جامعة دمشق
الكلية التطبيقية

قسم تقنيات الحاسوب

السنة الرابعة

د.م. رائد الشرع م. آلاء خسارة

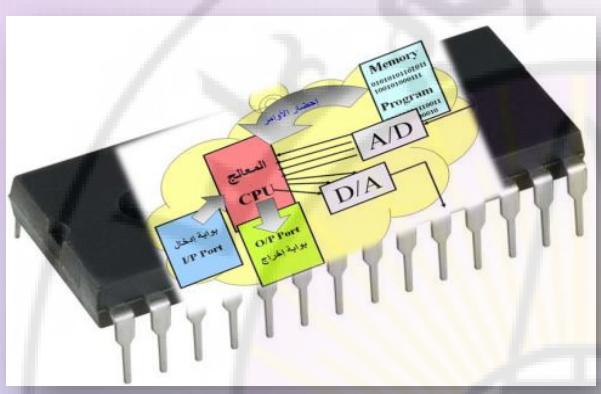
مقرر المعالجات والمتحكمات الصغرية

المعالجات والمتحكمات الصغرية
المحاضرة الأولى

مدخل إلى المعالجات الصغرية

تعريف المعالج الصغري:

المعالج الصغري هو عبارة عن شريحة او رقاقة من السيليكون مغلفة تشكل 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 . أي ان المعالج يقوم بتنفيذ البرنامج المخزن في الذاكرة ويتبادل المعطيات مع الوسط الخارجي من خلال بوابات دخل وخرج مختصة.

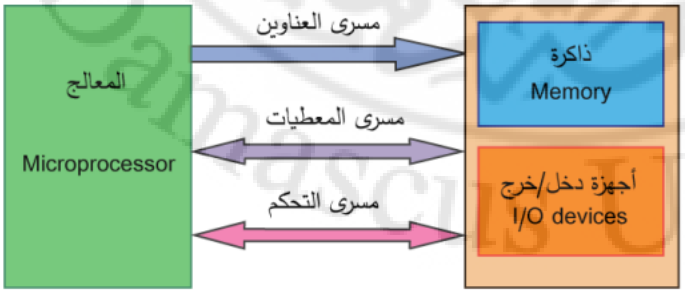


نظم المعالجة الحاسوبية:

مثل الحواسيب الشخصية واللوحية وغيرها من الأجهزة التي يشكل المعالج الجزء الأساسي في بنيتها .

يتألف أي نظام معالجة حاسوبي بشكل أساسي من ثلاثة أجزاء :

1. معالج
2. ذاكرة
3. تجهيزات دخل وخرج



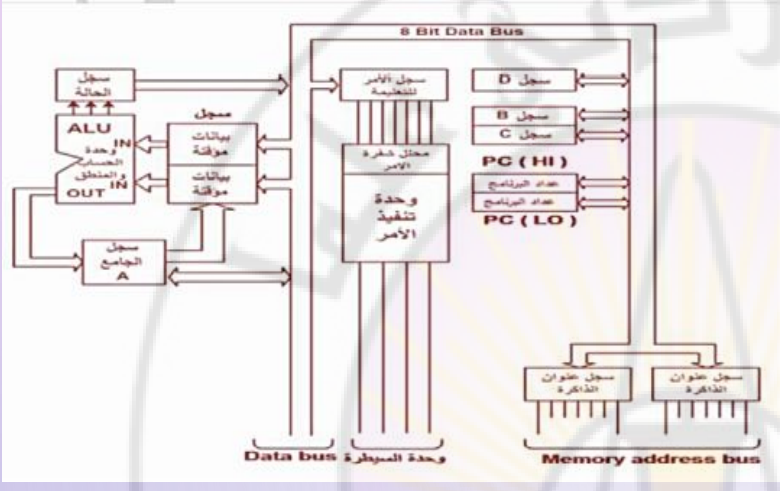
نموذج مبسط لنظام معالجة حاسوبي

1. المعالج :

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

أي أن عمل المعالج يتلخص بمعالجة البيانات الداخلة من خلال إجراء العمليات الحسابية والمنطقية عليها فضلا عن إصدار الأوامر والإيعازات المطلوبة إلى جميع الأجزاء والوحدات الأخرى والسيطرة على أعمالها وتحقيق التزامن المطلوب .

وبالتالي يقوم المعالج بتنفيذ البرنامج المخزن في الذاكرة الرئيسية وإجراء العمليات الحسابية والمنطقية.



تتصل الأجزاء بعضها ببعض من خلال ثلاث مساري وهي :

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

الذاكرة :

تستخدم الذاكرة لتخزين البرامج والمعطيات التي يتعامل معها البرنامج وهي على أنواع :

1. ذاكرة حية RAM : قابلة للقراءة والكتابة وتزول محتوياتها بانقطاع التغذية الكهربائية ، تسمى STATIC RAM وتكون عبارة عن مجموعة بنوك ، وذلك لكي يسهل التعامل معها ، حيث أن كل بنك (جزء) يمكن التعامل معه على أنه BLOCK واحد.
2. ذاكرة ميتة ROM : قابلة للقراءة فقط ولاتزل محتوياتها بانقطاع التغذية .
3. ذواكر أخرى (ثانوية) : أقراص صلبة ، ليزرية ، مغناطيسية ، والمعالجات لاتقوم بتنفيذ البرامج من هذه الذواكر بشكل مباشر وإنما بعد نسخها إلى الذاكرة الأساسية

RAM (Random Access الذاكرة الحية: Memory/ Read-Write memory)	ROM (Read Only الذاكرة الميتة: 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).

4

الذاكرة القابلة للبرمجة PROM :

الذاكرة القابلة للبرمجة Programmable ROM (PROM) أو الذاكرة القابلة للبرمجة مرةً واحدةً OTP NVM (One-Time Programmable Non-Volatile Memory) هي ذاكرةٌ رقميةٌ يكون فيها كل بت مقفلًا بعنصر إلكتروني لا يمكن التعديل عليه. تُكتب البيانات على هذه الذاكرة بعد تصنيعها وتبقى فيها

على الدوام دون إمكانية تعديلها أو إزالتها، ولكتابة محتواها توضع في جهاز يدعى "مبرمج "PROM يُستخدم هذا النوع في الأجهزة المحمولة، والمتحكمات المصغَّرةmicrocontrollers ، وغيرها من الأجهزة الإلكترونية.

5

الذاكرة القابلة لإعادة المسح والبرمجة EPROM

يمكن مسح محتوى هذه الذاكرة Erasable Programmable ROM (EPROM) خلافًا للنوع السابق بتعريض الخلايا للأشعة فوق البنفسجية ثمَّ إعادة الكتابة عليها. تستهلك هذه العملية (المسح وإعادة البرمجة) من عمر الذاكرة وتصل عدد مرات المسح وإعادة البرمجة إلى ١٠٠٠ مرة. يوجد أعلى الذاكرة فتحةٌ تسمح بمرور الأشعة فوق البنفسجة إلى الخلايا لمسح محتواها دفعةً واحدةً، لذا يجب نزع هذه الذاكرة من اللوحة ووضعها على جهاز البرمجة لبرمجتها مجدَّدًا وغالبا توضع على مقبسٍ لتسهيل نزعها وإعادة تركيبها.

6

الذاكرة القابلة لإعادة المسح والبرمجة إلكترونيًا EEPROM

تشبه بنية هذه الذاكرة (EEPROM) (Electrically Erasable Programmable ROM النوع EPROMباستثناء أنَّ عملية المسح والبرمجة تُجرى إلكترونيًّا أي لا داعي لإزالتها من اللوحة أو الدارة.

تجهيزات الإدخال والإخراج (تجهيزات العبور) :

تشكل تجهيزات الإدخال والإخراج (INPUT OUTPUT DEVICES) ، الوسيلة التي يتفاعل فيها المعالج مع العالم الخارجي .

من تجهيزات الإدخال : لوحة المفاتيح ، الفأرة ، لواقط الصوت ، الكاميرات ، الماسحات الضوئية .

أما من تجهيزات الإخراج فنذكر الشاشات والطابعات ومكبرات الصوت.

المسرى (ناقل البيانات) BUS :

يعتبر المسرى او ناقل البيانات وسيلة الاتصال بين وحدة المعالجة المركزية CPU والتجهيزات الطرفية ، أي أنه يقوم بنقل البيانات من المعالج وإليه ، وهو عبارة عن مجموعة من الأسلاك التي تحمل المعلومات على شكل بتات ، وكلما كان عدد خطوط الناقل أكثر كلما كان ذلك أفضل ، وهو عبارة عن ممر باتجاهين لأنه يرسل ويستقبل البيانات وبتألف من (8،32،64،128) أو أكثر من الخطوط المنفصلة .

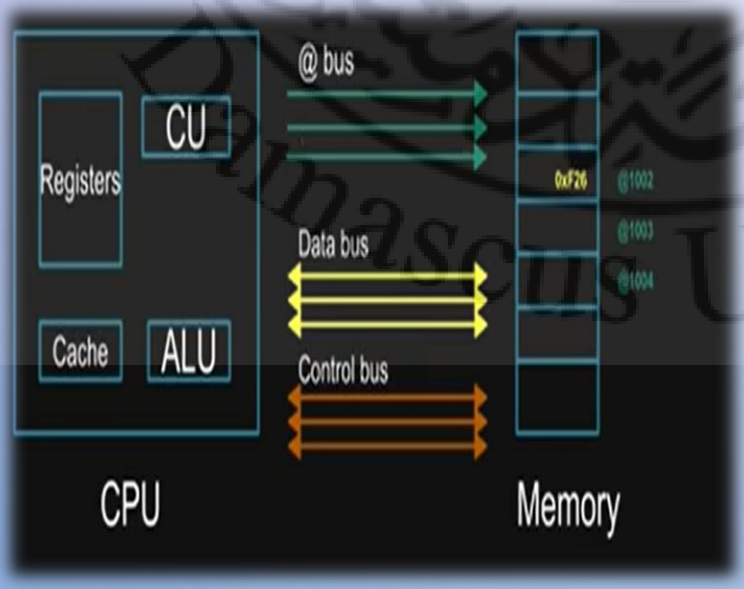
تتصل أجزاء نظم المعالجة الحاسوبية بعضها ببعض من خلال ثلاث مساري وهي :

1. **ADDRESS BUS ناقل إشارات العناوين** : مهمته تحديد عنوان الذاكرة أو البوابة الطرفية (دخل أو خرج)ويقوم بنقل البيانات من الذاكرة الرئيسية أو وحدات الإدخال الخارجية ليتم معالجتها في وحدة المعالجة المركزية ويكون ذو اتجاه وحيد. نرمز لخطوط مسرى العناوين بالرموز A0، A1.....AN

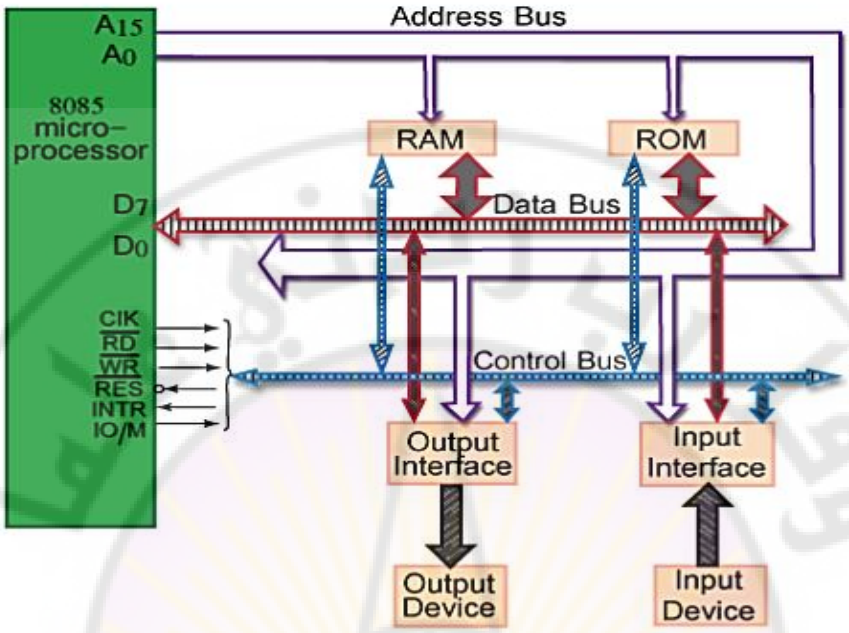
2. **DATA BUS ناقل البيانات**: مهمته نقل البيانات (القيم) في

الاتجاهين ، في حالة الكتابة من يكون الانتقال من المعالج إلى الذاكرة ، اما في حالة القراءة بالعكس ويعطي عدد البيانات الممكن قراءتها بشكل متزامن ، يحدد عرض ناقل البيانات (المعطيات) عدد بتات المعطيات التي يتم التعامل معها ومعالجتها في آن واحد ، نرمز لخطوط مسرى البيانات بالرموز D0، D1.....

3. **CONTROL BUS ناقل إشارات التحكم:** ينقل إشارات معينة خاصة بالقراءة أو الكتابة حسب المطلوب. وهوالناقل الذي يربط المعالج الدقيق بالوحدات والأجزاء الأخرى جيث يصل إشارات التحكم إلى ذواكر تجهيزات الدخل والخرج لضمان صحة تنفيذ العمليات. ويحتوي على إشارات التحكم مثل القراءة والكتابة والمقاطعة والمسك وغيرها .



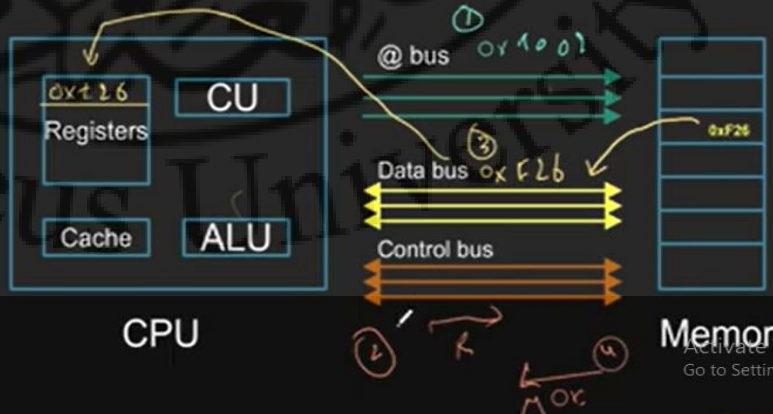
يوضح الشكل التالي مثالاً عن نظام معالجة وطريقة ربط مكوناته ، يبين الشكل مساري الربط بين المعالج والطرفيات حيث نعتبر في هذا لمخطط معالجاً على 8 بتات مثل المعالج Intel 8085.



مساري الربط بين المعالج والطرفيات

Example read data from memory

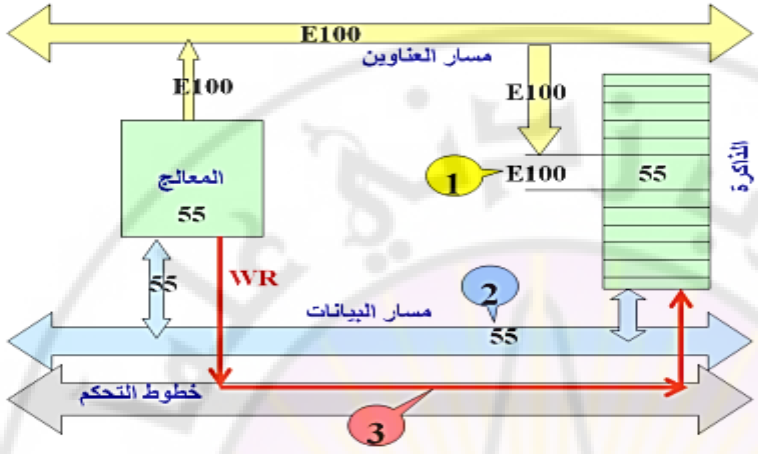
- CPU puts the @ 0x1002 in the @bus
- CPU sends the read signal via the control bus
- Memory puts the content of the @01x002 (0xF26) in the data bus
- Memory sends data-ready signal to the cpu via the control bus
- CPU reads the data from the data bus and puts it in one of its registers



عملية التسجيل على الذاكرة (الكتابة)

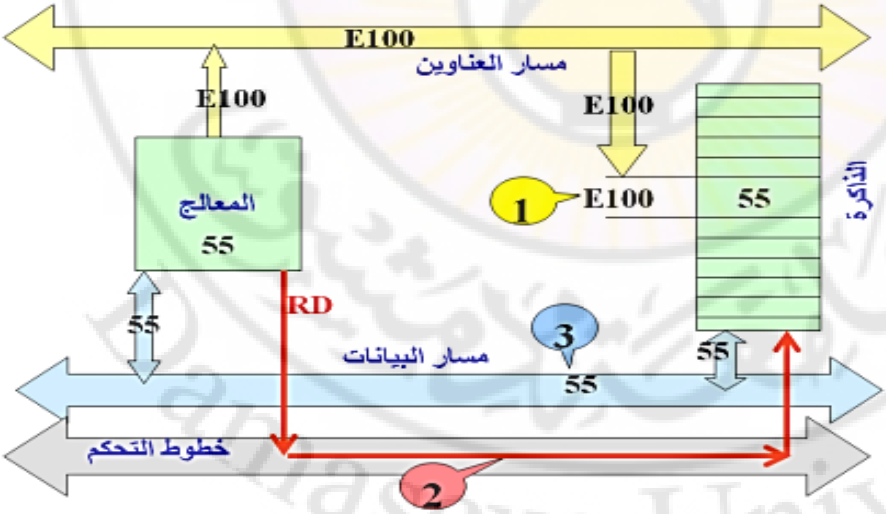
١- يضع المعالج العنوان (١٦ بت) الذى يريد التعامل معه على مسار البيانات (١٦ طرف).
٢- يضع المعالج المعلومة 55H الموجودة بداخله على مسار البيانات.
٣- يقوم المعالج بتنشيط خط التحكم WR بإشارة تخرج منه إلى الذاكرة تخبرها أن الغرض من هذا التعامل هو الكتابة أو التسجيل فى الذاكرة، لذلك يتم تنشيط هذا الخط WR.

لاحظ أن الخطوات الثلاث متعلقة بالمسارات الثلاثة (العناوين، والبيانات، والتحكم)، وبعد تنفيذ هذه الخطوات يتم نقل نسخة من المعلومة 55H وتسجيلها فى العنوان المحدد E100H.



عملية القراءة من الذاكرة

١- يقوم المعالج بوضع عنوان البايت E100H على مسار العناوين.
٢- هذه المرة يقوم المعالج بتنشيط خط التحكم RD ليخبر الذاكرة بأنه يريد القراءة من هذا العنوان.
٣- على الفور تقوم الذاكرة بوضع نسخة من المعلومة الموجودة فى هذه البايت وهى 55H على مسار البيانات، حيث يقوم المعالج بالتقاطها من على مسار البيانات وتسجيلها فى المكان المناسب بداخله.



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:

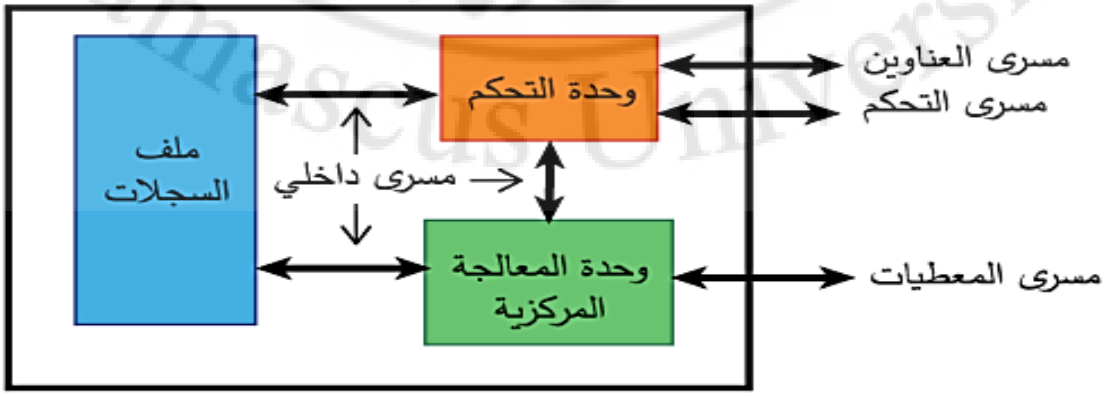
نظم المعالجة المدمجة (Embedded Systems) وهي نظم تقوم بمهمة محددة وتبرمج من قبل المصنع ولا يمكن للمستخدم أن يقوم بإعادة برمجتها مثل أجهزة الهاتف النقال والغسالات الآلية وأفران الميكروويف ...الخ.

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.

تصميم البنية الداخلية لوحدة المعالجة المركزية CPU

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

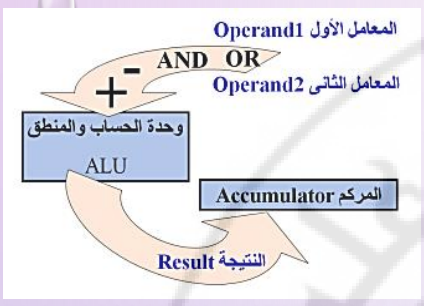
1. وحدة الحساب والمنطق ALU (Arithmetic Logic Unit): تقوم بالعمليات الحسابية والمنطقية.
2. ملف السجلات (Register File): وهو مجموعة من السجلات التي تستخدم لتخزين العناوين والمعطيات التي يتعامل معها المعالج في كل لحظة من تنفيذ البرنامج.
3. وحدة التحكم: إدارة عملية تنفيذ البرنامج من جلب للتعليمات وتفكيكها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



نموذج مبسط لبنية المعالج

وحدة الحساب والمنطق ALU:

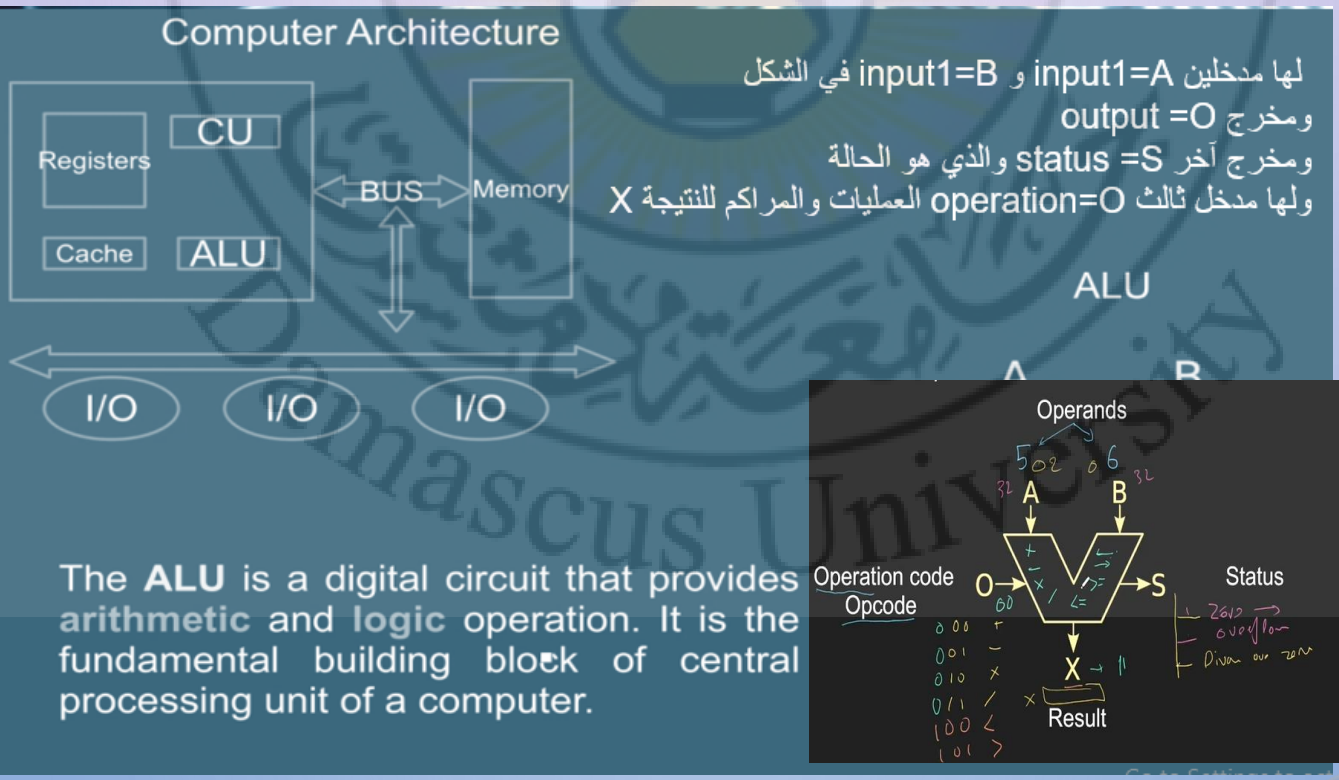
وهي المكون الأساسي في وحدة المعالجة المركزية وتشكل قلب أي معالج ، تقوم بجميع العمليات الحسابية والمنطقية ، حيث تتكون من وحدتين فرعيتين :



وحدة الحساب ARITHMETIC UNIT : لتنفيذ العمليات الحسابية

وحدة المنطق LOGIC UNIT: لتنفيذ العمليات المنطقية

أي أنها عبارة عن دارة تركيبية تقوم بمجموعة من العمليات الحسابية والمنطقية بناء على مايتم انتخابه من مداخل ، والخرج الخاص بها يكون على مراكم يتم الاحتفاظ به حتى يتم استدعاؤه لها مدخلين 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



ماهي مكونات ALU:

1. **OPERANDS:** هي الأشياء التي سوف تعمل عليها الآلة ، مثلا حالة جمع عددين Aو B ، يكون العدد الأول للمدخل A والثاني للمدخل B ، ثم تقوم عملية الجمع ضمن ALU وإخراج القيمة عن طريق X ...
2. **RESULT :** والتي من خلالها يتم إخراج القيمة المطلوبة .
3. **OPERATION UNIT :** وحدة الحساب والمنطق لاتقوم بعملية حسابية واحدة فقط ، أي أنها تدعم مجموعة من العمليات الحسابية (+،-،*،<،=.....إلخ) ، والعمليات المنطقية مثل (SHIFT إزاحة لليمين اواليسار .....إلخ). ويتم استنتاجها كجزء من الـ OPCODE الخاص بالتعليمةعندما نقوم بإعطاء المدخلات فإن وحدة الحساب والمنطق لاتكتفي بذلك وإنما تحتاج إلى تحديد نوعية وطبيعة العملية التي سوف تقوم بإجرائها على المدخلات ، وذلك عن طريق مدخل ثالث وهو OPCODE .
4. **المخرج STATUS :** وهومسجل الحالة يمثل حالة الخرج للعملية التي قامت بها وحدة الحساب والمنطق .وهو يعطي معلومة عن سير العملية الحسابية او المنطقية

4. مسجلات العزل : تمثل مسجلات العزل (BUFFER REGISTER) الواجهة البينية بين المعالج والذواكر المرتبطة به . ومسجلات العزل النظامية هي :

- مسجل عنوان الذاكرة MEMORY ADDRESS REGISTER (MAR)
- مسجل عزل الذاكرة MEMORY BUFFER REGISTER (MBR)

يتم وصل المسجل MAR مع مرابط العناوين الخارجية للمعالج ويحتوي هذا المسجل على العنوان المطلق لعنوان المعطيات أو التعليمة التي تريد النفاذ إليها . أما المسجل MBR والمعروف بمسجل معطيات الذاكرة فيكون موصول مع مرابط المعطيات للمعالج ويخزن كل المعطيات التي تتم قراءتها او كتابتها في الذاكرة.

5. مسجل الحالة : يخزن مسجل 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 .

6. مؤشر المكدس : يستخدم مسجل 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 مسجل عام خلال تنفيذ إجرائية جزئية او عند تخديم مقتطعة ما . يتم نقل المعطيات من المسجل العام إلى المكدس باستخدام تعليمة PUSH في بداية الإجرائية ، وفي نهاية التابع يتم استعادة المعطيات باستخدام تعليمة POP ، يستخدم المعالج المكدس كذاكرة مؤقتة لأن عملية تخزين واستعادة المعطيات باستخدام التعليمتين PUSH و POP يتم بشكل أسرع من نقل المعطيات من وإلى الذاكرة باستخدام التعليمة MOVE

7. مسجلات عامة :

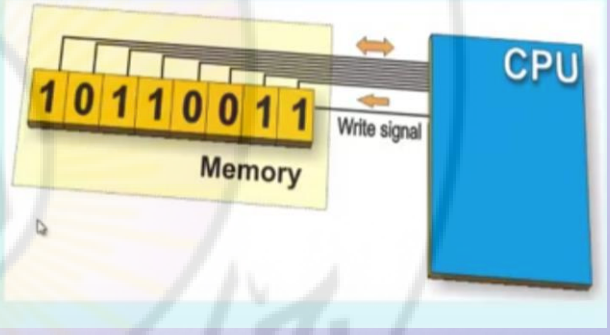
المسجلات العامة (GENERAL – PURPOSE REGISTER) مسجلات للاستخدام العام ، تستخدم بشكل صريح لتخزين معلومات المعطيات والعناوين ، تستخدم مسجلات المعطيات للعمليات الحسابية والمنطقية ، اما مسجلات العناوين فتستخدم للعنونة غير المباشرة كمؤشرات ذاكرة وبفضل هذه المسجلات يتم تسريع تنفيذ الخوارزميات عن طريق النتائج المرحلية وتجنب النفاذ إلى الذاكرة الخارجية لهذا الغرض ، وذلك لأن النفاذ إلى الذاكرة الخارجية يكون أبطأ من النفاذ إلى مسجلات المعالج الداخلية .

8. مسجلات مؤقتة :

تستخدم المسجلات المؤقتة (TEMPORARY REGISTERS) لتخزين المعطيات إذا لزم المر خلال تنفيذ التعليمة ، وهي مسجلات مخفية بشكل كامل عن مستخدم المعالج .

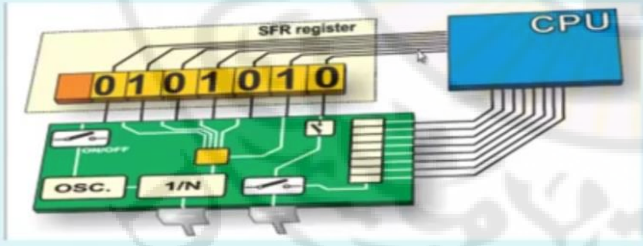
(2) REGISTER

- a register or a memory cell is an electronic circuit which can memorize the state of one byte.



(3) SFR REGISTERS

- Special function registers .
- every microcontroller has a number of registers (SFR) whose function is predetermined by the manufacturer .
- such as timers, A/D converter, oscillators and others .



وحدة التحكم CU(CONTROL UNIT)

وهي الوحدة التي تقوم بقيادة وتوجيه جميع العمليات التي تحدث في المعالج ، وهي التي تقوم بتنسيق عمل كل أقسام المعالج والتجهيزات الطرفية ، وذلك بالتزامن مع نبضة من نبضات الساعة الخاصة بالمعالج ، حيث تقوم بالتحكم بكافة العمليات المنفذة وتشرف على تسلسل تنفيذ العمليات ،كما تقوم بتنسيق عمل جميع تجهيزات الدخل والخرج، وهي تستخدم وحدة فك تشفير ثنائية لتحويل التعليمات المشفرة إلى إشارات توقيت وتحكم توجه تشغيل المكونات الأخرى .

أي انها تقوم بمايلي :

1. قراءة التعليمة من الذاكرة.
2. فك ترميز التعليمة.
3. إرسال إشارات تنفيذ التعليمة

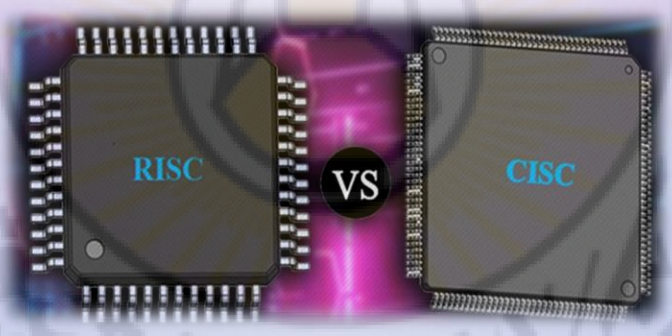
الأجزاء التي تشرف عليها وحدة التحكم :

1. MAR (MEMORY ADDRESS REGISTER) وحدة العنونة : وهو الجزء الذي يقوم بأخذ الأمر من قبل العداد PC ونقله إلى ممر المعطيات لإرساله إلى الذاكرة.
2. PC(PROGRAM COUNTER) وهو يقوم بتوليد عنوان الحجرة الذاكرية التي تحتوي على التعليمة التالية التي سوف يتم تنفيذها.
3. MBR(MEMORY BUFFER REGISTER) وهو عبارة عن مسجل يقوم بتخزين شيفرة التعليمة التي تم إحضارها من الذاكرة.
4. IR(INSTRUCTION REGISTER) وحدة التعليمات : وهو مسجل يحتوي على التعليمة الحالية التي سوف تنفذ في وحدة الحساب والمنطق
5. TIMER وحدة التحكم والتوقيت الزمني : وهو دارة توقيت تقوم بتوليد الفترات الزمنية لتنفيذ التعليمات.

بنى مسجلات التحكم

معمارية مجموعة التعليمات

CISC & RISC



من وجهة نظرية بنيان الحاسب ، يمكن تصنيف وحدات التحكم وفقا لطريقة بنائها إلى نمطين هما CICS (COMPLEX INSTRUCTION SET COMPUTER) مجموعة التعليمات المعقدة وRISC (REDUCED INSTRUCTION SET COMPUTER) أي مجموعة التعليمات المخفضة .
أي انها :

1. وحدات تحكم مرمزة CISC:
   - تصميم وحدة التحكم من هذا النمط سهل .
   - مرونة اكثر في تنفيذ التعليمات ولكن على حساب السرعة .

تنفذ التعليمات عن طريق برامج صغيرة تحوي عدد من التعليمات الجزئية الصغيرة ، يقدم هذا النمط مرونة اكثر في تنفيذ التعليمات من وحدات التحكم العتادية ولكنها تستغرق وقت طويل عند تنفيذ المهمة ، هدفها توفير مجموعة التعليمات التي تدعم اللغات عالية المستوى ومن ثم تسهيل عملية الترجمة COMPILING.

2. وحدات تحكم عتادية RISC:
   - بنية هذه الوحدة صغيرة وسريعة .
   - صعبة التصميم.

تستخدم مجموعة التعليمات ذات الحجم الصغير والمحسنةعلى نحو كبير ، حيث تتكون من وحدات منطقية تقوم بتوليد تسلسل الإشارات المناسبة لكل تعليمة من التعليمات . وتكون بنية هذه الوحدة صغيرة وسريعة ، يكون عدد التعليمات الممكنة في هذا التصميم قليل .

- تعليمة تقوم بمسح محتويات الحجرة 100h من الذاكرة وذلك باستخدام معالج CISC:

```
clear 0x0100
```

- برنامج مكافئ باستخدام معالج RISC:

```
load R , 0x0000
store R , 0x0100
```

أهمية المعالجات والمتحكمات الصغرية في الأتمتة الصناعية:

في الواقع، يوجد الكثير من التطبيقات الإلكترونية البسيطة التي لا يجدي اقتصادياً أتمتتها عن طريق حاسوب شخصي ولذلك بقيت هذه التطبيقات ضمن إطار الدارات التقليدية التي لا تعتمد على أي نوع من أنواع البرمجيات وإنما هي عبارة عن مكونات صلبة بحتة (Hardware) يحتاج تعديلها إلى إعادة بناء النظام من جديد وهذه عملية صعبة جداً وجاء المعالج الصغري ليحل هذه المشكلة عن طريق إمكانية البرمجة مع إمكانية لتعديل البرامج التي يشغلها المعالج ليتحكم بعمل الأجهزة المربوطة معه، و تم أيضاً اختراع جهاز حاسوب لا يختلف بحجمه عن حجم أي دارة متكاملة بحيث يصبح استخدامه في أي دارة اقتصادياً، وبالفعل تم تصنيع أول جهاز حاسوب كامل على شكل دارة متكاملة في منتصف سبعينات القرن السابق ولقد اقتصر استخدامه آنذاك على المجال العسكري والفضائي و ما لبث أن انتشر في الأسواق باسم (المتحكمات الصغرية – PLC) وأصبح متداولاً في سوق الدارات الكهربائية والإلكترونية.

إن أي عملية أتمتة صناعية تحتاج إلى أربع أجزاء رئيسية هي:

1- آلة ميكانيكية تمتلك كافة الحركات الممكنة لليد البشرية التي تنجز العمل قبل عملية الأتمتة.
2- دارات تحكم وقيادة إلكترونية يرتبط خرجها بمحركات أو مكابس الآلة من جهة ويرتبط دخلها بالمخارج الرقمية للمعالج الصغري أو للمتحكم أو الكومبيوتر المستخدم من أجل القيام بعملية الأتمتة.
3- طرفيات تتحسس عن حالة الآلة أو المشغولة وتدعى الحساسات Sensors ولها أنواع كثيرة.
4- المتحكم : وهو عبارة عن معالج صغري أو حاسوب مجمع في شريحة تختلف مقدراته من تطبيق إلى آخر وهو الذي يقوم بقيادة وإدارة جميع العمليات المستخدمة في الآلة.

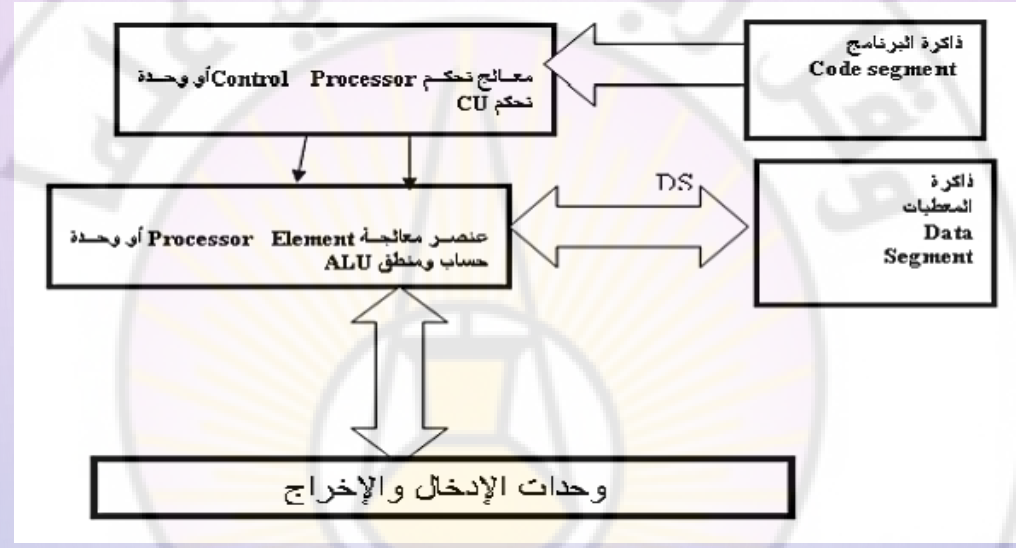
تعريف نظم المعالجات :

إن إضافة ذاكرات إلى معالج مع وجود منافذ PORTS يعطينا نظام معالج

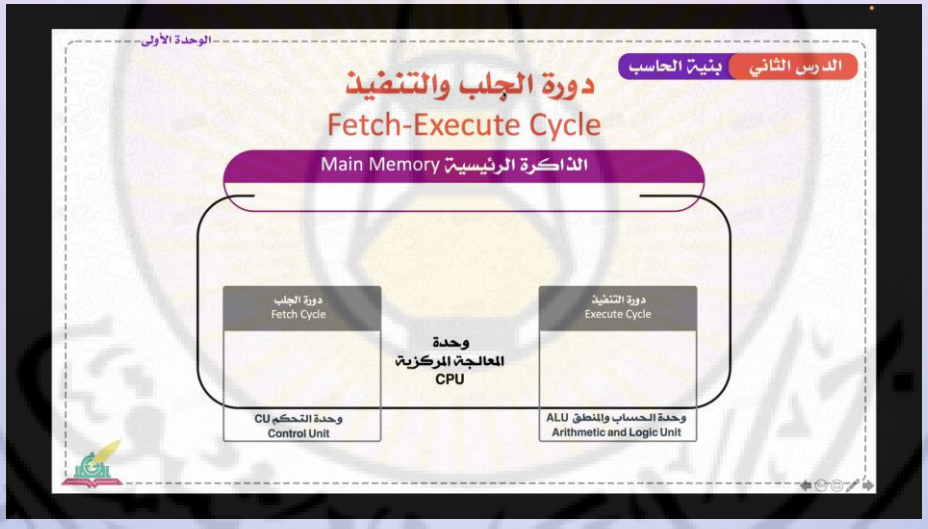
(نظام المعالج = معالج + منافذ + ذواكر)

بالنسبة لمعالج واحد يمكن بناء او تصميم أنواع كثيرة من نظم المعالجات بحسب تنوع الذاكرات التي تتعامل معها او تخزن فيها ، ويمكن أن نسمي الحاسوب في حالة خاصة نظام معالج صغري ، وهذا يعني أن مفهوم نظام المعالج الصغري هو أي نظام قياس اوتحكم يدخل فيه المعالج الصغري كعنصر أساسي.

وهذا يحتاج إلى نظام مسارات لضمان عمل البرامج التي ينفذها المعالج أوالحاسوب



مراحل عمل المعالج – دورة الجلب والتنفيذ
FETCH – EXECUTE CYCLE



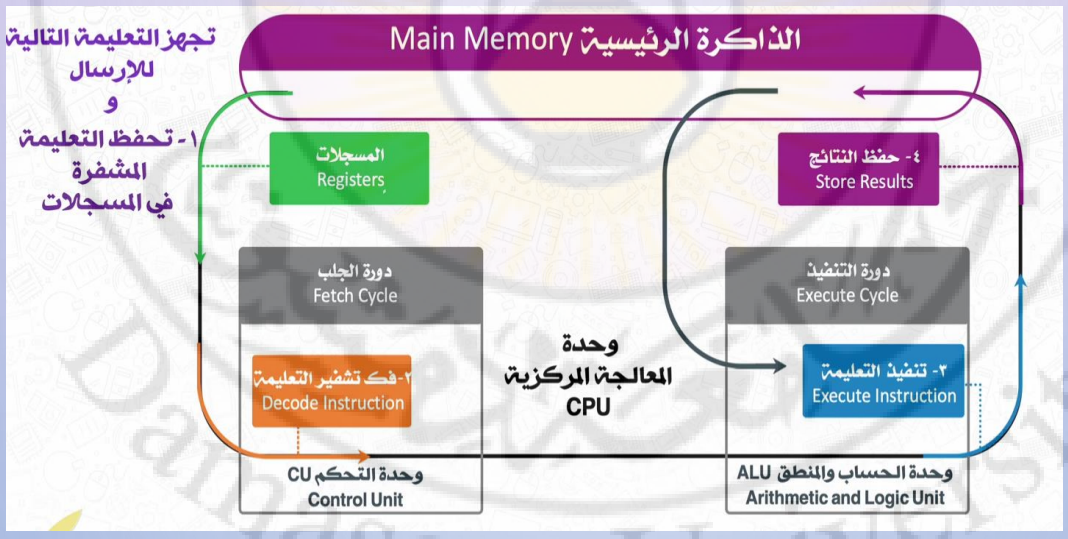
دورة الجلب : تحدث في وحدة التحكم حيث يقوم المعالج بجلب التعليمة من الذاكرة الرئيسية ثم يقوم بحفظها داخل المسجلات (حيث ان المسجلات عبارةعن ذواكر موجودة داخل المعالج cpu) يتم حفظ التعليمة المشفرة داخل المسجلات ومن ثم يتم تجهيز التعليمة التالية للإرسال .

(جلب التعليمة المجهزة إلى المسجلات – فك شيفرة التنفيذ في وحدة التحكم – تنفيذ العملية في وحدة الحساب والمنطق – الحصول على المزيد من البيانات من الذاكرة والرجوع إلى دورة التنفيذ- حفظ النتائج في الذاكرة) وهكذا.....

وبعدها يتم فك تشفير التعليمة داخل وحدة التحكم cu لتدخل إلى وحدة الحساب والمنطق ALU حيث تبدأ دورة تنفيذ التعليمة ..

دورة التنفيذ : يتم تنفيذ التعليمة داخل وحدة الحساب والمنطق ، وبعد ذلك تحفظ النتائج داخل الذاكرة الرئيسية .

إذا كان هناك أوامر أخرى مثل (عمليات حسابية +،-،*،/ أو منطقية ...OR,XOR)سوف تقوم الذاكرة الرئيسية بالحصول على المزيد من البيانات من دورة التنفيذ حتى تقوم بتنفيذ هذه العملية ، ثم يتم حفظها ...



نظام توصيل المسارات بين المعالج والذاكرة

BUSSING SYSTEM

وهما عبارة عن نموذجين تعتمد عليهما أنظمة الكمبيوتر ، والاختلاف الكبير بين هاتين البنيتين ينشأ وفقا للطريقة التي يتم فيها فصل وحدة المعالجة المركزية عن الذاكرة.

وهي قائمة على تقسيم المسارات بين المعالج والأجهزة المحيطة إلى ثلاث مسارات (العناوين والبيانات والتحكم)

معمارية فون نيومان:

تعتمد على ناقل وحيد لنقل البيانات والتعليمات بين الذاكرة ووحدة المعالجة المركزية.

في هذه البنية البرنامج يتم وضعه في الذاكرة وتقوم وحدة المعالجة المركزية باستدعاء أوامر البرنامج واحدا تلو الآخر حتى يتم الانتهاء من البرنامج ، لذلك تمتاز هذه الطريقة بوجود ذاكرة واحدة تتعامل معها وحدة المعالجة المركزية ، تحتوي البرامج والبيانات معا ، من خلال المسارات الثلاثة.

معمارية هارفارد :

تختلف ذاكرة البيانات عن ذاكرة التعليمات ولكل واحدة خطوط عنونة وتحكم وممر معطيات مختلف عن الأخرى ، أي تم فصل الذاكرة على جزأين جزء خاص بالبيانات يتم التعامل معه بوحدة البايت ، ويتصل مع وحدة المعالجة المركزية من خلال مسار بيانات ومسار عناوين خاص خاص بهذا الجزء من الذاكرة ، اما الجزء الثاني هو ذاكرة خاصة بالبرنامج فقط تكون وحدة التعامل معه هي عدد من البتات يساوي أكبر عدد من بتات أي أمر من الأوامر.

تاريخ المعالجات وتطورها:

1. 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.

اتجاهات تطور المعالجات :
هناك ثلاثة اتجاهات لتطوير المعالجات :

أولا - تسريع المعالجة من خلال :

a)-زيادة طول الكلمة من 4bit إلى 16 bit إلى 32 bit إلى 64bit .

b)- زيادة عدد أنواع المعطيات الممكن معالجتها و زيادة أنواع العنونة .

c)-زيادة سرعة المعالجات من خلال زيادة تردد الساعة وزيادة عدد الوحدات الفرعية التي ممكن أن تعمل معاً داخل المعالج , فالمعالج Pentium III الذي سرعته1100MHz مكون من11 وحدة تعمل على التوازي مع تردد ساعة 100MHz فتصبح السرعة وكأنها 1100 ، و منذ فترة تتوفر معالجات مزدوجة Dual (أي معالجين كاملين ضمن شريحة واحدة) بينما حالياً تتوفر معالجات ثمانية و ربما قريباً تظهر معالجات تحتوي على عدد مضاعف من المعالجات ضمن شريحتها.

d)-تمثيل بعض وظائف نظام التشغيل بالأجهزة Hardware داخل المعالج (مثل إدارة الذاكرة و حماية الموارد الحاسوبية)

ثانياً- رفع عدد الوحدات الموجودة في قطعة أو شريحة واحدة(رفع درجة التكامل) حتى أنه منذ عام 1970 تم تجميع كامل الحاسوب في قطعة واحدة .

و يمكن أن نقارن عدد الترانزستورات الموجودة داخل المعالج فالمعالج 4004 يحوي على2250 ترانزستور، ليتزايد هذا العدد إلى 70000 ترانزستور في المعالج 68000 و إلى 130000ترانزستور في المعالج 80286 الذي بدوره يحوي على دارات لإدارة الذاكرة, بينما في المعالج 80386 أصبح لدينا 275000 ترانزستور و المعالج Pentium1.3 يمتلك مليوناً من الترانزستورات.

***ثالثاً*- البحث عن تقنيات جديدة للمعالجات مثل تقنيةRisc(Reduced Instruction set computer)**

والتي تعني الاختصار الموجه للتعليمات أي تصنيع متحكمات أو معالجات بعدد مختصر من التعليمات يلاءم مجال معين من مجالات التحكم والقياس.

معايير اختيار المعالج :

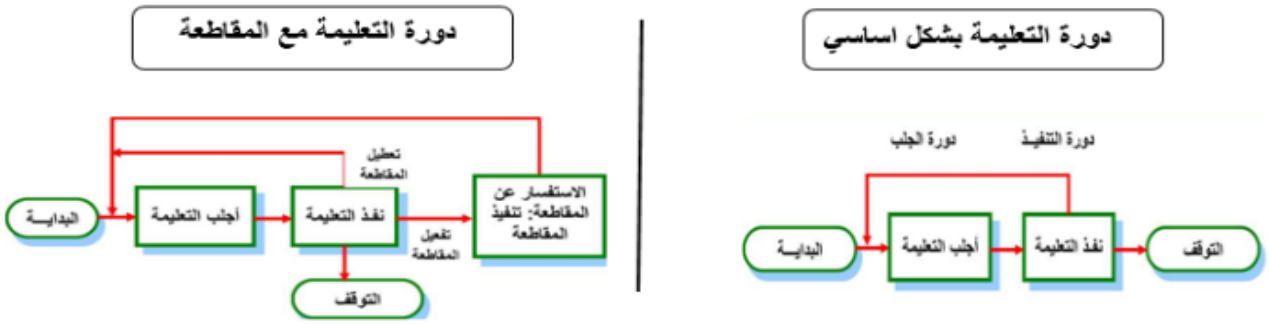
1. السعر : من اهم العوامل يجب ان يكون اقتصاديا .
2. استهلاك الطاقة : تتغير الاستطاعة المستهلكة من قبل المعالج تبعا لجهد التغذية وسرعة التشغيل .
3. الأداء : يجب معرفة درجة تعقيد البرامج التي سيشغلها المعالج ومتطلباتها من حيث السرعة وحجم الذاكرة .
4. التوفر : أن يكون متوفر ويمكن الحصول عليه بسهولة.
5. الدعم البرمجي : وذلك من أجل تطوير البرامج اللازمة لعمل المعالج فيجب توفر الأدوات البرمجية الضرورية .
6. كثافة الرماز : CODE هي النسبة بين حجم رماز المصدر وحجم النسخة التنفيذية ، فكلما كانت النسخة التنفيذية أصغر أفضل لأنها تتطلب حجم ذاكرة أقل .

المحاضرة الثالثة

خصائص المعالجات

البنية الداخلية لمعالج

8086INTEL



رسم يوضح العلاقة مابين المعالج و الذاكرة الرئيسية اثناء دورة الالة :

خصائص المعالجات :

هناك عدة معايير يمكن أن نقارن عن طريقها بين الأنواع المختلفة للمعالجات وهي:

1-مقدار التغذية أو جهد البطارية: ففي المعالجات القديمة كان أكثر من جهد فالمعالج 8080 مثلاً فيه ثلاث جهود (+5v,+12v, -5v)بينما المعالجات الحديثة أصبح لها جهد مستمر واحد وهو +5vبالنسبة لمعالجات Intel.

2- تردد الساعة: يتزايد باستمرار من 0.5 MHz في المعالج 8008 إلى5MHz في المعالج 8086إلى 100MHz في المعالج PentiumIII.

3-عدد الأوامر أو التعليمات : كان 48 في المعالجين 8008 ثم أصبح 78 في المعالج 8080 ثم عدة مئات في المعالج 8086 بينما مع وجود تقنية Risc ثم تقليل التعليمات بشكل ملحوظ لتصبح بين 24 و60 تعليمة.

4 - زمن معالجة عملية الجمع : فقد كان في المعالج 8008 20µs لينخفض إلى 2µs في المعالج 8080 وإلى 0.8µs في المعالج 8086.

5- مساحة العنونة : أي الذاكرة التي يستطيع المعالج السيطرة عليها و التعامل معها فقد كانت 16KByte في المعالج 8008 لترتفع إلى 1MB في المعالج 8086 .

6- عدد الترانزستورات في الشريحة : ذكرناه سابقاً.

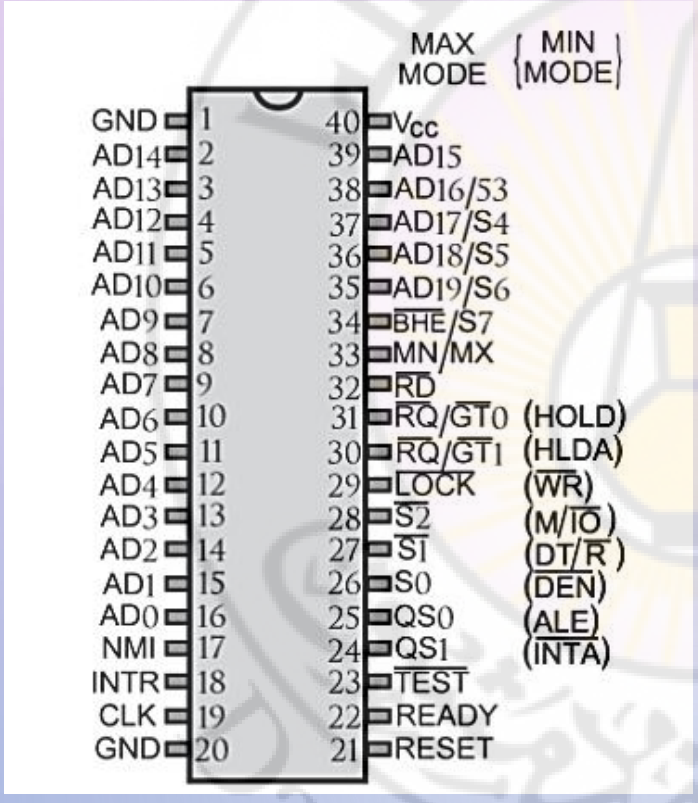
7- نوع تكنولوجيا التصنيع : فقد كان في المعالج 8008 PMOS بينما أصبحت في المعالج 8080 NMOS وفي المعالج 8086 HMOS .

8- عدد الوحدات المستقلة التي يضمها الحاسوب :

فحاسوب 8008 يحتوي على الأقل 20 وحدة بينما الحاسوب 8080 يحتوي على 5 وحدات (دارات متكاملة) بينما تعتبر الشريحة 8048 بكاملها حاسوب .

المواصفات الأساسية للمعالج :

1. يعمل على معطيات 16 بت.
2. له مسرى عناوين بعرض 20 بت أي يمكن النفاذ إلى ذاكرة بسعة 1 ميغا بايت.
3. يستطيع النفاذ إلى K64
4. يعمل باستخدام ساعة تصل إلى 10MHZ
5. يعمل باستخدان تغذية وحيدة 5V.
6. معلب ضمن دارة تكاملية ذات 40 مربط.



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.

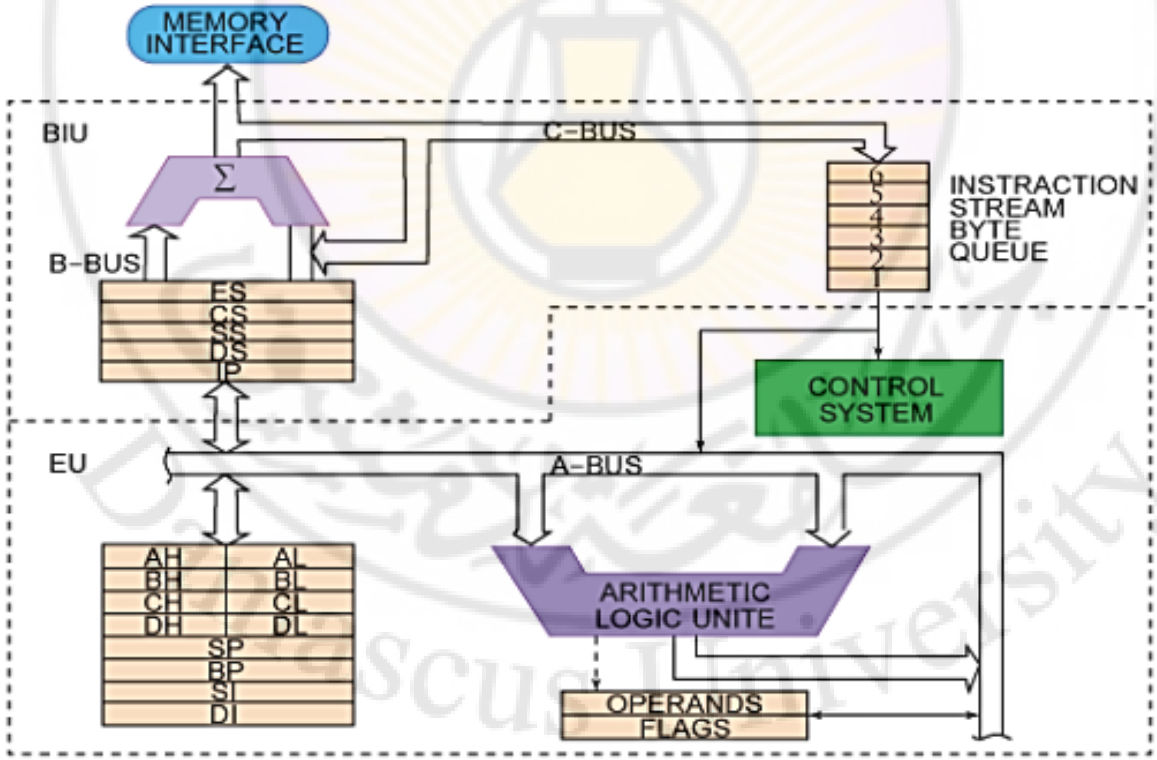
ومن الضروري، قبل كتابة برامج للمعالج المصغري 8086، فهم بنيته الداخلية ومعرفة سجلاته الداخلية.

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

وحدة واجهة المسرى: الرتل – سجلات القطاعات – مسجل التعليمات

أولا – 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.

ثانيا – سجلات القطاعات :

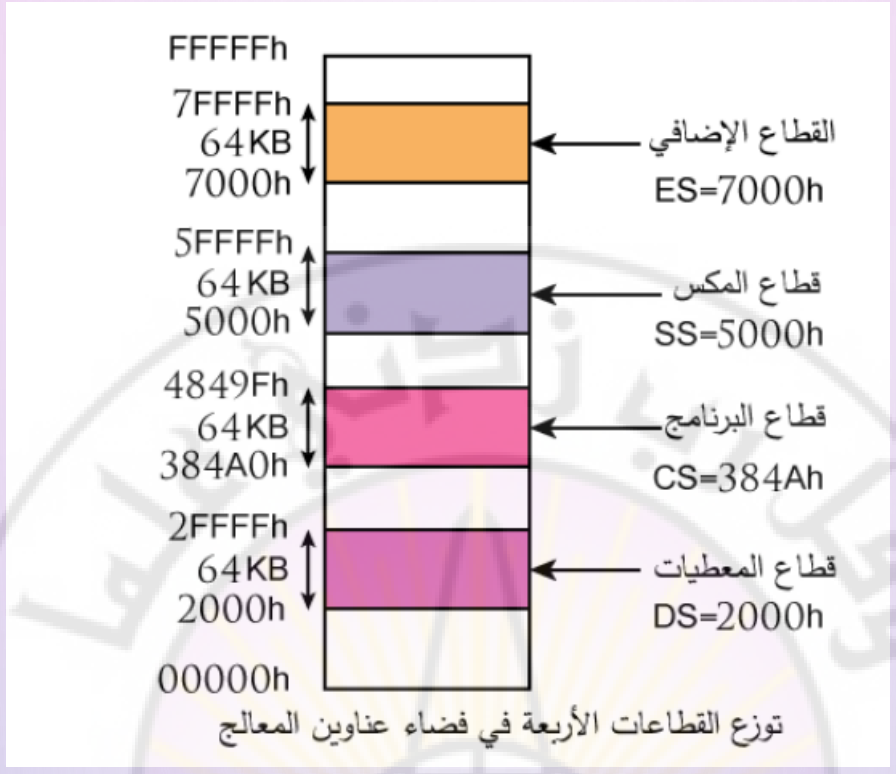
يقسم فضاء عنونة الذاكرة إلى أربعة قطاعات، كل منها بطول 64 كيلوبايت كحد أقصى، وهي:

- 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:

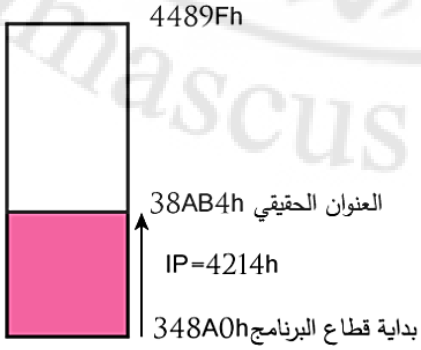
1. سجل قطاع البرنامج CS (Code Segment)
2. سجل قطاع المكدس SS (Stack Segment)
3. سجل قطاع المعطيات DS (Data Segment)
4. سجل القطاع الإضافي ES (Extra Segment)



توزع القطاعات الأربعة في فضاء عناوين المعالج

ثالثا – 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

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 مثلاً 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 و AH لحصول على زوج يسمى AX. وكذلك يدمج BL و BH للحصول على BX, ونفس الشيئ بالنسبة للبقية.

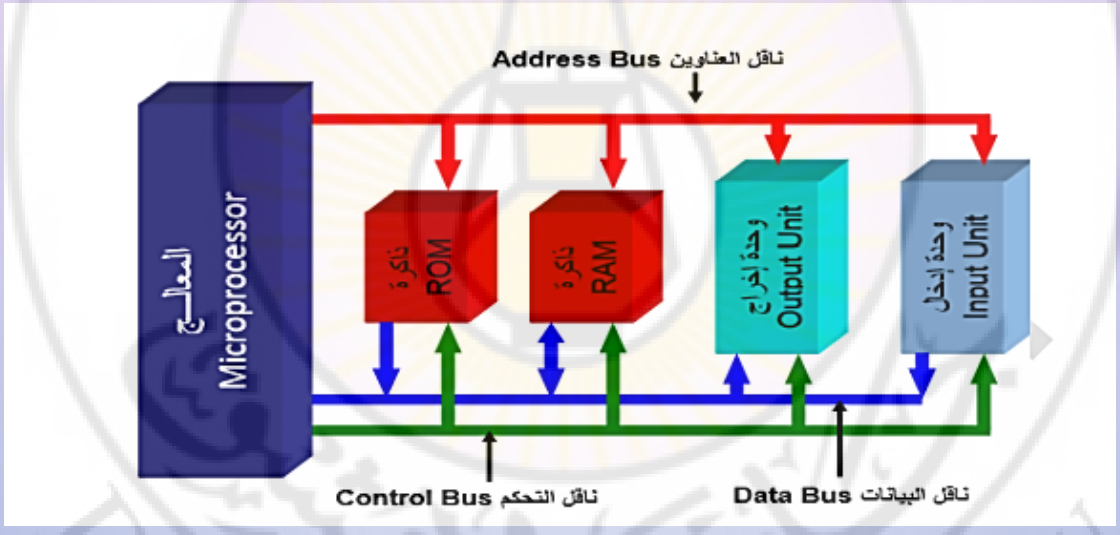
5.2.2. سجل مؤشر المكدس SP:

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.

طريقة اتصال المعالج الدقيق بالأجهزة المختلفة عبر استعمال النواقل



المحاضرة الرابعة

الوحدات المحيطية والدخل والخرج

–

البرمجة بلغة التجميع

استراتيجيات العبور:

من أهم استراتيجيات النفاذ بين تجهيزات العبور والذاكرة الأساسية:

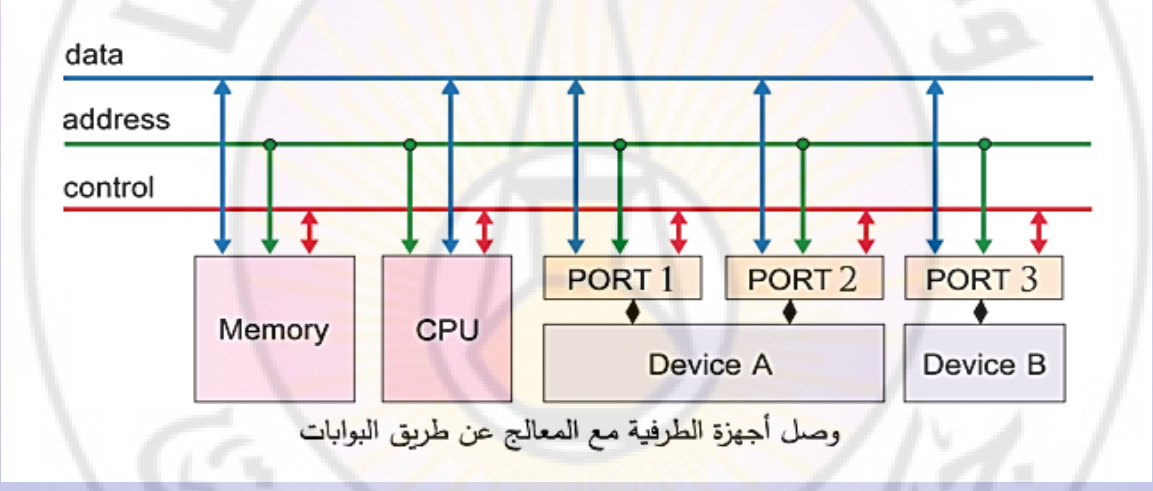
1. طرق العبور المبرمجة (Programmed IO):

- 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.

عمليات العبور المبرمجة :



وصل أجهزة الطرفية مع المعالج عن طريق البوابات

كل عملية نقل معطيات من جهاز العبور تتم عن طريق تنفيذ تعليمات من قبل المعالج . يتخاطب المعالج مع أجهزة العبور والذاكرة الأساسية بواسطة مسرى وحيد مشترك باستخدام نفس خطوط العنونة ، تسمى كل عقدة وصل بين المسرى الأساسي وجهاز العبور بوابة عبور .

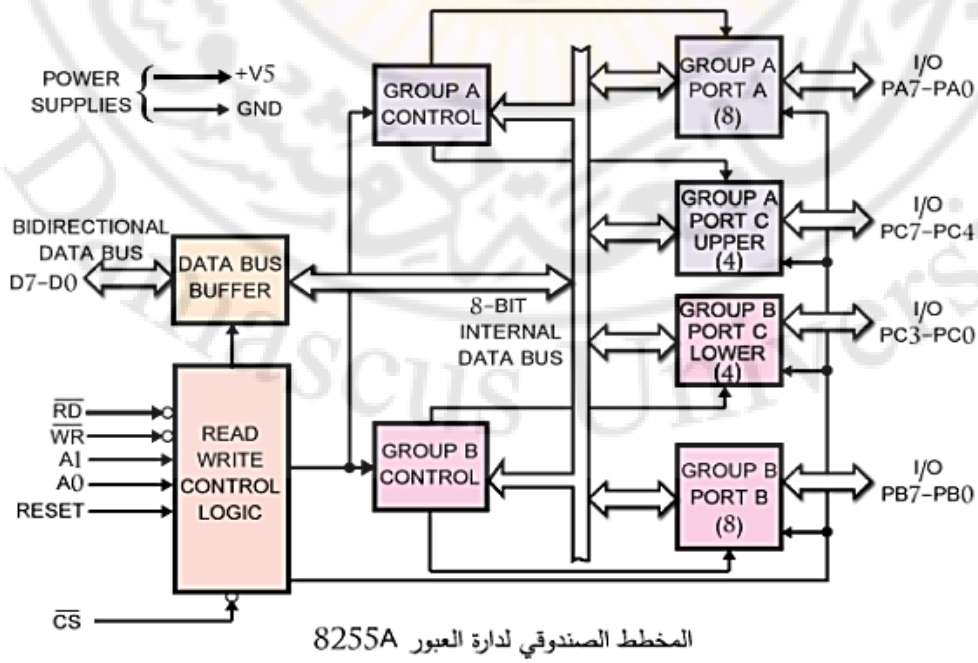
تعليمات العبور الأساسية :

1. تعليمة الكتابة OUT X.
2. تعليمة القراءة IN X .

طرق نقل المعطيات :

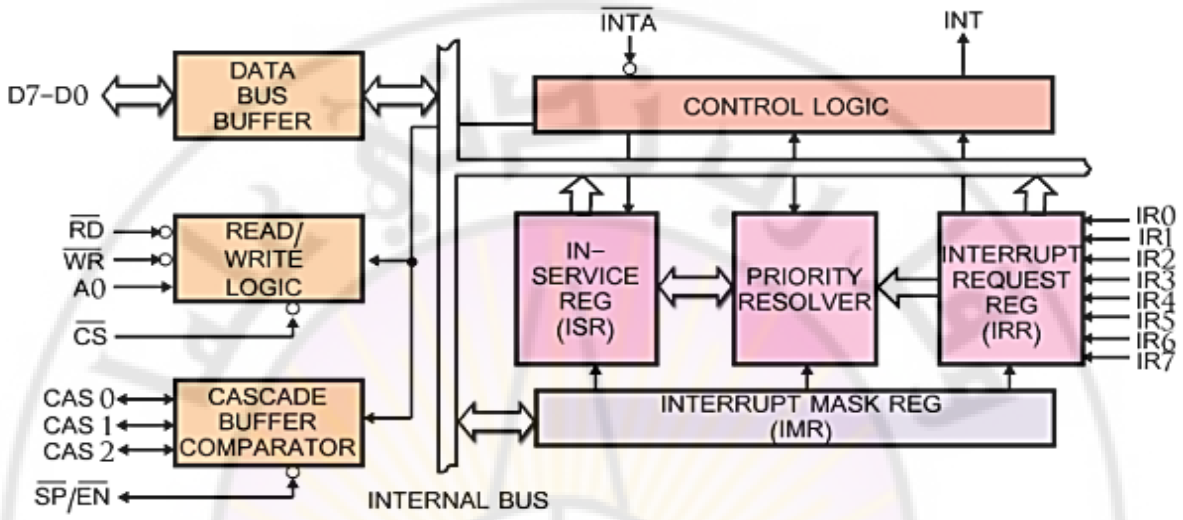
1. طريقة التقصي : POLLING بواسطة بوابة إضافية تسمى بوابة الحالة وجاهزية بوابة المعطيات .
2. طريقة القدح او المقاطعة : حيث يقوم جهازالعبور بتوليد إشارة المقاطعة ليقوم المعالج بالنفاذ إلى جهاز العبور.
3. عمليات العبور بالمصافحة الوحيدة : يضع الجهاز المحيطي المعطيات على مخارجة ويرسل إشارة للمعالج STB الذي يكشف الإشارة بالتقصي أو المقاطعة .
4. عمليات العبور بالمصافحة المزدوجة : يستعلم المرسل عن جاهزية المستقبل بواسطة خط القدح على المستوى المنخفض ثم يضع المرسل المعطيات ويجعل المستوى المنطقي لخط القدح عالي ثم يقوم المستقبل بوضع مستوى منطقي منخفض للاعلام بان المعطيات قد تم قراءتها.

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



المخطط الصندوقي لدارة العبور 8255A

- 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

لغة التجميع ASSEMBLY

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.

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.

تقنيات البرمجة بلغة التجميع :

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.

التعليمات الحسابية والمنطقية 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		

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

تعليمات القفز :

تعليمات 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
```

تعليمات القفز المشروط :
هي التعليمات التي تقوم بالقفز عند تحقق شرط معين وهي تعتمد على رايات سجل الحالة الستة التي تقوم وحدة الحساب والمنطق بوضع قيمتها بعد كل تعليمة

مثال:

```
JC    Save
```

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

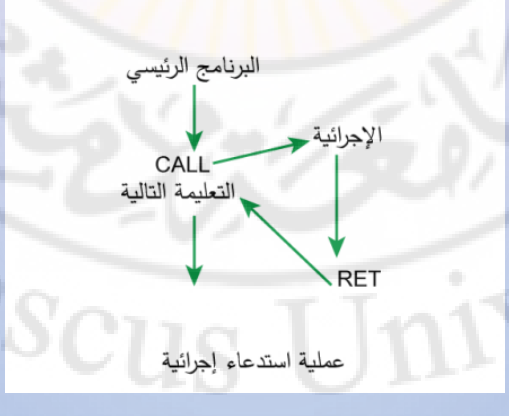
البرامج الفرعية والإجرائيات :

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

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.



عملية استدعاء إجرائية

المقاطعات
INTERRUPTS

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.

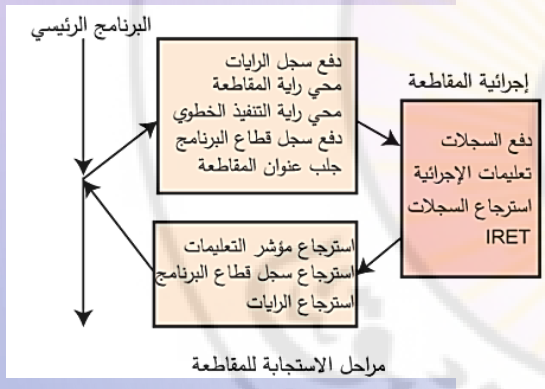
1. وصف مقاطعات المعالج:

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.



أنواع المقاطعات فى المعالج 8086

هناك عدة أنواع للمقاطعات في المعالج 8086

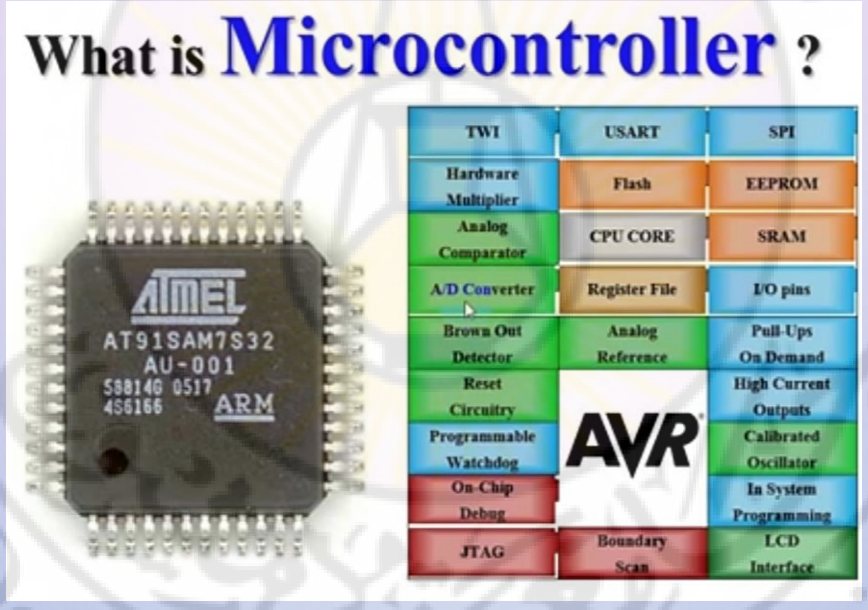
المقاطعات المخصصة:

- 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.

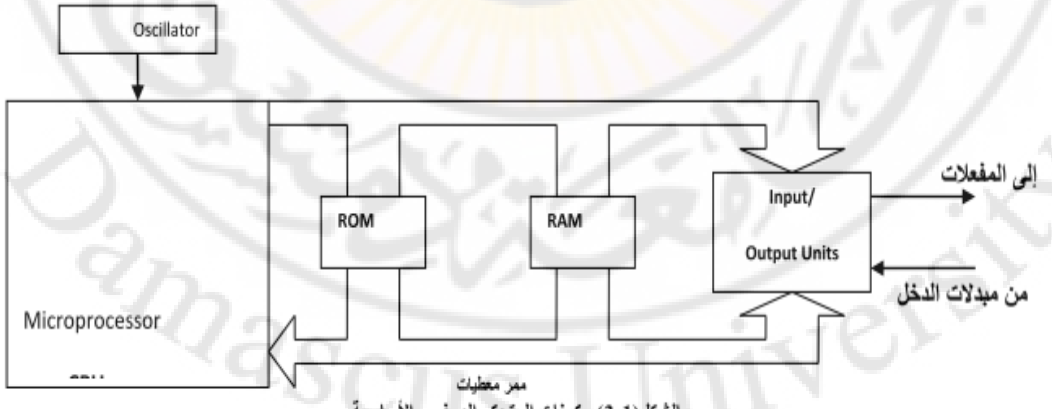
تفيد المقاطعات البرمجية في اختبار إجرائيات خدمة المقاطعة. فعلى سبيل المثال، يمكن استخدام INT 0 لتوليد مقاطعة برمجية من النوع 0 (القسمة على 0)، دون الحاجة إلى تنفيذ برنامج القسمة. كما يمكن استخدام التعليمة INT 2 لتنفيذ إجرائية المقاطعة المرتبطة بالدخل NMI دون ظهور إشارة مقاطعة خارجية.

المتحكمات الصغرية



إن إضافة ذاكرات إلى المعالج الصغري مع منافذ إدخال و إخراج و تجميعها معاً ضمن شريحة واحدة يعطينا حاسوباً مجمعاً في شريحة و لكن مع ظهور تقنية RISC و هي تقنية اختصار عدد التعليمات بحيث تصبح موجهة لغرض معين أصبح أسمه متحكماً صغرياً ، فالمتحكم الصغري هو أحد الأشكال الأساسية لنظام الحاسوبComputer System وهو الأقرب إلى الحاسوب الشخصي إذ أن بناء الحواسيب والمتحكمات الصغرية يتم من نفس العناصر الأساسية، والمتحكم الصغري هو عبارة عن عنصر منفذ لتعليمات ذات أهداف محددة. بما أن المتحكم الصغري عبارة عن نظام حاسوبي صغير ومتكامل فهو يحوي ذاكرة EEPROM مدمجة للبرنامج كذلك يحوي ذاكرة RAM مدمجة، هذا بالطبع فضلا" عن وجود وجدة المعالجة المركزيةCPU ووحدة جلب التعليمات، كذلك يوفر هذا المتحكم الصغري مجموعة من الأطرافPins التي تعمل كدخلinput أو خرجOutput، كما يمتاز المتحكم الصغري بوجود هزاز داخلي يستطيع العمل ضمن مجال ترددي محدد مثلاً من 0 إلى 24MHZ، كما ويمكن توسيع ذاكرته بوساطة ذاكرة خارجية تصل إلى 64Kbyte، يبين الشكل (1-3) مخططاً صندوقياً لنظام تحكم يعتمد على المعالج الصغري ونلاحظ أن هذا النظام يحتوي فقط المحيطيات(Peripherals) و العناصر الأساسية (RAM,ROM,I/O Ports) و يشكل ما يسمى متحكماً صغرياً .

نقوم في نظام التحكم السابق بتخزين برنامج التحكم كتعليمات في الذاكرة (ROM) ليقوم المعالج وبالاعتماد على هذا البرنامج بعد فحص حالة المداخل (القادمة من الحساسات والمفاتيح) بتوليد أوامره التي يتم إرسالها إلى وحدة الخرج ومنها إلى المفعلات (عناصر التنفيذ) والتي يمكن أن تكون لمبات إشارة أو عناصر مفتاحية تقوم بتشغيل محرك أو فرن .



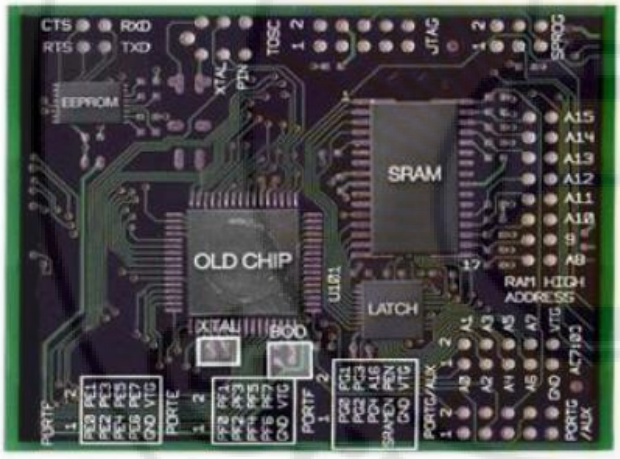
الشكل(1-3) مكونات المتحكم الصغري الأساسية

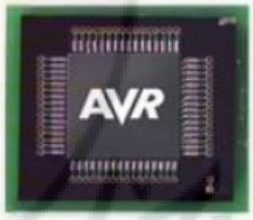
المتحكم المصغر

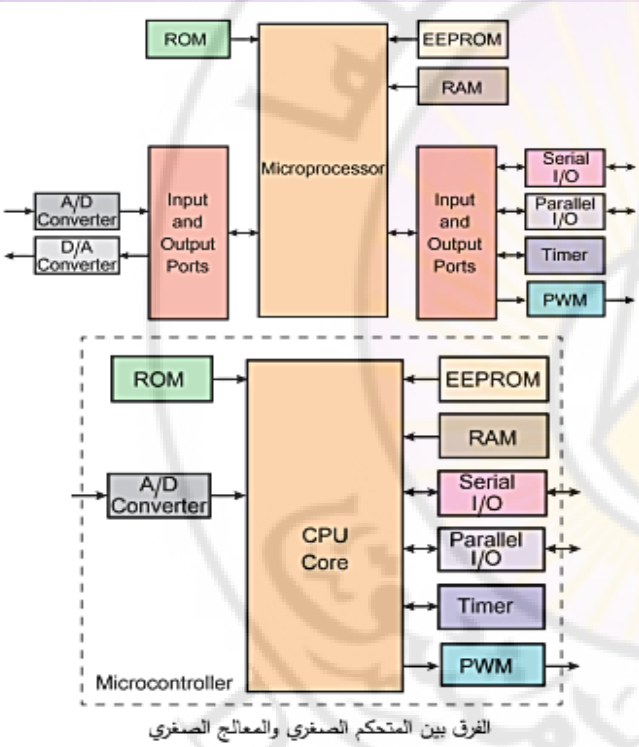
يعرف المتحكم المصغر بأنه "حاسب موضوع على شريحة "A Computer on a chip" وهو فعلاً دارة متكاملة Integrated Circuit تشكل حاسباً رقمياً بسيطاً كاملاً يحوي معظم العناصر الأساسية التي توجد في أي حاسب ويمكن أن يبدأ بتنفيذ التعليمات البرمجية المعدّة له دون الاعتماد إلا على عدد قليل من العناصر الخارجية المساندة وأحياناً دون أي عنصر آخر سواه على الإطلاق.

تتميز المتحكمات – باعتبارها حواسب صغيرة جداً – عادة ببنيتها البسيطة ومواردها التي تبدو محدودة مقارنة مع الحواسب الرقمية الشائعة لكنها غالباً ما تكون أكثر من كافية لتأدية المهمة الموكلة إليها، بالإضافة إلى أن بنيتها المبسطة تعود بالعديد من الفوائد: أهمها أن بساطة المتحكمات تمكن من وضعها بالكامل على شريحة واحدة وبتكلفة منخفضة جداً وهذان العاملان (صغر الحجم والتكلفة) من العوامل التي تجعلها مثالية لإدراجها في العديد من الأجهزة من حولنا: كالأجهزة الكهربائية المنزلية وأجهزة القياسات الرقمية وأنظمة التحكم والربط مع الحواسيب وفي المركبات وبعض ألعاب الأطفال وبعض أجزاء الحواسب الشخصية و محيطياته كالفأرة Mouse والعديد من التطبيقات المتنوعة.

مالفرق بين المعالج الصغري Microprocessor والمتحكم الصغري Microcontroller؟؟؟







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.

المعالج الصغري	المتحكم الصغري
لا يمكن تركيبه في الدارات الالكترونية بدون وحدات محيطية (I/O ، عداد ، مؤقت .....)	يمكن تركيبه في الدارات الالكترونية بشكل مباشر أي لايحتاج وحدات محيطية
يتصل مع الوسط الخارجي من خلال الوحدت المحيطية الخاصة به	تم تضمين جميع هذه الواحدات في شريحة واحدة وبحجم شريحة المعالج المصغر
مكلف	يستخدم في التطبيقات التي تعتبر فيها التكلفة والطاقة والمساحة بالغة في الأهمية
قوة معالجة عالية	قوة معالجة منخفضة
استهلاك طاقة كبير	استهلاك منخفض للطاقة
تركز مجموعة التعليمات على العمليات كثيفة المعالجة	عمليات على مستوى الـ bit

الخلاصة :تختلف المتحكمات (Microcontrollers) عن المعالجات (Microprocessors) بأنها تحتوي على ذاكرات ROM , RAM , و ربما EPROM و أنظمة محيطية أخرى كنوافذ دخل خرج ونوافذ اتصالات (Peripherals System Communication Ports) و محولات تشابهية رقمية ADCs . يمكننا بالاعتماد على المقارنة السابقة أن نكتب المعادلة الآتية:

Microcontrollers = Microprocessors +Memories+ Some Peripherals

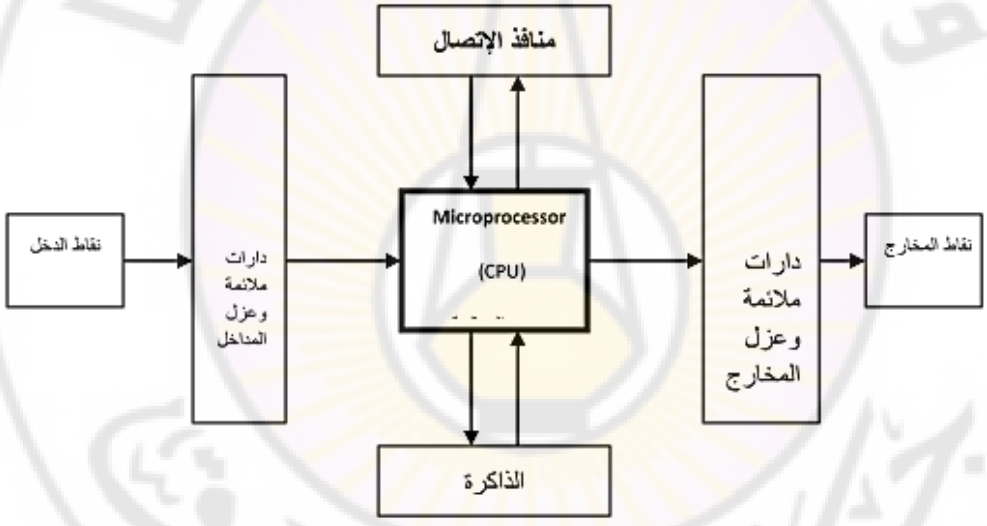
يتوفر الآن أنواع عديدة من المتحكمات نذكر منها متحكمات Atmel ومتحكمات Microchip ومتحكمات Siemens ومتحكمات Matra ومتحكمات أخرى عديدة، تتميز هذه المتحكمات فيما بينها بنوع الخدمات المدمجة ضمنها والوحدات المحيطية التي تجعل من بعض المتحكمات أقوى فعالية من الأخرى. ولما كان التحكم الصناعي يهدف إلى تخفيض عدد الشرائح وتقليل استهلاك الطاقة لزيادة وثوقية النظام فقد كانت هناك رغبة بمزيد من التكامل، لذا طور صانعوا الشرائح معالجات صغرية CPU وتم تزويدها بالذاكرات ودارات الملائمة وكانت هذه الشرائح هي بداية المتحكمات الصغرية Microcontrollers . المتحكم هو في الواقع جهاز حاسوب صغير مصمم خصيصا ليقوم بأعمال معينة. ويستخدم الذاكرة لتخزين الأوامر المبرمجة و القيام بتنفيذ هذه الأوامر مثل التشغيل والإطفاء التوقيت, العد، الحساب وغير ذلك من العمليات.

المتحكمات المنطقية القابلة للبرمجة PLCS

إن المتحكمات المنطقية القابلة للبرمجة PLCs في الحقيقة هي عبارة عن متحكمات في أساس تكوينها إلا أنه تم تدعيم هذه المتحكمات بالعديد من التطويرات الصلبة والبرمجية وفي المخطط الصندوقي المبين في الشكل(1-4) نبين الوحدات الأساسية التي يتكون منها المتحكم المنطقي PLC. يمكننا بالاعتماد على مخطط الشكل(1-4)أن نكتب المساواة التالية:

PLC = Microcontroller + Supporting Units

أي أن إضافة وحدات الدعم إلى المتحكم (Microcontroller) ستجعل منه جهاز PLC ومن أهم الوحدات الداعمة هي وحدات عزل وملائمة الدخل خرج و التي تؤمن العزل المناسب للجهاز عن الوسط المحيط ووحدة التغذية التي تكون غالباً من وحدات التغذية و التي تتميز بدقتها الكبيرة ومجال عملها على عدد كبير من الجهود فضلاً عن العزل الغلفاني الذي تؤمنه للمتحكم PLC , ولا يقتصر الدعم المقدم على الوحدات الصلبة وإنما يتعداه إلى الدعم البرمجي حيث تقدم هذه الأجهزة لغات برمجة سهلة التعلم والاستخدام أشهرها لغة المنطق السلمي.



الشكل(1-4) مكونات المتحكم المنطقي القابل للبرمجة PLC

مراحل تصميم الأجهزة المحتوية على معالج أو متحكم صغري:

يمكن تقسيم مراحل تصميم نظام أو جهاز يحتوي على معالج صغري إلى خمس وهي:

1. **مرحلة التصميم الأولي:** وفي هذه المرحلة يتم تحديد أو إنجاز أربع مهام وهي:
   - تحديد وظائف النظام العامة و مبدأ عمله
   - فصل الأعمال البرمجية Software عن الأعمال بالأجهزة Hradware
   - رسم المخطط الأولي (الصندوقي) للجهاز دون الخوض في التفاصيل
   - تحديد أنواع المعطيات ورسم مخطط صندوقي غير تفصيلي للبرنامج

   المهمتان الثالثة و الرابعة يمكن إنجازهما على التوازي.

2. **مرحلة التصميم الأساسي (التقني) للجهاز و برمجته:** وفي هذه المرحلة يتم تحديد أو إنجاز ثلاث مهام وهي:
   - وضع المخطط التدفقي للبرنامج (الخوارزمية)
   - كتابة البرنامج بلغة برمجة مناسبة أو بالشكل الأنسب
   - وضع المخطط الالكتروني التفصيلي للجهاز

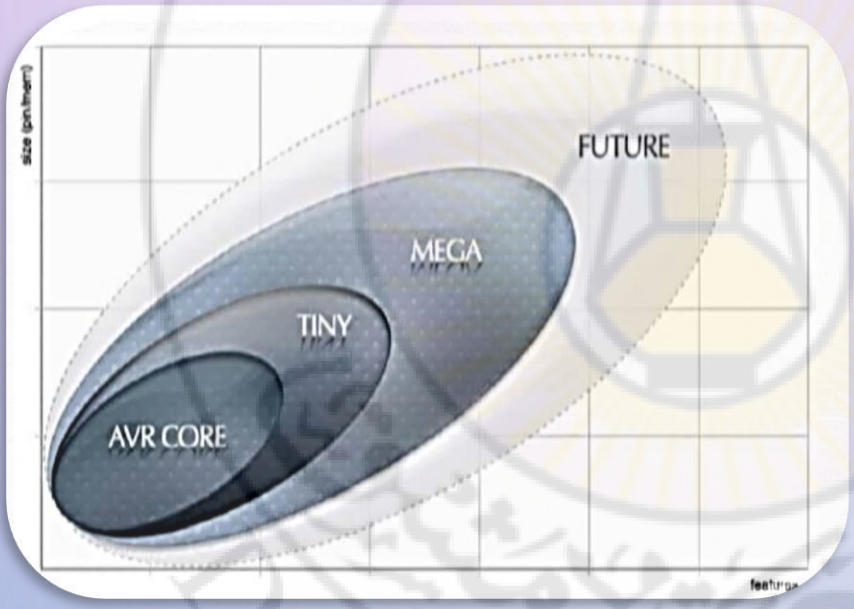
   و نلاحظ أن المهمتان الأولى والثانية تتمان على التوازي مع المهمة الثالثة.

3. **مرحلة التطبيق Implementation:** وفي هذه المرحلة يتم تحديد أو إنجاز ثلاث مهام (أو مهمتين في المتحكمات) وهي:
   - ترجمة البرنامج إلى لغة الآلة(إذا لم يكتب بها)
   - تخزين البرنامج في ذاكرة ثابتة ROM (أو EPROM).
   - إنجاز مونتاج الجهاز أي لحام القطع و تثبيتها في أماكنها

   و نلاحظ أن المهمتان الأولى والثانية تتمان على التوازي مع المهمة الثالثة.

4. مرحلة الاختبار والتجريب: أي تشغيل الجهاز وتجريبه
5. مرحلة طباعة دليل المستخدم User guide: أي تصميمه و طباعته

متحكمات AVR



عوائل متحكمات AVR:

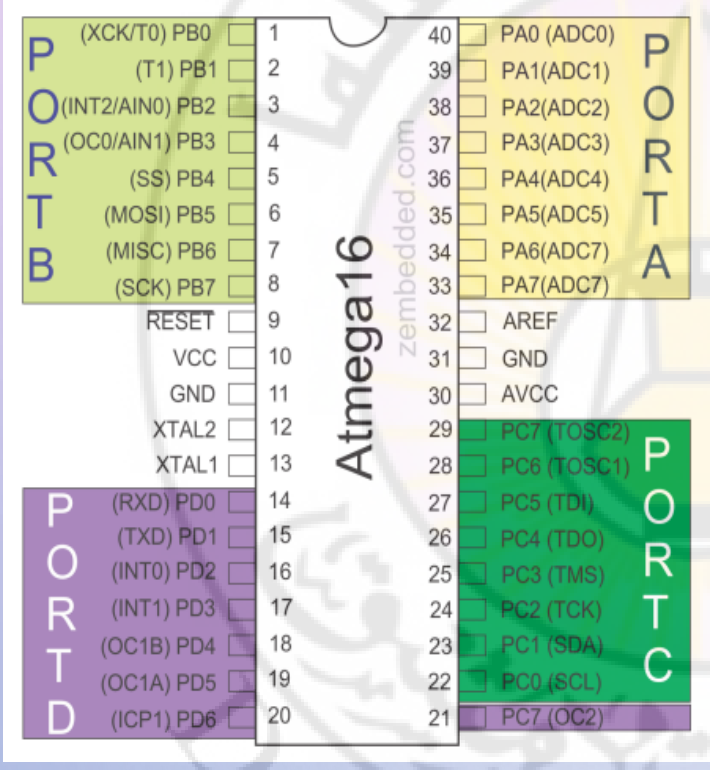
1. سلسلة Tiny: صغيرة الحجم والذاكرة ، للتطبيقات الصغيرة .
2. سلسلة Atmega: حجم ذاكرة متوسط للتطبيقات المتوسطة والمعقدة.
3. سلسلة Xtmega: تستخدم للتطبيقات المعقدة التي تحتاج سرعة عالية وحجم ذاكرة كبير

أهم مميزاتها

- ✓ من منتجات شركة ATMELالأمريكية
- ✓ تم تطويرها في مختبرات الشركة الموجودة في النرويج 1990
- ✓ تعتبر من أكثر المتحكمات المصغرة انتشاراً
- ✓ تفوّقت بشكلٍ كبير على العديد من نظيراتها من متحكمات bit-8
- ✓ تستخدم البنية Harvard + RISC التي تتميز بالأداء العالي وبالطاقة المنخفضة
- ✓ احتوت قائمة التعليمات في متحكمات AVRعلى 132 تعليمة
- ✓ يُنفذ معظمها خلال دورة آلة واحدة (1-cycle)
- ✓ حتى 20 مليون تعليمة في الثانية الواحدة
- ✓ إمكانية برمجة المتحكم دون فصلة عن النظام (In-system Programming)

البنية الداخلية لعائلة المتحكمات AVR

متحكم AVR ATMEGA16



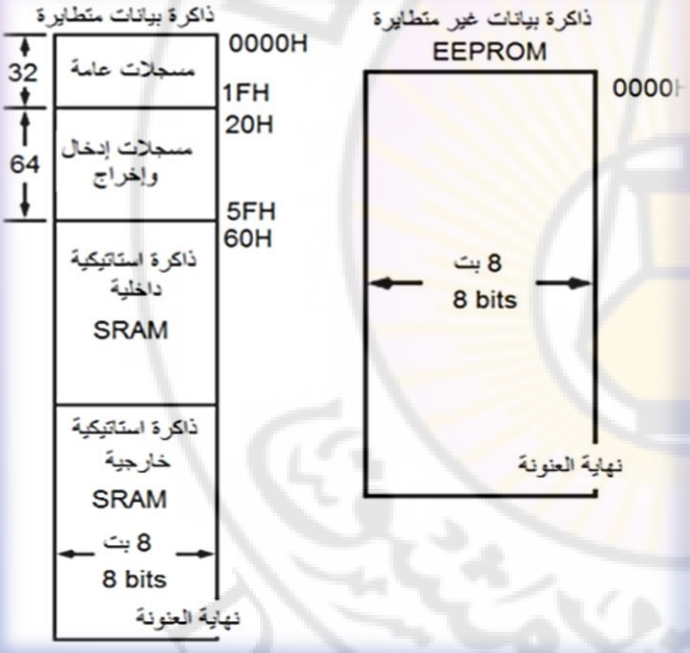
أقطاب المتحكم ATMEGA 16:

- أقطاب التغذية الرقمية (VCC, GND).
- أقطاب التغذية التشابهية (AVCC, AGND).
- أقطاب الدخل والخرج (GPIOs).
- أقطاب البرمجة (SPI).
- أقطاب المقاطعات الخارجية (INTx).
- أقطاب مصادر التوقيت (XTAL1, XTAL2).
- أقطاب المؤقتات/العدادات (T0, T1, T2).
- أقطاب المبدلات التشابهية الرقمية (ADC).
- أقطاب النوافذ التسلسلية (UART,SPI,...)
- أقطاب المقارن التشابهي (AIN0,1).
- أقطاب إشارات PWM (OCx).
- أقطاب نافذة المراقبة JTAG (Debug).

البنية العامة لمتحكمات AVR

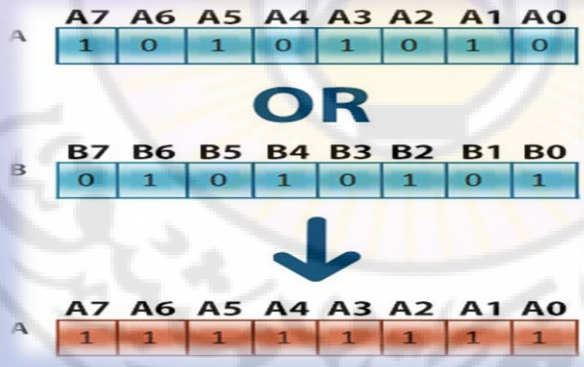
- تعتمد هذه المتحكمات على معمارية هارفارد التي تستخدم ذاكرة خاصة للبرنامج (لكي تخزن التعليمات) من النوع FLASH ROM التي لاتمحى بانقطاع التغذية الكهربائية . وذاكرة أخرى من النوع SRAM التي تتلاشى بانقطاع التغذية الكهربائية ، ومن ضمن ذاكرة SRAM يوجد مسجلات الأغراض العامة ومسجلات التحكم ، ويوجد ذاكرة ثالثة EEPROM يمكن استخدامها من قبل المبرمج لتخزين المتحولات الهامة التي يجب ان تبقى مخزنة بعد انقطاع التغذية الكهربائية ...

أقسام ذاكرة البيانات في متحكمات AVR:



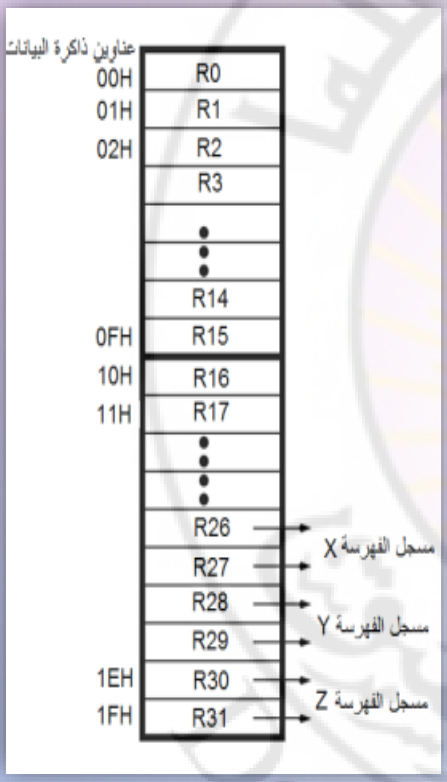
- تعتمد معمارية RISC ، حيث تركز على مجموعة تعليمات بسيطة ، وبالتالي تكون أسرع في التنفيذ (لكن الكود البرمجي يكون أطول) ، حيث يتم تنفيذ معظم التعليمات في نبضة ساعة واحدة – حيث ان المتحكم الذي يعمل بتردد 1MHZ يستطيع تنفيذ مليون تعليمة تقريبا في الثانية الواحدة –

- وتعتمد على المسجلات في تنفيذ التعليمات حيث تكون معاملات التعليمة (الدخل) مخزن في المسجلات وأيضا يتم تخزين النتيجة (خرجA التعليمة) في مسجل آخر ، حيث يحتوي 32 مسجل أغراض عامة كما في المثال التالي :



المسجلات العامة في متحكمات AVR:

يحوي معالج AVR مجموعة من المسجلات عامة الأغراض تدعى بملف المسجلات REGISTER FILE. يحوي هذا الملف 32 مسجل من R0 حتى R31 . يمكن إجراء جميع العمليات الحسابية والمنطقية على هذه المسجلات. تشكل هذه المسجلات جزءاً من ذاكرة المعطيات حيث تحتل مجال العناوين من 0X00 حتى 0X1F وذلك لتسهيل الوصول إليها. لكنها لا تعتبر ذاكرة RAM من الناحية الفيزيائية .

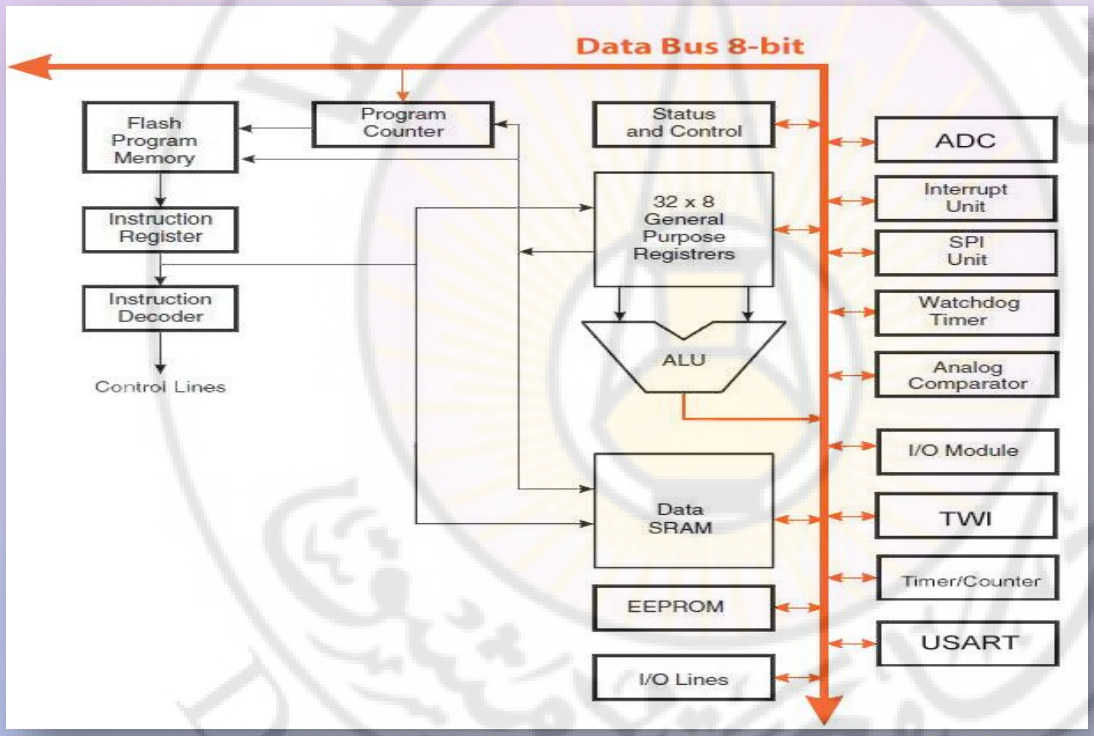


المؤشرات عامة الأغراض

يمكن استخدام المسجلات الستة الأخيرة من ملف المسجلات كثلاث مؤشرات إلى الذاكرة . يشكل كل زوج من المسجلات 8 بت مؤشر بطول 16 بت كما في الشكل التالي :

المسجلات الخاصة SFR

يتم التحكم بطرفيات المتحكم AVR من خلال مجموعة من مسجلات التحكم الخاصة . تشكل هذه المسجلات فضاء عنونة خاص يسمى فضاء الدخل/الخرج ويمكن الوصول إليه بتعليمات IN و OUT . يمتد مجال عنونة الدخل /الخرج من العنوان 0X00 إلى 0X3F . يمكن الوصول إلى مسجلات التحكم أيضاً من خلال فضاء عنونة ذاكرة المعطيات حيث يحتل المجال من 0X20 إلى 0X5F وتستخدم في هذه الحالة التعليمات LD و ST . أما ذاكرة RAM الحقيقة فتبدأ من العنوان 0X60 .



مخطط صندوقي
للمكونات الداخلية
للمتحكمات AVR

❑ أهم مكونات AVR

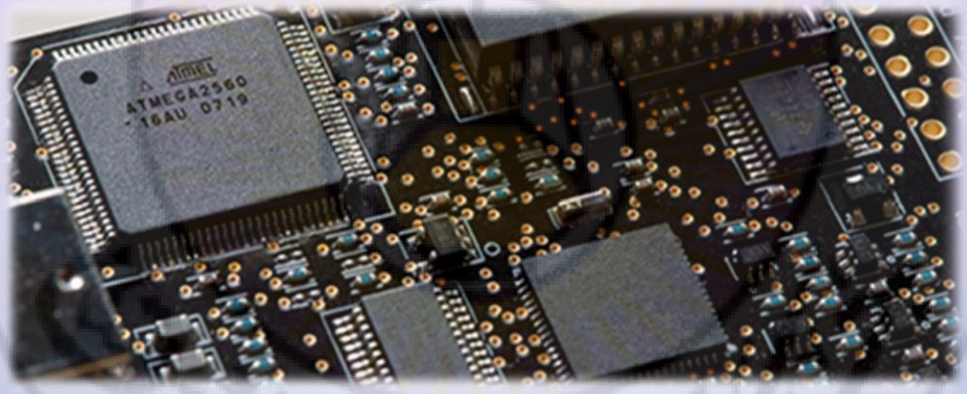
1. وحدة الحساب والمنطق.
2. عداد البرنامج PC : ليشير إلى عنوان الأمر عدد بتاته يكون حسب كمية الذاكرة للمتحكم.
3. مسجل الأوامر IR: يحوي شيفرة الأمر الذي يتم تنفيذه الآن
4. محلل شيفرة الأوامر INSTRUCTION DECODER.
5. مسجل الحالة SR ومنه:

1. علم الحمل أو الاستلاف CARRY FLAG C: البت رقم 0 يساوي صفر في مسجل الحالة وواحد عند حدوث حمل
2. علم الصفر ZERO FLAG Z: البت رقم 1 في مسجل الحالة يساوي واحد إذا كانت النتيجة صفر وصفر فيما عدا ذلك
3. علم السالبية NEGATIVE FLAG N: البت رقم 2 إذا كانت النتيجة موجبة (إذا كان آخر بت صفر) والنتيجة سالبة إذا كان آخر بت يساوي واحد
4. علم الفيضان OVERFLOW FLAG V: البت رقم 3 يبين الفيضان في قيم الجمع والطرح
5. علم الإشارة SIGN FLAG S: البت رقم 4 هو عملية XOR يبين الإشارة الصحيحة للنتيجة إذا حدث فيضان وتسبب في تدميره.

6. منافذ الدخل والخرج I/O PORTS يختلف عددها حسب المتحكم وتكون بعرض 8 BIT.
7. هزاز داخلي INTERNAL OSCILLATORS : ذو بنية مقاومة ومكثف RC .
8. مؤقت / عداد TIMER/ COUNTER : يمكن ان يكوت بعرض 8 او 16 BIT.
9. مؤقت مراقبة WATCHDOG TIMER: يعيد إقلاع المتحكم إذا تعطل تنفيذ التعليمات عند نقطة واحدة لفترة زمنية معينة ويوقف عملية تنفيذ البرنامج ويحمي البرنامج من أي أخطاء،
10. محول تشابهي رقمي ADC : يمكن أن يكون بدقة (10/ 12) BIT
11. واجهات اتصال متنوعة 12C – SPI – USART – TWI.
12. مقارن تشابهي ANALOG COMPARATOR
13. ساعة الزمن الحقيقي RTC CLOCK

كما انها تمتلك عدة أنماط لتوفير الطاقة ووجود حماية من انخفاض جهد التغذية التي تضع المتحكم في وضع إعادة الإقلاع إلى أن يعود جهد التغذية إلى المستوى الطبيعي

النشرة الفنية للمتحكم الصغري DATA SHEET




(XCK/T0) PB0
PA0 (ADC0)
Electrical Characteristics????
(SS) PB4
(MOSI) PB5
(MISO) PB6
PA4 (ADC4)
PA5 (ADC5)
PA6 (ADC6)
Absolute Maximum Ratings????
GND
XTAL2
XTAL1
AVCC
PC7 (TOSC2)
PC6 (TOSC1)
Maximum Frequency vs. VCC????
(INT1) PD3
(OC1B) PD4
(OC1A) PD5
(ICP1) PD6
PC2 (TCK)
PC1 (SDA)
PC0 (SCL)
PD7 (OC2)
15

Features

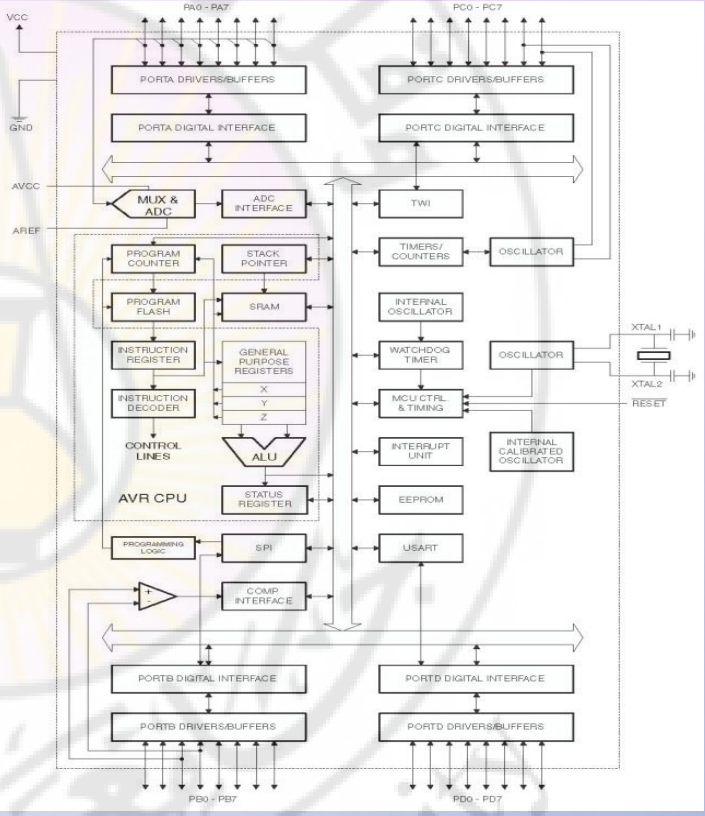
- High-performance, Low-power Atmel® AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 16 Kbytes of In-System Self-programmable Flash program memory
  - 512 Bytes EEPROM
  - 1 Kbyte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C[1]
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega16L
  - 4.5V - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16
- Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
  - Power-down Mode: < 1 µA

ATMEL

8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

ATmega16
ATmega16L

Rev. 2466T–AVR–07/10



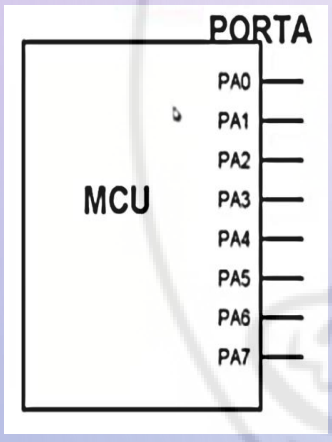
ATMEL

برمجة بوابات الدخل والخرج في متحكمات AVR

«GPIO»

برمجة المنافذ I/O في عائلة AVR (PORTs)

- يقصد بها البنات التي تتبادل المعطيات مع العالم الخارجي PORT = 8 PINs وذلك في متحكمات معالج core 8 bit ، وتعمل هذه الأقطاب كدخل أو خرج .
- والمتحكم المصغر الواحد يحتوي أكثر من PORT وتسمياتها وفي شركة Atmel (PORT A – PORT B – PORT C – PORT D)
والبنات الفرعية تسمى (... ... ... PA_1 – $P A_0$)



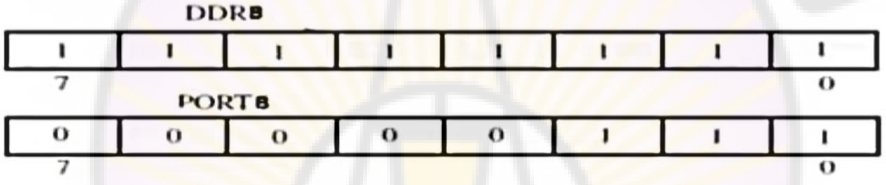
- وكل بورد يمتلك ثلاث مسجلات :
  1. مسجل تحكم DDRX أي (DDRA – DDRB.....)
  يمكن تحديد المنفذ كدخل أو خرج (111 خرج – 000دخل) وكل خانة من خانات هذا المسجل تؤثر على قطب من أقطاب البوردات (PIN).
  2. مسجل كتابة المعطيات PORTX: يستخدم لكتابة المعطيات.
  3. مسجل القراءة PINX : لقراءة المعطيات من الخارج فقط .

ملاحظة: تعامل المسجلات وكأنها متحولات بطول BIT8

يحتوي ATMEGA على 40 قطب ويتم تغذيته ضمن المجال (2.7 – 5) فولت

مثال ١: برمج PORTB لتعمل أقطابه مخارج واجعل حالة المخارج 0B00000111 .

الحل:




```
DDRB=0B11111111;
PORTB=0B00000111;
```

0B ترمز إلى نظام العد الثنائي

الترقيم يبدأ من اليمين لليسار

مثال ٢: برمج القطب PB3 في المنفذ PORTB كمخرج وباقي الأقطاب مداخل

الحل:

```
DDRB=0B00001000;
```

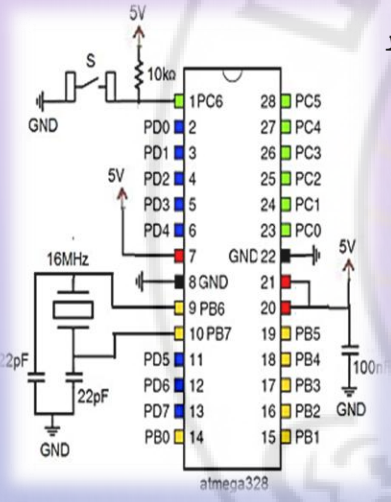
خطوات تصميم الأنظمة المدمجة باستخدام متحكم

ATMEGA 16 – AVR

حتى يعمل المتحكم بشكل سليم يحتاج إلى ثلاث توصيلات أساسية :

1. مصدر الطاقة (VCC- GND) POWER: أيتوصيل أرضي ومنبع للجهد V5 غالبا
2. توصيلة (MASTER CLEAR) MCLR : أي RESET وهي أساسية لعمل المتحكم تكون موصولة مع مقاومة وزر عند الضغط عليه يقوم بتصفير عمل المتحكم
3. النابض الذي يعطي نبضات 0 و 1
4. TIMER دوائر تحتوي مؤقتات ودوائر ليست بحاجة لها

ومن دونها لايعمل المتحكم



الشكل المجاور يبين تجهيز المتحكم من حيث توصيل الطاقة

الكهربائية وتوصيل مصدر النبضات في حالة استخدام مصدر خارجي.

خط الطاقة (التغذيةVCC) هو خط التغذية العام للمتحكم وكذلك الخط الأرضيGND

أما خط التغذية AVCC فهو خاص بالمحول التماثلي الرقمي والذي يجب توصيله إلى مكثف لضمان تقليل الضجيج على المحول ADC ، في حال توصيل نبضات تزامن خارجية يتم استخدام بللورة بالتردد المطلوب مع مكثفين .

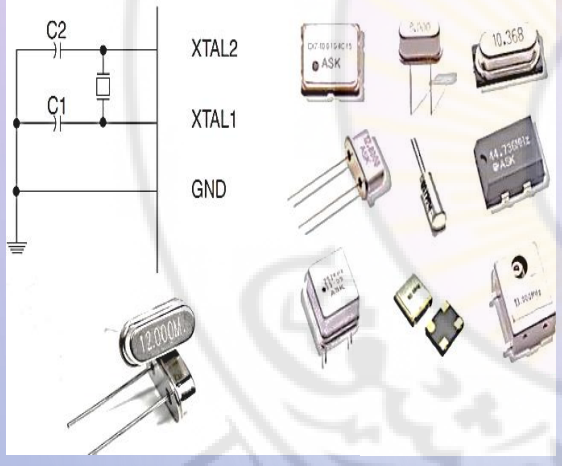
الطرف 1 يمكن استخدامه كطرف RESET للمتحكم عن طريق توصيل الطرف مع 5V من خلال مقاومة 10 كيلو اوم وعند الضغط على مفتاح S يحدث RESET .وبهذا تصبح الشريحة جاهزة للتوصيل مع دوائر خارجية

ماهية المتحكمات الصغرية

مصادر التغذية أو توزيع مصادر التوقيت

للعائلة AVR

لتنظيم الوقت في برامج المتحكمات نحتاج إلى عدة مؤقتات (ساعات) (مؤقت للمعالج، لل RAM، للFLASH، للADC ، وللمداخل والمخارج .....إلخ) وغيرها من العناصر الأخرى التي نستخدمها في الدارة.
تسمى اوسوليتر الذي يقوم بعمل النبضات والتي تسبه نبضات القلب لتنظيم وتنفيذ عمل المتحكم وإعطاء قيم دقيقة ، وقد لايستخدم في كل الدوائر ، ومنها مايكون داخلي ومنها خارجي وهو الأفضل ، وسندرس مصادر التوقيت في المتحكم AVR:



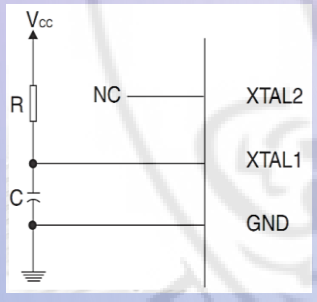
1. المهتز الكريستالي CRYSTAL OSCILLATOR :له أشكال عديدة نلاحظ وجود قيمة المهتز عليه ويتم وصله حسب الدارة في القطبين XTAL1 (INPUT) و XTAL2 (OUTPUT) ، هذا المهتز عبارة عن مادة كريستالية طبيعية لديها القدرة على أن تعطي نبضات (اهتزازات) عندما نعطيها جهد فولت (توتر)، وأيضا العكس تعطي توتر(فولت) عندما نقوم بالضغط عليها ، وهي دائرة خارجية ذات دقة أعلى وافضل من غيرها بكثير، لأن المادرة المكونة منها دقيقة جدا ، ولها سرعات مختلفة (8-16- .....).

2. المهتز الداخلي RC : وهي دائرة داخل المتحكم تقوم بإعطاء نبضات ولكن بدقة اقل من الكريستالي الخارجي ، فمثلا عند درجة حرارة 25 وجهد V5 يعطي المهتز الداخلي قيم.

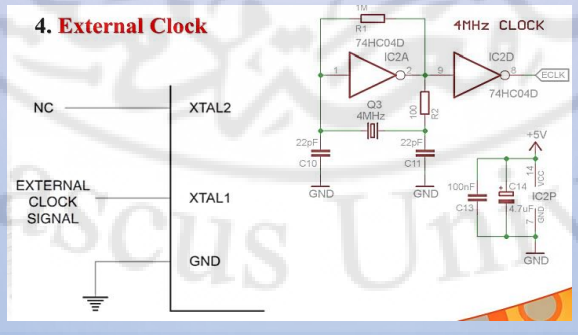
1.0MHz, 2.0MHz,
4.0MHz, 8.0MHz

at 5V and 25°C > ±3%

2. المهتز RTC (EXTERNAL RC): وهو النوع الأكثر دقة REAL TIME CLOCK ويوصل على القطبين TOSC1 و TOSC2



3. EXTERNAL CLOCK

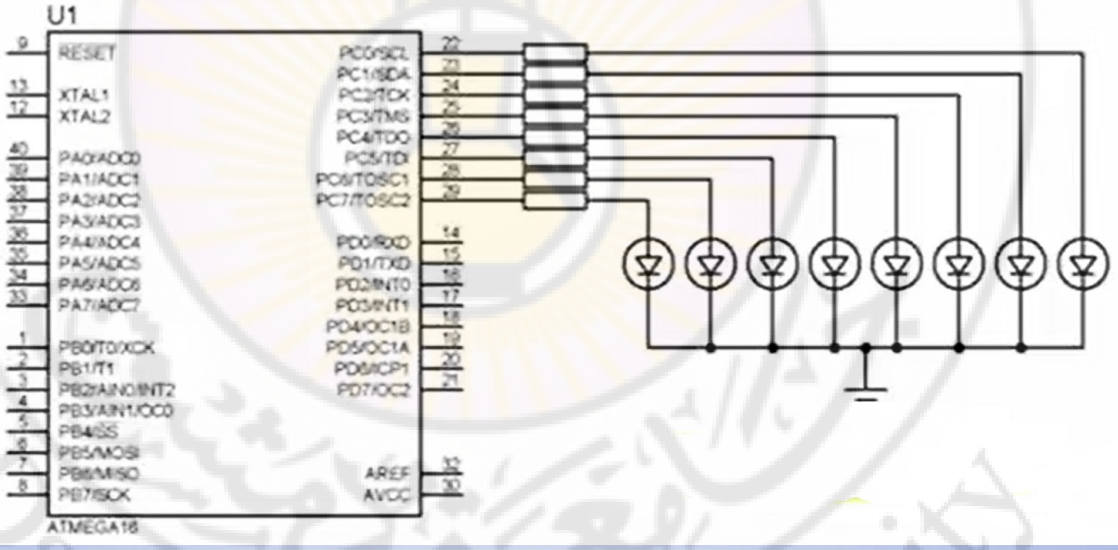


التجربة الثانية
FLASHER لأكثر من ليد ضوئي

تطبيق (الأضواء المتحركة)

برمج المتحكم ليتم أضاءة LEDs بخمسة طرق مختلفة وفق اللدارة التالي:

حسب الشكل يتم تعريف بنات ال PORTC جميعها على أنها خرج .. أي PORTC خرج

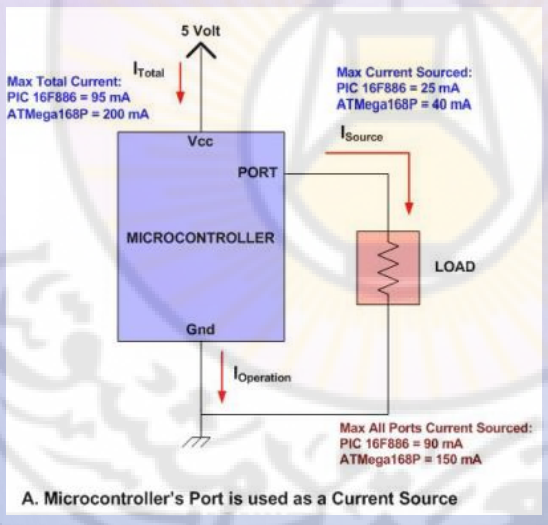


الكود البرمجي :

```
#include <mega16.h>
#include <delay.h>
void main(){
DDRC=0B11111111;
while (1)
    {
    PORTC=0B00011000;
    delay_ms(200);
    PORTC=0B00100100;
    delay_ms(200);
    PORTC=0B01000010;
    delay_ms(200);
    PORTC=0B10000001;
    delay_ms(200);
    PORTC=0B01000010;
    delay_ms(200);
    PORTC=0B00100100;
    delay_ms(200);
    }
}
```

1. هناك طريقتان لوصل الأحمال مع مخارج المتحكم :

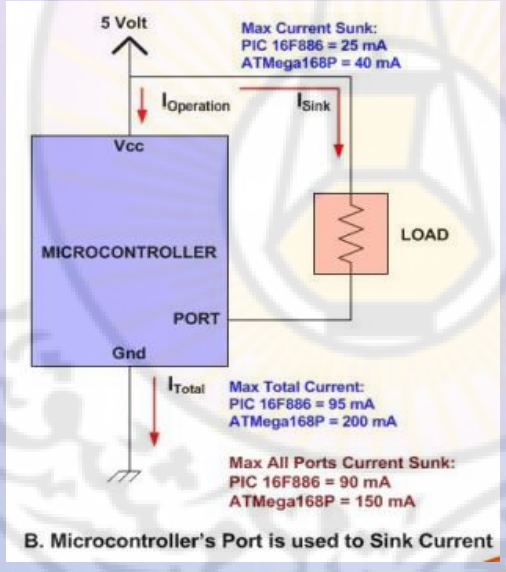
الـPORT أو PIN أي القطب يعمل كمنبع للتيار أي أحمال القطب (ليد مثلا) تكون موصولة من طرف على القطب (PORT) والطرف الآخر على الأرضي ، نلاحظ أن القطب سيصبح منبع للتيار



A. Microcontroller's Port is used as a Current Source

2. الـPORT أو PIN أي القطب يعمل كمصرف للتيار أي أحمال القطب موصولة من طرف على القطب (PORT) والطرف الآخر على التيار (الموجب) ، أي أن التيار يتحرك من الطرف الموجب ليدخل إلى الحمل .

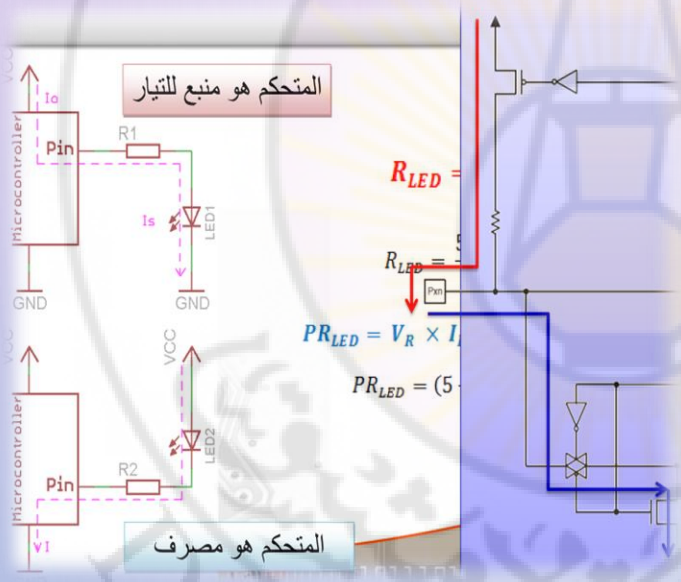
ومتحكمات الAVR تدعم طريقتي التوصيل



B. Microcontroller's Port is used to Sink Current

أهم الاعتبارات التي يجب أن تؤخذ بعين الاعتبار عند ربط أقطاب المتحكم إلى الأحمال هي :

1. التيار الأعظمي المستهلك من قطب المتحكم (Vcc to Gnd) الذي يمكن سحبه أو تصريفه عن طريق المتحكم بشكل كلي هو 200 ميلي امبير وفق مواصفات عائلة المتحكمات AVR.
2. قيمة التيار الممكن سحبه أو تصريفه لقطب خرج من أقطاب المتحكم تتراوح عادة من 20 ~ 40 mA حسب المواصفات الكهربائية للمتحكم المصغر AVR



والأحمال ذات الاستطاعات العالية تحتاج وسائل إضافية سنتعرف عليها لاحقا

تداول البيانات داخل متحكمات AVR

أنواع البيانات في الأنظمة المدمجة:

البيانات الرقمية : تعتبر اكثر الأنواع استخداما في عالم الأنظمة المدمجة فيه تعبر عن قيم المتغيرات مثل قراءة الحساسات او قيم المخارج مثل سرع المحركات

نوع البيانات	استهلاك الذاكرة (عدد Bytes)	أقصى قيمة يمكن وضعها داخل المتغير
int	2-bytes (16 bit)	-32,768 → +32,767
unsigned int	2-bytes (16 bit)	65,535
float	4-bytes (32 bit)	1.2E-38 to 3.4E+38
double	8-bytes (32 bit)	2.3E-308 to 1.7E+308
long	4-bytes (32 bit)	-2,147,483,648 → +2,147,483,647
unsigned long	4-bytes (32 bit)	4,294,967,295
long long	8-bytes (64 bit)	$-9.223372037\times10^{18}$ + 9.223372037×10^{18}
unsigned long long	8-bytes (64 bit)	1.844674407×10^{19}

ملاحظة: نوعي البيانات float و double دائماً يكونا من نوع Signed ولا يمكن استخدام العلامة unsigned معهما

جدول البيانات التقليدية

أنواع البيانات المعيارية ANSI C - C99 وهي صورة محسنة لأنواع البيانات وتمكننا من اختيار قيمة المتغيرات بالدقة والطول المطلوب دون أي اهدار للذاكرة.

Signed	أقصى قيمة	Unsigned	أقصى قيمة
int8_t	-128 → +127	uint8_t	0 → 255
int16_t	-32,768 → +32,767	uint16_t	0 → 65,535
int32_t	-2,147,483,648 → +2,147,483,647	uint32_t	0 → 4,294,967,295
int64_t	$-9.223372037\times10^{18}$ + 9.223372037×10^{18}	uint64_t	1.844674407×10^{19}

جدول البيانات المعيارية

مجموعة التعليمات

يمتلك AVR مجموعة تعليمات غنية مؤلفة من 131 تعليمة. تقسم هذه التعليمات إلى أربع مجموعات :

التعليمات الحسابية ، تعليمات البتات ، تعليمات التفرع وتعليمات نقل البيانات.

تعليمات التحميل ونقل البيانات

تقوم هذه التعليمات بنقل البيانات بين المسجلات أو بين مسجل والذاكرة وبالعكس أو بين مسجل ومسجل خاص وبالعكس أو تحميل قيمة فورية في مسجل ما.

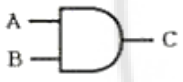
ملاحظات :

1. التعليمة LDI (وجميع التعليمات المنتهية بالحرف I) تحمل قيمة فورية في أحد المسجلات من R16 حتى R31 ولا يمكن استخدام المسجلات الأخرى لتحميل القيم الفورية.
2. تختلف تعليمات النقل من وإلى الذاكرة بنمط العنونة ، أي بطريقة حساب عنوان الحجرة المقصودة.
3. التعليمات OUT و IN تتعامل مع فضاء عنونة الدخل والخرج وتستخدم مع المسجلات الخاصة SFR.
4. التعليمات LPM و SPM تتعاملان مع ذاكرة البرنامج ROM .

العمليات المنطقية

OR	|
AND	&
NOT	~
XOR	^
Shift Left	<<
Shift Right	>>

AND



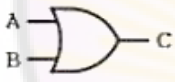
Inputs		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

أمثلة على AND

0 & 0 = 0

0 & 1 = 0
1 & 0 = 0
1 & 1 = 1

OR



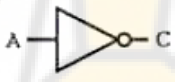
Inputs		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

أمثلة على OR

0 | 0 = 0

0 | 1 = 1
1 | 0 = 1
1 | 1 = 1

NOT

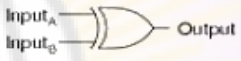


Input	Output
A	C
0	1
1	0

أمثلة على NOT

~ 0 = 1
~ 1 = 0

Exclusive-OR gate



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

أمثلة على XOR

0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0

عمليات الإزاحة SHIFT OPERATION:

وهي عملية تحريك البتات إلى اليمين أو اليسار داخل متغير أو مسجل أو أي قيمة رقمية . تعد هذه العمليات من أهم الأوامر التي تستخدم في برمجة المعالجات والمتحكمات الصغرية

الرقم الأصلي	0b00000101
إزاحة بمقدار 1 بت جهة اليسار <<	0b00001010
إزاحة بمقدار 2 بت جهة اليسار <<	0b00010100
إزاحة بمقدار 3 بت جهة اليسار <<	0b00101000

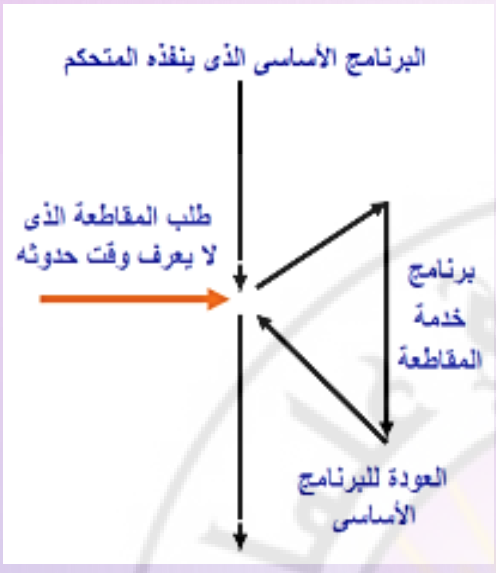
مثال: لنفرض أن لدينا 0b01100000 لنقم بتطبيق عمليات الإزاحة لجهة اليمين.

الرقم الأصلي	0b01100000
<< إزاحة بمقدار 1 بت جهة اليمين	0b00110000
<< إزاحة بمقدار 1 بت جهة اليمين	0b00011000
<< إزاحة بمقدار 1 بت جهة اليمين	0b00001100

المقاطعات في المتحكمات

INTERRUPTS

المقاطعة : هي عبارة عن حادثة تسبب توقف المتحكم عن تنفيذ البرنامج الرئيس ليذهب لتنفيذ برنامج خدمة المقاطعة (برنامج فرعي) وبعد الانتهاء منه يعود لتنفيذ البرنامج الرئيس ابتداءاً من التعليمة التي توقف عندها عند حدوث المقاطعة .



أي أنها إشارة إلى المعالج او المتحكم آتية من مصدر عتادي Hardware او برامجي Software توضح وجود حدث يستوجب الانتباه والخدمة الفورية من قبل المعالج ، حيث يقوم المتحكم بتعليق نشاطه الحالي ويذهب لخدمة المقاطعة وبعد الانتهاء يعود لاستئناف نشاطه من نفس المكان الذي توقف فيه ، الشكل التالي يبين رسم تخطيطي لمقاطعة المعالج او المتحكم.

<u>وبشكل رئيسي تقسم المقاطعات إلى نوعين:</u>

❑ مقاطعات خارجية :

هي استجابة لحوادث خارجية (من خارج الشريحة) مثل :

- مقاطعات الطلب الخارجي تغير مستوى منطقي على القطب – تغير مستوى جهد معين (INT0~INT7) .
- مقاطعة التصفير reset.

❑ مقاطعات داخلية :

هي استجابة لحوادث داخلية (داخل الشريحة) مثل :

- مقاطعات المؤقتات / العدادات طفحان للمؤقت – طفحان للعداد
- مقاطعات المسك للمؤقتات/عدادات ICP.
- مقاطعة اكتمال التحويل للـ ADC.
- مقاطعة اكتمال الإرسال للنافذة التسلسلية SPI (STC)
- مقاطعة النافذة التسلسلية USART (RX,TX,UDR)
- مقاطعة المقارن التشابهي (ANALOG COMP).
- مقاطعة النافذة التسلسلية TWI.

المفاهيم الأساسية في المقاطعات

1. مسجلات التحكم والحالة للمقاطعة Control and Status :
عن طريق خانات هذه المسجلات يتم تحديد حدث المقاطعة وتفعيلها أو حجبها .
2. برنامج خدمة المقاطعة Interrupt routine :
هو برنامج فرعي يتم تنفيذه عند حدوث مقاطعة .
3. شعاع المقاطعة (عنوان المقاطعة) Interrupt Vector :
شعاع المقاطعة هو العنوان الموجود فيها بداية برنامج خدمة المقاطعة وترتيبها يحدد أفضلية المقاطعة.
4. علم المقاطعة Interrupt Flag :
هو عبارة عن خانة BIT في مسجل أعلام المقاطعة تصبح قيمته واحد عند حدوث المقاطعة (يرفع العلم) وبعد الانتهاء من برنامج خدمة المقاطعة تصبح قيمتها صفر .

العمليات التي تجري عند حدوث مقاطعة :

وعند ورود إشارة المقاطعة تجري العمليات التالية :
1. ينتهي المتحكم من تنفيذ الأمر الحالي الذي يقوم بتنفيذه.
2. يتم رفع علم الخاص بالمقاطعة (تصبح قيمته واحد). ، لكي يقبل المتحكم عمل المقاطعة
3. يتم وضع عنوان بداية خدمة المقاطعة في مسجل عداد البرنامج (PC) ليقوم بتنفيذ برنامج المقاطعة . (يقفز المعالج إلى مكان محدد في متجه المقاطعة Interrupt vector ليعرف منه عنوان برنامج خدمة المقاطعة ISR).
4. يتم تنفيذ برنامج خدمة المقاطعة (Interrupt Routine).
5. عند الانتهاء من برنامج خدمة المقاطعة الذي يكون آخر أمر فيه هو RETI ، يتم تنزيل العلم الخاص بالمقاطعة (تصبح قيمته صفر).
6. يرجع المتحكم بمتابعة تنفيذ تعليمات البرنامج الرئيس ابتداءاً من التعليمة التي توقف عندهاعندما وردت إشارة المقاطعة .

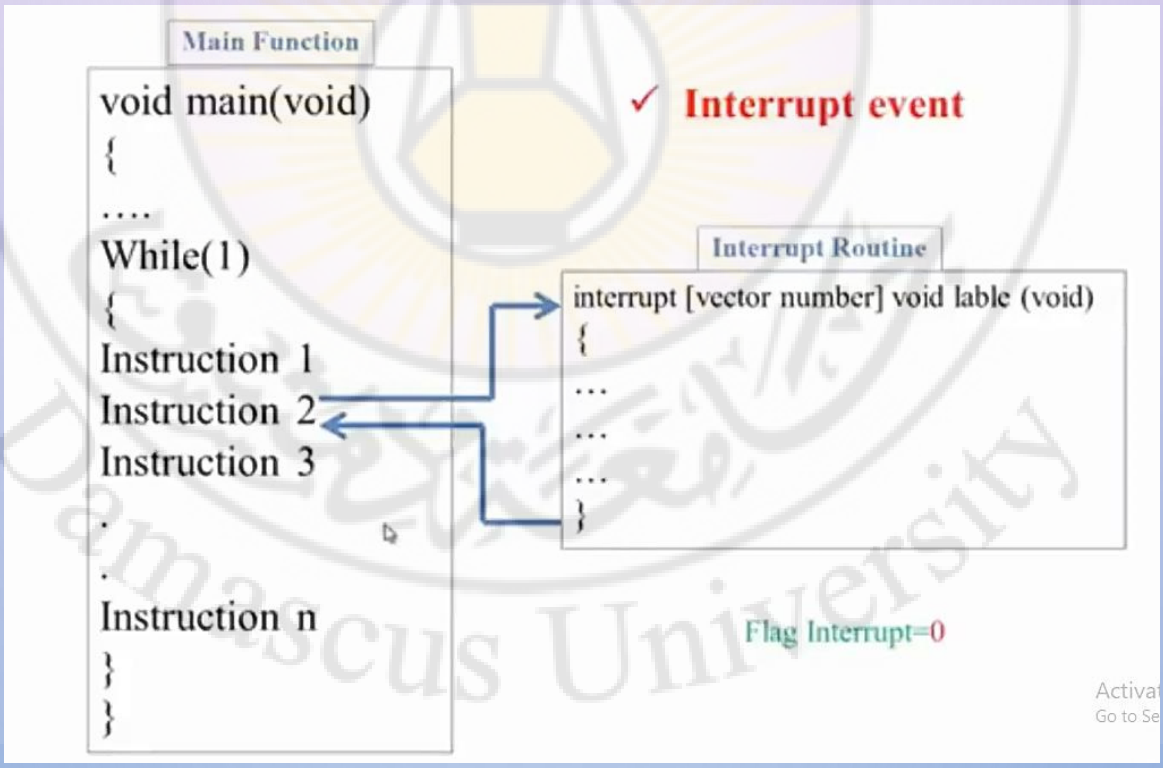
كل هذه العمليات تتم عن طريق Hardware

العمليات التي تجري عتد حدوث مقاطعة :

وعند ورود إشارة المقاطعة تجري العمليات التالية :

1. يتم رفع علم الخاص بالمقاطعة (تصبح قيمته واحد).
2. يتم وضع عنوان بداية خدمة المقاطعة في مسجل عداد البرنامج (PC) ليقوم بتنفيذ برنامج المقاطعة .
3. يتم تنفيذ برنامج خدمة المقاطعة .
4. عند الانتهاء من برنامج خدمة المقاطعة يتم تنزيل العلم الخاص بالمقاطعة (تصبح قيمته صفر).
5. يرجع المتحكم بمتابعة تنفيذ تعليمات البرنامج الرئيس ابتداءاً من التعليمة التي توقف عندهاعندما وردت إشترة المقاطعة .

كل هذه العمليات تتم عن طريق Hardware



أولويات المقاطعات :

ماذا سيحدث لو أنه حدثت مقاطعتان للمتحكم في نفس التوقيت ؟

- بشكل عام المقاطعة ذات الأولوية الأعلى هي التي يتم خدمتها أولا ، في المتحكمات AVR تم تحديد أولوية المقاطعات حسب ترتيب وضعها في متجه المقاطعة ، تأتي المقاطعة العتادية RESET في الأولوية .
- عند ورود طلب مقاطعة من النمط (int 0) إلى المتحكم أثناء تنفيذ برنامج مقاطعة من النمط (int 0) (من نفس النمط) فإن المتحكم سوف يهمل هذا الطلب ولن يستجيب لهذه المقاطعة (لأن خانة علم المقاطعة (INTF0) تحمل القيمة (1) منطقي) ، والأمر نفسه بالنسبة للمقاطعة (int 1) .
- أما إذا ورد طلب مقاطعة من نمط مختلف أثناء تنفيذ برنامج مقاطعة ما فإن استجابة المتحكم لتلك المقاطعة مرتبطة بأولوية المقاطعة الأولى على الثانية ، وتتحدد أولويات المقاطعات المختلفة وفق القاعدة التالية :

المقاطعة ذات العنوان الأصغر تمتلك الأولوية على المقاطعة ذات العنوان الأكبر فمثلا المقاطعة (RESRT) ذات العنوان (000$) لها أولوية على المقاطعة (Int0) ذات العنوان (001$) والتي لها الأولوية هي الأخرى على المقاطعة (Int 1) ذات العنوان (0X002 أو 002$) وهكذا بالنسبة إلى بقية المقاطعات.

المقاطعات الخارجية
(EXTERNAL INTERRUPTS)

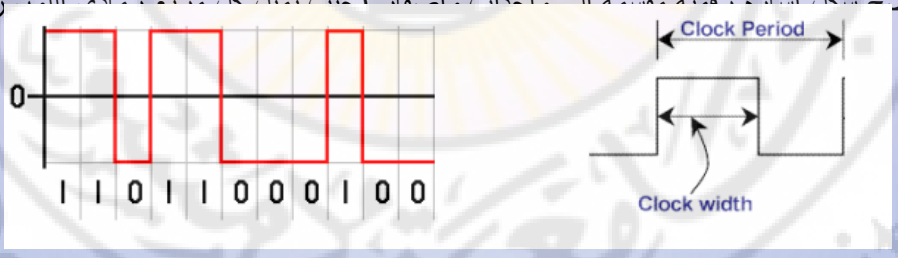
مقاطعات الطلب الخارجي على الأقطاب
(INT0~INT2)

يقصد بالمقاطعات الخارجية تلك التي يمكن توليدها بإشارات من خارج المتحكم . يوجد في ATMega16 ثلاثة أقطاب مقاطعة يمكن من خلالها توليد مقاطعات وهي INT0 وINT1 و INT2.

قبل شرح الأوامر الخاصة بضبط المقاطعات الخارجية ، سوف نتعرف على بعض الأشياء المتعلقة بالإشارات الكهربية الرقمية .

الإشارات الرقمية:

تنقسم الإشارات الكهربية الرقمية إلى نوعين وهما logic level و falling or rising level النوع الأول معروف إما أن يكون HIGH أو LOW ويعبر عن القيم الرقمية (1 و0) وتكون كل إشارة لها زمن محدد يقاس على حسب ال clock المستخدمة لتشغيل المتحكم الدقيق.فمثلا لو كان المتحكم يعمل بـ clock=1 MHZ (مليون هرتز) بأن زمن النبضة الواحد =1 ميكروثانية ويكون هذا الزمن هو نفس الزمن المطلوب لعمل إشارة كهربية بقيمة 1 أو 0 .
الصورة التالية توضح شكل إشارة رقمية مقسمة إلى وحدات وأسفلها (حيث يمثل كل مربع صغير زمن إشارة واحدة).



أنماط عمل المقاطعة :

عمل المقاطعة يكون بشكل نبضات حيث يمكن توليد المقاطعات على الأقطاب -

يمكن تحديد نمط عمل المقاطعة الخارجية بحيث:

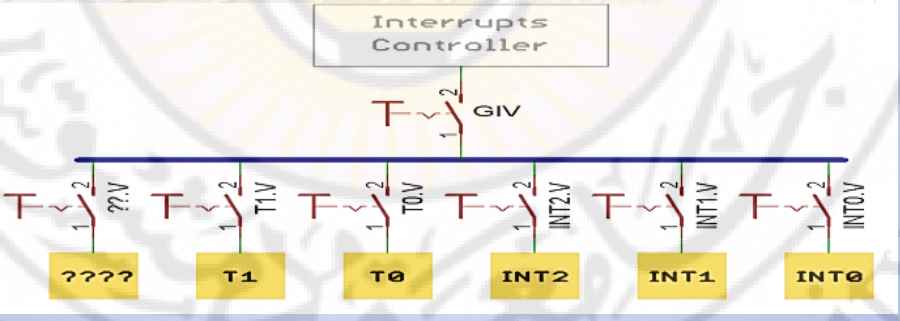
1) تقدح عن الجبهة الصاعدة (Rising Edge).
2) تقدح عن الجبهة الهابطة (Falling Edge).
3) تقدح عن مستوى الجبهة (Low Level).
4) تقدح عن تغير المستوى (Level Change).

مراحل تشغيل مقاطعات الطلب الخارجي (INTx) :

1. تحديد نمط المقاطعة الخارجية (Rising – Falling – Low – Level)
2. تفعيل شعاع المقاطعة المطلوبة تشغيلها .
3. تفعيل شعاع المقاطعات العام .

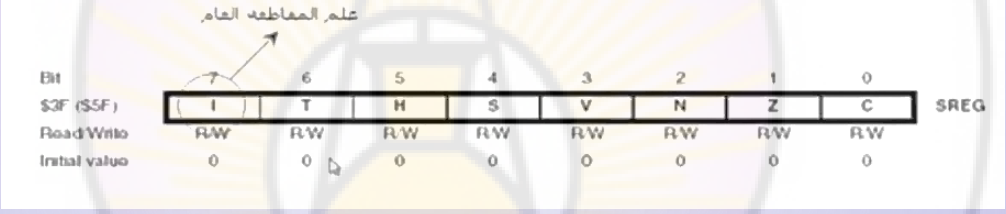
تفعيل شعاع المقاطعات العام (Global Interrupt Vector):

يمكن ان يتم تشبيهه بالقاطع الرئيسي الذي يقوم بفصل الكهرباء كاملة عند فصله ولايمكن تشغيل أي قاطع فرعي إلا بعد ان يتم تشغيله ، أي من أجل تشغيل أي مقاطعة يجب تفعيل المقاطعة وتفعيل شعاع المقاطعات العام .



علم المقاطعة العام I

هناك علم (Flag) في مسجل الحالة (SERG) يدعى بعلم المقاطعة العام (I) كما في الشكل وهو متوضع في الخانة الثامنة من مسجل الحالة :



فإذا وضعنا في هذه الخانة القيمة (1) منطقي فهذا يعني أننا فعلنا خدمة المقاطعات وبالتالي عند ورود أي مقاطعة للمتحكم سوف يستجيب لها بشكل طبيعي . وبالعكس إذا وضعنا فيه القيمة (0)م منطقي فهذا يعني أننا حجبنا المقاطعات بأكملها وبالتالي فإن أي مقاطعة ترد إلى المتحكم سوف يتم حجبها ولن يستجيب لها المتحكم أبدا إلى أن يتم تفعيل العلم (I) من جديد.

مسجلات التحكم بالمقاطعات :

- يمكن التعامل مع المقاطعات عن طريق السجلات التالية:

MCUCR : MCU control Register
MCUCSR : MCU control &status Register
GICR : General interrupt control Register
GIFR : General interrupt flag register

يتم التعامل أولا عن طريق المسجلات الثلاثة الأولى : المسجلين الأول والثاني لتفعيل المقاطعات وأعلام المقاطعات وهي بطول 8 بت والثالث هو مسجل المقاطعة العام أما GIFR هو علم المقاطعة العام الذي هو عبارة عن بت يرتفع إلى واحد منطقي عند تفعيل المقاطعة ثم بعد الانتهاء من عمل المقاطعة يعود للصفر.

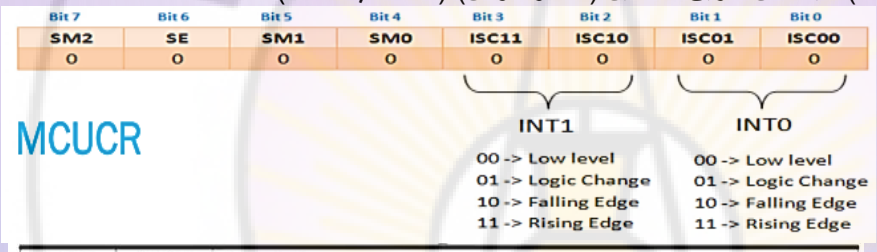
1. مسجل التحكم المقاطعة العام General Interrupt Control:

هو مسجل التمكين ، وهو مسؤول عن تفعيل وحجب المقاطعات الخارجية ، فبالإضافة إلى علم المقاطعة العام (I) هناك خانة تمكين منفصلة لكل مقاطعة من المقاطعات الخارجية INTx ، فإذا وضعنا في هذه الخانة (1) واحد منطقي نكون قد فعلنا المقاطعة الخارجية المقابلة وإذا وضعنا فيها (0) صفر منطقي نكون قد حجبنا المقاطعة الخارجية المقابلة لهذه الخانة ، وخانتي تمكين المقاطعتين (INT0,INT1,INT2),

Bit Number	7	6	5	4	3	2	1	0
GICR	INT1	INT0	INT2	—	—	—	IVSEL	IVCE
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

أما المسجل الثاني مسجل التحكم MCU (Control Register)

أهمها أول أربع بتات ، يتم تحديد الحالات أو الأخداث التي يستجيب لها قطب المقاطعة الخارجية ، عن طريق هذا المسجل يتم تحديد حدث المقاطعة (INT0, INT1) حيث عن طريق الخانتين (بت اول وثان) (ISC00,ISC01)



ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

يتم تحديد حدث المقاطعة INT0 وفق الجدول :

ISC11	ISC10	Description
0	0	The low level of INT1 generates an interrupt request.
0	1	Any logical change on INT1 generates an interrupt request.
1	0	The falling edge of INT1 generates an interrupt request.
1	1	The rising edge of INT1 generates an interrupt request.

وعن طريق الخانتين , ISC11

ISC10 يتم تحديد حدث

المقاطعة INT1 وفق الجدول

التالي :

المسجل الثالث: مسجل التحكم والحالة MCU (Control and Status Register)

أهمها البت السادس ، وعن طريق هذا المسجل يتم تحديد حدث المقاطعة لـ INT2 حيث عن طريق الخانة ISC2 وفق التالي:

ISC2=0 Falling edge

ISC2=1 Rising edge

MCU Control and Status Register – **MCUCSR**

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

If ISC2 is written to zero, a falling edge on INT2 activates the interrupt.

If ISC2 is written to one, a rising edge on INT2 activates the interrupt

المسجل الرابع هو : **علم المقاطعة العام General Interrupt Flag :**

أهمها آخر ثلاث بتات ، الخامس هو علم المقاطعة الثانية – السادس علم المقاطعة الأولى – السابع علم المقاطعة الأولى

Bit	7	6	5	4	3	2	1	0	
	INTF1	INTF0	INTF2	-	-	-	-	-	GIFR
Read/Write	R/W	R/W	R/W	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	

برمجة المقاطعة INTx

أمثلة لإعداد المسجلات :

مثال 1 : قم بإعداد المقاطعة INT0 لتعمل عند الجبهة الصاعدة :

مايهمنا هو من المسجل الأول البت صفر والبت واحد ويتم تحديد قيم الجبهة الصاعدة وفق الجدول التي مرت سابقا ، أي يكون ISC00=1 و ISC01=1 وكما يجب تفعيل البت السادس في المسجل الثاني GICR أي وضع قيمته واحد لتفعيل هذه المقاطعة INT0=1، اما المسجل الثالث لانحتاج لبرمجته .

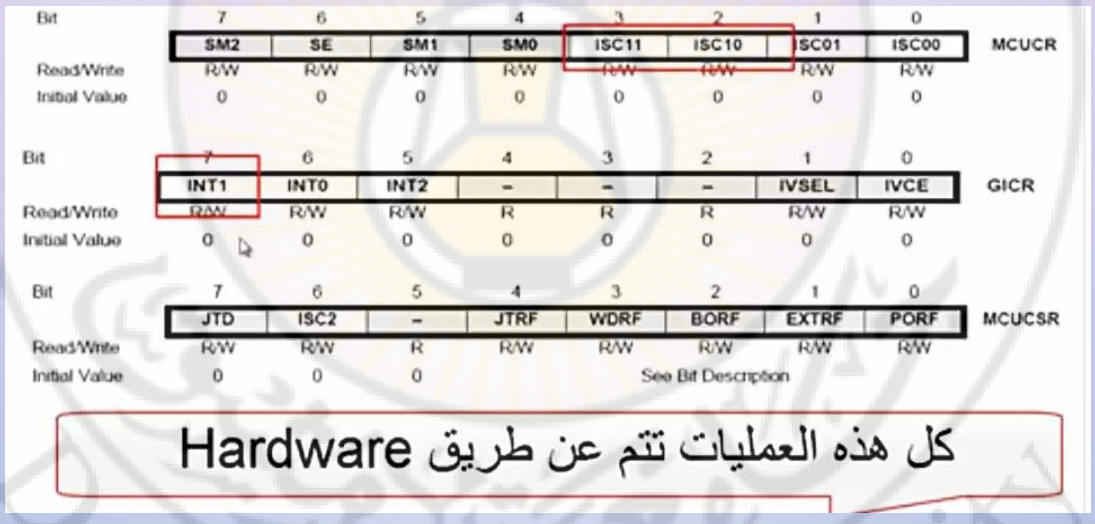
Bit	7	6	5	4	3	2	1	0	
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	MCUCR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	GICR
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	MCUCSR
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	See Bit Description					

مثال 2 : قم بإعداد المقاطعة INT1 لتعمل عند الجبهة الهابطة :

البتات الثانية والثالثة في المسجل الأول (خانة 3و4) والبت السابع في المسجل الثاني ، ويتم اختيار قيم البتات من الجدول عندما تكون ISC01=0 و ISC11=1 والمسجل الثاني INT1=1 أي تفعيل المقاطعة عند البت السابعة.



الكود التالي يمثل التركيب البسيط للـ ISR مع الـ main program

```
int main()
{
.....
}
ISR(interrupt_type)
{
.....
}
```

خطوات استخدام وبرمجة المقاطعة:

1. إعداد حدث المقاطعة عن طريق مسجلات التحكم الخاصة بالمقاطعة.
2. تفعيل المقاطعة المطلوبة عن طريق المسجلات الخاصة بالمقاطعة .
3. تفعيل المقاطعات بشكل عام عن طريق تفعيل على المقاطعة العام I بإحدى الطريقتين:

```
SREG.7=1; // اسم المسجل ورقم الخانة
OR
#asm("sei")  //باستخدام تعليمة الاسمبلي
```

كتابة برنامج خدمة المقاطعة :

هو عبارة عن تابع فرعي يكتب قبل التابع الرئيسي.

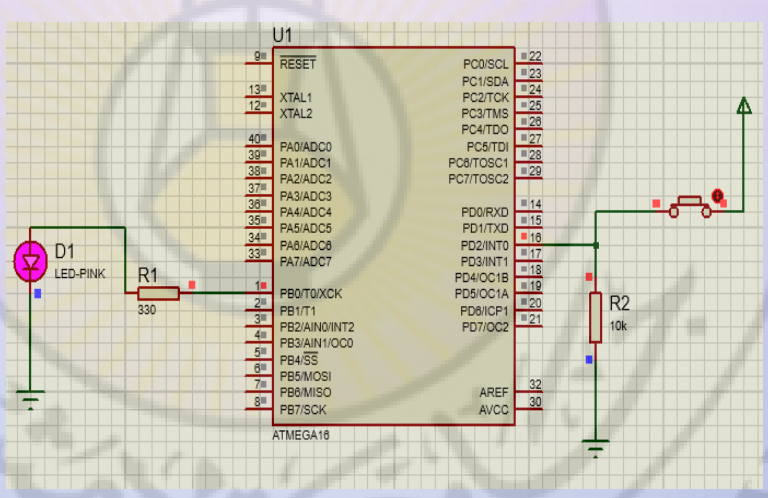
```
Interrupt [vector number] void lable (void)        //        lable اسم اختياري
{ ---                                              //        vector number  يؤخذ من رقم شعاع المقاطعة ----}
جدول المقاطعة
```

تمارين عن المقاطعات

تمارين على المقاطعات:

تشغيل وإطفاء ليد باستخدام push button

عند الضغط على الزر يقوم بتشغيل الليد وعند الضغط عليه مرة ثانية يقوم بإطفاء الليد وهكذا



ممكن أن يكون إطفاء وتشغيل أي جهاز عن طريق كباس لحظي ، كما في التمرين تم استخدام متحكم Atmega16 والكباس موصول على زر المقاطعة الخارجية INT0 والذي هو الوظيفة الثانية لـ PD2 وموصول مع مقاومة خفض خارجية ، والليد أو أي جهاز موصول مع PB0 ، في لحظة ضغط الكباس سيعطي جبهة صاعدة أي من 0 إلى 1 منطقي ، وعندما نرفع الضغط يعطي جبهة هابطة أي ينتقل من 1 إلى 0 منطقي (لحظة الكبس تتشكل الجبهة الصاعدة وعند الترك تتشكل الهابطة)

```
#include <mega16.h>
interrupt[2]void external_int0(void)
{
PORTB.0=~ PORTB.0;
}
void main(void)
{
 DDRB.0=1;
 MCUCR=0B00000011; //RISING EDGE FOR INT0
 GICR=0B01000000; //ENABLE INT0
 #asm("sei")    //ENABLE GENERAL INT0
 while (1)
     {}
}
```

كتابة برنامج خدمة المقاطعة وهو عكس حالة البورت B

قطب خرج PB0

برمجة مسجل حدث المقاطعة وفق الجداول السابقة لتفعيل الجبهة الصاعدة

برمجة المسجل الثاني لتفعيل المقاطعة INT0

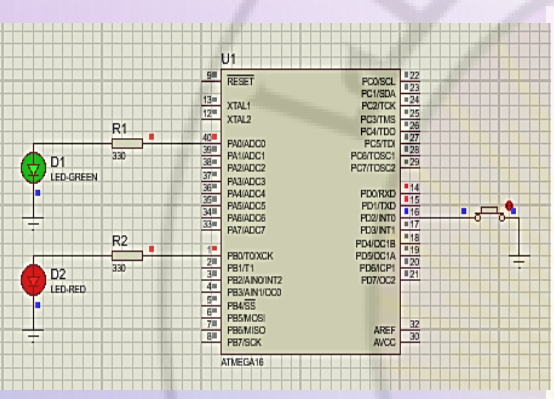
تعليمة الاسمبلي لتفعيل علم المقاطعة العامة

ثم كتابة حلقة لانهائية

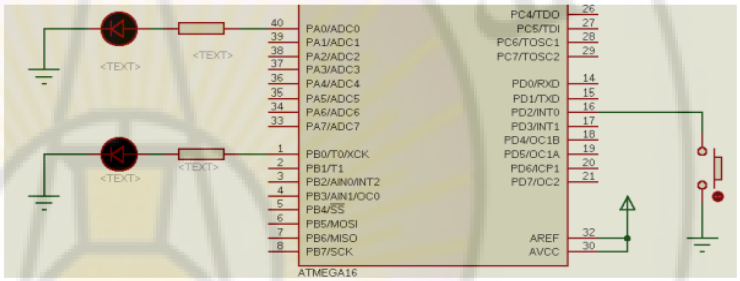
أي يقوم المتحكم ببرمجة PB.0 كخرج ثم يقوم بتفعيل الجبهة الصاعدة ثم تفعيل المقاطعة ومن ثم تفعيل علم المقاطعة العام

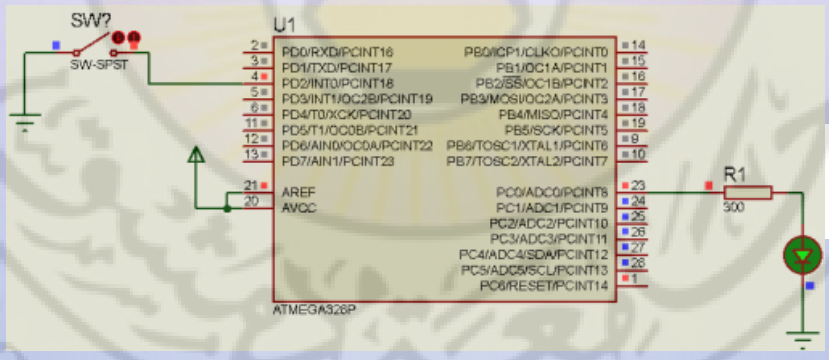
.

ويدخل بالحلقة اللانهائية ولايتوقف حتى تأتيه مقاطعة خارجية والتي هي ضغط الزر



الدائرة التالية عبارة عن 2 دايود ضوئي + مفتاح الدايود المتصل بالطرف PA0 سنقوم بتشغيله بصورة طبيعية ليقوم بعمل Blink كل مئة مللي ثانية. أما الدايود المتصل بالطرف PC0 سيتم تشغيله أو إطفاؤه فقط عند حدوث مقاطعة على الطرف INT0.





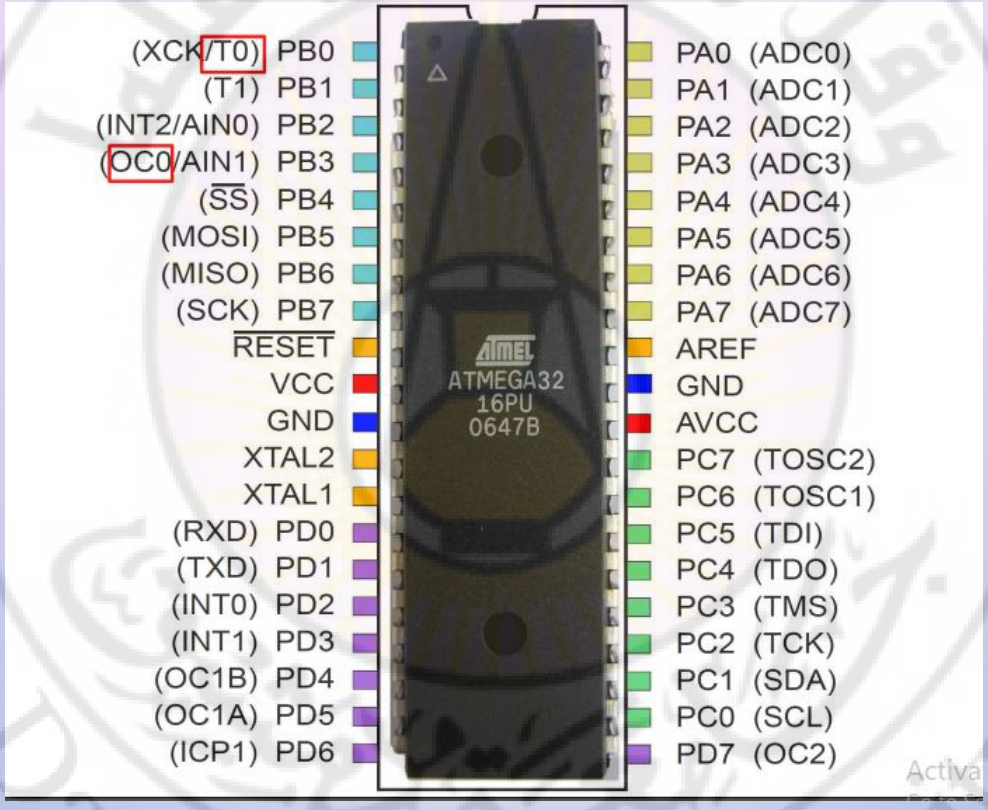
تفعيل المقاطعة باستخدام switch

الكود البرمجي

```
#INCLUDE <MEGA16.H>
#INCLUDE <DELAY.H>
INTERRUPT [EXT_INT0] VOID EXT_INT0_ISR(VOID)
{ PORTB.0=~PORTB.0;
PORTA.0=0;  }
VOID MAIN(VOID)
{
DDRA=(0<<DDA7) | (0<<DDA6) | (0<<DDA5) | (0<<DDA4) | (0<<DDA3) | (0<<DDA2) | (0<<DDA1) | (1<<DDA0);
PORTA=(0<<PORTA7) | (0<<PORTA6) | (0<<PORTA5) | (0<<PORTA4) | (0<<PORTA3) | (0<<PORTA2) | (0<<PORTA1) | (0<<PORTA0);
DDRB=(0<<DDB7) | (0<<DDB6) | (0<<DDB5) | (0<<DDB4) | (0<<DDB3) | (0<<DDB2) | (0<<DDB1) | (1<<DDB0);
PORTB=(0<<PORTB7) | (0<<PORTB6) | (0<<PORTB5) | (0<<PORTB4) | (0<<PORTB3) | (0<<PORTB2) | (0<<PORTB1) | (0<<PORTB0);
GICR|=(0<<INT1) | (1<<INT0) | (0<<INT2);
MCUCR=(0<<ISC11) | (0<<ISC10) | (1<<ISC01) | (0<<ISC00);
MCUCSR=(0<<ISC2);
GIFR=(0<<INTF1) | (1<<INTF0) | (0<<INTF2);
WHILE (1)
    {
    PORTA.0=1;
    DELAY_MS(250);
    PORTA.0=0;
    DELAY_MS(250);
    }  }
```

مصادر نبضات تزامن المتحكم

المؤقتات/العدادات


(XCK/T0) PB0
(T1) PB1
(INT2/AIN0) PB2
(OC0/AIN1) PB3
(SS) PB4
(MOSI) PB5
(MISO) PB6
(SCK) PB7
RESET
VCC
GND
XTAL2
XTAL1
(RXD) PD0
(TXD) PD1
(INT0) PD2
(INT1) PD3
(OC1B) PD4
(OC1A) PD5
(ICP1) PD6
ATMEL
ATMEGA32
16PU
0647B
PA0 (ADC0)
PA1 (ADC1)
PA2 (ADC2)
PA3 (ADC3)
PA4 (ADC4)
PA5 (ADC5)
PA6 (ADC6)
PA7 (ADC7)
AREF
GND
AVCC
PC7 (TOSC2)
PC6 (TOSC1)
PC5 (TDI)
PC4 (TDO)
PC3 (TMS)
PC2 (TCK)
PC1 (SDA)
PC0 (SCL)
PD7 (OC2)

جميع التطبيقات في الأنظمة الحاسوبية تحتاج إلى ضبط الوقت وذلك لـ :

- ضبط الأزمنة في حال حدوث أحداث أخرى مع تحديد الأوقات بدقة .
- معرفة الأزمنة بين الأوامر والأحداث التي يقوم بها المتحكم .
- تحديد الأوقات عند الحاجة للقيام بمهام محددة مثل المقاطعات وأزمنة التأخير وغيرها.

في نظام عمل المؤقت يمكن القيام بأعمال أخرى على خلاف تعليمة التأخير DELAY والتي توقف عمل المعالج حتى الانتهاء منها .

الساعة أو المؤقت يعمل بحالتين :

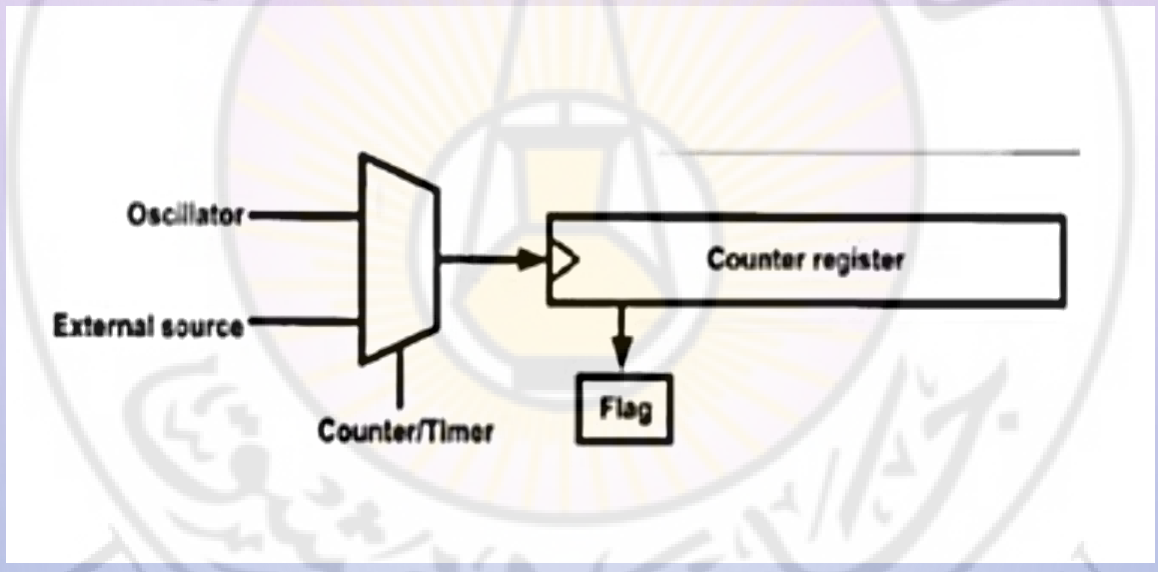
مؤقت (ساعة توقيت داخلية)

مؤقت (ساعة توقيت) خارجية

أي يقوم بالعد عند جبهات الساعة داخلية أو خارجية.

تحوي متحكمات AVR نوعين من المؤقتات 8 بت و 16 بت . يحوي المتحكم ATMega16 ثلاثة مؤقتات اثنان منهما 8 بت وواحد 16 بت .

شكل نظام التوقيت في متحكمات ATMEGA



❖ المؤقت Timer0 – أنماط العمل:

✓ :Normal Mode

Loop ($0x00_{(Bottom)}$ - $0xFF_{(Top)}$ > TOV0 = 1)

✓ :Clear Timer on Compare Match (CTC) Mode

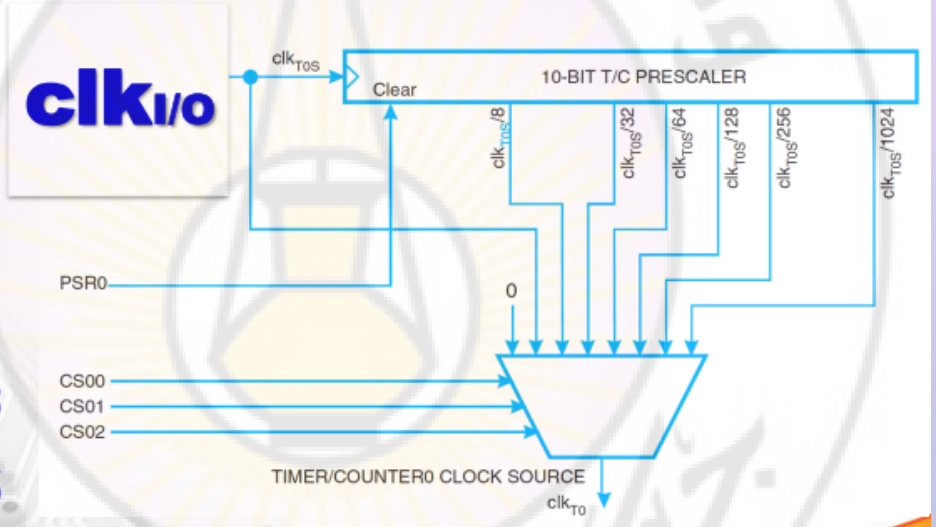
Loop ($0x00_{(Bottom)}$ – $OCR0_{(Max)}$ > OCF0 = 1)

✓ .Fast PWM Mode

✓ .Phase Correct PWM Mode

❖ المؤقت Timer0 – المقسم الترددي:

- $clk_{T0S}/1$
- $clk_{T0S}/8$
- $clk_{T0S}/32$
- $clk_{T0S}/64$
- $clk_{T0S}/128$
- $clk_{T0S}/256$
- $clk_{T0S}/1024$

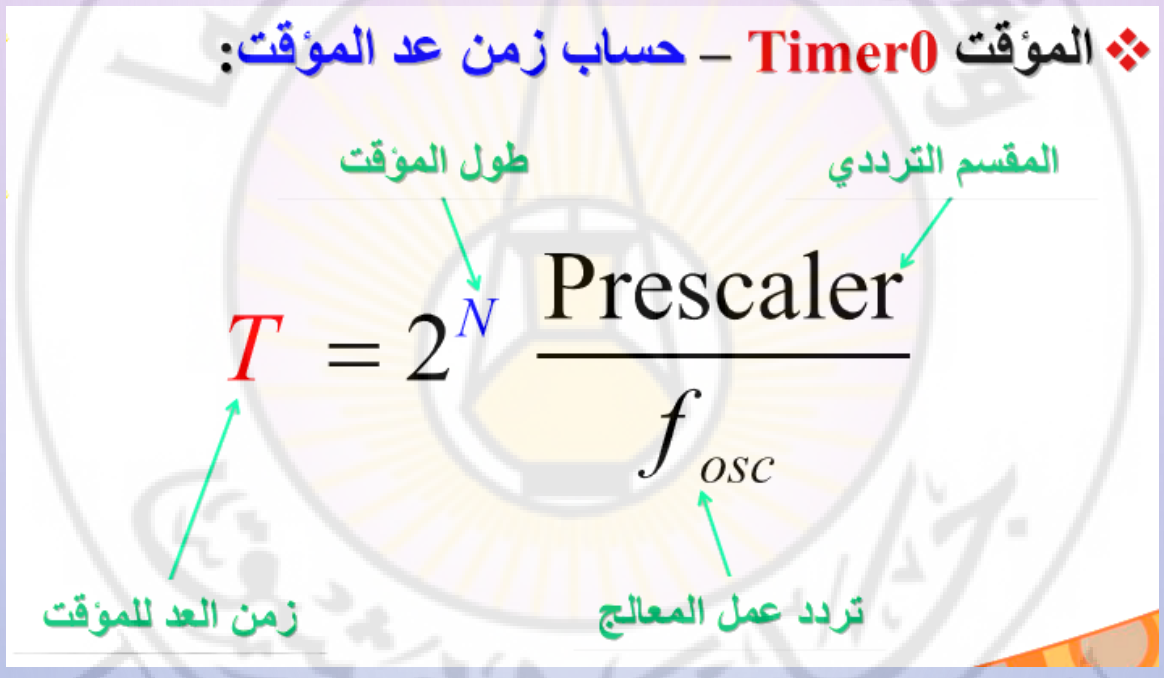


عند استخدام المؤقت الداخلي يتم ضبط المقسم الترددي لجعل عدات المؤقت على أبطأ معدل ممكن.

فمثلا حركة الغسالة تكون العداد فيها بالثواني والدقائق وعدات المؤقت بالميلي أو الميكرو ثانية ، لذلك نقوم بالتقسيم وتقليل معدل المؤقت ليحصل توازن مابين العدد الخارجي المطلوب والهزاز الداخلي (المؤقت الداخلي).

نظام الساعة 1MHZ يعني (الساعة تتغير كل 1 ميكرو ثانية) ، وعندما نقوم بتقسيمه على 46 يحصل تزايد في عداد الساعة كل 64 ميكرو ثانية .

❖ المؤقت Timer0 – حساب زمن عد المؤقت:



❖ المؤقت Timer0 – زمن العد الأعظمي للمؤقت:

$$T = 2^N \frac{Prescaler}{fosc} = 2^8 \frac{1}{1 \times 10^6} = 0.256_{mSec}$$

$$T = 2^N \frac{Prescaler}{fosc} = 2^8 \frac{1024}{1 \times 10^6} = 262.144_{mSec}$$

$$T = 2^N \frac{Prescaler}{fosc} = 2^8 \frac{1}{16 \times 10^6} = 16_{uSec}$$

$$T = 2^N \frac{Prescaler}{fosc} = 2^8 \frac{1024}{16 \times 10^6} = 16.384_{mSec}$$

زيادة قيمة المقسم الترددي تزيد من زمن العد للمؤقت .
زيادة تردد الهزاز الكريستالي تنقص زمن العد للمؤقت .

أكبر زمن للمؤقت : الهزاز على أصغر قيمة والمقسم على أعلى قيمة .
أعلى دقة للمؤقت : الهزاز على أعلى قيمة والمقسم على أصغر قيمة .

حساب دقة المؤقت : وهي أصغر زمن يستطيع المؤقت عده (زمن نبضة توقيت واحدة مطبقة على مدخل clock للمؤقت)

$$Timer_{RESOLUTION} = \frac{1}{Input\ Ferequesncy}$$

دقة المؤقت — تردد عمل المؤقت

❖ المؤقت Timer0 – حساب دقة المؤقت:

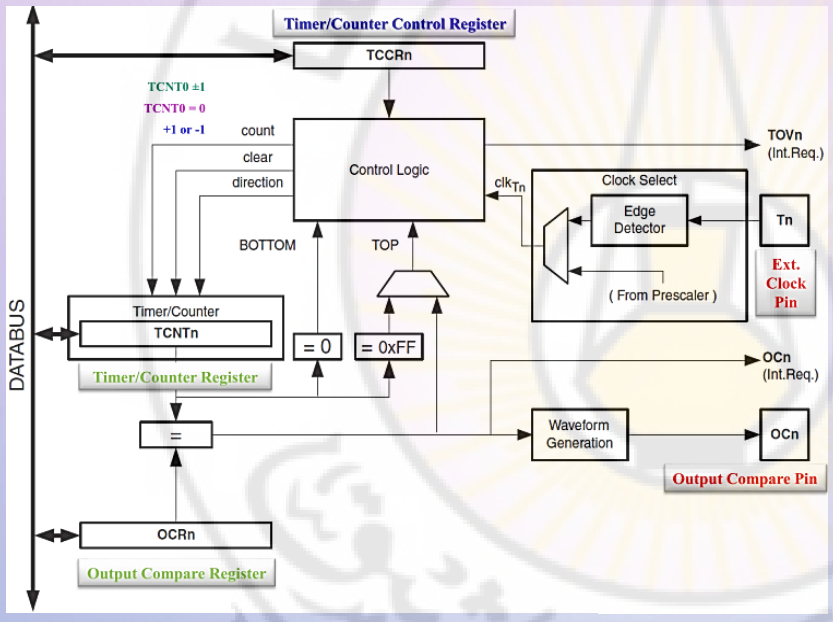
مثـــال: بفـــرض أن تـــردد الهـــزاز الكريســـتالي للمعـــالج f_{OSC}=2MHZ والمقسم الترددي Prescaler=64، وبالتالي فإن تردد عمل المؤقت هو:

$$Timer_{CLOCK} = \frac{f_{osc}}{Prescaler} = \frac{2000000}{64} = 31250HZ$$

وبالتالي فإن دقة المؤقت تساوي:

$$Timer_{RESOLUTION} = \frac{1}{Input\ Ferequesncy} = \frac{1}{31250} = 0.000032\ Sec = 32\mu S$$

الشكل التالي يبين المخطط الصندوقي للمؤقت 8 بت TIMER0:

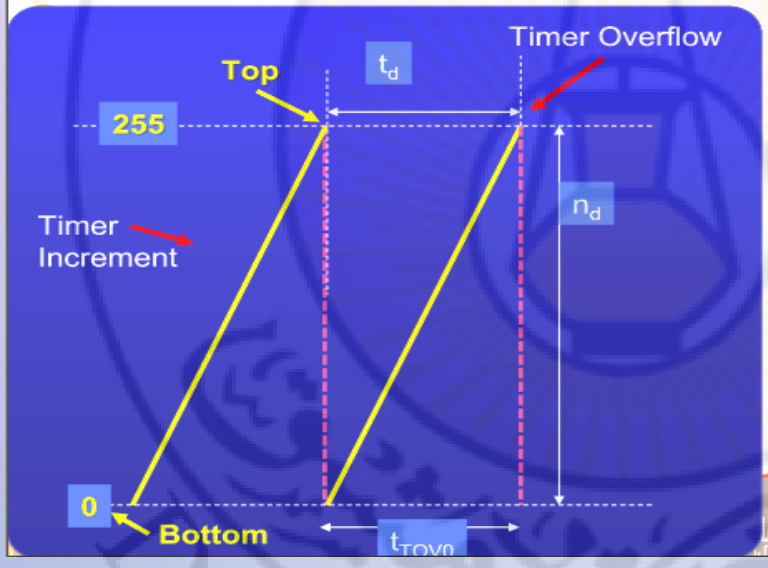


المسجلات الداخلية

1. TCNT0 مسجل العد.
2. TCCR0 مسجل التحكم .
3. OCR0 مسجل المقارنة الخارجي.
4. TIMSK مسجل قناع المقاطعة للمؤقت/العداد.
5. TIFR مسجل علم المقاطعة .
6. SFIOR مسجل (دخل/خرج) اوامرخاصة

❖ المؤقت Timer0 – نمط العمل العام (Normal Mode):

Loop ($0x00_{(Bottom)}$ - $0xFF_{(Top)}$ > TOV0 = 1)



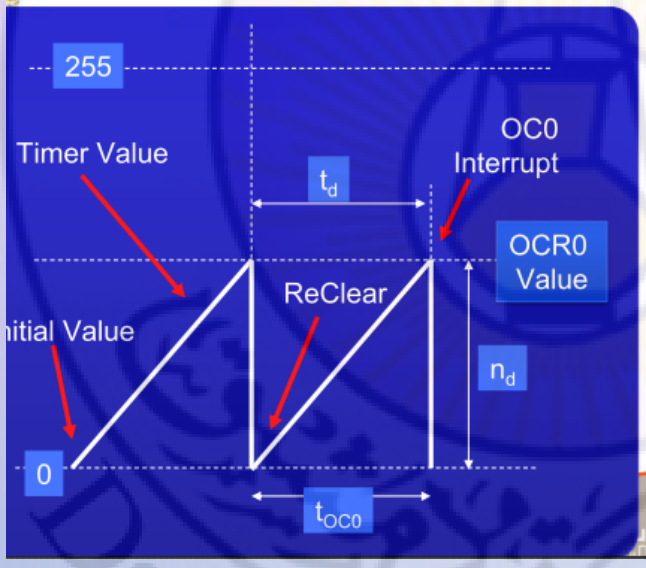
المطلوب استخدام المؤقت T0 لتوليد زمن تأخير بحدود 250mSec.

01-OVF0


Unleash your Creativity!
❖ المؤقت Timer0 – نمط العمل العام (Normal Mode):
يمكن حساب عدد النبضات على مدخل التوقيت
للمؤقت من أجل زمن معين بالعلاقة التالية:
$Target_{TIMER\ COUNT} = \frac{Target_{TIME} \times f_{OSC}}{Prescaler}$
$Target_{TIMER\ COUNT} = \frac{250 \times 10^{-3} \times 1 \times 10^{6}}{1024} = 244.114$
$Target_{TIMER\ ERROE} = 0.114 \times Timer\ RES = 0.114 \times 1.024 = 0.11674\ ms$

❖ المؤقت Timer0 – نمط العمل CTC (CTC Mode):

Loop ($0x00_{(Bottom)}$ – $OCR0_{(Max)}$ > OCF0 = 1)



المطلوب استخدام المؤقت T0 لتوليد زمن تأخير 250mSec دقيق.

❖ المؤقت Timer0 – نمط العمل CTC (CTC Mode):

يمكن حساب قيمة الشحن لمسجل نظير المراقبة للمؤقت من أجل زمن معين بالعلاقة التالية:

$$OCR0_{VALUE} = \frac{f_{osc} \times T_{required}}{Prescaler}$$

$$OCR0_{VALUE} = \frac{250 \times 10^{-3} \times 1 \times 10^{6}}{1024} = 244$$

❖ المؤقت Timer0 – نمط العمل CTC (CTC Mode):

المطلوب استخدام المؤقت T0 في نمط الـCTC

لتوليد إشارة على القطب OC0 بتردد 10KHz.

$$OCR0_{VALUE} = \frac{f_{osc} \times T_{required}}{Prescaler}$$

$$OCR0_{VALUE} = \frac{1 \times 10^6 \times 0.0001}{1} = 100 \times 0.5 = 50$$

Example: Timer/Counter 0

The frequency of **Timer/Countern (TC0)**

$$f_{OC0x} = \frac{f_{clk\_I/O}}{2 \times N \times (1 + OCR0x)}$$

$N = 1024$

$$f_{OC0x} = \frac{16,000,000}{2 \times 1024 \times (1 + 255)} = 30.52 \text{ Hz}$$

$$T_{Overflow} = \frac{1}{30.5} = 0.03277 \text{ Sec}$$

Over Flow Mode Example

***Example 3** : Write a c Code for AVR Atmega16 , to Flash a led with frequency of 0.5 Hz*

$$freq = \frac{1000000}{1024} = 976.5625\ Hz \longrightarrow T_{clk} = 1.024\ mS$$

$$8\ Bit\ Register = 2^8 - 1 = 255\ Count \longrightarrow Register\ over\ flow\ at = 255 * 1.024\ mS = 0.26112\ Sec$$

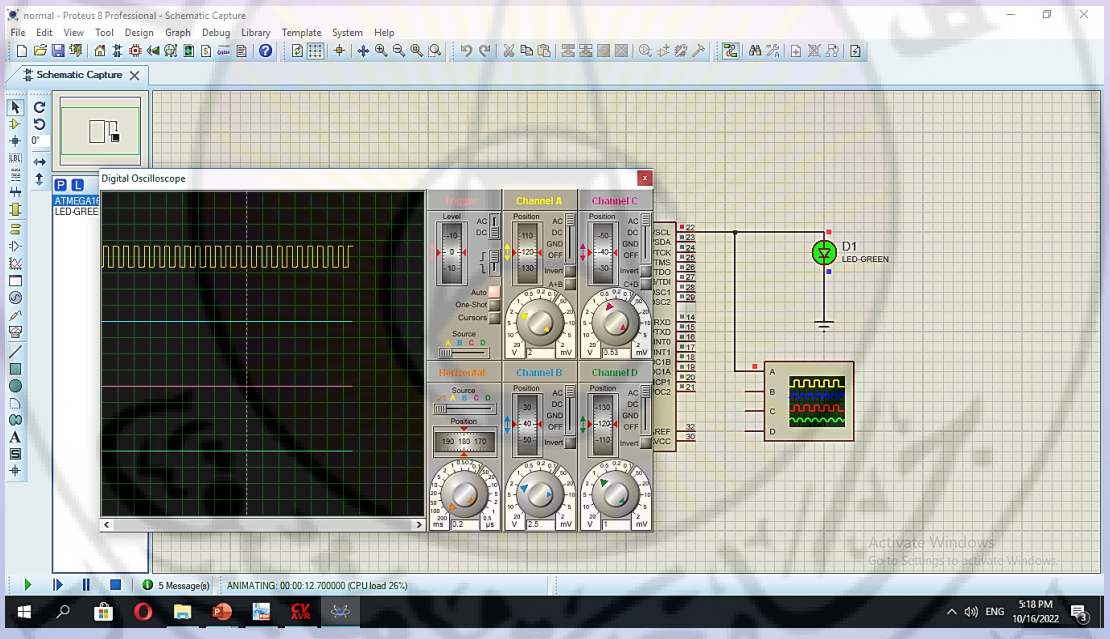
$$Number\ of\ Over\ Flows = \frac{1}{0.26112} = 3.898 \cong 4 \longrightarrow f_{LED} = 0.47\ Hz$$

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define LED_Toggle   PORTC ^= (1<<PC0) ;

 int Turn = 4 ;

void Timer_Config(void)
{
    TCCR0 |= (1<<CS00)|(1<<CS02) ;  // Prescaler = 1024
    TIMSK |= (1<<TOIE0);            // OverFlow Interrupt Enable
}
```

```
int main(void)
{
    DDRC  |=  (1<<DDC0) ;   // set port C pin 0 as output
    Timer_Config();     // configure the Timer0
    sei();
    while (1)
    {

     }
}
```

```
ISR(TIMER0_OVF_vect)
{
     Turn -- ;
    if(Turn == 0 ){
    LED_Toggle
    Turn = 4 ;
    }
}
```

وضع NORMAL


normal - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
Digital Oscilloscope
Trigger
Channel A
Channel C
Horizontal
Channel B
Channel D
D1
LED-GREEN
5 Message(s)
ANIMATING: 00:00:12.700000 (CPU load 26%)
Activate Windows
Go to Settings to activate Windows.
ENG
5:18 PM
10/16/2022

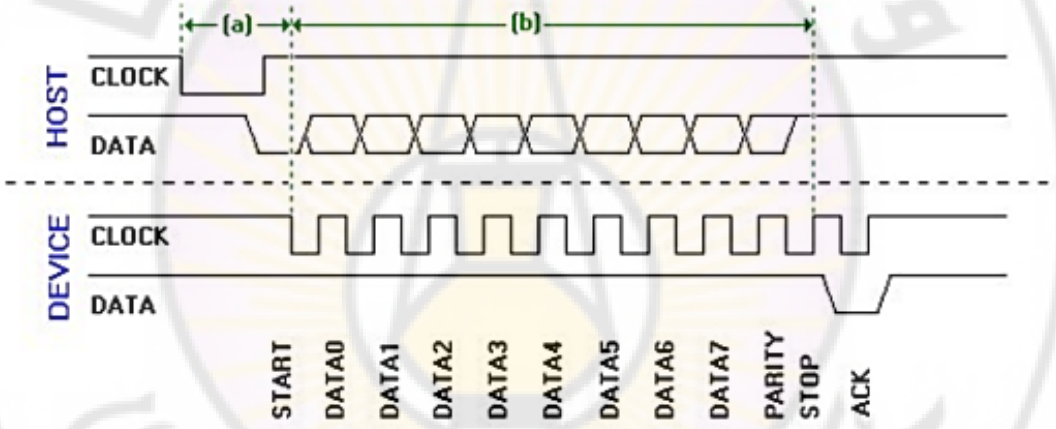
الكود

```
#INCLUDE <MEGA16.H>
INT COUNTER=31;
INTERRUPT [TIM0_OVF] VOID TIMER0_OVF_ISR(VOID)
{COUNTER--;
 IF (COUNTER==0)
   { PORTC.0=~PORTC.0;
   COUNTER=31;  }}
VOID MAIN(VOID)
DDRC=(0<<DDC7) | (0<<DDC6) | (0<<DDC5) | (0<<DDC4) | (0<<DDC3) | (0<<DDC2) | (0<<DDC1) | (1<<DDC0);
PORTC=(0<<PORTC7) | (0<<PORTC6) | (0<<PORTC5) | (0<<PORTC4) | (0<<PORTC3) | (0<<PORTC2) | (0<<PORTC1) | (0<<PORTC0);
TCCR0=(0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (0<<CS02) | (1<<CS01) | (0<<CS00);
TCNT0=0X00;
OCR0=0X00;
TIMSK=(0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) | (0<<OCIE1B) | (0<<TOIE1) | (0<<OCIE0) | (1<<TOIE0);
#ASM("SEI")
WHILE (1)
    { }
}
```

الكود

```
#INCLUDE <MEGA16.H>
INTERRUPT [TIM0_COMP] VOID TIMER0_COMP_ISR(VOID)
{
  PORTA.0=~PORTA.0; }
VOID MAIN(VOID)
{
   DDRA=0B11111111; // SET PD AS OUTPUT PORT
   PORTA.0=1;
   OCR0= 29; // SET THE MAX VALUE IN OCR0A
   TIMSK=0B00000010; // ENABLE OCIE0A INTERRUPT MASK
    #ASM("SEI")
   TCCR0=0B0011101;  // SET PRESCALAR TO 1024
WHILE (1)
  {   }
}
```

بروتوكولات الاتصالات التسلسلية



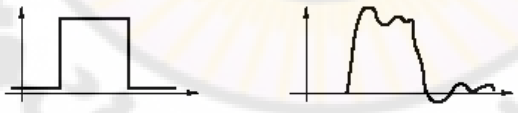
في هذا الفصل سنتعرف على أحد أشهر طرق إرسال البيانات بصورة تسلسلية بين المُتحكِمات الدقيقة والعالم الخارجي وذلك عبر بروتوكول UART والذي يعتبر أشهر بروتوكول معياري لتبادل البيانات.

مقدمة

عندما يتواصل المُتحكِم مع العالم الخارجي، فإن إرسال واستقبال البيانات يكون بشكل حزم مكونة من 8 بت(1 بايت). بالنسبة لبعض الأجهزة مثل الطابعات القديمة داخل كابل الـ Parallel port يتم إرسال البيانات من ناقل البيانات 8 بت (8-bit data bus) من الكمبيوتر إلى ناقل البيانات 8 بت في الطابعة.

يعيب هذا الأسلوب في نقل البيانات وجوب أن تكون المسافة بين الجهازين قصيرة. لأن الأسلاك تشوه شكل الإشارات الكهربية مع طول المسافة، كما أن الأسلاك المستخدمة لنقل 8 بت في نفس الوقت يكون سعرها مرتفع.

أيضاً تحدث مجموعة من الظواهر كهربية تسمى "المكثفات الطفيلية Parasitic Capacitance" و "الملفات الطفيلية Parasitic inductance" هذه الظواهر تحدث للوصلات النحاسية المتقاربة من بعضها البعض. وتتسبب في تشويه كبير لشكل الإشارة. الصورة التالية توضح شكل إشارة كهربية على صورة "نبضة pluse" بعد التشويه.



شكل الإشارة بعد التشويه الناتج من المكثفات والملفات الطفيلية

شكل الإشارة الأصلية

مبدأ عمل الاتصال التسلسلي

تستخدم تقنية الاتصال التسلسلي طرف (سلك) واحد فقط لنقل البيانات من جهاز لآخر بدلاً من 8 أسلاك كما في حالة الاتصال المتوازي Parallel ولكي يتم إرسال البيانات بشكل تسلسلي يتم أولاً تحويل البيانات من 8 بت Parallel إلى 8 بتات متسلسلة وذلك باستخدام شريحة إلكترونية (متواجدة داخل المُتحكِم الدقيق) تسمى Parallel-in-Serial-out shift register وهو عبارة عن مُسجِل إزاحة يكون دخله 8 بتات parallel وخرجه 8 بتات متسلسلة. وعلى الجانب الآخر، يجب أن يمتلك المُستقبل شريحة أخرى تقوم بعكس هذه العملية وتسمى Serial-in-Parallel-out shift register، لتحويل البيانات مرة أُخرَى إلى 8 بت متوازية.

Serial transmission

ملاحظة: كلمة بروتوكول Protocol تعني طريقة تنظيم إرسال واستقبال البيانات مثل سرعة البيانات وطريقة ترتيبها وترقيم البيانات المرسلة وكذلك الأطراف المستخدمة لهذا الإرسال والاستقبال

أنواع الإرسال التسلسلي :

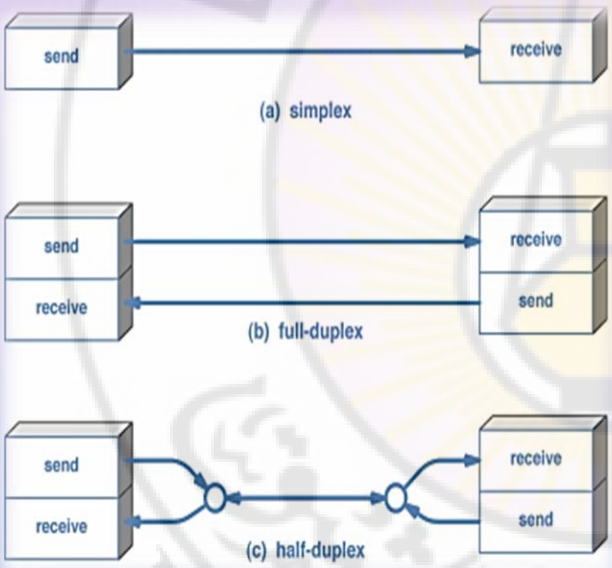
يمكن نقل البيانات تسلسليا ببروتوكول UART بطريقتين :

1. الاتصال التسلسلي المتزامن SYNCHRONOUS: يستخدم لنقل كمية من البيانات دفعة واحدة (BLOCK OF DATA) .
2. الاتصال التسلسلي غير المتزامن ASYNCHRONOUS: يستخدم لنقل بايت واحد في كل مرة .

ويتم ذلك عن طريق دوائر الكترونية مدمجة داخل المتحكم مخصصة للاتصال التسلسلي

هناك نوعان للإرسال التسلسلي :

1. SIMPLEX: عندما يكون هناك إرسال فقط أو استقبال فقط مثل الطابعة فالكمبيوتر هو الوحيد الذي يرسل البيانات.
2. DUPLEX: عندما يكون هناك قابلية للإرسال والاستقبال وينقسم إلى نوعين :
   1. HALF-DUPLEX : وذلك عندما تكون هناك القابلية للإرسال والاستقبال ولكن ليس في آن واحد مثل : جهاز اللاسلكي .
   2. FULL-DUPLEX : عندما تكون هناك القابلية للإرسال والاستقبال في أن واحد مثل الهاتف المحمول .

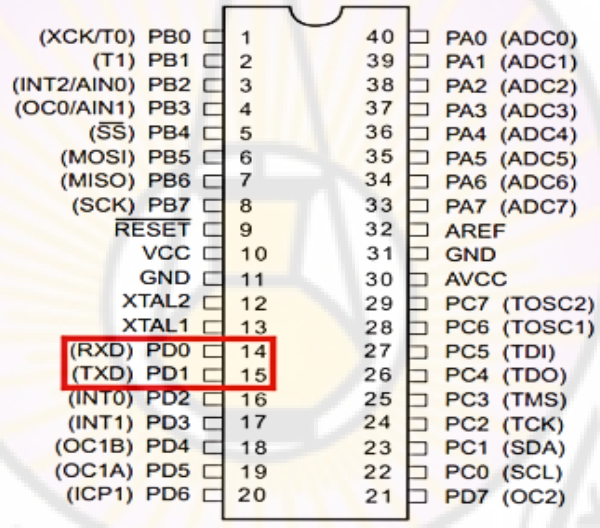


معدل إرسال البيانات:Baud rate

يقاس معدل إرسال البيانات في الاتصال التسلسلي ب bps أي bits per second. بت في الثانية. وتعتمد سرعة إرسال البيانات على النظام المستخدم، فقد كانت أجهزة IBM القديمة ترسل البيانات بسرعات تتراوح بين 100 إلى 9600 bps. ومع التطور استطاعت أجهزة المودم إرسال البيانات بسرعة تصل إلى 56kbps .

في الوقت الحالي تدعم معظم المُتحكِمات الدقيقة (بما في ذلك AVR) سرعة أنظمة الإرسال التسلسلية من نوع Asynchronous بحد أقصى 115200 بت في الثانية (نحو 100 كيلوبت في الثانية).

أطراف الإرسال والاستقبال في المُتحكِم ATmega16/32



الطرف التي تحمل الرمز **RXD** تستخدم **للاستقبال:** ويتم توصيلها بالطرف الخاصة بالإرسال في المُتحكِم الآخر.

الطرف التي تحمل الرمز **TXD** تستخدم **للإرسال:** ويتم توصيلها بالطرف الخاصة بالاستقبال في المُتحكِم الآخر.

تهيئة الـ UART الداخلي لمتحكمات AVR

يتم تهيئة ال UART للعمل عن طريق ضبط الإعدادات الخاصة ب: معدل نقل البيانات - عدد بتات الإرسال - عدد بتات النهاية... وغيره من الإعدادات والتي يتم ضبطها عن طريق تغيير قيم المُسجِلات التي تتحكم في ال UART.

يتحكم في ال UART 5 مسجلات وهي:

1- UBRR [H: L]: USART Baud Rate Register.

2- UCSRA: USART Control and Status Register A.

3- UCSRB: USART Control and Status Register B.

4- UCSRC: USART Control and Status Register C.

5- UDR: USART I/O Data Register.

1- مسجل معدل البود

2- مسجل الحالة والتحكم A

3- مسجل التحكم والحالة B

4- مسجل التحكم والحالة C

5- مسجل بيانات التراسل وهما اثنان للقراءة والكتابة

مثال : تهيئة الـ UART للعمل كمرسل أو مستقبل

نبدأ أولا بتحديد معدل نقل البيانات BAUD RATE ويتم تخزين القيمة في المسجلين UBRRH و UBRRL .

مثلا معدل إرسال بيانات يساوي BPS 9600 القيمة التي يجب تخزينها بالمسجلين يتم عن طريق العلاقة :

$$UBRR = \frac{f_{OSC}}{16BAUD} - 1$$

FOSC : تردد المذمذب الداخلي أو CRYSTAL

BAUD: قيمة معدل إرسال البيانات .

بالتعويض بالقيم تكون قيمة UBRR 103.16667 يتم وضع هذه القيمة في الكود البرمجي

تشغيل الشريحة USRAT للإرسال والاستقبال مع نفسها

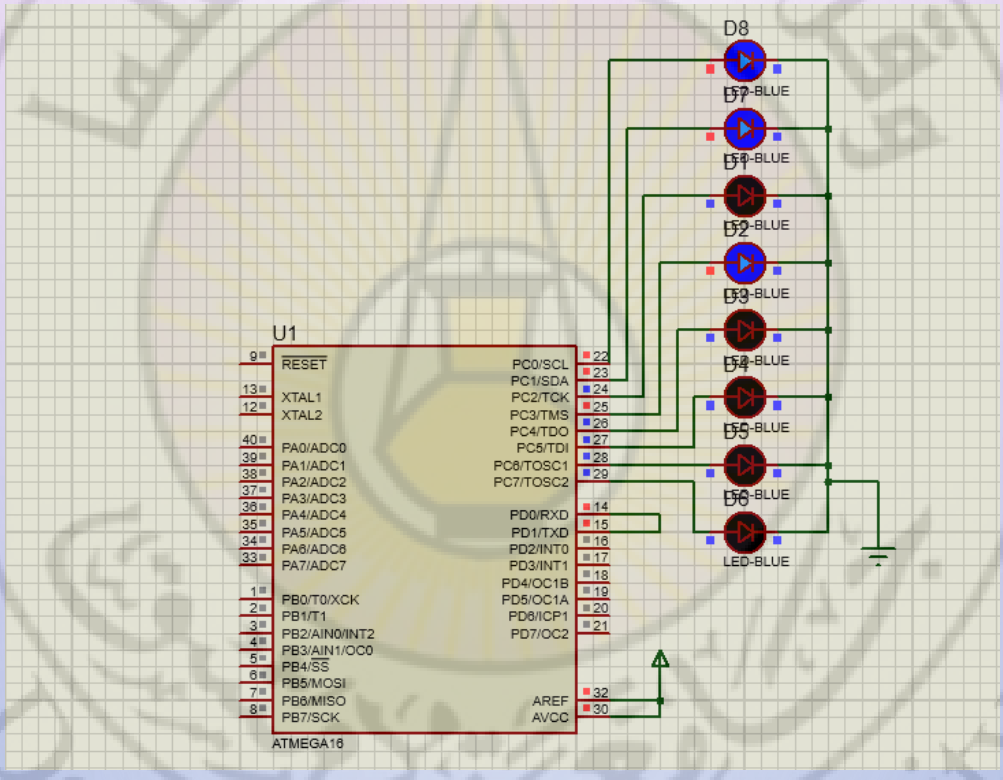
سنقوم بتوصيل طرف الإرسال TX مع طرف الاستقبال RX بحيث ان أي بيانات يتم إرسالها على الطرف TX يتم استقبالها على الطرف RX . في هذا المثال ستبدأ بيانات الإرسال بالقيمة 0X00 وعند استقبال هذه البيانات على الطرف RX نجمع واحد عليها ونعرضها على البوابة C .اي ان البوابة C في هذه الحالة ستكون بمثابة عداد ثنائي يمكننا متابعته على الخرج والكود كما يلي :

```
#include <mega16.h>
#include <delay.h>
#define F_CPU 1000000
#define BAUD 9600
#define BAUDRATE ((F_CPU)/(BAUD*16UL)-1)
void main(void)
{
unsigned char data;
  data=0x00;
  DDRC=0xFF;
  UBRRH = (BAUDRATE>>8);
  UBRRL = BAUDRATE;
  UCSRB|=(1<<TXEN)|(1<<RXEN);
  UCSRC|=(1<<UCSZ0)|(1<<UCSZ1);
while (1)
    {
    while (!( UCSRA & (1<<UDRE)));
    UDR = data;
    while(!(UCSRA & (1<<RXC)));
    data=UDR;
    PORTC=data;
    delay_ms(500);
    data++;
    }}
```

تحديد معدل بود بالقيمة ٩٦٠٠ وحساب محتويات المسجل UBRR0.

تحديد البايت العليا والصغرى من مسجل البود UBRR0K، ثم تفعيل الإرسال والاستقبال، ثم اختيار عدد بتات البيانات يساوي ثمانية. فيما عدا ذلك لن يكون هناك باريتي سيكون هناك بت توقف واحدة (قيم تلقائية).

انتظار أن يكون مسجل إرسال البيانات فارغ ثم إرسال البيانات.
ثم انتظار استكمال استقبال البيانات.


D8
D7
D6
D5
D4
D3
D2
D1
LED-BLUE
U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA16

الإرسال والاستقبال بين متحكمين عبر الشريحة USART

في هذا المثال نقوم بربط شريحتين USART عن طريق توصيل طرف الإرسال TX في الشريحة الأولى بطرف الاستقبال RX في الشريحة الثانية ، وطرف الاستقبال RX في الشريحة الأولى بطرف الارسال TX في الشريحة الثانية . ونكتب برنامج لكل من المرسل والمستقبل يقرأ البيانات المستقبلة وعرضها ، ثم يزيد عليها واحد ويعيد إرسالها للطرف الآخر .

البرنامج لكل من المرسل والمستقبل كما يلي :

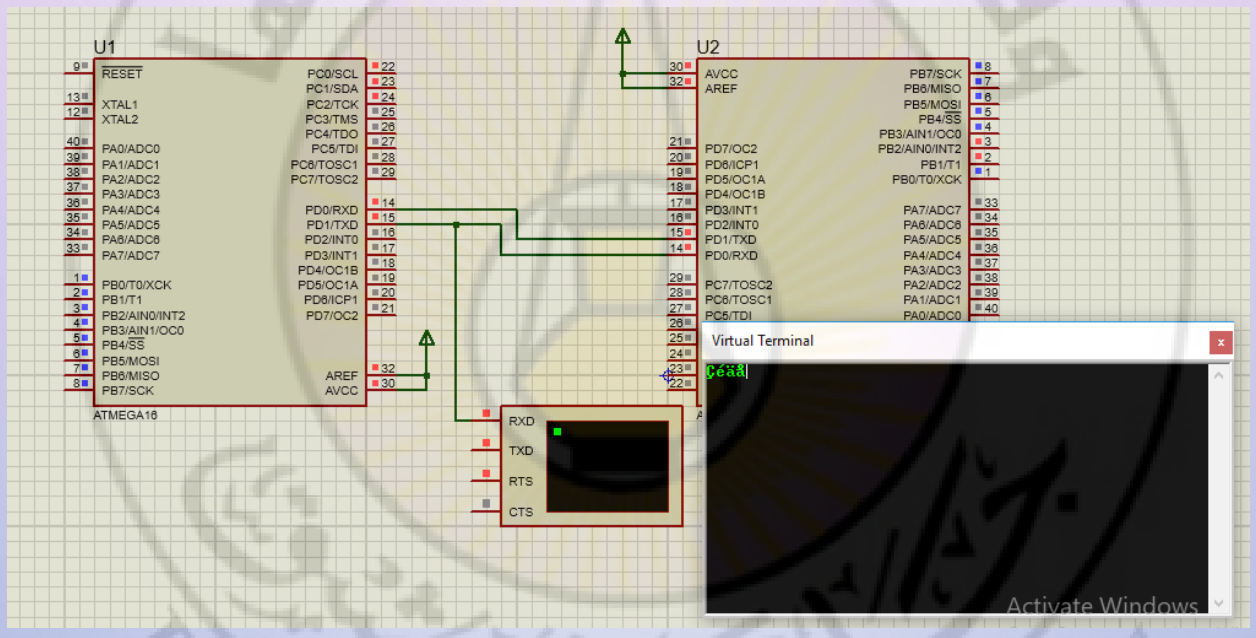
برنامج المرسل:

```
#DEFINE F_CPU 1000000
#DEFINE BAUD 9600
#DEFINE BAUDRATE ((F_CPU)/(BAUD*16UL)-1)
#INCLUDE <MEGA16.H>
#INCLUDE <DELAY.H>
VOID MAIN(VOID)
{
  UNSIGNED CHAR DATA;
  DATA=0X00;
  DDRB=0XFF;
  UBRRH = (BAUDRATE>>8);
  UBRRL = BAUDRATE;
  UCSRB|= (1<<TXEN)|(1<<RXEN);
  UCSRC|= (1<<UCSZ0)|(1<<UCSZ1);
while (1)
  {
  while (!( UCSRA & (1<<UDRE)));
  UDR = data;
  while (!(UCSRA & (1<<RXC)));
  data=UDR;
  PORTC=data;
  delay_ms(500);
  data++;    }}
```

برنامج المستقبل :

```
#DEFINE F_CPU 1000000
#INCLUDE <MEGA16.H>
#DEFINE BAUD 9600
#DEFINE BAUDRATE ((F_CPU)/(BAUD*16UL)-1)
#INCLUDE <DELAY.H>

VOID MAIN(VOID)
{
   UNSIGNED CHAR DATA;
   DATA=0X55;
   DDRB=0XFF;
   UBRRH = (BAUDRATE>>8);
   UBRRL = BAUDRATE;
   UCSRB|= (1<<TXEN)|(1<<RXEN);
   UCSRC|= (1<<UCSZ0)|(1<<UCSZ1);
```

```
while (1)
   {
   while(!(UCSRA & (1<<RXC)));
   data=UDR;
   PORTB=data;
   delay_ms(500);
   data++;
   while (!( UCSRA & (1<<UDRE)));
   UDR = data;

   }
}
```


U1
RESET
XTAL1
XTAL2
PA0/ADC0
PA1/ADC1
PA2/ADC2
PA3/ADC3
PA4/ADC4
PA5/ADC5
PA6/ADC6
PA7/ADC7
PB0/T0/XCK
PB1/T1
PB2/AIN0/INT2
PB3/AIN1/OC0
PB4/SS
PB5/MOSI
PB6/MISO
PB7/SCK
PC0/SCL
PC1/SDA
PC2/TCK
PC3/TMS
PC4/TDO
PC5/TDI
PC6/TOSC1
PC7/TOSC2
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/OC1B
PD5/OC1A
PD6/ICP1
PD7/OC2
AREF
AVCC
ATMEGA16
U2
AVCC
AREF
PD7/OC2
PD6/ICP1
PD5/OC1A
PD4/OC1B
PD3/INT1
PD2/INT0
PD1/TXD
PD0/RXD
PC7/TOSC2
PC6/TOSC1
PC5/TDI
PB7/SCK
PB6/MISO
PB5/MOSI
PB4/SS
PB3/AIN1/OC0
PB2/AIN0/INT2
PB1/T1
PB0/T0/XCK
PA7/ADC7
PA6/ADC6
PA5/ADC5
PA4/ADC4
PA3/ADC3
PA2/ADC2
PA1/ADC1
PA0/ADC0
RXD
TXD
RTS
CTS
Virtual Terminal
Çéää
Activate Windows

استخدام المؤشرات والسلاسل النصية - إرسال البيانات كسلاسل نصية :

لإرسال قيمة متغير أو جملة او استقبالها مثلا لإرسال كلمة UART يوجد طريقتين :

الطريقة الأولى: إرسال حروف متتالية

```
while(!(UCSRA & (1<<UDRE)));
  UDR = 'U';                    // إرسال حرف U
while(!(UCSRA & (1<<UDRE)));
  UDR = 'A';                    // إرسال حرف A
while(!(UCSRA & (1<<UDRE)));
  UDR = 'R';                    // إرسال حرف R
while(!(UCSRA & (1<<UDRE)));
  UDR = 'T';                    // إرسال حرف T
```

وسيتم إرسالها.

والتي تتطلب الكثير من الكواد البرمجية أي استهلاك حجم من الذاكرة

الطريقة الثانية : استخدام المؤشرات

أي POINTER فالكلمة مكونة من عدد من الأحرف بجانب بعضها كما يلي :

```
char *word = "UART";
while(*word > 0)
{
        while(!(UCSRA & (1<<UDRE)));
        UDR = *word++;

}
```

في هذا الكود استخدمنا مؤشر يشير إلى بداية الكلمة . ومن اساسيات علم الكمبيوتر أن أي STRING يحتوي آخره على الرقم 0 يدعى "NULL CHARACTER" لذلك استخدمنا الحلقة التكرارية

WHILE(*WORD>0) أي طالما أنه يشير إلى أكبر من الـ 0 ستستمر الحلقة بالتكرار .

الأمر : UDR=*WORD++ يقوم بإرسال الحرف الذي يشير إليه المؤشر حاليا ، ثم يقوم بزيادة المؤشر ليشير إلى الحرف التالي حتى يصل إلى نهاية الكلمة فيشير إلى الرقم 0 الذي يتواجد بنهاية STRING فلا يتحقق شرط الحلقة التكرارية وينتهي تنفيذها

أي كلما أردنا استخدام الـ UART يجب كتابة الأسطر الخاصة بتهيئة الـ UART وأيضا عند إرسال حرف أو كلمة ...

للتخفيف من ذلك لجعل الكود أكثر قابلية للاستخدام المتكرر باستخدام الدوال ونضع بداخل كل دالة مجموعة الأوامر التي ستنفذها هذه الدالة مثال :

الربط باستخدام البوابات التفرعية
PARALLEL PORT

مقدمة :

هناك العديد من طرائق الاتصال بالحاسب مع الأجهزة المحيطة وتختلف هذه الطرائق من حيث تقنية التخاطب وسرعة الارسال والاستقبال وكذلك حجم المعلومات المنتقلة وطريقة نقلها .

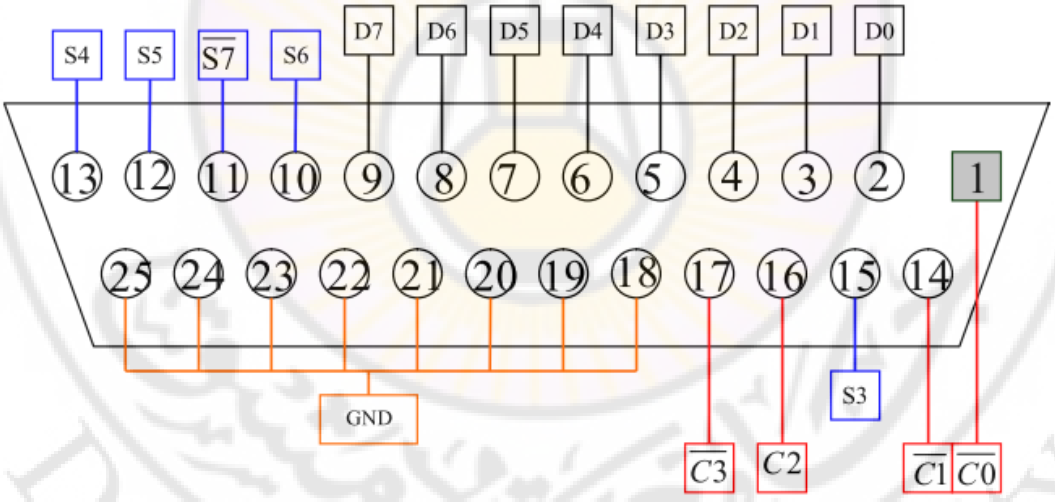
ومن هذه الطرائق المستخدمة لدينا مثلاً:

- بوابة الـــCOM (الاتصال التسلسلي).
- بوابة الـــLPT (الاتصال التفرعي).
- كرت الـــISA:ولكن هذا النوع من الاتصال يتم انشاؤه من قبل المستخدم.
- بوابة الألعابGame Port.
- حديثاً وصلة الـــUSB.

لكل طريقة لها محاسنها ولها مساوئها ولكن يمكن للمبرمج استخدام الاتصال الذي يفي بغرضه وبمشروعه .

مخطط البوابة التفرعية LPT

تتألف البوابة LPT من ٢٥ رجل موزعة بين دخل وخرج ومعطيات وخطوط أرضي كما يبينه الشكل التالي:



تتألف بوابة الـــLPT من الأرجل التالية :

- ٨ أرجل من أجل المعطيات (D0~D7).
- ٥ أرجل للدخل (S3~S7).
- ٤ أرجل للخرج (C0~C3).
- ٨ أرجل للأرضي (GND).
-

أنماط عمل بوابة الـــ LPT:

- النمط العادي :يكون في هذه الحالة جميع خطوط المعطيات تعمل كمخارج فقط.
- النمط المطور : يكون في هذه الحالة خطوط المعطيات كدخل وكخرج معاً أي أنهـا ثنائيـة الاتجـاه ويـتم الحصول على النمط المطور بارسال واحد منطقي على المخرج الخامس أي OUT5=1 ولكـن كمـا هـو ملاحظ أن هذا المخرج غير موجود في أرجل الـــLPT ولكن داخلياً يتواجد هذا المخـرج ، حيـث يمكـن برمجته بواسطة الحاسب بأحد لغات البرمجة المعروفة مثل (c++ أو VB6).