الاسمبلي ببساطة

تأليف: حذيفة مهيار

يتضمن شرح للغة الاسمبلي مع الامثلة

الطبعة الأولى
٣-٣-٢٠١٩

hoziva2017@gmail.com

عن الكاتب

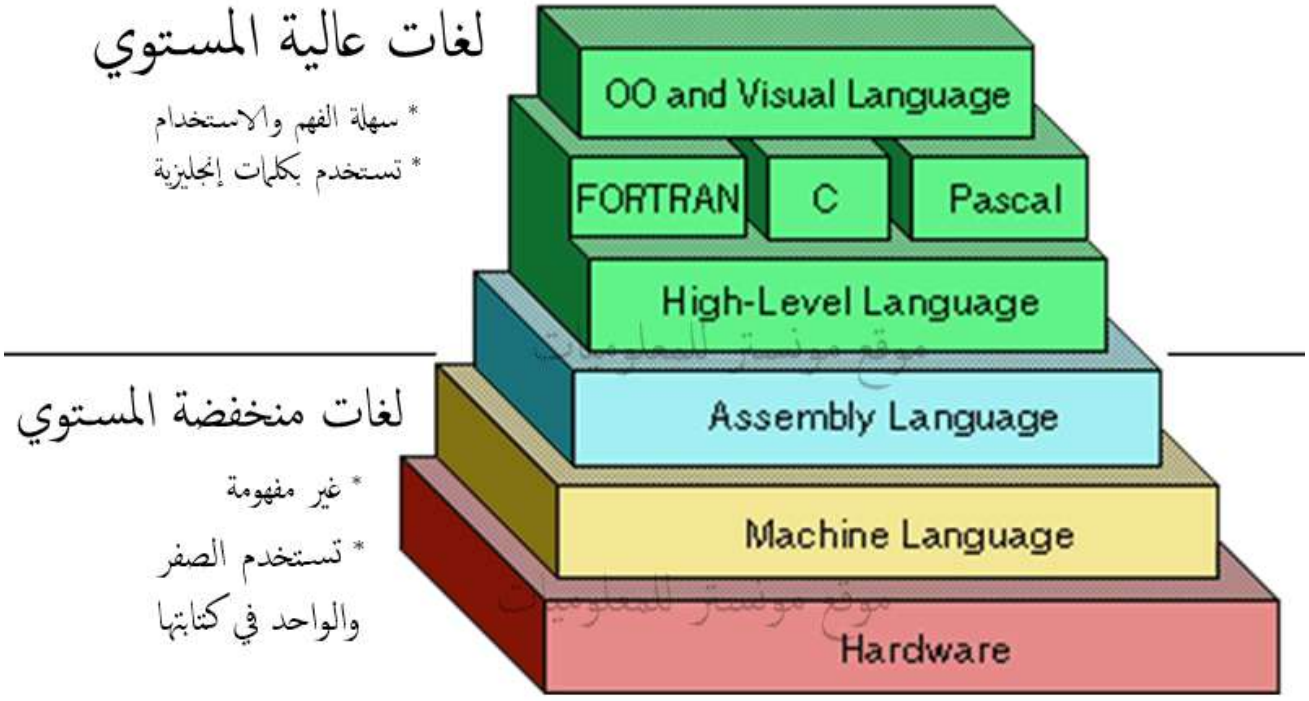
حذيفة مهيار الحمود: مواليد سوريا / دير الزور ٧٨٩١ حي الشيخ ياسين درست في كلية الترجمة جامعة دمشق ولم أنهي دراستي درست لفترة في جامعة الفرات في كلية الأدب قسم اللغة الإنكليزية كتبت مجموعة من المقالات في لغات برمجة مختلفة كتبت مجموعة من المقالات في أسس الهندسة العكسية مقالات في البرمجة بدون كود لي مجموعة من المؤلفات أبرزها المعالجات الأنواع والأصناف

الفهرس

مدخل إلى اسمبلي

مقـدمـــة الاسمبلى:

الكثير من المبرمجين المبتدئين الذين يودون تعلم لغة الاسمبلي أو حتى اساسيات الهندسة العكسية يعانون من هذه العقبة الكبيرة فأنظمة العد الغير إنسانية (أنظمة الحاسب الآلي) كنظام العد الثنائي والست عشري تختلف كلياً عن نظام العد العشري ولدراسة أي من مبادئ البرمجة بلغة المجمع (الاسمبلي) أو حتى علوم الهندسة العكسية لابد لك من أن تتعلم أساسيات نظم العد وسوف نقوم بدراسة موجزة عن أنظمة العد وطرق التحويل فيما بينها وللعلم فإن أي استخدام لحاسبة برمجية سيفي بالغرض ولا حاجة لأن نطيل البحث في موضع العمليات الرياضية لكل نوع من أنظمة العد وسنكتفي بالشرح ومن ثم طرق التحويل فيما بينها قبل الدخول في موضوع أنظمة العد لابد لنا من معرفة وحـدات التخـزين في الحاسـب الآلــي لأنها مهمة لنا في هذا البحث



وحدات التخزين للحاسب الآلي:

إن الوسائط المتعددة المرتبطة بالحاسب الآلي تمتلك مجموعة من وحدات التخزين بمختلف أشكالها (الذاكرة – القرص الصلب – مسجلات المعالج – القرص المرن – القرص الليزري – الذاكرة المخبئة داخل وحدات المعالجة) والكثير من الوسائط الأخرى التي لا يسعنا ذكرها جميعاً وطبعاً وحدات التشغيل هذه تمتلك نظام مساحة تخزين عالمي يبدأ من أصغر جزء ويسمى Bite ويمثل رقم واحد واحتمالين إما واحد أو صفر مروراً بالبايت Byte والكلمة Word وصولاً إلى أحجام ضخمة جداً ولا زالت هذه الوحدات في تزايد وطبعاً فإن ما يهمنا الآن هي الاحجام الصغيرة من وحدات التخزين هذه كونها متداخلة في علوم الهندسة العكسية أو لغة الاسمبلي فالأحجام الكبيرة لوحدات التخزين لا تدرس إلا للطلاب الراغبين في معرفة أجزاء التخزين العالية المساحة مثل القرص الصلب والتي تصل إلى آلاف الميجات والأقراص المضغوطة الخ.... إذاً ماهي هذه الوحدات تابع معنا:

القسم الأول هو البت Bite:

يعتبر البت أصغر وحدة قياس في مجال أحجام البيانات التي تأخذ حيز داخل وسائط التخزين ويمثل البيت رقم واحد فقط لا غير واحتمالين لا ثالث لهما إما الصفر أو الواحد وهذه الأعداد تتعامل مباشرة مع الآلة دون الحاجة إلى مفسر كما هو الحالة في لغات البرمجة وهذان العدادان يشيران إلى تشغيل الدارة أو إيقافها (On-Off) أو يشيران إلى الشطرين البرمجيين المشهورين صح أو خطأ (True-false) الموجودان في معظم لغات البرمجة طبعا تشكل أربع أرقام مع بعضها البعض يسمى في وحدات التخزين باسم Nibbles وهي الوحدة التي تلي البت في صغر الحجم لاحظ المثال التالي :

الخانات المتكونة من أربع أرقام تسمى Nibbles لاحظ هذه خمسة أمثلة مختلفة

1001-1101-1011-1000-1110

أما عندما نشير إلى عدد واحد من هذه الأرقام فإن نسميه Bite

طبعاً كل ما زادت الخانات زاد حجم البيانات التي تحجزها على وسيط التخزين وكل ما تشكلت الخانة من مجموعة أكبر من الأرقام الثنائية تغير قيمة البيانات التي تمثلها طبعاً هذا أمر طبيعي بكل الأحوال فالحاسوب يتعامل مع أحجام التخزين وفق سلسلة معتمدة عالمياً تبدء برقم مع البت واربع ارقام مع النبلز و ثمانية أرقام مع البايت وستة عشر مع الكلمة لاحظ أن الأعداد تتضاعف مع كل تدرج في وحدات التخزين لذلك لا يوجد وحدة تخزين فيما بينها على سبيل مثال وحدة تمثل خمسة أرقام أو حتى تسع أرقام لذلك وجب التنبيه انظر للجدول وشاهد كيفية التدرج في وحدات التخزين

البت Bite	يمثل رقم واحد	1
النبلز Nibbles	يمثل أربع ارقام ثنائية	1011
البايت Byte	يمثل ثمانية ارقام	11101110
الكلمة Word	يمثل ستة عشر رقم	10011100-10101110

القسم الثاني هو البايت Byte:

طبعاً بدون أي شك يعتبر البايت من أكثر الوحدات شهرة وقد ظهرت للمرة الأولى مع معالجات 80-x86 أي أن أصغر وحدة يتم اخالها إلى ذاكرة الحاسوب ليتم معالجتها هي

البايت أي ٨ بت في معالجات x86 ففي حالة كنت مستخدم قديم للحاسبات فلابد من إنك قد تعرفت على هذه الوحدة وخاصة من خلال الاقراص المرنة الصغيرة الحجم طبعاً بما أن البايت يتألف من ثمانية أرقام (بتات) في نظام العد الثنائي فإن ترقيم خانات هذا النظام يبدأ بالرقم صفر إلى السبعة

7	6	5	4	3	2	1	0

البت صفر إلى البت الثالث يسمى برتبة البت الدنيا Low Order Bit والبت من سبعة الى اربعة يسمى برتبة البت العليا High Order Bit مما يجعلها مقسومة إلى وحدتي Nibbles التي تتألف من اربع بتات طبعا عدد الاحتمالات في البايت يصل إلى ٢٥٦ احتمال (نطاق الاحتمالات يبدأ من 0إلى 255 بالنسبة للأعداد التي لا تحوي على إشارة أما نطاق الاحتمالات بالنسبة للأعداد التي تحوي على إشارة فيبدأ من 128- الى 127) وهذا مختلف عما هو موجود في الوحدة السابقة البت التي تكتفي باحتمال واحد حيث يتم تمثيله رياضياً 2^8 وتأتي أهمية هذه الوحدة التخزينية من كونها قادرة على التعامل مع شيفرة Ascii الشهيرة المتعلقة بالحروف الابجدية وكذلك قدرت البايت في التعامل مع الكثير من البيانات الأخرى كالمقاطعات على سبيل المثال كون النطاق الواسع من الاحتمالات التي تقدمه كما أشرنا سابقاً إضافة إلى قدرته على التعامل مع المتغيرات والتي لها دور كبير في عالم البرمجيات كونها اعتمدت كخزان للقيم العددية والرقمية فيما بعد في كل لغات البرمجة العليا والدنيا حيث يمكن كتابة المتغير التالي مع هذا النوع من وحدات التخزين (byteVar: byte;) وهذا كان غير ممكن سابقاً

القسم الثالث هو الكلمة Word:

عندما نتكلم عن هذه الوحدة فعلينا أن نعرف بأن هذه الوحدة أكبر حجماً من سابقتها وتأخذ حيز من الأرقام يصل إلى 16 رقم أي بت والتمثيل بالنظام الثنائي دائماً تخلق هذه الوحدة مقارنة بسابقتها مجالاُ واسعاُ من الاحتمالات وللعلم كل ما ازداد نطاق الاحتمالات ازدادت القوة في تعامل النظام مع البيانات من حيث الامان (لاحظ أن معالجات 64 بت تمتلك قدرة كبيرة من الأمان مقارنة مع المنصات الأقدم ٣٢بت و ١٦ بت) يكون عدد خانات الكلمة على الشكل التالي :

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

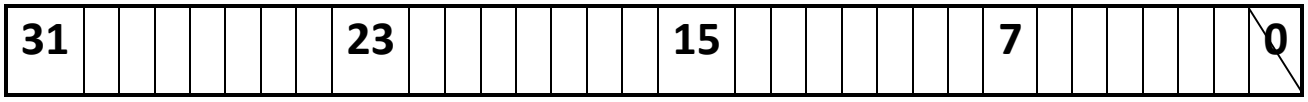
نلاحظ من المثال السابق أن هذه الوحدة تتألف من 2byte كل بايت يمثل ثمانية أرقام كما هو ظاهر معنا

أرقام البت من صفر إلى البت السابع تسمى برتبة البت الدنيا Low Order Bite أما الأرقام من الخمسة عشر إلى ثمانية رجوعاً إلى الخلف هي أرقام تسمى برتبة البت العليا High Order Bite وقد قمن بتميزيها بالألوان حتى تعرف الفرق بينها طبعا كل اربع بتات تمثل Nibble كما ذكرنا سابقاً أما بالنسبة لعدد الاحتمالات التي يقدمها هذا النوع من الأنظمة فيصل إلى (65536) احتمال يتم تمثيله رياضياً بالعدد اثنان مضروباً بست عشرة 2^{16} طبعاً بالنسبة لنطاق الارقام بدون إشارة فإن الاعداد تبدأ في هذا النوع من وحدة التخزين بصفر حتى نصل إلى الرقم 65535 أم الاعداد التي نحوي على إشارة فإن النطاق يكون بين العددين 32,768- إلى 32,767 طبعاً سمحت هذه الاحتمالات بظهور ما يعرف بنظام UNICODE والذي سمح بالتعامل مع المحارف الغير رومانية كالمحارف الأسيوية والمحارف اليونانية والروسية

طبعاً هذا النظام سمح بكتابة متغيرات بنظام ١٦ بت بالنسبة للمبرمجين ومعه ظهرت ما يعرف بلغات البرمجة العليا مبدئيا التي كتبت تطبيقاتها بهذا المنصة فيما بعد والتي تدعمها أنظمة تشغيل موجدة إلى هذه اللحظة

القسم الرابع هو الكلمة المزدوجة DWord:

الكلمة المزدوجة تتكون من اربع بايتات ومن اسمها يتبين إنها ضعف خانات الكلمة العادية أي ٣٢ بت مقارنة مع ١٦ بت في وحدة التخزين السابقة Word تحوي الكلمة المزدوجة على عدد كبير من الاحتمالات يصل طبعا إلى 4,294,967,295 وهو عدد هائل من الاحتمالات مقارنة مع أقسام وحدات التخزين الأخرى ويقع نطاق التعامل مع الاعداد ذات الاشارة بين الرقمين 2,147,483,647..2,147,483,648- بكل الأحوال فإن وحدة التخزين هذه تمثل نقلة نوعية كسابقتها من أنظمة العد الأخرى لأننا اصبحنا نتكلم الآن عن الحواسيب الحديثة العهد والتي تتعامل بشكل واسع مع هذا النوع من وحدات القياس



طبعا الارقام من صفر إلى خمسة عشر تسمى برتبة البت الدنيا أما الأرقام من واحد وثلاثين إلى الرقم ستة عشر نسميها برتبة البت العليا أما بالنسبة للمتغيرات البرمجية فقد حصلنا على إمكانية أوسع في هذا المجال مع زيادة عدد البتات والذي يعني كتابة متغير بطول 32 بت

القسم الرابع وحدات التخزين الأخرى:

نتكلم في هذا القسم عن الوحدات الأخرى المتبقية في عالم تخزين البيانات الصغيرة الحجم بشكل سريع والتي تتمثل أولاً بالوحدة التخزينية Quad Word الكلمة الرباعية والتي تمثل 8Byte أي بالنظام الثنائي 64 بت هذا النوع من الوحدات لايزال يعتمد في مسجلات الحاسب الآلي من نوع ٦٤ بت أو في نمط عمل انواع من ذواكر الحاسوب بشكل علمي أصبحنا نتكلم الآن على نطاق واسع جداً من الاحتمالات فنحن الآن نتكلم عن 18,446,744,073,709,551,615 لاحظ معي كمية الأرقام التي يعجز الكثيرين عن كتابتها أو حتى قراءتها الوحدة التخزينية التي تليها تسمى Paragraph والآن أصبحنا نتكلم عن 16 بايت و128 بت ومجال احتمال أوسع من سابقه وبهذا ننهي دراستنا لوحدات التخزين الصغيرة الحجم والتي تلعب دور كبير إلى وقتنا هذا في طريقة تخزين بيانات بسرعة ودقة كبيرة وكل ذلك سيتم ذكره لاحقاً

اسـم الوحدة	عـــدد البتـــات	الاحتمـــالات	البايت
البت Bite	١	٢	٠
البايت Byte	٨	٢٥٦	١
الكلمة WORD	١٦	٦٥٥٣٦	٢
الكلمة المزدوجة DWORD	٣٢	4,294,967,295	٤
الكلمة الرباعية QWORD	٦٤	18,446,744,073,709,551,615	٨
المقطع Paragraph	١٢٨	عدد ضخم من الاحتمالات	١٦

القسم الخامس وحدات التخزين الكبيرة الحجم:

وهذه الوحدات تعتبر معروفة مقارنة بسابقتها من وحدات التخزين الأخرى كونها مستخدمة في الأقراص الصلبة والمرنة والليزرية الوحدات التخزينية الصغيرة الحجم تستخدم في أجزاء معينة من العتاد الصلب بشكل رئيسي وبحجم ثابت وأحياناً بمعدل نقل ثابت للبيانات كالمسجلات أما الكبيرة الحجم منها فتستخدم اليوم في الذاكرة والقرص الصلب وغيرها بكل الأحوال سنقوم بعرضها على شكل جدول صغير حتى لا نطيل دراستها كثيراً كونها لا تدخل كثيراً في دراستنا لعلوم الهندسة العكسية

الوحدة	الرمز	حجم التخزين
الكيلوبايت Kilo Byte	KB	١٠٢٤ بايت
الميجابايت Mega Byte	MB	١٠٢٤ كيلوبايت
الجيجابايت Giga Byte	GB	١٠٢٤ ميغابايت
التيرابايت Terabyte	TB	١٠٢٤غيغابايت
البيتابايت Petabyte	PB	١٠٢٤ تيرابايت

بعد أن قمنا بدراسة وحدات التخزين الصغيرة والكبيرة الحجم يمكنا القيام بدراسة أنظمة العد للحاسب الآلي وطرق التحويل بين هذه الأنظمة وهذا البحث يعتبر متداخل مع وحدات التخزين لذلك لابد من فهم كل ما أخذناه سابقاً قبل الشروع للدخول في عالم الأنظمة العددية

الأنظمة العددية

تعتبر الأنظمة العددية جزء مهم من كينونة البشر وجزء أساسي في عمل الآلات حولنا ومن بينها الحاسب الآلي وتتعامل الحاسبة الآلية مع عدة أنواع من الأنظمة العددية سنتعرف عليها بشكل سريع ولابد من ذكر نقطتين مهمتين أساسيتين أولا لن أتطرق للنظام الثماني والذي يدخل أيضا في عمل الآلة وثانيا أن الآلة الحاسبة والكثير من أدوات التحويل المباشر تفي بالغرض لذلك لنتعرف على ثلاث أنظمة عد أساسية وهي كالتالي:

نظام العد العشري

نظام العد العشري هو النظام المتبع عند جميع البشر فعند استخدامك لأرقام هذه المجموعة فإنك لن تفكر أبداً بالقيمة العددية المرادفة لها كما هو متبع في الأنظمة العددية الأخرى التي تستخدمها الحاسبات الآلية وسميت بنظام العد العشري نسبة إلى أصابع يد الإنسان العشرة والتي كان يستخدمها البشر منذ قدم التاريخ للقيام بعملياتهم الحسابية البسيطة وفي حالة أردنا القيام بشرح مفصل للعمليات الِحسابية فيما بينهم وفي حالة أردنا شرح مفصل للعمليات الحسابية في هذا النوع من الأنظمة العددية فبالتأكيد سيعود ذلك بنا إلى سنوات دراستنا الأولى للأعداد من جمع وطرح وضرب وقسمة وكل ما تقدمت سنوات الدراسة ازدادت العمليات الحسابية تعقيداً وأصبحنا ندرس الجذور والمكملات والمصفوفات وغيرها بكل الأحوال فإن الثورة التي جاءت مع نظام العد العشري كانت كبيرة وواكبتها تطورات في استخدام حاسبات تقوم بإحصاء مراتب الأعداد كالآحاد والمئات والآلاف ونتكلم عن ذلك منذ فترة بعيدة لن نطيل البحث في هذا النوع من أنظمة العد كونه نظام عد متبع للعديد من دول العالم وسنقوم بالتركيز على بعض النقاط الأساسية حول الموضوع

يقسم الرقم في النظام العشري إلى عدة مراتب حيث يتم تجزئة الرقم ٣٥٥ إلى ثلاث مراتب الخمسة من اليمين هي الآحاد والخمسة الأخرى هي العشرات والثلاثة هي المئات فتصبح العملية الحسابية مجزئة كالتالي:

$$5\times100 \qquad 5\times10 \qquad 5\times1$$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$5\times100^{2} \qquad 5\times10^{1} \qquad 5\times1^{0}$$

العد في النظام العشري يبدأ من الصفر حتى الوصول إلى الرقم تسعة وفي حالة أردنا الاستمرار في العد فلا بد من العودة إلى الصفر ثم نأخذ خانة جديدة يمثل الرقم واحد وعند الوصول إلى الرقم تسعة عشر وفي حالة القفز إلى العدد التالي نستخدم الرقم اثنان وهكذا ثلاثة والاربعة

1-2-3-4-5-6-7-8-9

10-11-12-13-14-15-16-17-18-19

20-21-22-23-24-25-26-27-28-29

الانتقال في مراتب الأعداد نبدأ بالأحاد برقم واحد العشرات تمثل رقمين حت الرقم ٩٩ من بعدها ندخل المئات حتى الرقم ٩٩٩ وبعدها ننتقل إلى الآلف مع أربعة أرقام في أنظمة العد الأخرى الأمور تبدو مختلفة كلياً

مرتبة الآحاد 4-3-2-1

مرتبة العشرات 30-20-10

مرتبة المئات 390-280-185-120

مرتبة الآلاف 9000-8400-1555-1222

نظام العد الثنائي

هو نظام العد الاساسي في لغة الآلة وبشكل خاص في أنظمة الحاسب الآلي ويمثل النظام العد في هذا النوع وجود رقمين (1.0) لذلك سمي هذا النوع من أنظمة العدد بالنظام الثنائي كل واحد من هذه الأرقام يساوي خانة واحدة يعني البت أصغر وحدة قياس كما ذكرنا سابقاً في درستنا لوحدات التخزين فأي شيء يتم التعامل معه في دارات الحاسب الآلي سواء أكان بيانات رقمية أو كتابية يتم رده إلى النظام الثنائي فعلى سبيل المثال يتم إرجاع الرقم 15 في النظام الثنائي إلى 1111 أربع أرقام تمثل الواحد وتمثل ما يعرف Nibble وهي وحدة قياس تتألف من أربع بتات

في حالة النظام الثنائي يمكن استخدام الحاسبة العلمية لتنفيذ عمليات حسابية تخضع لهذا النوع من أنظمة العد وبقية أنظمة العد الأخرى بمختلف أنواعها طبعاً يمكن التحويل بين هذه الأنظمة من دون استخدام الحاسبة الآلية وهذا ما سنقوم بدراسته لاحقاً وسنقوم بتخليص أهم الافكار المتعلقة بهذا النوع من الأنظمة العددية

نظام العد الثنائي يمثل رقمين الصفر والواحد فقط لا غير وطريقة العد في هذا النظام تختلف عن النظام العشري حيث يبدأ العد ب 00 ثم العد 01 وهكذا

تمثل الرقمين في نظام العد الثنائي الواحد والصفر وجود فولت أو عدم وجوده (5.5v-0.5v) داخل دارات الحاسب الألي كما يمثل العبارتين البرمجيتين صح وخطأ الشهيرتين True-False

العمليات الحسابية في النظام الثنائي تختلف كليا عما هو موجود في نظام العد العشري

التحويل من النظام الثنائي إلى النظام العشري:

التحويل من النظام الثنائي إلى النظام العشري نقوم بضرب العدد بالرقم اثنان مرفوع للأس بحسب عدد خانات الحرف لاحظ التالي

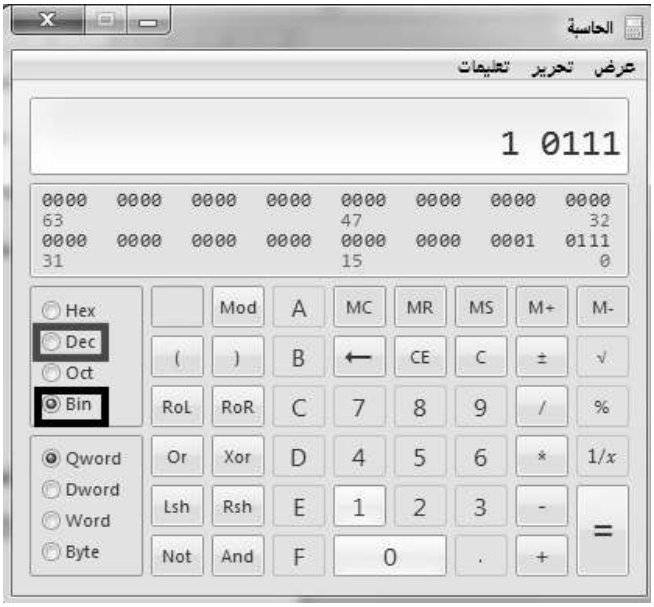
الرقم الثنائي	1	0	1	1	1
مضروب بالعدد 2 لأنه نظام ثنائي	2	2	2	2	**2**
مرفوع للأس حسب الخانات بدء من الصفر حتى أخر خانة	4	3	2	1	**0**

كيف يتم التحويل هذا العدد (10111) الممثل بالجدول الاساسي وما ذا يعني هذا الجدول لاحظ التالي وهي طريقة كتابة معادلة تحويل العدد الثنائي إلى عشري ضبط كما قلنا سابقاً بالضبط فقد ضربنا القيمة الثنائية بالعدد اثنان ثم رفعنا للأس رقم خانة العدد حسب ترتيبه بدءً من الصفر وحتى نهاية اخر بت

$1*2^0 + 1*2^1 + 1*2^2 + 0*2^3 + 1*2^4 = 23$

1+2+4+0+16=23

لاحظ أن نتيجة التحويل من النظام الثنائي إلى العشري هي ٢٣ وهي النتيجة الصحيحة وللتأكد قم باستخدام الآلة الحاسة في ويندوز وأختر النمط البرمجي من قائمة عرض وقم بعملية التحويل باختيار الاختصار Bin ثم كتابة العدد بالنظام الثنائي وبعدها نقوم باختيار الاختصار Dec وبالتالي تتم عملية التحويل وهذا ينطبق على جميع عمليات التحويل الأخرى بين نظم الأعداد بكل الأحوال عليك وضع الرقم 10111 للتأكد من صحة العملية التي قمنا بها انظر للشكل الذي في الأسفل والذي يمثل الحاسبة البرمجية وشاهد العلامة على مربعي التحويل من وإلى النظامين العددين



الآن سنقوم بأخذ مثال ثاني لعملية التحويل من النظام الثنائي إلى النظام العشري وسنقوم بتحويل العدد الثنائي (11001) بنفس عدد خانات المثال السابق لاحظ التمثيل ضمن الجدول

الرقم الثنائي	1	1	0	0	1
مضروب بالعدد 2 لأنه نظام ثنائي	2	2	2	2	**2**
مرفوع للأس حسب الخانات بدء من الصفر حتى أخر خانة	4	3	2	1	**0**

الآن سنقوم بالتطبيق عملياً للتحويل

$1*2^C + 0*2^1 + 0*2^2 + 1*2^3 + 1*2^4 = 25$

1+0+0+8+16=25

طبعاً عملية الضرب تبدأ من اليمين إلى اليسار والناتج لعملية التحويل يكون بالتأكيد هو العدد العشري ٢٥ وهذه عملية تحويل بسيطة ولكن ماذا لو أردنا التحويل بين النظامين لكن بفاصلة يبقى الأمر كما هو بالنسبة للأعداد على يسار الفاصلة أما بالنسبة للأعداد التي على يمين الفاصلة فالأمر مختلف تماماً حيث أن الأس المرفوع لهذه الأعداد يكون سلبياً لاحظ الجدول التالي والذي يمثل العدد (110.1) والتحويل بالنظام الثنائي

الرقم الثنائي	1	1	0	.	1

2	.	2	2	2	مضروب بالعدد 2 لأنه نظام ثنائي
-1	.	2	3	4	مرفوع للأس حسب الخانات بعد الفاصلة تدريجياً بدء 1-

وعملية تطبيق التحويل تكون بالطريقة الأتية

110,1

$0*2^0+1*2^1+1*2^2+1*2^{-1}=7.50$

0.50 =7,50+1+2+4

إنا نتيجة ضرب العدد $1*2^{-1}$ هو 1*1/2 وبالتالي فالنتيجة النهائية لهذه العملية هو 0.5 كلما تقدمنا بعد الفاصلة تضاعف الرقم المقسوم عليه يعني على النحو التالي -1/4-1/2 1/8-1/16 وهكذا تتم العملية التحويلية سنأخذ مثال أخر حيث سنقوم بتحويل المثال التالي (10011.11)

$1*2^0+1*2^1+0*2^2+0*2^3+1*2^4+1*2^{1-}+1*2^{2-}=19.75$

1+2+0+0+16+1/2+1/4=19.75

يمكنك تجريب هذه العملية باستخدامك للآلة الحاسبة البرمجية

نظام العد الستة عشري:

أحد أنظمة العد الشهيرة الخاصة بأنظمة الحاسب الآلي والتي تستخدم في عنونة الذاكرة وسمي هذا النظام بهذا الاسم لأنه يستخدم ستة عشر رقما ومجال هذه الأرقام هي

٠-١-٢-٣-٤-٥-٦-٧-٨-٩-F-B-C-D-E-F-

يعتبر هذا النوع من أنظمة العد ذو أهمية كبيرة في مجال البرمجة العكسية حيث تتعامل معظم أدوات تحليل البرامج وتنقيحها مع النظام الستة عشري كبرنامج ollydbg وبرنامج Hex Editor وبقية الأدوات الأخرى بكل الأحوال يتم مقابلة الأعداد الستة عشرية بمكافئتها في النظام العشري على الشكل التالي

النظام العشري	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
النظام الستة عشري	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

يرمز للنظام العشري بحرف h يوضع في نهاية كل رقم ويأخذ شكل الهوية التي تثبت شكل النظام المعمول به في فقرة الكود البرمجي ويمكن اجراء العمليات الحسابية واستدعاء المقاطعات من خلاله بشكل مباشر ومثال على ذلك يمكن كتابة الرقم ٦٨٩٩ بالنظام الستة عشري على نحو 2AF3h الآن سننتقل إلى عملية التحويل بين النظام العشري والنظام الستة عشري

التحويل من النظام العشري إلى النظام الستة عشري

تتم عملية التحويل بين هذاين النظامين من خلال ناتج عملية قسمة العدد المحول على ١٦ بعد تجزئته فمثلا لدينا الرقم ٤٨ بالنظام العشري وأردنا تحويله فنقوم بعمل التالي

٣=١٦\٤٨

٠=١٦\٣

وبتجميع الرقمين فيكون الرقم ٣٠ هو ناتج عملية التحويل إلى النظام الستة عشري

التحويل من النظام السداسي عشر إلى العشري

للتحويل من النظام السداسي عشر إلى العشري نستعمل قانون التمثيل الموضعي للأعداد مع مراعاة أن أساس هذا النظام هو ١٦.

مثال تحويل العدد AF3٢

$$
\begin{aligned}
&(2AF3)_H \\
&= 2\times16^3 + A\times16^2 + F\times16^1 + 3\times16^0 \\
&= 8192 + 10\times16^2 + 15\times16^1 + 3 \\
&= 8192 + 2560 + 240 + 3 \\
&= (6899)_D
\end{aligned}
$$

مثال آخر لكن بفاصلة حيث سنقوم بتحويل العدد 0.2A

$$
\begin{aligned}
&(0.3A)_H \\
&= 3\times16^{-1} + A\times16^{-2} \\
&= \frac{3}{16} + \frac{10}{256} \\
&= 0.1875 + 0.0390625 \\
&= (0.2265625)_D
\end{aligned}
$$

بينية الحاسب الآلي:

يتداخل عمل الحاسوب بشكل مباشر مع الهندسة العكسية كونه يعتبر المعالج الأساسي للبيانات التي تعتمد عليها التطبيقات لذلك كان لابد من إجراء بحث شامل عن كيفية عمل المعالج وكيف تتم عملية إنجاز البيانات داخله

تعريف الحاسب الآلي

يُعرَّف الحاسب الآلي بأنه نظام إلكتروني لمعالجة البيانات، ويتألف من قسمين أساسيين:

القسم الأول: البرمجيات (Software):

وتقسم إلى قسمين نظم التشغيل والتطبيقات الفرعية وهي تعتبر عنصر أساسي في التعامل مع الحاسوب وقد شهد هذا الحقل تطوراً كبيراً في السنوات الأخيرة والسبب هو تطور لغات البرمجة التي أصبحت أقرب إلى لغة الإنسان بعد أن كان التعامل مع اللغات الأدنى عملية صعبة جداً ويحتاج إلى تفرغ كبير لتعلم حتى مبادئ العمليات المنطقية التي يقوم بها المعالج ويعتبر مجال تصميم التطبيقات مربح جداً لمريديه بشتى أنواعها كتطبيقات الحاسب والهاتف الذكي والجولات إضافة إلى التطور الكبير الحاصل في نظم التشغيل حول العالم

القسم الثاني: الكيان الصلب (Hardware):

ويمثل الكيان الصلب كل ما هو ملموس بدءً من الفأرة إلى الشاشة إلى المعالج والذاكرة الخ.... ويشكل توليفة ضرورية مع التطبيقات حيث لا يمكن الاستغناء عن الاثنين بكل الأحوال فإن للعتاد الصلب أربعة أقسام وهي:

١-وحدة الدخل: تتم من خلالها إدخال المعطيات الرقمية.

٢-وحدة الخرج: تتم من خلالها إظهار النتائج بعد معالجة المعطيات.

٣-وحدة المعالجة المركزية: هي المسؤولة عن العمليات الحسابية والمنطقية ومعالجة البيانات.

٤-وحدة الذاكرة: تخزن البرامج والمعطيات.

أجهزة الإدخال والإخراج (Input-Output):

يعرف بأنه الكيان الخارجي للحاسوب أو العالم الخارجي وتشرف على التحكم بها أربعة أجزاء هامة وهي:

١-أجهزة ذات إشارات دخل منطقي و تمثل هذه الأجهزة الحساس ، العداد ،.....الخ

٢-أجهزة ذات إشارات دخل نظري وتمثل تيار ، جهد ، ضغط ، حرارة .....الخ

٣-أجهزة ذات إشارة خرج نظري وتمثل محرك ،تيار ، جهد .....الخ

٤-أجهزة الربط بالإنسان مثل لوحة المفاتيح ، الفأرة ، الطابعة .....الخ

ما هو المعالج:

المعالج هو عبارة عن قطعة مربعة الشكل هذا ما يبدو للمرة الأولى لكن هذ التعريف لا يتجاوز كونه تعريفاً تجريدياً فقط لا غير المعالج هو عقل الحاسوب ويحتوي هذ العقل على ملايين الترانزستورات وهذا أشبه بتلافيف الدماغ تمارس هذه الترانزستورات وظائف منطقية ورياضية مختلفة طبعاً كلما أزداد عدد الترانزستورات ازدادت سرعة

تعامل المعالج مع البيانات وهذا ما قد يساهم أحياناً في ارتفاع درجة حرارة المعالج في بعض المعالجات

تطور معماريات المعالجات :

تمارس وحدات المعالجة من جميع الأحجام (المتوسطة – الصغيرة – العملاقة) عملية نقل الشيفرات الثنائية عبر ممراتها في الطبقة الدنيا من مرحلة تنفيذ العمليات ويكمن فهم المعالج لهذه الشيفرات الثنائية من خلال الرقاقة الرئيسية والتي برمجت من قبل الشركة المصنعة على فهم هذه اللغة الثنائية وتسمى هذه الشيفرات بالتعليمات وتختلف هذه التعليمات من حيث كميتها والمعطيات التي تقدمها باختلاف نوع هذه الرقاقات أو المعالجات حيث تقوم الشركات بعمليات تحسين شبه دائمة لهذه المعالجات وسنلي ذكر أشهر أنواع المعالجات التي قدمتها شركة إنتل الأوسع انتشاراً حول العالم

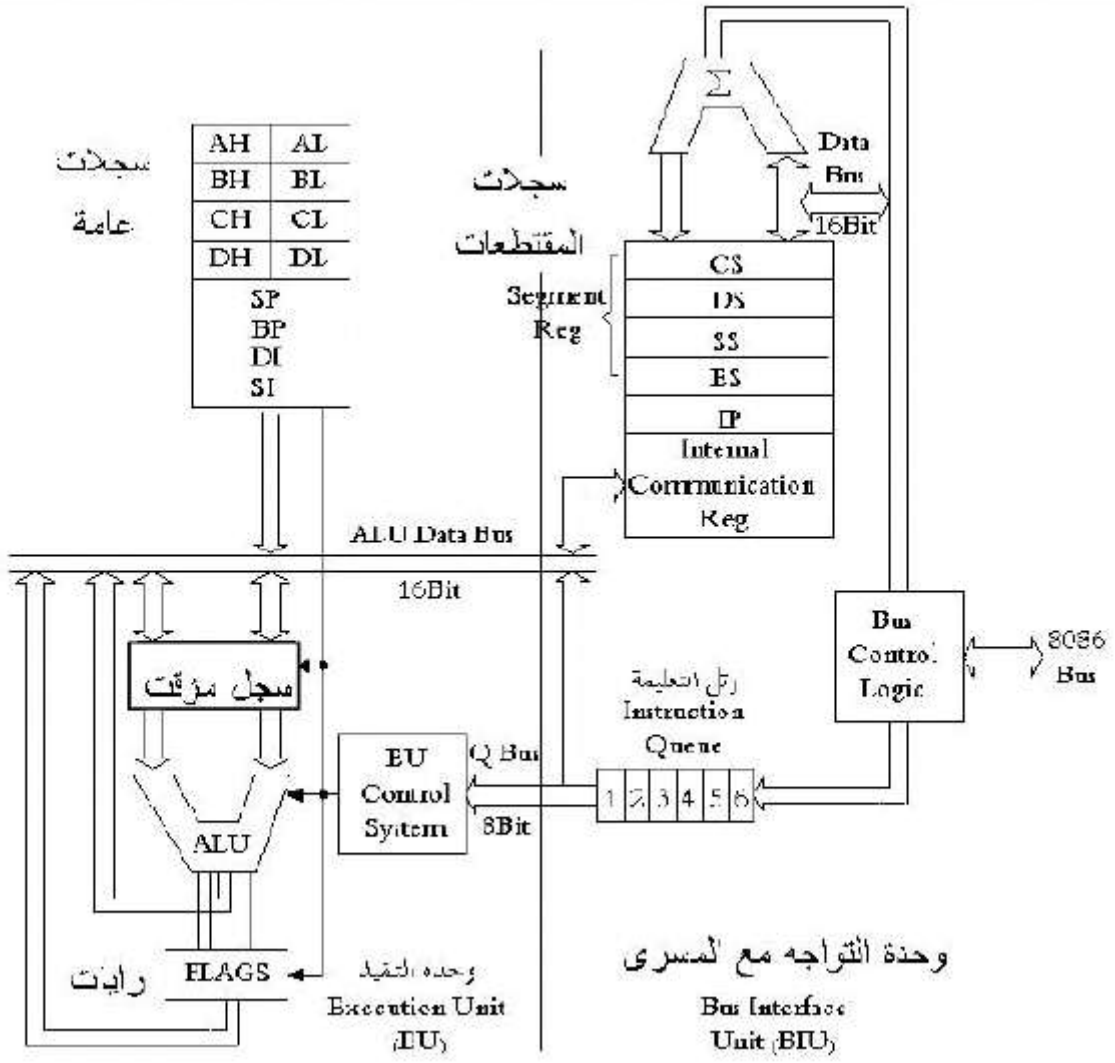
المعالج ذات المعمارية ٨٠٨٦-٨٠٨٨:

قامت شركة إنتل في عام ١٩٧٨ بطرح معالجات تدعم معمارية ٨٠٨٦ وهو معالج يتعامل مع كلمة بطول ١٦ بت أما المعمارية الأحدث فقد تم طرحها في المعمارية ٨٠٨٨ وهو مشابه للمعالج ٨٠٨٦ من ناحية التركيبة ويختلف عنه في طريقة التعامل الخارجي بطول ٨ بت أما معالج ٨٠٨٦ يكون الاستخدام بواسطة نبضة سريعة

فزيادة سرعة النبضة تعني تلقائياً زيادة سرعة التردد تقيم سرعة تردد المعالج من خلال اختبار السرعة القصوى له إي عندما أقول بأن معالج من شركة إنتل تبلغ سرعته ٣٠٠ميغاهيرتز فأعلم أنها السرعة النهائية للمعالج وليست السرعة الثابتة ، قامت

شركت IBM بتبني معمارية ٨٠٨٨ لسببين الأول رخص تكلفته مقارن بالأجيال السابقة والثانية لسهولة التعامل معه مقارنة بالمعالجات السابقة

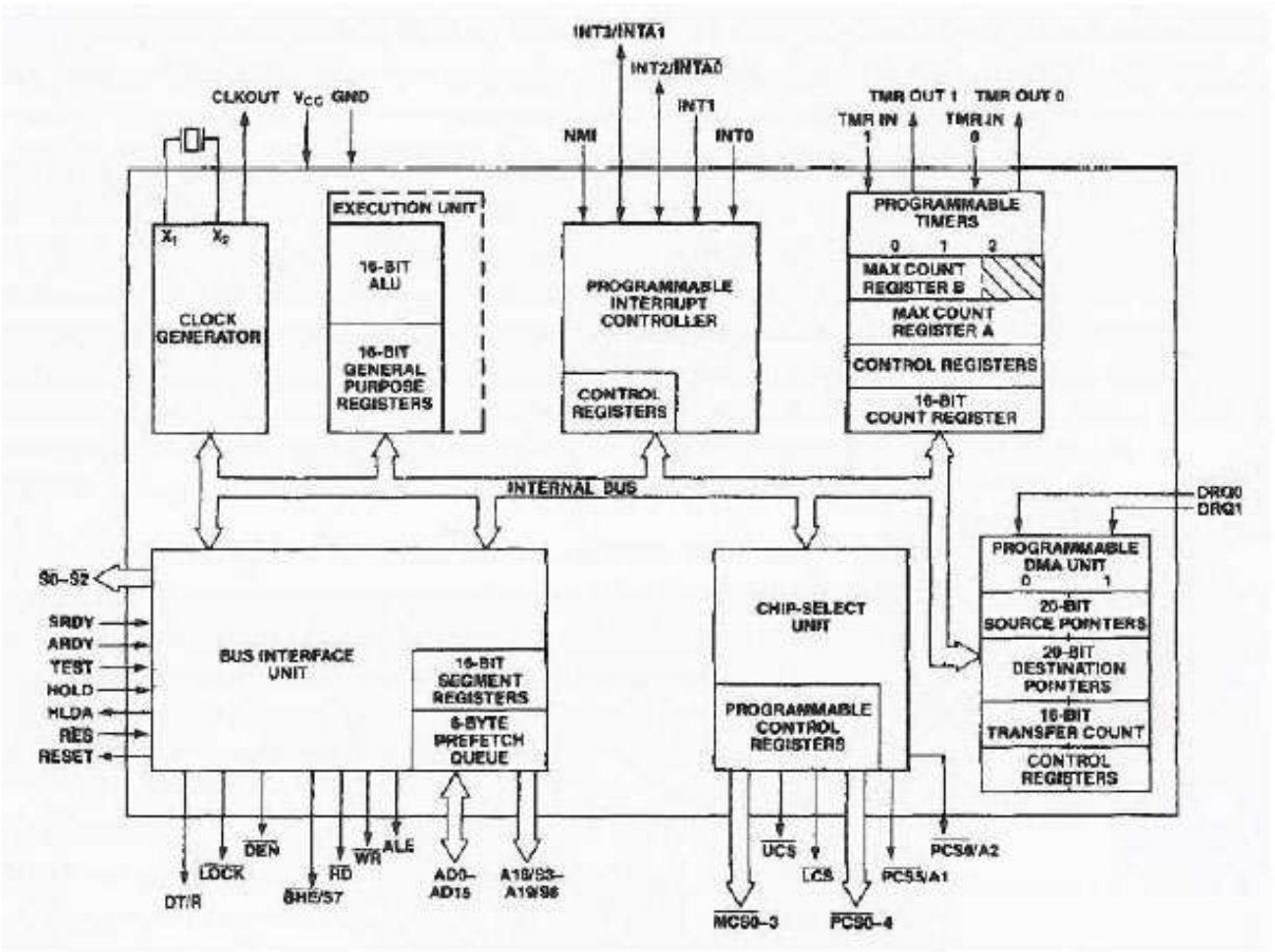
المعالجان السابقان يتعاملان مع نفس التعليمات ويعتبران أساس الثورة الكبيرة التي حدثت فيما بعد في عالم الرقاقات الصغيرة وبالتالي نهضة موازية بالحاسبات الشخصية فلا تخلو المعالجات الحديثة من دمج هذه المعمارية داخل رقاقتها مما يجعل هذه الرقاقات تدعم التطبيقات القديمة التي صممت لتناسب المنصتين القديمتين من إنتل



المعالجان ذات المعمارية ٨٠١٨٦-٨٠١٨٨

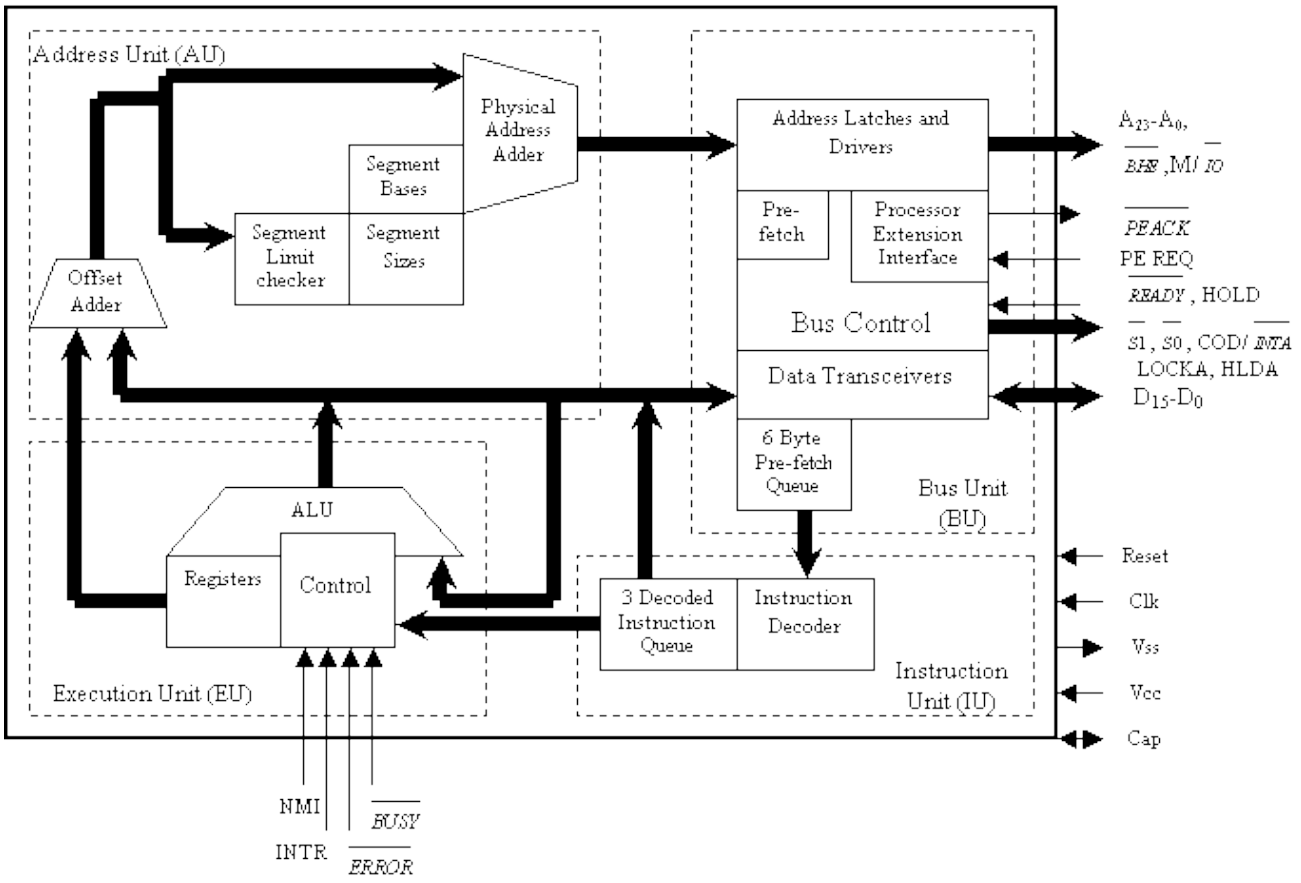
هذان المعالجان هما تطوير للمعالجان السابقان من شركة إنتل لا يحمل هذان المعالجان شيء جديد بالنسبة للجيل السابق سو بدعمه حزمة تعليمات موسعة مخصصة للعمليات المنطقية والرياضية لا يعني ذلك إنه حقق سرعات كبيرة عن نظيراتها السابقة بكل

الأحوال طرح شركة إنتل لمعالج ٨٠٢٨٦ في الأسواق عجل بنهاية سريعة للمعالجان ٨٠١٨٦-٨٠١٨٨



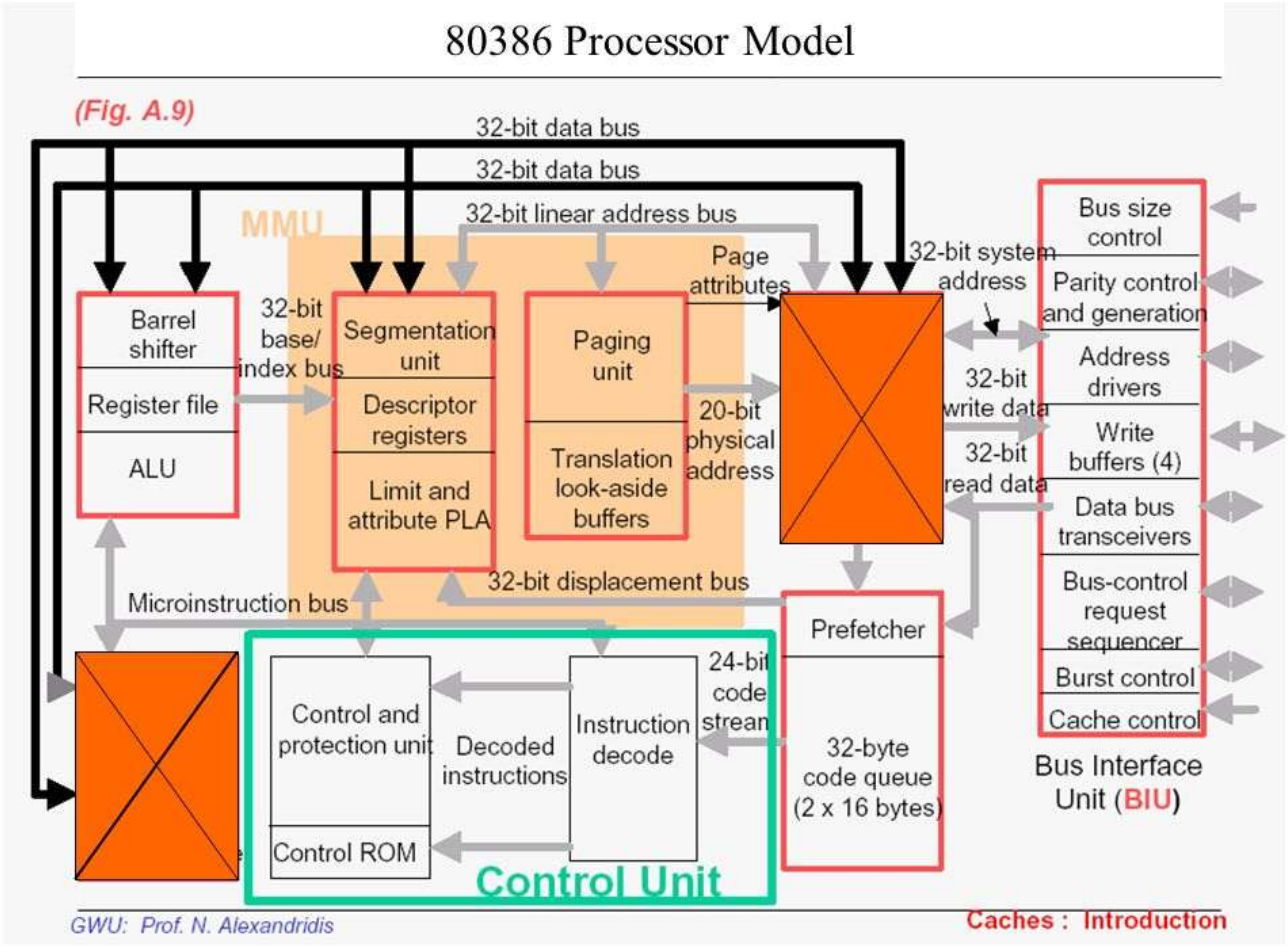
المعالج ذو المعمارية ٨٠٢٨٦:

يعتبر هذا المعالج نقلة كبيرة في عالم الرقاقات الصغيرة العجيب تم طرحه في عام ١٩٨٢ يتعامل مع منصات ١٦ بت ولكنه أسرع بكثير تصل سرعته إلى ١٢,٥ميغاهيرتز وهو أول معالج يعمل بنمط المهام المتعددة حيث يتم تشغيل أكثر من تطبيق عليه في نفس الوقت وهذا ما يسمى بميزة النمط المحمي أما بالنسبة للنمط الحقيقي وهي إحدى المميزات المهمة كذلك وهو عمله كمعالج ذو المعمارية ٨٠٨٦ تماما مما أعطى توافقية كبيرة مع التطبيقات التي تدعم هذا النوع من المعالجات دون تدخل من الخارج للقيام بتعديلات تتوافق مع هذه المعالجات هذا النوع من المعالجات يمكنه التعامل مع ذاكرة يصل حجمها إلى ١٦ ميغابايت إضافة إلى قدرته في التعامل مع ذاكرة افتراضية يصل حجمه إلى ١ غيغابايت



المعالج ذو المعمارية ٨٠٣٨٦:

تم طرح هذا المعالج في عام ١٩٨٥ الجديد الذي قدمته إنتل في هذا المعالج هو دعمه لمنصة ٣٢ بت مما جعله يتفوق بشكل كبير على المعالج السابق لنفس الشركة وذلك بسبب مضافة طول الكلمة لتصل إلى ٣٢ بت بكل الأحوال فإن الميزات التي جاء بها هذا المعالج هي نفسها الموجودة في المعالج السابق ٨٠٢٨٦ كالنمط الحقيقي ودعم ذاكرة افتراضية وميزة النمط المحمي إضافة إلى توافقه مع حجم ذاكرة أساسية يصل إلى ٤ غيغابايت



المعالج ذو المعمارية ٨٠٤٨٦:

طرح هذا المعالج من قبل الشركة الأم سنة ١٩٨٩ وهي عبارة عن نسخة محدثة من المعالج السابق ٨٠٣٨٦ بنفس الميزات جاء هذا المعالج ببعض الإضافات البسيطة والهامة كذاكرة صغيرة الحجم مدمجة مع المعالج السابق تسمى (Memory Cash) بحجم ٨كيلوبايت الهدف منه استخدامها لنقل شيفرة البيانات من الذاكرة إلى المعالج وجاءت إنتل معها بحزمة تعليمات موسعة تساعد على القيام بعمليات رياضية ومنطقية بشكل أسرع وجعله كفأ في هذا النوع من التطبيقات بكل الأحوال فإن هذا المعالج قدمته شركة إنتل ليكون الأسرع بين زمرته في عالم المعالجات الصغيرة الحجم حيث تصل سرعته إلى ١٠٠ميغاهيرتز

كيف يعمل المعالج؟

يمارس المعالج أربع عمليات رئيسية وهي جلب البيانات وفك التشفير والتنفيذ وإعادة الكتابة ولفهم هذه العملية بشكل أوسع سنتطرق إلى عملية شرح لكيفية المعالجة

١-عملية جلب البيانات:

تتم هذه العملية من خلال جلب البيانات من الذاكرة حيث تأتي البيانات من الذاكرة نحو وحدة المعالجة المركزية فالبيانات تنتقل إلى الذاكرة ليتم عنونتها حتى لا تختلط الملايين من البيانات مع بعضها البعض وهذا أشبه بتسجيل أرشيف برقم خاص وعنونته في الديوان أو داخل دائرة السجلات حتى لا تضيع هذه السجلات في المكتبة يشرف على تنظيم هذه البيانات عداد الأوامر الذي يقوم بعملية إحصاء كمية البيانات المتدفقة نحو المعالج ثم يقوم المعالج بإرسال أمر لجلب التعليمة التالية) تقوم دائرة التحكم في الذاكرة بتوجيه المعالج إلى العنوان المناسب تقوم دائرة التحكم في الذاكرة بتوجيه المعالج لتحميل البيانات (Data) المتعلقة بالتعليمة في الخطوة السابقة ، حيث يتم تحميلها وحفظها في أحد المسجلات بالنسبة للمسجلات سنقوم بشرح وافي لها فيما بعد حيث أن المسجلات التي تعمل وفق منصة ٣٢بت تعمل على المنصة ٦٤بت

٢-عملية فك الشيفرة والتنفيذ:

تقوم وحدة فك التشفير (Instruction Decoding) بتحليل التعليمة الموجودة في مسجل التعليمات، وتحويلها إلى خطوة واحدة أو عدة خطوات من العمليات التي تقوم بها وحدة الحساب والمنطق. تقوم وحدة الحساب والمنطق بتنفيذ العمليات الحسابية أو المنطقية على البيانات الموجودة في المسجلات بحسب ما تطلبه التعليمة المحملة قد يكون من نتائج تنفيذ التعليمة السابقة حفظ بعض البيانات في الذاكرة، أو إدارة جهاز خارجي متصل بالحاسب. بعد استكمال تنفيذ جميع العمليات المتعلقة بالتعليمة السابقة، تقوم الدائرة التي تتحكم في ذاكرة العمليات بتوجيه المعالج إلى العنوان التالي المحفوظ فيها التعليمة التالية، وتتكرر الخطوات السابقة إلى أن يستكمل تنفيذ كافة البرنامج.

٣-عملية إعادة الكتابة:

وهي العملية التي يتم من خلالها إعطاء المستخدم نتائج المدخلات التي قام بطرحها على وحدة المعالجة المركزية ويتم تخزين النتيجة داخل الذاكرة ويتم أرشفتها من خلال عداد المعالج الذي يعطي هذه النتيجة رقم خاص به إي هي عملية لإعادة كتابة البيانات الخاصة بالمعطيات أو النتائج فوق البيانات التي قام المستخدم بإدخالها في الذاكرة وهي أخر مرحلة من مراحل عمل المعالج ولنفهم تفاصيل أكثر عما يحتويه المعالج من أجزاء قمنا بشرح مفصل وكامل بعد أن أتممنا شرح هذه الجزئية الهامة لذلك وجب التركيز على المعلومات التي في الأسفل فهي تمثل روح عمل المعالجات

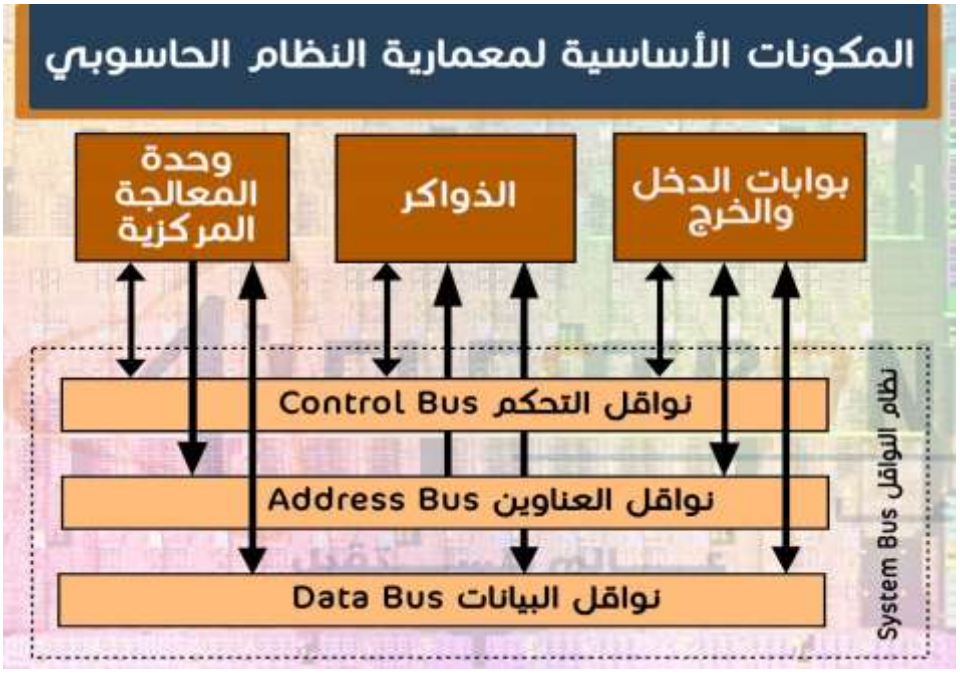
تعليمات المعالج:

تمارس وحدات المعالجة من جميع الأحجام (المتوسطة – الصغيرة – العملاقة) عملية نقل الشيفرات الثنائية عبر ممراتها في الطبقة الدنيا من مرحلة تنفيذ العمليات ويكمن فهم المعالج لهذه الشيفرات الثنائية من خلال الرقاقة الرئيسية والتي برمجت من قبل الشركة المصنعة على فهم هذه اللغة الثنائية وتسمى هذه الشيفرات بالتعليمات وتختلف هذه التعليمات من حيث كميتها والمعطيات التي تقدمها باختلاف نوع هذه الرقاقات أو المعالجات حيث تقوم الشركات بدمج هذه التعليمات بداخلها

معالجة شيفرة التعليمات:

عند عمل المعالج فإن شيفرة التعليمات يتم قراءتها من خلال الذاكرة التي يتم تخزين هذه التعليمات عليها كل تعليمة من هذه التعليمات تحتوي على بايت أو أكثر من المعلومات التي تأمر المعالج للقيام بتنفيذ وظيفة معينة

وكل تعليمة منها يتم تخزينه في الذاكرة وقراءته في نفس الوقت عند حاجة البيانات له وللعلم فإن البايتات الموجودة في الذاكرة والتي تحوي شيفرة التعليمات هي نفسها الموجودة في المعالج وتقوم مؤشرات خاصة بوظيفة مهمة وهي عملية جعل المعالج يحافظ على مسار البيانات عند اتجاهها نحو الذاكرة حيث تخزن شيفرة التعليمات وهذا ما يسمى بعملية الإشراف والتنظيم ويقوم به على وجه الخصوص مؤشر التعليمات ويقوم بتحديد الرمز التالي الذي سيتم معالجته فور انتقال الرمز الذي تم معالجته من الذاكرة

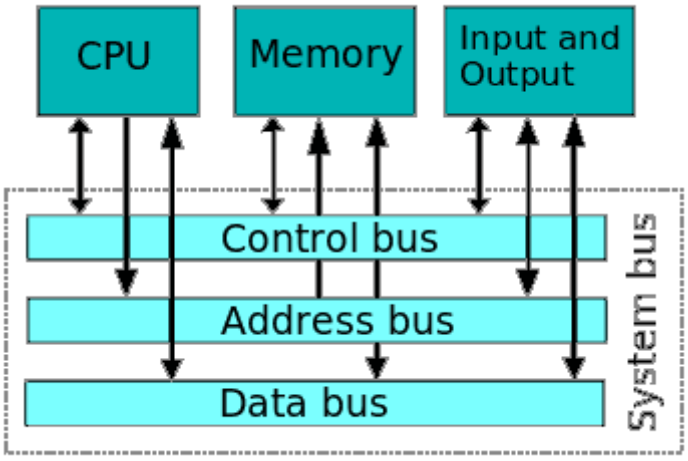


وبالطبع فهناك أوامر خاصة تقوم بنقل مؤشر التعليمات إلى موقع أخر كليا كعملية القفز إلى موقع محدد داخل البرنامج أما بالنسبة إلى مؤشر البيانات فهو يقوم بعملية الحفاظ على مسار المعالج في منطقة البيانات في بداية الذاكرة وتسمى هذه المنطقة بالمكدس ففي حالة وصول بيانات جديدة إلى الذاكرة فإن المؤشر يتجه نحو القسم السفلي من الذاكرة وفي حالة قراءة هذه البيانات فإنها تقوم بنقل المؤشر إلى الأعلى بعيدا عن المكدس

مسارات المعالجة:

يعتبر مسار النظام صلة وصل ما بين المعالج ومتحكم الذاكرة الأساسية وتقوم هذه المسارات بنقل البيانات بين أقسام الحاسب المختلفة مسار المعالج يقسم إلى ثلاثة مسارات وهي

- مسار البيانات (Data Bus)
- مسار العناوين (Address Bus)
- مسار التحكم (Control Bus)



مسار البيانات (Data Bus) :

مسار البيانات هو عبارة عن خطوط كل خط يمثل بت واحد وعندما يكون هنالك ٣٢ خط فإن مسار البيانات يكون بطول ٣٢ بت ويستخدم المسار في نقل البيانات من وحدة التحكم إلى متحكم الذاكرة والذي يتواجد داخل أحد الرقاقات على اللوحة الأم وتدعى الجسر الشمالي وبسبب أن حجم المسارات ثابت فإنه يتطلب معالجة خاصة عند إرسال بيانات بطول أقل من طول المسارات المسؤولة عن نقل البيانات وللعلم في حالة عدم استخدام بيانات بطول أقل من طول مسار البيانات فإن المعالج يقوم بإضافة أصفار في

الخطوط الغير مستخدمة وفي حالة كانت أطول فإن عملية النقل تتم على مراحل كل واحدة من هذه المراحل ترسل ٣٢ بت من البيانات

مسار العناوين (Address Bus):

يستخدم مسار العناوين في نقل عنوان الذاكرة المراد استخدامه سواءً للقراءة منه أو الكتابة عليه ويحدد حجم مسار العناوين أكبر عنوان يمكن الوصل إليه في الذاكرة وبالتالي يحدد حجم الذاكرة التي يستطيع الحاسب التعامل معها وفي الاجهزة التي تستخدم معالجات انتل ٨٠٨٦ كان حجم المسار هو ٢٠ بت وبالتالي فأن أقصى ذاكرة يتعامل معها المعالج هي ١ ميجا أما اطقم المعالجات ٨٠٣٨٦ فإن حجم المسار فيها هو ٢٤ بت وفي المعالجات التي يليها تم زيادة الحجم إلى ٣٢ بت وبالتالي يمكن تنصيب ذاكرة بحجم ٤ جيجا

مسار التحكم (Control Bus):

يستخدم مسار التحكم في إرسال الأوامر مثل القراءة من العنوان الموجود على مسار العناوين أو أوامر الكتابة على العنوان المطلوب ويتألف هذا المسار من عدد الخطوط وكل خط بت يؤدي وظيفة محددة أحد هذه الخطوط هو خط الكتابة (write) والذي يعني أن العنوان الموجود على خط العناوين يجب أن تعين له القيمة الموجودة في مسار البيانات الخط الآخر هو خط القراءة Read والذي يدل على أن العنوان الموجود في مسار العناوين يجب أن تقرأ قيمته إلى مسار البيانات

آخر خط يهمنا هو خط الولوج والذي يحدد ما إذا كان العنوان موجه إلى متحكم الذاكرة أم إلى متحكم الإدخال والإخراج وفي حالة كانت قيمة هذا الخط هي القيمة ١ فأن هذا يعني أن العنوان موجه إلى متحكم أجهزة الإدخال والاخراج وبالتالي سيتم القراءة من هذا العنوان أو الكتابة إليه وذلك بحسب قيمة الخطين القراءة والكتابة

المسجلات (Register):

إن أهم وظيفة للمعالج هو التعامل مع البيانات ومنها تلك البيانات التي تأتي من الذاكرة ولذلك ولسوء الحظ فإن عملية انتقال البيانات إلى الذاكرة تتم على دفعتين هي التخزين والقراءة حيث يقوم بترحيل البيانات مستخدما ممرات تسمى ممرات التحكم (control Bus) وتدخل إلى وحدة تخزين الذاكرة (Memory Storage) وهذا يؤدي إلى بطئ في عملية انتقال البيانات فوضعية المعالج في هذه الحالة تتوقف على انتظار البيانات ريثما يتم معالجتها من قبل الذاكرة وانتقالها إلى وحدة المعالجة ومن هنا

تأتي أهمية المسجلات وهي مواقع لذاكرة داخلية مدمجة مع رقاقة المعالج قادرة على تخزين عناصر البيانات لمعالجتها بدون الدخول إلى وحدة التخزين في الذاكرة ويذكر بأن الجانب السلبي الوحيد للمسجلات هو العدد المحدود لها ولكن لا مشكلة مادامت الشركة أجبرت المعالج على قتل الوقت الضائع بالقوة من خلال هذه المسجلات وذلك بقراءة البيانات في هذا الوقت الفارغ

أنواع المسجلات:

المعالجات ذات المعمارية ٣٢ بت تمتلك مجموعة متعددة من المسجلات وتختلف المسجلات بطبيعة الحال بين معالجات ٣٢ بت وتلك الموجدة في أطقم تعليمات ١٦ بت وبكل الأحوال فإنه كما تعودنا مع شركة إنتل فالمعالجات المصممة بنمط ٦٤ بت تدعم المعماريات الأدنى منها (٣٢-١٦بت) أما العكس فهاذا لا يجوز طبعاً بكل الأحوال فلنتعرف على هذه المسجلات:

١- مسجلات المقطع (Segment) وعدد هذه المسجلات ستة تستخدم لمعالجة البيانات من هذا النوع عند دخولها للذاكرة

٢- مسجلات الهدف العام (General Purpose) وعددها ٨ مسجلات تستخدم للتعامل مع البيانات التي تحدث الآن (تخزين بيانات من نوع ٣٢ بت)

٣- مؤشر التعليمات (Instruction Pointer) مؤشر وحيد من نوع ٣٢ بت يشير إلى رمز الأمر التالي

٤- بيانات مؤشر الفيض (Floating Point Data) عدد السجلات ٨ تستخدم هذه السجلات للتحكم بعملية حساب مؤشر الفيض

٥- التحكم (Control) ٥ مسجلات تستخدم لتقرير نمط تشغيل المعالج

٦- المنقح (Debug) وعددها ٨ مسجلات تحتوي على تعليمات خاصة تستعمل عند تنقيح عمليات المعالجة

مقياس قاعدة الفهرسة (The Sib Byte)

وهي اختصار لثلاثة حقول من المعلومات والتي تمثل الكلمات الثلاثة

١- المقياس Scale: يمثل هذا الحقل عامل المقياس لعملية المعالجة

٢- الفهرسة Index: أما هذا الحقل فيمثل مسجل الدليل وذلك من أجل الوصول إلى الذاكرة

٣- القاعدة Base: يحدد السجل المستخدم كقاعدة

كلاً من المجموعتين SIB والنمط RM يقومان بإنشاء قاعدة بيانات التي تعرف العديد من المجموعات المختلفة الاحتمالات في الذاكرة إضافة إلى السجلات الخاصة بصفحة مواصفات معالجات إنتل بنتيوم (Intel Pentium

شكل شيفرة التعليمات (Instruction Code Format):

المعمارية المستخدمة حالياً في جميع المعالجات الصغيرة الحجم تحوي كما قلنا سابقاً اطقم تعليمات ٣٢ بت وهذا موجود في جميع معالجات الجيل الحالي من (Intel-AMD) ومنها المعالجات الأحدث المتعددة الأنوية من انتل فهم هذه التعليمات له قيمة كبيرة في برمجة شيفرة خاصة بك وهي تتألف من أربع أجزاء هامة :

١- بادئة الأوامر الاختيارية

٢- الرمز العامل حالياً

٣- عنصر البيانات الاختيارية

٤- المُعدل الاختياري (المحرر)

كل جزء يستخدم لتعريف كامل لتعليمة محددة يتم معالجتها من قبل رقاقة المعالجة سيتم شرح هذه الأجزاء لاحقاً في الأقسام القادمة

شيفرة العمليات:

هذا الجزء هو مهم وإجباري على عكس الأجزاء البقية كل أمر يجب أن يتضمن معرف خاص يحدد الوظيفة الأساسية أو المهمة اللتان سيقوم المعالج بها المبرمج يمكن أن يستخدم شيفرة تعليمات إضافية وذلك للحصول على معلومات من مسجلات المعالج لتقرير نوع وشكل المعالج الذي يقوم بتشغيل البرامج إن هذا الامر أشبه بعملية كتابة اسمك على ورقة الامتحان فهي التي تعرف المدرس على علامة هذا الطالب ويتم تسجيلها على دفتر علامات الطلاب عند المدرس فيما بعد

Instruction Prefix

١- تحتوي البادئة من ١بايت إلى ٤ بايت التي تقوم بتعديل سلوك شيفرة العمليات (opcode)

٢- تقسم إلى أربعة أجزاء مهمة مستندة إلى وظيفة البادئة

٣- فقط بادئة من كل مجموعة يمكن أن تستعمل مرة واحدة لتعديل شيفرة العمليات (opcode)

١- مسجلات الهدف العام:

المسجلات العامة تستخدم لتخزين البيانات بشكل مؤقت على رقاقة المعالج نفسها كما شرحنا سابقاً ويتميز هذا النوع من المسجلات بأنها مسجلات تابعة للمنصة ٣٢-١٦-٨ بت وقد شهدت تطور كبير منذ ظهور المعالجات ذات المعمارية ٨٠٨٠ وكانت تستخدم مسجلات ذات منصة ٨ بت وبكل الأحوال فالمعالجات اليوم هي تدعم المنصات الأقدم أي أن المعالجات الأحدث من إنتل تدعم منصة ١٦- ٨ بت في مسجلاتها بالإضافة إلى دعمها الطبيعي لمسجلات من نوع ٣٢بت ونخص بالذكر تلك المعالجات التي تطرحها شركة إنتل في الأسواق وهي واسعة الانتشار وتستخدم معمارية ٨٠٨٦ وبكل الأحوال فبينما تستخدم المسجلات العامة للاحتفاظ بأي نوع من

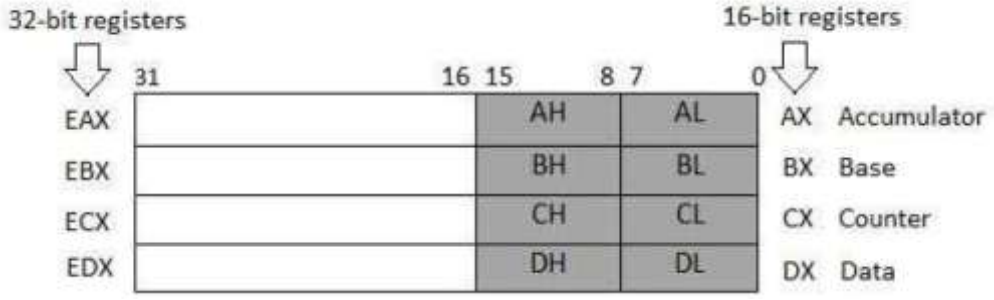
البيانات فإن لبعضها الأخر استخدامات خاصة وتستخدم بثبات في لغة التجميع (Assembly) سنقوم بشرح موجز من خلال الجدول الذي في الأسفل

وصف العملية	نوع المسجل
مؤشر المكدس يقوم بتخزين الموقع الحالي في المكدس	ESP
مؤشر بيانات المكدس	EBP
مصدر عمليات السلسة	ESI
مؤشر الإدخال والإخراج	EDX
الإشراف على عمليات التكرار والسلاسل	ECX
مؤشر البيانات في مقطع ذاكرة البيانات	EBX
مؤشر البيانات الذي يشرف على اتجاه عملية السلاسل	EDI
مسجل المراكم	EAX

المسجلات في الجدول السابق هي جميعها مسجلات من نوع ٣٢ بت وتعمل على منصات إنتل جميعا أي عند طلب هذه المسجلات لاستخدامها فإنه يمكنك الوصول لها مثلاً عن طريق المسجل EAX أما في حالة الوصول إلى أول ١٦ بت مثلاً فيتم ذلك عن طريق مسجل من هذا النوع وهو AX لاحظ أنا مسجلات الهدف يمكنها التخزين في مسجلات لعدة منصات مختلفة بالنسبة لمسجلات التابعة للمنصة ١٦ بت فهي مسجلات تقليدية ومعروفة وتقسم هذه المسجلات بطبيعتها إلى قسمين كل قسم يحجز ٨ بت في خانات المسجل من الأعلى إلى الأسفل وتمثل بالحرفين (L-H) لتكون مسجلين إضافيين من نوع ٨ بت وهذه المسجلات وأقصد هنا مسجلات ١٦بت عددها أربع مسجلات وهي كالتالي :

- المسجل AX (register Accumulator) ما يعرف بمسجل المراكم يقوم هذا المسجل بالإشراف على العمليات الرياضية والمنطقية ويعتبر مسجل أساسي في المعالجات الأقدم من إنتل حيث أن جميع العمليات من قسمة وضرب وجمع وطرح تصب فيه باستقبال أحد طرفي رقمي العملية الحسابية مع وضع قيمة هذه العملية في منطقة خارجية تحصل من خلالها على النتيجة المطلوبة يقابله المسجل EAX في منصة ٣٢ بت ويقسم المسجل إلى قسمين مسجل AL و مسجل AH
- المسجل BX (Base Register) مسجل القاعدة يستخدم لفهرسة العناوين الذاكرة تتغير قيمة هذا المؤشر عند القيام بسمح بيانات من على الذاكرة ويمثل هذا المسجل في منصة ٣٢ بت EBX ويقسم هذا المسجل إلى قسمين مسجل BH ومسجل BL
- المسجل CX (Counter Register) مسجل العداد ويقوم بالإشراف على عمليات التكرار ويأتي غالباً قبل عمليات القفز في لغة المجمع ويمثله المسجل ECX في معالجات ٣٢ بت ويقسم كذلك لقسمين المسجل CL والمسجل CH
- المسجل DX (Data Register) يشرف هذا المسجل على عمليات الدخل والخرج وتخزين البيانات والعمليات الرياضية ويمثله المسجل EDX في منصة المعالجات ٣٢ بت وهي على قسمين كالمسجلات الباقية المسجل DL والمسجل DH

الصورة التي سنقوم بعرضها تشمل فهم أوسع لهذا النوع من المسجلات أنظر في الأسفل



- المسجل EAX-EDX-EBX تستخدم هذه المسجلات الثلاثة لممارسة العديد من الوظائف الرياضية والمنطقية وتخزين عمليات الذاكرة

- المسجل EIP هو مسجل مسؤول عن مؤشر المكدس لوحدة المعالجة المركزية تخزن الموقع الحالي في المكدس لذلك أي شيء يتم دفعه إلى المكدس يصبح تحت هذا العنوان وهذا المسجل تتم بشكل منسق
- المسجل EPB يمكن استخدامه كمسجل عام وأيضاً يستخدم في معظم الأوقات كمؤشر بيانات المكدس عند اتحاد مؤشر القاعدة مع مؤشر المكدس ينتج عنه إطار المكدس، إطار المكدس يمكن تعريفه كوظيفة الحالية لمنطقة المكدس الذي يوجد بين مؤشر القاعدة ومؤشر المكدس ، مؤشر القاعدة يشير إلى موقع المكدس مباشرة بعد العنوان العائد من الوظيفة أما إطار المكدس فإنه يستخدم للوصول السريع والسهل إلى كل المتغيرات الحالية والبارمترات مروراً إلى الوظيفة الحالية
- مسجل ESI-EDI هو مسجل عام أيضاً يستخدم كثيراً كمؤشر المصدر/ الاتجاه في التعليمات التي تنسخ الذاكرة DI تمثل مراصف لدليل المصدر SI تمثل مراصف لدليل الاتجاه
- مسجل ECX يستخدم هذا المجل كعداد يتيح تكرار التعليمات البرمجية داخل التطبيق

٢- مسجلات المقطع:

وهي مسجلات تابعة لمنصة ١٦ بت وتستخدم هذه المسجلات للرجوع إلى مواقع الذاكرة ويتم الدخول عبرها إلى ذاكرة النظام بثلاثة طرق وهي

- عن طريق سطح الذاكرة Flat memory models
- عن طريق مقطع الذاكرة Segmented memory models
- عن طريق نمط العنوان الحقيقي Real addresses mode

في الطريقة الأولى وهي سطح الذاكرة يتم فيها تقديم ذاكرة النظام كمساحة لعنوان أوامر البيانات كما يحتوي المكدس على نفس هذه المساحة حيث يتم دخول هذه العناوين في منطقة محددة مواقع الذاكرة في المقطع يتم تعريفها بواسطة عنوان منطقي هذا

العنوان المنطقي يتألف من عنوان المقطع وعنوان الإزاحة المعالج يقوم بترجمة العنوان المنطقي إلى خطوط مراسلات التي تذهب من ناحية أخرى إلى مواقع الذاكرة لإدخال بايت في الذاكرة أما مسجلات المقطع فتستخدم حاويات عناوين المقطع من أجل إدخال بيانات محددة الجدول التالي يوضح هذه المسجلات:

الوصف	نوع المسجل
شيفرة المقطع	CS
بيانات المقطع	DS
مكدس المقطع	SS
مؤشرات مقطع إضافية	ES-FS-GS

المسجل CS يحتوي على مؤشر المقطع في الذاكرة أما شيفرة المقطع فتكون حيث تتم عملية الاحتفاظ بشيفرة التعليمات في الذاكرة يعتمد المعالج على هذا المسجل CS ليقوم بعملية استرجاع شيفرة التعليمات من الذاكرة بناءً على قيمته وقيمة مسجل مؤشر التعليمات EIP

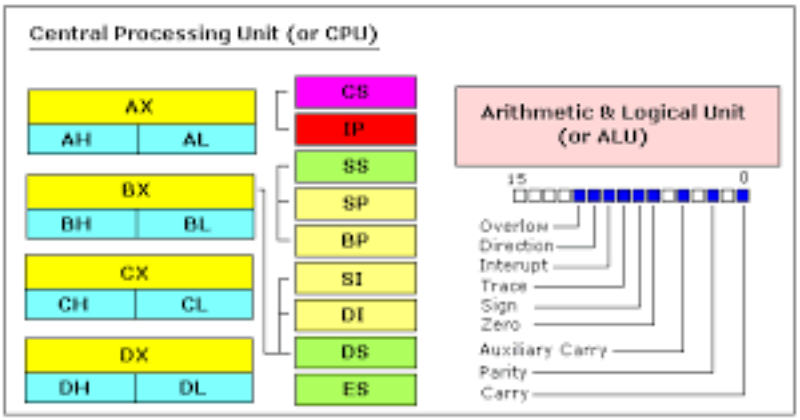
البرنامج لا يستطيع أن يحمل أو يجري أي تعديل في المسجل CS لذلك يقوم المعالج بتخصيص قيمة للبرنامج في مساحة محددة من الذاكرة

ملاحظة: مسجلات المقطع تعمل مع مسجلات الهدف العام للوصول إلى عنوان الذاكرة

بالنسبة للمسجل DS فهو يشير إلى مقطع البيانات وتقوم المسجلات الثلاثة الإضافية FS-GS-ES بعملية فصل لعناصر البيانات لضمان عدم تداخلها مع بعضها البعض البرنامج يقوم بتحميل مسجل مقطع البيانات بقيمة المؤشر الملائم للمقطع بالنسبة لمواقع الذاكرة القريبة فإنها تستخدم قيمة موازية

المسجل SS هو مسجل يشير إلى مقطع المكدس والذي يحتوي بدوره على قيمة البيانات المارة إلى الوظائف والإجراءات ضمن البرنامج

إذا كان البرنامج يستخدم وضعية العنوان الحقيقي تشير مسجلات المقطع إلى القيمة العددية صفر في العنوان ولن يحدث تغيير في البرنامج جميع عناصر البيانات وعناصر المكدس وشيفرة التعليمات تدخل مباشرة باستخدام العنونة الخطية



مسجل مؤشر التعليمات:

يرمز لهذا المسجل (EIP) ويقوم بالإشارة إلى التعليمة التالية التي سيتم تنفيذها البرنامج لا يستطيع القيام بعملية تعديل مباشرة لمؤشر التعليمات لا تستطيع تحديد عنوان الذاكرة وتضعه في سجل مؤشر البيانات بدلاً من ذلك عليك استخدام كود برمجي للتحكم بالتطبيق كعملية القفز (Jump) حيث نقوم بتعديل الأمر القادم ليتم قراءته في الذاكرة البديلة. باستخدام طريقة الذاكرة المسطحة مؤشر التعليمات يحتوي على عناوين خطية لمواقع الذاكرة من اجل تنفيذ شيفرة التعليمات التالية إذا قام التطبيق باستخدام نمط ذاكرة المقطع هنا مؤشر التعليمات يشير إلى عنوان ذاكرة منطقية

مسجلات التحكم:

هي عبارة عن خمسة مسجلات تستخدم لتقرير نمط تشغيل المعالج وخصائص المهمة المنفذة حالياً

القيم في مسجلات التحكم لا يمكن أن تدخل للمعالجة مباشرة لكن محتوى البيانات في مسجل التحكم يمكن أن ينتقل إلى مسجل الهدف العام

عندما تكون البيانات في مسجل الهدف العام البرنامج يقوم بعملية فحص الأعلام في المسجل ليقوم بعمل تقرير لحالة تشغيل المعالج أو المهمة المنفذة حالياً إذا حدث تغيير في قيمة العلم لمسجل التحكم سينعكس ذلك على البيانات الموجودة في مسجلات الهدف العام والمسجل ينقلها إلى مسجل التحكم يقوم مبرمجي الأنظمة بتعديل القيم في سجلات التحكم على عكس البرامج العادية بحيث لا يقوم هؤلاء المبرمجون بأي تعديل على

مداخل مسجلات التحكم على الرغم بأنهم قد يشككون أحيانا بقيم العلم لتقرير توافقية رقاقة المعالج المضيف (الرقاقة المسؤولة عن تشغيل التطبيق في الوقت الحالي)

حالة الأعلام:

تشير حالة الأعلام إلى نتائج عمليات رياضية يقوم بها المعالج

نوع العلم	القيمة	الاختصار
تحمل علم	٠	CF
تعادل علم	٢	RF
يعدل علم	٤	AF
العلم صفر	٦	ZF
اشارة العلم	٧	SF
علم الفيض	١١	OF

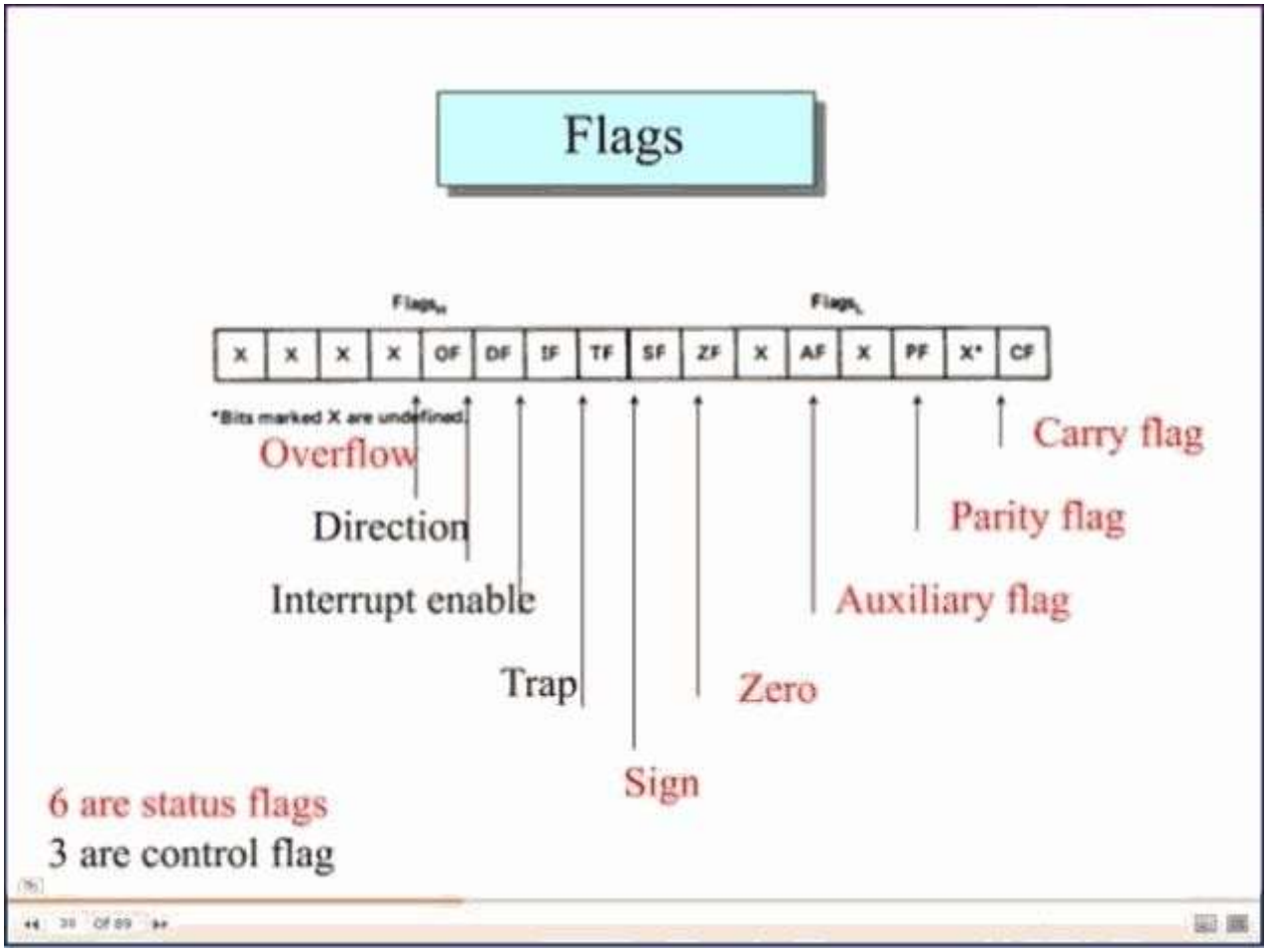
لكل عملية يتم إنجازها من قبل وحدة المعالجة يجب أن يكون هناك آلية متبعة لتقرير إذا ما كانت هذه العملية ناجحة أو لا الأعلام هي التي اعتادت على تقديم هذا النوع من التقارير أدائها لهذه الوظيفة جعلها مهمة في لغة التجميع وأساس مهم يعتمد عليه المبرمجون في الهندسة العكسية على سبيل المثال إذا قام تطبيق معين بعملية طرح على نتيجة معينة فإن قيمتها ربما تكون سلبية أحد الأعلام المخصصة في هذا الأمر ستقوم بالإشارة إلى هذه النتيجة مباشرة بدون إجراء عملية فحص وهذه الطريقة الوحيدة التي يستفيد منها المجمع ليتعرف المبرمج على نتيجة هذه العملية

الإعلام:

منصة معالجات ٣٢ بت تستخدم مسجل وحيد لتحتوي على مجموعة من الرايات أيضاً من نوع ٣٢ بت وهي

- علم الحالة Status Flag
- علم التحكم Control Flag
- علم النظام System Flag

تحتوي هذه الأعلام على قيم معينة داخلها والتي تمثل عملية قام بها التطبيق ولم يعرف نتيجتاها بعد إلا من خلالها وبكل الأحوال فإن هذه الرايات تحجز عدة بايتات من أجل استخدامها في المستقبل



أعلام التحكم (Control Flags)

تستخدم للسيطرة على سلوك معين في المعالج المعالجات الحالية تحتوي على علم واحدة فقط تتبع لهذا العنوان وهي علم الاتجاه ويرمز لها (DF) وتستخدم للتحكم بطرق السلاسل التي يتم إداراتها من قبل المعالج

عندما تكون إشارة DF=1 فإن سلاسل التعليمات تقوم بعملية إنقاص عناوين الذاكرة لتحتوي على البايت القادم في السلسة والعكس صحيح فعندما تشير علم الاتجاه إلى الصفر فإن عناوين الذاكرة تزداد تلقائياً للحصول على البايت التالي في السلسة

رايات النظام (System Flag)

تستخدم هذه الرايات في التحكم بنظام التشغيل ومستوى العمليات ويقوم المبرمجون عادة بتجنب العبث بهذه الرايات لما تسببه من ضرر كبير في بنية النظام وأحيانا تسبب إعادة إقلاع الجهاز مباشرة وعدد هذه الرايات عشرة:

نوع العلم	الاختصار
علم الخطوة الوحيد	TF
لتفعيل المقاطع	IF
تشرف على عملية الإدخال والإخراج	IOPL
الإشراف على مهمة التداخل	NI
علم الاستمرار	RF
علم النمط الافتراضي	VM
علم مراقبة التخطيط	AC
علم المقاطعات المعلقة	VIP
علم المقاطعات الافتراضية	VIF
التعريفات	ID

١- علم الخطوة الواحدة :Trap Flag

يساعد هذه العلم على تفعيل نمط الخطوة الواحدة حيث يقوم المعالج بإنجاز رمز واحد في نفس الوقت والانتظار من أجل معالجة الأمر التالي وهذه الميزة مفيدة جداً في لغة التجميع حيث يقوم المبرمجون بمتابعة العملية الحالية التي ينفذها البرنامج في المنقح

للبحث عن معرفة ما أو لتصحيح خطأ معين داخل البرنامج وسيلي ذكرها في تطبيقاتنا العملية

٢- علم المقاطعة Interrupted Flag:

يشرف هذا العلم على عمل العتاد الصلب من خلال استقبال مؤشر يدل على دخول جهاز جديد حيز التنفيذ يستخدم هذا العلم في عملية تصميم أنظمة التشغيل بشكل أساسي وخاصة في عملية برمجة المحمل يستخدم هذا العلم رقمين منطقيين هما الصفر والواحد وذلك للإشارة على إمكانية تفعيل هذه المقاطعة أو تعطيلها

٣- علم الإدخال والإخراج I\O Flag:

تشرف هذه العلم على عملية الوصول إلى امتيازات عناوين الإدخال والإخراج وتقوم بالإشارة إلى المهمة الجارية حالياً

٤- علم التداخل

تشير إلى المهمة المنفذة حالياً والتي ترتبط كلياً بالمهمة المنفذة سابقاً

٥- علم النمط الافتراضي

وقد قمنا بشرحها أكثر من مرة ففي النمط الافتراضي يعمل المعالج بمعمارية المعالجات ٨٠٨٦ ليتيح له الاستفادة من الميزات التي تتحها هذه المعمارية ومنها تشغيل التطبيقات التي تعمل على هذا النوع من المعالجات

حزمة التعليمات الموسعة في معالجات إنتل:

تمتلك معالجات إنتل بنتيوم العديد من الميزات داخل رقاقاتها وهذه الميزات تسمى بالحزمة الموسعة وتهدف إلى تسريع التعامل مع المعطيات أثناء معالجتها ظهرت هذه الحزم جلياً في معالجات ٨٠٣٨٦ والتي تدعم منصة ٣٢بت

١- FPU وحدة الفاصلة العائمة

المعالجات المتقدمة التي تدعم منصة ٣٢ بت من عائلة إنتل تدعم هذه الميزة والتي تقوم على فصل رقاقة المعالجة من أجل إتمام عملية رياضية خاصة بعملية وحدة الفاصلة العائمة وقد زودت شركة إنتل المعالجات ذات المعمارية ٨٠٢٨٧ ومعالجات ٨٠٣٨٧ بهذا النوع من الميزات وذلك بسبب حاجة المبرمجين لهذه الخاصية في ذلك الوقت

حيث تقوم أيضاً بإعطاء قوة ودعم كبير في تعامله مع العتاد الصلب ربما هي عملية لإشباع رغبة المبرمجين حول العالم للاستفادة من هذه الميزات لتحقيق مكاسب أكبر في عالم التطبيقات والعتاد الصلب معاً

وتأتي هذه الوظائف مدمجة داخل رقاقة المعالج لتقديم دعم كافي للوظائف على شكل شيفرة تعليمات الإضافية ويستفاد منها عندما تطلبها وحدة التنفيذ أو حتى المسجلات

تمكن مسجلات FPU وشيفرة التعليمات من من معالجات وظائف وعمليات معقدة جداً بسرعة كبيرة على سبيل المثال البرامج التي تعتمد على الجرافيكس والتطبيقات الخاصة بمدراء الأعمال حيث أجريت العديد من الدراسات على هذه الميزة والتي أضفت في النهاية إلى بأن معالج يحتوي على هذه الميزة أسرع من تلك المعالجات التي تفتقد لهذا النوع من التقنية

وظيفة المسجل	نوع المسجل
مسجلات ٨٠ بت من أجل بيانات نقطة الطوفان	مسجلات البيانات
مسجلات ١٦ بت لإعطاء تقرير عن حالة FPU	مسجلات الحالة
مسجلات ١٦ بت للتحكم في دفة FPU	مسجلات التحكم
مسجل ٤٨ بت يعود إلى العملية التالية التي يتم تنفيذها	مسجل مؤشر التعليمات
مسجل ٤٨ بت يشير إلى البيانات في الذاكرة	مسجل مؤشر البيانات
مسجل ١١ بت للتحكم بالتعليمة الأخيرة في عملية المعالجة بواسطة FPU	مؤشر شيفرة العمليات
مسجلات ١٦ بت لوصف محتوى مسجلات البيانات الثماني	مسجلات العلامة

وحدة ملائمة الممرات Bus Interface Unit:

ويرمز لها BIU تستخدم هذه الوحدة لإحداث توافقية ما بين المعالج والعالم الخارجي وتقوم هذه الوحدة بعملية الإشراف على ممر البيانات وممر التحكم وممر العناوين سنقوم بشرح مبسط لهذه الممرات فيما بعد، تحضر التعليمات من الذاكرة كل بايت على حدا وتضعها فيما يسمى بصف التعليمات الذي يتسع لست بايتات كحد أقصى ومن الطبيعي أن التعليمة التي تدخل صف التعليمات أولاً يتم تنفيذها أولاً للمحافظة على ترتيب التعليمات ويدعى هذا المبدأ بـالداخل أولاً خارج أولاً First In Last Out ونرمز لهذا المبدأ بـ FIFO.

إن إحضار شيفرة التعليمة التالية يتم عندما تكون وحدة التنفيذ EU مشغولة بتنفيذ التعليمة الحالية وهذه الميزة موجودة فقط في المعالجات ذات المعمارية ٨٠٨٦ وما فوق أما المعالجات الأقدم فتتوقف عن العمل ريثما يتم تنفيذ التعليمة الحالية

عندما تفك وحدة التنفيذ EU شيفرة تعليمة ما من صف التعليمات وتكون هذه التعليمة تعليمة تؤدي إلى تغيير تسلسل تعليمات البرنامج (قفز إلى برنامج فرعي مثلاً) عندها يتم تصفير صف التعليمات وإعادة ملئه من جديد بتعليمات البرنامج الفرعي (لأن وحدة ملائمة الممرات BIU تجلب التعليمات دون معرفة ما تؤديه هذه التعليمات)

تقسم وحدة ملائمة الممرات إلى أربع أجزاء وهي:

١. جامع العناوين
٢. مسجلات المقاطع
٣. وحدة التحكم بالمحرف
٤. صف التعليمات.

وحدة التنفيذ Execution Unit

وهي مسؤولة عن فك شيفرة التعليمات وتنفيذها وتتألف من:

١. وحدة الحساب والمنطق.
٢. مسجل الأعلام.
٣. ثمانية مسجلات للأغراض العامة.
٤. مسجلات مؤقتة.

٥. منطق التحكم بـ EU.

تجلب وحدة التنفيذ EU التعليمات من مقدمة صف التعليمات في وحدة ملائمة الممرات BIU وتفك شيفرتها وتقوم بالعمل الذي تمليه كل تعليمة فإذا احتاجت هذه الوحدة ((EU إلى معلومة مخزنة في الذاكرة فإنها تأمر وحدة ملائمة الممرات BIU بإحضارها و ذلك عن طريق إعطائها عنوان هذه المعلومة في الذاكرة.

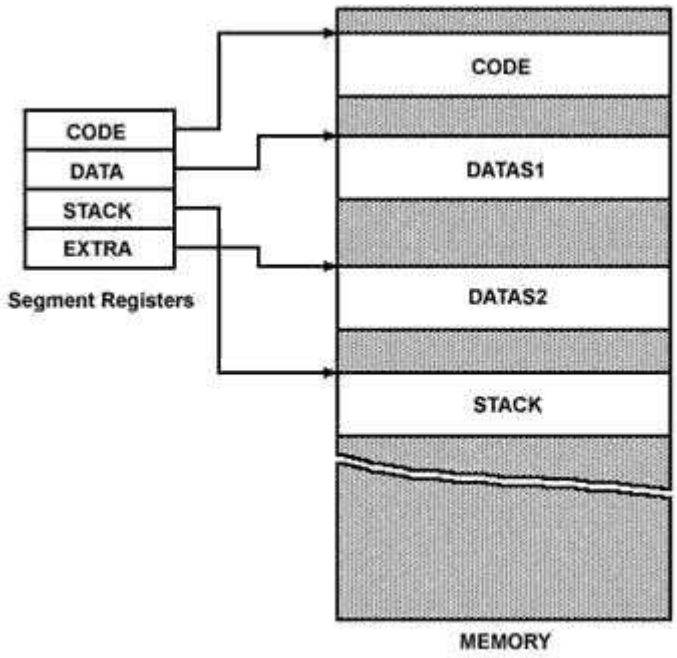
إن من أحد أهم وظائف EU هو تنفيذ العمليات الحسابية والمنطقية على المعلومات، وأثناء سير التنفيذ تقوم EU بفحص مسجل الأعلام بعد كل تعليمة

بنية الذاكرة:

تتألف الذاكرة من حجرات متسلسلة سعة كل منها ٨ بت (واحد بايت)، ترقم هذه الحجرات من الصفر وحتى نهاية الذاكرة ويستخدم النظام الست عشري عادة في عملية الترقيم وبذلك يكون لكل حجرة رقم يميزها عن غيرها، يدعى هذا الرقم بعنوان تلك الحجرة.

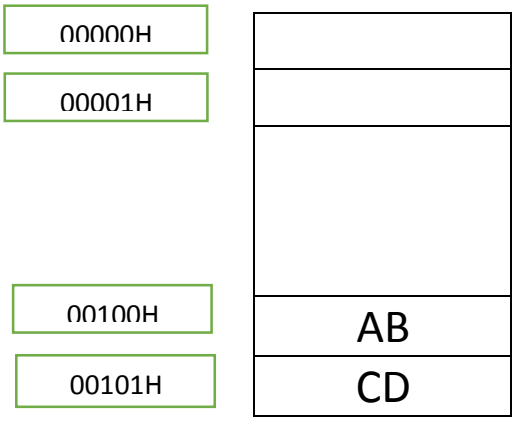
يوضع داخل كل حجرة رقم ست عشري يتراوح بين ٠ وFF ويدعى هذا الرقم بمحتوى تلك الحجرة.

يوجد بين المعالج والذاكرة ممران هما ممر المعطيات بعرض ١٦ بت وممر العناوين بعرض ٢٠ بت.

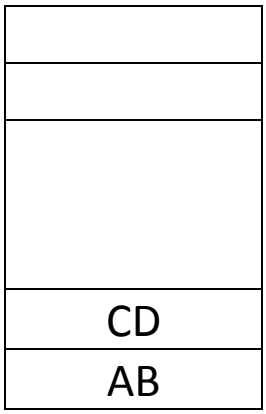


كيف تتعامل الذاكرة مع المعطيات:

تعرف هذه العملية بتنظيم الذاكرة ويتم ادخال البيانات إلى الذاكرة بحجم ٢بايت واحد قادرة على تخزين حرفين ويوجد طريقتين للتخزين تعرفان باسم Big endian و little endian تقوم بتخزين البيانات وفق نظام عكسي من الأعلى للأسفل أو من الأسفل للأعلى أو من اليمين لليسار أو اليسار لليمين فمثلا عندما نريد تخزين بيانات مكونة من الاحرف ABCD فإن عملية التخزين على طريقة big endian



بينما عملية التخزين تتم لطريقة little endian تتم وفق التالي :



مسجلات الفهرسة والتأشير

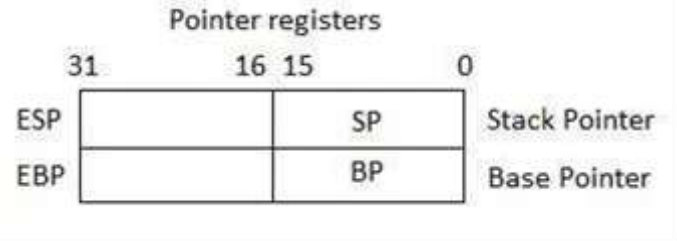
وهي عبارة عن أربعة مسجلات مساعدة تساعد في إيجاد العنوان الفيزيائي بالتعاون مع مسجلات المقاطع، وطول هذه المسجلات ١٦ بت أي ٢ بايت، وهي:

١. **مسجل دليل المصدر Source Index SI:** يخزن فيه عنوان يدل على الإزاحة ضمن مقطع المعطيات DS وبمعنى آخر يستعمل في إمساك العناوين الفعالة من أجل التعليمات التي تتناول المعطيات المخزنة في مقطع المعطيات في الذاكرة.

٢. **مسجل دليل الهدف Destination Index DI:** يخزن فيه عنوان يدل على الإزاحة ضمن مقطع المعطيات الإضافي ES، وبمعنى آخر يستعمل مسجل دليل الهدف DI من أجل استنتاج العنوان الفيزيائي الذي يحدد حجرة متحول الهدف.

٣. **مسجل مؤشر المكدس Stack Pointer SP:** يسمح مؤشر المكدس بوصول سهل للحجرات في مقطع المكدس الموجود في الذاكرة حيث أن القيمة في SP تمثل العنوان الفعال لحجرة المكدس التالية التي يمكن الوصول إليها نسبة إلى العنوان الحالي الموجود في مسجل مقطع المكدس SS ويحتفظ SP دوماً بقيمة تدل على قمة المكدس، هذا وإن قيمة هذا المسجل تتعدل تلقائياً عند وضع أو سحب معلومة بالمكدس.

٤. **مسجل مؤشر القاعدة Base Pointer BP:** يحوي قيمة تدل على الإزاحة بالنسبة لمقطع المكدس SS وهو يستخدم لقراءة المعطيات ضمن مقطع المكدس بدون إزالتها من المكدس.



ما هو نظام الترميز Encoding System:

هو نظام المحارف والرموز في الحاسب والآلي وهي عملية تحويل الأرقام الثنائية التي يفهمها الحاسب الآلي إلى رموز ومفاتيح أو ما يعرف بالتمثيل البصري Representations أي أن جميع العمليات الكتابية التي تقوم بها على حاسوبك الشخصي تتبع هذا النوع من الأنظمة وتعمل وفق النظام الثنائي أي أن لكل حرف أو رمز في لوحة المفاتيح له رقم يقابله في جدول الترميز بنظام encoding في بداية ظهور الحاسب الآلي كانت اللغة الرئيسية المستخدمة هي اللغة الإنكليزية لذلك لم يكن هناك اختلاف في عدد الأحرف والرموز وكان النظام المتداول أنداك يعرف باسم

الآسكي ASCII ويحتل كل حرف أو رمز حجرة خاصة بها وفق النظام الستة عشري لنتعرف أكثر على أنظمة الترميز تابع معي

نظام الترميز ASCII:

هو النظام الأقدم المستخدم في الحواسيب الآلية وأيضا في الكثير من الأدوات والأجهزة الرقمية المتداولة اليوم كالجوال والساعات الذكية وتقنيات المنزل الذكي يعتمد هذا النظام على حجز ١٢٨ خانة للأحرف الإنكليزية بصيغتين الكبير والصغير أي أن الحرف A له خانة خاصة به تختلف عن الحرف a الصغير إضافة الى خانات أخرى تتعلق بمفاتيح التحكم والأرقام ففي حال استدعائها بلغة الاسمبلي فإن العملية تحتاج إلى تغير نمط كتابة الكود وذلك بالعودة إلى جدول الاسكي بكل الأحوال فإن النسخة الأولى من نظام التشفير Ascii كانت تحتوي على ١٢٨ محرف فقط كل منها يحجز خانة بحجم ٢بت موزعة على النحو التالي

- أحرف التحكم وعددها ٣٣ حرف
- الاحرف الهجائية وعددها ٥٢ حرفا للغة الإنكليزية فقط
- رموز مختلفة يبلغ عددها ٣٢ رمزا
- الأرقام وعددها عشرة بدء من الصفر وحتى التسعة

ثم تم تطويره فيما بعد ليقبل ٢٥٦ حرفا ورمزا حيث كانت الرموز الجديدة التي تم إدخالها عبارة عن اشكال العملة الأجنبية عدا الدولار حيث عانى الكثير من مستخدمي أنظمة الحاسب الآلي من هذه المشكلة وخاصة مع ظهور نظام التشغيل Windows والذي كان يدعم الاسكي فقط قبل أن يتم تطوير Unicode إضافة إلى إدخال الرموز الصوتية الإنكليزية والتي كانت مهمة جدا في الجامعات والمعاهد الامريكية

لماذا لم يتم تطوير نظام أوسع:

يعود السبب الرئيسي في عدم تطوير نظام يقبل جميع لغات العالم إلى ضعف السعة التخزينية للذواكر وغلاء ثمنها حيث كانت الإمكانات المتاحة هي من تفرض تطوير نظام الترميز ضمن مساحة محدودة حتى أن النسخ الأولى لنظام التشغيل win95 كانت تدعم الاسكي فقط قبل أن تقوم الشركة بتطوير النظام فيما بعد وفق معايير Unicode لكل النسخ السابقة من أنظمة التشغيل بما فيها win3.1

هل يدعم اسكي اللغة العربية:

بالنسبة للغة العرية فإن الاسكي في إصداراته الأولى لا يدعم إلا اللغة الإنكليزية ومحارفها بالكامل كذلك الإصدار الثاني كان عبارة عن تطوير للمحارف التي تعتمد اللغة الإنكليزية فقط كما ذكرنا سابقا لذلك تم ابتكار طريقتين الأولى هو الاستغناء عن الرموز والاحرف الصوتية التي وجدت في نظام الاسكي ASCII الثاني والذي يتألف من ٢٥٦ خانة منها ١٢٨ بيت محجوزة وتفي بالغرض للكتابة بالإنكليزية ١٢٨ الأخرى يتم استبدالها بمحارف للغات أخرى لكن هذا الامر لم ينجح على الاطلاق فبعض اللغات تمتلك عدد حروف ورموز ضخم أكثر مما تملكه اللغة الإنكليزية لذلك تم تطوير مشروع متقدم عن لغة الترميز الاسكي ويعرف بنظام الترميز DBCS

النظام الوليد DBCS المطور من ASCII:

نظام DBCS هو اختصار لعبارة Double Byte Character Set يعتمد هذ النظام على تنسيق الاحرف وتوزيعها بحيث تحجز المحارف التي تحتاج ٢بت خانة بحجم ٢بت فيما تبقى الاحرف الأخرى لتحجز بت واحد بدلا من اثنين كما كان معمول سابقا في نظام الاسكي وهذا يعني إدخال رموز ومحارف جديدة إضافية جاءت هذه الفكرة من كون أن نظام الاسكي مرتب على حجر بحجم ثمانية بت يتم استغلال سبعة منها فقط في حين يبقى واحد بت غير مستغل المشروع لم يلقى نجاحا كبيرا والسبب يعود إلى حاجة اللغات إلى حجم ذاكرة أوسع لتشمل أكبر عدد من الرموز مع اعتبار أن الاحرف والأرقام تم تعينها لتحجز كامل حجرات الذاكرة المخصصة لنظام الترميز

Dec	Hex	Char
128	80	Ç
129	81	ü
130	82	é
131	83	â
132	84	ä
133	85	à
134	86	å
135	87	ç
136	88	ê
137	89	ë
138	8A	è
139	8B	ï
140	8C	î
141	8D	ì
142	8E	Ä
143	8F	Å
144	90	É
145	91	æ
146	92	Æ
147	93	ô
148	94	ö
149	95	ò
150	96	û
151	97	ù
152	98	ÿ
153	99	Ö
154	9A	Ü
155	9B	¢
156	9C	£
157	9D	¥
158	9E	₧
159	9F	ƒ

Dec	Hex	Char
160	A0	á
161	A1	í
162	A2	ó
163	A3	ú
164	A4	ñ
165	A5	Ñ
166	A6	ª
167	A7	º
168	A8	¿
169	A9	⌐
170	AA	¬
171	AB	½
172	AC	¼
173	AD	¡
174	AE	«
175	AF	»
176	B0	░
177	B1	▒
178	B2	▓
179	B3	│
180	B4	┤
181	B5	╡
182	B6	╢
183	B7	╖
184	B8	╕
185	B9	╣
186	BA	║
187	BB	╗
188	BC	╝
189	BD	╜
190	BE	╛
191	BF	┐

Dec	Hex	Char
192	C0	└
193	C1	┴
194	C2	┬
195	C3	├
196	C4	─
197	C5	┼
198	C6	╞
199	C7	╟
200	C8	╚
201	C9	╔
202	CA	╩
203	CB	╦
204	CC	╠
205	CD	═
206	CE	╬
207	CF	╧
208	D0	╨
209	D1	╤
210	D2	╥
211	D3	╙
212	D4	╘
213	D5	╒
214	D6	╓
215	D7	╫
216	D8	╪
217	D9	┘
218	DA	┌
219	DB	█
220	DC	▄
221	DD	▌
222	DE	▐
223	DF	▀

Dec	Hex	Char
224	E0	α
225	E1	ß
226	E2	Γ
227	E3	π
228	E4	Σ
229	E5	σ
230	E6	µ
231	E7	τ
232	E8	Φ
233	E9	Θ
234	EA	Ω
235	EB	δ
236	EC	∞
237	ED	φ
238	EE	ε
239	EF	∩
240	F0	≡
241	F1	±
242	F2	≥
243	F3	≤
244	F4	⌠
245	F5	⌡
246	F6	÷
247	F7	≈
248	F8	°
249	F9	∙
250	FA	·
251	FB	√
252	FC	n
253	FD	2
254	FE	■
255	FF	

الأسمبلى لغة التجميع:

سنقوم في البداية بشرح لميزات يمكن الحصول عليها من خلال تعلم لغة الأسمبلي:

١- اتصال البرامج مع نظام التشغيل والمعالج و وحدة ادخال واخراج المعطيات (BIOS) فالكثير من البرامج التي يتم تصميمها من قبل المطورين والتي تستهدف تعريف قطع العتاد الصلب تحتاج الى خبرة في مجال هذه اللغة ولابد من دقة البرمجة في هذا المجال لتحقيق توافقية كاملة ما بين العتاد الصلب والنظام والرابط ما بين الاثنين الاف الاسطر من شيفرات الاسمبلي حيث لا يوجد بتاتا مجال للخطأ يوجد العديد من الابحاث التي تظهر استخدام الهندسة العكسية في عملية برمجة مكتبات ربط للتعامل مع الاجهزة المحيطة قد نتطرق لها نظريا

فيما بعد كون أن هذا المجال من اختصاص مبرمجي نظم التشغيل وليس لنا متسع كبير للدخول في مجالات عملية مع هذا القسم

٢- تمثيل البيانات في الذاكرة والأجهزة الخارجية الأخرى وهذا تم شرحه سابقاً في الفصل السابق

٣- كيفية قيام المعالج بإدخال وتنفيذ الأوامر وهذا الامر تم شرحه سابقا في قسم كامل

٤- كيفية تعامل البرنامج مع الجهاز الخارجي ولهذه اللغة دور بالغ في عملية ادخال هذا العتاد الى الحاسب الآلي وعملية السيطرة عليه داخل نظم التشغيل وقبلها داخل وحدة ادخال واخراج المعطيات وقد ورد شرح بسيط لكيفية القيام بذلك

٥- يسمح للعتاد الصلب بإجراءات معقدة بطرق سهلة وللعلم فإن لغة الاسمبلي هي لغة العتاد الصلب بشكل عام مهما اختلف نوع هذا العتاد ومهما كان استخدامه في كل زمان ومكان

٦- سرعة التنفيذ وذاكرة أقل حجما وهنا يأتي دور المنشآت الحكومية والعسكرية والمعامل العملاقة والتي تقوم باستغلال الاسمبلي كلغة أولى في عملية ادارتها وذلك لحاجتها لهذه السرعة والكفاءة العالية فحتى المنشآت النووية تستخدم برامج عالية الدقة صممت بواسطة المجمع للتحكم بها مما يجعل مجال الخطأ داخل هذه المنشآت يكاد يكون ضيق وهي النتيجة المرجوة التي يريد الكثيرون من الحكومات او جنرالات الهندسة العسكرية او مديري المعامل العملاقة الحصول عليها

تعليمات اسمبلي:

كما ذكرنا سابقاً بأن الأجهزة الداخلية للحاسوب تشمل الذاكرة والمسجلات و المعالج وتقوم بنقل البيانات والعناوين عبر ممرات خاصة تم تفصيلها سابقاً لتنفيذ تطبيق معين قام المصمم ببرمجته و ليتم تنفيذ تطبيق ما النظام يقوم بنسخ الملفات عن طريق جهاز خارجي إلى الذاكرة المعالج الذي يقوم بتنفيذ تعليمات هذا البرنامج مجموعة التعليمات التي يتعامل معها المعالج في لغة الاسمبلي هي المسؤولة عن عمليات رياضية ومنطقية معقدة وسنقوم بدراسة مطولة لهذه التعليمات لتداخلها الكبير في الهندسة العكسية فكل ما في هذا العلم هو عبارة عن فهم واسع للعمليات الحسابية التي يقوم المعالج بالإشراف عليها المعالجات ذات المعمارية ٨٠٨٦ تحتوي على ١١٧ تعليمة خاصة بلغة المجمع

يردفها تعليمات أخرى فرعية تتفرع من التعليمة الأساسية لكن قبل الدخول إليها لا بد من معرفة أساسيات هامة في كتابة نص برمجي بهذه اللغة

شكل تعليمة أسمبلي

تتكون تعليمة أسمبلي من أربعة أجزاء رئيسية تقوم بتنفيذ عملية ما يتم استقبالها من قبل لغة الآلة كشيفرة تعليمات ثنائية (نظام العد الثنائي الواحد والصفر) الهدف الأساسي من هذه التعليمات هي القيام بإجراء عملية محددة كتخصيص مساحة من الذاكرة لمتغير ونقل رقم معين إلى مسجل ما للقيام بعملية تجاوز لرسالة اعتراض وضعت من قبل مبرمج ما والكثير من الأمور الأخرى أما شكل هذه الأجزاء الأربعة فهي كالتالي:

الاسم	العملية	المتأثر	التعليق
Start	MOV	CX,AX	Any Comment

يتم الفصل بين هذه التعليمات من خلال مفتاح المسطرة بمسافة واحدة على الأقل لاحظ أننا قمنا في المثال السابق بعملية نقل قيمة المسجل CX إلى المسجل AX طبعاً طريقة الشيفرة البرمجية هي عامة لكل التعليمات الأخرى بالنسبة للاسم والتعليق هو أمران اختياريان وبشرح مفصل أكثر

الاسم (Name):

يستخدم هذا القسم لعنونة عملية محددة أو لكتابة متغير ما أو حتى للقيام باستدعاء برنامج فرعي بكل الأحوال فإن هذا القسم يتم تسجيله في الذاكرة كعنوان للعملية التي تم تنفيذها ويشترط في كتابة الاسم ثلاث أمور

١- ألا يبدأ الاسم برقم

٢- أن يكون عدد الأحرف ٣١ كحد أقصى

٣- عدم الفصل بين الكلمة نفسها بمسافات

التعليمة (Instruction):

هذا القسم إجباري في عملية كتابة الأوامر ويحتوي كما شاهدنا سابقاً على التعليمة المراد تنفيذها من قبل لغة المجمع بكل الأحوال هذه التعليمات تتميز بأنها ثابتة ومتفرعة

إلى تعليمات أخرى إضافية وسريعة التنفيذ من قبل رقاقة المعالجة الرئيسية التي تتعامل معها كلغة الآلة

المتأثر (Operand):

طبعاً هذا القسم مهم جدا أيضاً فالتعليمة البرمجية التي يتم تنفيذها تقوم بالتأثير مباشرة على هذا الجزء الذي يتعامل مع مكونات الحاسب كالذاكرة الرئيسة والمسجلات والمراكم ويستفيد من التعليمة في إجراء هذه العمليات ويلاحظ من خلال كتابة شيفرة التعليمات السابقة أن المتأثر الأول غالباً ما يقع عليه التغيير من خلال التعديل الذي يريد المبرمج تنفيذه على هذا الجزء ومثال على ذلك

```
ADD CX,1

MOV CX,AX
```

من خلال المثال السابق يمكننا القول بأن التأثير كله حدث على المسجل CX أي الجزء الأول كما قلنا سابقاً حيث قمنا بإضافة واحد إلى المسجل CXثم قمنا بعد ذلك بنقل قيمة هذا المسجل إلى المسجل AX من نفس النوع إذا المتأثر الأول هو الذي يتم التعديل عليه أما المتأثر الثاني فيمكن تسميته المتأثر المصدر

التعليقات (Comment):

تعتبر التعليقات جزء اختياري في كتابة التعليمات بلغة المجمع وتستخدم بشكل عام لشرح الشيفرة التي تم كتابتها من قبل المبرمج لتوضيح ماهية عمله وتستخدم الفاصلة المنقوطة حصراً لفصل التعليمات عن التعليق وجعله غير قابل للتنفيذ ومثال على ذلك:

```
MOV AX,4 ; نقل العدد ٤ الى المسجل الف اكس

MOV BX,3 ; نقل العدد ٣ الى مسجل البي اكس

ADD AX,BX ; جمع المسجلين مع بعض والناتج هنا ٧
```

يفضل كتابة التعليق بعد كتابة التعليمة ليتم شرحها مباشرة أما إذا أردت شرح مجموعة من التعليمات دفعة واحدة فيمكن كتابة التعليق فوق هذه التعليمات وليس على يمينها لشرح ما تفعله هذه المجموعة من التعليمات

```
عملية حسابية لجمع قيمة مسجلين;
MOV AX,4
MOV BX,3
ADD AX,BX
```

طبعاً عرضنا مثالين مختلفين لكيفية كتابة التعليق بعد التعليمة أو فوق مجموعة من التعليمات كما وضحنا سابقاً وللعلم فإن هذه الطريقة متبعة في عدد كبير من لغات البرمجة

١- التعليمة ADD:

تعتبر هذه التعليمة هي المسؤولة عن عملية جمع قيمتان لمتأثر واحد وتخزين هذه القيمة في المتأثر الأول في حالة تم إزاحة هذه القيمة فإن مؤشر علم التحكم سوف يشير إلى ذلك وتسمح أطقم تعليمات المعالج بالتعامل مع هذه التعليمة وفق الآتي

- من مسجل إلى مسجل
- من مسجل إلى الذاكرة
- من الذاكرة إلى المسجل

الآن سنقوم بطرح مجموعة من الأمثلة على هذه التعليمة وطريقة تمثيلها في لغة المجمع والمثال الاول هو المثال السابق:

```
عملية حسابية لجمع قيمة مسجلين;
MOV AX,4
MOV BX,3
ADD AX,BX
```

الآن سنقوم بعملية حسابية ثانية لجمع عددين بنظام العد الست عشري وهي على الشكل التالي D4H+25H=F9 طبعا العملية الحسابية كما نلاحظ في نهاية كل رقم الرمزH والذي يشير الى انه رقم بالنظام الست عشري

```
عملية حسابية لجمع قيمة مسجلين بالنظام الست عشري;
MOV AX,D4
ADD AX,25H
```

في هذا المثال سوف نقوم بعرض عملية حسابية داخل مسجلات من نوع ٣٢ بت في العمليات السابقة اظهرنا العملية الحسابية على مسجلات من نوع ١٦ بت

```
عملية حسابية لجمع قيمة مسجلين بالنظام الست عشري على مسجل ٣٢ بت;
MOV EAX,D4
ADD EAX,25H
```

تتعامل هذه التعليمة مع الذاكرة بشرط ان يتم الجمع من موقع في الذاكرة الى مسجل والعكس صحيح أما بالنسبة لعملية الإضافة داخل الذاكرة مباشرة فهذا مستحيل لابد من نقل القيمة الى مسجل ثم اضافته الى الذاكرة أي لا بد من وسيط لضمان صحة العملية وهذا مثال بسيط يشرح العملية مع اعتبار أن ٣٢١٢-٥٠٠١-١٢٢٢ هي عبارة عن عناوين في الذاكرة

```
عملية حسابية لجمع قيمة مسجل مع عنوان في الذاكرة على مسجل ٣٢ بت;
MOV EAX,3212
ADD EAX,1222
ADD EAX,5001
```

لاحظ في المثال السابق كيف تمت عملية نقل عنوان الذاكرة في البداية الى المسجل ثم بعد ذلك قمنا بعملية اضافة تلك العناوين انظر الى المثال التالي:

```
عملية حسابية لجمع قيمة عنوان مع عنوان في الذاكرة وهي عملية ;
خاطئة لابد من وسيط سجل
ADD [5001],[1222]
```

الجمع في الطريقة السابقة خاطئة سنقوم بأخذ مثال جديد لكن داخل المسجلات نفسها ولاحظ وجود الخطأ في هذه العملية الحسابية

عملية حسابية لجمع قيمة مسجل ٣٢ بت مع مسجل ١٦ بت عملية خاطئة;

```
MOV EAX,4DH
ADD AX,30H
ADD EAX,AX
```

عملية حسابية لجمع قيمة مسجل ٨ بت مع مسجل ١٦ بت عملية خاطئة;

```
MOV AX,4DH
ADD AL,30H
ADD AX,AL
```

لاحظ أن العمليتان السابقتان تحتاج لعمليتا النقل والاضافة توافر مسجلات من نفس النوع أي لا يمكن أن نقوم بإضافة قيمة إلى المسجل من نفس النوع وقد اخدنا دراسة كافية عن أنواع المسجلات وشرحناها في السابق يمكنك مراجعتها والتركيز عليها حتى لا تقع في الخطأ

إضافة قيمة إلى عنوان ذاكرة من نوع كلمة مزدوجة أي أربع بايتات ممثلة بالعدد ٢٥

```
add dword ptr [100h], 25
```

إضافة قيمة عنوان في الذاكرة الى مسجل من نوع ٣٢ بت

```
add dword ptr [100h], ecx
```

٢- التعليمة SUB:

تمثل هذه التعليمة عملية طرح قيمة المتأثر الثاني من المتأثر الأول وتخزين نتيجة هذه العملية في المتأثر الأول ويتم التعامل مع هذه التعليمة وفق الاحتمالات الآتية

- من مسجل إلى الذاكرة
- من الذاكرة إلى المسجل
- من مسجل إلى مسجل

سنقوم بطرح مجموعة من الأمثلة يرجى الانتباه حتى نتجنب الوقوع في الخطأ كما أننا سنقوم بإضافة أي عملية قمنا بدراستها سابقا من أجل فهم التعليمات بشكل تام لنلاحظ المثال الأول

```
عملية حسابية لجمع قيمة مسجلين ثم طرح قيمة من احد المسجلات;
MOV AX,4
MOV BX,3
ADD AX,BX
SUB AX,2
```

الآن سنقوم بأخذ مثال مشابه للسابق على مسجل من نوع ٣٢ بت لاحظ جيداً للعلم فإن أفضل مسجل للتعامل مع قيم حسابية ورياضية هو المسجل AX من نوع ١٦ بت وحتا المسجل من نوع ٣٢ بت EAX واجزاء المسجل AX وهي المسجلين الهامين AL-AH

من فئة ٨ بت هذا تم دراسته سابقاً يعرف هذا المسجل بأنه مسجل مراكم فقط هذا للتذكرة حتى لا يتم الخلط بين وظائف هذه المسجلات

```
عملية حسابية لجمع قيمة مسجلين ثم طرح قيمة من احد المسجلات;
MOV EAX,4
MOV EBX,3
ADD EAX,EBX
SUB EAX,2
```

بالتأكيد هذه التعليمة ينطبق عليها ما ينطبق على التعليمة السابقة وهي تعليمة الجمع بالنسبة للتعامل مع المسجلات والذاكرة أي لا يمكننا إجراء عملية بين موقعين في الذاكرة حتى لا نقع في الخطأ المثال التالي هو مثال صحيح للعلم

```
عملية حسابية لجمع قيمة في الذاكرة  ثم طرح قيمة من احد المسجلات;
MOV AX , FFFF
SUB AX , CX
```

طبعا في المثال السابق القيمة FFFF هي قيمة في الذاكرة يمكنك كتابة بعض الأمثلة لتمرن نفسك أكثر وللعلم فأن عملية الطرح من قيمة تؤثر على الأعلام داخل المنقح

سنأخذ مثال خاص بكتابة أمر اسمبلي تحت بيئة نظام التشغيل MS DOS من خلال اجراء عمليات الطرح والجمع طبعا لابد من استخدام الأمر INT 21H وهو مقاطعة خاصة بنظام التشغيلDOS

```
.model small
.data
.code
 main proc
mov dl, 2
sub dl ,1
add dl, 48
mov ah, 2h
int 21h
endp
end main
```

٣- التعليمة MOV:

تستخدم هذه التعليمة لنقل بيانات معينة من المتأثر المصدر إلى المتأثر الهدف مهما كانت نوعية هذا المتأثر أما الاحتمالات التي تتعامل معها هذه التعليمة فهي كثيرة وتتم وفق الآتي:

- من المراكم إلى الذاكرة والعكس صحيح أيضاً

- من المسجل إلى المسجل
- من الذاكرة إلى المسجل والعكس صحيح
- من مسجل ١٦بت إلى مسجل مقطع
- من ذاكرة إلى مسجل مقطع
- من مسجل مقطع إلى مسجل ١٦بت

الآن سنقوم بأخذ أمثلة على هذه التعليمة وعلينا التدقيق بكيفية كتابتها بالطريقة الصحيحة

المثال الاول هو مثال سابق لاحظ في البداية كيفية نقل قيمة الى مسجل باستخدام هذه التعليمة:

```
عملية حسابية لجمع قيمة مسجلين ثم طرح قيمة من احد المسجلات;
MOV AX,4
MOV BX,3
ADD AX,BX
SUB AX,2
```

التعامل مع المسجلات من نوع ٨ بت أو حتى ١٦ بت طبعاً مشابه لما جاء في العمليات الحسابية السابقة لاحظ المثال التالي

```
عملية حسابية لجمع قيمة مسجلين من نوع ٨بت ثم طرح قيمة من احد المسجلات;
MOV Al,4
```

```
MOV Bl,3
ADD Al,Bl
SUB Al,2
عملية حسابية لجمع قيمة مسجلين من نوع ٣٢بت ثم طرح قيمة من احد المسجلات;
MOV EAX,4
MOV EBX,3
ADD EAX,EBX
SUB EAX,2
```

يمكن أخذ بعض الأمثلة الأخرى بالنسبة للتعليمة MOV وستكون هذه التعليمات خاطئة لاحظ معنا فكل ما ينطبق في السابق على ما سبق ذكره من تعليمات ينطبق هنا

```
عملية حسابية لنقل قيمة الى مسجلين من نوع ٨بت و١٦ بت وجمعهما وهي عملية ; خاطئة
MOV Al,4
MOV BX,3
ADD Al,BX

عملية حسابية لنقل قيمة الى مسجلين من نوع ٣٢بت و١٦ بت وجمعهما وهي عملية ; خاطئة
MOV AX,4
MOV EBX,3
```

```
ADD AX,EBX
```

الآن سنقوم بأخذ مثال لتعامل هذه التعليمة مع عناوين داخل ذاكرة وللعلم فإن ما ينطبق على ما سبق من تعليمات رياضية ينطبق على هذه التعليمة أي أنه لا يمكن نقل موقع داخل ذاكرة الى موقع داخل الذاكرة مباشرة دون استخدام وسيط غالبا هو المسجل على كل حال لاحظ المثال التالي:

```
عملية حسابية لنقل قيمة في الذاكرة  ثم طرح قيمة من احد المسجلات;
MOV AX , FFFF
MOV CX,1
SUB AX , CX
```

تستخدم التعليمة MOV كما ذكرنا سابقاً للتعامل مع المكدس ومسجلات المقطع وسنقوم بأخذ مجموعة من الأمثلة لمعرفة كيفية استخدامها بالطريقة الصحيحة لكن قل كل شيء يجب التذكير ببعض التعاريف الهامة المتعلقة بالمكدس حتى نتقن الكتابة بشكل صيح وللعلم فإن ما سيرد الآن تم شرحه سابقاً لكنه ذو أهمية بالنسبة للمبتدئين فوجب التذكير به لاحظ الجدول:

الوصف	نوع المسجل
شيفرة المقطع	CS
بيانات المقطع	DS
مكدس المقطع	SS
مؤشرات مقطع إضافية	ES-FS-GS

المسجلات السابقة هي مجموعة مسجلات من نوع ١٦ بت لذلك علينا الانتباه إلى هذه الملاحظة على كل حال المثال التالي يشرح كيفية نقل محتويات ثلاث خانات في المكدس وذلك دون تغيير محتويات المكدس

```
MOV BP , SP
MOV AX , [ BP]
MOV BX , [ BP + 2]
MOV CX , [ BP + 4]
```

ملاحظة هامة: مسجل مؤشر القاعدة Base Pointer BP يحوي قيمة تدل على الإزاحة بالنسبة لمقطع المكدس SS وهو يستخدم لقراءة المعطيات ضمن مقطع المكدس بدون إزالتها من المكدس ودون إحداث تبديل داخل المكدس وللعلم فإن هذا المسجل هو من فئة ١٦ بت ويمثله مسجل رديف له من فئة ٣٢ بت في المعالجات الأحدث EPP

الآن سنقوم بإعطاء مثال عن كيفية التعامل مع عنوان ذاكرة حيث سنقوم بعملية نقل جزء من عنوان إلى أحد المسجلات وللعلم فإن تمثيل القيمة العددية يكون بطريقة الازاحة على كل حال انظر الى المثال التالي

لاحظ كيفية كتابة التعليمةALهنا عملية نقل قيمة عنوان إلى المسجل

```
mov al, Byte ptr [0000005Bh]
```

في حال تم عملية نقل قيمة بمقدار كلمة أي اربع كيلوبايت فيتم بالطريقة التالية

```
mov eax, Dowrd ptr [0000005Bh]
```

٤- التعليمة CMP:

تعليمة المقارنة مهمة جدا في مجال الهندسة العكسية حيث تعتبر المفتاح الرئيسي للقيام بعمليات مقارنة بين قيمة متأثرين ويتم إعطاء إيعاز لأعلام الحالة إذا ما تم التحقق من العملية من أجل تنفيذ الشطر التالي من التعابير البرمجية والذي غالباً ما يرتبط بتعليمة القفز JMP وهذا يمثل الجملة الشرطية لكن مختلف نوعاً ما عن التعليمات المستخدمة في لغات البرمجة العليا أما احتمالاتها فهي

- من المسجل إلى الذاكرة
- من الذاكرة إلى المسجل
- من مسجل إلى مسجل

تعمل وظيفة المقارنة على طرح قيمة مسجلين ثم بعد ذلك يتم التأثير على الأعلام والتي بدوره تعطي إيعاز لإتمام عملية القفز من عدمه وكل ذلك يحدث دون أن تأثر عملية الطرح من قيمة المسجلين حيث تبقى القيم ثابت وإنما عملية التعديل تحدث على الأعلام

مقارنة ومعرفة النتيجة من خلال الراية zf-cf

لدينا مثال لعملية المقارنة من خلال نقل قيمة معينة ومن ثم اجراء مقارنة بين قيمة المسجل ax والعدد ٥ جرب باستخدام المحاكي Emu8086

```
.model small
.data
 .code
  main proc
   mov ax, ٥
```

```
cmp ax, ٥
Endp
```

٥- التعليمة XCHG:

تقوم هذه التعليمة بتبديل قيمة المؤثر الأول وإلحاقها بالمؤثر الثاني ولهذه التعليمة عدة احتمالات في التعامل مع كيان الحاسب وهي كالآتي:

- من المسجل إلى الذاكرة
- من الذاكرة إلى المسجل
- من مسجل إلى مسجل
- من ذاكرة إلى ذاكرة

المثال التالي يوضح عملية تبديل عنصري مسجل ببعضهما على سبيل المثال العنصرeax يحتوي القيمة ٣ والمسجل ebx يحتوي الرقم ٩ فإن عملية تبديل المسجلين ستعكس القيم فيذهب الرقم ٣ إلى المسجل ebx والرقم ٩ إلى المسجل eax

```
طبعاً هذا ينطبق على المثال السابق الذي تم شرحه
Xchng EAX , EBX
يمكن تبديل محتويات في عنوان ذاكرة إلى مسجل يحمل قيمة هذه البيانات
XHNG [000FA],EAX
```

٦-تعليمة NEG:

يتم من خلال هذه التعليمة قلب العملية الحسابية من الموجب الى السالب أو العكس ايضا من السالب الى الموجب وتتعامل هذه التعليمة مع الذاكرة إضافة الى تعامله مع المسجلات

٧-التعليمة LOOP:

تنتج هذه العملية تكرار الأحداث لعدة مرات يقوم المبرمج بتحديده أو حتى عملية تكرار مفتوحة حتى يتحقق شرط ما في المثال التالي نلاحظ أنه عندما cx لا يساوي الصفر يتم القفز إلى مربع اللافتة القصيرة إن عملية القفز والمقارنة والتكرار من العملية المهمة جدا في مجال البرمجة العكسية ولابد من فهم هذه التعليمات إضافة إلى تعليمتي الازاحة حيث تقوم هذه التعليمات بعملية التحقق من حدث قبل السماح بالمعالج باتخاذ قرار المناسب ما إذا كانت نافذة معينة ستفتح او لا إضافة إلى دورها الكبير في عملية التسجيل التطبيق العملي سيدخل المعلومة أكثر في بالك حاول اللعب بالكود كي تتمكن من فهمه أكثر داخل المحاكي

```
.model small
.data
.code
 main proc
  mov cx, 5
  mov bx, 5
   lop:
   mov dl, 6
   add dl, 48
```

```
 mov ah, 2h
 int 21h
cmp bx, 5
loopz lop
Endp
end main
```

٨- التعليمة INC -DEC:

تقوم هذه التعليمة بزيادة قيمة المؤثر بمقدار العدد (١) في مسجل ما أو حتى في تعاملها مع قيمة داخل الذاكرة أما تعليمة DEC فتقوم بعملية إنقاص قيمة هذا المسجل أو قيمة داخل الذاكرة بمعدل العدد واحد عملية معاكسة تماماً البرنامج التالي الذي سنقوم بكتابته يمثل عملية زيادة قيمة معينة بمقدار ثلاثة مضافة للعدد ٤٨ داخل المسجل DL

```
.model small
.data
.code
main proc
mov dl, 3
inc dl  ;      تتم عملية الزيادة من خلال هذا الامر
add dl, 48
```

```
mov ah, 2h
int 21h
endp
end main
```

عكس العملية السابقة حيث قمنا بكتابة الاسطر لتمثل عملية إنقاص العدد ثلاثة من القيمة الموجودة داخل المسجل DL وهي العدد ٤٨ من خلال الامر DEC

```
.model small
.data
.code
main proc
mov dl, 3
dec dl                    ; الانقاص يحدث هنا
add dl, 48
mov ah, 2h
int 21h
endp
end main
```

٩- التعليمة MUL-DIV:

تشرف هذه التعليمة على تنفيذ عملية الضرب بين المؤثر ومسجل التراكم ويقوم بتخزين قيمة هذه النتيجة في مسجل التراكم وينطبق هذا الأمر على التعليمة DIV وتعني القسمة حيث يقوم بعملية قسمة بين المؤثر ومسجل التراكم وتخزين القيمة النهائية في المسجل

```
طبعاً هنا عملية ضرب لمسجلين من نوع ٣٢ بت
add eax , 2
add ecx , 3
mul eax , ecx
```

المثال التالي للتعامل مع عنوان الذاكرة من خلال التعليمة Add من ثم اجراء العملية الحسابية الضرب أو القسمة

```
ثم ضرب eaxالعملية التالية تعبر عن إضافة قيمة من عنوان ذاكرة الى مسجل
قيمة مسجلين ٣٢بت
add eax, dword ptr [100h]
add ecx , 2
mul eax , ecx
```

العملية التالية هي عملية معاكسة للعملية السابقة حيث يتم إضافة قيمة المسجل الى الذاكرة وبعدها الضرب

```
add dword ptr [100h], ecx
add ecx , 2
mul eax , ecx
```

برنامج صغير وهو عبارة عن اجراء عملية قسمة بين عددين أحد هذه الاعداد تم تسجيل قيمته وفق النظام الستة عشري ووضعه في المسجل ax إضافة على وضع قيمة تمثل العدد ٢ في المسجل bl وطبع النتيجة على الشاشة

```
.model small
.data
code
main proc
mov ax, 0080h
mov bl, 2
div bl
Endp
end main
```

وضع قيمتين في مسجلين ومن ثم اجراء عملية الضرب وطبع الناتج على الشاشة تابع كيف تستجيب الرايات داخل المحاكي لهذا النوع من العمليات الحسابية

```
.model small
.data
.code
 main proc
  mov al, 5h
  mov bl, 2h
  mul bl
 Endp
 end main
```

١. الامر TEST:

تقوم هذه التعليمة بإجراء عملية فحص خانة معينة داخل مسجل إذا أردنا فحص خانة فإن البرنامج يشير في هذه الحالة الى الرقم (١) مشيرة الى إجراء العملية وفي حالة عدم قيامه بفحص خانة معينة فإنه يشير الى الرقم (٠) دائما الرقمان يشيران بعلوم البرمجة إلى شطر صحيح وشطر خاطئ هذه التعليمة مشابهة من حيث طريقة تنفيذها للتعليمة الرياضية AND لكن الفرق بينها وبين البقية هي أنها لا تحدث تغير على شطري التعليمة في لغة الاسمبلي وإنما تقوم بعملية فحص وتعديل في الرايات طبعاً لن أقوم بشرح هذه العملية رياضياً فقط راجع التعليمة AND فهي مشابهة لها تماماً فقط نأخذ مثال بسيط

TEST AL, BL

في التعليمة AND سيتم تغير قيمة المؤثر كما نعرف أما بالنسبة لهذه التعليمة فالتعديل يكون على الراية دون العبث بقيمة المؤثر الآن سنقوم بأخذ هذا البرنامج الصغير والذي من خلاله نقوم بفحص قيمة المسجل ah عادة تأتي عملية الفحص قبل قفزة ما وهي أشبه بالجملة الشرطية وذلك من أجل تحقيق

```
.model small
.data
.code
  main proc
   mov ah, 01111110b
   test ah, 01111110b
  endp
end main
```

١. التعليمة Call:

تعرف هذه التعليمة باسم تعليمة المنادى ومن خلالها يتم استدعاء برنامج فرعي وبدوره فإن البرنامج الفرعي يعيد التحكم الى البرنامج الاساسي وعلى سبيل المثال في حالة القيام بعملية تسجيل برنامج حاسوب تقليدي فإن المنقح يقوم بعدة عمليات استدعاء فرعي للتأكد من إن المستخدم قام بإدخال بيانات حقيقية قبل العودة الى البرنامج الاساسي وإعطائه ايعاز بتنفيذ الامر التالي في حال كان تسجيل البرنامج صحيح التطبيقات العملية القادمة ستتيح لنا عملية فهم أكبر لهذا الامر

٢. التعليمة RET:

هذه التعليمة اختصار لكلمة عودة باللغة الانكليزية وتعني هنا العودة الى برنامج المستدعي وتتم هذه العودة عن طريق تخزين البيانات في مؤشر التعليمات وغالبا ما تأتي لتمثل نهاية روتين ما داخل النص البرمجي يمكن

٣. التعليمة :PUSH-POP

إن التعليمة المستخدمة لحفظ البارامترات في المكدس هي تعليمة الدفع PUSH والتعليمة المستخدمة لاسترجاعها هي تعليمة POP. بعد سياق التحويل إلى البرنامج الفرعي نجد أنه من الضروري عادة حفظ محتويات المسجلات الرئيسية أو بعض بارامترات البرنامج الرئيسي هذه القيم يتم حفظها بواسطة دفعها إلى المكدس. وبهذه الطريقة يتم حفظ المحتويات سليمة في مقطع المكدس للذاكرة أثناء تنفيذ البرنامج الفرعي، وقبل العودة إلى البرنامج الرئيسي فإن المسجلات المحفوظة و بارامترات البرنامج الرئيسي يُعاد تخزينها بواسطة سحب القيم المحفوظة من المكدس تستخدم التعليمة push من اجل دفع مربع حوار أو إجراء برمجي ونلاحظ من خلال استخدام المنقحات فيما بعد من ان بداية كل روتين تظهر هذه التعليمة بكل الأحوال لدينا المثال التالي والذي من خلاله نقوم بطباعة عبارة

مرحبا على الشاشة مستخدمين مجموعة من التعليمات إضافة إلى متغير Dword ومن ثم انشاء اجراء رئيسي من اجل عرضه كرسالة وتتم عملية الدفع من خلال دفع المتغير نفسه ليتم عرضه إلى الشاشة

```
.model small
.data
Verible dw "Hello"
.code
 main proc
 push f
 pop Verible
```

```
push Verible
pop f
mov ah, 2h
int 21h
endp
end main
```

١٠- التعليمة INVOKE:

تستخدم هذه التعليمة من اجل استدعاء عدد من الدوال دون استخدام كم كبير من عمليات الدفع للحول على نفس النتيجة مما يوفر مجهود ووفت في عملية طباعة البارمترات

١١- تعليمة XLAT:

ترتبط هذه التعليمة حصرا مع المجل AL، إن تعامل هذه التعليمة يتم مع الجداول المخزنة في الذاكرة فلو وضعنا في BX إزاحة بداية الجدول نسبة إلى مقطع المعطيات DS ووضعنا في AL إزاحة العنصر نسبه إلى بداية الجدول، عندها تقوم تعليمة XLAT بجمع محتويات المسجل AL مع محتويات المسجل BX وتعتبر الناتج إزاحة بالنسبة إلى مقطع المعطيات، ثم تقوم بوضع قيمة الحجرة المعطى إزاحتها في AL.

١٢- التعليمة NOP:

تشير هذه التعليمة الى عدم احداث تأثير على البرنامج تستخدم هذه التعليمة من أجل القيام بإيقاف جميع التعليمات في الاسطر التي تليها وبالتالي فإن التطبيق يتوقف عن القيام بأي عملية ضمن مربع الحوار المعدل فقط وتمنع هذه التعليمة في حالة التعديل على التطبيق من حدوث أخطاء كالتكرار أو حتى عدم استجابة البرنامج بمجرد مروره على الاسطر التالية بعد التعديل على أحد البارامترات السابقة وسيتم الشرح في عملية التطبيق العملي عليها

١٣- التعليمة INT:

تقوم هذه التعليمة باستدعاء المقاطعات تستخدم هي عبارة عن استدعاء اجراء ضروري وايقاف كافة العمليات البرمجية حتى يتم استدعاء هذا الاجراء وتقسم المقاطعات إلى نوعين:

Hardware interrupt: هو إشارة إلكترونية تنبيهيه ترسل إلى المعالج من جهاز خارجي ويحتوي كل جهاز على رقم مقاطعة خاصة به يتم استدعائه ليتم الاستفادة منه برمجيا

Software interrupts: هو عادة ما يكون عبارة عن أمر يسمى System Call في مجموعة التعليمات عند تنفيذه يقوم أيضا بنقل المعالج لتنفيذ روتين المقاطعة

١٦- التعليمتان SHR-SHL:

تستخدم هذه التعليمتان من اجل القيام بعملية ازاحة لقيمة ما من اليمين بمعدل معين كما في التعليمة SHL او حتى من جهة اليسار كما في التعليمة SHR

```
.model small
.data
.code
 main proc
   mov ax, 1
   shl ax, 1
 Endp
 end main
```

١٤- التعليمة SAR-SAL:

تمثل هذين التعليمتين ما يعرف بالإزاحة الرياضية فيما تمثل التعليمات السابقة ما يعرف بالتعليمات المنطقية

١٥- التعليمة ROL-ROR:

تقوم التعليمة ROR بعملية إزاحة قيمة داخل مسجل ما على سبيل المثال إلى اليمين بمقدار عدد من الخانات يتم تحديده أما بالنسبة للتعليمة ROL فإن عملية الازاحة تكون نحو اليسار بمقدار عدد خانات محدد أي أنّ التعليمتان متشابهتان باستثناء اتجاه الإزاحة ويحدث ذلك تأثيراً على علم التحكم

١٦- التعليمة RCL-RCR:

هذه التعليمة مشابهة من حيث العمل للتعليمة السابقة لكن التأثير مختلف من حيث التعامل مع الأعلام حيث أن التعليمتين السابقتين تأثر على العلمين OF,CF بينما في هذه التعليمتين تؤثر على اعلام الحالة فقط

التعليمة	معنى العملية	التأثر على الاعلام
ROL	إزاحة لليسار	OF , CF
ROR	إزاحة لليمين	OF , CF
RCL	حمل الراسة الى الموقعين LSB-MSB	أعلام الحالة
RCR	حمل الراسة الى الموقعين LSB-MSB	أعلام الحالة

في هذه التعليمة نلاحظ انه القيمة في الموقع LSB يتم نقلها الى راية الحمل CF وان محتويات راية الحمل قبل تنفيذ التعليمة يتم نقلها الى الموق MSB والصيغة العامة كالتالي: RCR operand , operand2

في هذا المثال نقوم في السطر الأول بتعين العلم وملئه بالقيمة ١ ومن ثم تحريك من خلال التعليمة STC القيمة 1CH الى المسجل AL ومن ثم نقوم باستخدام تعليمة الحمل لليمين بمقدار خانة لتتحول القيمة في السجل AL من 00011100b الى 10001110b مع تغير قيمة الحمل في الراية الى صفر

```
STC
MOV AL, 1Ch
RCR AL, 1
RET
```

١٧- تعليمات التحكم بالراية:

وهي مجموعة من ثلاثة تعليمات تتحكم بحالة الراية قبل تنفيذ التعليمة أو بعدها راجع المثال السابق والذي استخدمنا فيه التعليمة STCوتتمثل هذه التعليمات بالجدول التالي:

التعليمة	المعنى	التأثير
CLC	مسح قمية الراية	CF = 0
STC	حمل قيمة الراية	CF = 1
CMC	الغاء الراية	CF= not CF

١٨- التعليمات المرتبطة بالسلاسل:

هذه المجموعة من التعليمات تستهدف عملية نقل بيانات من قيمة معينة داخل مسجل أصغر إلى قيمة ذات مسجل أكبر على سبيل المثال ولقد رأينا أنّ عملية النقل تتم من مسجل الى مسجل مشابه له مثلاً من مسجل ١٦ بت الى مسجل من نفس النوع أو من مسجل ٨ بت إلى مسجل من نفس النوع وهكذا أما بالنسبة لهذه التعليمات فأن الأمر مختلف يمكن على سبيل المثال نقل قيمة من مسجل الهدف من نوع ٨ بت إلى مسجل من نوع ١٦ بت والعكس غير صحيح لأن مسجل من نوع ٨ بت لا يمكنه تخزين قيم من نوع ١٦ بت بسبب قلة سعته التخزينية وهناك أمر هام جداً يتعلق بهذه التعليمات وهي تعامله مع الذاكرة بطريقة مباشرة أي يمكن نقل و إضافة بيانات تخزينية ضمن الذاكرة نفسها دون الرجوع إلى المسجلات بشتى أنواعها ومن أشهر هذه التعليمات :

١- التعليمة MOVS: تتعلق هذه التعليمة بعملية نقل بيانات رقمية من ذاكرة إلى ذاكرة ثانية أو بين المسجلات نفسها وهذا ما يجعلها مختلفة عن التعليمة MOV التي ذكرناها سابقاً وتتفرع من هذه التعليمة ثلاثة تعليمات أخرى وهي

- MOVSB حيث تتعامل بايت
- MOVSW تتعامل مع كلمة
- MOVSD تتعامل مع كلمة مزدوجة

```
MOV AX,٠١٠٠٠H     تحديد قيمة داخل المسجل
MOV DS,AX         نقل القيمة الى المسجل
MOV AX,٠٢٠٠٠H
MOV ES,AX نقل القيمة على المقطع الوجهة
MOV DI,٠
MOV SI,٠
MOV CX,٦٤ * ١٠٢٤D  السطرين التالين مسؤولين عن عملية تكرار العملية
REP MOVSW
END
```

٢- التعليمة CMPS: تطرح هذه التعليمة المقارنة بين عنصر الوجهة المعنون ب SI من عنصر المصدر المعنون ب DI ، كما تحدّث التعليمة SI و DI ليؤشروا إلى العنصرين التاليين من السلسلتين. لا تغير هذه التعليمة من محتوى المصدر أو الوجهة وإنما ُتحدث فقط حالة الرايات تبعًا للعلاقة بين عنصر المصدر وعنصر الوجهة. يمكن أن يكون طول كل من معامل المصدر والوجهة بايت أو كلمة.

٣- التعليمة LODS تنسخ هذه التعليمة بايتًا أو كلمة معنونة إلى AL أو AX ولها ثلاثة تعليمات متفرعة حيث يمكن نقل كلمة من خلال التعليمة LODSW أو بايت من خلال التعليمة الفرعية LODSB او كلمة مزدوجة من خلال التعليمة LODSD

```
CLD
```

```
MOV AX,٠٠٠٠H     نقل قيمة للمسجل
MOV DS,AX  تحديد المقطع الذي نتعامل معه
MOV CX,١٠٠٠H عدد مرات التكرار
MOV SI, ٠     تحديد بدء العملية من القيمة
MOV AH,٢H    طباعة القيمة ينطبق على الاكواد التي في الاسفل
START:
LODSB
MOV DL,AL
INT ٢١H
REP START
END
```

٤- التعليمة STOS تنسخ هذه التعليمة بايتًا أو كلمة من AL أو AX إلى العنصر من سلسلة المحارف المعنونة في الذاكرة

٥- التعليمة SCAS تطرح هذه التعليمة عنصر الوجهة المعنون ب DI من محتوى المراكم، وُتحدّث التعليمة DI ليؤشر إلى عنصر السلسلة التالي. لا تغيير هذه التعليمة من محتوى الوجهة أو المراكم وإنما تحدّث فقط حالة الرايات تبعًا للعلاقة بين عنصر المصدر والمراكم. يجري استخدام المراكم AL عندما يكون طول معامل الوجهة ١ بايت، وAX عندما يكون طول معامل الوجهة ٢ بايت.

٦- التعليمة PUSHF: تستخدم هذه التعليمة من أجل تخزين محتوى الراية داخل المكدس بما أن التعليمة بقوم بدفع المسجلات إلى المكدس فإن هذه التعليمة تساهم في دفعها مع الرايات كما في المثال التالي فيكون دفع كل من المسجل AX ,C,BX مع الراية

```
PUSHF
PUSH AX
PUSH C
PUSH [BX]
```

٧- التعليمة POPF: أبضا تستخدم هذه التعليمة من أجل تخزين محتوى الراية داخل المكدس المثال التالي نفس المثال السابق بالضبط

```
POPF
POP AX
POP C
POP [BX]
```

٨- التعليمة IMUL حيث تعمل على ضرب قيمتين الأولى منها إما ان تكون مخزنة في مسجل او موقع في الذاكرة طوله بايت او كلمة والمعامل الثاني اما أن يكون المسجل AL او AX وذلك بناء على طول المعامل الأول ومن الجدير بالذكر أنه من غير الصواب ضرب قيمة فورية مباشرة فالعملية IMUL 10 هي عملية غير صحيحة ولإنجاز مثل هذه العملية يجب تخزين القيمة في مسجل او موقع في الذاكرة أولا ومن ثم يتم إنجاز العملية وكما يلي:

```
Org 100h
MOV AL, -2
MOV BL, -4
IMUL BL     ; AX = 8
RET
```

٩- التعليمة IDIV: عمل هذه التعليمة مشابه للتعليمة السابقة حيث يتم قسمة قيمتين في مسجل حسب طول المسجل ويمكن ضرب علمين ببعضهما وتتأثر كل من اعلام CF,OF

```
org 100h
mov ax , 9
mov bx , -3
idiv bx
Ret
```

١٠-التعليمة INS: ترتبط هذه التعليمة بالعتاد الصلب للحاسب وتكمن أهميتها في كونها تساعد على ادخال معطيات إلى وحدات الادخال للحاسب الآلي تتعامل هذه التعليمة مع مقاطع الذاكرة دون وسيط ومثال على ذلك إذا علمت أن عنوان البوابة الفرعية LTP1 للحاسوب الشخصي هو 358h فيمكنك قراءة هذه البوابة كما يلي:

```
MOV DX, 358h      استدعاء عنوان البوابة
IN AL, DX         قراءة البيانات
```

تتفرع من هذه التعليمة ثلاثة تعليمات فرعية خاصة كسابقتها تحدد طول البيانات المراد نقلها سواء أكانت بايت INSB او كلمة INSW أو كلمة مزدوجة INSD

١١-التعليمة OUT: نفس سابقتها ترتبط هذه التعليمة بالعتاد الصلب للحاسب وتكمن أهميتها في كونها تساعد على ادخال معطيات إلى وحدات الاخراج للحاسب الآلي

تعليمات القفز المشروط:

يبلغ عدد تعليمات القفز المشروطة ثمانية عشر تعليمة وترتبط بشكل كلي بنتيجة سابقة قام الحاسب بالتحقق منها وهي اشبه بالجمل الشرطية في لغات البرمجة العليا وتستخدم هذه التعليمات كمفتاح اساسي لأقفال العديد من التطبيقات كما تأخذ حيزاً من التفكير الطويل قبل اتخاد القرار بتعديل العمليات الحسابية التي تسبقها لإنهاء القفز أو لإتمام عملية القفز طبعا سنقوم بشرح واسع لهذه التعليمات لاحقاً في التطبيق النظري

الحدث	التعليمة
القفز إذا كان CF = 1	JC
القفز إذا كان CF = 0	JNC
القفز إذا كان OF = 1	JO
القفز إذا كان OF = 0	JNO
القفز إذا كان SF = 1	JS
القفز إذا كان SF = 0	JNS
القفز إذا كان CX = 0000	JCXZ
القفز في حالة التساوي/أو إذا كان الناتج يساوي الصفر	JE/JZ
القفز إذا كان أكبر أو يساوي/القفز إذا لم يكن أصغر	JGE/JNL
القفز إذا كان فوق/القفز إذا لم يكن تحت أو يساوي	JA/JNBE
القفز إذا كان فوق أو يساوي/القفز إذا لم يكن تحت	JAE/JNB
القفز إذا كان تحت/القفز إذا لم يكن فوق أو يساوي	JB/JNAE
القفز إذا كان تحت أو يساوي/القفز إذا لم يكن فوق	JBE/JNA
القفز إذا كان أكبر/القفز إذا لم يكن أصغر أو يساوي	JG/JNLE
القفز إذا كان أصغر أو يساوي/القفز إذا لم يكن أكبر	JLE/JNG
القفز إذا لم يكن يساوي/القفز إذا كان الناتج يساوي قيمة غير صفرية	JNE/JNZ
القفز إذا كانت خانة Parity غير موجودة/القفز إذا كان PF = 0	JNB/JBO
القفز في حالة وجود خانة Parity/القفز إذا كان PF = 1	JP/JPE

تحدث تعليمة القفز استجابة لعدة أوامر شرطية خاصة في الاسطر الخاصة بعملية فحص المسجلات وقيمتها العددية ولا تتم العملية حتى يتحقق الشرط الكثير من المبرمجين العكسين يركزون على عمليات القفز لعدة اسباب منها لإحداث تغيرعلى فحص نافذة ما أو تخليص برنامج معين من عدد من المكاتب التي لا يحتاجها أو لتلافي

رسالة تسجيل البرنامج أو مربعات حوار إعلامية من خلال استخدام ما يعرف بالقفزات الغبية كذلك لتنقيح برنامج يحتوي على رسالة خطأ معينة

مثال على القفز عندما يكون flag=fz راقب مربع الاعلام في المحاكي عند تطبيق التعليمات البرمجية التي في الأسفل

```
.model small
.data
.code
 main proc
  mov ax, 5
  cmp ax, 5
  jz  lp1
   mov bx, 1
  lp1:
   mov bx, 6
 Endp
 end main
```

مثال عن القفز في حال التساوي

```
.model small
.data
.code
 main proc
 mov ax, 100
  cmp ax, 100
  je  lp1
   mov bx, 1
  lp1:
   mov bx, 6
 Endp
 end main
```

١٩- تعليمات تعريف القيم (المتغيرات):

تلعب هذه التعليمات نفس الدور الذي تقوم المتغيرات بلعبه في لغات البرمجة العليا حيث تقوم بتعريف قيم عددية او نصية داخل البرنامج وتقسم هذه التعليمات الى عدة انواع

التعليمة الفرعية	الحدث

DB (Define Byte)	تعريف متغير من نوع خانة واحدة
DW (Define Word)	تعريف متغير من نوع خانتين
DD (Define Double Word)	تعريف متغير من نوع أربع خانات
DQ (Define Quad Word)	تعريف متغير من نوع ثمان خانات
DT (Define Ten Bytes)	تعريف متغير من عشر خانات

مثال عملي عن كيفية استخدام المتغيرات في حالة كتابة كود بلغة الاسمبلي ٨٠٨٦ كما في المثال التالي والذي قمنا بتعريف المتغير var1 وضعنا به القيمة ٢ طبعا طريقة تعريف المتغير هي شبيهة تماما بلغات البرمجة الأخرى حيث أن المتغير يكون حاوية لحفظ أعداد او أرقام

```
.model small
.data
var1 db 2
.code
main proc
mov dl, var1
add dl, 48
mov ah, 2h
int 21h
endp
```

```
end main
```

٢٠- التعليمات المنطقية:

وهي مجموعة من التعليمات المستخدمة في العديد من لغات البرمجة الاخرى والتي تخلق نوع من الديناميكية والمرونة في عملية برمجة التطبيقات في مختلف لغات البرمجة واي من مستخدمي لغات البرمجة حول العالم له معرفة اكيدة بها وسوف نقوم بدراستها سريعا وكيفية التعامل معها في المجمع وللعلم فإن هذه البرمجة خاصة بتطبيقات تصميم البرامج بواسطة لغة اسمبلي ومحرراتها والتي سيتم شرحها لاحقا واغلبها لا يدخل في مجال المنقحات

١- التعليمة THEN-IF الشرطية:

تعتبر هذه التعليمة الشرطية أشهرها والتي يتم الاستعاضة عنها بالمنقح بتعليمة القفزات وذلك من أجل تحقيق شرط ما داخل البرنامج وسنقوم بإعطاء مثال بسيط عن هذه التعليمة حيث يتم العمل فيها تحت برنامج MASM او احد التطبيقات الخاصة للبرمجة بلغة الاسمبلي:

```
Processor = 80386; Set to 8086 for 8086-only code….

if Processor eq 80386

shl ax, 4

else ;Must be 8086 processor.

mov cl, 4

shl ax, cl

endi
```

٢- التعليمة ELSE الاستثناء:

تأتي هذه التعليمة بالتكامل مع التعليمة الشرطية السابقة IF وذلك للتحقق من مجموعة من الشروط قبل تنفيذ العبارة البرمجية الصحيحة

٣- التعليمة UNTIL – REPEAT:

وتشكل هذه التعليمة تكرار مجموعة من الاحداث البرمجية حتى يتحقق الشرط الصحيح في النهاية الممثل بتعليمه UNTIL التي تأتي هنا أخر العبارة البرمجية في اغلب الاوقات

٤- التعليمة WHILE:

وتختص هذه التعليمة بتكرار مجموعة من الاحداث مادام الشرط لهذه التعليمة متحقق وعند انتهاء الشرط ينتهي التكرار وهذه عكس التعليمة السابقة REPEAT لاحظ بنفسك

```
MOV EAX,3
WHILE EAX>BCX
DEC 1
ENDW
```

٥- التعليمة OR:

تستخدم هذه التعليمة لتحقيق صيغة شرطية بين أثنين من الصيغ وفي حالة تحقق واحدة منها فإن البرامج ينتقل الى الحدث التالي طبعاً كون احد الحدثين أو الشرطين صحيح التعليمة المنطقية OR تمثل عملية مقارنة بين قيمتين تأخذ التعليمة أحد الشرطين صح أو خطأ فيها بحسب العملية الحسابية ففي حال كان احد الطرفين أو كلاهما يمثل TRUE فإن العملية تكون صحيحة وإذا كانت كلاهما خاطئة فالعملية الشرطية تكون خاطئة FALSE كما هو العادة في لغات البرمجة الأخرى في حالة تمثيلها للشرطين (TRUE-FALSE) تتمثل بالرقمين ٠ و ١ ولنتعلم كيفية تعامل الاسمبلي مع عمليات تتضمن هذه التعليمة المنطقية لاحظ التالي :

قيمة المسجل AL تمثل الرقم 63H بالنظام الستة عشري

قيمة المسجل BL تمثل الرقم 4BH بالنظام الستة عشري

نتيجة العملية ستكون بالتأكيد هي 6BH حاول التأكد منها وتحويلها بالآلة الحاسبة وللعلم فإن عملية التمثيل ستكون بالنظام الثنائي حصراً كسائر العمليات الأخرى من أجل الفهم الصحيح لاحظ الجدول التالي

AL	BL	AL+BL (مع تعليمة OR)
١	١	١
١	٠	١
٠	٠	٠
٠	١	١
٠	٠	٠
١	١	١
١	١	١

السطر الأول يتضمن قيمتين صحيحة فالجواب يكون صحيح كم أشرنا سابقاً السطر الثاني يتضمن قيمة واحدة صحيحة فالجواب صحيح كذلك أي ١ أما بالنسبة للسطر الثالث فالقيمة خاطئة فالجواب يكون خاطئا أي صفر وقيمة الجول السابق متمثلة بالنظام الثنائي لقيمة مسجلي من نوع ٨ بت لاحظ المثال التالي والذي يعتبر تطبيق عملي على المحاكي ركز على النتيجة وحالة الأعلام

```
.model small
.data
.code
 main proc
  mov ah, 00000100b
```

```
mov bh, 00000011b
or ah, bh
endp
end main
```

٦- التعليمة AND:

تعتبر هذه العملية المنطقية معاكسة لعملية OR حيث يتم اختيار الشرط الخاطئ في حال كان أحد الطرفين خاطئ أو صحيح وفي حالة كان الطرفين خاطئ فالنتيجة تكون خاطئة كذلك أما في حالة كان الطرفين صحيح فالنتيجة تكون صحيحة وهذا معاكس تماماً للتعليمة المنطقية السابقة ولتقريب الصورة أكثر لاحظ المثال التالي وهو يعتبر نفس المثال السابق لكن التغير في النتيجة النهائية:

قيمة المسجل AL تمثل الرقم 63H بالنظام الستة عشري

قيمة المسجل BL تمثل الرقم 4BH بالنظام الستة عشري

النتيجة النهائية للتعامل الحسابي مع هذه التعليمة يكون ٤٣ جرب العملية على الآلة الحاسبة

(مع تعليمة AND)	BL	AL
١	١	١
٠	٠	١
٠	٠	٠
٠	١	٠
٠	٠	٠
١	١	١

١	١	١

طبعاً عملية التعلم تتطلب أخذ كمية كبيرة من الأمثلة بالرغم من سهولة هذا النوع من التعليمات المنطقية إلا أن الوضع سوف يختل بمجرد التعامل مع التعليمات في الهندسة العكسية والسبب هو الكم الهائل من المعلومات التي سوف تحصل عليها في حالة تنقيح

المسجل EAX يحتوي على القيمة 45H بالنظام الست عشري

المسجل ECX يحتوي على القيمة 3FH بالنظام الست عشري

بالنسبة لقيمة عملية الحسابية مع استخدام التعليمة 45H لاحظ التمثيل بالنظام الثنائي

(مع تعليمة AND)	ECX	EAX
١	١	١
٠	١	٠
٠	١	٠
٠	١	٠
١	١	١
٠	١	٠
١	١	١

المثال التالي يشرح عمل and بلغة الاسمبلي من خلال عمل برنامج صغير يشرح عمل التعليمة على المحاكي

```
.model small
.data
.code
 main proc
  mov ah, 00000101b
  mov bh, 00000001b
    and ah, bh
 endp
end main
```

٧- التعليمة NOT:

هذه التعليمة تختلف عن بقية التعليمات الرياضية بتعاملها مع مؤثر واحد فقط حيث أن التغير يطرأ عليه مباشرة وتقوم هذه التعليمة بعكس الشرط فإذا كانت قيمة الشرط صحيحة أي تساوي الواحد فالنتيجة تكون خاطئة وإذا كانت قيمتها خاطئة فالجواب يكون صحيح ولنفهم أكثر لاحظ التالي:

لدينا التعليمة التالية NOT AX مع العلم بأن AX يساوي 47H فكم تكون النتيجة النهائية لهذه التعليمة لاحظ الجدول التالي

استخدام NOT	AX
٠	١
١	٠
١	٠
١	٠

٠	١
٠	١
٠	١

لاحظ في الجول السابق كيف تم قلب الشطر والنتيجة النهائية لهذه العملية هو 38H بالنظام الستة عشري وتم التغير على مؤثر واحد

```
.model small
.data
.code
 main proc
  mov ah, 01111110b
  not ah
 endp
end main
```

٨- التعليمة XOR:

طبعاً هذه التعليمة تقوم بفحص الأرقام بالنظام الثنائي ففي حالة كانت هذه الأرقام تشابه القيمة كعمليتين صحيحتين (يعني الرقمان هما واحد) أو حتى خاطئتين (الرقمان هما الصفر) فإنه يشير هنا الى الخطأ وفي حالة كانت مختلفتين واحدة خاطئة وواحدة صحيحة فإنه يشير إلى الجوا الصحيح ولنفهم العملية الحسابة بشكل أفضل سنقوم بأخذ هذا المثال:

قيمة المسجل AL تمثل الرقم 63H بالنظام الستة عشري

قيمة المسجل BL تمثل الرقم 4BH بالنظام الستة عشري

القيمة النهائية للمسجل ستون 28H النظام الستة عشري استخدم الآلة الحاسبة البرمجية

(مع تعليمة XOR)	BL	AL
٠	١	١
١	٠	١
٠	٠	٠
١	١	٠
٠	٠	٠
٠	١	١
٠	١	١

كما تلاحظ في الجول في حالة تشابه الشرطين النتيجة خاطئة وفي حالة اختلافهما فالنتيجة صحيحة ولنفهم أكثر سنقوم بعرض المثال التالي والذي يمثل عمل تعليمة xor

```
.model small
.data
.code
 main proc
 mov ah, 11111111b
   mov bh, 11111110b
   xor ah, bh
```

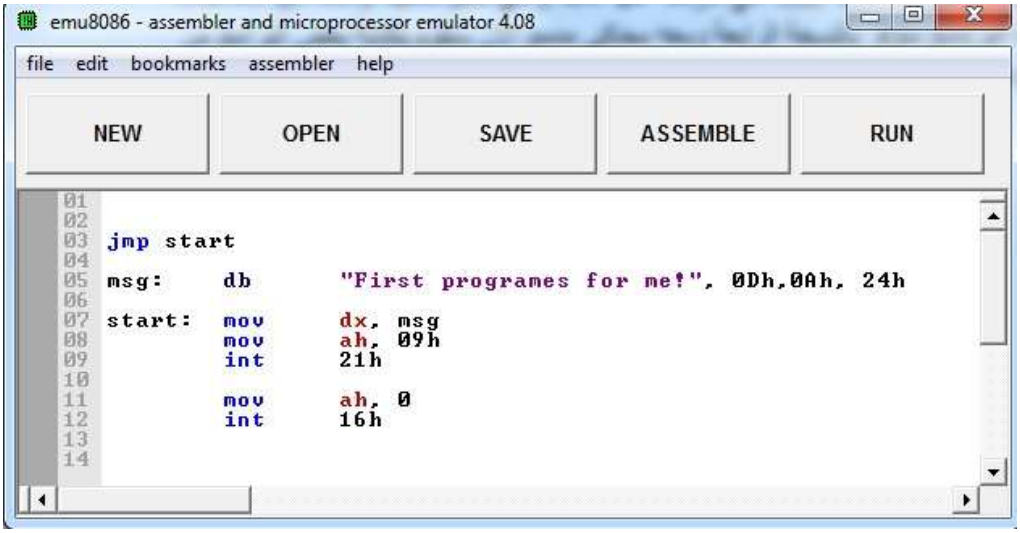
```
endp
end main
```

- **البرمجة بالأسمبلي تحت بيئة MSDOS:**

طبعا هذا النوع من البرمجة يحتاج إلى محاكي وبرنامج خاص وسنستخدم EMU8086 الخاصة بالبرمجة على المعمارية X86 التي شرحنا عنها سابقا والتي تمثل تشكيلة واسعة من المعالجات طبعا البرنامج متوفر بالنسخة الرابعة ومعه محاكي مدمج الأن سنقوم بكتابة بعض البرامج من أجل فهم عملية كتابة الاسطر البرمجية بلغة المجمع ولكن قبل ذلك اشير على التالي وهو أن جميع الاكواد البرمجية التي تكتب في المحاكي والتي ذكرناها سابقا والتي سنذكرها لاحقا تحوي الأمر int21 والذي يمثل استدعاء مقاطعة من أجل تشغيل البرنامج داخل نظام التشغيل Dos و طباعة العمليات البرمجية داخل صندوق مؤشر النوافذ في نظام التشغيل من خلال المحاكي

البرنامج الأول:

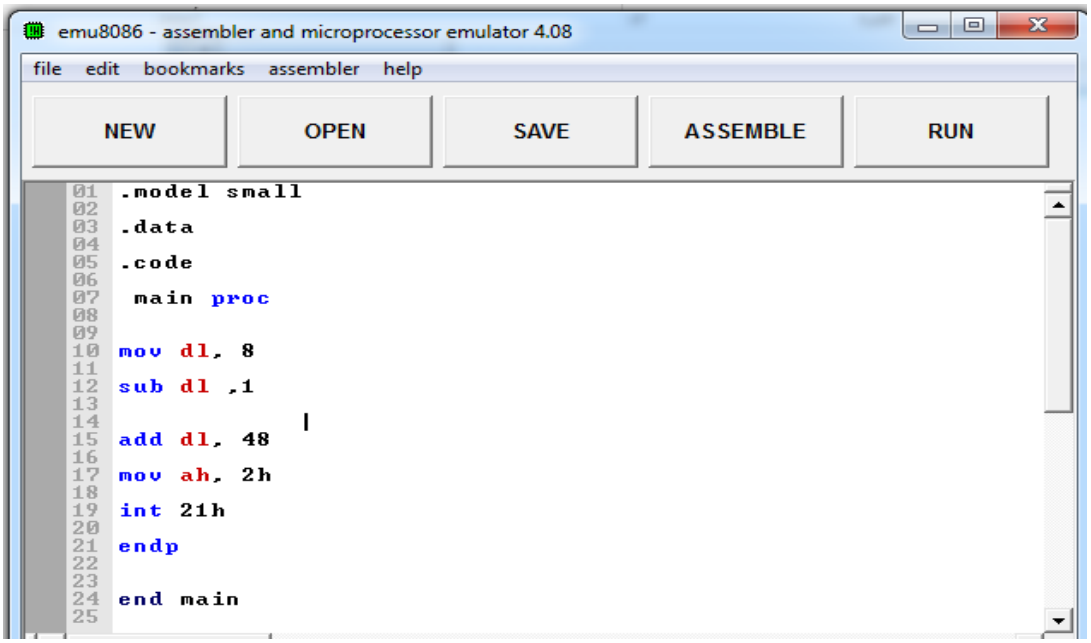
سنقوم بعمل رسالة باللغة الإنكليزية مكتوب داخلها هذا برنامجي الأول طبعا السطر الأول يتم فيه القفز إلى كتلة start والتي تحوي مجموعة من الأسطر من ٧ إلى ١٢ وتحوي على نقل رسالة إلى مسجل dx ونقل القيمة 21h إلى المسجل ah ويتم تنفيذها أولاً من ثم عرض الرسالة



البرنامج الثاني:

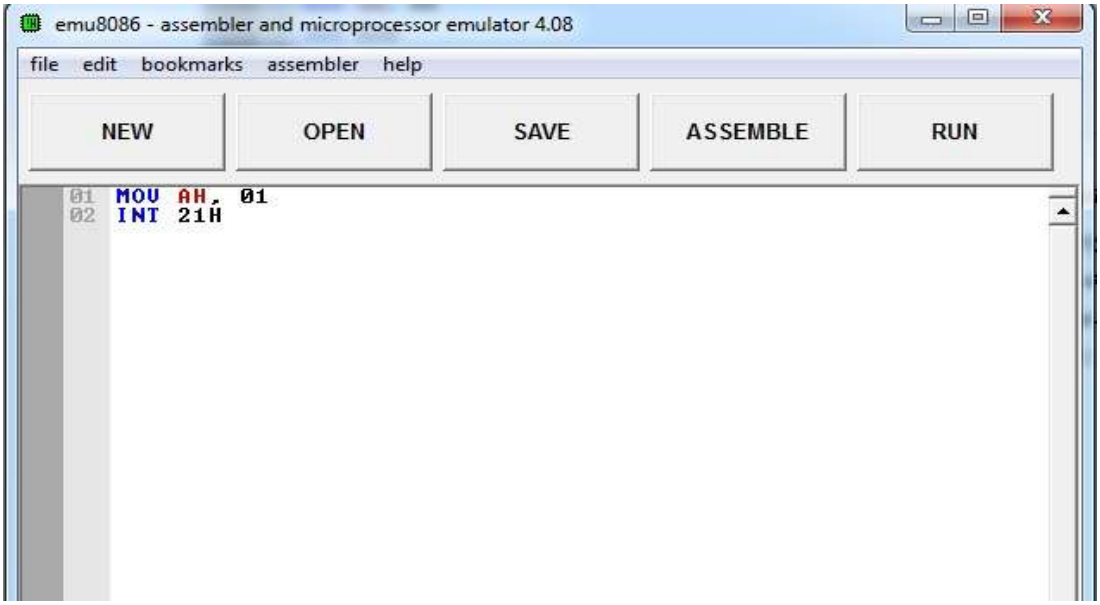
البرنامج الثاني جميل جدا وهو يحتوي على التالي عملية إضافة الرقم ثمانية إلى المسجل dl من ثم القيام بعملية طرح القيمة واحد من المسجل ومن ثم تأتي كتلة البرمجية من سطر ١٥ وحتى السطر الأخير والتي يتم من خلالها حجز المسجلات المسؤولة عن عرض النتيجة طبعا بالنسبة للتعليمة int فهي عبارة عن استدعاء للمقاطعة المسؤولة عن طباعة النتيجة أيضا ضمن نفس الكتلة وهذا خاص بنظام التشغيل dos أي أن هذه الاستدعاء لا ينطبق على نظم تعمل وفق البيئة win32

في بداية السطر تم تعريف البرنامج من خلال كتابة model small يمكن الاستغناء عن هذا السطر فالمحاكي يتولى تعريف البرنامج بشكل تلقائي ومن ثم قسم dataقسم بيانات البرنامج ثم القسم code الخاص بشيفرة البرنامج من ثم proc أي العملية والموجودة ضمن الكتلة الرئيسة لتنفيذ البرنامج بالنسبة للتعليمة int21 فهي خاصة بتعريف أن البرنامج يعمل على نظام التشغيل Dos وهي خاصة بهذا النظام فقط ويترتب من عملية استدعائها مجموعة من الوظائف



البرنامج الثالث:

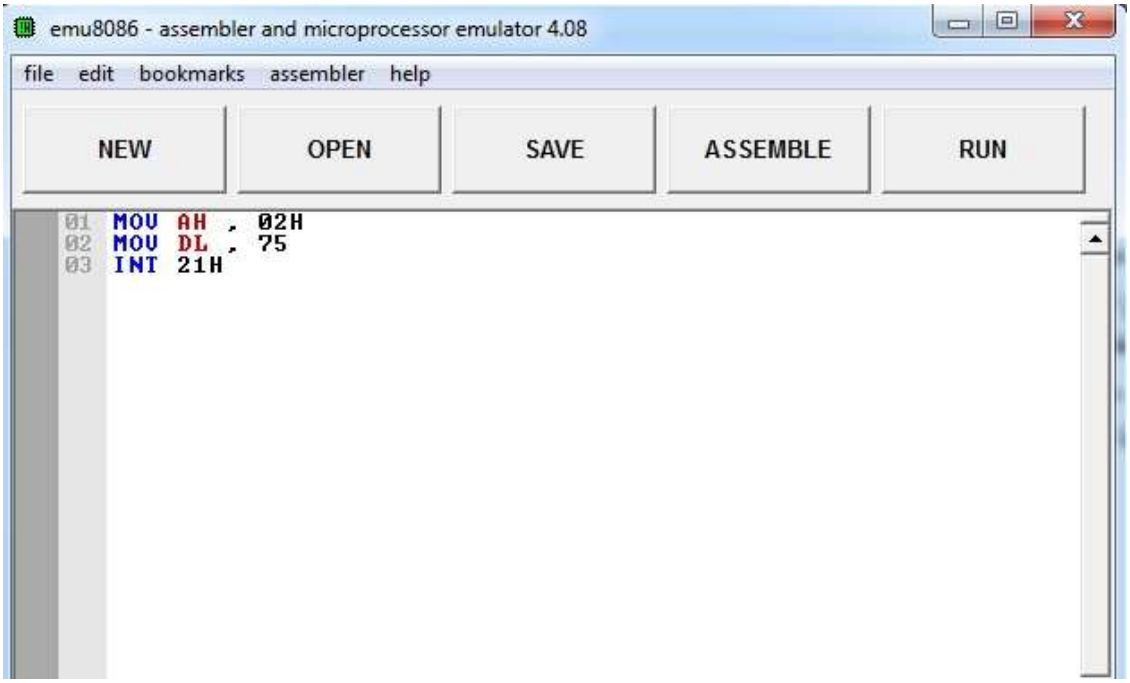
سنقوم في هذا البرنامج باستدعاء حرف من جدول الاسكي ASCII من خلال لغة المجمع كما نعرف فإن لكل حرف معرف رقمي خاص به كما أشرنا إلى ذلك في الكتاب نفسه لكن كيف نستدعي هذا الحرف إلى البرنامج تابع معي هذه الاسطر أولا سنقوم بالذهاب إلى المسجل AH وهذا المسجل خاص بالبيانات أي معرف الاسكي طيب إذا أردنا بناء برنامج يطبع الحرف الذي قمنا بضغطه على زر لوحة المفاتيح فنقوم بكتابة التالي



لاحظ أننا قمنا بالانتقال إلى مسجل AH ثم بعد ذلك وضعنا القيمة 01 والتي تمثل بكل بساطة قيمة كل المفاتيح ذات الأرقام والأحرف باستثناء المفاتيح مثل F1وبF2 وغيرها فقيمتها هي الصفر ثم استدعينا المقاطعة 21H الخاصة بنظام التشغيل DOS كما شرحنا سابقا قم بتشغيل البرنامج ثم اطبع أي حرف لترى النتيجة يجب حفظ هذه الأكواد فهي مهمة في عملية البرمجة

البرنامج الرابع:

الآن سننتقل إلى البرنامج 02 ضمن جدول الاسكي والخاص بطباعة حرف معين نحن نختاره مسبقا دون اللجوء إلى لوحة المفاتيح وسنقوم بكتابة الأسطر التالية والتي تمثل هذه العملية



لاحظ في المثال التالي استدعينا البرنامج 02 في المثال السابق كان 01 واحد لكن عمل برنامج كان مختلفا فالبرنامج 02 مكلف بطباعة الحرف بدون اللجوء للوحة المفاتيح من خلال عملية تعريفه بشكل مباشر بينما البرنامج 01 مكلف بتنفيذ ما يمليه عليه لوحة المفاتيح

البرنامج الخامس:

ضمن البرنامج الخامس سنقوم بشرح بعض العمليات التي تنطبق على البرنامج ضمن إطار السطر MOV AH ,02 أو ضمن الاستدعاء 02 لنكون دقيقين أكثر حيث يمكن إجراء التالي

الرقم	الوظيفة
7	اصدار صوت Beep
8	ترك مسافة
9	تحرك بمقدار خطوة
A	سطر جديد

D	بداية السطر الحالي

ولكتابة برنامج يقوم بعمل صوت عند فتحه فنقوم بكتابة الاسطر التالية وجميعها صارت مفهومة

- **البرمجة باللغة الاسمبلي تحت بيئة WIN32:**

طبعا تستخدم مجموعة من البرامج المجانية او التجارية من اجل تصميم تطبيقات بهذه اللغة ونمط عملها يعتمد بالدرجة الاولى على ما يسمى (linker) والذي يقوم بتحويل النص البرمجي الى لغة الآلة وحفظ هذا التطبيق كملف تنفيذي ومن هذه البرامج التي تستخدم بكثرة البرمجة بالاسمبلي تحت هذه المنصة مختلفة كليا حيث أننا نعتمد بالغالب على مكتبات نظام التشغيل فيها من أجل تنفيذ البرنامج وهي تختلف عن الاسمبلي العادي كون الأول يربط لغة البرمجة بالهاردوير بشكل مباشر عبر نوافذ سوداء وتعتبر تمثيل عملي وصريح للغة الآلة بينما الثانية تستخدم الدوال والمكتبات داخل نظام التشغيل Windows من أجل صنع واجهة عملية تلائم نظام النوافذ وتعمل وفق بيئة مايكروسوفت بدون مشاكل

١- **برنامج MASM:** يعتبر من البرامج الرائدة في هذا المجال حيث يتم استخدامه من قبل المبرمجين في مجال الهندسة العكسية هذا البرنامج نسخة مرخصة لشركة

مايكروسوفت وقد اصدرت له نسخة خاصة لأنظمة ٣٢بت بواجهة مرئية كالموجودة بالفيجوال بيزك مثل إضافة الازرار والقوائم ومربعات الحوار والبرنامج مطروح من قبل الشركة على الموقع http://www.masm32.com واخر اصدار هو الثامن

٢- **برنامج TASM:** يستخدم هذ البرنامج كسابقه في عملية تصميم تطبيقات بلغة الاسمبلي هذا المنتج أطلقته شركة بورلند الرائدة في مجال البرمجة الحوسبية والتي أطلقت اللغة الشهيرة دلفي وبكل الاحوال فإن النسخة الخاصة بهذا البرنامج تجارية وتحتاج الى شراء من الشركة الام

٣- **برنامج NASM:** يعتبر هذا البرنامج تطبيق مفتوح المصدر ومجاني تم تصميمه من قبل مطورين ليلعب نفس الدور التي يقوم بها تطبيقات اخرى مماثلة ويمكن الحصول عليه من خلال زيارة الرابط التالي
http://sourceforge.net/projects/nasm

ادوات تحرير النصوص البرمجية:

يوجد عدد كبير من المحررات حول العالم والتي تستخدم كأدوات برمجية متعددة المنصات أو متعلقة بلغة برمجية معينة كالأسمبلي ومن أشهرها واكثرها استخداما:

١- برنامج المفكرة NOTEPAD: وتأتي كنسخة مدمجة داخل اقراص ويندوز ويعتبر من أسهلها استخداما ويمكن كتابة اكواد برمجية داخله إلا انه يعتبر من المحررات التي تفتقر الى ميزات كثيرة تجعله مرنا وأكثر تنظيما مقارنة مع غيره من المحررات

٢- محرر :VISUAL STUDIO يعتبر من المحررات الهامة والذي يأتي غالبا مع نسخ ويندوز الموسعة أو حتى مع برنامج الفيجوال بيزك العملاق في مجال تصميم تطبيقات الحاسب مقارنة بالمفكرة فإن هذ المحرر يحقق تقدماً كبيراً في مجال الميزات اللامحدودة كتنظيم النص البرمجي والدوال المدمجة داخله والتلوين النصي والعديد من المميزات الاخرى

٣-محرر ULTRAEDIT: من اقوى البرامج في هذا المجال واكثره شعبية وفيه الكثير من المميزات التي تجعله من المحررات المفضلة لديك يمكن الحصول عليه من موقعه الاساسي على الانترنت http://www.ultraedit.com

كتابة الشيفرات البرمجية في لغة الأسمبلي:

يتم تقسيم البرامج في لغة المجمع الى ثلاثة اقسام تمثل الطريقة المنتظمة للكتابة بهذه اللغة:

١-قسم البيانات DATA: ويتم من خلال هذا القسم تعريف الثوابت أي بيانات غير قابلة للتغير وتشمل على سبيل المثال اسماء الملفات أو قيم عددية ثابتة

```
section .data
```

٢-قسم المتغيرات BSS: ويشمل هذا القسم تعريف المتغيرات الخاصة بالبرنامج وكما درسنا سابقاً فأن المتغير هو عبارة عن حاوية لقيم عددية او سلاسل نصية ويعرفها جميع من لديه اطلاع على لغات البرمجة الاخرى بشكل عام

```
section .bss
```

٣-القسم TEXT: ويستخدم من اجل وضع الشفرة الاساسية للبرنامج ويجب أن يبدأ هذا القسم بعبارة (**global main**) والذي يخبر نواة النظام اين سيتم تنفيذ البرنامج

```
section .text
```

```
global main
main:
```

الإشارات الحسابية:

وهي تعني العمليات الحسابية التي كنت تستخدمها منذ الصغر من قسمة وطرح وجمع وضرب والعديد من العمليات الأخرى المعقدة أو حتى البسيطة وتستخدم في إطار الاسمبلي ٣٢ بشكل رئيسي إضافة إلى تفرعات الاسمبلي الأخرى

١- إشارة أكبر من (>) إشارة الضرب (*)

٢- إشارة أصغر من (<) إشارة القسمة (/)

٣- إشارة أكبر ويساوي (=>) إشارة السالب (-)

٤- إشارة أصغر ويساوي (=<) إشارة الجمع (+)

٥- إشارة يساوي (=)

البرنامج الأول بلغة اسمبلي ٣٢

سنقوم ببرمجة رسالة مرحبا بالعالم من خلال استخدام أداة Nasm سنقوم في البداية باستدعاء المكتبات المطلوبة وهي مكتبة

Kernel32.dll ومن ثم الدالة ExitProcess

User32.dll ومن ثم الدالة MessageBoxA

من ثم نقوم بتعريف متغير يحتوي رسالة Hello world من خلال استدعاء الامر db كما شرحنا سابقا نقوم من ثم بتعريف لاحقة المستند والبدء ببرمجة مربع الحوار على النحو التالي

push dword MB_OK زر موافق وسط مربع الحوار

push dword title عنوان مربع الحوار

push dword message النوع رسالة مباشرة

push dword 0 إعطاء المربع رقم تعريفي

الآن سنقوم باستدعاء أمر رسالة ومن ثم دفعه وإعطاء الامر خروج في حالة ضغط المستخدم زر موافق من خلال الأسطر التالية

call [MessageBoxA]

push dword 0

call [ExitProcess]

ليكتمل عندنا نص البرنامج ويصبح كالتالي:

```
; This line is a comment.
include "win32n.inc"
extern MessageBoxA
import MessageBoxA user32.dll
extern ExitProcess
import ExitProcess kernel32.dll
segment .data USE32
title db "hello",0
message db "hello world",0

segment .bss USE32
var1 resb 32
```

```
segment .code USE32
..start:
push dword MB_OK
push dword title
push dword message
push dword 0
call [MessageBoxA]
push dword 0
call [ExitProcess]
```

تم بحمد الله

• المراجع

- **Microprocessor**
- **Art Of Intel x86 Assembly**
- **Assembly Language step by step**
- **Reverse Engineering**
- **Software reverse engineering education**
- **Tiny guide to x86 assembly**

بعض المراجع والمقالات المتفرقة باللغة العربية

❖ إهداء الكتاب إلى

- امي العزيزة ووالدي الحبيب
- إلى اخوتي
- الى اخي الغالي وابن خالي الشهيد الدكتور انس عصام خليفة
- والى جميع اهلي واقاربي