ; point SI at next location array
        INC    SI
        DEC    BX                      ; decrement sample counter
        JNZ    NEXT_VALUE              ; repeat until 100 samples done
STOP:
NOP

WAIT_1MS: PROC NEAR
MOV  CX, 23F2h                 ; delay constant into CX
HERE:
LOOP HERE                      ; loop until CX=0
RET
WAIT_1MS: ENDP

CODE_HERE: ENDS

END
```

نصرح في البداية بوجود قطاع المعطيات ونسميه Data_Here بواسطة العبارتين: Data_Here Segment و Data_Here ENDS. أما العبارة SAMPLES DW 100 DUP(0) فهي تحجز 100 كلمة في الذاكرة لتخزين العينات المقروءة. وتُعطى هذه المواقع قيم بدائية معدومة. وليس من الضروري إعطاء قيم بدائية لهذه المصفوفة لأننا سوف نكتب فيها القيم المقروءة ولكن هذا يساعد في عملية تنقيح البرنامج.

نصرح عن وجود قطاع المكدس بواسطة العبارتين: Stack_Here Segment و Stack_Here ENDS. وتحجز العبارة DW 40 DUP(0) أربعين كلمة في هذا القطاع من الذاكرة وتعطيها قيم ابتدائية معدومة (غير ضرورية أيضاً). تُرفِق العبارة Stack_Top LABEL WORD اسماً للموقع ذي العنوان الزوجي التالي لأعلى عنوان في قطاع المكدس.

والآن لننظر إلى البرنامج الرئيسي. ينبغي لنا ن نعلم المجمع عن القطاعات المنطقية المستخدمة في البرنامج. ولهذه تكتب عبارة ASSUME. وهي عبارة لا تضع قيمة ابتدائية في السجلات CS و SS و DS ولكنها توجه عمل المجمع فقط ليحسب الانزياحات بشكل صحيح عند الضرورة.

ينبغي لنا، لاستهلال سجلات القطاعات، أن ننقل القيمة أولاً إلى أحد سجلات المعالج، ثم نشحنها في سجل القطاع الموافق. بعد ذلك، تبدأ المعالجة الفعلية للمسألة المطروحة. فتشحن قيمة ابتدائية في المؤشر SI بالتعليمة

LEA ليؤشر على الموقع الأول في الجدول SAMPLES. ونختار السجل BX، مثلاً ليكون عداداً للعينات المتبقية. ويُشحن ذلك السجل بالقيمة الابتدائية 100.

نقرأ عينة من البوابة FFF8h ونعزل البتات الدنيا بالعملية المنطقية AND مع القيمة الثابتة 0FFFh. وبالتالي يتم تصفير البتات الأربعة العليا. وسبب ذلك أن قيم العينات الموجودة على البوابة هي على 12 بت فقط أما البتات العليا فهي غير موصولة فيزيائياً وتعطي قيم عشوائية عند قراءتها.

يتم تخزين القيمة الناتجة في الجدول باستخدام المؤشر SI. وتُستدعى بعد ذلك إجرائية الانتظار WAIT_1MS بتعليمة طلب قريب ومباشر لأن الإجرائية تقع في القطاع ذاته.

نستعمل لكتابة البرنامج الفرعي WAIT_1MS التوجيهين PROC وENDP في بداية ونهاية الإجرائية على الترتيب. نحصل على التأخير 1ms بشحن السجل CX بالقيمة 23F2h ثم تتكرر عملية إنقاص هذا العداد حتى يصل للصفر وبعدها تنتهي الإجرائية بتعليمة العودة RET التي تعيد السيطرة إلى البرنامج الذي استدعى هذه الإجرائية.

بعد العودة إلى البرنامج الرئيسي، نزيد المؤشر SI مرتين ليؤشر على الكلمة التالية وننقص BX بمقدار واحد. ثم تُفحص راية الصفر، فإن لم تكن مرفوعة قفز المعالج إلى اللصاقة Next_Value وإلا فإننا نصل إلى نهاية البرنامج.

7. تمرير المعاملات من وإلى الإجرائيات:

نرغب عادةً، عند طلب إجرائية ما، أن ندخل للإجرائية بعض المعاملات من البرنامج الأساسي كمعاملات دخل، وكذلك أن تعطي الإجرائية بعض المعاملات كنتائج لهذه الإجرائية. توجد عدة طرق لتمرير المعاملات (Parameter Passing) أهمها: التمرير بالسجلات، والتمرير بمواقع الذاكرة، والتمرير بالمكدس. ولتوضيح الفرق سوف نأخذ المثال البسيط التالي: نعرف إجرائية SUM لها معاملي دخل X وY ومعامل خرج واحد Z. حيث تقوم هذه الإجرائية بجمع X وY وتضع ناتج الجمع في Z.

1.7. التمرير من خلال السجلات:

عندما يكون عدد المعاملات قليل، فإنه يمكن وضع المعطيات في بعض سجلات المعالج قبل طلب الإجرائية ومن ثم تقوم الإجرائية بأخذ هذه السجلات وإجراء العمليات اللازمة ومن ثم وضع النتيجة في السجلات. وبعد استدعاء الإجرائية نأخذ النتائج المخزنة في السجلات ونتابع البرنامج. يوضح البرنامج التالي طريقة تنفيذ إجرائية SUM بهذه الطريقة.

```
DATA_HERE SEGMENT
X DW ?
Y DW ?
Z DW ?
```

```
DATA_HERE ENDS

CODE_HERE SEGMENT
ASSUME CS:CODE_HERE, DS:DATA_HERE, SS:STACK_HERE

        MOV   AX, X
        MOV   BX, Y
        CALL  SUM
        MOV   Z, CX

SUM: PROC NEAR
        PUSHF
        ADD   AX, BX
        MOV   CX, AX
        POPF
        RET
SUM: ENDP
CODE_HERE: ENDS
END
```

في هذا البرنامج نعرف ثلاث متحولات من نمط WORD في قطاع المعطيات وهي X وY وZ بقيم ابتدائية غير محددة. قبل استدعاء الإجرائية يتم شحن معاملات الدخل X وY في السجلين AX وBX على الترتيب. وبعد استدعاء الإجرائية نأخذ الناتج في السجل CX ونخزنه في معامل الخرج Z.

أما في داخل الإجرائية، فإننا نقوم بجمع سجلي الدخل AX وBX التي تمثل المعاملين X وY وضع الناتج في السجل CX الذي اخترناه ليمثل معامل الخرج Z. قمنا بداخل الإجرائية بتخزين واستعادة سجل الحالة من خلال التعليمتين PUSHF وPOPF بفرض أننا لا نريد أن تغير الإجرائية محتوى سجل الحالة.

2.7. التمرير من خلال الذاكرة

في بعض الحالات قد لا تكفي السجلات لتمرير المعاملات، فيستعاض عنها بالذاكرة. وهنا يقوم البرنامج الأساسي بتخزين المعاملات في مكان معين من الذاكرة. حيث تقوم الإجرائية بقراءة هذا لمعاملات من المواقع المحددة في الذاكرة وتخزين النتائج في المكان المخصص لها في الذاكرة.

```
CODE_HERE SEGMENT
ASSUME CS:CODE_HERE, DS:DATA_HERE, SS:STACK_HERE

        CALL  SUM

SUM: PROC NEAR
        PUSHF
        MOV   AX, X
        MOV   BX, Y
        ADD   AX, BX
        MOV   Z, AX
        POPF
        RET
SUM: ENDP
CODE_HERE: ENDS
```

قمنا في هذه البرنامج باستدعاء الإجرائية SUM مباشرة في البرنامج الأساسي. أما في داخل الإجرائية فإننا قمنا بشحن قيم المعاملات من الذاكرة مباشرة وإجراء عملية الجمع وتخزين الناتج في الذاكرة. وبالتالي لجمع أي قيمتين يجب نقل هاتين القيمتين إلى المكانين المخصصين في الذاكرة لمعاملات الدخل في المتحولين X و Y. ونحصل على الناتج حصراً في المتحول Z. وهذا ما يكافئ بلغات البرمجة العالية أننا قمنا بتعريف المعاملات كمتحولات عامة كلية (Global) في البرنامج الأساسي.

3.7. التمرير من خلال المكدس:

تعتبر هذه الطريقة شائعة جداً في مترجمات اللغات عالية المستوى. يجب لتمرير المعاملات إلى إجرائية ما عبر المكدس، أن ندفعها أولاً إلى المكدس في البرنامج الرئيسي قبل طلب الإجرائية. ثم تقوم الإجرائية بقراءة هذه المعاملات من المكدس. وبالطريقة ذاتها، يجب عند تمرير نتائج الإجرائية إلى البرنامج الرئيسي، على الإجرائية أن تكتبها في المكدس. ثم يقوم البرنامج الرئيسي بقراءتها من المكدس.

من أجل قراءة القيم المخزنة في المكدس من دون تغيير مؤشر المكدس فإننا نستخدم النفاذ غير المباشر باستخدام السجل BP. حيث نقوم بنسخ محتوى SP في BP ومن ثم نحسب انزياح القيمة المخزنة عن قمة المكدس.

```
CODE_HERE SEGMENT
ASSUME CS:CODE_HERE, DS:DATA_HERE, SS:STACK_HERE
        MOV   AX, X
        MOV   BX, Y
        PUSH AX
        PUSH BX
        CALL  SUM
        POP    AX
        POP    AX
        MOV   Z, AX

SUM: PROC NEAR
        PUSHF
        PUSH BP
        MOV   BP,SP
        MOV   AX,[BP+8]
        MOV   BX,[BP+6]
        ADD    AX, BX
        MOV   [BP+8],AX
        POP    BP
        POPF
        RET
SUM: ENDP
CODE_HERE: ENDS
```

في هذا لبرنامج نقوم بتخزين معاملات الدخل في المكدس قبل طلب الإجرائية. وعند طلب الإجرائية بشكل قريب يقوم المعالج بتخزين مؤشر التعليمات في المكدس. وفي داخل الإجرائية نقوم أيضاً بدفع سجل الحالة وسجل مؤشر القاعدة BP إلى المكدس. وذلك لأننا سوف نستخدم السجل BP في النفاذ غير المباشر إلى ذاكرة المكدس.

لقراءة معاملات الدخل من ذاكرة المكدس فإننا لا نستطيع استخدام تعليمة POP وذلك لأن هذه التعليمة تقوم فقط بقراءة آخر قيمة مخزنة في المكدس. لذلك نقوم بنسخ محتوى مؤشر المكدس SP إلى السجل BP. ولتحديد مكان وجود المتحول X في المكدس نلاحظ أنه يوجد بانزياح قدره 8 عن قمة المكدس لأنه تم دفع أربع كلمات بعده في المكدس. فنستخدم تعليمة النفاذ غير المباشر مع انزياح ثابت لقراءة قيمة X المخزنة في المكدس من خلال التعليمة MOV AX, [BP+8]. وبنفس الطريقة يتم قراءة المعامل Y الذي يقع على بعد 6 بايتات عن قمة المكدس. بعد ذلك نقوم بجمع القيمتين وتخزين الناتج في المكدس مكان معامل الدخل X لأننا لم نعد بحاجة إليه. بعد طلب الإجرائية في البرنامج الأساسي يكون مؤشر المكدس عند قيمة المعامل Y لذلك نقوم بعملية POP أولى للعودة إلى مكان المتحول X ومن ثم نقوم بعملية POP ثانية لقراءة قيمة الناتج الذي تم تخزينه مكان المعامل X.

ويجب الانتباه عند استخدام المكدس، إلى مشكلة فيضان المكدس (Stack Overflow) التي تحدث عندما يمتلئ قطاع المكدس ويفيض إلى قطاع آخر. وهذا ما قد يحدث بسبب تخزين كمية كبيرة من المعطيات في المكدس أو نتيجة خطأ برمجي. فإذا لم يعد مؤشر المكدس SP إلى الموقع الذي كان عليه قبل طلب الإجرائية، فإنه سيفيض نتيجة للطلب المتكرر ليؤشر على مكان خارج القطاع. ويحدث هذا عندما يكون عدد تعليمات PSUH لا يساوي عدد تعليمات POP داخل الإجرائية.

8. طلب الإجرائيات البعيدة

نصف إجرائية ما بأنها بعيدة (Far) إذا وقعت في قطاع مغاير للقطاع الذي يحوي تعليمة الطلب CALL. وفي هذه الحالة، يشحن المعالج سجل قطاع البرنامج CS، ومؤشر التعليمات IP بالقيم المناسبة للانتقال إلى الإجرائية البعيدة. ويتم التصريح عن ذلك برمجياً باستبدال الموجه NEAR بالموجه FAR في مثال طلب الإجرائيات القريبة.

أسئلة الفصل السادس

1. كيف يقوم المعالج بتنفيذ إجرائية جزئية يتم طلبها بتعليمة CALL في المعالج ؟

تعمل الإجرائيات على النحو التالي: عندما يصادف المعالج تعليمة CALL في البرنامج الرئيسي، فإنه يخزن أولاً قيمة مؤشر التعليمات في ذاكرة المكدس ومن ثم يشحن عنوان بداية الإجرائية في مؤشر التعليمات IP. وعندئذٍ، يبدأ المعالج بتنفيذ الإجرائية انطلاقاً من أول تعليمة في الإجرائية. يستمر تنفيذ التعليمات إلى أن يصادف المعالج RET التي تشير إلى نهاية الإجرائية، فيقوم المعالج بإعادة شحن مؤشر التعليمات بالقيمة التي خزنها في ذاكرة المكدس وبالتالي الذهاب إلى تنفيذ التعليمة التي تلي تعليمة الاستدعاء CALL في البرنامج الرئيسي.

2. هناك أربعة أنواع لتعليمة CALL، وذلك تبعاً لطريقة الحصول على عنوان بداية الإجرائية، فما هي؟

1. تعليمة الاستدعاء القريبة لمباشرة.
2. تعليمة الاستدعاء القريبة غير المباشرة.
3. تعليمة الاستدعاء البعيدة المباشرة.
4. تعليمة الاستدعاء البعيدة غير المباشرة.

3. ما هي السجلات التي تدير عمل المكدس في المعالج 8086؟

سجل قطاع المكدس SS وسجل مؤشر المكدس SP.

4. عدد طرق تمرير المعاملات من وإلى الإجرائيات.

1. تمرير المعاملات عن طريق السجلات.
2. تمرير المعاملات عن طريق الذاكرة.
3. تمرير المعاملات عن طريق المكدس.

5. اشرح تعليمات الإجرائية التالية وبين عملها.

```
WAIT: PROC NEAR
        MOV   CX, 23F2h
HERE:
        LOOP HERE
        RET
WAIT: ENDP
```

نعرف هنا إجرائية قريبة باسم WAIT. يتم فيها تعريف حلقة تدعى HERE لا تحتوي على أية تعليمة باستخدام التعليمة LOOP التي تقوم بانقاص CX بمقدار واحد وإعادة لتكرار بمقدار 23F2h=9202 مرة. ومن ثم الخروج من الإجرائية بتعليمة RET.

تقوم هذه الإجرائية بتأخير زمني معين يتعلق بتردد ساعة عمل المعالج.

6. ما هو عمل التعليمتين التاليتين PUSHF وPOPF؟

تقوم التعليمة PUSHF بتخزين سجل الحالة في المكدس وتقوم التعليمة POPF باستعادة قيمة سجل الحالة من المكدس.

أسئلة خيارات متعددة

1. ما الذي لا يشكل فائدة من استخدام الإجرائية الجزئية في كتابة البرامج؟

A. تقليص حجم ملف المصدر.

B. تنظيم البرنامج وسهولة فهمه.

C. تسريع عمل البرنامج.

D. سهولة التطوير.

2. المكدس في المعالج 8086 هو:

A. جزء من ذاكرة البرنامج يتم فيها تخزين تعليمات الإجرائيات الفرعية.

B. جزء من قطاع المعطيات يتم فيها تخزين المعطيات والعناوين.

C. قطاع مستقل من ذاكرة المعطيات لتخزين المعطيات والعناوين.

D. جزء من قطاع الذاكرة الإضافية لتخزين المعطيات ذات الحجم الكبير مثل المصفوفات.

3. بفرض أن قيمة مؤشر المكدس هي SP=1258h. ما هي قيمة هذا السجل بعد تعليمة PUSH AX؟

A. SP=1256h

B. SP=125Ah

C. SP=1259h

D. SP=1257h

4. عند تمرير المعاملات إلى الإجرائية عن طريق المكدس فإننا نستخرج هذه المعاملات من المكدس عن طريق:

A. تعليمة POP بعدد معاملات الإجرائية.

B. تعليمة POPF بعدد معاملات الإجرائية.

C. تعليمة MOV غير المباشرة باستخدام السجل BP.

D. تعليمة MOV غير المباشرة باستخدام السجل SP.

مسائل

1. نعرف قطعة تدعى VARS من الذاكرة تحوي على متحولات البرنامج X وY و Z وجميعها على 16 بت (Word) بالإضافة إلى مصفوفة تدعى SAMPLES تحوي على 100 كلمة على 16 بت وذلك بالشكل التالي:

```
VARS SEGMENT
X DW 0100h
Y DW 0040h
Z DW ?
SAMPLES DW 100 DUP(0)
VARS ENDS
```

- على ماذا يدل السطر الثاني X DW 0100h؟
- على ماذا يدل السطر الثالث Y DW ?؟
- على ماذا يدل السطر الخامس SAMPLES DW 100 DUP(0)؟
- اشحن سجل قطاع المعطيات DS بالقيمة المناسبة لكي تكون بداية القطاع في بداية كتلة المعطيات VARS.

الحل

- يدل السطر الثاني على حجز متحول من نمط Word يدعى X قيمته البدائية هي 0100h.

- يدل السطر الرابع على حجز متحول من نمط Word يدعى Z بقيمة بدائية غير محددة.
- يدل السطر الخامس على حجز مصفوفة من نمط Word تدعى SAMPLES بقيم بدائية معدومة.
- يتم الشحن باستخدام التعليمتين:

```
MOV   AX, VARS
MOV   DS, AX
```

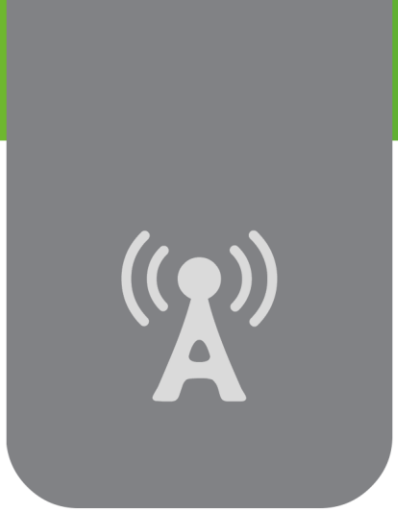
2. اكتب إجرائية قريبة تدعى SUM10 تقوم بجمع مجموعة من 10 قيم كل منها على 16 بت، باستخدام تمرير المعاملات عن طريق السجلات حيث يتم تمرير عنوان بداية المصفوفة عن طريق السجل BX ويتم تخزين الجواب في السجل AX.

الحل:

نقوم في البداية بتصفير قيمة المجموع AX=0 وتعريف عداد باستخدام السجل CX بقيمة بدائية مقدارها 10. ثم نقوم بتعريف حلقة تدعى S1 يتم فيها جمع عنصر واحد من المصفوفة باستخدام العنونة غير المباشرة. وفي كل مرة يتم زيادة مؤشر المصفوفة بمقدار 2 (كل كلمة هي 2 بايت) وإنقاص عداد الحلقة بمقدار واحد حتى يصل لقيمة الصفر فتنتهي الحلقة ونخرج من الإجرائية.

```
SUM10: PROC NEAR
       MOV   CX, 10
       MOV   AX, 0
S1:    ADD   AX, [BX]
       INC   BX
       INC   BX
       DEC   CX
       JNZ   S1
       RET
SUM10: ENDP
```

رقم التمرين	الإجابة الصحيحة
1	C
2	C
3	A
4	C



الفصل السابع: المقاطعات

الكلمات المفتاحية:

المقاطعات، إشارات المقاطعات، جدول متجهات المقاطعة، إجرائية المقاطعة.

ملخص:

يهدف هذا الفصل إلى فهم مفهوم المقاطعات في المعالج وأهميتها في تمكين المعالج من الاستجابة لأحداث داخلية أو أحداث قد ترد من الطرفيات الخارجية. سوف نبين في هذا الفصل كيفية التعامل مع المقاطعات وطريقة كتابة إجرائيات الاستجابة لها. كما نبين أنواع المقاطعات ووظيفة كل نوع منها.

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

- مصادر المقاطعة في المعالج 8086
- أنواع المقاطعات
- آلية توجيه المقاطعات من خلال جدول متجهات المقاطعة
- طريقة الاستجابة لمقاطعة من خلال إجرائية المقاطعة

الفصل السابع: المقاطعات

يهدف هذا الفصل إلى فهم مفهوم المقاطعات في المعالج وأهميتها في تمكين المعالج من الاستجابة لأحداث داخلية أو أحداث قد ترد من الطرفيات الخارجية. سوف نبين في هذا الفصل كيفية التعامل مع المقاطعات وطريقة كتابة إجرائيات الاستجابة لها. كما نبين أنواع المقاطعات ووظيفة كل نوع منها.

تسمح معظم المعالجات الصغرية بمقاطعة تنفيذ البرنامج بناءً على إشارات خارجية أو تعليمات خاصة بها. فعند مقاطعة المعالج، فإنه يتوقف عن تنفيذ برنامجه الحالي، ويستدعي إجرائية لتخديم هذه المقاطعة. وبعد الانتهاء من تنفيذ هذه الإجرائية، يعود المعالج إلى البرنامج السابق، ويستأنف التنفيذ من النقطة التي توقف عندها.
سنعرض في هذا الفصل آلية استجابة المعالج 8086 للمقاطعات وطريقة كتابة إجرائيات خدمة المقاطعة.

1. وصف مقاطعات المعالج:

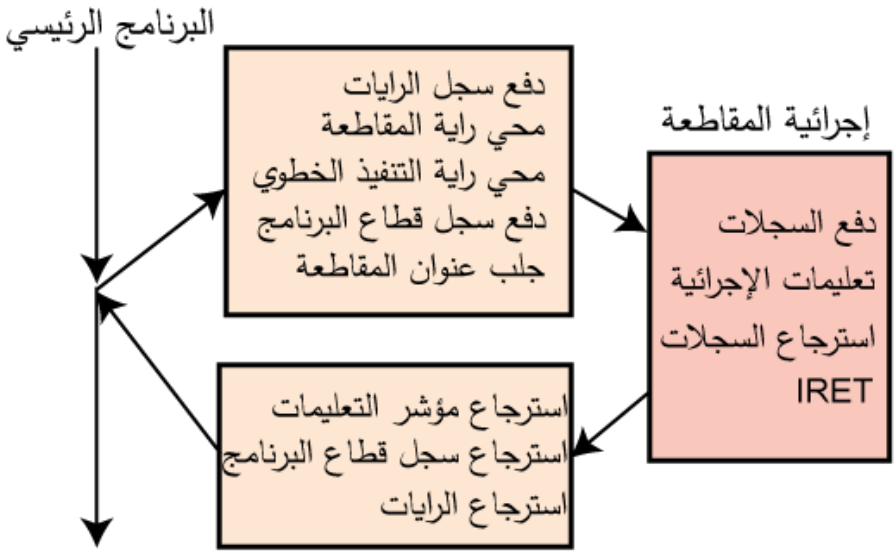
يمكن أن يُقاطع عمل المعالج 8086 من ثلاث مصادر، وهي:

1. إشارة خارجية تطبق على المربط NMI (وهي مقاطعة غير قابلة للحجب)، أو على المربط INTR (وهي مدخل مقاطعة قابلة للحجب). وتسمى المقاطعة الناجمة عن مثل هذه الإشارات مقاطعة الكيان الصلب (Hardware Interrupt).
2. تنفيذ تعليمة المقاطعة INT، وتدعى المقاطعة المبرمجة (Software Interrupt).
3. تحقق شرط معين نتيجة تنفيذ تعليمة ما في المعالج وهي عادة تدل على خطأ داخلي. ومثال ذلك، المقاطعة الناتجة عن التقسيم على صفر، إذ يتوقف البرنامج تلقائياً عندما نحاول قسمة عدد ما على الصفر. وتسمى هذه المقاطعات الشرطية بالمقاطعات البرمجية أيضاً.

يفحص المعالج في نهاية كل دور تعليمة (Instruction Cycle) حالة المقاطعات، فإذا وجد أن هناك مقاطعة ما، فإنه يستجيب لها بالخطوات التالية:

1. ينقص مؤشر المكدس بمقدار 2، ويدفع بسجل الرايات إلى المكدس.
2. يلغي تأهيل المدخل INTR بمحو راية المقاطعة في سجل الرايات.
3. يضع صفراً في راية التنفيذ الخطوي TRAP المحتواة في سجل الرايات.
4. ينقص مؤشر المكدس بمقدار 2، ويدفع محتوى سجل قطاع البرنامج الحالي إلى المكدس.
5. ينقص مؤشر المكدس مرة ثانية بمقدار 2، ويدفع محتوى مؤشر التعليمات الحالي إلى المكدس.
6. يجري المعالج قفزاً غير مباشر إلى بداية الإجرائية المكتوبة لخدمة المقاطعة.

بكلمات أخرى، يدفع المعالج سجل لرايات إلى المكدس، ويلغي تأهيل التنفيذ الخطوي، ويحجب مخل المقاطعة INTR، ثم يقوم باستدعاء إجرائية خدمة المقاطعة استدعاءً بعيداً وغير مباشر.

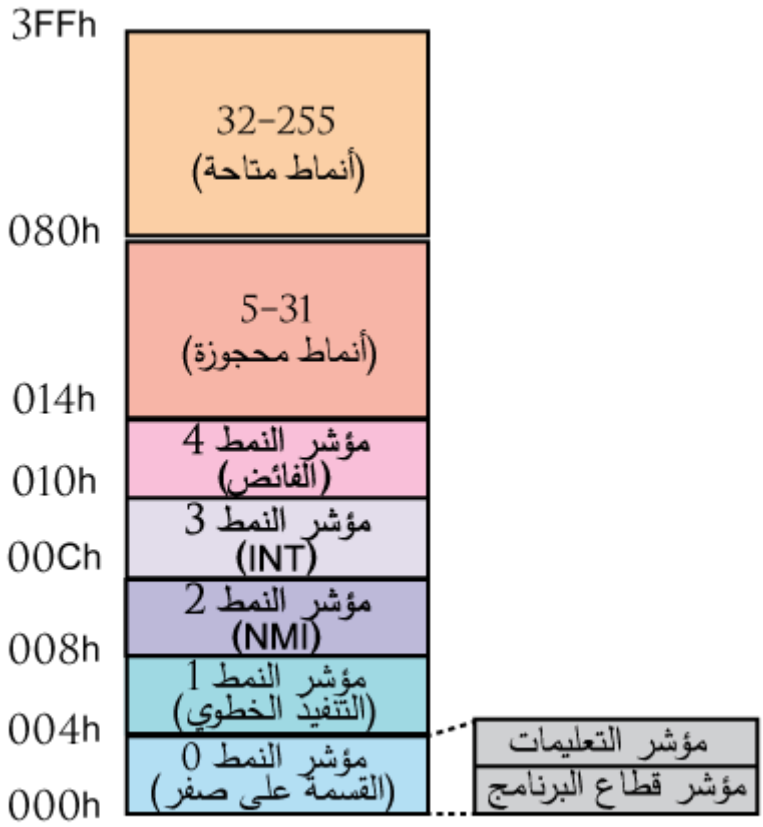


مراحل الاستجابة للمقاطعة

نلاحظ أن تعليمة RETI في نهاية إجرائية خدمة المقاطعة تعيد السيطرة إلى البرنامج الرئيسي. ونذكر بأن المعالج 8086، عندما يطلب إجرائية بصورة بعيدة وغير مباشرة، يضع قيمة جديدة في سجل قطاع البرنامج CS وقيمة جديدة في مؤشر التعليمات IP. ويحصل المعالج في تلك الحالة على القيم الجديدة للسجلات CS وIP من أربعة مواقع متلاحقة في الذاكرة. وبالمماثلة، عندما يستجيب المعالج إلى إحدى المقاطعات فإنه يتوجه إلى الذاكرة في بداية إجرائية المقاطعة للحصول على قيمة السجلين CS وIP.

يحتوي أي نظام حاسوبي مبني على المعالج 8086 في ذاكرته المحصورة بين العنوان 0h والعنوان 03FFh على جدول يخزن عناوين البداية لبرامج خدمة المقاطعات. ولما كانت كل إجرائية مقاطعة تتطلب تخزين أربع بايتات للسجلين CS وIP فإن الجدول يستطيع تخزين العناوين المتعلقة بـ 256 إجرائية خدمة مقاطعة. ويسمى عنوان بداية كل إجرائية متجه المقاطعة (Interrupt Vector) أو مؤشر المقاطعة (Interrupt Pointer)، كما يسمى الجدول، جدول متجهات المقاطعة (Interrupt Vector Table). يظهر الشكل التالي توضع متجهات المقاطعة في لذاكرة. ويشار إلى كلمة مزدوجة (ي ربع بايتات) بعدد يبدأ بالصفر وينتهي بالقيمة 255. ويدعى هذا العدد نوع مقاطعة (Interrupt type).

من ناحية أخرى، تخصص المقاطعات الخمس الدنيا لعمليات معينة في المعالج، مثل القسمة على الصفر والمقاطعة غير القابلة للحجب. أما الأنواع من 5 إلى 32 فهي غير معرفة وتستخدم في معالجات الأجيال المتقدمة. ويستطيع المستثمر أن يستخدم أنواع بمقاطعة المحصورة بين 32 و255 كمقاطعات برمجية أو مقاطعات الكيان الصلب.



توضع متجهات المقاطعة في الذاكرة

عندما يستجيب المعالج لمقاطعة ما، فإنه يذهب تلقائياً إلى الموقع المحدد بتلك المقاطعة في جدول متجهات لمقاطعة، ويحصل منه على عنوان البداية لإجرائية خدمة المقاطعة. ويحصل منه على عنوان البداية لإجرائية خدمة المقاطعة. ولا تجري عملية شحن عناوين البداية في جدول متجهات المقاطعة تلقائياً بل ينبغي للمستثمر أن يشحن في بداية برنامجه، القيم المناسبة في ذلك الجدول.

لنعرض الآن مثالاً تفصيلياً على آلية عمل إحدى المقاطعات.

2. استجابة المعالج لمقاطعة من النوع 0:

تحدث المقاطعة من النوع 0 عند تعرض المعالج لقسمة عدد م على القيمة 0، ونبين فيما يلي طريقة كتابة إجرائية الخدمة لهذه المقاطعة.

يوجد في المعالج تعليمتان للقسمة هما DIV وIDIV. تسمح التعليمة الأولى DIV بقسمة عدد بلا إشارة مرمز على 16 بت ومخزن في السجل AX، على عدد بلا إشارة مرمز على 8 بتات ومخزن في سجل ما من المعالج أو في موقع من الذاكرة. وبعد تنفيذ القسمة، يحتوي السجل AL على القسم الصحيح من خارج القسمة، في حين سيخزن باقي القسمة في السجل AH.

يمكن بواسطة هذه التعليمة قسمة عدد بلا إشارة على 32 بت ومخزن في السجلين DX وAX على عدد من 16 بت ومخزن في سجل ما أو في موقعين متتاليين من الذاكرة. وعندئذٍ، يخزن القسم الصحيح من خارج القسمة في السجل AX. أما باقي القسمة فيخزن في السجل DX. وتعمل التعليمة IDIV عمل التعليمة السابقة ولكنها تسمح بقسمة أعداد ذوات إشارة. عندما يكون المقسوم عليه مساوياً للصفر تصبح عملية القسمة بلا معنى، والخارج هو اللانهاية. وهذا ما يتعذر ترميزه في السجلين AX وDX، فيتعرض عندئذٍ المعالج لمقاطعة من النوع 0.

ينقص المعالج، مستجيباً لهذه المقاطعة، مؤشر المكدس بمقدار 2، ويدفع سجل الرايات إلى المكدس، ثم يكتب القيمة 0 في راية التنفيذ لخطوي TF وراية المقاطعة IF لإلغاء عملهما. يخزن المعالج بعد ذلك عنوان العودة في المكدس. ولإجراء ذلك ينقص مؤشر المكدس بمقدار 2 أيضاً، ويدفع محتوى السجل CS إلى المكدس. ثم ينقص المكدس ثانية بمقدار 2، ويدفع محتوى مؤشر التعليمات IP إلى المكدس.
يحصل المعالج بعد ذلك من جدول متجهات المقاطعة على عنوان البداية لإجرائية المقاطعة من النوع 0، فيشحن محتوى الموقع 02h والموقع 03h في السجل CS، ومحتوى الموقعين 00h و01h في السجل IP.

بعد ذلك، يبدأ المعالج بتنفيذ التعليمة الأولى في إجرائية خدمة المقاطعة. عندما يصل المعالج إلى نهاية الإجرائية، يصادف تعليمة العودة IRET. فيسترجع القيم المخزنة في المكدس ويشحنها في السجلين CS وIP ويزيد مؤشر المكدس في كل مرة مقدار 2. كما يسترجع المعالج سجل الرايات من المكدس في كل مرة بمقدار 2. كما يسترجع المعالج سجل الرايات من المكدس ويزيد مؤشر المكدس بمقدار 2. وكما ذكرنا سابقاً، يلغي المعالج عند مقاطعته تأثير الرايتين TF وIF. ولذا، فإنه يكتب فيهما، عند الانتهاء من إجرائية المقاطعة، قيمهما قبل المقاطعة. وبمعنى آخر، فإن التعليمة IRET تعيد السيطرة إلى البرنامج الرئيسي وتعيد إلى الرايتين IF وTF قيمهما قبل المقاطعة.

1.2. تعريف المسألة وكتابة الخوارزمية:

نريد هنا قسمة أربع كلمات مخزنة في الذاكرة على قيمة مرمزة على بايت واحد، لنحصل في النتيجة على أربع بايتات تمثل القيم النهائية المطلوبة. فإذا كانت نتيجة القسمة غير صالحة (أي أن لقيمة كبيرة بحيث لا يمكن ترميزها في سجل المعالج) فإننا نكتب القيمة 0 في النتيجة النهائية. تُكتب الخوارزمية على النحو التالي:

البرنامج الرئيسي

1. استهلال
2. كرر مايلي
   - احصل على القيمة
   - أجرِ عملية القسمة
   - إذا كانت نتيجة القسمة صالحة افعل
     - خزن النتيجة

وإلا

- خزن القيمة 0

3. إلى أن تنتهي قيم الدخل

إجرائية خدمة المقاطعة

- خزن السجلات
- رفع راية الخطأ
- استرجاع السجلات
- عد إلى البرنامج الرئيسي

يقوم إذاً البرنامج الرئيسي بقراءة القيم من الذاكرة، وإجراء عمليات القسمة، ثم تخزين النتيجة إذا كانت القسمة صالحة. وإلا، فإنه يخزن القيمة 0 بدلاً منها. أما إجرائية خدمة المقاطعة فهي تقوم، بعد تخزين سجلات، برفع راية تدل على الخطأ. إن هذه الراية لا تنتمي إلى سجل رايات المعالج 8086، ولكن يقصد بالراية هنا كتابة قيمة معينة في موقع من الذاكرة للدلالة على حدوث الخطأ. تسترجع بعدئذ إجرائية خدمة المقاطعة السجلات المخزنة في المكدس وتعيد السيطرة إلى البرنامج الرئيسي.

تُفحص نتيجة القسمة في البرنامج الرئيسي بعد تنفيذ القسمة. ويجري ذلك بفحص قيمة راية الخطأ. فإذا كانت راية الخطأ مساوية للصفر، فهذا يدل على أن القسمة صالحة، ومن ثم فإننا نكتب خارج القسمة في الذاكرة. وإلا، فإننا نخزن القيمة 0 في جدول النتائج. ففي هذه الحالة تحتوي راية الخطأ على قيمة غير معدومة ناتجة من حدوث مقاطعة القسمة على صفر. ويتكرر تنفيذ عمليات القسمة إلى أن تنتهي قيم الدخل.

2.2. كتابة قائمة الاستهلال:

ينبغي بعد كتابة خوارزمية البرنامج كتابة قائمة الاستهلال. ففي هذا المثال، تتألف القائمة مما يلي:

- جدول متجهات المقاطعة: يجب وضع عنوان البداية لإجرائية خدمة المقاطعة في المواقع من 0 إلى 3
- قطاع المعطيات: الذي تُخزن فيه قيم الدخل والقيم النهائية
- سجل قطاع المعطيات DS: الذي يؤشر على العنوان القاعدي لقطاع المعطيات ويحتوي على قيم الدخل
- المكدس: الذي تستخدم لتخزين عنوان العودة
- سجل قطاع المكدس SS ومؤشر المكدس SP
- مؤشر إلى بداية جدول النتائج النهائية
- عداد لعدد قيم الدخل المعالجة
- مؤشر إلى بداية قيم الدخل.

3.2. البرنامج بلغة المجمع:

```
DATA_HERE: SEGMENT WORD PUBLIC
INPUT_VALUES DW 0035h, 0855h, 2011h, 1359h
SCALED_VALUES DB 4 DUP(0)
SCALE_FACTOR DB 09h
BAD DIV_FLAG DB 0
DATA_HERE ENDS

STACK_HERE SEGMENT STACK DW 100 DUP(0)
                                    ; set up stack of 100 words
TOP_STACK LABEL WORD                ; pointer to top of stack
STACK HERE ENDS

PUBLIC BAD DIV FLAG                 ; make flag available to
                                    ; other modules

INT'_PROC_BERE SEGMENT WORD PULIC
EXTRN BAD_DIV: FAR                  ; let assembler know
                                    ; procedure of BAD DIV is
                                    ; in other assembly module

CODE HERE SEGMENT WORD PUBLIC
ASSUME CS: CODE_HERE, DS: DATA HERE, SS:STACK_HERE
START:
MOV AX, STACK_HERE          ; initialize stack segment register
MOV SS, AX
MOV SP, OFFSET TOP_STACK    ; initialize stack pointer.
MOV AX, DATA HERE           ; initialize data segment register
MOV DS, AX
; store the address for the BAD DIV routine at address 0000: 0000
; address 0000-0003 is where type 0 interrupt gets interrupt
; service procedure address, CS at 0002 and 0003, IP at 0000 and 0001
```

```
MOV AX, 0000
MOV ES, AX
MOV WORD PTR ES:0002, SEG BAD_DIV
MOV WORD FTR ES :0000, OFFSET BAD_DIV
MOV SI, OFFSET INPUT_VALUES
; initialize pointer for input values
 MOV BX, OFFSET SCAT ED_VALUES
                                ; point BX at start of result array
MOV CX, 0004h                   ; initialize data value
                                ; to AX for divide
NEXT: MOV AX, [SI]              ; divide by Scale factor
DIV SCALE_FACTOR                ; Bring a value to AX for divide
CMP BAD DIV_FLAG, 0lh           ; check if divide produced
                                ; invalid result
JNE OK                          ; NO: go save scaled value
MOV BYTE PTR[BX], 00            ; YES: load a 0 as scaled value
JMP SKIP
OK:
MOV [BX], AL                    ; save scaled value
SKIP:
MOV BAD_DIV FLAG, 0             ; reset BAD_DIV FLAG
                                ; before doing next

ADD SI, 02h                     ; point at location of
                                ; next input value
INC BX                          ; point at location for next result
LOOP NEXT                       ; repeat until all values done
STOP:
NOP
CODE HERE: ENDS

END START
```

```
; service divide by zero interrupt
DATA_HERE SEGMENT WORD PUBLIC
EXTRN BAD DIV_FLAG: BYTE   ; let assembler know
                           ; BAD DIV FLAG
DATA_HERE ENDS             ; is in another assembly module

PUBLIC BAD_Div             ; make procedure BAD DIV
                           ; available to other
                           ; assembly modules

INT_PROC_HERE SEGMENT WORD PUBLIC
                           ; set up a segment for all interrupt
                           ; service procedures
BAD_DIV PROC FAR           ; procedure for type 0 interrupt
ASSUME CS: 1NT_PROC_HERE, DS: DATA_HERE
PUSH AX                    ; save AX of interrupted program
PUSH DS                    ; save DS of interrupted program
MOV AX, DATA_HERE          ; load DS value needed here
MOV DS, AX
MOV BAD ...DIV_FLAG, 01    ; set LSB of BAD_DIV FLAG
                           ; byte
POP DS                     ; restore DS of
                           ; interrupted program
POP AX                     ; restore AX of
                           ; interrupted program
IRET                       ; return to next instruction in
                           ; interrupted program
BAD DIV ENDP
INT_PROC_HERE ENDS
END
```

كتب البرنامج الرئيسي في ملف مستقل عن الملف الذي كتبت فيه إجرائية خدمة المقاطعة. وهذا ما يسوّغ استخدام التوجيهين EXTRN و PUBLIC.
نصرح في بداية البرنامج الرئيسي بوجود قطاع يسمى Data_Here يحوى المعطيات التي سيعالجها البرنامج. ويطلب التوجيه WORD من المجمع بدء هذا القطاع في أول عنوان زوجي متاح.
أما التوجيه PUBLIC فإنه يعرف القطاع عمومياً، أي يمكن لوحدات برمجية مكتوبة في ملفات خارجية الوصول إليه. ولما كانت كلمات الدخل قيماً مرمزة على 16 بت، فإننا نستخدم التوجيه DW لتصريح هذه القيم. أما قيم النتائج فهي بايتات، ولذا نستخدم التوجيه DB لحجز أربعة مواقع ذاكرة لها. إضافة إلى ذلك، تشحن تلك المواقع الأربعة بقيمة الصفر بواسطة التوجيه DUP(0).

أما العبارة Scale_Factor DB 09h فهي تحجز بايت خاص لتخزين العدد الذي سيقسم كافة قيم الدخل. وسبب استخدام DB بدلاً من EQU في تعريف المتحول Scale_Factor هو أن القيمة المحددة بـ DB تكون موجودة في الذاكرة الحية RAM، ولذا يمكن تغييرها ديناميكياً في البرنامج. أما التوجيه EQU فإنه يعرف قيمة لا يمكن تغييرها إلا بإعادة تجميع ا لبرنامج من جديد.

ثم يقوم ا لبرنامج الرئيسي بتعريف مكدس لتخزين عنوان العودة والمعاملات الممررة من/إلى الإجرائية. ولذا،عرف القطاع Stack_Here بوضع مؤشر على قمة المكدس بالعبارة
Top_Stack Label WORD. تستخدم هذه اللصاقة عند الاستهلال ليشير مؤشر المكدس نحو الموقع الذي يلي قمة المكدس مباشرة.

يلي ذلك تصريح راية الخطأ BAD_DIV_Flag. والمعرفة بالتوجيه PUBLIC، لكي تستطيع الوحدات البرمجية المكتوبة في ملفات خارجية من الوصول إليه. و من أجل ذلك يجب أن تعرف هذه الراية في تلك الوحدات باستخدام التوجيه EXTERN، للدلالة على أن هذا المتحول موجود في ملف آخر. وبكلمات أخرى، يسمح التوجيه PUBLIC بتصدير أسماء اللصاقة نحو الوحدات البرمجية الخارجية، في حين يسمح التوجيه EXTERN باستيرادها.

توجه العبارتان INT_PROC_here SEGMENT WORD WORD PUBLIC و INT_PROC_here ENDS المجمع لتعريف الإجرائية BAD_DIV في قطاع اسمه INT_PROC_here.
يُعرف بعد ذلك قطاع البرنامج بالعبارة CODE_HERE SEMENT WORD PUBLIC، وتشير الكلمة PUBLIC إلى أن القطاع يمكن جمعه مع قطاعات لها الاسم ذاته وموجودة في وحدات برمجية خارجية.
ويفيد التوجيه ASSUME في تحديد القطاعات المنطقية المستخدمة، كقطاع البرنامج وقطاع المعطيات والمكدس. ثم تأتي تعليمات الاستهلال التقليدية لسجل قطاع المكدس SS ومؤشر المكدس SP وسجل قطاع

المعطيات DS. تشحن التعليمات الأربع التالية عنوان الإجرائية BAD_DIV في الموقع المناسب في جدول متجهات المقاطعة، إذ نستخدم هنا المقاطعة من النوع 0.
يجري بعد ذلك وضع قيمة ابتدائية مناسبة قي السجل SI ليؤشر على بداية قيم الدخل، والسجل BX ليؤشر نحو بداية جدول النتائج، ويستخدم أيضاً السجل CX كعداد لقيم الدخل المعالجة، فعندما يصبح CX=0 فهذا يعنى أن جميع القيم قد قسمت.
ثم تبدأ عمليات المعالجة، إذ تقرأ كلمة دخل، وتوضع فى السجل AX، ثم تقسم على القيمة Scale_Factor. وتصبح نتيجة القسمة موجودة في السجل AL، فإذا كانت النتيجة كبيرة جداً بحيث يتعذر تخزينها في ذلك السجل، تولدت مقاطعة من النوع 0 ، وهذا ما يدفع المعالج إلى القفز نحو إجراء خدمة المقاطعة، بعد الحصول على عنوانه من جدول متجهات المقاطعة (المواقع من 0 إلى 3). ومن ثم، فالمعالج يقفز نحو الإجرائية Bad_Div.
تبدأ الإجرائية Bad_Div بتوجيه للمجمع بحيث يعرف أن اللصاقة Bad_Div_Flag مثل متحولاً على بايت واحد، وأنه معرف في قطاع خارجي يسمى Data_Here. ثم نعرف الإجرائية Bad_Div إجرائية عمومية Public بحيث يمكن وحدات برمجية خارجية الوصول إليها. يلي ذلك تعريف قطاع منطقي للمعطيات نسميه IN_Proc_here. نلاحظ أنه بالإمكان كتابة إجرائية المقاطعة في القطاع المنطقي للبرنامج الرئيسي ذاته، ولكن الفصل بينهما يجعل البرنامج الكلي أكثر منهجية.

تعرّف العبارة Bad_Div Proc FAR البداية الفعلية لإجرائية المقاطعة، وتوجه المجمع لكي يخزن قيم السجلين CS وIP عند القفز لأن الإجراء من النوع البعيد. كما تحدد العبارة ASSUME بداية الإجرائية وتخبر المجمع بأسماء القطاعات الواجب استخدامها للبرنامج والمعطيات.

تبدأ بعدئذٍ الإجرائية بحفظ السجلات التي ستُستخدم داخله. وينبغي تطبيق هذه العملية في جميع إجرائيات المقاطعة. تُحفظ السجلات في المكدس، لتُسترجع عند نهاية تنفيذ الإجرائية قبل تعليمة العودة IRET. وفي مثالنا، نحفظ السجلين DS وAX.

ونوجه الانتباه إلى أهمية تخزين DS، إذ من الممكن أن تحدث المقاطعة، وبعدئذٍ نرفع راية الخطأ بكتابة القيمة 1 داخل الموقع BAD_DIV_FLAG. ونلاحظ أن الوصول إلى هذا المتحول أصبح ممكناً لأنه قد عرف في الملف الحالي كمتحول خارجي، وفي البرنامج الرئيسي كمتحول عمومي.
تسترجع الإجرائية، بعد رفع الراية، قيمة السجلين DS وCS، ثم تعود إلى البرنامج الرئيسي بالتعليمة IRET. تختلف هذه التعليمة عن التعليمة RET (وهي تعليمة العودة من الإجرائية)، بحيث تسترجع تلقائياً سجل الرايات المخزّن في المكدس.
لنعد الآن إلى البرنامج الرئيسي، فبعد التعليمة DIV تُفحص الراية Bad_Div_FLAG فإذا لم تكن مرفوعة، أي قيمتها معدومة، فهذا يدل على عدم حدوث المقاطعة، ومن ثم فعملية القسمة صالحة. نخزن إذاً ناتج القسمة في

الموقع الذي يؤشر عليه السجل BX. أما في الحالة المعاكسة، أي عند حدوث مقاطعة، فإن المعالج يقفز إلى اللصاقة SKIP مباشرةً دون أن يخزن ناتج القسمة في جدول النتائج.

وفي كلتا الحالتين (حالة التخزين أو عدم التخزين) يمحو البرنامج راية الخطأ استعداداً لعملية القسمة التالية، ويزيد مؤشر الدخل SI بمقدار 2، ومؤشر الخرج بمقدر 1، ثم يقفز إلى اللصاقة NEXT بواسطة التعليمة LOOP، التي تفحص ذاتياً السجل CX قبل القفز.

3. أنواع المقاطعات:

نعرض هنا بالتفصيل الطرائق المختلفة التي يتقاطع فيها المعالج 8086، وطرائق استجابة المعالج لها. وسيشمل العرض كافة أنواع القاطعات، ولذا ليس من الضروري معرفة تفاصيل كل نوع من الأنواع من القراءة الأولى للنص، ولكن يمكن العودة إلى الفقرة المناسبة حين اللزوم.

هناك عدة أنواع للمقاطعات في المعالج 8086

المقاطعات المخصصة:

- النوع 0: القسمة على 0
- النوع 1: مقاطعة التنفيذ الخطوي
- النوع 2: المقاطعة الغير قابلة للحجب
- النوع 3: مقاطعة نقطة التوقف
- النوع 4: مقاطعة الفائض

المقاطعات البرمجية من 32 حتى 255.

سنشرح كل نوع بشكل مفصل في الفقرات القادمة.

1.3. النوع 0: القسمة على 0:

يتقاطع المعالج ذاتياً بهذا النوع عندما يتعرض لعملية قسمة على القيمة 0، أو عندما يكون ناتج القسمة كبيراً إلى حد يحول دون ترميزه في سجل الوجهة. ففي حالة هذا النوع، يدفع المعالج سجل الراية إلى المكدس، ويضع القيمة 0 في الرايتين IF وTF، ويدفع عنوان العودة إلى المكدس، ثم يجلب عنوان البداية لإجرائية خدمة المقاطعة من جدول متجهات المقاطعة. فيشحن في السجل CS محتوى الموقعين 02h و03h وفي السجل IP محتوى الموقعين 00h و01h.

ولما كانت الاستجابة لهذه المقاطعة آلية ولا يمكن إلغاؤها، فيجب توقعها عند استخدام التعليمتين DIV أو IDIV في البرنامج الرئيسي. ويمكن الحيلولة دون حدوث هذه المقاطعة بفحص المقسوم عليه والتوثق أن قيمته تختلف عن 0 قبل القسمة.

يمكن لمعالجة هذا النوع من المقاطعات كتابة إجرائية خدمة لهذه المقاطعة كما في المثال السابق. ويتميز هذا الحل بمتانته، إذ ليس من الضروري اختيار المقسوم عليه في كل مرة. فعند حدوث خطأ ما، يقفز المعالج إلى

إجرائية خدمة المقاطعة للقيام بما يلزم. ويجب ألا ننسى، عند استخدام أي مقاطعة، كتابة عنوان البداية لإجرائية الخدمة في جدول متجهات المقاطعة.

2.3. النوع 1: مقاطعة التنفيذ الخطوي:

نحتاج أثناء تطوير برنامج ما إلى تنفيذه تعليمة فتعليمة، لمراقبته وتحقق صحة أدائه. إذ نفحص بعد كل تعليمة محتوى السجلات ووضع الرايات، وإذا كانت القيم سليمة نطلب منه المتابعة. وبكلمات أخرى، يتوقف البرنامج في نمط التنفيذ الخطوي بعد كل تعليمة وينتظر ايعازات جديدة من المستثمر. يسمح النوع 1 من المقاطعات Trap بتحقيق هذا التنفيذ الخطوي Single-Step Interrupt.

فعند تأهيل الراية TF يتعرض المعالج آلياً لمقاطعة من النوع 1 بعد كل تعليمة ينفذها. وعند ظهور هذه المقاطعة. يدفع المعالج سجل الرايات إلى المكدس، ويضع القيمة صفر في الرايتين TF وIF، ثم يدفع محتوى السجلين CS وIP إلى المكدس لتخزين عنوان العودة. يجلب المعالج بعدئذٍ عنوان البداية لإجرائية خدمة المقاطعة من العنوانين 06h و07h ويشحنهما في السجل CS، كما يجلب العنوانين 04h و05h ويضعهما في السجل IP.

ويجب لاستخدام هذا النوع من المقاطعة تحقيق مايلي:

- وضع راية التنفيذ الخطوي TF على 1
- كتابة إجرائية خدمة المقاطعة المناسب، الذي يخزن كافة السجلات المستخدمة في الإجرائية
- شحن عنوان البداية في جدول متجهات المقاطعة بدءاً من العنوان 04h وحتى 07h

أما محتوى الإجرائية فيختلف بحسب المهمة المطلوبة بعد التنفيذ الخطوي. فمثلاً، قد تتضمن الإجرائية إرسال قيمة السجلات إلى وحدة الإظهار ليفحصها المستثمر.

يمكن تأهيل راية التنفيذ الخطوي بإتباع التسلسل التالي:

```
PUSH    F                   ; pus flags on stack
MOV     BP, SP              ; copy SP to BP for use as index
OR      [BP+0], 0100h       ; set TF bit
POPF                        ; restore FLAG
```

ولإلغاء هذه الراية تستبدل بالتعليمة OR التعليمة AND [BP+0], FEFFh.

3.3. النوع 2: المقاطعة غير القابلة للحجب:

يتعرض المعالج 8086 لمقاطعة من النوع 2، عند ظهور جبهة هابطة على مدخله NMI (ويُقصد بجبهة هابطة انتقال مستوى الإشارة المطبقة علي المدخل NMI من المستوى المرتفع إلى المستوى المنخفض). واستجابة لهذه المقاطعة، يدفع المعالج سجل الراية إلى المكدس، ويلغي عمل الرايتين TF وIF، كما يدفع محتوى السجلين CS وIP إلى المكدس لتخزين عنوان العودة. ثم يجلب عنوان البداية لإجرائية خدمة المقاطعة من جدول متجهات المقاطعة، فيشحن في السجل IP محتوى الموقعين 08h و09h، وفي السجل CS محتوى الموقعين 0Ah و0Bh.

توصف هذه المقاطعة بأنها غير قابلة للحجب، لأن المعالج لا يستطيع إلغاءها برمجياً. فعند ظهور الجبهة الهابطة على المدخل NMI، يقفز المعالج حتماً إلى إجرائية خدمة المقاطعة المرافقة.

تفيد هذه المقاطعة في إعلام المعالج بحدث خارجي. فقد يُوصل إلى ذلك المدخل محس الضغط لغلاية مثلاً. فإذا ارتفع الضغط فوق حد معين، فإن المحس يرسل إشارة مقاطعة إلى المعالج. وفي هذه الحالة. تستطيع إجرائية خدمة المقاطعة إعطاء أمر لإطفاء الغلاية، وفتح صمام الضغط، وتشغيل الإنذار. وثمة استخدام شائع لهذه المقاطعة، وهو تخزين معطيات البرنامج عند حدوث خلل في التغذية. إذ تسمح دارات خارجية بتحسس خلل التغذية، ونرسل إشارة المقاطعة NMI إلي المعالج. ولما كان الزمن اللازم لانقطاع التيار كبيراً بالمقارنة بزمن تنفيذ التعليمات، فإن المعالج يستطيع خلال هذا الزمن. وبعد وصول المقاطعة، تخزين المعطيات المهمة في أي وسيطة من وسائط التخزين. وعند عودة التيار، تسترجع أولاً المعطيات المخزنة، ويتابع المعالج العمل دون ضياع لأي معلومة.

4.3. النوع 3: مقاطعة نقطة التوقف:

تتولد مقاطعة نقطة التوقف Break Point بتنفيذ التعليمة INT3. إن الاستخدام الرئيسي لهذه المقاطعة هو إضافة نقطة توقف إلى البرنامج. ويُقصد بذلك توقف المعالج عن تنفيذ البرنامج عند وصوله إلى تعليمة معينة، وهذا ما يسمح بفحص محتوى سجلاته، والتوثق من صحة تنفيذ البرنامج.

وتختلف نقطة التوقف عن التنفيذ الخطوي، في أن المعالج قد ينفذ عدة تعليمات قبل الوصول إلى نقطة التوقف. أما في حالة التنفيذ الخطوي فالمعالج يتوقف حتماً بعد كل تعليمة.

لذا، عند وضع نقطة التوقف في برنامج ما تضاف التعليمة INT في تلك النقطة، فيقوم المعالج عند تنفيذ هذه التعليمة بدفع سجل الراية إلي المكدس، ويضع القيمة 0 في الرايتين TF وIF، كما يدفع السجلين CS وIP إلى المكدس لتخزين عنوان العودة. يجلب المعالج بعد ذلك عنوان البداية لإجرائية خدمة المقاطعة من جدول متجهات المقاطعة من الموقع 0Ch وحتى 0Fh. أما الإجرائية ذاتها فتخزن كافة السجلات في المكدس. وتختلف المهمة التي تقوم بها تبعاً للنظام. فقد تقوم وحدة إظهار مثلاً بإخراج السجلات على الشاشة حتى يستطيع لمستثمر تحقق برنامجه.

5.3. النوع 4: مقاطعة الفائض:

ترفع راية الفائض OF إذا تعذر تمثيل ناتج عملية حسابية معينة في سجل الوجهة أو في موقع من الذاكرة. فعلى سبيل المثال، إذا أضفنا ا لعدد 6Ch (108dec) على 8 بتات بصيغة المتمم الثنائية إلى العدد 51h (81dec)، فإن النتيجة ستكون BDh (189dec). إن هذه النتيجة صحيحة إذا كان الجمع لأعداد بلا إشارة. أما في حال الأعداد ذات الإشارة، تفهم النتيجة على أنها مساوية (67dec−)، وهذا خطأ.

توجد للحيلولة دون وقوع هذه الأخطاء طريقتان رئيسيتان. الأولى تعتمد على القفز عند الفائض (تعليمة JO) وتستخدم بعد التعليمة الحسابية مباشرة. فإذا كانت راية الفائض مرفوعة بسبب العملية الحسابية، فسيقفز المعالج إلى العنوان المحدد بالتعليمة JO، وفي ذلك العنوان يعالج الخطأ على الوجه المناسب. أما الطريقة الثانية فتعتمد على اكتشاف خطا الفائض بالمقاطعة. ويجري ذلك بتعليمة INTO التي توضع مباشرة بعد التعليمة الحسابية في البرنامج. فإذا لم ترفع راية الفائض، فإن التعليمة ستتصرف كتعليمة NOP. وإلا. فإنها ستولد مقاطعة من النوع 4. واستجابة لهذه المقاطعة، يدفع المعالج سجل الراية إلى المكدس، ويضع القيمة 0 في الرايتين TF وIF، ويدفع السجلين CS وIP إلي المكدس. ثم يجلب المعالج عنوان البداية لإجرائية خدمة المقاطعة من جدول متجهات المقاطعة من المواقع 010h وحتى 013h.

يُعالج خطأ الفائض داخل إجرائية المقاطعة، فقد توضع قيمة تدل على الخطأ في موقع ما من الذاكرة، كما فعلنا في المثال BAD_DIV. وتمتاز الطريقة الثانية عن الأولى بأنها يمكن الوصول إليها من أي مكان في البرنامج.

6.3. المقاطعات البرمجية 0 إلى 255 :

يمكن أن تستخدم تعليمة INT لتوليد مقاطعة من أي نوع من الأنواع الممكنة، ويُحدد النوع في التعليمة ذاتها. فمثلاً، تؤدي التعليمة INT 32 إلى توليد مقاطعة من النوع 32. وعند حدوث مثل هذه المقاطعات البرمجية، يدفع المعالج سجل الراية إلى المكدس، فيضع القيمة 0 في الرايتين TF وIF ويخزن السجلين CS وIP في المكدس للحفاظ على عنوان العودة. يجلب المعالج بعد ذلك عنوان البداية لإجرائية خدمة المقاطعة بحسب النوع المحدد في التعليمة. فمثلاً، يجلب المعالج عنوان إجرائية المقاطعة من النوع 32 من الموقع 32x4=128 (80h)، ويشحنه في السجل IP. كما يضع في السجل CS محتوى العنوان 82h و83h.

تفيد المقاطعات البرمجية في اختبار إجرائيات خدمة المقاطعة. فعلى سبيل المثال، يمكن استخدام INT 0 لتوليد مقاطعة برمجية من النوع 0 (القسمة على 0)، دون الحاجة إلى تنفيذ برنامج القسمة. كما يمكن استخدام التعليمة INT 2 لتنفيذ إجرائية المقاطعة المرتبطة بالدخل NMI دون ظهور إشارة مقاطعة خارجية.

وثمة فائدة مهمة أيضاً للمقاطعات البرمجية، وهي استدعاء الإجرائيات المطلوبة من أي برنامج. ففي الحاسوب الشخصي PC، توجد مجموعة من الإجرائيات المضمنة في الذاكرة ROM، بحيث تؤدي كل إجرائية منها عملاً معيناً، مثل قراءة محرف من لوحة المفاتيح أو كتابة بعض المحارف على الشاشة أو قراءة محارف من القرص.

تسمى مجموعة الإجرائيات هذه إجرائيات الدخل والخرج الأساسية BIOS (Basic Input Output Subroutines).

تستدعى الإجرائيات BIOS بواسطة تعليمة INT. فمثلاً، إذا أردنا إرسال بعض المحارف إلى الطابعة في حاسوب شخصي، فبالإمكان استخدام إجرائية BIOS مخصصة لذلك وهي التعليمة INT 17.

أسئلة الفصل السابع

1. ما هي أنواع المقاطعات التي يمكن أن تقاطع عمل المعالج 8086؟

1. مقاطعات الكيان الصلب: إشارة خارجية تطبق على المربط NMI (وهي مقاطعة غير قابلة للحجب)، أو على المربط INTR (وهي مدخل مقاطعة قابلة للحجب).
2. المقاطعة المبرمجة: تنفيذ تعليمة المقاطعة INT.
3. تحقق شرط معين نتيجة تنفيذ تعليمة ما في المعالج وهي عادة تدل على خطأ داخلي. ومثال ذلك، المقاطعة الناتجة عن التقسيم على صفر.

2. كيف يستجيب المعالج 8086 لحدوث مقاطعة؟

1. ينقص مؤشر المكدس بمقدار 2، ويدفع بسجل الرايات إلى المكدس.
2. يلغي تأهيل المدخل INTR بمحو راية المقاطعة في سجل الرايات.
3. يضع صفراً في راية التنفيذ الخطوي TRAP المحتواة في سجل الرايات.
4. ينقص مؤشر المكدس بمقدار 2، ويدفع محتوى سجل قطاع البرنامج الحالي إلى المكدس.
5. ينقص مؤشر المكدس مرة ثانية بمقدار 2، ويدفع محتوى مؤشر التعليمات الحالي إلى المكدس.
6. يجري المعالج قفزاً غير مباشر إلى بداية الإجرائية المكتوبة لخدمة المقاطعة.
7. عند وصول المعالج إلى تعليمة IRET يعود البرنامج إلى المكان الذي كان فيه قبل حدوث المقاطعة باسترجاع قيمة قطاع البرنامج ومؤشر التعليمات من المكدس.

3. ما هي الفائدة من مقاطعة التنفيذ الخطوي TRAP في المعالج 8086؟

نحتاج أثناء تطوير برنامج ما إلى تنفيذه تعليمة فتعليمة، لمراقبته وتحقق صحة أدائه. إذ نفحص بعد كل تعليمة محتوى السجلات ووضع الرايات، وإذا كانت القيم سليمة نطلب منه المتابعة. وبكلمات أخرى، يتوقف البرنامج في نمط التنفيذ الخطوي بعد كل تعليمة وينتظر ايعازات جديدة من المستثمر.

4. لماذا نسمي المقاطعة NMI في المعالج 8086 بأنها المقاطعة الغيرقابلة للحجب؟

لأن المعالج لا يستطيع إلغاءها برمجياً. فعند ظهور الجبهة الهابطة على المدخل NMI، يقفز المعالج حتماً إلى إجرائية خدمة المقاطعة المرافقة.

5. مالفرق بين مقاطعة نقطة التوقف ومقاطعة التنفيذ الخطوي؟

تختلف نقطة التوقف عن التنفيذ الخطوي، في أن المعالج يتوقف فقط عند الوصول إلى تعليمة INT 3، أما في حالة التنفيذ الخطوي فالمعالج يتوقف حتماً بعد كل تعليمة.

6. كيف يتم إضافة نقطة توقف في البرنامج بهدف اختباره؟

تضاف التعليمة INT 3 في تلك النقطة.

7. من المعلوم أنه لا يمكن النفاذ بشكل مباشر إلى سجل الحالة في المعالج 8086، اكتب التعليمات التي يمكن من خلالها تفعيل أو إلغاء تفعيل بت المقاطعة IF في سجل الحالة؟

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

سجل الحالة في المعالج 8086.

يمكن ذلك بالطريقة التالية:

```
PUSHF                       ; push flags on stack
MOV     BP, SP              ; copy SP to BP for use as index
OR      [BP+0], 0100h       ; set TF bit
POPF                        ; restore flags
```

ولإلغاء هذه الراية تستبدل بالتعليمة OR التعليمة AND [BP+0], FEFFh.

8. كيف يتم توجيه المقاطعة من النمط 32 في المعالج 8086 في جدول متجهات المقاطعة؟

يتم ذلك عن طريق كتابة 4 بايتات في ابتداءً من الموقع 32x4=128 تضم قيمة مؤشر القطاع CS أولاُ ومن ثم مؤشر التعليمات IP الموافقين لموقع إجرائية المقاطعة.

أسئلة خيارات متعددة

1. في النظم الحاسوبية التي تستخدم المعالج 8086 يتم حجز اذاكرة من الموقع 0 إلى موقع 3FFh من أجل:
   A. كتابة إجرائيات معالجة المقاطعات.
   B. كتابة تعليمات استهلال البرنامج الأساسي.
   C. تخزين جدول متجهات المقاطعة.
   D. غير مستخدمة ومحجوزة لأغراض التطوير اللاحق للمعالج.

2. في جدول متجهات المقاطعة للمعالج 8086 يتم حجز عدد من البايتات لكل نوع من المقاطعات يبلغ:
   A. 2 بايت
   B. 4 بايت
   C. 6 بايت
   D. 8 بايت

3. يبلغ عدد أنماط المقاطعة المتاحة للمبرمج في المعالج 8086:
   A. 5
   B. 27
   C. 224
   D. 256

4. تحدث المقاطعة من النوع 0 في المعالج 8086 :
   A. المقاطعة الخارجية الغير قابلة للحجب NMI.
   B. المقاطعة الخارجية القابلة للحجب INTR.
   C. التنفيذ الخطوي.
   D. القسمة على صفر.

5. تستخدم الراية IF في سجل الحالة للمعالج 8086 بهدف:
   A. تفعيل المقاطعة NMI.
   B. تفعيل المقاطعة الخارجية INTR.
   C. تفعيل المقاطعات البرمجية
   D. تفعيل مقاطعة التنفيذ الخطوي.

6. تستخدم الراية TF في سجل الحالة للمعالج 8086 بهدف:

A. تفعيل المقاطعة NMI.

B. تفعيل المقاطعة الخارجية INTR.

C. تفعيل المقاطعات البرمجية

D. تفعيل مقاطعة التنفيذ الخطوي.

7. تتم معالجة خطأ الفائض بعد أي عملية حسابية عن طريق:

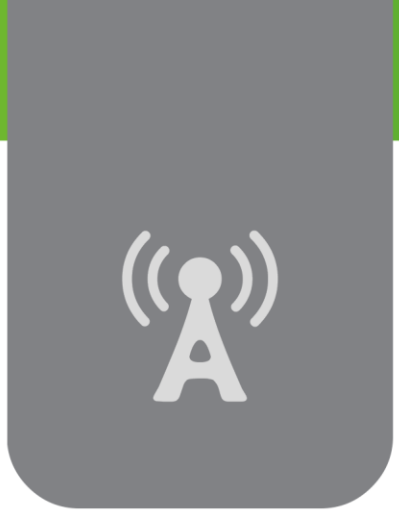
A. استخدام تعليمة القفز الشرطية JO نحو إجرائية معالجة مقاطعة الفائض.

B. استخدام التعليمة INTO لاستدعاء إجرائية مقاطعة الفائض.

C. إحدى الطريقتين في (أ) أو (ب).

D. طريقة أخرى غير مذكورة.

رقم التمرين	الإجابة الصحيحة
1	C
2	B
3	C
4	D
5	B
6	D
7	C



الفصل الثامن: المتحكم الصغري 8051

الكلمات المفتاحية:

المتحكمات الصغرية، المتحكم 8051، ذاكرة البرنامج، ذاكرة المعطيات، الذاكرة القابلة للعنونة بالبت، سجلات العمل، سجلات الوظائف الخاصة، بوابات المتحكم.

ملخص:

يهدف هذا الفصل إلى توضيح مفهوم المتحكم الصغري وأوجه الشبه والاختلاف بينه وبين المعالج الصغري. كما يهدف إلى شرح بنية المتحكم الصغري الداخلية وذلك من خلال دراسة البنية الداخلية للمتحكم 8051 من شركة Intel. حيث نتطرق إلى تنظيم الذاكرة والسجلات الداخلية في هذا المتحكم. كما نبين آلية عمله والطرفيات المدمجة في بنيته الداخلية ووظائف مرابطه الخارجية.

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

- الفرق بين المتحكمات والمعالجات
- البنية الداخلية للمتحكم 8051
- تنظيم الذاكرة في المتحكم
- بوابات الدخل والخرج في المتحكم

الفصل الثامن: المتحكم الصغري 8051

يهدف هذا الفصل إلى توضيح مفهوم المتحكم الصغري وأوجه الشبه والاختلاف بينه وبين المعالج الصغري. كما يهدف إلى شرح بنية المتحكم الصغري الداخلية وذلك من خلال دراسة البنية الداخلية للمتحكم 8051 من شركة Intel. حيث نتطرق إلى تنظيم الذاكرة والسجلات الداخلية في هذا المتحكم. كما نبين آلية عمله والطرفيات المدمجة في بنيته الداخلية ووظائف مرابطه الخارجية.

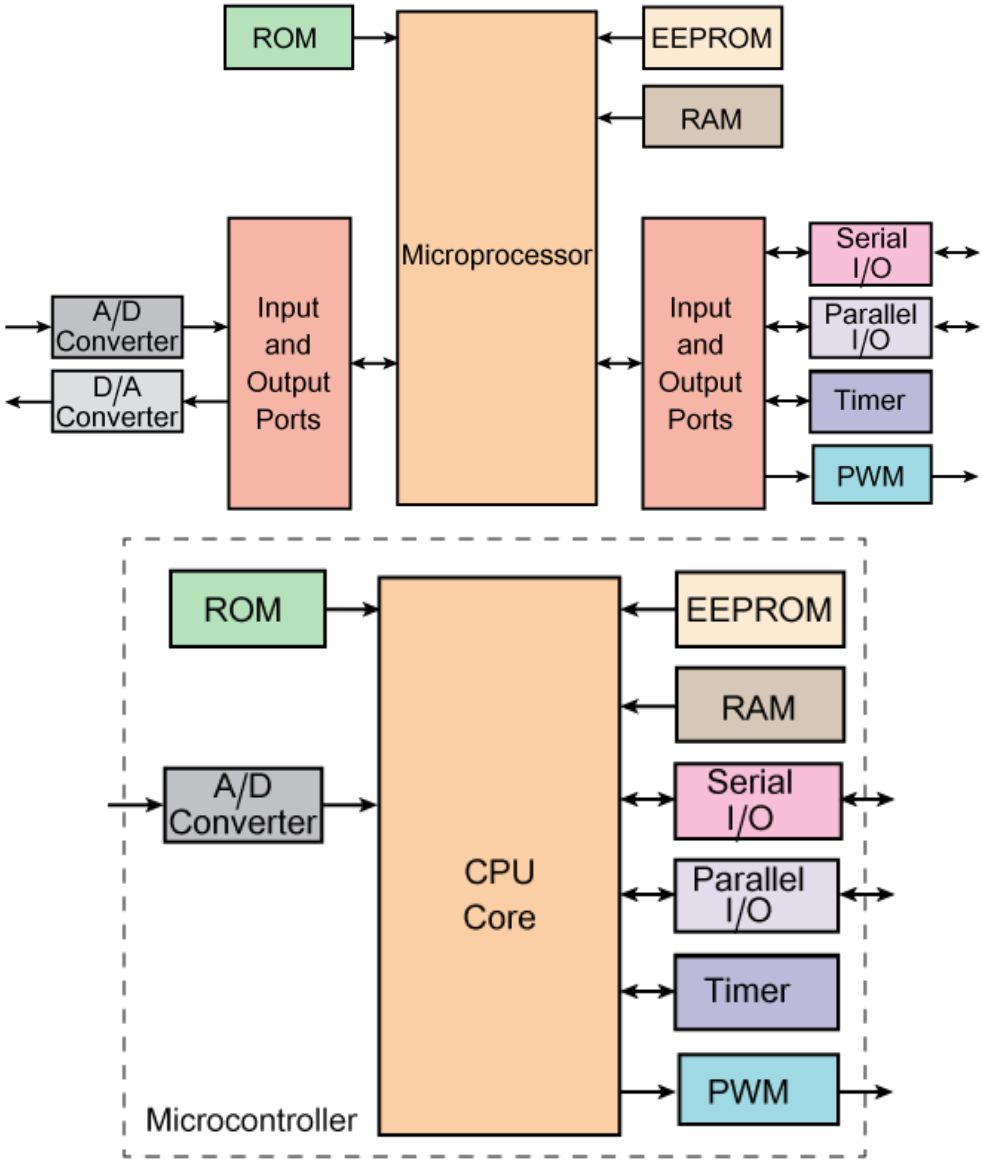
1. مقدمه:

تُستخدم المتحكمات الصغرية اليوم في الكثير من التجهيزات الحديثة مثل أجهزة التحكم ولوحة المفاتيح والطابعات وآلات الغسيل وغيرها. ويمثل المتحكم الصغري نظام معالجة حاسوبي مصغر على دارة تكاملية واحدة، حيث يتم استخدامه في تصميم الأجهزة المختلفة بغرض الاستفادة من الإمكانيات البرمجية للنظم الحاسوبية في تسهيل التصميم وإضافة بعض المزايا "الذكية" في عمل الأجهزة. ولكن ما الفرق بينه وبين المعالجات الصغرية؟ هذا ما سنجيب عنه في الفقرة التالية.

2. المتحكمات والمعالجات الصغرية:

سبق وأن تعرفنا على بنية المعالج الصغري وقلنا بأنه يحتوي بشكل أساسي على وحدة حساب مركزية مع وحدة تحكم وسجلات داخلية وذلك من أجل تنفيذ برنامج موجود ضمن ذاكرة خارجية. ويحتاج المعالج الصغري للعديد من العناصر والمكونات الخارجية الضرورية لعمله وربطه مع العالم الخارجي مثل الذواكر وبوابات الدخل والخرج ودارة التحكم بالمقاطعات ومؤقتات وغيرها. أما في المتحكمات الصغرية فقد تم إضافة كل العناصر الأساسية لعمل وحدة الحساب المركزية في نفس الدارة المتكاملة لكي يستطيع المتحكم العمل بمفرده كنظام معالجة حاسوبي متكامل من دون الحاجة إلى دارات خارجية.

يوضح الشكل التالي الفرق بين المتحكم الصغري والمعالج الصغري.



الفرق بين المتحكم الصغري والمعالج الصغري

حيث نلاحظ في النظام الحاسوبي الذي يستخدم معالجاً صغرياً (القسم لعلوي من الشكل 1) بأن ذاكرة البرنامج (وهي عادة من نوع ROM أو EEPROM) أو ذاكرة المعطيات (من نوع RAM) موجودة خارج المعالج ومربوطة معه من خلال مسرى خارجي. كما أن جميع الطرفيات الضرورية، مثل المؤقت ووحدة العبور التفرعية (Parallel I/O) وحدة العبور التسلسلية (Serial I/O) والمبدلات التمثيلية الرقمية (A/D, D/A converters)، ترتبط كلها مع المعالج عبر المسرى من خلال بوابات عبور. وبالتالي فإن عملية تشغيل البرنامج وعمليات النفاذ تتم عبر المسرى الخارجي للمعالج. بينما في المتحكم الصغري (القسم السفلي من الشكل 1) فقد تم دمج جميع الذواكر والطرفيات داخل الدارة التكاملية للمتحكم وموصولة مع وحدة الحساب المركزية عن طريق مسرى داخلي. وذلك مما يجعل زمن النفاذ إلى هذه الطرفيات أقل ويساهم في تقليل الكلفة المادية للنظام الحاسوبي وتصغير حجمه.

كل هذا يجعلنا ننظر إلى المتحكم الصغري وكأنه نظام حاسوبي متكامل ولكن حسب مواصفات محددة من قبل المصنع. فمثلاً، نستطيع عند تصميم نظام حاسوبي اعتماداً على المعالج الصغري أن نختار بأنفسنا حجم الذاكرة المناسب للتطبيق، بينما في المتحكم الصغري فإن الذاكرة الداخلية ذات حجم معين ولا يمكن تغييره ولذلك لا بد من اختيار المتحكم المناسب الذي يحتوي على ذاكرة كافية للتطبيق الذي نرغب به (وهذا هو الحل المتبع في أغلب الأحيان) أو نقوم بتوسيع هذه الذاكرة من خلال إضافة ذاكرة خارجية للمتحكم، وهذا ما تسمح به معظم المتحكمات، ولكن يأتي ذلك على حساب زيادة تعقيد النظام. ولا يتم اللجوء إلى هذا الحل إلا في حالة التطبيقات التي يكون فيها حجم البرنامج كبيراً جداً.

ولهذا فإن الأنواع المختلفة للمتحكمات الصغرية لا تختلف فقط بالبنية الداخلية ومدى سرعتها في تنفيذ التعليمات بل وتختلف أيضاً في حجم الذواكر الداخلية وعدد ونوعية الطرفيات المدمجة في داخلها. إن عملية الاختيار المناسب للمتحكم يعتمد على نوعية التطبيق ومتطلباته من حيث السرعة والذواكر والطرفيات اللازمة بالإضافة إلى الاعتبارات الأخرى المتعلقة بسعره ومدى توفره وتوفر الأدوات البرمجية الضرورية لتطوير البرنامج.

لكي نفهم بنية المتحكم الداخلية وآلية عمله، سوف نقوم بدراسة المتحكم 8051 من شركة إنتل كمثال عن المتحكمات. حيث يتميز هذا المتحكم ببساطته كما أنه لا يحتوي على الكثير من الطرفيات بالمقارنة مع المتحكمات الحديثة. إلا أن فهم بنية وعمل هذا المتحكم يمهد الطريق في فهم باقي المتحكمات وذلك بسبب التشابه الكبير في بنية وآلية عمل معظم المتحكمات.

3. البنية الداخلية للمتحكم 8051:

يعتبر هذا المتحكم واحداً من عائلة المتحكمات MCS-51 وهو من المتحكمات التي لاقت رواجاً واسعاً في السوق العالمية كمتحكم على 8 بتات. تم تصنيعه لأول مرة في عام 1980 من قبل شركة Intel ولكن تم تصنيعه فيما بعد من قبل العديد من الشركات الأخرى مثل Atmel و Philips وغيرها.

1.3. أهم ميزات المعالج:

يمكن أن نلخص أهم الموصفات الفنية للمتحكم 8051 بالنقاط التالية:

- يعمل بترد ساعة يصل إلى 24MHz قادر على تنفيذ 2 مليون تعليمة بالثانية
- معلب كدارة متكاملة ذات 40 مربط
- ذاكرة البرنامج 4KB
- ذاكرة المعطيات 128 بايت بما فيها 32 سجل عمل
- مسرى المعطيات على 8 بت

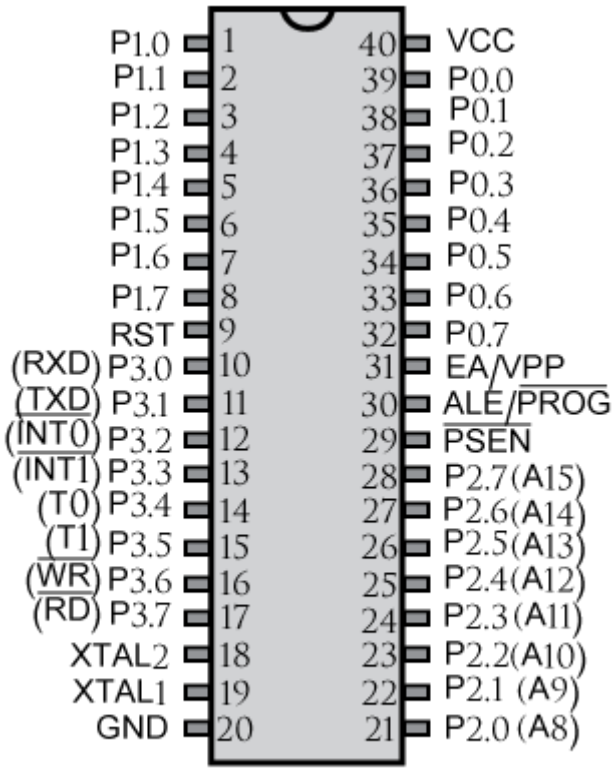
- مسرى العناوين على 16 بت
- يحتوي على مؤقتين مستقلين على 16 بت
- يحتوي على أربعة بوابات دخل/خرج كل منها على 8 بت
- له خمس مصادر للمقاطعة
- يحتوي على وحدة تراسل تسلسلي غير متزامن (UART)

يبين الجدول التالي مجموعة من المتحكمات التي تنتمي إلى نفس العائلة وأهم مواصفاتها الفنية:

جدول 1: مجموعة من المتحكمات من عائلة 8051.

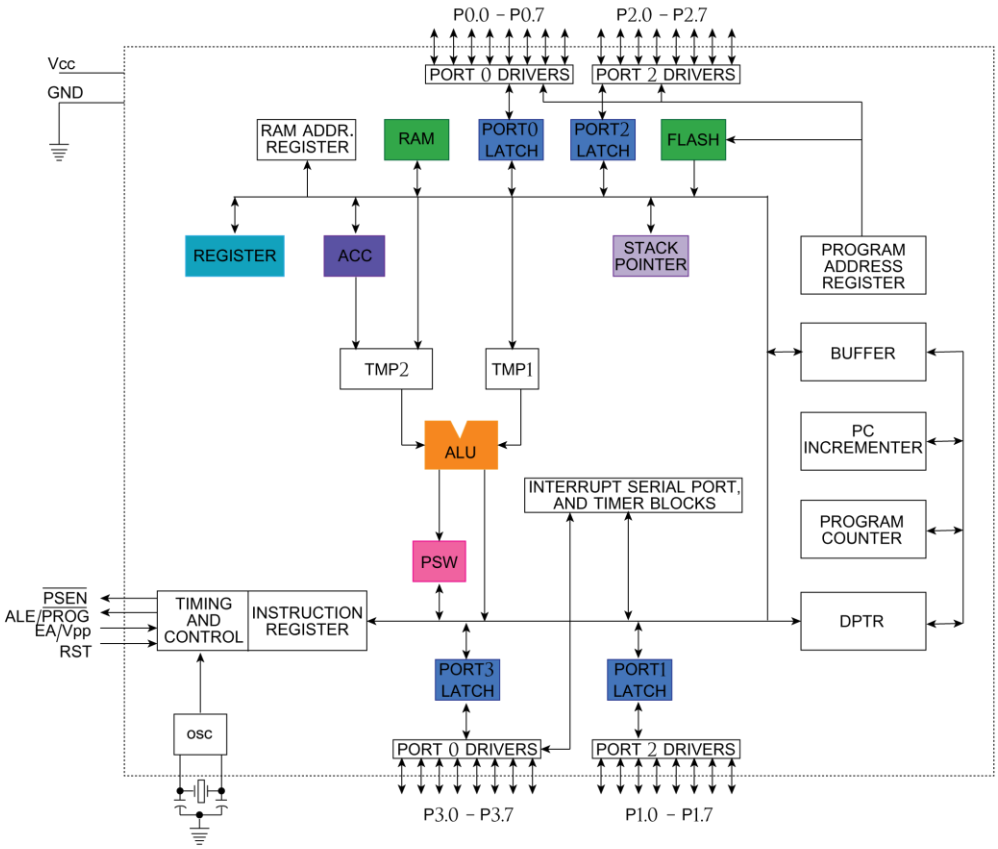
DEVICE	ON-CHIP DATA MEMORY (bytes)	ON-CHIP PROGRAM MEMORY (bytes)	16-BIT TIMER/ COUNTER	NO. OF INTERUPTS
8031	128	None	2	5
8032	256	none	2	6
8051	128	4k ROM	2	5
8052	256	8k ROM	3	6
8751	128	4k EPROM	2	5
8752	256	8k EPROM	3	6
AT89C51	128	4k Flash	2	5
AT89C52	256	8k Flash	3	6

ويبين الشكل التالي مرابط المتحكم 8051 المختلفة.



مرابط المتحكم 8051

يبين الشكل التالي المخطط الصندوقي للبنية الداخلية للمتحكم 8051 والتي سنقوم بشرح مختلف مكوناته تباعاً.



البنية الداخلية للمنحكم 8051

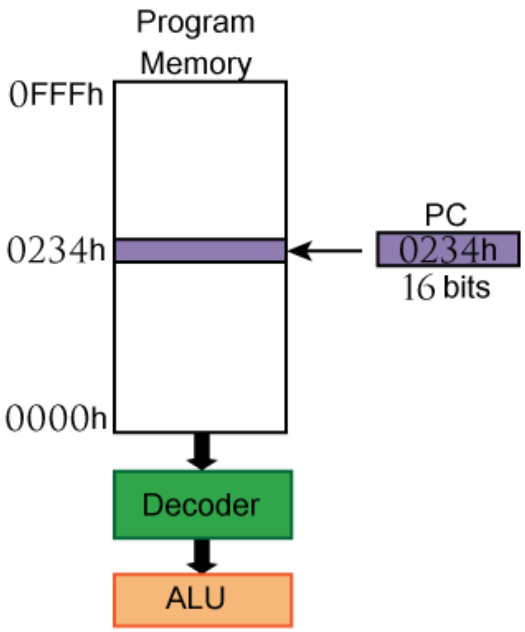
يتكون المتحكم بشكل أساسي من المكونات التالية:

- ذاكرة البرنامج
- ذاكرة المعطيات (RAM)
- وحدة الحساب والمنطق ALU
- بوابات الدخل والخرج
- الطرفيات
- سجلات الوظائف الخاصة
- دارة المهتز

2.3. ذاكرة البرنامج:

يحتوي هذا المتحكم في بنيته الداخلية على ذاكرة من نوع EEPROM أو Flash لتخزين البرنامج، وهي ذاكرة ميتة ولكنها قابلة للمسح وإعادة البرمجة كهربائياً. وقد كانت هذه الذاكرة في بداية ظهور هذه العائلة من المتحكمات من نوع ROM قابلة للبرمجة لمرة واحدة فقط. وتبلغ سعة هذه الذاكرة 4KB ولذلك فإن هذا المتحكم معد من أجل التطبيقات الصغيرة نسبياً.

يتكون مسرى العنونة لذاكرة البرنامج من 16 بت وبالتالي فهو قادر على تشغيل برنامج يصل طوله حتى 64kB. وعندما يكون طول البرنامج الذي نرغب بتنفيذه أكبر من 4KB وأقل من 64KB فمن الممكن إضافة ذاكرة خارجية للمتحكم ولكن يتطلب هذا المزيد من دارات الربط الخارجية ويزيد من تعقيد النظام. ويمكن إعلام المعالج مكان وجود البرنامج إما في ذاكرته الداخلية أو في ذاكرة خارجية من خلال المربط $\overline{EA}$، فمن أجل تنفيذ البرنامج من الذاكرة الداخلية يجب وضع هذا المربط على السوية العليا (+5V) ومن أجل تنفيذ البرنامج من ذاكرة خارجية يجب وضع هذا المربط على السوية الدنيا (0V).



ذاكرة البرنامج

يُستخدم سجل عداد البرنامج PC (Program Counter) كمؤشر تعليمات وهو على 16 بت ويدل على عنوان التعليمة التي يتم تنفيذها حالياً من قبل المتحكم (بينما في المعالج 8086 فإن سجل مؤشر التعليمات IP يدل على عنوان التعليمة التالية). ولا يتم هنا استخدام مبدأ التجزئة الذي رأيناه في حالة المعالج الصغري 8086 للحصول على العنوان الحقيقي، بل إن محتوى عداد البرنامج PC يمثل فعلاً العنوان الحقيقي.

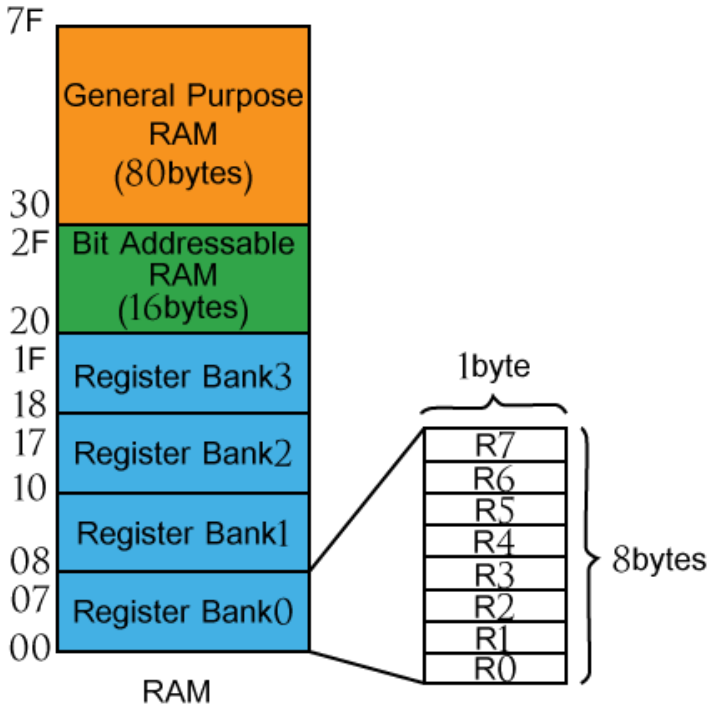
من الأمور الأساسية التي تميز هذا المتحكم عن المعالج الصغري 8086 هو الفصل ما بين ذاكرة البرنامج وذاكرة المعطيات من الناحية الفيزيائية ومن حيث فضاء العنونة. حيث تتوضع ذاكرة البرنامج في فضاء عنونة مستقل عن ذاكرة المعطيات . ترتبط ذاكرة البرنامج مع وحدة الحساب المركزية بمسرى مستقل عن مسرى ذاكرة المعطيات، مما يمكن من جلب التعليمات من الذاكرة في نفس الوقت الذي تقوم فيه وحدة الحساب المركزية بالنفاذ إلى الذاكرة الحية لقراءة أو كتابة المعطيات. لذلك يوجد في المتحكم، كما سنرى لاحقاً، تعليمات للنفاذ إلى ذاكرة البرنامج (MOVC) تختلف عن التعليمات المستخدمة للنفاذ إلى ذاكرة المعطيات (MOV).

بما أن ذاكرة البرنامج هي ذاكرة موجودة داخل المتحكم، لذلك تم تزويد المتحكم ببعض المرابط التي تُفيد في برمجة ذاكرة البرنامج الداخلية (VPP,$\overline{PROG}$, $\overline{PSEN}$). ولا نريد الخوض في تفاصيل كيفية برمجة المتحكم أو في كيفية ربط الذواكر الخارجية مع المتحكم فهذا خارج عن نطاق اهتمامنا.

3.3. ذاكرة المعطيات:

يمتلك المتحكم ذاكرة معطيات صغيرة نسبياً مكونة فقط من 128 بايت. ولذلك فهو مُعد، كما ذكرنا سابقاً، للتطبيقات البسيطة ذات المتطلبات القليلة بالنسبة للذاكرة الحية. وعندما يتطلب التطبيق حجم أكبر من ذاكرة المعطيات فمن الممكن إضافة ذاكرة حية خارجية قد تصل إلى 64KB ولكن يكون هذا على حساب زيادة تعقيد النظام وكذلك يكون زمن النفاذ إلى الذاكرة الخارجية أكبر من زمن النفاذ إلى الذاكرة الداخلية. عند استخدام ذاكرة معطيات خارجية، فإنه يمكن النفاذ إليها فقط بشكل غير مباشر باستخدام التعليمة MOVX مع سجل مؤشر المعطيات DPTR (Data Pointer) وهو سجل على 16 بت.

تُقسم الذاكرة الداخلية إلى ثلاثة أقسام كما هو مبين في الشكل التالي:



توزيع الذاكرة الحية الداخلية في المتحكم 8051

1.3.3. القسم الأول: سجلات العمل:

يتكون القسم الأول من 32 بايت من العنوان 00h وحتى 1Fh وهو مقسم بدوره إلى أربعة كتل كل منها يحتوي على 8 بايتات. يشكل هذا القسم مكاناً لتخزين سجلات العمل العامة (Working Registers). حيث تحتوي كل كتلة على 8 سجلات عمل R0,..., R7. إلا أنه لا يمكن لوحدة الحساب المركزية التعامل إلا مع كتلة واحدة من السجلات في نفس اللحظة. حيث يمكن انتخاب كتلة سجلات العمل الحالية عن طريق البتين RS1, RS0 من السجل PSW وذلك حسب الجدول 2 التالي:

جدول 2: انتخاب كتلة سجلات العمل.

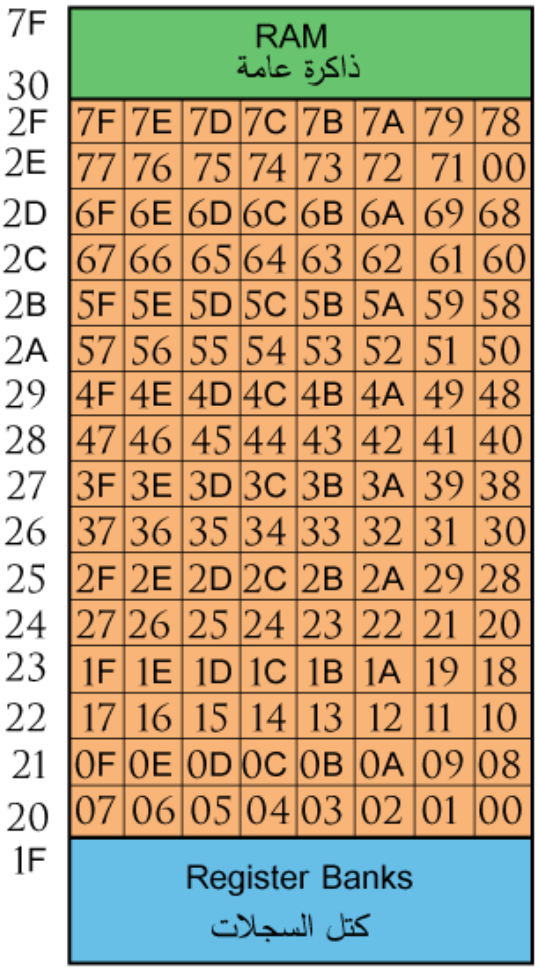
RS1 RS0	Register Bank
0 0	Bank 0: 00h–07h
0 1	Bank1: 08h–0Fh
1 0	Bank 2: 10h–17h
1 1	Bank 3: 18h–1Fh

إن الهدف الأساسي من وجود عدة كتل لسجلات العمل هو توفير وقت البرنامج في حفظ سجلات العمل الحالية في الإجرائيات والمقاطعات. فبدل أن تقوم إجرائية المقاطعة مثلاً بحفظ سجلات العمل الحالية من الكتلة 0، فإنها تقوم بالتحول إلى كتلة سجلات أخرى، مثل الكتلة 1، والعمل باستخدامها ثم العودة إلى الكتلة 0 عند نهاية الإجرائية. وبذلك نكون قد وفرنا لزمن اللازم لحفظ السجلات في بداية إجرائية المقاطعة واستعادتها من جديد عند نهاية الإجرائية.

يمكن توزيع استخدام هذه الكتل بين مختلف أجزاء البرنامج. فمثلاً يمكن للبرنامج الأساسي أن يستخدم الكتلة الأولى بينما يتم تخصيص الكتلة الثانية لإجرائية الاستجابة لمقاطعة ما، والكتلة الثالثة لمقاطعة أخرى. أما بالنسبة للكتل التي لا يتم استخدامها كسجلات عمل، فيمكن استخدامها كذاكرة عادية.

2.3.3. القسم الثاني: الذاكرة القابلة للعنونة بالبت

يتكون القسم الثاني من 16 بايت، وذلك انطلاقاً من العنوان 20h وحتى العنوان 2Fh. وهي عبارة عن مواقع ذاكرة قابلة للعنونة بالبايت كباقي الذاكرة، إلا أنها تتميز بكونها قابلة للعنونة بالبت (Bit Addressable)، أي يمكن النفاذ إلى كل بت من بتات هذه الذاكرة بشكل مباشر ومستقل دون الحاجة إلى شحن البايت الذي يحتوي هذا البت ومن ثم النفاذ إلى البت المرغوب سواءً في القراءة أم الكتابة. وتفيد هذه المنطقة من الذاكرة في تعريف المتحولات المنطقية التي تأخذ قيمتين: 0 أو 1 دون الحاجة لاستخدام بايت كامل لهذه المتحولات. يبين الشكل التالي عناوين البتات في الذاكرة حسب مكان وجودها.



عناوين البتات في القسم القابل للعنونة بالبت

3.3.3. القسم الثالث: ذاكرة عامة

يمكن استخدام القسم المتبقي من الذاكرة من الموقع 30h وحتى 7Fh كمواقع ذاكرة عامة لتخزين متحولات البرنامج والمعطيات التي يتعامل معها. ومن ضمن هذه الذاكرة، يمكن تخصيص جزء منها كذاكرة مكدس لتخزين المعطيات بشكل مؤقت وعناوين العودة من الإجرائيات والمقاطعات. ويُستخدم سجل مؤشر المكدس SP (Stack Pointer) وهو سجل على 8 بت للإشارة إلى قمة المكدس. ولكن في حالة المتحكم 8051 فهو مؤشر تصاعدي بعكس المعالج 8086 الذي يعمل فيه المكدس بشكل تنازلي. فمثلاً عند دفع قيمة ما إلى المكدس باستخدام التعليمة PUSH فإنه يتم زيادة مؤشر المكدس بمقدار واحد ومن ثم تخزين القيمة (على 8 بت) في الموقع الذي يشير إليه مؤشر المكدس.

تكون القيمة الابتدائية لمؤشر المكدس عند إقلاع المتحكم هي SP=7 الأمر الذي يعرف بداية ذاكرة المكدس في العنوان 08h. ولكن عادةً ما يتم تغييرها في بداية البرنامج ووضعها في القسم العلوي للذاكرة الداخلية لمنع تراكب ذاكرة المكدس مع كتل السجلات.

4.3. وحدة الحساب والمنطق:

تدعم وحدة الحساب والمنطق في المتحكم معظم العمليات الحسابية والمنطقية. حيث تعتمد هذه الوحدة على مفهوم المراكم (Accumulator) كسجل أساسي في معظم تعليمات المتحكم ويشمل ذلك السجل A والسجل B، حيث يستخدم هذا الأخير بشكل خاص في عمليات الضرب والقسمة. كما تستعين هذه الوحدة في عملها بسجلات العمل R0, …, R7. وكما هو معروف بالنسبة لمعظم وحدات الحساب المركزية، فإن هذه الوحدة تقوم برفع بعض الرايات، بعد كل عملية حسابية، في سجل الرايات أو سجل الحالة PSW (Program Status Word) على 8 بت. يبين الشكل التالي توضع رايات سجل الحالة:

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	–	P

سجل الحالة PSW

ويحتوي على الرايات التالية:

- راية الحمل CY: وهو بت الحمل (أو الاستعارة) الناتج عن البت الثامن عند إجراء عملية جمع (أو طرح) على 8 بت
- راية الحمل المساعد AC: وهو بت الحمل الناتج عن البت الرابع (b3) إلى البت الخامس (b4) ويفيد خلال إجراء العمليات الحسابية على الأرقام بصيغة الأرقام العشرية المرمزة ثنائياً BCD (Binary Coded Decimal)
- راية الفائض OV: يرفع عندما يكون هناك فائض عن العملية الحسابية لدى التعامل مع معاملات بصيغة المتمم الثنائي
- راية الزوجية P: يرفع عندما يكون عدد الوحدات في سجل المراكم A زوجياً

ومن الجدير بالذكر هنا، أنه لا وجود لراية الصفر كما في حالة المعالج 8086.

أما البت F0: فهو بت للاستخدام العام يمكن للمبرمج أن يكتب فيه أو يقرأ منه بشكل حر. ويُستخدم البتين RS1, RS0 لانتخاب كتلة السجلات المستخدمة.

5.3. بوابات الدخل والخرج:

يمتلك المتحكم أربعة بوابات دخل/خرج وهي P0, P1, P2, P3 كل منها على 8 بت موصولة مع مرابط المتحكم. حيث يمكن استخدام أي مربط منه بشكل مستقل كمربط دخل أو خرج.

مثلاً، يمكن التعامل مع البوابة P2 كبوابة على 8 بت وإخراج أية قيمة عليها بالشكل P2=07h أو التعامل مع كل مربط على حدة بالشكل: P2.2=1, P2.1=1, P2.0=1.

بالإضافة إلى إمكانية استخدام كل مربط كمربط دخل خرج عام، تمتلك بعض المرابط وظائف خاصة أيضاً. حيث تستخدم البوابتان P0 وP2 كمسرى للعناوين والمعطيات بشكل متناوب عند وصل ذاكرة خارجية مع المتحكم. حيث يستخدم المربط ALE في قدح العنوان قبل خروج أو دخول المعطيات بنفس الطريقة التي يعمل بها المعالج 8086.

كما تستخدم مرابط البوابة P3 لعدة وظائف خاصة، مثل المربطين $\overline{RD}$ و$\overline{WR}$ كإشارات تحكمية للقراءة والكتابة عند التعامل مع ذواكر خارجية. بالإضافة إلى المرابط الخاصة بالمؤقتات (T0, T1) والبوابة التسلسلية (TXD, RXD) والمقاطعات الخارجية ($\overline{INT0}, \overline{INT1}$).

6.3. الطرفيات:

تم تزويد هذا المتحكم بمؤقتين داخليين مستقلين هما Timer0 وTimer1 كل منهما على 16 بت. كما تم تزويد هذا المتحكم بوحدة تراسل تسلسلي غير متزامن ثنائي الاتجاه (Full Duplex) وتستخدم في تراسل المعطيات بشكل تسلسلي بين المتحكم والتجهيزات البعيدة.

بالإضافة إلى ذلك فإن المتحكم مزود بمتحكم مقاطعات يمكن أن يقاطع تنفيذ البرنامج الأساسي وله خمسة مصادر للمقاطعة: مقاطعتان خارجيتان (INT0, INT1)، ومقاطعة من كل مؤقت، ومقاطعة من وحدة التراسل التسلسلية. وسوف نشرح هذه الطرفيات بشكل مفصل في الفصول القادمة.

7.3. سجلات الوظائف الخاصة:

سجلات الوظائف خاصة SFR (Special Function Registers) هي مجموعة السجلات المستخدمة في البنية الداخلية للمتحكم والتي يُسمح للمبرمج النفاذ إليها مثل سجل المراكم A والسجل B وسجل الحالة PSW وسجل مؤشر المكدس SP وسجل مؤشر المعطيات DPTR وسجلات التحكم بالطرفيات. أما سجل عداد البرنامج PC والسجلات المؤقتة على مدخل وحدة الحساب المركزية فلا يمكن للمبرمج النفاذ إليها.

يبين الشكل 7 جميع سجلات الوظائف الخاصة في المتحكم وموقعها في فضاء عنونة المعطيات. مع العلم أن جميع السجلات الموجودة في العمود اليساري هي سجلات قابلة للعنونة بالبت أيضاً وتأخذ فيها البتات العناوين من 80h إلى FFh وموزعة بنفس طريقة توزع عناوين الذاكرة الداخلية القابلة للعنونة بالبت. فمثلاً يمكن كتابة ACC.3 للنفاذ إلى البت الرابع من المراكم ويكون عنوان هذا البت هو:

80h+12*8+3=128+96+3=227=E3h

0F8h								
0F0h	B 00000000							
0E8h								
0E0h	ACC 00000000							
0D8h								
0D0h	PSW 00000000							
0C8h								
0C0h								
0B8h	IP XX000000							
0B0h	P3 11111111							
0A8h	IE 0X000000							
0A0h	P2 11111111							
098h	SCON 00000000	SBUF XXXXXXXX						
090h	P1 11111111							
088h	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000		
080h	P0 11111111	SP 00000111	DPL 00000000	DPH 00000000				PCON 00XX0000

سجلات الوظائف الخاصة في المتحكم 8051 وقينها الإبتدائية

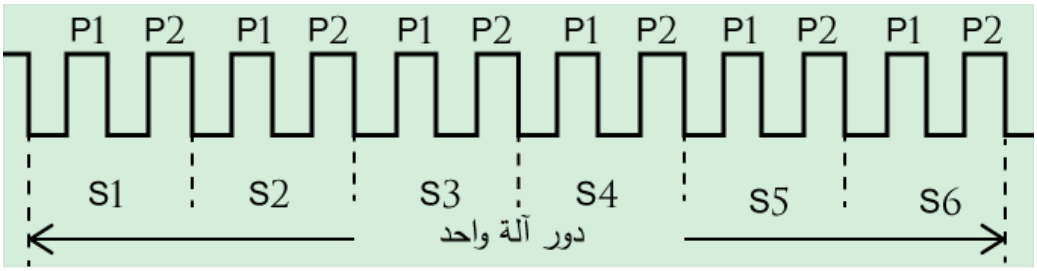
نلاحظ أن هذه السجلات موجودة في فضاء عنونة المعطيات من الموقع 80h إلى الموقع FFh. أما المواقع الخالية فهي لا تحتوي على أي سجل. وهي متروكة للاستخدام المستقبلي في المتحكمات اللاحقة.

جميع هذه السجلات، هي سجلات على 8 بت ما عدا سجل مؤشر المعطيات DPTR فهو على 16 بت وتم فصله إلى قسمين DPH للبايت ذي الوزن الأعلى وDPL للبايت ذي الوزن الأدنى.

8.3. دارة المهتز:

تؤمن دارة المهتز في المتحكم ساعة العمل له وذلك من خلال وصل كريستاله خارجية على مرابطه الخارجية XTAL1, XTAL2 بالتردد المرغوب.

تتكون دورة الآلة في هذا المتحكم من 12 دور ساعة، حيث تقسم دورة الآلة إلى 6 مراحل يرمز إليها من S0 إلى S6 وتشمل كل مرحلة على صفحتين (Phase) أو نبضتي ساعة يرمز إليها P1 وP2.



دور الآلة في المتحكم ٨٠٥١

يكون زمن تنفيذ معظم التعليمات في هذا المتحكم هو دورة آلة واحدة ولكن هناك بعض التعليمات التي تأخذين دورين أو ثلاثة أو أربعة أدوار. ويمكن الرجوع إلى النشرة الفنية للمتحكم عند الحاجة لمعرفة زمن تنفيذ أية تعليمة من التعليمات.

أسئلة الفصل الثامن

1. ما هو الفرق بين المعالج والمتحكم الصغري؟

المعالج هو دارة تكاملية تحتوي على وحدة حساب مركزية ووحدة تحكم تُستخدم في بناء نظم المعالجة الحاسوبية العامة لتنفيذ برنامج باستخدام الطرفيات المناسبة من ذواكر وبوابات دخل خرج وطرفيات متنوعة.

المتحكم هو عبارة عن نظام معالجة حاسوبي مصغر ومدمج في دارة تكاملية واحدة مع ذواكر وطرفيات مدمجة من أجل تنفيذ برامج لأهداف خاصة.

2. ما هي الذاكرة القابلة للعنونة بالبت في المتحكم 8051 وكم حجمها وأين تقع؟

هي ذاكرة مكونة من 16 بايت من الموقع 20H إلى الموقع 2Fh من الذاكرة الحية، وهي ذاكرة يمكن النفاذ إلى كل بت فيها بشكل مستقل باستخدام العنوان المباشر للبت.

3. ما هي السجلات ذات الوظائف الخاصة في المتحكم 8051؟

هي السجلات الداخلية في المتحكم التي يمكن النفاذ إليها من قبل المبرمج بالإضافة إلى سجلات التحكم بالطرفيات.

4. ما هو دور السجل PSW في المتحكم 8051؟

يحتوي هذا السجل على رايات وحدة الحساب والمنطق ويشمل ذلك رايات الحمل والحمل الثانوي والفائض والزوجية بالإضافة إلى بتات انتخاب كتلة سجلات العمل.

5. ما هي الفائدة من تقنية التبديل بين كتل السجلات في المتحكم 8051؟

توفير زمن حفظ واستعادة سجلات العمل في إجرائيات المقاطعات.

6. ما هو زمن دور الآلة في المتحكم 8051 الذي يعمل على تردد ساعة قدره 12MHz.

$$T = 12Tosc = 12\frac{1}{Fosc} = 12\frac{1}{12 \times 10^6} = 1\mu s$$

أسئلة خيارات متعددة

1. المتحكم 8051 هو متحكم على:
   A. 4 بت
   B. 8 بت
   C. 12 بت
   D. 16 بت

2. ماهو حجم ذاكرة البرنامج في المتحكم 8051:
   A. 4KB
   B. 8KB
   C. 16KB
   D. 64KB

3. ماهو حجم ذاكرة المعطيات المتحكم 8051:
   A. 32 بايت
   B. 80 بايت
   C. 128 بايت
   D. 256 بايت

4. ما هو عدد سجلات العمل في المتحكم 8051:
   A. 8 سجلات
   B. 32 سجل
   C. 32 سجل في أربعة كتل
   D. 128 سجل

5. أي من السجلات التالية لا يمكن للمبرمج النفاذ إليها مباشرة؟
   A. PSW
   B. PC
   C. ACC
   D. SP

6. كم عدد مرابط الدخل والخرج الموجودة في المتحكم 8051؟

A. 4

B. 8

C. 16

D. 32

7. ما هو دور السجل SP في المتحكم 8051؟

A. عداد تعليمات البرنامج

B. مؤشر لذاكرة المكدس

C. سجل انتخاب كتلة سجلات العمل

D. سجل الحالة لوحدة الحساب والمنطق

8. كم عدد السجلات على 16 بت في المتحكم 8051؟

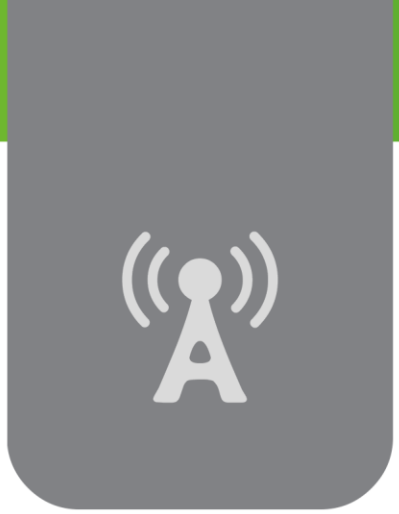
A. 0

B. 1

C. 2

D. 3

رقم التمرين	الإجابة الصحيحة
1	B
2	A
3	C
4	C
5	C
6	D
7	B
8	C



الفصل التاسع:برمجة المتحكم 8051

الكلمات المفتاحية:

أنماط العنونة في المتحكم 8051، تعليمات المتحكم 8051،.حلقات التأخير، النفاذ إلى بوابات الدخل والخرج.

ملخص:

يهدف هذا الفصل إلى التعرف على تعليمات المتحكم 8051 وفهم العلاقة بين بنية المتحكم والتعليمات التي يدعمها ومقارنة طريقة برمجة هذا المتحكم مع ما قد سبق وتعرفنا عليه في طرق برمجة لمعالج 8086. سيدرك الطالب من خلال دراسة هذا المتحكم مدى سهولة فهم عمل هذا المتحكم وغيره في ضوء فهمه لعمل المعالج بشكل عام مع تسليط الضوء على الخصوصيات التي يوفرها هذا المتحكم ولا سيّما فصل فضاء عنونة ذاكرة البرنامج عن ذاكرة المعطيات وكذلك تعليمات التعامل مع المعطيات على مستوى البت.

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

- أنماط العنونة في المتحكم 8051
- تعليمات المتحكم بلغة التجميع
- برمجة المتحكم من خلال مثالين عن حلقات التأخير واستخدام بوابات الدخل والخرج

الفصل التاسع: برمجة المتحكم 8051

يهدف هذا الفصل إلى التعرف على تعليمات المتحكم 8051 وفهم العلاقة بين بنية المتحكم والتعليمات التي يدعمها ومقارنة طريقة برمجة هذا المتحكم مع ما قد سبق وتعرفنا عليه في طرق برمجة المعالج 8086. سيدرك الطالب من خلال دراسة هذا المتحكم مدى سهولة فهم عمل هذا المتحكم وغيره في ضوء فهمه لعمل المعالج بشكل عام مع تسليط الضوء على الخصوصيات التي يوفرها هذا المتحكم ولا سيّما فصل فضاء عنونة ذاكرة البرنامج عن ذاكرة المعطيات وكذلك تعليمات التعامل مع المعطيات على مستوى البت.

1. مقدمه:

تتشابه جميع المعالجات والمتحكمات في تعليمات لغة التجميع مع بعض الاختلافات الطفيفة والتي تتعلق بخصوصية كل متحكم أو معالج حسب بنيته الداخلية. وقد رأينا في الفصل السابق بأن المتحكم 8051 له بنية داخلية مختلفة عن بنية المعالج 8086 حيث أن هناك اختلافاً في عرض مسرى المعطيات والعناوين بالإضافة للفصل بين ذاكرة البرنامج وذاكرة المعطيات في المتحكم 8051. ولكن مع ذلك فهناك تشابه كبير في آلية العمل. لذلك سوف نرى في هذا الفصل كيفية انعكاس ذلك على لغة التجميع الخاصة بالمتحكم وكيفية برمجته.

2. تعليمات المتحكم:

يمتلك المتحكم 8051 عدداً كبيراً من التعليمات حيث يمكن تقسيمها إلى عدة فئات حسب الوظيفة التي تقوم بها وهي تتكون من أربعة فئات رئيسية:

1. تعليمات نقل المعطيات.
2. التعليمات الحسابية والمنطقية.
3. تعليمات التفريع والقفز.
4. التعليمات الخاصة بالتعامل مع البتات.

سوف نشرح هذه التعليمات على التوالي في الفقرات القادمة.

1.2. تعليمات نقل المعطيات:

يدعم المتحكم 8051 العديد من تعليمات نقل المعطيات والنفاذ إلى الذاكرة الحية وذاكرة البرنامج والذواكر الخارجية. مع العلم أن عمليات النفاذ إلى ذاكرة البرنامج تنحصر في عمليات القراءة فقط إذ أن ذاكرة البرنامج هي ذاكرة ميتة لا يمكن الكتابة فيها.

نميز في هذا المتحكم العديد من أنماط العنونة المستخدمة للنفاذ إلى الذواكر، وهذا ما سنتطرق إليه في الفقرة القادمة.

1.1.2. أنماط العنونة:

العنونة الفورية:

تستخدم العنونة الفورية (Immediate addressing) لشحن القيم الثابتة على 8 بت في السجلات والذواكر، ولذلك نستخدم السابقة "#" للتعبير عن قيمة ثابتة، مثل:

```
MOV     A,#20h          ; load the value 20h in A
MOV     R3,#130         ; load the value 130 in R3
```

العنونة المباشرة:

في نمط العنونة المباشرة (Direct Addressing) يمكن النفاذ إلى موقع ضمن الذاكرة أو سجلات الوظائف الخاصة SFR بشكل مباشر عن طريق عنوان الموقع المرمز على 8 بت ضمن التعليمة.

```
MOV     A, 035h         ; move the content of memory location 35h to A
MOV     A, SP           ; move the content of SP to A
```

حيث يتم تخزين عنوان الذاكرة (هنا 035h) أو عنوان السجل المستخدم (هنا 081h للسجل SP) ضمن رمز التعليمة بلغة الآلة.

العنونة بالسجل:

في نمط العنونة بالسجل (Register Addressing) يمكن النفاذ إلى قسم السجلات من العنوان 00h إلى العنوان 1Fh عن طريق السجلات R0 إلى R7 وذلك بعد انتخاب كتلة السجلات التي يقع فيها هذا الموقع (عن طريق البتين RS0, RS1 في سجل الحالة PSW).

```
MOV     PSW,#10h    ;select register bank 2 RS1=1 RS0=0
MOV     R4, #40     ; load R0 (i.e the location 14h) with the value 40
```

ولا ننسى هنا عند قيامنا بشحن قيمة PSW بالقيمة 10h نكون قد صفرنا جميع رايات السجل PSW. ولتجنب ذلك يمكن وضع قيم البتين RS0 و RS1 بشكل مستقل من دون التأثير على بقية البتات باستخدام تعليمات النفاذ إلى البتات كما سنرى لاحقاً وذلك لأن السجل PSW هو قابل للعنونة بالبت.

العنونة غير المباشرة:

في العنونة غير المباشرة (Indirect Addressing) يمكن استخدام السجل R0 أو R1 كمؤشر للموقع المراد النفاذ إليه في الذاكرة الداخلية أو الخارجية عندما يكون العنوان مرمزاً على 8 بت، مثل:

```
MOV     R0,#35h     ;put 35h in R0
MOV     A,@R0       ;read the content of memory pointed by R0(35h) to A
MOV     R0,#10h     ; put 10 in R0
MOVX@R0,A       ;move A to external memory pointed by R0 (10h)
```

كما يمكن النفاذ إلى الذاكرة الخارجية بشكل غير مباشر عندما يكون العنوان مرمزاً على 16 بت باستخدام مؤشر المعطيات DPTR، مثل:

```
MOV     DPTR, #1000h    ;put the value 1000h in DPTR
MOVXA, @DPTR            ;read the external memory pointed by DPTR
```

العنونة غير المباشرة بالدليل:

في العنونة غير المباشرة بالدليل (Indexed Addressing) يمكن استخدام سجل المراكم كدليل عند النفاذ إلى ذاكرة البرنامج فقط بشكل غير مباشر، حيث يتم شحن مؤشر المعطيات بالعنوان القاعدي ويتم شحن المراكم بقيمة الانزياح (الموجب دوماً) عن العنوان القاعدي. مثال:

```
MOV     DPTR,#0200h
MOV     A, #10
MOVCA,@A+DPTR
```

حيث نقوم هنا بشحن محتوى موقع ذاكرة البرنامج ذي العنوان 0210h إلى المراكم A.

كما يمكن استخدام سجل عداد البرنامج PC كسجل قاعدي بدل من DPTR، مثل:

```
MOVCA,@A+PC
```

يفيد هذا النمط من العنونة في النفاذ إلى جدول يحتوي على قيم ثابتة مخزنة في ذاكرة البرنامج مثل قراءة سلسلة من المحارف تمثل عبارة نصية مخزنة في ذاكرة البرنامج لعرضها على شاشة عرض مثلاً.

2.1.2. النفاذ إلى الذاكرة الحية الداخلية:

يمكن النفاذ إلى الذاكرة الحية الداخلية بشكل كامل باستخدام التعليمة MOV التي تدعم جميع أنماط العنونة التي ذكرناها. بالإضافة إلى تعليمة MOV هناك تعليمتي PUSH و POP اللتين تستخدمان في التخزين والجلب من ذاكرة المكدس. وأخيراً يوجد تعليمتين للمبادلة بين المراكم وموقع من الذاكرة الحية هما XCH و XCHD.

Mnemonic	Operation	Addressing Modes				Cycles
		Dir	Ind	Reg	Imm	
MOV A, <src>	A = <src>	X	X	X	X	1
MOV <dest>, A	<dest> = A	X	X	X		1
MOV <dest>, <src>	<dest> = <src>	X	X	X	X	2
MOV DPTR, #data 16	DPTR = 16-bit immediate constant				X	2
PUSH <src>	INC SP: MOV "@SP", <scr>	X				2
POP <dest>	MOV <dest>, "@SP": DEC SP	X				2
XCH A, <byte>	ACC and <byte> Exchange Data	X	X	X		1
XCHD A, @Ri	ACC and @ Ri exchange low nibbles		X			1

ملاحظة-1: لا يمكن النفاذ إلى سجلات الوظائف الخاصة إلا باستخدام نمط العنونة المباشرة فقط ولا يمكن استخدام العنونة غير المباشرة.

ملاحظة-2: نستخدم الرمز ACC للمراكم بدل A للدلالة إلى العنوان المباشر للمراكم A (وهو E0h)، أي يتم التعامل مع المراكم كأي سجل من سجلات الوظائف الخاصة. أما عند وجود الرمز A فإنه يكون من ضمن التعليمة ولا يمكن تبديله بأي سجل آخر أو عنوان ذاكرة.

3.1.2. القراءة من ذاكرة البرنامج:

يمكن قراءة أي موقع في ذاكرة البرنامج باستخدام التعليمة MOVC باستخدام نمط العنونة بالدليل. وتُستخدم هذه التعليمة لقراءة بعض المعطيات الثابتة والمخزنة في ذاكرة البرنامج مثل العبارات النصية وجداول المعطيات الثابتة وغيرها.

Mnemonic	Operation	Cycles
MOVC A, @A + DPTR	Read Pgm Memory at (A + DPTR)	2
MOVC A, @A + PC	Read Pgm Memory at (A + PC)	2

4.1.2. النفاذ إلى الذاكرة الخارجية:

يمكن النفاذ إلى الذاكرة الحية الخارجية عند إضافة ذاكرة معطيات خارجية باستخدام التعليمة MOVX وهي تعليمة تدعم العنونة غير المباشرة سواء على 8 بت باستخدام المؤشر R0 أو R1، أم على 16 بت باستخدام مؤشر المعطيات DPTR.

Address Width	Mnemonic	Operation	Cycles
8 bits	MOVX A, @Ri	Read external RAM @ Ri	2
8 bits	MOVX @ Ri, A	Write external RAM @ Ri	2
16 bits	MOVX A, @ DPTR	Read external RAM @ DPTR	2
16 bits	MOVX @ DPTR, A	Write external RAM @ DPTR	2

مثال: نسخ سلسلة محارف من ذاكرة البرنامج إلى الذاكرة الحية:

نريد كتابة إجرائية تقوم بنسخ سلسلة من المحارف الموجودة ضمن ذاكرة البرنامج في العنوان المحدد باللصاقة MSG إلى الذاكرة الحية انطلاقاً من الموقع 30h. لا نعرف عدد محارف السلسلة ولكن يتم التعرف على نهايتها عن طريق وجود قيمة الصفر المخزنة في نهاية السلسلة.

يبين البرنامج التالي الإجرائية ReadMsg التي تقوم بالعملية المطلوبة.

```
ReadMsg:
        MOV   DPTR,#MSG
        MOV   R0,#30h
```

```
        MOV   R7,#00h
Loop:
        MOV         A,R7
        MOVCA,@A+DPTR
        MOV         @R0,A
        INC         R7
        INC         R0
        JNZ         Loop
        RET

MSG: DB "Hello", 0
```

نقوم أولاً بشحن مؤشر معطيات بعنوان بداية سلسلة المحارف المعطى باللصاقة MSG ونقوم بشحن السجل R0 بعنوان بداية تخزين سلسلة المحارف في الذاكرة الحية. كما نقوم بتصفير قيمة الدليل باستخدام السجل R7. ثم نقوم بإنشاء حلقة يتم فيها عملية نسخ المحارف واحداً تلو الآخر. من أجل كل محرف ننسخ قيمة الدليل إلى المراكم ثم نستخدم التعليمة MOVC لقراءة المحرف الموجود في ذاكرة البرنامج في الموقع A+DPTR ونضع النتيجة في المراكم. ثم نخزن المحرف الذي في المراكم في العنوان الذي يشير إليه السجل R0. بعد ذلك، نزيد كلاً من سجل الدليل R7 ومؤشر الذاكرة الحية R0 بمقدار واحد من أجل المحرف التالي. تفحص التعليمة JNZ قيمة المراكم –الذي يحتوي على المحرف المقروء– فإذا كان لا يساوي الصفر يعود إلى الحلقة لينسخ المحرف التالي وإلا فنكون قد وصلنا إلى نهاية سلسلة المحارف، حيث أننا وضعنا قيمة الصفر في آخر سلسلة المحارف لمعرفة نهاية السلسلة ونخرج من الحلقة لنصل إلى نهاية الإجرائية.

2.2. التعليمات الحسابية والمنطقية:

يدعم المتحكم التعليمات الحسابية التقليدية على 8 بت مثل الجمع والطرح والضرب والقسمة والزيادة والانقاص.

Mnemonic	Operation	Addressing Modes				Cycles
		Dir	Ind	Reg	Imm	
ADD A, <byte>	A = A + <byte>	X	X	X	X	1
ADDC A, <byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A, <byte>	A = A – <byte> – C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A – 1	Accumulator only				1
DEC <byte>	<byte> = <byte> – 1	X	X	X		1
MUL AB	B:A = B x A	ACC and B only				4
DIV AB	A = Int [A/B] B = Mod [A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

تعتمد معظم هذه التعليمات على المراكم كمعامل أول للتعليمة، كما ويتم تخزين النتيجة في المراكم أيضاً.

كما يدعم المتحكم التعليمات المنطقية الأساسية مثل AND وOR وXOR وNOT بالإضافة إلى عمليات الإزاحة والدوران. وبشكل مماثل للعمليات الحسابية، فإن معظم هذه العمليات تتم باستخدام المراكم.

Mnemonic	Operation	Addressing Modes				Cycles
		Dir	Ind	Reg	Imm	
ANL A, <byte>	A = A AND <byte>	X	X	X	X	1
ANL <byte>, A	<byte> = <byte> AND A	X				1
ANL <byte>, # data	<byte> = <byte> AND # data	X				2
ORL A, <byte>	A = A OR <byte>	X	X	X	X	1
ORL <byte>, A	<byte> = <byte> OR A	X				1
ORL <byte>, # data	<byte> = <byte> OR # data	X				2
XRL A, <byte>	A = A XOR <byte>	X	X	X	X	1
XRL <byte>, A	<byte> = <byte> XOR A	X				1
XRL <byte>, # data	<byte> = <byte> XOR # data	X				2
CLR A	A = 00H	Accumulator only				1
CLP A	A = NOT A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

نعلم أن التعليمات الحسابية ممكن أن تؤثر في سجل الحالة من خلال وضع رايات الحمل والحمل المساعد والفائض. يبين الجدول التالي قائمة بالتعليمات التي يمكن أن تؤثر على قيمة الرايات. وكما ذكرنا في الفصل السابق، فلا وجود لراية الصفر كما في حالة المعالج 8086. أما راية الزوجية P فقيمتها متعلقة بقيمة المراكم A فقط، وتتغير حسب قيمة المراكم.

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

3.2. عمليات التفريع والقفز:

يوجد في المتحكم 8051 عدة تعليمات للقفز اللاشرطي إما بشكل مباشر باستخدام اللصاقات أو بشكل غير مباشر باستخدام سجل مؤشر المعطيات مع المراكم. أما تعليمات القفز المباشر فهي SJMP وAJMP وLJMP وجميعها تقوم بنفس العمل ولكن الاختلاف بينها يكمن في ترميز مقدار انزياح اللصاقة التي سيتم القفز إليها، إما على 8 بت بالنسبة للتعليمة SJMP أو على 11 بت بالنسبة للتعليمة AJMP. أما التعليمة LJMP فيتم إعطاء العنوان الجديد كاملاً على 16 بت ضمن التعليمة. فتعليمة SJMP تستخدم للقفز القصير وAJMP للقفز المتوسط، أما LJMP فللقفز الطويل. أما تعليمة JMP فهي للقفز الطويل غير المباشر إلى العنوان الناتج عن جمع سجل المعطيات DPTR مع المراكم .A

يمكن طلب إجرائية ما باستخدام تعليمة استدعاء الإجرائيات ACALL أو LCALL. تستخدم تعليمة ACALL للإجرائيات غير البعيدة التي يمكن ترميز انزياح موقعها عن تعليمة الطلب على 11 بت. أما LCALL فتستخدم للإجرائيات البعيدة.حيث يتم شحن عداد البرنامج بعنوان الإجرائية على 16 بت.ويعود المتحكم إلى البرنامج الأساسي في نهاية الإجرائية باستخدام التعليمة RET. أما التعليمة RETI فتستخدم للعودة من إجرائيات المقاطعات.

Mnemonic	Operation	Cycles
SJMP addr	Short jump to addr (8 bits)	2
AJMP addr	Absolute jump to addr (11 bits)	
LJMP addr	Long jump to addr (16 bits)	
JMP @A + DPTR	Jump to A + DPTR	2
ACALL addr	Call subroutine at addr (11 bits)	2
LCALL addre	Call subroutine at addr (16 bits)	
RET	Return from subroutine	2
RETI	Return from interrupt	2

أما تعليمات القفز المشروط فهي تعليمات تقوم بالقفز عند تحقق شرط معين. وجميع هذه التعليمات هي من نوع القفز القصير أي يجب أن يكون مقدار انزياح عنوان اللصاقة (rel) التي سيقفز إليها البرنامج عن موقع تعليمة القفز واقع في المجال [127+ ,128−].

Mnemonic	Operation	Addressing Modes				Cycles
		Dir	Ind	Reg	Imm	
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A □ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNZ A,<byte>,rel	Jump if A = <byte>	X			X	2
CJNE <byte>,#data, rel	Jump if <byte> = #data		X	X		2

Mnemonic	Operation	Cycles
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit, rel	Jump if bit = 1	2
JNB bit, rel	Jump if bit = 0	2
JBC bit, rel	Jump if bit = 1 ; CLR bit	2

4.2. تعليمات التعامل مع البتات:

يمكن للمتحكم النفاذ إلى 256 بت بشكل مستقل وهي مقسمة إلى قسمين: القسم الأول يتكون من 128 بت موجودة في الذاكرة الحية القابلة للعنونة بالبت (من البايت 20h إلى 2Fh). أما القسم الثاني فهو مكون من 128 بت موجودة في العمود الأول لسجلات الوظائف الخاصة، حيث نعلم أن جميع السجلات في هذا العمود قابلة للعنونة بالبت.

يمكن في التعليمات التي تتعامل مع البتات أن يتم تعيين البت المراد بطريقتين:

الطريقة الأولى بتحديد البايت الذي يقع فيه هذا البت ورقمه في البايت مثل:

```
MOV   C, 20h.7
MOV   C, ACC.3
```

أو عن طريق تحديد عنوان البت حيث يتم ترقيم البتات تصاعدياً ابتداءً من البت 20h.0 فهو يوافق البت ذو العنوان 00h والبت 20h.1 هو البت ذو العنوان 01h والبت 21h.0 هو البت ذي العنوان 08h وهكذا حتى الوصول إلى البت 2Fh.7 ذي العنوان 127. أما البتات من 128 إلى 255 فهي موجودة في سجلات الوظائف الخاصة بالترتيب التصاعدي لعناوين السجلات القابلة للعنونة بالبت. ومثال ذلك يمكن كتابة التعليمتين السابقتين وفق طريقة العنونة المباشرة كالتالي:

```
MOV   C, 07h          ; 20h.7 is the bit no 7
MOV   C, 0E3h         ; ACC.3 is the bit no 227=E3h
```

يدعم المتحكم عمليات النقل على مستوى البت. وهنا يلعب بت الحمل C في السجل PSW دور المراكم بالنسبة للعمليات على مستوى البت. كذلك يدعم المتحكم بعض العمليات المنطقية على مستوى البت مثل AND وOR وNOT. كما يمكن وضع قيمة البت على الواحد أو على الصفر باستخدام التعليمتين SET وCLR.

Mnemonic	Operation	Cycles
ANL C,bit	C = C AND bit	2
ANL C,/bit	C = C AND (NOT bit)	2
ORL C,bit	C = C OR bit	2
ORL C,/bit	C = C OR (NOT bit)	2
MOV C,bit	C = bit	1
MOV bit,C	bit = C	2
CLR C	C = 0	1
CLR bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = NOT C	1
CPL bit	bit = NOT bit	1

5.2. مثال حلقات التأخير:

نريد كتابة إجرائية بلغة التجميع للمتحكم 8051 تدعى delay_1ms تقوم بتأخير زمني مقداره 1 ms مع العلم أن المتحكم يعمل على تردد ساعة مقداره 12 MHz. أي عند طلب الإجرائية بتعليمة CALL كالتالي:

```
CALL        delay_1ms
```

يجب أن يستغرق المتحكم زمن 1 ms لتنفيذ الإجرائية والذهاب إلى التعليمة التالية.

بما أن تردد الساعة الرئيسية للمتحكم هي Fosc=12 MHz، بالتالي يكون زمن دور الآلة Tm هو:

Tm=12/Fosc=1 µs

وذلك لأن كل دور آلة في المتحكم يستغرق 12 نبضة ساعة.

وبالتالي من أجل تأخير زمني كلي مقداره T=1 ms نحتاج إلى تأخير عدد N من دورات الآلة مقداره:

N=T/Tm=1 ms/ 1 µs= 1000 Machine Cycles

لذلك نكتب حلقة تقوم بإنقاص عداد (نأخذه السجل R7 مثلاً) في كل تكرار، ويتم شحنه بقيمة بدائية معينة x (سنقوم بحساب هذه القيمة لاحقاً) حتى يصل للصفر. بما أن سجلات المتحكم هي على 8 بت فقط، فيجب أن لا تتجاوز القيمة التي يمكن شحنها في العداد عن 255. ويمكن ذلك بجعل قيمة العداد مساوية لـ 250 بحيث نقوم في كل تكرار بتأخير مقداره 4 دورات آلة كالتالي:

```
delay_1ms:
            MOV   R7,#x?        ;1
```

```
Loop:       NOP          ;1
            NOP          ;1
DJNZ  R7, Loop           ;2
            RET          ;2
```

حساب زمن التنفيذ:

انطلاقاً من قائمة تعليمات المتحكم، يمكن أن نحسب زمن التعليمات مقدراً بعدد دورات الآلة، داخل الإجرائية كالتالي:

$$T1=1+x(1+1+2)+2=3+4x$$

بإضافة زمن تعليمة CALL في البرنامج الرئيسي (وهو 2) نجد أن الزمن الكلي هو:

$$T=T1+2=5+4x$$

نريد أن يكون هذا الزمن مساوياً 1000 ومنه نستنتج قيمة x:

$$5+4x=1000 \Rightarrow x=248.75$$

ولكن يجب أن تكون هذه القيمة صحيحة، لذلك نأخذ x=248. أي أن القيمة البدائية التي يجب شحنها في السجل R7 هي 248. ويكون بذلك الزمن الفعلي الكلي هو

$$T=5+4*248=997$$

فإذا أردنا أن يكون هذا الزمن مساوياً تماماً 1000، يجب إضافة 3 دورات آلة داخل الإجرائية، كإضافة ثلاث تعليمات NOP قبل تعليمة RET.

6.2. مثال عن البوابات:

نريد كتابة برنامج يقوم بقراءة قيمة من البتات الستة الدنيا للبوابة P1 (قيمة من 0 وحتى 63) ويقوم بحساب خانة الآحاد وخانة العشرات لهذه القيمة إخراج النتيجة على البوابة P2 بحيث يتم إخراج خانة الآحاد على البتات الأربعة الدنيا للبوابة P2 وخانة العشرات على البتات الأربعة العليا لنفس البوابة.

```
MOV     A,P1      ;read P1 value
ANL     A,#3Fh    ;remove last 2 bits
MOV     B,#10     ;load B with 10
DIV     AB        ;divide A/B: A contains result, B contain remainder
SWAP    A         ;swap low 4 bits of A with the high 4 bits
OR      A,B       ;put low 4
MOV     P2,A      ;output result to P2
```

في هذا المثال نقوم أولاً بقراءة محتوى البوابة P1 ونضع النتيجة في المراكم A. وبما أننا لا نحتاج إلا للبتات الستة الدنيا، لذلك نقوم بتصفير البتين العلويين من القيمة المقروءة باستخدام تعليمة AND المنطقية مع القيمة 3Fh (0011 1111b). وبعد ذلك نشحن قيمة السجل B بالقيمة 10 وذلك تمهيداً لإجراء عملية قسمة القيمة المقروءة على 10 لفصل الآحاد عن العشرات. حيث تقوم التعليمة DIV AB بتقسيم محتوى المراكم A على السجل B وبعد تنفيذ هذه العملية، يكون ناتج القسمة الصحيح في السجل A (وهو قيمة خانة العشرات) بينما يكون في السجل B باقي القسمة (وهو قيمة خانة الآحاد). نريد الآن دمج القيمتين ممثلاً على 8 بت بحيث تكون خانة العشرات في البتات الأربعة العليا. لذلك نستخدم التعليمة SWAP التي تقوم بالتبديل بين البتات الأربعة الدنيا من المراكم مع البتات الأربعة العليا. ثم نقوم بوضع البتات الأربعة الدنيا التي تمثل خانة الآحاد والموجودة في السجل B في المراكم باستخدام تعليمة OR. وهكذا تصبح النتيجة جاهزة للإخراج على البوابة P2 من خلال التعليمة الأخيرة.

أسئلة الفصل التاسع

أسئلة عامة

1. ما هي السجلات الممكن استخدامها كمؤشرات في العنونة غير المباشرة للذاكرة الداخلية في المتحكم 8051؟

السجلين R0 و R1 فقط.

2. ما هي السجلات الممكن استخدامها كمؤشرات في العنونة غير المباشرة للذاكرة الخارجية في المتحكم 8051؟

السجلين R0 و R1 من أجل العنونة على 8 بت والسجل DPTR من أجل العنونة على 16 بت.

3. ما هي أنماط العنونة المتاحة للنفاذ إلى الذاكرة الداخلية في المتحكم 8051؟

نمط العنونة المباشرة، ونمط العنونة غير المباشرة.

4. ما هي أنماط العنونة المتاحة للنفاذ إلى ذاكرة البرنامج في المتحكم 8051؟

نمط العنونة بالدليل

5. ما هي أنماط العنونة المتاحة للنفاذ إلى ذاكرة المعطيات الخارجية في المتحكم 8051؟

نمط العنونة غير المباشرة.

6. ما هو الفرق بين التعليمات الثلاث التالية MOV و MOVC و MOVX في المتحكم 8051؟

تستخدم التعليمة MOV للنقل بين موقعين في الذاكرة الداخلية.

تستخدم التعليمة MOVC لقراءة موقع من ذاكرة البرنامج.

تستخدم التعليمة MOVX للنفاذ إلى ذاكرة المعطيات الخارجية.

7. ما هو الفرق بين تعليمات القفز التالية: SJMP و AJMP و LJMP.

SJMP للقفز اللاشرطي النسبي القصير حيث يكون الانزياح مرمزاً على 8 بت بصيغة المتمم الثنائي.
AJMP للقفز اللاشرطي النسبي القصير حيث يكون انزياح مرمزاً عى 11 بت بصيغة المتمم الثنائي.
LJMP للقفز اللاشرطي المطلق إلى الموقع المحدد والمرمز على 16 بت.

8. ما هو الفرق بين تعليمات الاستدعاء التالية: ACALL و LCALL.

ACALL لطلب إجرائية بموقع نسبي مرمز على 11 بت بصيغة المتمم الثنائي.
LCALL لطلب إجرائية بموقع مطلق مرمز على 16 بت.

9. ما هو محتوى المراكم بعد تنفيذ التعليمات التالية في المتحكم 8051؟

```
MOV  30,#10
MOV  A,#30
ADD  A,30
```

الجواب: A=40

10. ما هو محتوى المراكم بعد تنفيذ التعليمات التالية في المتحكم 8051؟

```
MOV  40h,#17h
MOV  R0,#40h
MOV  A,@R0
```

الجواب: A=17h

11. ما هو محتوى المراكم بعد تنفيذ التعليمات التالية في المتحكم 8051؟

```
MOV  A,#10
ADD  A,#10
```

الجواب: A=20

12- ما هو محتوى المراكم بعد تنفيذ التعليمات التالية في المتحكم 8051؟

```
MOV  10,#10
SUB  A,10
```

الجواب: A=0

12. ما هو محتوى المراكم A والسجل B بعد تنفيذ التعليمات التالية في المتحكم 8051؟

```
MOV  A,#100
ADD  B,#10
MUL  AB
```

الجواب: A=E8h، B=3 وذلك لأن 10*100=1000=03E8h.

13. ما هو محتوى المراكم بعد تنفيذ التعليمات التالية في المتحكم 8051؟

```
MOV  A,#245
MOV  B,#10
DIV  AB
```

الجواب: A=24، B=5 وذلك لأن A يحتوي على ناتج القسمة الصحيح و B يحتوي على باقي القسمة.

أسئلة خيارات متعددة

1. هل يمكن النفاذ إلى سجلات الوظائف الخاصة بشكل غير مباشر؟

A. نعم

B. لا

2. ما هو قيمة السجل SP بعد تعليمة PUSH ACC بفرض أن قيمته قبل تنفيذ التعليمة هي SP=50h.

A. 4Fh

B. 49h

C. 52h

D. 51h

3. ما هو سجل مؤشر المكدس؟

A. PC

B. B

C. SP

D. PSW

4. ما هو أكبرحجم برنامج يمكن للمتحكم أن ينفذه؟

A. 4KB

B. 16KB

C. 64KB

D. 68KB

5. ما هو محتوى المراكم بعد تنفيذ التعليمة ORL A,#05h بفرض أن المراكم يحتوي على القيمة A=50h قبل العملية؟

A. 50h

B. 55h

C. 05h

D. A0h

مسائل

1. اكتب برنامجاً جزئياً يدعى Sum16 لجمع قيمتين كل منهما على 16 بت، حيث تكون القيمة الأولى مخزنة في السجلين R3 R2 والقسمة الثانية في السجلين R5, R4 وتخزين النتيجة في السجلين R7, R6.

```
Sum16:    MOV  A,R2
          ADD  A,R4
          MOV  R7,A
          MOV  A,R3
          ADDC A,R5
          MOV  R7,A
          RET
```

2. اكتب برنامجاً جزئياً يدعى Sum10 لجمع عشر قيم في الذاكرة مخزنة في المواقع من 30h إلى 39h ووضع النتيجة في السجلين R3, R2.

```
Sum10:
          MOV  R2, #0
          MOV  R3,#0
          MOV  R7,#10
```

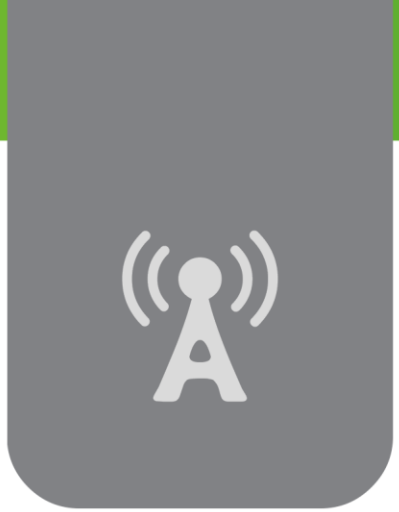
```
        MOV   R0,#30h
Loop:
        MOV   A,R2
        ADD   A,@R0
        MOV   R2,A
        CLR   A
        ADDC A, R3
        MOV   R3, A
        INC    R0
        DJNZ  R7,Loop
        RET
```

3. احسب زمن تنفيذ التعليمات التالية مقدراً بعدد دورات الآلة

```
        MOV   R7,#10        ;T1=1
Loop:  MOV   R6,#5         ;T2=1
        NOP                 ;T3=1
        DJNZ  R6, $         ;T4=2
        DJNZ  R7,Loop       ;T5=2
```

T=T1+10*(T2+T3+10*T4+T5)=1+10*(1+1+2*5+2)=141 Cycles/

رقم التمرين	الإجابة الصحيحة
1	
2	D
3	C
4	C
5	B



الفصل العاشر:
المقاطعات والمؤقتات في المتحكم 8051

الكلمات المفتاحية:

المقاطعات في المتحكم 8051، المؤقتات الداخلية، أنماط عمل المؤقت.

ملخص:

يهدف هذا الفصل إلى التعرف أنواع المقاطعات في المتحكم 8051 و آلية عمل المقاطعات عند حدوثها. كما نبين أيضاً كيفية تفعيل المقاطعات والتحكم بأولوياتها وكتابة إجرائيات تخديم المقاطعات. كما نتعرف في هذا الفصل إلى طرفية مهمة وضرورية في معظم التطبيقات ألا وهي المؤقت. سنتعرف في هذا الفصل على المؤقتات الموجودة في المتحكم المدروس وأنماط العمل المختلفة للمؤقت والمقاطعة المرتبطة به. سوف نبين من خلال الأمثلة المدروسة كيفية كتابة البرامج التي تستخدم المؤقتات والمقاطعات في تنفيذ المهمات المطلوبة.

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

- مصادر المقاطعة في المتحكم 8051
- آلية توجيه المقاطعات في المتحكم
- طريقة الاستجابة للمقاطعات
- المؤقتات الداخلية في المتحكم وأنماط عملها
- مقاطعات المؤقت واستخدامها في كتابة البرامج

الفصل العاشر: المقاطعات والمؤقتات في المتحكم 8051

يهدف هذا الفصل إلى التعرف أنواع المقاطعات في المتحكم 8051 وآلية عمل المقاطعات عند حدوثها. كما نبين أيضاً كيفية تفعيل المقاطعات والتحكم بأولوياتها وكتابة إجرائيات تخديم المقاطعات. كما نتعرف في هذا الفصل إلى طرفية مهمة وضرورية في معظم التطبيقات ألا وهي المؤقت. سنتعرف في هذا الفصل على المؤقتات الموجودة في المتحكم المدروس وأنماط العمل المختلفة للمؤقت والمقاطعة المرتبطة به. سوف نبين من خلال الأمثلة المدروسة كيفية كتابة البرامج التي تستخدم المؤقتات والمقاطعات في تنفيذ المهمات المطلوبة.

1. المقاطعات:

لا تختلف فلسفة معالجة المقاطعات في المتحكمات كثيراً عن فلسفة معالجة المقاطعات في المعالجات. ويكمن الاختلاف فقط في تعريف مصادرها وتنظيم عملية الاستجابة لها من الناحية البرمجية فقط. لذلك سوف نبين في هذه الفقرة مصادر المقاطعات المختلفة الموجودة في المتحكم 8051 وآلية الاستجابة لهذه المقاطعات.

1.1. مصادر المقاطعة:

يمكن أن يقاطع عمل البرنامج في المتحكم 8051 من قبل خمسة مصادر:

المقاطعات الخارجية:

يوجد للمتحكم مقاطعتان خارجيتان من المربطين ($\overline{INT0}, \overline{INT1}$) من البوابة P3. تحدث المقاطعة الخارجية $\overline{INT0}$ عندما يهبط المربط الموافق من السوية العليا إلى السوية الدنيا. حيث بالإمكان أن تتم المقاطعة عند الجبهة الهابطة فقط. أو أن تبقى هذه المقاطعة ما دام المربط موجوداً على السوية الدنيا. ويمكن الاختيار بين هاتين الامكانيتين من خلال البت IT0 في السجل TCON. وعند حدوث حدث المقاطعة يتم رفع راية المقاطعة IE0 في السجل TCON. وكذلك يكون دور البتين IT1 و IE1 بالنسبة للمقاطعة الخارجية الثانية $\overline{INT1}$.

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

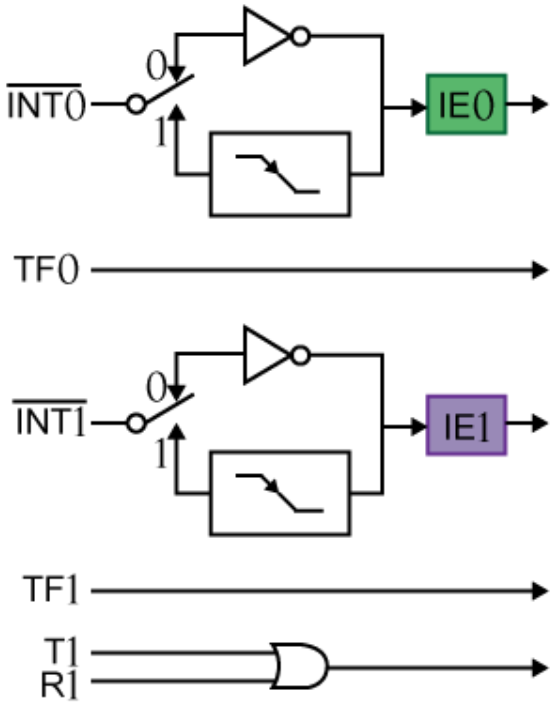
السجل TCON المسؤول عن المقاطعات الخارجية والمؤقتات

مقاطعات المؤقتات:

يوجد في المتحكم مؤقتين Timer0 و Timer1 لكل منهما مقاطعته الخاصة. تحدث مقاطعة المؤقت Timer0 عندما ينتهي هذا المؤقت من العد والانتقال من القيمة FFFFh إلى 0000h. وهذا ما يدعى بفيضان المؤقت (Timer overflow). عند ذلك يتم رفع راية مقاطعة المؤقت TF0 في السجل TCON. وكذلك يكون دور الراية TF1 بالنسبة للمؤقت الثاني Timer1.

مقاطعة من البوابة التسلسلية:

تحدث هذه المقاطعة عند استقبال محرف من البوابة التسلسلية والذي يتم الإعلان عنه برفع الراية RI في السجل SCON (المسؤول عن التحكم بالبوابة التسلسلية) أو عند الانتهاء من إرسال محرف والذي يتم الإعلان عنه برف الراية TI من نفس السجل.



مصادر المقاطعة في المتحكم 8051

يمكن تفعيل أو عدم تفعيل هذه المقاطعات الخمسة بشكل مستقل من خلال سجل خاص بتفعيل المقاطعات وهو السجل IE المبين فيما يلي:

7	6	5	4	3	2	1	0
EA	-	-	ES	ET1	EX1	ET0	EX0

السجل IE المسؤول عن تفعيل المقاطعات

حيث يتم وضع البت EX0 على قيمة 1 عند تفعيل المقاطعة الخارجية الأولى أو على قيمة 0 لإلغاء تفعيل هذه المقاطعة. وبنفس الطريقة بالنسبة للمقاطعة الخارجية الثانية من خلال بت التفعيل EX1. أما البتين ET0 و ET1 فهما مسؤولان عن تفعيل مقاطعة المؤقتين الأول والثاني على الترتيب. وكذلك البت ES لتفعيل مقاطعة البوابة التسلسلية.

عند استخدام أية مقاطعة يجب وضع البت EA على القيمة 1 وإلا فلن يتم تفعيل أية مقاطعة.

حيث يستخدم هذا البت لإلغاء تفعيل جميع المقاطعات دفعة واحدة دون تغيير بتات التفعيل الخاصة بكل مقاطعة. ونحتاج أحياناً لمثل هذا الأمر عند تنفيذ مهمة حرجة لا نريد فيها أن يتم مقاطعتنا من قبل أية مقاطعة. لذلك يتم تصفير البت EA عند بداية المهمة الحرجة، وإعادته على القيمة 1 بعد الانتهاء من المهمة الحرجة ليعود الوضع على ما كان عليه قبل بدء المهمة.

2.1. جدول متجهات المقاطعة:

عند حدوث مقاطعة فإن البرنامج لا يقوم بشحن عداد البرنامج بقيمة مخزنة في الذاكرة كما هي الحال في المعالج 8086 بل يقفز مباشرة للتنفيذ إلى مكان محدد في ذاكرة البرنامج. يبين الجدول التالي العنوان الذي سيقفز إلية البرنامج عند حدوث أي من المقاطعات.

Interrupt	Program	Priority
INT0	0003h	1 (high)
Timer 0	000Bh	2
INT1	0013h	3
Timer 1	001Bh	4
Serial Port	00023h	5 (low)

جدول متجهات المقاطعات وأولوياتها الافتراضية.

ولما كان المجال المتاح لكل مقاطعة لا يزيد عن 8 بايتات والذي قد لا يكفي في معظم الأحيان لكتابة إجرائية المقاطعة، فإننا نعيد توجيه المقاطعة إلى مكان آخر باستخدام تعليمة القفز كما هو موضح فيما يلي:

```
ORG  0000h
AJMP Start
ORG  0003h
AJMP ISR_Ext0
ORG  000Bh
AJMP ISR_Timer0
ORG  0013h
AJMP ISR_INT1
ORG  001Bh
AJMP ISR_Timer1
ORG  0023h
AJMP ISR_Serial
```

كما يبين الجدول أولويات هذه المقاطعات، فعند حصول مقاطعتين في نفس الوقت فإن المتحكم سيستجيب أولاً للمقاطعة ذات الأولوية الأعلى. ولكن يوفر المتحكم إمكانية تغيير هذه الأولويات باستخدام سجل الأولويات IP المبين في الشكل التالي:

7	6	5	4	3	2	1	0
–	–	–	PS	PT1	PX1	PT0	PX0

السجل IP المسؤول عن أولويات المقاطعات

يمكن تعريف سويتين من الأولوية. أولوية عليا وأولوية دنيا. يمنكن وضع أية مقاطعة في السوية العليا من خلال وضع البت الموافق في السجل IP على قيمة 1. أما إذا وضعنا البت الموافق على قيمة 0 فإننا سوف نضع هذه المقاطعة في السوية الدنيا. عند حصول مقاطعتين من نفس السوية في نفس الوقت، فإن المتحكم سوف يستجيب للمقاطعة ذات الأولوية الثابتة الأعلى المبينة في الجدول السابق.

3.1. آلية عمل المقاطعة:

يستجيب المتحكم 8051 للمقاطعات بنفس الطريقة التي يستجيب بها المعالج 8086. لنأخذ على سبيل المثال، المقاطعة الخارجية INT0. فعند حصول هذه المقاطعة، يقوم المتحكم بإنهاء التعليمة الحالية قيد التنفيذ ثم يقوم بتخزين عداد البرنامج PC في ذاكرة المكدس في موقعين متتاليين. ثم يقوم المتحكم بالقفز إلى العنوان 0003h في ذاكرة البرنامج حسب جدول متجهات المقاطعة. ويبدأ المتحكم بتنفيذ إجرائية المقاطعة حتى الوصول إلى تعليمة العودة RETI وعندها يقوم المتحكم باستعادة قيمة عداد البرنامج من المكدس وبالتالي العودة إلى المكان الذي كان فيه قبل حصول المقاطعة.

عند استجابة المتحكم لمقاطعة ما فإنه يقوم أيضاً في بعض الأحيان بتصفير راية المقاطعة الموافقة. وينطبق هذا على المقاطعات الخارجية عند تفعيلها على الجبهة الهابطة، كذلك رايات مقاطعة المؤقتات. أما في حالة المقاطعة الخارجية المفعّلة على السوية الدنيا وكذلك مقاطعة البوابة التسلسلية، فإن المتحكم لا يقوم بتصفير الرايات الموافقة بشكل تلقائي، بل يجب القيام بذلك بشكل برمجي ضمن إجرائية الاستجابة للمقاطعة.

4.1. مثال عن برمجة المقاطعة:

نريد كتابة برنامج يقوم عن طرق المقاطعة بقلب حالة المربط P2.0 عند حصول المقاطعة الخارجية INT0 والتي تحدث عند ورود جبهة هابطة على المربط $\overline{INT0}$.

في البداية يجب في بداية البرنامج برمجة المقاطعة الخارجية INT0 لتعمل على الجبهة الهابطة من خلال برمجة السجل TCON. سوف نعتبر أن لهذه المقاطعة الأولوية العليا ولذلك نضع البت الموافق في السجل IP على قيمة 1. ومن ثم يجب تفعيل هذا المقاطعة خلال السجل تفعيل البت الخاص بها والبت EA في السجل IE.

ولكتابة برنامج الاستجابة للمقاطعة، يجب توجيه هذه المقاطعة في العنوان الخاص بها في ذاكرة البرنامج في العنوان 0003h نحو إجرائية الاستجابة لخاصة بها التي سندعوها ISR_INT0 باستخدام تعليمة القفز.

```
ORG  0000h
     AJMP Start

ORG  00003h
     AJMP ISR_INT0

ISR_INT0:
     PUSH PSW
     CPL  20h.0
     MOV  C, 20h.0
     MOV  P2.0, C
     POP  PSW
     RETI

Start:
     MOV  TCON, #01h
     MOV  IP, #01h
     MOV  IE, #81h
     CLR  20h.0
Main:
     …
     …
```

...

AJMP Main

أما في إجرائية الاستجابة للمقاطعة فإننا نقوم أولاً بتخزين سجل إحالة في المكدس باستخدام تعليمة PUSH ثم نقوم بقلب حالة بت حالة الخرج الموجود في الموقع 20h.0 (أي البت 0 من البايت 20h) باستخدام تعليمة CPL. ومن ثم يتم كتابة هذا البت على مربط الخرج P2.0 عن طريق بت الحمل C الذي يقوم بدور المراكم بالنسبة للبتات (حيث لا يدعم المعالج نقل بت إلى بت مباشرةً دون المرور بالبت C). في النهاية نستعيد قيمة سجل المكدس باستخدام تعليمة POP وأخيراً العودة من إجرائية المقاطعة إلى البرنامج الأساسي باستخدام التعليمة RETI.

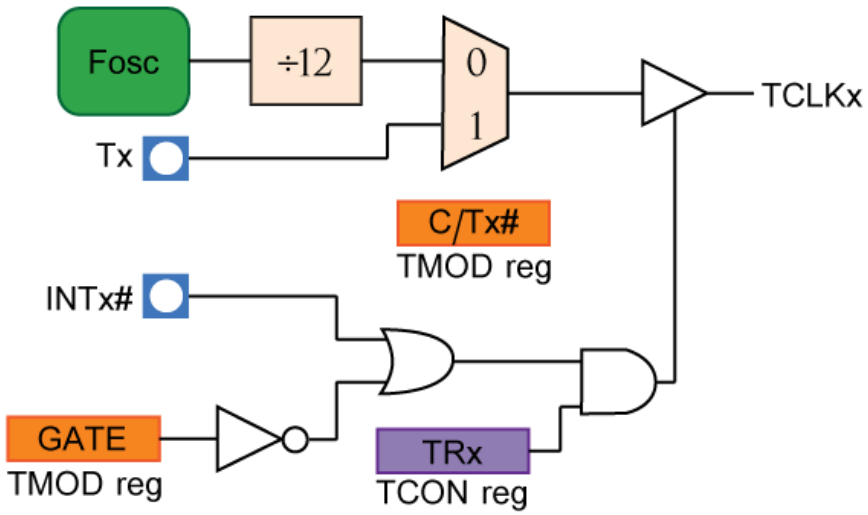
2. المؤقتات:

يحتوي المتحكم 8051 على مؤقتين/عدادين مستقلين ومتماثلين هما Timer0 و Timer1 كل منهما على 16 بت. ويمثل السجلين THx و TLx إلى البايت الأعلى والبايت الأدنى لقيمة المؤقت، حيث يرمز x لرقم المؤقت: 0 بالنسبة للمؤقت Timer0 و 1 بالنسبة للمؤقت Timer1.

يعمل المؤقت وفق عدة أنماط عمل وله مجموعة من سجلات الوظائف الخاصة التي تحدد نمط عمله وتتحكم بعمله وهي السجلات TCON و TMOD وسوف نتعرف فيما يلي أنماط العمل المختلفة للمؤقت ووظيفة بتات سجلات.

1.2. ساعة عمل المؤقت:

يعمل المؤقت انطلاقاً من ساعة داخلية يتم توليدها انطلاقاً من الساعة الداخلية للمتحكم بعملية تقسيم على 12، أو من ساعة خارجية يتم تزويدها للمتحكم عن طريق المربط T0 بالنسبة للمؤقت Timer0 أو T1 بالنسبة للمؤقت Timer1. وعند تشغيل المؤقت انطلاقاً من ساعة خارجية فإننا نسمي المؤقت عندئذٍ "عداد" (Counter). ويمكن اختيار مصدر ساعة عمل المؤقت إما داخلية أو خارجية عن طريق أحد بتات السجل TMOD وهو البت T/C. يبين الشكل التالي المخطط الصندوقي لانتخاب ساعة عمل المؤقت.



المخطط الصندوقي لساعة عمل المؤقت

ولكي يعمل المؤقت يجب أن يكون بت التشغيل TRx في السجل TCON على القيمة 1. بالإضافة إلى ذلك يجب أن يتحقق أيضاً أحد أمرين لكي يعمل المؤقت وهما:

1. أن يكون البت GATE=0 في السجل TMOD.
2. إذا كانت قيمة GATE=1 فيجب أن تكون قيمة المربط INTx على القيمة المنطقية العليا أي 1. وفي هذه الحالة نستطيع تشغيل أو إيقاف المؤقت عن طريق إشارة خارجية يتم وصلها إلى المربط INTx. ولهذا العمل بعض التطبيقات التي نستخدم فيها المؤقت لقياس العرض الزمني لنبضة خارجية.

2.2. أنماط عمل المؤقت:

يعمل المؤقت وفق أربعة أنماط عمل حيث يمكن انتخاب نمط العمل عن طريق بتين M1, M0 الموجودين في السجل TMOD وذلك وفق الجدول التالي:

M1 M0	Timer Mode
0 0	Mode 0
0 1	Mode 1
1 0	Mode 2
1 1	Mode 3

انتخاب نمط عمل المعالج.

يبين الشكل التالي السجل TMOD وتوزع بتاته، حيث يُقسم إلى قسمين: البتات الأربعة الدنيا للتحكم المؤقت Timer0 والبتات الأربعة العليا للتحكم بالمؤقت Timer1.

7	6	5	4	3	2	1	0
GATE1	C/T1	M11	M01	GATE0	C/T0	M10	M00

السجل TMOD

كما يبين الشكل التالي السجل TCON ومكان وجود بتات التشغيل المؤقت TRx الموافقة لكل مؤقت. أما البتات TFx فهي رايات حدوث مقاطعة المؤقت التي سنأتي على شرحها لاحقاً.

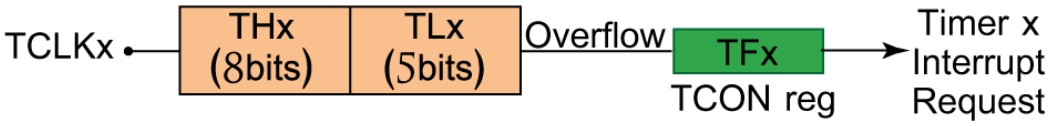
7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT1

السجل TCON

1.2.2. نمط العمل 0: مؤقت على 13 بت:

في نمط العمل 0 (Mode 0) يعمل المؤقت على 13 بت: 8 بتات في السجل THx وخمسة بتات في السجل TLx وهي البتات العليا منه، أما البتات الثلاثة الدنيا من TLx فهي غير مستخدمة في هذا النمط.

يمثل الشكل التالي المخطط الصندوقي للمؤقت في النمط 0.



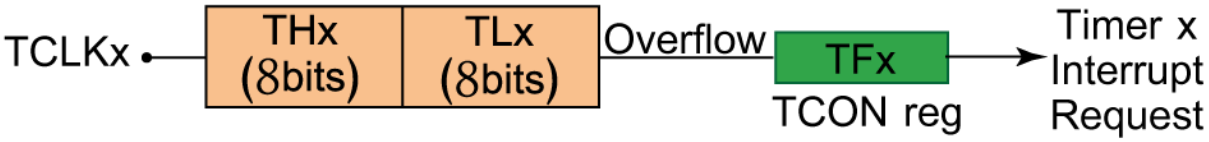
نمط عمل المؤقت على 13 بت

عندما يتم تشغيل المؤقت، يبدأ المؤقت بالعد التصاعدي انطلاقاً من قيمته الابتدائية عند كل نبضة ساعة. وعندما يصل إلى القيمة العظمى وهي FF F8h فإن نبضة ساعة جديدة سوف تجعله يعود إلى الصفر. عندئذٍ، نقول أن المؤقت قد فاض بالعد. عند حصول ذلك يتم رفع راية مقاطعة المؤقت TFx في السجل TCON. فإذا كانت مقاطعة المؤقت مؤهلة، فإنه سيتم مقاطعة عمل البرنامج الأساسي والقفز نحو إجرائية معالجة مقاطعة المؤقت كما رأينا سابقاً عند مناقشة المقاطعات. من الجدير بالذكر أن عند الاستجابة لمقاطعة المؤقت فإن المتحكم يقوم بتصفير راية مقاطعة المؤقت TFx بشكل آلي.

لا يتم استخدام هذا النمط عادةً لأنه مشمول بالنمط 1 الذي سنعرضه في الفقرة المقبلة، ولقد تم وضع هذا النمط للحفاظ فقط على التوافق مع النسخ السابقة للمتحكم.

2.2.2. نمط العمل 1: مؤقت على 16 بت:

يعمل المؤقت في النمط 1 (Mode 1) بشكل مماثل لعمله في النمط 0، إلا أن الفرق الوحيد يكمن في أن المؤقت يعمل على 16 بت. ويحدث الفائض عندما ينتقل المؤقت من القيمة العظمى FF FFh إلى الصفر. يوضح الشكل التالي المخطط الصندوقي لعمل المؤقت في هذا النمط.

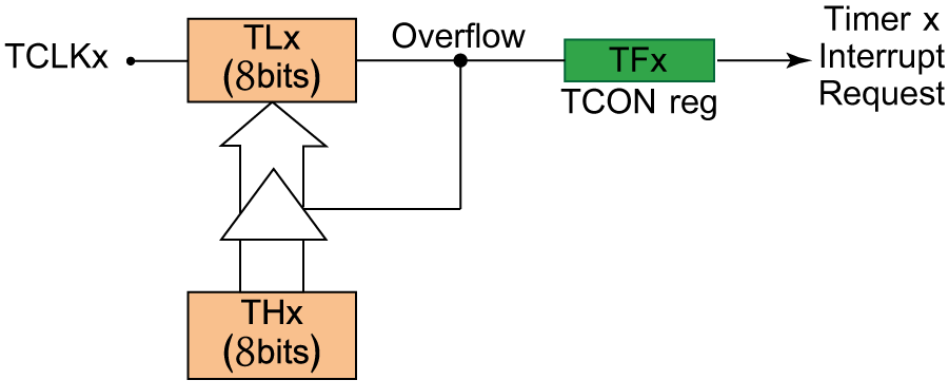


نمط عمل المؤقت على 16 بت

يستخدم هذا النمط عند الرغبة في الحصول على أزمنة مقاطعة طويلة نسبياً تصل حتى زمن 2^{16}=65536 دورة آلة.

3.2.2. نمط العمل 2: مؤقت على 8 بت مع إعادة شحن تلقائي:

في نمط العمل 2 (Mode 2) يعمل المؤقت على 8 بت فقط، ويستخدم السجل TLx للعد. وعندما تبلغ قيمة المؤقت القيمة العظمة FFh فإنه لا يعود إلى الصفر وإنما يقوم بشحن قيمة السجل THx في السجل TLx بشكل تلقائي. وعندئذٍ يرفع المؤقت راية المقاطعة TFx.



نمط عمل المؤقت على 8 بت مع إعادة شحن تلقائي

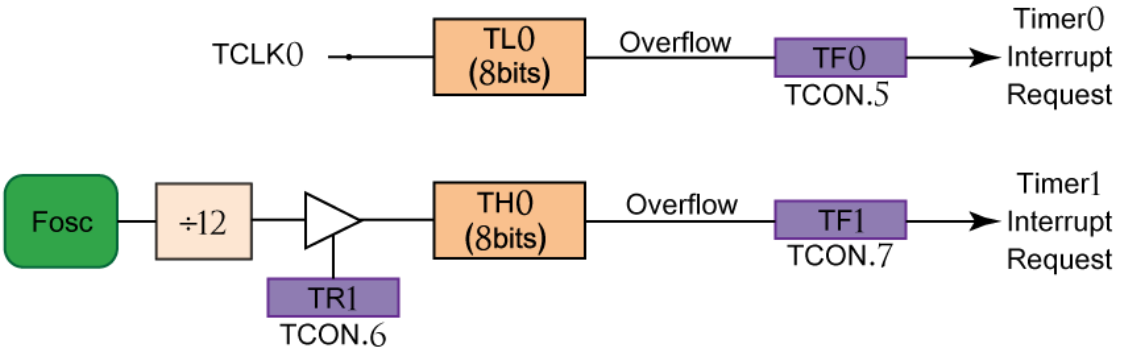
يسمح هذا النمط من العمل بالحصول على مقاطعات منتظمة في الزمن بفاصل دقيق يعتمد على قيمة إعادة الشحن التلقائي. أما في بقية الأنماط فإن عملية إعادة شحن المؤقت بقيمة ما يتم برمجياً ضمن إجرائية المقاطعة. ولما كان زمن الذهاب للمقاطعة يتغير حسب زمن تنفيذ التعليمة الحالية، فإن الزمن الفاصل بين المقاطعات يتغير بمقدار عدة دورات آلة حسب التعليمة التي تكون قيد التنفيذ عند حدوث فيضان المؤقت.

يستخدم هذا النمط عادة للمؤقت Timer1 عند استخدامه كمولد لسرعة تراسل البوابة التسلسلية التي سنراها في الفصل القادم، مما يسمح بضبط سرعة التراسل بشكل دقيق عند استخدام السرعات العالية للتراسل.

4.2.2. نمط العمل 3: مؤقتين منفصلين على 8 بت:

يستخدم نمط العمل 3 (Mode 3) للمؤقت Timer0 فقط ولا يمكن استخدامه للمؤقت Timer1. في هذا النمط يتم فصل المؤقت Timer0 إلى مؤقتين مستقلين كل منهما على 8 بت. حيث يستخدم السجل TL0 للمؤقت الأول والسجل TH0 للمؤقت الثاني.

يأخذ المؤقت الأول TL0 ساعته إما من الساعة الداخلية أو من المربط الخارجي مثل الأنماط السابقة، ويرفع هذا المؤقت راية المقاطعة TF0 عند حصول الفائض. أما المؤقت الثاني فإنه يعمل على الساعة الداخلية حصراً، ويستخدم راية المؤقت TF1 (التابعة للمؤقت Timer1) للإبلاغ عن حصول مقاطعة كما ويستخدم بت التشغيل TR1 لتشغيل وإيقاف المؤقت TH0. وهذا يعني أنه عند استخدام المؤقت Timer0 في هذا النمط لا يمكننا استخدام مقاطعة للمؤقت Timer1 وعدم القدرة على التحكم بتشغيله وإيقافه عن طريق بت التشغيل TR1. ولكن نستطيع إيقاف عمل المؤقت Timer1 بوضعه في نمطا العمل 3، مما يسبب وقف تشغيله لأن هذا النمط لا يطبق على المؤقت Timer1.



نمط عمل المؤقت Timer0 كمؤقتين منفصلين على 8 بت

يستخدم هذا النمط من العمل عند الرغبة بالحصول على مؤقت إضافي للحصول على 3 مؤقتات بدل 2. ولكن ذلك قليل الاستخدام لأنه هناك متحكمات من نفس العائلة مثل المتحكم 8052 والذي يملك 3 مؤقتات مستقلة كل منها على 16 بت.

3.2. مثال عن عمل المؤقت:

لتوضيح كيفية استخدام المؤقت، سوف نأخذ مثالاً بسيطاً نريد فيه توليد نبضة بعرض قدره 10ms على أحد المرابط الخارجية للمتحكم وليكن P1.0 بمعدل نبضة كل 100ms. في هذا المثال، نفرض أن المتحكم يعمل باستخدام ساعة عمل ذات تردد قدره Fosc=12 MHz.

في هذا المثال سوف نستخدم المؤقت Timer0 لتوليد قاعدة زمنية مقدارها 1ms أي الحصول على مقاطعات دورية بفاصل زمني 1 ms. سوف نستخدم هذه القاعدة الزمنية لكتابة إجرائية تأخير زمني تدعى Delay_ms تقوم بتأخير لفترة ما مقدرة بالميللي ثانية، حيث أن فترة التأخير تعطى كمعامل للإجرائية يتم تمريرها عبر أحد السجلات. سوف نستخدم هذه الإجرائية لتوليد النبضة الدورية حسب الأزمنة المعطاة.

سوف يتم تشغيل المؤقت باستخدام الساعة الداخلية، وبالتالي فإن تردد ساعة العمل هو:

$$CLK0 = \frac{Fosc}{12} = \frac{12MHz}{12} = 1MHz$$

وبالتالي فإن المؤقت سوف يقوم بالعد كل زمن دورة آلة أي كل 1 µs. من أجل الحصول على زمن قدره 1 ms= 1000 µs يجب أن يقوم المؤقت بعد 1000 قيمة قبل حصول الفائض. من ذلك نستنتج أنه يمكننا استخدام المؤقت في النمط 0 أو النمط 1 لكي نتمكن من العد حتى القيمة 1000 والتي تحتاج إلى أكثر من 8 بت لتمثيلها. نختار هنا أن يعمل المؤقت في النمط 1.

في نمط العمل 1، يفيض المؤقت عندما ينتقل من القيمة العظمى FFFFh إلى الصفر. ومن ذلك نستنتج أن القيمة الابتدائية للمؤقت يجب أن تكون:

[TH0 TL0]=65536−1000=64536= FC18h

تعليمات الاستهلال:

في بداية البرنامج الأساسي، لا بد من وضع القيم الابتدائية لسجلات الوظائف الخاصة وخصوصاً مؤشر المكدس SP وسجلات التحكم بالمؤقتات والمقاطعات.

SP= 60h

IE= 82h;

IP= 02h;

TH0= 0FCh

TL0= 18h

TMOD= 01h

TCON= 10;

سوف نختار أن تبدأ ذاكرة المكدس انطلاقاً من العنوان 60h وذلك يعني أننا سنقوم باستخدام مواقع الذاكرة من 60h إلى 7Fh (نهاية الذاكرة) للمكدس. من ثم نريد تفعيل المقاطعة الخاصة بالمؤقت من خلال السجل IE، لذلك نقوم بوضع البت الموافق لمقاطعة المؤقت Timer0 في السجل IE على قيمة الواحد بالإضافة إلى بت التفعيل العام EA. ونفترض أن هذه المقاطعة ذات أولوية عليا لذلك قمنا بوضع بت الأولوية الموافق في السجل IP على قيمة الواحد مع أن هذا غير ضروري في مثالنا هذا إذ لم نقم بتفعيل سوى مقاطعة واحدة، فلا داعي لتعريف الأولويات.

ننتقل الآن إلى المؤقت، حيث قمنا بوضع قيمته الابتدائية التي قمنا بحسابها في السجلين TH0 و TL0. ومن ثم قمنا باختيار نمط العمل 1 للمؤقت Timer0 الذي سيعمل من الساعة الداخلية في نمط مؤقت وليس عداد (C/T0=0) وذلك من خلال وضع قيمة السجل TMOD على القيمة الموافقة. وأخيراً لتشغيل المؤقت يجب وضع البت TR0=1 في السجل TCON وهذا ما يعطي القيمة الابتدائية لهذا السجل.

```
PULSE       EQU  P1.0
F1ms        EQU  20h.0

ORG 0000h          ;Beginning of the Program
        AJMP Start
ORG 000Bh          ; Direct Timer0 interrupt to  ISR_Timer0
        AJMP ISR_Timer0
;============================
ISR_Timer0:        ; Interrupt Service Routine for Timer 0
        MOV  TH0,#0FCh
        MOV  TL0,#18h
        SETB F1ms
        RETI
;============================
Delay_ms:
Loop:
        CLR  F1ms
        JNB  F1ms,$
        DJNZ R7,Loop
        RET
```

```
;============================
Start:
        MOV   SP,#60h
        MOV   IE,#82h
        MOV   IP,#82h
        MOV   IP,#02h
        MOV   TH0,#0FCh
        MOV   TL0,#18h
        MOV   TMOD, #01h
        MOV   TCON,#10h
Loop:
        SETB  PULSE         ;PULSE=1
        MOV   R7,#10                ;Delay for 10 ms
        ACALL     Delay_ms
        CLR   PULSE         ;PULSE=0
        MOV   R7,#90        ;Delay for 90 ms
        ACALL     Delay_ms
        AJMP  Loop
END.
```

إجرائية المقاطعة:

في إجرائية المقاطعة يجب إعادة وضع القيمة الابتدائية للعداد الموافقة للزمن المطلوب. لذلك يجب إعادة شحن السجلات TH0 و TL0 بالقيم الابتدائية. ومن ثم نضع قيمة أحد البتات ويدعى F1ms على قيمة الواحد والذي سوف يلعب دور راية لإعلام البرنامج بحدوث مقاطعة المؤقت. وعند انتهاء إجرائية المقاطعة نعود إلى البرنامج الأساسي باستخدام التعليمة RETI.

إجرائية التأخير بالاعتماد على المؤقت:

بدل الاعتماد على حلقات التأخير لانجاز تأخير زمني معين كما رأينا سابقاً، سوف نستخدم هنا المؤقت لكتابة إجرائية التأخير. نفترض أن مقدار التأخير الذي نريده موجود في السجل R7.

في البداية يتم تصفير الراية F1ms ثم نقوم بتنفيذ حلقة تراقب الراية F1ms وتنتظر ما دامت الراية على الصفر باستخدام التعليمة (JNB F1ms, $) حيث يدل الرمز $ على عنوان نفس التعليمة. فعند رفع هذه الراية يتم الخروج من حلقة الانتظار و يتم إنقاص السجل R7 بمقدار واحد، وتتكرر هذه العملية حتى تصبح قيمة السجل R7 إلى الصفر. عندئذٍ نخرج من التابع بتعليمة RET.

البرنامج الأساسي:

نسمي مربط الخرج P1.0 في هذا البرنامج PULSE. في البداية يتم وضع PULSE=1 ومن ثم نطلب إجرائية التأخير الزمني بعد شحن قيمة السجل R7 بالقيمة 10 الأمر الذي يؤدي لتأخير بمقدار 10 ms. ومن ثم يتم وضع المربط PULSE=0 ومن ثم يتم طلب إجرائية التأخير لفترة 90 ms. وتتكرر هذه العملية بشكل دائم مما يؤدي إلى توليد النبضة دورياً.

أسئلة الفصل العاشر

أسئلة عامة

1. ماهو ترتيب أولويات المقاطعات عندما يكون السجل IP=0Ch؟

1. INT1
2. Timer1
3. INT0
4. Timer0
5. Serial.

2. كيف يستجيب المتحكم عند حدوث مقاطعة مفعلة في المتحكم 8051؟

عندما تحدث المقاطعة يقوم المتحكم بإتمام التعليمة قيد التنفيذ ومن ثم يقوم بتخزين سجل عداد البرنامج في المكدس والذهاب إلى موقع إجرائية المقاطعة حسب جدول متجهات المقاطعة، وتنفيذ الإجرائية حتى الوصول إلى تعليمة العودة RETI وعندها يعود المتحكم إلى المكان الذي كان فيه قبل حدوث المقاطعة عن طريق استرجاع قيمة عداد البرنامج من ذاكرة المكدس.

3. ما هو جدول متجهات المقاطعة في المتحكم 8051؟

هو جدول ثابت يحدد موقع إجرائية معالجة المقاطعة في ذاكرة البرنامج وهو كالتالي:

المقاطعة	الموقع
INT0	0003h
Timer0	000Bh
INT1	0013h
Timer1	001Bh
Serial	0023h

4. ما هي المصادر الممكنة لساعة المؤقت Timer0 وكيف يتم انتخاب المصدر.

يمكن تزويد المؤقت بساعة داخلية بتردد Fosc/12 أو تزويده بساعة خارجية من المربط T0 ويتم انتخاب مصدر الساعة عن طريق البت C/T في السجل TMOD.

5. ما هو دور البت GATE في السجل TMOD؟

يستخدم هذا البت لانتخاب طريقة وايقاف المؤقت إما عن طريق البت TRx فقط عندما يكون GATE=0 أو عن طريق إشارة خارجية على المربط INTx عندما يكون GATE=1 و TRx=1.

6. ما هي ميزة عمل المؤقت في النمط 0 كمؤقت على 13 بت مقارنة مع النمط 1 كمؤقت على 16 بت؟

النمط 0 للمؤقت مشمول بالنمط 1 ولكن تم وضعه بهدف التوافق مع الإصدارات السابقة للمتحكم.

أسئلة خيارات متعددة

1. كم عدد مصادر المقاطعة التي يمكن أن تقاطع المعالج 8051؟

A. 2
B. 3
C. 4
D. 5

2. عندما تحدث مقاطعة ما في المعالج 8051 فإن البرنامج يقوم بالقفز إلى:

A. إلى مكان محدد في ذاكرة البرنامج بحسب المقاطعة.
B. إلى العنوان الذي يشير إليه السجل DPTR.
C. إلى العنوان الذي يشير إليه السجل PC.
D. إلى العنوان المخزن في ذاكرة المكدس.

3. نريد تأهيل مقاطعة المؤقت Timer0 مع مقاطعة البوابة التسلسلية Serial، ما هي قيمة السجل IE المناسبة؟

A. IE=12h
B. IE=92h
C. IE=82h
D. IE=12h

4. ما الفائدة من بت التأهيل العام EA في السجل IE؟

A. تفعيل جميع المقاطعات دون الحاجة لتفعيل كل مقاطعة كل على حدة.

B. إزالة تفعيل جميع المقاطعات دون الحاجة لإزالة تفعيل كل مقاطعة كل على حدة.

C. كلا الحالتين السابقتين.

D. تفعيل المقاطعات الخارجية.

5. نريد وضع أولوية المقاطعة الخارجية INT0 و مقاطعة المؤقت Timer1 أعلى أولوية بقية المقاطعات الأخرى فإننا نقوم بوضع قيمة السجل IP على القيمة:

A. IP=09h

B. IP=03h

C. IP=04h

D. IP=18h

6. ماعدد المؤقتات على 16 بت التي يحتوي عليها المتحكم 8051؟

A. 1

B. 2

C. 3

D. 4

7. ما عدد الأنماط التي يمكن أن يعمل عليها المؤقت Timer 0؟

A. 1

B. 2

C. 3

D. 4

8. من هو السجل المسؤول عن انتخاب نمط المؤقت؟

A. TCON

B. TMOD

C. TH1

D. PSW

9. ما هو نمط التشغيل المناسب للمؤقت Timer1 عند استخدامه لتوليد قاعدة زمنية طويلة نسبياً ؟

A. النمط 0

B. النمط 1

C. النمط 2

D. النمط 3

10. يتميز النمط 2 للمؤقت (إعادة الشحن التلقائي) مقارنة مع بقية الأنماط بالتالي:

A. عدم الحاجة لإعادة شحن المؤقت عن طريق المقاطعة.

B. سهولة برمجته.

C. كلا الميزتين السابقتين بآن واحد

D. القدرة على الحصول على أزمنة مقاطعة طويلة نسبياً.

مسائل

1. اكتب برنامجاً يقوم بتفعيل المقاطعة الخارجية INT1 على الجبهة الهابطة ويقوم بعد المقاطعات التي تحصل باستخدام عداد يدعى Count على 8 بت وإخراج قيمة العداد على البوابة P1. اكتب أولاً تعليمات الاستهلال التي تحدد قيم السجلات TCON وIE وIP، ثم إجرائية المقاطعة.

```
Count EQU  30h

       MOV  TCON,#00h
       MOV  IE,#84h
       MOV  IP,#04h

ISR_INT1:
       INC   Count
       MOV  P1, Count
       RETI
```

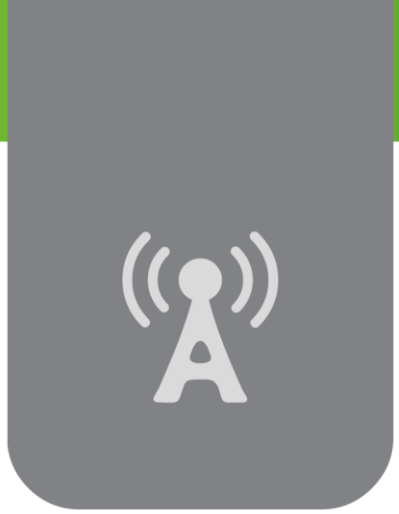
2. اكتب برنامجاً يقوم بتوليد قاعدة زمنية باستخدام المؤقت Timer0 مقدارها T=50ms بفرض أن ساعة المتحكم هي Fosc=12MHz. اكتب أولاً تعليمات الاستهلال ثم إجرائية مقاطعة المؤقت التي تقوم براية تدعى F50ms يتم تعريفها كبت في ذاكرة المتحكم.

```
F50ms       EQU  20h.0

        MOV  TH1,#3Ch
        MOV  TL1,# B0h
        MOV  TMOD,#01h
        MOV  TCON,#10h

ISR_Timer0:
        MOV  TH1,#3Ch
        MOV  TL1,# B0h
        SETB F50ms
        RETI
```

رقم التمرين	الإجابة الصحيحة
1	D
2	A
3	B
4	C
5	A
6	B
7	D
8	B
9	B
10	A



الفصل الحادي عشر:
البوابة التسلسلية في المتحكم 8051

الكلمات المفتاحية:

التراسل التسلسلي الغير متزامن، البوابة التسلسلية، أنماط التراسل التسلسلي.

ملخص:

يهدف هذا الفصل إلى فهم مبدأ التراسل التسلسلي الغير متزامن وفوائده في تحقيق الوصل بين المتحكم والطرفيات الخارجية. بالإضافة إلى ذلك سيتم في هذا الفصل التعرف على البوابة التسلسلية في المتحكم 8051 وأنماط عملها المختلفة وكيفية برمجة سرعة التراسل باستخدام المؤقت. كما نبين المقاطعات التي تولدها البوابة التسلسلية وكيفية استخدامها لتخديم عمليات التراسل في الإرسال أو في الاستقبال.

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

- مبدأ التراسل التسلسلي الغير متزامن
- البوابة التسلسلية في المتحكم
- ضبط سرعة التراسل باستخدام المؤقتات
- مقاطعات البوابة التسلسلية والاستجابة لها

الفصل الحادي عشر: البوابة التسلسلية في المتحكم 8051

يهدف هذا الفصل إلى فهم مبدأ التراسل التسلسلي الغير متزامن وفوائده في تحقيق الوصل بين المتحكم والطرفيات الخارجية. بالإضافة إلى ذلك سيتم في هذا الفصل التعرف على البوابة التسلسلية في المتحكم 8051 وأنماط عملها المختلفة وكيفية برمجة سرعة التراسل باستخدام المؤقت. كما نبين المقاطعات التي تولدها البوابة التسلسلية وكيفية استخدامها لتخديم عمليات التراسل في الإرسال أو في الاستقبال.

1. مقدمه:

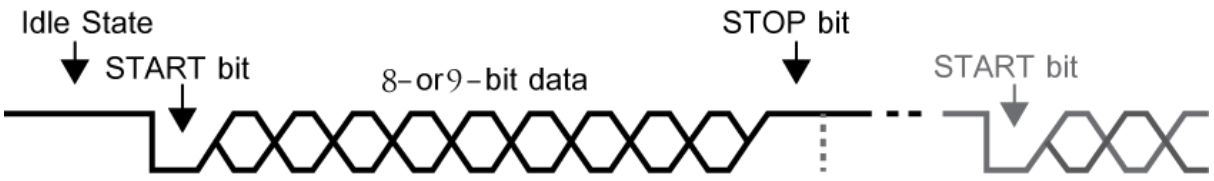
توفر البوابة التسلسلية في المعالج إمكانية التراسل التسلسلي الغير متزامن UART (Universal Asynchronous Receiver Transmitter) الثنائي الاتجاه (Full Duplex) ضمن عدة أنماط عمل بالإضافة إلى نمط من التراسل التسلسلي المتزامن. يوفر التراسل التسلسلي إمكانية نقل المعطيات بين المتحكم والطرفيات الخارجية باستخدام خط وحيد للارسال وخط وحيد للاستقبال، وذلك يساهم في تخفيض الكلفة المادية والتعقيد المطلوب مقارنة مع الربط التفرعي الذي يحتاج إلى مجموعة من الخطوط المتوازية، ولا سيما عندما تكون الطرفية الخارجية بعيدة فيزيائياً عن المتحكم.

2. مبدأ التراسل التسلسلي:

يعتمد مبدأ التراسل التسلسلي على فكرة نقل المعطيات بت تلو الآخر بشكل متسلسل على نفس خط النقل بحيث بفواصل زمنية معينة (والتي عادة ما تكون منتظمة) تبعاً لسرعة نقل المعطيات. ويقوم المستقبل بإعادة تجميع هذه البتات لتصبح بنفس الشكل الموجودة عليه في المرسل. غالباً ما يتم تنظيم المعطيات في الذاكرة أو في المتحكم بوحدة البايت. وليتم نقل هذه المعطيات بشكل تسلسلي لا بد أن يعرف المستقبل البت الأول من البايت والبت الأخير ليقوم بتجميعها معاً ومعالجتها. كما يجب أن يعرف المستقبل لحظة وصول البت الواحد ليتم قراءته وتخزينه.

في التراسل التسلسلي المتزامن يتم معرفة لحظة وصول البت عن طريقة إشارة إضافية تدل على ساعة الإرسال، حيث يمكن تعريف لحظة وصول البت عند وجود جبهة صاعدة لساعة الإرسال. ومن الممكن أيضاً استخدام خط إضافي آخر لمعرفة البت الأول من كل مجموعة أو بداية إطار معطيات (Frame) يتم استقباله. أما في التراسل التسلسلي الغير متزامن فلا يتم إرسال إشارة ساعة أو أية إشارة إضافية مع المعطيات المرسلة تسلسلياً، ولكن يقوم المستقبل بتحديد لحظات ورود البتات وترتيبها من طريقة التراسل. وتسمى طريقة التراسل عادةً بروتوكول التراسل (Protocol)

ففي بروتوكول التراسل التسلسلي الغير متزامن يتم الإرسال حسب الشكل التالي:



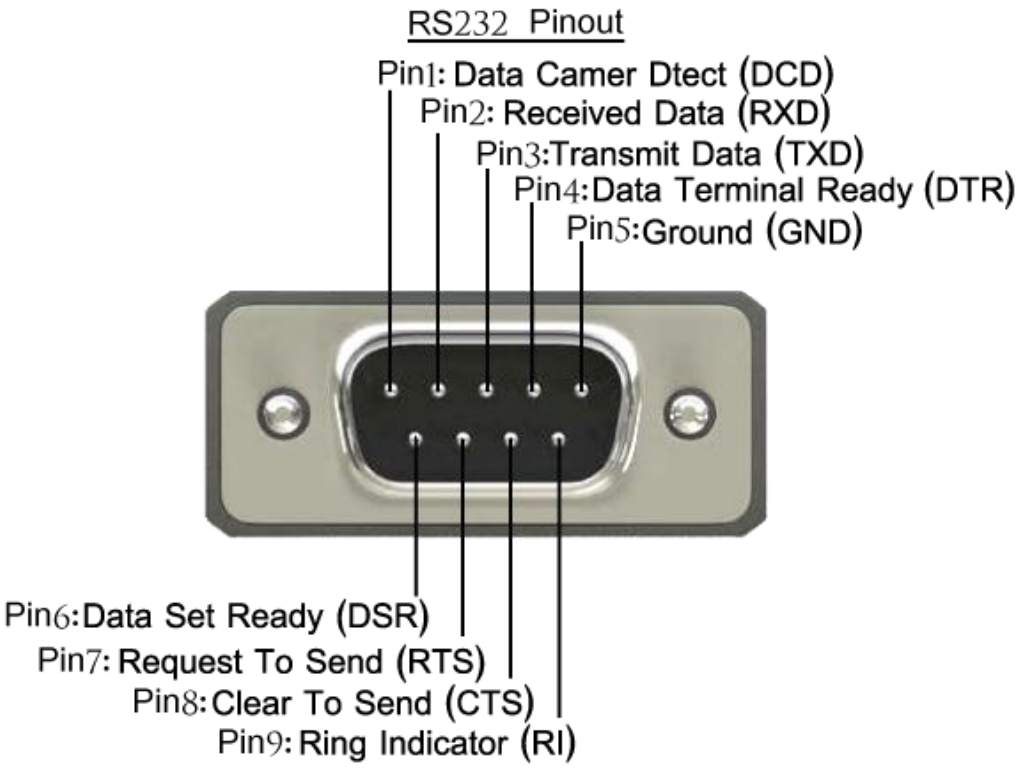
إشارة المعطيات في التراسل التسلسلي غير المتزامن

في حالة عدم الإرسال يكون الخط بدايةً على السوية العليا أو الواحد. وعند إرسال مجموعة من البتات بطول 8 أو 9 بتات يتم أولاً وضع الخط على السوية الدنيا أو الصفر لفترة إرسال زمن البت الواحد ويدعى هذا الرمز بت البداية (Start bit). ثم يتم إرسال البتات الواحد تلو الآخر حسب اتفاق مسبق بين المرسل والمستقبل وعادة ما يتم إرسال البت ذو الدلالة الدنيا أولاً. وهكذا حتى يتم إرسال جميع بتات المجموعة. وأخيراً يتم وضع الخط على السوية العليا من جديد لفترة زمن بت واحد أو أكثر ويدعى هذا الرمز بت التوقف (Stop bit). يبقى الخط على السوية العليا حتى يبدأ إرسال مجموعة أخرى من البتات بنفس الطريقة.

يقوم المستقبل بالتعرف على لحظة بداية إرسال المجموعة عن طريق الجبهة الهابطة لبد البداية. وبما أن المستقبل يعرف زمن إرسال البت والذي نرمز له هنا Tb، فإنه ينتظر عتى فترة Tb/2 ليصل إلى منتصف بت البداية ويتأكد أن الخط ما زال على قيمة الصفر، وبعدها يقوم بقراءة حالة الخط بشكل منتظم وبفواصل زمنية قدرها Tb حتى ينتهي من قراءة مجموعة البتات المرسلة. وبعدها ينتظر المرسل في اللحظة التالية أن يعود الخط إلى الواحد (بت التوقف) ليتأكد من نهاية إرسال المجموعة وإلا فإن هناك خطأ ما قد حصل والذي قد ينجم عادةً بسبب الاختلاف في سرعة التراسل بين المرسل والمستقبل.
لكي يتم التراسل بشكل سليم وفق هذا البروتوكول، يجب أن يعلم المستقبل بزمن إرسال كل بت وعدد البتات المرسلة وإلا فلن يتم التراسل بشكل صحيح.

المعيار التسلسلي RS232:

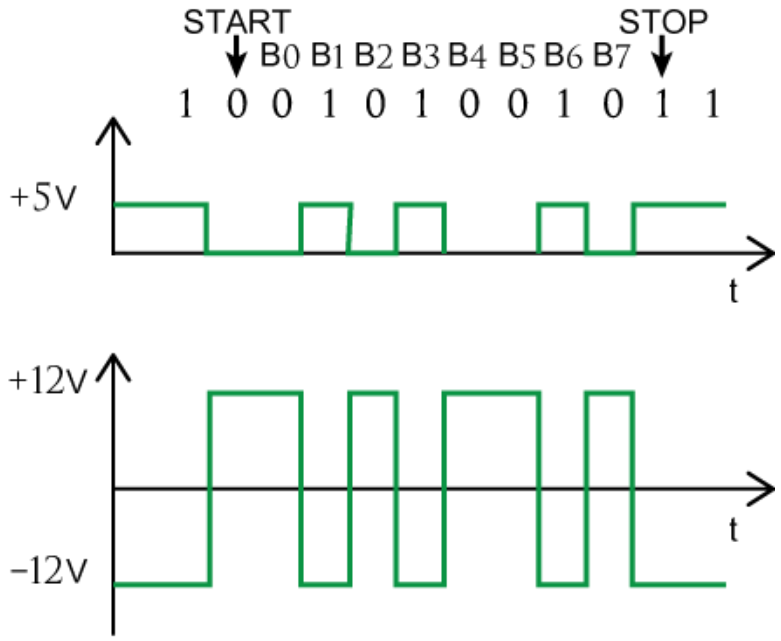
في المعيار التسلسلي RS232 والذي هو شائع الاستخدام في بعض الطرفيات والحواسيب الشخصية، يتم استخدام بروتوكول التراسل التسلسلي الغير متزامن UART. بالإضافة إلى ذلك يتم تعريف مقدار الجهود المعبرة عن السوية الأعلى والأدنى بالإضافة إلى إشارات أخرى تستخدم للمصافحة تخرج عن سياق موضوعنا هنا.



إشارات الوصلة التسلسلية في المعيار RS232

Pin	Signal	Signal Name	DTE Signal direction
1	DCD	Data Carrier Detect	In
2	RXD	Receive Data	In
3	TXD	Transmit Data	Out
4	DTR	Data Terminal Ready	Out
5	GND	Ground	–
6	DSR	Data Set Ready	In
7	RTS	Request to Send	Out
8	CTS	Clear to Send	In
9	RI	Ring Indicator	In

الشكل 1: إشارات الوصلة التسلسلية في المعيار RS232.



جهود الإشارات التسلسلية على مربط المتحكم وفي المعيار RS232

حيث يتم التعبير عن سوية الواحد حسب هذا المعيار بالجهد السالب 12V− وسوية الصفر بالجهد الموجب 12V+. وذلك يعاكس طريقة تمثيل البتات باستخدام الجهود 0V و5V+ التي تمثل الإشارة على مخرج مربط المعالج. وهناك دارات متكاملة متخصصة للتحويل من الجهود 0V, +5V إلى جهود المعيار RS232, +12V −12V مثل الدارة MAX232 من شركة Maxim.

3. البوابة التسلسلية في المتحكم:

يوجد للمتحكم مربطين ضمن البوابة P3 وهما RX والذي يمثل خط الاستقبال، وTX الذي يمثل خط الإرسال. تستخدم البوابة التسلسلية المؤقت Timer1 لتوليد قاعدة زمنية لضبط زمن إرسال البت وبالتالي تحديد سرعة التراسل. حيث يمكن اختيار سرعة التراسل في بعض أنماط عمل البوابة التسلسلية عن طريق برمجة المؤقت بشكل مناسب كما سنرى لاحقاً.

يتم التحكم بعمل البوابة التسلسلية عن طريق سجلات الوظائف الخاصة SBUF وSCON وPCON.
يستخدم السجل SBUF كسجل معطيات الإرسال فعند كتابة قيمة ما في هذا السجل، تقوم البوابة بالبدء بالإرسال. وكما يستخدم أيضاً لتجميع بتات الاستقبال حيث عند اكتمال استقبال كلمة، يمكن أن يقوم المتحكم بقراءتها من هذا السجل. في الحقيقة إن هذا السجل هو في الواقع عبارة عن سجلين منفصلين فيزيائياً: سجل للإرسال وسجل للاستقبال ولكن يتم النفاذ إليهما باستخدام نفس العنوان حيث يتم الكتابة في سجل الإرسال عند الكتابة إلى السجل SBUF كما يتم القراءة من سجل الاستقبال عند قراءة السجل SBUF.

7	6	5	4	3	2	1	0
FE/SM0	SM1	SM2	REN	TB8	RB8	TI	RI

الشكل 2: السجل SCON.

7	6	5	4	3	2	1	0
SMOD1	SMOD0	–	POF	GF1	GF0	PD	IDL

الشكل 3: السجل PCON.

1.3. أنماط عمل البوابة التسلسلية:

تعمل البوابة التسلسلية في المتحكم 8051 ضمن 4 أنماط عمل. يمكن اختيار نمط العمل عن طريق البتين SM0, SM1 في السجل SCON وفق الجدول التالي:

جدول 1: انتخاب نمط عمل البوابة التسلسلية.

SM0 SM1	Operation	Baud Rate
0 0	Mode 0: Shift Register	Fosc/12
0 1	Mode 1: 8 bits UART	Variable
1 0	Mode 2: 9 bits UART	Fosc/32 or Fosc/64
1 1	Mode 3: 9 bits UART	Variable

النمط 0: سجل إزاحة بسرعة محددة

في النمط 0 للبوابة التسلسلية لا يتم فيه استخدام البروتوكول UART للتراسل إذا يتم إرسال مجموعات مكونة من 8 بتات يتم وضعها في السجل SBUF ولكن بدون بت البداية أو بت التوقف. وتكون سرعة الإرسال ثابتة وقدرها Fosc/12 بت بالثانية.

النمط 1: تراسل على 8 بت بسرعة متغيرة

في النمط 1 للبوابة التسلسلية يتم الإرسال باستخدام البروتوكول التسلسلي الغير متزامن، إذ يتم إرسال بت بداية و8 بتات معطيات يتم وضعها في السجل SBUF وبت توقف. وبالتالي إرسال 10 بتات عند إرسال كل بايت. وتكون سرعة الإرسال متغيرة قابلة للبرمجة عن طريق برمجة المؤقت Timer1 وتحدد سرعة الإرسال فيه حسب العلاقة التالية

$$Baud\ Rate = \frac{2^{SMOD1}}{32} \times (Timer1\ Overflow\ Rate)$$

حيث SMOD1 هو بت من السجل PCON ويأخذ إما 0 أو 1. أما (Timer1 Overflow Rate) فهو معدل فيضان المؤقت Timer1 ويعطى عند تشغيل المؤقت في نمط إعادة الشحن التلقائي على 8 بت (النمط 2 للمؤقت) بالعلاقة التالية:

$$\text{Timer1 Overflow Rate} = \frac{\text{Fosc}/12}{256 - [\text{TH1}]}$$

ويُستخدم المؤقت Timer1 غالباً في هذا النمط عند استخدامه لضبط سرعة التراسل وذلك بسبب دقته في تحديد زمن البت.

يبين الجدول التالي قيم الشحن التلقائي للمؤقت من أجل قيم مختلفة لساعات العمل وسرعات التراسل. أما الخانات الفارغة من الجدول فتدل على عدم إمكانية تحقيق سرعة التراسل المطلوبة.

جدول2 : قيم الشحن التلقائي للمؤقت من أجل سرعات تراسل مختلفة.

Baud Rate	Fosc. (MHz)					SMOD1
	11.0592	12	14.7456	16	20	
150	40 h	30 h	00 h			0
300	A0 h	98 h	80 h	75 h	52 h	0
600	D0 h	CC h	C0 h	BB h	A9 h	0
1200	E8 h	E6 h	E0 h	DE h	D5 h	0
2400	F4 h	F3 h	F0 h	EF h	EA h	0
4800		F3 h	EF h	EF h		1
4800	FA h		F8 h		F5 h	0
9600	FD h		FC h			0
9600					F5 h	1
19200	FD h		FC h			1
38400			FE h			1
76800			FF h			1

ومن أجل سرعات التراسل المنخفضة، يمكن تشغيل المؤقت Timer1 في نمط مؤقت على 16 بت، وعندها يكون معدل فيضان المؤقت معطى بالعلاقة:

$$\text{Timer1 Overflow Rate} = \frac{\text{Fosc}/12}{65536 - [\text{TH1:TL1}]}$$

وعند استخدام المؤقت في هذا النمط لضبط سرعة التراسل فإنه يجب تفعيل مقاطعة المؤقت وإعادة شحن المؤقت بالقيم الابتدائية ضمن إجرائية المقاطعة. ولكون سرعة التراسل بطيئة فإن الارتياب في زمن حدوث المقاطعة وإعادة الشحن بعدة دورات آلة لا يؤثر بشكل كبير على سرعة التراسل وبالتالي على النقل الصحيح للمعطيات.

مثال: احسب قيمة الشحن التلقائي للمؤقت Timer1 من أجل الحصول على سرعة تراسل 4800 بت بالثانية من أجل تردد ساعة قدره Fosc=12 MHz.

نعلم أن علاقة سرعة التراسل من أجل نمط إعادة الشحن التلقائي للمؤقت Timer1 تعطى بالعلاقة:

$$\text{Baud Rate} = \frac{2^{\text{SMOD1}}}{32} \times \frac{\text{Fosc}/12}{256 - [\text{TH1}]}$$

ومنه نستنتج أن قيمة الشحن التلقائي بدلالة سرعة التراسل تعطى بالعلاقة:

$$\text{TH1} = 265 - \frac{2^{\text{SMOD1}}}{32}\frac{\text{Fosc}/12}{\text{Baud Rate}} = 256 - \frac{2^{\text{SMOD1}}}{32}\frac{10^6}{4800}$$

فمن أجل SMOD1=0 نجد أن TH1=249.49. نأخذ القيمة الصحيحة الأقرب، أي TH1=249. وبسبب التقريب فإن سرعة التراسل الفعلية لن تكون مساوية 4800 بت بالثانية. نحسب سرعة التراسل الفعلية بتعويض قيمة TH1 في العلاقة الأولى، فنجد Baud Rate=4464 bit/s. أي بخطأ نسبي قدره 336/4800=7%.

ومن أجل SMOD1=1 نجد أن TH1=242.98. نأخذ القيمة الصحيحة الأقرب أي TH1=243. فتكون السرعة الفعلية بتعويض هذه القيمة Baud Rate=4808 bit/s. أي بخطأ نسبي قدره 8/4800=0.2%.

نأخذ القيمة ذات الخطأ النسبي الأقل، أي TH1=243 مع SMOD1=1.

ملاحظة هامة: ومن الجدير بالذكر أن الخطأ النسبي يجب أن لا يتجاوز 3% لكي نتمكن من التراسل بشكل صحيح. بالتالي. أما عندما يتجاوز الخطأ النسبي هذه القيمة فإنه لا يمكن ضمان التراسل الصحيح للمعطيات التسلسلية وهذا هو سبب وجود خانات فارغة في جدول قيم إعادة الشحن التلقائي السابق بسبب تجاوز نسبة الخطأ النسبي الحد المسموح به.

النمط 2: تراسل على 9 بت بسرعات محددة

في النمط 2 للبوابة التسلسلية يتم التراسل على 11 بت: بت بداية و9 بتات معطيات و بت توقف. تتألف البتات التسعة من 8 بتات في السجل SBUF أما البت التاسع فيتم وضعة في البت TB8 من السجل SCON من أجل الإرسال، وفي الاستقبال يتم تخزينه في البت RB8 من نفس السجل. في هذا النمط، يمكن اختيار سرعة التراسل بين قيمتين Fosc/32 أو Fosc/64 عن طريق البت SMOD1 في سجل التحكم PCON.

النمط 3: تراسل على 9 بت بسرعة متغيرة

في النمط 3 للبوابة التسلسلية يتم التراسل بشكل مماثل للنمط 2 ولكن تكون سرعة التراسل متغيرة ويتم تحديدها باستخدام نفس العلاقة التي تم عرضها في النمط 1.

2.3. مقاطعة البوابة التسلسلية:

للبوابة التسلسلية مقاطعة واحدة، ولكن يمكن أن تحدث هذه المقاطعة عن طريق أحد الرايتين:

- راية الاستقبال RI: ويتم رفعها عندما يتم استقبال كلمة على البوابة التسلسلية
- راية الإرسال TI: ويتم رفعها عندما ينتهي المرسل إرسال كلمة

بالتالي فعلى المبرمج أن يحدد سبب حدوث المقاطعة التسلسلية ضمن إجرائية المقاطعة من خلال فحص الرايتين RI و TI. مع التذكير بأن المتحكم لا يقوم بتصفير هاتين الرايتين بشكل تلقائي عند الاستجابة للمقاطعة كما هي الحال في مقاطعات المؤقتات وذلك ليسمح للمبرمج بتحديد مصدر حدوث هذه المقاطعة.

4. مثال عن التراسل التسلسلي:

نريد كتابة برنامج لمتحكم يقوم بقراءة قيمة مقاسه ومرمزة على 8 بت على مدخل البوابة P1 والتي تمثل خرج مقياس رقمي لقياس الجهد وإرسال النتيجة إلى الحاسب الموصول مع المتحكم عن طريق البوابة التسلسلية. تتم هذه العملية فقط عند تلقي المتحكم أمر من الحاسب عبر البوابة التسلسلية وذلك بإرسال الحاسب إلى المتحكم القيمة 5Ah. نفرض أن سرعة التراسل بين الحاسب والمتحكم هي 4800 بت بالثانية. وأن المتحكم يعمل على تردد ساعة قدره Fosc=12 MHz.

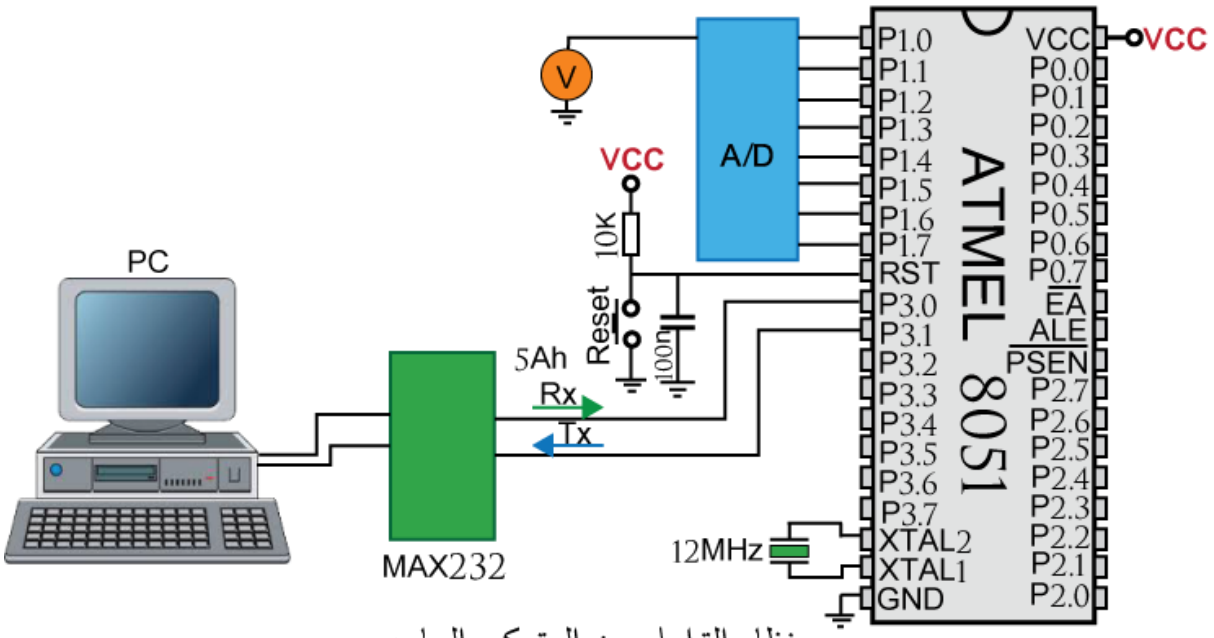
```
RxChar      EQU  30h
NewChar     EQU  20h.0

ORG  0000h
     AJMP Start

ORG  0023h
```

```
        AJMP ISR_Serial
;=========================
ISR_Serial:
IsTI:   JNB   TI, IsRI
        CLR   TI
IsRI:   JNB   RI,LB_END
        CLR   RI
        MOV   RxChar, SBUF
        SETB  NewChar
LB_END:
        RETI
;=========================

Start:
        MOV   SP,#60h
        MOV   IE,#90h
        MOV   IP,#10h
        MOV   SCON,#50h
        MOV   PCON,#80h
        MOV   TH1,#243
        MOV   TL1,#243
        MOV   TMOD,#20h
        MOV   TCON,#40h
Loop:
        JNB   NewChar,$
        CLR   NewChar
        MOV   A, RxChar
        CJNE  A,#5Ah, Loop
        MOV   A,P1
        MOV   SBUF,A
        AJMP Loop
END.
```



نظام التراسل بين المتحكم والحاسب

تعليمات الاستهلال:

في البداية نقوم بحساب قيم سجلات الوظائف الخاصة في المتحكم.

نحدد عنوان بداية ذاكرة المكدس من خلال وضع قيمة مؤشر المكدس SP=60h. نفعل المقاطعة التسلسلية من خلال وضع بت تأهيل المقاطعة التسلسلية في السجل IE بالإضافة إلى بت التفعيل العام للمقاطعات EA في نفس السجل مما يعطي القيمة IE=90h. وبما أننا لم نفعل سوى مقاطعة واحدة فلا داعي لتحديد الأولويات فنضع IP=00H. نريد أن تعمل البوابة التسلسلية على 8 بت بسرعة 4800 بت بالثانية، فيكون النمط 1 هو النمط المناسب لذلك نضع قيم البتات M0, M1 في السجل SCON لاختيار هذا النمط بالإضافة لوضع البت REN=1 لتفعيل الاستقبال في البوابة التسلسلية. إذا تم وضع REN=0 فإن البوابة التسلسلية سوف تعمل في الإرسال فقط ولن تقوم باستقبال أي كلمة. سبق وقمنا بحساب قيمة الشحن التلقائي للمؤقت Timer1 في النمط 2 لكي نحصل على سرعة تراسل 4800 بت بالثانية من أجل تردد ساعة عمل Fosc=12 MHz ووجدنا TH1=243 مع SMOD1=1. لذلك نضع قيمة PCON=80h. ثم نشحن سجلات المؤقت بالقيمة الابتدائية التي قمنا بحسابها. وكذلك نضع قيم السجلين TMOD وTCON لكي يعمل المؤقت Timer1 في نمط إعادة الشحن التلقائي.

مقاطعة البوابة التسلسلية:

نعرف متحول عام اسمه RxChar في الموقع 30h من الذاكرة، سنستخدمه لتخزين الكلمة المستقبلة من الحاسب، وكذلك سوف نعرف البت NewChar في الموقع 20h.0 سنستخدمه كراية لإعلام البرنامج باستقبال كلمة جديدة عبر البوابة التسلسلية. في إجرائية مقاطعة البوابة التسلسلية، نفحص مصدر حدوث المقاطعة، فإذا كانت بسبب انتهاء إرسال كلمة (TI=1) فإننا نقوم بتصفير الراية TI فقط. أما إذا كان مصدر المقاطعة بسبب استقبال كلمة (RI=1) فنقوم بتخزين الكلمة المستقبلة في المتحول RxChar=SBUF ورفع الراية NewChar=1. ولا ننسى أن نقوم بتصفير الراية RI من أجل إعلام المتحكم بأننا قمنا بمعالجة المقاطعة التسلسلية.

البرنامج الأساسي:

ننتظر في البرنامج الأساسي استقبال كلمة عبر البوابة التسلسلية وذلك بالفحص المستمر للراية NewChar. عند استقبال كلمة جديدة يذهب المتحكم إلى المقاطعة ويخزن الكلمة المستقبلة في المتحول RxChar ويضع الراية NewChar=1. فيترك المتحكم حلقة الانتظار(JNB) وينفذ التعليمات التالية التي نقوم فيها بفحص الكلمة المستقبلة إذا كانت مساوية لأمر القياس أو لا. عند المساواة نقوم بقراءة البوابة P1 وإرسال محتواها عبر البوابة التسلسلية والعودة إلى بداية البرنامج لانتظار أمر قياس جديد. أما عند عدم المساواة، سنعود لعملية انتظار كلمة جديدة من البوابة التسلسلية.

أسئلة الفصل الحادي عشر

أسئلة عامة

1. ما الفرق بين التراسل التسلسلي المتزامن وبين التراسل التسلسلي غير المتزامن؟

التراسل التسلسلي المتزامن يحتاج إلى إشارات تزامن بالإضافة إلى إشارة المعطيات مثل إشارة الساعة لتحديد لحظة إرسال كل بت وإشارات تزامن أخرى لتحديد بداية كل كلمة.

التراسل التسلسلي غير المتزامن لا يحتاج إلى إشارات تزامن إضافية ولكن تكون سرعة التراسل متفق عليها بين المرسل والمستقبل ويتم تحديد بداية ونهاية كلمة التراسل من إشارة المعطيات عن طريق بت البداية وبت التوقف.

3. ما هي الجهود المستخدمة في المعيار RS232 لتمثيل إشارات التراسل التسلسلية؟

الجهد 12V+ من أجل القيمة المنطقية. والجهد 12V− من أجل القيمة المنطقية 1.

4. ماهي مرابط المتحكم المستخدمة من قبل البوابة التسلسلية في المتحكم 8051؟

المربط RX والمربط TX من البوابة P3 للمتحكم.

5. ما هي أنماط عمل البوابة التسلسلية، وكيف يتم انتحاب نمط معين في المتحكم 8051؟

يوجد للبوابة التسلسلية 4 أنماط وهي:

النمط 0: نمط سجل إزاحة على 8 بت بسرعة وحيدة FOSC/12 b/s من دون بت بداية وبت توقف.

النمط 1: نمط تراسل غير متزامن على 8 بت سرعة قابلة للضبط عن طريق المؤقت Timer1.

النمط 2: نمط تراسل غير متزامن على 9 بت بسرعتين FOSC/32 و Fosc/64 بت بالثانية.

النمط 3: نمط تراسل غير متزامن على 9 بت سرعة قابلة للضبط عن طريق المؤقت Timer1.

يتم انتخاب النمط عن طريق البتين SM1, SM0 من السجل SCON.

أسئلة خيارات متعددة

1. كم عدد البوابات التسلسلية الغير متزامنة UART في المتحكم 8051؟
   A. 1
   B. 2
   C. 3
   D. 4

2. ما هي أنماط عمل البوابة التي تدعم التحكم بسرعة التراسل في المتحكم 8051؟
   A. النمط 0 والنمط 2
   B. النمط 1
   C. النمط 3
   D. النمط 1 والنمط 3

3. ما هي أنماط عمل البوابة التي تدعم التراسل على 9 بت في المتحكم 8051؟
   A. النمط والنمط 1
   B. النمط 2 والنمط 3
   C. النمط 3.
   D. النمط 1 والنمط 3

4. أين يتم وضع البت التاسع في نمط التراسل على 9 بت في الإرسال والاستقبال؟
   A. في TI و RI
   B. في SM0 و SM1
   C. في TB8 و RB8
   D. في P من PSW

5. ماهو السجل المستخدم لاختيار نمط عمل البوابة التسلسلية ؟
   A. TH1
   B. SBUF
   C. SCON
   D. PCON

مسائل

1. احسب قيمة الشحن التلقائي TH1 للمؤقت Timer1 من أجل ضبط سرعة التراسل بسرعة 1200 b/s. ثم احسب سرعة التراسل الفعلية والخطأ النسبي.

نطبق العلاقة

$$Baud\ Rate = \frac{2^{SMOD1}}{32} \times \frac{Fosc/12}{256 - [TH1]}$$

ومنه

$$[TH1] = 256 - \frac{2^{SMOD1}}{32} \times \frac{Fosc/12}{Baud\ Rate}$$

من أجل SMOD1=0:

$$[TH1] = 256 - \frac{1}{32} \times \frac{10^6}{1200} = 229.96 \cong 230$$

$$Baud\ Rate = \frac{1}{32} \times \frac{10^6}{256 - 230} = 1202$$

من أجل SMOD1=1:

$$[TH1] = 256 - \frac{2}{32} \times \frac{10^6}{1200} = 203.92 \cong 204$$

$$Baud\ Rate = \frac{2}{32} \times \frac{10^6}{256 - 204} = 1202$$

الخطأ النسبي في كلا الحالتين

$$error = \frac{1202 - 1200}{1200} = \frac{2}{1200} = 0.17\% < 3\%$$

وهو خطأ مقبول من أجل صحة التراسل.

2. في تطبيق ما، نريد التراسل بسرعة 110 b/s على 9 بت. ما هو نمط عمل المؤقت المناسب؟ ثم قم ببرمجة السجلات المناسبة لكي يعمل المتحكم بهذه السرعة. كما يتطلب هذا التطبيق تفعيل مقاطعة البوابة التسلسلية ومقاطعة المؤقت Timer0 الذي يعمل في النمط 2 والمستخدم لتوليد قاعدة زمنية ما.

أن نمط العمل المناسب للبوابة التسلسلية هو النمط 3 لأنه النمط الغير متزامن على 9 بت والذي يمكن برمجة سرعة التراسل فيه.

من أجل معرفة نمط العمل المناسب للمؤقت، نحسب الحد التالي في علاقة سرعة التراسل :

$$\frac{1}{32} \times \frac{Fosc/12}{Baud\ Rate} = 284.1$$

وهي قيمة أكبر من 255 لذلك لا يمكن استخدام نمط إعادة الشحن التلقائي للمؤقت. بالتالي فإننا نحتاج للنمط 1 على 16 بت ويجب تفعيل مقاطعة المؤقت لكي نقوم بإعادة شحن المؤقت بالقيمة البدائية.

في هذا النمط نحسب قيمة إعادة شحن المؤقت من العلاقة:

$$[\mathrm{TH1:TL1}] = 65536 - \frac{2^{SMOD1}}{32} \times \frac{Fosc/12}{Baud\ Rate}$$

من أجل SMOD1=0 نجد

$$[\mathrm{TH1:TL1}] = 65536 - 284.1 = 65251.9 \cong 65252 = FEE4h$$

$$Baud\ Rate = \frac{1}{32} \times \frac{10^6}{65536 - 65252} = 110.04$$

من أجل SMOD1=1 نجد

$$[\mathrm{TH1:TL1}] = 65536 - 2 \times 284.1 = 64967.82 \cong 64968 = FDC8h$$

$$Baud\ Rate = \frac{2}{32} \times \frac{10^6}{65536 - 64968} = 110.04$$

وبما أن الحالتين متماثلتين، نختار الحالة الأولى SMOD1=0 و $[\mathrm{TH1:TL1}] = FEE4h$. كما يجب تفعيل مقاطعة المؤقت Timer0 والمقاطعة التسلسلية وكذلك مقاطعة المؤقت Timer1 الذي يقوم بضبط سرعة التراسل مع إعطاء الأولوية لمقاطعة المؤقت Timer1 لكي تتم إعادة شحن المؤقت فوراً.

SCON=C0h

PCON=00H

TL1=E4h

TH1=FEh

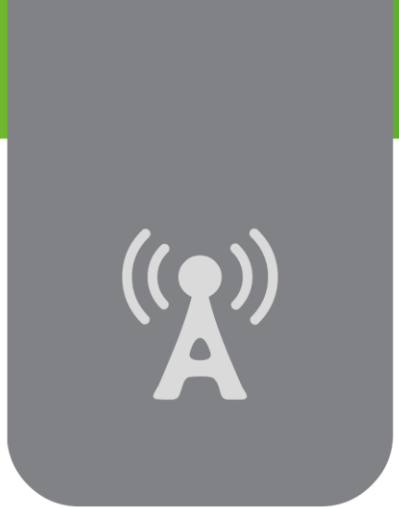
TMOD=12h

TCON=50h

IE=9Ah

IP=08h

رقم التمرين	الإجابة الصحيحة
1	A
2	D
3	B
4	C
5	C



الفصل الثاني عشر:
المعالجات والمتحكمات الحديثة

الكلمات المفتاحية:

مقارنة سرعة المعالجات، معالجات بنتيوم، معالجات PowerPC، متحكمات AVR، متحكمات PIC. معالجات الإشارة الرقمية.

ملخص:

يهدف هذا الفصل إلى التعرف على مجموعة من المعالجات والمتحكمات الحديثة وإلقاء الضوء على التطويرات التي تم إدخالها في بنية هذه المعالجات بهدف تحسين الأداء من حيث السرعة والاستجابة لمتطلبات التطبيقات المختلفة. سوف نستعرض عائلة معالجات بنتيوم الشائعة الاستخدام في الحواسيب الشخصية. كما سوف نبين أهم أنواع المتحكمات الحديثة وأبرز مواصفاتها. وأخيراً سوف نذكر بشكل سريع معالجات الإشارة الرقمية وأهم النقاط التي تميزها عن المعالجات ذات الأهداف العامة.

الأهداف التعليمية:

يتعرف الطالب في هذا الفصل على:

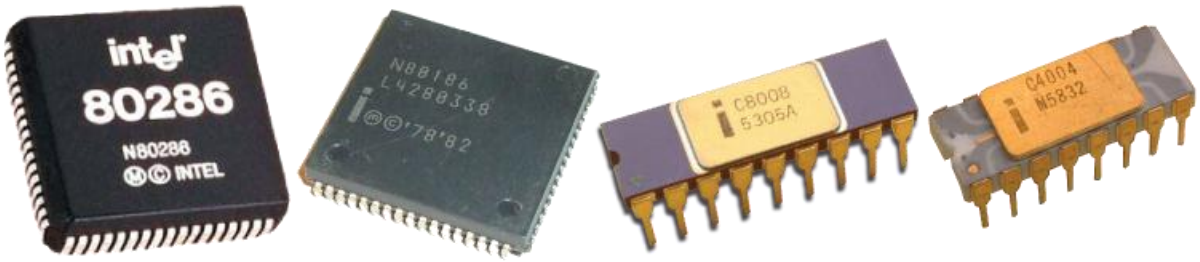
- معايير مقارنة سرعة المعالجات
- عائلة معالجات بنتيوم والتطويرات المدخلة فيها
- عائلتي المتحكمات الحديثة AVR و PIC ومزاياها العامة
- الفرق بين المعالجات ذات الأهداف العامة ومعالجات الإشارة الرقمية

الفصل الثاني عشر: المعالجات والمتحكمات الحديثة

يهدف هذا الفصل إلى التعرف على مجموعة من المعالجات والمتحكمات الحديثة وإلقاء الضوء على التطويرات التي تم إدخالها في بنية هذه المعالجات بهدف تحسين الأداء من حيث السرعة والاستجابة لمتطلبات التطبيقات المختلفة. سوف نستعرض عائلة معالجات بنتيوم الشائعة الاستخدام في الحواسيب الشخصية. كما سوف نبين أهم أنواع المتحكمات الحديثة وأبرز مواصفاتها. وأخيراً سوف نذكر بشكل سريع معالجات الإشارة الرقمية وأهم النقاط التي تميزها عن المعالجات ذات الأهداف العامة.

1. مقدمه:

كما نعلم، كانت البداية الفعلية لظهور المعالجات هي مع ظهور المعالج INTEL 4004 في عام 1971. وبعد ذلك سريعاً ظهر المعالج INTEL 8008 على 8 بتات. ومن ثم توالى ظهور المعالجات مثل INTEL 8086، INTEL 80186، INTEL 80286، INTEL 80386، INTEL 80486. وكان التوجه العام باستخدام رقم مختلف لكل معالج جديد، حيث نستخدم التسمية 80x86 للإشارة إلى هذه العائلة من المعالجات.



بعض المعالجات الأولى من شركة إنتل

بعد ذلك، تم التوجه بإطلاق تسميات مختلفة مع أرقام على المعالجات، وكان سبب ذلك أمور قانونية تمنع امتلاك الأرقام كعلامة تجارية فظهر المعالج Pentium والذي كان يراد تسميته INTEL 80586.

المعالج بنتيوم

في أواخر عام 1973 أنتجت شركة إنتل المعالج INTEL 8080A وهو معالج على 8 بت مع مسرى عناوين بعرض 16 بت. وكان مصنوعاً في قالب يحتوي على 40 مربط، الأمر الذي أصبح مرجعاً بالنسبة لكل المعالجات على 8 بت.; وفي نفس الوقت تقريباً أنتجت شركة موتورولا المعالج MC6800 بمواصفات مماثلة لمعالج إنتل إلا أنه لم يكن يحتوي على سجلات عامة داخلية بل كان يحتوي على مراكمين فقط. وكان يعتمد على تخزين المعاملات في الذاكرة الخارجية. بالرغم من أن أداء هذا المعالج كان أسرع قليلاً من معالج إنتل، إلا أنه لم يحقق القفزة النوعية ليتغلب على معالج إنتل وحل في المرتبة الثانية في الأسواق العالمية.

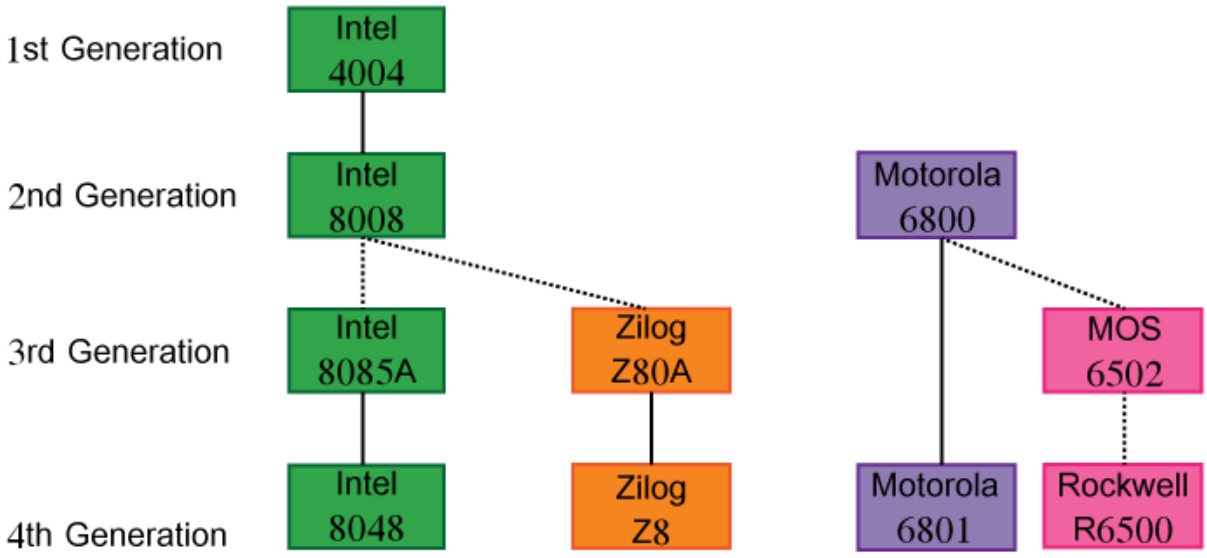
في هذه الأثناء ظهر منافس وهو شركة Zilog والتي عملت على تطوير المعالج INTEL 8080 لإنتاج المعالج Z80 والذي يعمل على تغذية وحيدة 5V+ حيث تم فيه استخدام سجلات داخلية مع زيادة عددها لتصبح 14 سجل بدل 8. مع استعارة بعض الأفكار المفيدة من معالج موتورولا. يدعم المعالج Z80 جميع تعليمات معالج إنتل مع بعض التعليمات الإضافية.

مع تطور المعالجات، ولتلبية حاجة تطبيقات التحكم الصناعي، تم التفكير في دمج العناصر الضرورية لعمل المعالج من ذاكرة ميتة وذاكرة حية ضمن نفس الدارة التكاملية وخصوصاً أن تطبيقات التحكم لم نكن بحاجة لكثير من الذاكرة. فقامت شركة إنتل بإنتاج INTEL 8048 والذي لم يكن فقط عبارة المعالج INTEL 8085 مع ذواكر داخلية (1KB ذاكرة ميتة مع 64 بايت ذاكرة حية) ولكن تم تعديل بنيته الداخلية ووضع مجموعة جديدة من التعليمات الغير متوافقة مع أي من المعالجات السابقة. وقبل تسمية هذه لدارات بالمتحكمات (Microcontroller) كانت تسمى في البداية بالحاسوب على دارة واحدة (One-Chip Microcomputer).

وسريعاً ما تم مضاعفة حجم هذه الذاكرة في التحكم INTEL 8048 ضمن النسخة الجديدة من هذا المتحكم INTEL 8049. ولاحقاً تم إضافة عنصر هام وهو المؤقت لهذه المتحكمات.

لم تكن شركة Zilog متأخرة كثيراً فسارعت إلى إطلاق المتحكم Z8 الذي كان قريباً من المتحكم INTEL 8049 مع مؤقتين داخليين، وإمكانية إضافة ذواكر خارجية تصل حتى 64KB في السعة سواء من الذاكرة الميتة أو الحية.

قامت شركة موتورولا بالرد على ذلك بإطلاق المتحكم MC6801. الذي كان متوافقاً مع المعالج MC6800 بمزايا مشابهة للمتحكمات المنافسة مع إضافة مؤقت ووحدة تراسل تسلسلية UART. هذا وكان هناك شركات منافسة أخرى مثل AMD وشركات أخرى أقل شهرة من الشركات السابقة مثل شركة MOS و RockWell.



ظهور المتحكمات نتيجة لتطوير المعالجات

2. معايير المقارنة بين سرعة المعالجات:

مع ظهور العديد من المعالجات والمتحكمات الحديثة، فإن عملية انتقاء المعالج الأسرع هو ليست عملية بسيطة كما نعتقد. فمعالج ما قد يكون أسرع في تنفيذ برنامجاً اختبارياً ما من معالج آخر، ولكن قد يكون أبطأ من أجل برنامجاً اختبارياً آخر. ولكن بشكل عام هناك بعض المقاييس التي تستخدم لقياس سرعة المعالج أو المتحكم.

1.1.2. معيار عدد التعليمات بالثانية:

يبدو مقياس عدد ملايين التعليمات التي يستطيع المعالج تنفيذها بالثانية MIPS (Millions of Instructions Per Second) هو مقياس سهل الحساب لتقييم سرعة معالج ما.

فعلى سبيل المثال، إذا أخذنا المعالج Athlon من شركة AMD الذي يعمل على ساعة بتردد 2 GHz. وهو قادر على تنفيذ حتى 9 تعليمات بكل دور ساعة، بالتالي تكون سرعته 18 GMIPS. ولكن الأمر ليس بهذه البساطة. بعض التعليمات تحتاج إلى عدة دورات ساعة لكي يتم تنفيذها. ولذلك فإن الشركات المتنافسة تعلن هذا القياس بالاعتماد على أقصر التعليمات الموجودة لإعطاء انطباع مبالغ فيه في تقدير سرعة المعالج. فإذا أخذنا مثلاً المعالج 80386 الذي يعمل على تردد ساعة قدره 25 MHz، ونظرنا إلى نشرته الفنية لوجدنا أن تعليمة الجمع تحتاج إلى دوري ساعة أما تعليمة القسمة فتحتاج إلى 46 دور ساعة، حتى أن هناك بعض التعليمات التي تحتاج إلى 316 دور ساعة. حتى لو أخذنا نفس مجموعة التعليمات، فلكل معالج تميز في تنفيذ بعض من هذه التعليمات دون الأخرى. لذلك فإن المقارنة بين المعالجات بالاعتماد على هذا المقياس فقط ليس ذو دلالة كبيرة في الحسم بأفضلية معالج على آخر بشكل عام.

2.1.2. معيار عدد تعليمات الفاصلة العائمة بالثانية:

من أجل التغلب على مشكلة أي التعليمات سنختار بهدف المقارنة بين المعالجات، تم اختيار عدد من عمليات الفاصلة العائمة لقياس ما يسمى FLOPS (Floating-point Operations Per Second). حتى مع هذا المقياس فهناك الكثير من الاعتراضات على اعتباره مقياساً صريحاً لقياس سرعة المعالج.
يطبق هذين القياسين MIPS و FLOPS لمقارنة المعالجات وليس الأنظمة. أما من أجل النظم فهناك مقاييس أخرى يتم استخدامها مثل: علامة الكفاءة (Benchmarks) حيث يتم استخدام مجموعة من البرامج وتشغيلها على مختلف النظم المراد قياس مدى سرعتها. وعدد عمليات الدخل والخرج بالثانية وعدد عمليات النقل بين الذواكر وغيره.

3. المعالجات الحديثة:

سوف نبين في هذه الفقرة مجموعة محدودة من المعالجات الحديثة نسبياً لتبيان التطويرات اللاحقة التي إدخالها في بنية المعالجات مقارنة مع المعالج 8086 الذي سبق لنا دراسته.

1.3. عائلة معالجات بنتيوم:

1.1.3. معالج بنتيوم:

إن معالج بنتيوم (Pentium) هو معالج 32 بت كسابقه 80486 ولكن تم تحسينه بشكل واضح لزيادة سرعته ويحتوي على 296 مربط. وهو معالج من نوع التعليمات المعقدة CISC حيث يدعم أكثر من 400 تعليمة. وكانت النسخة الأولى منه في عام 1992 بسرعة 66 MHz. ومن المواصفات الأساسية لهذا المعالج نذكر:

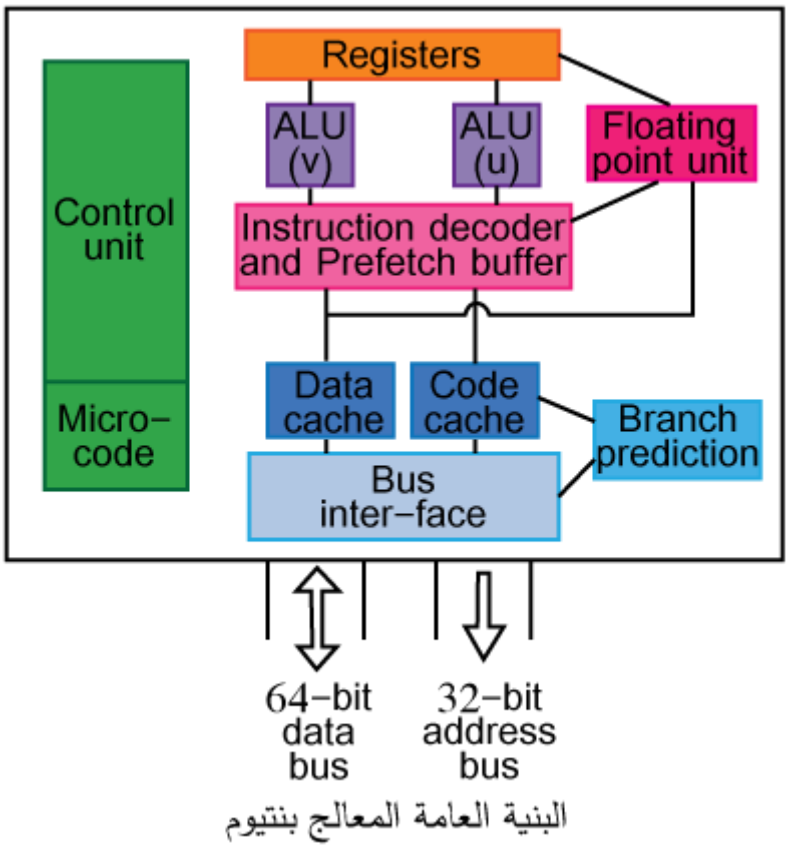
ذاكرة مخبأة للمعطيات والبرنامج:

يتم في هذا المعالج الوصل مع العالم الخارجي باستخدام مسرى معطيات بعرض 64 بت ومسرى عناوين بعرض 32 بت. حيث يتم شحن تعليمات البرنامج والمعطيات من الذاكرة الخارجية بشكل سريع إلى الذاكرة الداخلية المخبأة (Cashe) بسعة 8 KB لذاكرة المعطيات و8KB لذاكرة البرنامج. وتتم عملية النقل هذه بشكل رشقات سريعة عندما يكون ذلك ممكناً وذلك أجل زيادة سرعة العمل وتجنب انتظار المعالج حتى قراءة التعليمات والمعطيات من الذاكرة الخارجية المكلف زمنياً.

ذاكرة البحث المسبق عن التعليمة:

إن ذاكرة البحث المسبق عن التعليمة (prefetch buffer) هي عبارة عن ذاكرة صغيرة يتم فيها تخزين مجموعة التعليمات التي تنتظر التنفيذ. حيث يتم نقل التعليمات واحدة تلو الأخرى من الذاكرة المخبأة إلى ذاكرة البحث المسبق مم يؤمن استمرارية عمل المعالج. في الحقيقة فإن هذه الذاكرة مقسومة إلى قسمين، كل قسم يغذي وحدة

حساب ومنطق ALU، حيث يوجد في هذا المعالج وحدتي حساب ومنطق u و v تعملان على التوازي كما هو مبين على الشكل.



البنية العامة المعالج بنتيوم

وحدات الحساب والمنطق:

يمتلك المعالج وحدتي حساب ومنطق كل منهما تعمل بتواردية بعمق خمسة تعليمات بحيث يمكن تنفيذ تعليمتين على التوازي عند كل نبضة ساعة. يمكن تنفيذ كل التعليمات ما عدا تعليمات الفاصلة العائمة على الوحدة u أما الوحدة v فتقوم بتنفيذ مجموعة محدودة أكثر من التعليمات. مع هذا، فلا يمكن تشغيل الوحدتين على التوازي دائماً وذلك بسبب اعتماد التعليمات على بعضها، فعندما تحتاج تعليمة على نتيجة التعليمة التي تسبقها فلا يمكن تنفيذ التعليمتين على التوازي كل تعليمة في وحدة مستقلة. ولكن هذا يفيد من أجل التعليمات المستقلة عن بعضها.

وحدة الفاصلة العائمة:

يمتلك المعالج وحدة الفاصلة العائمة FPU (Floating Point Unit) من أجل العمليات الحسابية بالفاصلة العائمة مع استخدام مفهوم التواردية داخل هذه الوحدة الأمر الذي يجعلها تعمل بشكل أسرع بحوالي 10 مرات من الوحدة FPU الموجودة في المعالج 80486.

التنبؤ بالقفز:

عندما يصل البرنامج إلى تعليمة قفز فإن المعالج سينتقل إلى مكان آخر في ذاكرة البرنامج. وفي معظم الأحيان تكون تعليمة القفز شرطية. فعندما يتحقق الشرط ويريد المعالج القفز فلا بد من إزالة التعليمات التالية التي دخلت في مراحل التواردية ويسمى ذلك تفريغ (Flushing) مراحل التواردية مما يسبب تأخير بقد 4 أو 5 دورات ساعة حسب عمق التواردية. لتجنب ذلك، يتم تزويد المعالج بوحدة تنبؤ فيما إذا كان المعالج سوف يقفز أم لا نتيجة تعليمة القفز الشرطية. وتتم عملية التنبؤ بتذكر ماذا فعل المعالج عند المرور بهذه التعليمة سابقاً: هل قفز أم لا. فإذا كان قد قفز في المرة السابقة فإنه على الأرجح سيقفز هذه المرة أيضاً ولذلك يقوم المعالج بشحن التعليمات الجديدة من مكان القفز وبهذا يوفر التأخير الناجم عن إفراغ محتوى مراحل التواردية. وتأتي هذه القاعدة من أن معظم تعليمات القفز الشرطي تنتج عن الحلقات المتكررة التي تقوم بالقفز دائماً ما عدا في المرة الأخيرة. أما في الذاكرة المؤقتة فيقوم المعالج بتجهيز التعليمات في كلا الحالتين، سواء قفز المعالج أم لا.

2.1.3. المعالج بنتيوم MMX:

في معالج بنتيوم MMX (MultiMedia eXtensions) تمت إضافة بعض التطويرات لزيادة سرعة تطبيقات الوسائط المتعددة والاتصالات. وتم ذلك نتيجة لتحليل الكثير من التطبيقات التي تتعامل مع الرسومات والفيديو والألعاب والتعرف على الكلام وغيرها. وتم الكشف عن العمليات المتكررة الشائعة في هذه التطبيقات ودمج كل مجموعة ضمن تعليمة واحدة من نوع جديد يدعى تعليمة واحدة متعددة المعطيات SIMD (Single Instruction Multiple Data) مثل تعليمة تغيير لون مجموعة من النقاط على الشاشة. فبدل تغيير لون كل نقطة على حدة، تمت إضافة تعليمة تقوم بتغيير لون 8 نقاط دفعة واحدة بتعليمة واحدة.

3.1.3. المعالج بنتيوم 4:

في تصنيع المعالج بنتيوم 4 (Pentium 4) تم استخدام تقنية تصنيع الترانزستورات بعرض 0.13 µm بدل من 0.18µm في المعالجات التي سبقته. الأمر الذي سمح بزيادة عدد الترانزستورات المستخدمة ليصل إلى 55 مليون ترانزستور (على مساحة 1 mm^2 فقط) بدل 42 مليون وتخفيض جهد التغذية من 1.75V إلى 1.5V الأمر الذي سمح بزيادة السرعة لتصل إلى 1.8 GHz وتخفيض كلفة الإنتاج بنحو 20%. يستهلك هذا المعالج استطاعة قدرها 55 واط وبالتالي فلا بد من تبريده بشكل جيد ليعمل بشكل صحيح. وبالتالي تمت إضافة دارة تبريد تقوم بقياس درجة حرارة المعالج وزيادة سرعة مروحة التبريد عندما ترتفع درجة الحرارة.

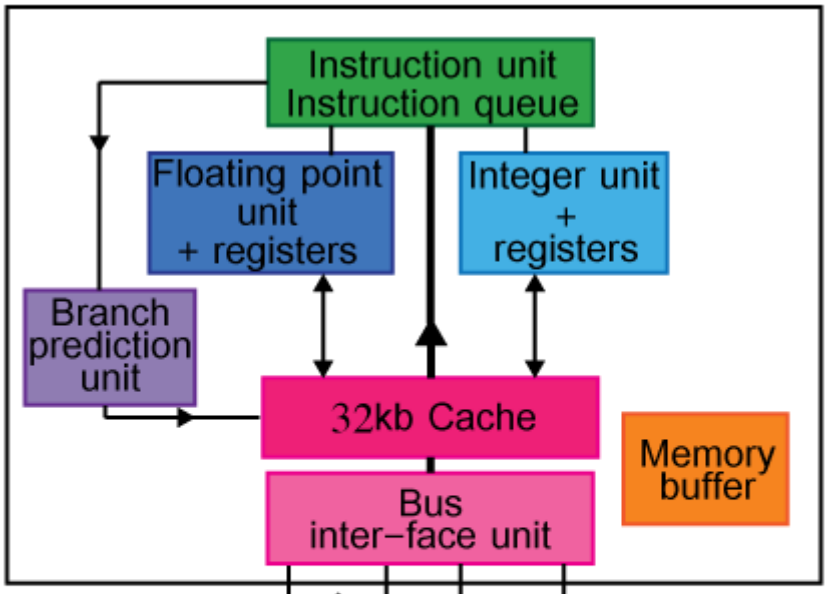
4.1.3. المعالج سيليرون:

يعتبر المعالج سيليرون Celeron النسخة الاقتصادية من المعالج بنتيوم. حيث تم تخفيض سعر هذا المعالج على حساب إزالة بعض المكونات الغير أساسية مثل حجم الذاكرة المخبأة وتخفيض سرعة المسرى الداخلي.

2.3. عائلة معالجات PowerPC:

في الوقت الذي كانت فيه شركة إنتل تنتج معالجات من نوع CISC بالتحاف مع شركة Microsoft، تولد تحالف من شركات IBM و Motorola و Apple لإنتاج معالجات من نوع RISC والذي تمخض عن إنتاج عائلة معالجات PowerPC. تم الاهتمام في تصميم هذه المعالجات بتغطية التطويرات المستقبلية لتسهيل عملية الانتقال إلى الأجيال القادمة فعلى سبيل المثال تم التصميم الأولي للمعالج على 64 بت بالرغم من استخدام 32 بت فقط في النسخ الأولية.

تم إنتاج المعالج MPC601 في عام 1994 باستخدام 2.8 مليون ترانزستور، وذلك أقل من عدد الترانزستورات المستخدمة في المعالج بنتيوم، ولكن الزيادة الموجودة في معالج بنتيوم كان من أجل الحفاظ على التوافق مع المعالجات السابقة.



البنية العامة المعالج MPC601

تم استخدام هذه المعالجات في إنتاج حواسيب Apple-Mac باستخدام أنظمة التشغيل UNIX و LINUX. وفي نسخة حديثة لهذه العائلة تم إنتاج المعالج PowerPC 970 بمواصفات قريبة من مواصفات المعالج بنتيوم 4 ولكن باستخدام مسرى داخلي بسرعة 900MHz مقابل 530MHz في المعالج بنتيوم 4. لقد لاقت هذه المعالجات مكاناً في التطبيقات الصناعية، حيث تم اعتمادها للاستخدام في سيارات فورد الحديثة.

وأخيراً نذكر المعالج Athlon 64 من شركة AMD المنافس الأكبر لشركة إنتل في هذا المجال.

جدول 1: مقارنة بين ثلاث معالجات من ثلاث شركات.

	Power PC 970	Pentium 4	Athlon 64
Clock speed	2GHz	2.8GHz	2GHz
Bus speed	900MHz	533MHz	533MHz
Bits	64	32	64
Process size	0.09/0.13 microns	0.13 microns	0.13 microns
Op systems	OSX IBM Linux	Windows	Windows
Comparative speed	1988	1984	2372
Max memory	Terabytes	40GB	Terabytes

مقارنة بين ثلاث معالجات من ثلاث شركات

4. المتحكمات الحديثة:

1.4. عائلة متحكمات AVR:

تعتبر متحكمات AVR من شركة ATMEL متحكمات ذات بنية محدودة التعليمات RISC. حيث أن معظم التعليمات تتطلب دور ساعة واحدة أو دورين للتنفيذ. مع العلم أن هذه المتحكمات تعمل على ساعة بتردد من 8MHz إلى أكثر من 20MHz. مما يمثل زيادة في سرعة المتحكم مقارنة بالمتحكم 8051 بمعدل من 4 إلى 10 مرات تقريباً. وهي من أولى المتحكمات التي تستخدم ذاكرة من نوع Flash كذاكرة برنامج والتي هي نوع خاص من الذواكر EEPROM القابلة للبرمجة بالكتل بدل من برمجة بايت بعد آخر وذلك باستخدام جهد برمجة +5V فقط.

جدول2 : جدول يبين بعض من متحكمات عائلة AVR وأهم مواصفاتها.

MC	Flash (KB)	SRAM (Bytes)	EEPROM (Byte)	SPI	I2C	UART	ADC Chnnl (10bit)	PWM Chnnl	Timer	RTC
ATtiny2312	2	128	128	2	1	1		4	2	No
ATtiny84	8	512	512	1	1	0	8	4	2	No
ATtiny85	8	512	512	1	1	0	4	6	2	No
ATmega8	8	1024	512	1	1	1	8	3	3	Yes
ATmega168	16	1024	512	2	1	1	8	6	3	Yes
ATmega328	32	2048	1024	2	1	1	8	6	3	Yes

وهناك عدد كبير من هذه المتحكمات على 8 بت وعلى 32 بت. وتختلف في عدد المرابط وسعة الذواكر الداخلية ونوعية الطرفيات المدمجة في داخله. حيث تم تزويد هذه المتحكمات بالكثير من الإضافات والطرفيات المفيدة مثل ذواكر من نوع EEPROM لحفظ المعطيات ومؤقتات تدعم توليد إشارات تعديل عرض النبضة PWM (Pulse Width Modulation) الشائع الاستخدام في مجال التحكم الصناعي. بالإضافة إلى وجود مبدلات تمثيلية رقمية لقراءة الجهود المستمرة وتحويلها إلى قيم رقمية وكذلك دعم بروتوكولات التراسل التسلسلي المتزامن من نوع SPI (Serial Peripheral Interface) أو I^2C. وحتى أن هناك بعض المتحكمات من هذه العائلة التي تدعم الربط التسلسلي عن طريقة البوابة التسلسلية USB أو بوابة ايثرنت (Ethernet).

يمكن برمجة هذه المتحكمات بشكل تسلسلي ضمن الدارة (In Circuit Programming) الأمر الذي يمكن من تعديل برنامج المتحكم عن بعد دون الحاجة إلى إزالة المتحكم من الدارة وهذا الأمر مفيد جداً في عمليات التطوير والصيانة عن بعد للأنظمة التي تستخدم هذه المتحكمات.

2.4. عائلة متحكمات PIC:

وهي متحكمات من إنتاج شركة Microchip ذات بنية RISC أيضاً. ويكون عدد التعليمات المستخدمة في هذه العائلة أقل بشكل عام من عدد التعليمات في معالجات AVR. ويكون في هذه المعالجات طول التعليمة ثابتاً على عدد معين من البتات (14 بت مثلاً)، ولذلك فإن تنظيم ذاكرة البرنامج ليس بالبايتات ولكن بالكلمات بحيث تعبر كل كلمة عن تعليمة.

في هذه العائلة، يتم تقسيم تردد الساعة على 4 للحصول على دور الآلة، ولهذا السبب نلاحظ أن ترددات ساعة العمل في معالجات PIC هي أعلى من ترددات ساعة العمل في معالجات AVR. ولكن هناك تقارب في عدد التعليمات بالثانية MIPS بين العائلتين.

بشكل عام هناك تقارب كبير في الميزات الموجودة كل من معالجات PIC و AVR ولذلك هناك تنافساً قوياً بين هاتين العائلتين في السوق العالمية.

5. معالجات معالجة الإشارة:

تعتبر معالجات معالجة الإشارة (Digital Signal Processors) عبارة عن معالجات مختصة في معالجة الإشارة الرقمية، وتستخدم في التطبيقات التي تتطلب تطبيق خوارزميات رياضية ضمن الزمن الحقيقي مثل الترشيح الرقمي وحساب تحويلات فورييه والترابط بين الإشارات وغيرها من الخوارزميات المستخدمة في مجال معالجة الإشارة. في معظم هذه الخوارزميات نحتاج لحساب قيمة جداء داخلي بين شعاعين كما في العبارة التالية:

$$y = \sum_{N-0}^{N-1} a_k b_k$$

وهي عبارة عن مجموع مضاريب. ولذلك فمن الشائع قياس سرعة معالجات معالجة الإشارة بعدد عمليات الضرب المراكمة MAC (Multiply and ACcumulate) في الثانية.

تم تصميم هذه المعالجات بحيث تقدم أداءً أفضل بالمقارنة مع المعالجات ذات الأهداف العامة مثل بنتيوم عند التعامل مع خوارزميات معالجة الإشارة. ونذكر مثالاً على ذلك المعالج TMS320C6455 والمعالج TMS320C6713 من شركة Texas Instrument. فالأول يتعامل مع الأرقام الصحيحة فقط بعرض يصل 64 بت، والثاني قادر على إجراء العمليات بالفاصلة العائمة ولكن بسرعة أقل.

ومن الأمور الهامة في بنية معالجات الإشارة استخدامه العديد من الوحدات الحسابية التي تعمل على التوازي في بنيته الداخلية بهدف تسريع تنفيذ البرامج.

- تم أمثلة بنية معالجات الإشارة لتنفيذ الحلقات التكرارية بشكل أفضل
- دعم أنماط عنونة جديد مثل العنونة الدوارة (circular addressing) والعنونة بعكس البتات (bit reverse addressing) الضرورية في بعض عمليات معالجة الإشارة
- دمج وحدات إدخال وإخراج سريعة للتعامل مع تدفق عينات الإشارة الرقمية بشكل مستمر
- وجود مسرى معطيات عريض يصل إلى 128 بت لتسريع النفاذ للمعطيات
- وجود تعليمات متخصصة غير موجودة في المعالجات العامة مثل تعليمة الجداء الداخلي DOTP2

في الحقيقة، تعد المعالجات الصغرية أقرب في بنيتها الداخلية للمتحكمات من المعالجات وذلك بسبب دمج الذواكر والكثير من الطرفيات المفيدة ضمن بنيتها الداخلية. ولكن تم الحفاظ على استخدام مصطلح معالج بدل من متحكم بسبب تخصصها في تطبيقات معالجة الإشارة وليس التحكم فقط.

أسئلة الفصل الثاني عشر

1. اذكر معيارين لمقارنة سرعة المعالجات مع بعضها؟

معيار عدد التعليمات في الثانية MIPS ومعيار عدد تعليمات الفاصلة العائمة بالثانية FLOPS.

2. اذكر ثلاث مزايا لمعالج بنتيوم تمت إضافتهما لتسريع عمل المعالج.

1. ذاكرة مخبأة للمعطيات والبرنامج
2. ذاكرة البحث المسبق عن التعليمة
3. وجود وحدتي حساب ومنطق ALU.

3. اشرح عملية التنبؤ بالقفز في المعالج بنتيوم وما هي فائدته؟

عند وجود تعليمة قفز شرطي في البرنامج وإذا كان المعالج قد مر عليها سابقاً فإن المعالج سوف يتذكر هل قام بالقفز أم لا عندما في المرة السابقة، وسيفترض أنه سيقوم بنفس الشيء في هذه المرة ولذلك يقوم بشحن التعليمات المناسبة في صف التواردية.

تفيد هذه العملية في تجنب التأخير الناجم عن تفريغ صف التواردية في حالة القفز ويساهم ذلك في تسريع عمل البرنامج ولا سيما في تنفيذ الحلقات الشرطية.

4. ماهي أهم التطويرات المدخلة في المعالج بنتيوم MMX؟

في معالج بنتيوم MMX تمت إضافة بعض التطويرات لزيادة سرعة تطبيقات الوسائط المتعددة والاتصالات.

5. ما الذي سمح بزيادة السرعة في المعالج بنتيوم 4 مقارنة مع المعالجات السابقة من نفس العائلة؟

تخفيض جهد التغذية من 1.75V إلى 1.5V الأمر الذي سمح بزيادة السرعة لتصل إلى 1.8 GHz.

6. كيف تم تخفيض كلفة إنتاج المعالج سيليرون مقارنة مع المعالج بنتيوم؟

حيث تم تخفيض سعر هذا المعالج على حساب إزالة بعض المكونات الغير أساسية مثل حجم الذاكرة المخبأة وتخفيض سرعة المسرى الداخلي.

7. ما الفرق بين عائلة معالجات بنتيوم ومعالجات PowerPC من حيث البنية الداخلية؟

عائلة معالجات PowerPC هي معالجات ذات بنية RISC أما عائلة معالجات بنتيوم فهي ذات بنية CISC.

8. ما هي الشركات التي ساهمت في إنتاج معالجات PowerPC؟

شركات IBM و Motorola و Apple.

9. كم يبلغ دور الآلة في عائلة متحكمات AVR مقارنة بمتحكمات عائلة 8051 مقدراً بعدد نبضات ساعة العمل؟

في المتحكمات AVR يكون دور الآلة مكون من نبضة ساعة واحدة، بينما في عائلة متحكمات 8051 فإن دور الآلة مكون من 12 نبضة ساعة.

10. كم يبلغ دور الآلة في عائلة متحكمات AVR مقارنة بمتحكمات عائلة PIC مقدراً بعدد نبضات ساعة العمل؟

في المتحكمات AVR يكون دور الآلة مكون من نبضة ساعة واحدة، بينما في عائلة متحكمات PIC فإن دور الآلة مكون من 4 نبضات ساعة.

11. ما الفرق بين معالجات الإشارة الرقمية والمعالجات ذات الأهداف العامة مثل معالجات بنتيوم؟

تم تصميم معالجات الإشارة الرقمية لتنفيذ خوارزميات معالجة الإشارة الرقمية مثل الترشيح والترابط وتحويل فورييه بشكل أسرع كما نحتوي على مساري معطيات أعرض وتعليمات وأنماط عنونة متخصصة مثل العنونة الدوارة.

أسئلة خيارات متعددة

1. كم عدد وحدات الحساب والمنطق ALU في المعالج بنتيوم؟
   A. 1
   B. 2
   C. 3
   D. 4

2. يوجد في المعالج بنتيوم وحدة حساب بالفاصلة العائمة أسرع بعشر مرات من المعالج 486 وذلك بفضل:
   A. استخدام تقنية التواردية داخل الوحدة.
   B. استخدام تقنية تصنيع متطورة.
   C. استخدام عدد ترانزستورات أكثر.
   D. استخدام عدة ضوارب على التوازي.

3. تساهم الذاكرة المخبأة في تسريع عمل المعالج عن طريق:
   A. تسريع عملية النفاذ إلى الذاكرة الخارجية للمعالج.
   B. توفير مساحة كافية لتخزين كامل البرنامج في الذاكرة الداخلية.
   C. توفير مساحة كافية لتخزين كامل المعطيات في الذاكرة الداخلية.
   D. تجنب انتظار المعالج حتى يتم جلب التعليمة قيد التنفيذ من الذاكرة الخارجية.

4. تستخدم ذاكرة البحث المسبق عن التعليمة من أجل:
   A. نقل التعليمات من الذاكرة الخارجية إلى وحدة الحساب المركزية.
   B. نقل التعليمات من الذاكرة المخبأة إلى وحدة الحساب المركزية.
   C. نقل التعليمات من الذاكرة الخارجية إلى وحدة الحساب المركزية.
   D. تخزين مجموعة التعليمات التي تنتظر التنفيذ.

5. من هي العائلة التي لا تنتمي للمتحكمات؟
   A. 8051
   B. VAR
   C. PowerPC
   D. PIC

رقم التمرين	الإجابة الصحيحة
1	B
2	A
3	D
4	D
5	C