اسم المادة : هيكليةالحاسوب ولغة أسمبلي

تجمع طلبة كلية التكنولوجيا والعلوم التطبيقية - جامعة القدس المفتوحة

acadeclub.com

وُجد هذا الموقع لتسهيل تعلمنا نحن طلبة كلية التكنولوجيا والعلوم التطبيقية وغيرها من خلال توفيروتجميع كتب وملخصات وأسئلة سنوات سابقة للمواد الخاصة بالكلية, بالإضافة لمجموعات خاصة بتواصل الطلاب لكافة المواد:

للوصول للموقع مباشرة اضغط هنا

وفقكم الله في دراستكم وأعانكم عليها ولا تنسوا فلسطين من الدعاء ♡

بسم الله الرحمن الرحيم

هيكلية الحاسوب ولغة التجميع (أسمبلي)

الوحدة الأولى :هيكلية الحاسوب

***مهام الحاسوب:**

١ .إدخال البيانات Data والبرامج. programs

٢ .تخزين البيانات والبرامج.

٣ .معالجة البيانات بواسطة البرامج وتخزين النتائج .

٤ .إخراج البيانات والبرامج بالشكل المناسب.

****لكي يتمكن الحاسوب من أداء هذه المهام يجب أن يحتوي على الوحدات التالية-:**

- **وحدة إدخال** تستخدم لنقل المدخلات من العالم الخارجي إلى الحاسوب.
- **وحدة ذاكرة** تستخدم لتخزين البيانات والبرامج قبل المعالجة وكذلك لتخزين النتائج بعد المعالجة .
- **وحدة معالجة :** تستخدم لتنفيذ البرامج على البيانات وتحويل البيانات إلى معلومات.
- **وحدة إخراج :** تستخدم لعرض البيانات والبرامج والمعلومات.

***كيفية تمثيل البيانات والبرامج في داخل الحاسوب:**

لكي يستطيع الحاسوب أن يتعامل مع البيانات والبرامج الممثلة بلغات إنسانية يجب أن تحول هذه البيانات والبرامج إلى لغة الآلة . تتكون البرامج المكتوبة بلغة الآلة من أصفار وواحدات (1,0) فقط . وباختصار يجب تحويل القيم العددية والرمزية والرسومات ،والبيانات والبرامج إلى قيم ثنائية مكونة من أصفار وواحدات فقط . (الاطلاع على شكل 1 صفحة ٧ ، المخطط الصندوقي للحاسوب).

****نظام : EBCDIC** نظام شيفرة يستخدم 8 ثنائيات لتخزين أي رمز ويستخدم غالبا في الحواسيب الكبيرة.

****نظام : ASCII** نظام شيفرة يستخدم 7 أو 8 ثنائيات لتخزين أي رمز ، ويستخدم غالبا في الحواسيب الميكروية.

****وحدة المعالجة المركزية :Central Processing Unit CPU:**

المعالجة الفعلية للبيانات تتم في وحدة المعالجة المركزية(وتسمى في الحواسيب الدقيقة بالمعالج الميكروي) .

**** تتكون وحدة المعالجة المركزية من ثلاث وحدات أساسية هي:**

- **وحدة الحساب والمنطق "Arithmetic & Logic Unit" ALU**

تقوم وحدة الحساب والمنطق بتنفيذ العمليات الحسابية مثل : الجمع، الطرح، الضرب، القسمة والعمليات المنطقية مثل : الضرب المنطقيAND ، الجمع المنكقيOR ، النفيNOT ، والمقارنة. تتكون وخدة الخساب والمنطق من دارة جامع (adder)واحدة. غير أن بعض أنواع الحواسيب يمكن أن تحتوي على العديد من هذه الدارات.

- **المسجلات : "Registers" REG** وحدة ذاكرة تمتاز بسرعتها العالية بالمقارنة بوحدات الذاكرة الأخرى وتستخدم غالبا في ال. CPU

*** تمتاز السجلات بأنها** عبارة عن مواقع تخزينية ذات سرعة عالية مقارنةً مع مواقع التخزين في وحدات الحاسوب الأخرى

**** تستخدم المسجلات عادةً لتخزين :**

- القيم المشاركة في العمليات التي تنفذ في وحدة الحساب والمنطق .
- القيم الناتجة عن العمليات التي تنفذ في وحدة الحساب والمنطق.
- تعليمات البرنامج أو أجزاء منها.

*تختلف المسجلات في الحواسيب من حيث: عددها، تسمياتها، أطوالها، واستخداماتها.

***يمكن تصنيف المسجلات في وحدة المعالجة المركزية حسب استخداماتها إلى نوعين:**

١.**مسجلات عامة الأغراض** : **General purpose Registers** تستخدم من قبل المبرمجين لتخزين جميع أنواع البيانات **ومن أمثلتها :** مسجلات الفاصلة الثابتة (Fixedpoint Registers) ، الفاصلة العائمة (Floating point Registers) ، مسجلات البيانات (Data Registers) ، والمراكم.

٢.**مسجلات خاصة الأغراض** : تستخدم من قبل الحاسوب لتخزين بيانات أو عناوين خاصة ولا يسمح للمبرمج باستخدامها.

***من الامثلة على المسجلات خاصة الأغراض:**

-**مسجل التعليمة** : **Instruction Register IR** الذي يستخدم لتخزين التعليمة التي يجري تنفيذها حالياً.

- عداد البرامج Program Counter CP : الذي يستخدم لتخزين عنوان التعليمة (ترتيبها) اللاحقة في التنفيذ .

- مسجل الرايات Status Register : الذي يستخدم لتخزين حالة البرنامج وحالة الحاسوب بعد تنفيذ كل تعليمة . يتكون مسجل الرايات من مجموعة ثنائيات Bits تسمى كل ثنائية منها راية Flag .

- مسجل البيانات Data Register الذي يستخدم لتخزين البيانات القادمة من الذاكرة الرئيسة أو المرسلة إليها .

- مسجل العنوان Address Register الذي يستخدم لتخزين عنوان البيانات القادمة من الذاكرة الرئيسة أو المرسلة إليها.

- **وحدة التحكم"Control Unit "CU :**

* تتلخص مهام وحدة التحكم في الإشراف على عمل وحدة المعالجة المركزية بشكل عام بحيث تقوم جميع مكوناتها باداء المهام المنوطة بها على أكمل وجه .

*تقوم وحدة التحكم كذلك بتنظيم عمليات الإتصال وتبادل البيانات بين وحدة المعالحة المركزية من جهة ، وبين الذاكرة الرئيسة ووحدات الإدخال والإخراج من جهة أخرى .

* إن وحدة التحكم تقوم بإصدار الإشارات اللازمة لتنفيذ جميع العمليات والنشاطات في الحاسوب حسب تعليمات البرنامج الذي يجري تنفيذه حاليا .

* نستنتج أن وحدة التحكم هي بمثابة المدير العام للحاسوب . ويجب الاشارة هنا الى طريقة ربط الوحدات في وحدة المعالجة المركزية ، وترتيب المسجلات فيها تؤدي إلى بناء معالجات ذات هيكليات متباينة .

*المعالج يشكل قلب نظام الحاسوب حيث أن قدرة وكفاءة الحاسوب تعتمد على قدرة وكفاءة المعالج نفسه.

*** تعتمد قدرة وكفاءة المعالج على العوامل التالية :**

١) عدد الثنائيات التي تعالج في نفس اللحظة . فمثلا يسطتيع المعالج الميكروي 6502 معالجة 8 ثنائيات، بينما يستطيع المعالج الميكروي 8086 معالجة 16 ثنائية في نفس اللحظة . أما المعالجات 80486 و MC68030 فتستطيع معالجة 32 ثنائية في نفس اللحظة.

٢) عدد التعليمات التي يستطيع المعالج تنفيذها .

٣) الإمكانيات المتوفرة في كل تعليمة .

٤) زمن تنفيذ التعليمة الواحدة .

٥) عدد التعليمات التي تنفذ في نفس اللحظة.

٦) سعة الذاكرة الرئيسة التي يستطيع إدارتها .

**** الذاكرة الرئيسية Main Memory**

**أن عملية تنفيذ البرامج تتم فعليا في وحده المعالجه المركزيه (cpu) غير أن وحدة المعالجة المركزية لا تحتوي على ذاكرة تكفي لتخزين المعطيات والبرامج والنتائج. وذلك لأن عدد المسجلات في وحدة المعالجة المركزية محدودة جداً.

**هناك نوعان من من الذاكرة الرئيسيه: رئيسيه ومساعدة.

****تتكون الذاكرة الرئيسيه** من عدد كبير من مواقع التخزين (Locations) ومن أجل التميز بين كل المواقع يتم ترقيمها بارقام صحيحة متسلسلة تسمى بالعناوين .addresses تبدأ العناوين من الصفر وتزداد بانتظام بمقدار 1وتنتهي بالرقم N-1 حيث N عدد كلمات الذاكرة.

يحمل كل موقع عنوانا خاصا به يختلف عن عناوين المواقع الأخرى. يمكنك أن تستخدم أي مواقع لتخزين أنواع مختلفة من البيانات مثل : الارقام (الصحيحة،الكسريه)،الرموز ،الرسومات أو الصور . يتسع الموقع الواحد عادة لحجم معين من كل نوع من البيانات. ****شكل 2صفحة10: الذاكرة الرئيسيه في الحاسوب.

**يمكنك الإستنتاج بأن موقع التخزين يتميز بعنوانه ومحتوياته وسعته (ثابت لجميع المواقع في نفس الذاكرة).

**يسمى طول موقع التخزين طول كلمة الذاكرة. أطول كلمة الذاكرة الاكثر شيوعاً هي: 16 8,4,2,1 بايت

**** تمتاز الذاكرة الرئيسيه بالخصائص التاليه:**

1.السعه: يقصد بها عدد مواقع التخزين في الذاكرة.

2.طول الكلمه

3.زمن الوصول :Acess timeاي من اللازم لقراءة (لتخزين) محتويات موقع واحد.

**تعرفت من مساقات الحاسوب التي درستها سابقاً أن اصغر وحده لقياس المعلومات هي الثنائية ،وان المجموعة المكونه من 8ثنائيات تسمى بايتا .

****اهم وحدات قياس سعة الذاكرة الرئيسيه**

1كيلو بايت =1024بايت

1ميجابايت =1024كيلو بايت

1جيجابايت =1024ميغابايت

1تيرابايت=1024جيجابايت

****تصنف الذاكرة الرئيسيه إلى الأصناف التاليه: أولا :حسب تكنولوجيا التصنيع**

أ. ذاكرة الحلقات الممغنطة

تتكون من عدد كبير من الحلقات المصنوعة من مواد سهله المغنطه. والتي يمر من خلالها أسلاك كهربائية للتحكم بمغنطتها . فعندما يمر من خلالها هذه الأسلاك تيار باتجاه معين يخزن في الحلقه 1. أما إذا مر التيار بالاتجاه المعاكس فيخزن فيها 0اي أن الحلقه تستخدم لتخزين ثنائية واحده .

**** ومن أهم خصائص ذاكرة الحلقات الممغنطة ما يلي:**

1.ذاكرة الحلقات الممغنطة هي غير متطايرة وثابته أي أنها لا تفقد محتوياتها في حالة انقطاع التيار الكهربائي.

2.كبيرة الحجم

3.كثيرة الأعطال

4.قليلة السرعه نسبياً

5.كلفتها العاليه بسبب أن صناعتها تتم بطريقة يدوية.

**وبسبب هذه المساوئ فإن ذاكرة الحلقات الممغنطة قد تم الاستغناء عنها.

ب. ذاكرة أشباه الموصلات Semi-conducters

من دوائر الكترونية ميكروسكوبيه مبنيه على رقاقات مصنوعه من مواد أشباه الموصلات مثل السيليكون والجرمانيوم . تمتاز ذاكرة أشباه الموصلات بصغر حجمها وقلة كلفتها وسرعتها الفائقة ودرجة وثوقيتها العاليه (عدم تعطلها) غير أنها **ذاكرة متطايرة volatile :**وهي:الذاكرة التي تفقد محتوياتها عند إنقطاع التيار الكهربائي

****اما الذاكرة غير المتطايرة Nonvolatile :**الذاكرة اتي لا تفقد محتوياتها عند إنقطاع التيار الكهربائي

****تستخدم الذاكرة الرئيسيه لتخزين:**

1.تعليمات البرامج التي يجري تنفيذها حالياً.

2.مدخلات البرامج

3.النتائج الوسطية

4.النتائج النهائية

*تخزن هذه المواد في الذاكرة الرئيسيه بشكل مؤقت أثناء تنفيذ البرنامج. أي أن الذاكرة الرئيسيه هي ذاكرة مؤقته ؛ فعند الإنتهاء من تنفيذ برنامج ما يقوم به الحاسوب إما بشطبه أو تخزينه في الذاكرة المساعده حسب رغبة المستخدم لكي يفسح المجال لتنفيذ البرامج الاخرى . والسبب في ذلك يعود إلى ارتفاع ثمن الذاكرة الرئيسيه نسبياً مما أدى إلى تقليل سعتها . هذا بالإضافة إلى أنه هناك حاجه ماسه لإبقاء البرامج غير الضرورية في وقت ما في الذاكرة الرئيسيه.

****الذاكرة الثانويه (المساعده)Secondary Memory**

أن الذاكرة الرئيسيه تستخدم لتخزين البيانات والبرامج بشكل مؤقت ولا يمكن استخدامها بشكل دائم وذلك للأسباب التالية:

.1يمتاز الجزء الأكبر من الذاكرة الرئيسيه بخاصية التطاير مما يجعلها غير صالحة للتخزين الدائم.

.2سعة الذاكرة الرئيسيه محدودة إذا ما قورنت بحجم المعلومات المراد تخزينها.

.3ارتفاع ثمن الذاكرة الرئيسيه بالمقارنة مع أنواع الذاكرة الأخرى

.4استحالة نقل محتويات الذاكرة الرئيسيه من حاسوب لآخر بدون استخدام وسائل خاصه أو إتصال مباشر بين الحواسيب.

**إن هذه الأسباب تقودنا إلى ضرورة استخدام نوع آخر من الذاكرة, وهو الذاكرة الثانوية.

*** من أهم خصائص الذاكرة الثانوية:**

1. غير متطايرة 2. سعتها غير محدودة عملياً

3. رخيصة الثمن

4. قليلة السرعة بالمقارنة مع الذراكرة الرئيسية أو المسجلات وذلك لأن وحدات الذاكرة الثانوية تحتوي على أجزاء ميكانيكية متحركة.

***تتوافر عدة أنواع من الذاكرة الثانوية منها:** الوحدات المغناطيسية : تتكون من وحدة الذاكرة المغناطيسية من جزئين رئيسيين أحداهما إلكتروني ويسمى المنظم controller, والآخر ميكانيكي ويسمى مشغلاًDriver.

*يقوم المنظم بالاشراف على عملية نقل البيانات وإشارات التحكم بين الذاكرة المغناطيسية وأجزاء الحاسوب الأخرى.

*بينما يقوم المشغل بتحريك وسك التخزين المغناطيسي.

*إن أكثر وحدات التخزين المغناطيسي انتشاراً في الوقت الحالي هي تلك الحبيبات التي تستخدم الشريط المغناطيسي أو القرص المغناطيسي.

*الشريط المغناطيسي هو عبارة عن شريط بلاستيكي مغطى من جهة واحدة بمادة سهلة المغنطة مثل أوكسيد الحديد.

*تتكون المادة سهلة المغنطة من حبيبات صغيرة. تخزن البيانات بواسطة مجالات كهرومغناطيسية تؤدي إلى ترتيب الحبيبات على الشريط في وضعية معينة.

***يقسم سطح الشريط إلى مسارات:** 1. طولية (tracks). .2عرضي(frames)تمثل كل منها بايت واحدة.

يختلف عدد المسارات الطولية من شريط الى آخر . غير أن الأشرطة الأكثر شيوعاُ هي تلك ذات 9مسارات طولية.

*يستخدم المسار العرضي الواحد لتخزين رمز واحد. ويسمى عدد الرموز التي يمكن تخزينها في وحدات الطول"كثافة التخزين" تتراوح كثافة التخزين بين 200 و 6250 رمز لكل إنش, ويمكن أن تصل إلى 3800 رمز لكل انش في الحواسيب الحديثة مثل IBM 3480.

*تنظم البيانات على الشريط المغناطيسي في مجموعات تسمى كل واحدة منها سجلاُ Record. ويمكن تخزين السجلات بطريقة افرادية أو كتلية Blocks.

*وحسب الطريقة الافرادية يخزن كل سجل بحيث يكون مسبوقا ومتبوعا بمنطقة فراغ تسمى فجوة سجلية. " interrecord gao" وبهذا يكون عدد الفجوات التي تفصل بين السجلات مساويا لعدد السجلات تقريباً. ونتيجة لذلك يصبح جزء كبير من الشريط غير مستخدماً لتسجيل البيانات. شكل 3 صفحة 14 .

*من أجل تقليل المساحات الضائعة على الشريط بسبب الفجوات نستخدم طريقة الكتل

*تعتمد طريقة الكتل على دمج السجلات بعضها مع بعض في كتل

*يسمى عدد السجلات بالكتلة الواحدة بمعامل التكتل Blocking factor.

*تفصل الكتل عن بعضها بواسطة الفجوات الكتلية Interblock gaps

*** إن استخدام طريقة الكتل في تخزين البيانات على الشريط يؤدي الى:**

1. زيادة السعة الفعالة للشريط بتقليل عدد الفجوات بين السجلات
2. زيادة سرعة نقل البيانات بين الشريط وأجزاء الحاسوب الأخرى
3. ضرورة زيادة حجم الذاكرة الرئيسية التي تتعامل مع الكتل بدل السجلات
4. ضرورة استخدام برامج خاصة لتكوين الكتل وتفكيكها.

***يمكن تلخيص أهم حسنات الاشرطة المغناطيسية على النحو الآتي:**

1. تعتبر الأشرطة المغناطيسية ذاكرة غير متطايرة

2. رخيصة الثمن ويمكن اعادة استعمالها.

3. كثافة التخزين عالية مما يزيد من قدرتها الاستيعابية على التخزين

4. الأشرطة المغناطيسية ذات مواصفات قياسية مما يجعلها قابلة للتدتول والتبادل بين الحواسيب المختلفة.

*** مساوئ الأشرطة المغناطيسية:**

1. يمكن الوصول الى سجلات الأشرطة المغناطيسية بطريقة تسلسلية فقط مما يؤدي لزيادة زمن الوصول الى البيانات المطلوبة.

2. حساسة للغبار والرطوبة والتغير في درجات الحرارة

* أصبحت الاقراص المغناطيسية من أكثر وحدات التخزين شيوعا بسبب امكانية الوصول المباشر الى البيانات المخزنة على القرص.

***هناك نوعان من الأقراص الممغنطة:** الأقراص الصلبة والمرنة.

* وحدة الأقراص عادة على قرص واحدة أو عدة أقراص وتسمى في هذه الحالة حزمة الأقراص.

**يتراوح عدد الأقراص في الحزمة بين 1 وأكثر من 20 قرص

*أما القرص المرن فيصنع من البلاستيك ويغطى أحد وجهيه أو كليهما بمادة سهلة المغنطة.

***تستخدم الأقراص المرنة بشكل كبير في** الحاسبات الميكروية والشخصية. تتميز الأقراص المرنة بقياس قطرها فهناك الأقراص ذات قطر 3.5 إنش و 5.25 إنش

***يقسم سطح القرص الى** مسارات داآرية مركزية يتراوح عددها بين 40 مسارا في الأقراص المرنة وأكثر من 800 مسارا في الأقراص الصلبة. *تقسم المسارات الى أجزاء يسمى كل منها قطاعا.

*تنظم البيانات على الأقراص الممغنطة في المسارات والقطاعات حسب طريقة الاسطوانة او طريقة القطاع

*لتوضيح مفهوم الأسطوانة تذكر أن المسارات تكون مرقمة من الخارج الى الداخل ابتداءً من الرقم صفر وبزيادة 1 حتى الوصول الى أقصى مسار في الداخل.

**تثبت الأقراص في وسطها على محور يدور بسرعة محددة حوالي 3000 دورة الدقيقة.

**تسمى المسارات المتناظرة على جميع الأقراص والتي تحمل نفس الرقم إسطوانة

*حسب طريقة الاسطوانة يتم الوصول الى البيانات بتحريك رؤوس القراءة والكتابة الى الأسطوانة المطلوبة وعندها تتم عملية القراءة أو الكتابة في مسارات هذه الأسطوانة.

* يتم الانتقال الى الأسطوانة التالية المجاورة في حالة الانتهاء كليا من الاسطوانة الحالية . إن استخدام طريقة الاسطوانة يؤدي الى زيادة سرعة الوصول الى البيانات عن طريق تقليل عدد عمليات العمود الحامل لرؤوس القراءة والكتابة, والذي يتحرك أفقياً بين الأسطوانات دخولاً أو خروجاً

*تستخدم طريقة القطاع في تنظيم البيانات على سطح القرص الذي يقسم القطاعات وترقم من صفر الى 8 في المسار الواحد.

*يتم الوصول الى البيانات بتحديد: رقم السطح ورقم المسار, ورقم القطاع.

تعتمد سعة وحدة الأقراص المغناطيسية على العوامل التالية:

1. كثافة التخزين التي تحدد سعة القطاع وسعة المسار.
2. عدد الأوجه في وحدة الأقراص.
3. عدد المسارات في الوجه الواحد 0 .

*الشكل 5صفحة: 17: تنظيم البيانات على الاقراص المغناطيسية.

وحدات الادخال والإخراج Units Input & Output

*يقصد بعملية الادخال input نقل البيانات من العالم الخارجي الى ذاكرة الحاسوب

*عملية الإخراج output نقل المعلومات من ذاكرة الحاسوب الى العالم الخارجي

تصنف وحدات الادخال حسب نوعية البيانات التي يتم إدخالها الى الأنواع التالية:

-1وحدات ادخال النصوص (الرموز) ومن الأمثلة عليها : لوحة المفاتيح

-2وحدات ادخال الرسومات والصور ومن الأمثلة عليها المرقماتDigitizers

-3وحدات ادخال الصوت

-4وحدات ادخال أخرى مثل :الفأرة ,القلم الضوئي , وشاشات اللمس

**وبغض النظر عن اختلاف وحدات الادخال الا انها جميعا تعمل حسب مبدأ واحد وهو تحويل البيانات من صورتها الإنسانية (رموز , صور , صوت ....) الى صورتها الالكترونية (اصفار وواحدات)وبعد ذلك تصبح البيانات مفهومة للحاسوب وقابلة للمعالجة والحصول على النتائج وتخزينها بصورتها الالكترونية ومن اجل الاستفادة كم هذه النتائج يجب استخراجها بصورتها الطبيعية المفهومة للإنسان

**أنواع وحدات الإخراج غير ان اكثرها شيوعا هي الشاشات والطابعات والراسمات

ويمكن تصنيف الطابعات حسب طريقة تشكيل الرموز الى نوعين:

-1طابعات الرموز الثابتةFully -formed character printers تمتاز هذه الطابعات بجودة الرموز المطبوعة , غير انها بطيئة السرعة نسبيا ولا يمكن استخدامها لطباعة الرسومات والصور

-2**طابعات المصفوفة النقطية**Dot-matrix printers يتكون الرمز في هذه الطابعات من مصفوفة نقاط لذا تعتمد على جودة الرموز المطبوعة على عدد النقاط في المصفوفة ومدى تقاربها ببعضها البعض ومن اهم ما يميز طابعات المصفوفة النقطية قدرتها على طباعة الرسومات والاشكال والصور بالإضافة الى الرموز التي يمكن طباعتها بأشكال وأساليب مختلفة حسب نوع الخط وارتفاعه **شكل (6)تحول البيانات في وحدات الادخال والإخراج

****ويمكن كذلك تصنيف الطابعات حسب طريقة نقل الرموز الى الورق اثناء الطباعة الى نوعين:**

-1**طابعات مطرقية Impact printers**

تستخدم هذه الطابعات مطارق او دبابيس للضرب على الورق من خلال شريط حبري

ومنها : نفث (رش)الحبر ,اشعة الليزر ,الشحنات الكهروستاتيكية , او الحرارة.

تتميز الطابعات اللامطرقية بسرعتها العالية بالمقارنة مع الطابعات المطرقية وكذلك كونها تعمل بهدوء بدون ضجيج

اما من حيث عدد الرموز التي يمكن طباعتها في نفس اللحظة **فيمكن تصنيف الطابعات الى ثلاثة أنواع:

-1**طابعات رمزية Character Printers** تطبع رمز واحد في اللحظة الواحدة

-2**طابعات سطرية Line Printers** تطبع سطر واحد في اللحظة الواحدة

-3**طابعات صفحية Pag Printers** تطبع صفحة واحدة في اللحظة الواحدة

**كيفية عرض البيانات على الشاشة فالبيانات المراد عرضها على الشاشة يجب ان تخزن مسبقا في منطقة معينة في الذاكرة الرئيسية تسمى ذاكرة الفيديو فعند تنفيذ امر العرض على الشاشة تنتقل المعلومات من ذاكرة الفيديو الى الشاشة

**وتتكون الشاشة من عناصر صغيرة الحجم يسمى كل عنصر منها بكسيل تتكون الرموز والصور على الشاشة بإضافة مجموعة من البكسيلات بواسطة حزمة من الالكترونيات المتجهة الى الشاشة

****وحسب طريقة تخزين الرموز والصور في ذاكرة الفيديو تصنف الشاشات الى نوعين:**

-1**شاشات رمزية**

حيث يخصص لكل رمز مجموعة البكسيلات تحد شكل ذلك الرمز يخزن الرمز مع خصائصه في بايت او بايتين في ذاكرة الفيديو ولتحويل الرموز المخزنة في ذاكرة الفيديو الى مجموعة البكسيلات المناسبة تستخدم دارة الكترونية خاصة تسمى مولد الرموز generator Character

-2**شاشات بكسيل** حيث يخصص لكل بكسيل على الشاشة ثنائية واحدة او مجموعة من الثنائيات في ذاكرة الفيديو ففي الشاشات أحادية اللون يخصص ثانية واحدة لكل بكسيل اما الشاشات الملونة فيخصص ثنائيتان اثنتان على الأقل ومن هنا نستنتج ان حجم ذاكرة الفيديو يعتمد على عدد البكسيلات المكونة للشاشة وكذلك على عدد الألوان التي يمكن عرضها على الشاشة.

* وبالتالي فإن درجة وضوح الصورة على الشاشة Resolution تعتمد على عدد البكسيلات المكونة للشاشة تحدد درجة وضوح الصورة بواسطة رقمين يحدد الأول عدد البكسيلات الافقية ويحدد الثاني عدد البكسيلات العمودية **فمثلا:**

*اذا كانت درجه وضوح الصورة هي 480*640 فهذا يعني ان الشاشة مكونة من 640 بكسيلا افقيا و 480 بكسيلا عموديا.

*يجب أن يحتوي الحاسوب على بطاقة (card)خاصة تناسب شاشات المستخدمة من حيث درجة وضوح الصورة

***هناك أربعة أنواع من البطاقات القياسية :**

-بطاقة الرسم الملونة CGA Color Graphics Adapter 320×200 مع اربع الالوان

-بطاقة الرسم الملونة المحسنة EGA Extended Color Graphics Adapter 640×480 مع 16لون

-مصفوفة الرسم الفيديوية VGA Video Graphics Array 640×480 مع 16لون أو 320×200 مع 256لون

-مصفوفة الرسم السوبر فيديوية SVGA Super Video Graphics Array 1024×768 مع 23768لون

***الناقلة**:**Bus** تعرف الناقلة على أنها مجموعة من الأسلاك (الخطوط) ووحدات وصل لنقل البيانات وإشارات التحكم بين الذاكرة الرئيسية ووحدة المعالجة المركزية من جهة، وبين وحدة المعالجة المركزية ومنظم وحدات الادخال والإخراج من جهة اخرى

***إن استخدام الناقلات في الحاسوب يزيد من مرونته وذلك عن طريق توفير الإمكانيات اللازمة :**

-لزيادة سعة الذاكرة الرئيسة إذا دعت الحاجة لذلك

-لإضافة وحدات ادخال أو اخراج أو وحدات تخزين مساعدة

-لتوصيل الحاسوب مع غيره من الحواسيب

***تتكون الناقلة من ثلاثة أنواع من الخطوط :**

اولا : خطوط العنوان Address lines

*تستخدم خطوط العنوان لنقل الإشارات التي تحدد عنوان موقع الذاكرة الرئيسة المراد القراءة منها أو التخزين فيه .اي أنه يجب ان تكون هناك علاقة تربط عدد خطوط العنوان والسعة القصوى للذاكرة الرئيسة .هذه العلاقة هي $N=2^m$ حيث أن m هي عدد خطوط العنوان ، Nهي السعة القصوى للذاكرة الرئيسة

*فمثلا إذا كان عدد خطوط العنوان في حاسوب ما هو 16خط ، فإن هذا الحاسوب يستطيع عنونة ذاكرة رئيسة ذات سعة تصل لغاية 64=كيلو بايت

*ويمكن كذلك استخدام نفس العلاقة السابقة لتحديد عدد خطوط العنوان اللازمه لعنونة اي ذاكرة رئيسة اذا علمت سعتها

*فمثلا إذا علمت أن سعة الذاكرة الرئيسة في احد الحواسيب تساوي 1ميجابايت 1048576=بايت ،فان عدد خطوط العنوان mيحدد كما يلي $N=2^m$ $m=\log_2 N$

بتعويض قيمة$N= 1048576$ في المعادلة الأخيرة نجد أن$m=20$

*ومن المهم أن نعرف أن إشارات العنوان تنتقل باتجاه واحد من المعالج إلى الذاكرة الرئيسية

ثانيا: خطوط البياناتData lines

*تستخدم خطوط البيانات لنقل البيانات وتعليمات البرامج من وإلى الذاكرة الرئيسة يحدد عدد خطوط البيانات سرعة نقل البيانات بين الذاكرة الرئيسة والمعالج

*فمثلا إذا كان عدد خطوط البيانات هو 8خطوط ويراد نقل قيمة طولها 32ثانية من الذاكرة الرئيسة إلى مكان ما في الحاسوب ،فانه يلزم 4حركات نقل (ينقل 8ثنائيات في كل حركة) ،اذا ما كان عدد خطوط البيانات 16خط،فانه يلزم حركتان فقط

*واذا كان عدد خطوط البيانات 32خط ،فانه يلزم حركة واحدة يتراوح عدد خطوط البيانات بين 8خطوط في الحواسيب الميكروية والشخصية و 128خط في الحواسيب العملاقة

*إن خطوط البيانات تستخدم لنقل البيانات في اتجاهين من الذاكرة الرئيسة وإليها

ثالثا:خطوط التحكم Control lines

*تستخدم خطوط التحكم لنقل إشارات التحكم من المعالج إلى جميع وحدات الحاسوب الأخرى وبالعكس ومن الأمثلة على إشارات التحكم : إشارة القراءة ,read وإشارة الكتابة ,write وإشارة تصفير النظام .... reset وغيرها

****التنظيم الهيكلي للحواسيب الكبيرة**

*الشكل الذي يظهر فيه الحاسوب من وجهة نظر المبرمجين بلغة اسمبلي وتتكون هيكلية الحاسوب من : تمثل البيانات ،وطرق العنونه ،صيغ التعليمات .... وغيرها

***تتضمن الهيكلية العناصر التالية** : انواع البيانات وطرق تمثيلها ،طرق العنونة ،صيغ التعليمات وانواعها، النموذج البرمجي للمعالج ،تنظيم الذاكرة الرئيسية ،نظام الادخال والإخراج ،الاعتراضات ،انظمة التشغيل

***عناصر هيكلية الحاسوب :**

أولاً:انواع البيانات في حواسيب IBM 360/370

*تستطيع حواسيبIBM 360/370 ان تتعامل مع أنواع مختلفة من البيانات

*ويشكل عام تصنف البيانات عددية وبيانات رمزية

***أما البيانات العددية فتقسم إلى أربعة أنواع هي:**

-الاعداد الثنائية بصيغة الفاصلة الثابتة Fixed-point binary

-الاعداد الثنائية بصيغة الفاصلة العائمة Floating-point binary

-الاعداد العشرية المحزومة packed decimal

-الاعداد العشرية غير المحزومة (الممتدة) Unpacked decimal

*سميت الاعداد بصيغة الفاصلة الثابتة نظرا لكون الفاصلة الكسرية تحتل موقعا ثابتا على يسار اخر رقم

*تتكون **الاعداد الثنائية الممثلة بصيغة الفاصلة الثابتة** من جزئين هما : الإشارة والمقدار .تمثل الاعداد السالبة عادة في نظام المكمل لاثنين complement 2`s تتضمن حواسيبIBM 360/370 تعليمات لمعالجة الاعداد الممثلة بصيغة الفاصلة الثابتة التي يكون طولها 32او 16ثانية

*شكل (8) تمثيل البيانات في حواسيب IBM 370/360

*أما **الاعداد الممثله بصيغة الفاصلة العائمة** لقد سميت بهذا الإسم نظرا لكون الفاصلة الكسريع يمكن أن تحتل اي موقع في العدد ، اي ان الفاصله تعوم من موقع الي أخر في العدد . * تتضمن حواسيب IBM 370/360 تعليمات لمعالجه الاعداد الممثلة بصيغة الفاصلة العائمة التي يكون طولها 32 او 128 ثنائيه .

****تتكون الأعداد الثنائيه الممثلة بصيغة الفاصلة العائمه من ثلاث حقول هي :**

1.حقل إشارة العدد (ثنائية واحدة) : ويخصص لها ثنائية الواقعية في اقصى يسار تمثيل العدد .

2.حقل الأس : ويخصص له 8او 11 ثنائيه .

3.حقل الكسر (mantissa) ويخصص له 23 او 52 ثنائيه .

*يتكون العدد العشري من الأرقام العربية 9....،1 ،0 . وتمثل الأعداء العشرية في حواسيب IBM 360/370 حسب نظام"(BCD) Binary - coded decimal"

*حيث يمثل كل رقم (في النظام) في العدد بواسطة 4 ثنائيات (من 0000 الى 1001)

*كما في الجدول التالي جدول (1): تمثيل الارقام العشرية حسب نظامBCD

*فمثلا يحول العدد العشري 1986 الي نظام BCD كما في الشكل 9: شكل (9): تحويل الإعداد العشرية الي نظامBCD

****تصنيف الاعداد العشرية حسب طريقة تخزينها في الذاكرة إلى نوعين :**

1**الاعداد العشرية المحزونة (المضغوطة Packed BCD**

حيث يخزن في البايت الواحد رقمين عشريين .

* أما إشارة الرقم فيخصص لها النصف الايمن في البايت الواقع من أقصى يمين تمثيل العدد فمثلا لتخزين العدد العشري 1985 يلزمنا ثلاث بايتات كما هو مبين في الشكل (10- أ) صفحة 28

*شكل (10) : تمثيل الاعداد في نظامBCD

-2**الاعداد العشرية غير المحزومة (الممتدة)Unpacked BCD**

حيث يخصص بايت واحد لكل رقم عشري . ونظراً لأن تمثيل الرقم العشري الواحد يحتاج إلى 4 ثنائيات ، والبايت يتكون من 8 ثنائيات ، فإن الرقم العشري يحتل النصف الايمن في البايت . أما النصف الأيسر (يسمى نطاقا) فيمكن أن يحتوي على أية قيمة محصورة بين الرقم 0000 والرقم 1111. أما البايت في أقصى اليمين فيحتوى إشارة الرقم بدلاً من النطاق. لذا فالرقم 1985 يحتاج إلى 4 بايت كما هو مبين في الشكل (10-ب).

ثانياً: بنية معالج حواسيبIBM 360/370

* يتكون معالج حواسيل IBM 360/370 من ثلاث وحدات رئيسة هي : وحدة الحساب والمنطق، وحدة التحكم ، وحدة المسجلات .

***تتكون وحدة الحساب والمنطق (انظر الشكل رقم 11) من ثلاثة أجزاء هي :**

1_وحدة الفاصلة الثابتة : تستخدم لتنفيذ العمليات المختلفة على القيم الممثلة بصيغة الفاصلة الثابتة.

2-وحدة الفاصلة العائمة : تستخدم لتنفيذ العمليات على القيم الممثلة بصيغة الفاصلة العائمة .

3-وحدة الأعداد العشرية :BCD تستخدم لتنفيذ العمليات على القيم العشرية BCD

*شكل (١) بنية المعالج في حواسيب IBM 370/360

*يحتوي معالج حواسيب IBM 370/360 على 3انواع من المسجلات :

1. مسجلات الفاصلة العائمة وعددها اربعة (4) تستخدم لتخزين القيم الممثلة بصيغة الفاصلة العائمة ، وطول المسجل الواحد يساوي 64 ثانيه .

2. مسجلات عامة الاغراض عددها 16 وطول كل منها 32 .

3.مسجلات خاصة الأغراض عددها 16 **ومن امثلتها:**

-مسجل البيانات " DataRegister" .DR

-مسجل العنوان . "AR " AdressRegister

-مسجل التعليمة "IR "Instruction Register

-مسجل حالة البرنامج "PSW" Program Status Word

*يستخدم مسجل حالة البرنامج psw لتخزين حالة الحاسوب والبرنامج المنفذ حاليا ويحتوي على معلومات تحدد حالة جميع وحدات الادخال والاخراج

بالاضافه الي حالة المعالج الرئيسي ومعالجات الادخال والاخراج . IOP "Input/Output Processros " اي ان مسجل حالة البرنامج psw يشبه مسجل الرايات الذي ذكرناه سابقا .

*نقصد بالنموذج البرامجي للمعالج المسجلاتالتي يسمح للمبرمج استخدامها اثناء كتابة برامج بلغة الاسمبلي . يبين شكل (12) صفحه 31 النموذج البرامجي لمعالج حواسيب IBM 370/360

ثالثاً: صيغ التعليمات وطرق العنونة

*تتكون تعليمات حواسيب IBM 360/370 من 2 بايت أو 4 بايت أو 6 بايت. ويتراوح عدد العناوين فيها من عنوان واحد إلى ثلاثة عناوين.

*حسب مواقع تخزين المعاملات وطرق عنونتها تصنف التعليمات إلى الأنواع الخمسة التالية:

1.تعليمات) RR مسجل-مسجل) تحتوي تعلميات RR على معاملتين اثنتين في المسجلات .

2.تعليمات) RX مسجل-ذاكرة) تحتوي تعليمات RX على معاملتين اثنتين احداهما في أحد المسجلات والأخرى في أحد مواقع الذاكرة الرئيسية.

3.تعليمات) RS مسجل-ذاكرة) تحتوي تعليمات RS على ثلاثة معاملات : الأولى والثانية في المسجلات والثالثة في أحد مواقع الذاكرة الرئيسية.

4.تعليمات) SI ذاكرة-تعليمة) تحتوي تعليمات SI على معاملتين اثنتين احداهما في أحد مواقع الذاكرة الرئيسية والأخرى في التعليمة نفسها.

5.تعليمات) SS ذاكرة-ذاكرة) تحتوي تعليمات SS على معاملتين اثنتين كلاهما في مواقع الذاكرة الرئيسية.

***طرق العنونة :** الأساليب المتبعة في تحديد مواقع البيانات .ومن أجل تحديد مواقع القيم المخزنة في مسجلات يكفي معرفة اسم أو رقم المسجل الذي يحتوي القيمة المطلوبة.

*تسمى طريقة العنونة هذه العنونة باستخدام المسجلات، اما اذا كانت القيمة مخزنة في مواقع الذاكرة الرئيسية، فيلزم تحديد عنوان هذه المواقع باستخدام احدى طرق العنونة المخصصة لذلك.

***طرق العنونة المستخدمة في الحواسيب IBM 360/370 فهي التالية:**

- العنونة الفورية Immediate addressing
- العنونة باستخدام المسجلات Register addressing
- العنونة المباشرة Direct addressing
- العنونة باستخدام الأساس Based addressing
- العنونة باستخدام الأساس والفرسBased-Indexed addressing

*فمثلا اذا استخدمت طريقة العنونة باستخدام الأساس والفرس **فإن العنوان يتكون من ثلاثة عناصر وهي :**

-محتويات مسجل الأساسBase B

-مقدار الازاحة Displacement D

-محتويات مسجل الفهرسIndex X

**ولحساب العنوان الفعلي تجمع القيم الثلاثة المذكورة سابقة كما يلي Addess=B+X+D :

رابعاً: الذاكرة الرئيسية ونظام الإدخال والاخراج

*تصل سعة الذاكرة الرئيسية في حواسيب IBM 360/370 لغاية 16 ميغابايت (حوالي 16 مليون بايت) ، وهي مقسمة إلى كلمات طول كل منها 4 بايت . ويمكن تجزئة الكلمة Word إلى جزئين متساويين طول كل منهما بايتان إثنان يسمى كل منهما نصف كلمة Halfword ، ويمكن كذلك تجزئة نصف الكلمة إلى نصفين طول كل منهما بايت واحد.

***الاطلاع على شكل ١٣ : تنظيم الذاكرة الرئيسية في حواسيب IBM 360/370 صفحة ٣٣.**

*من أجل الوصول إلى كل كلمة وكل بايت في الذاكرة يحمل كل بايت عنوانا مختلفا عن عناوين البايتات الأخرى . وترقم البايتات في الكلمات من اليسار إلى اليمين.

*تعتمد سرعة الحاسوب على عدة عوامل من أهمها سرعة المعالج المستخدم وكذلك سرعة الوصول إلى البيانات في الذاكرة الرئيسية. وبما أن سرعة الذاكرة الرئيسية أقل بكثير من سرعة المعالج، أي أن الذاكرة الرئيسية تعتبر نقطة الضعف في الحاسوب، فقد تركزت الأبحاث لزيادة فعالية الذاكرة الرئيسية. ومن الطرق المتبعة لزيادة فعالية الذاكرة الرئيسية في حواسيب IBM 360/370 استخدام ذاكرة كاشي Cache memory والذاكرة الارتباطية. Associative memory

***تعرف ذاكرة كاشي** بأنها ذاكرة صغيرة الحجم(لا تتجاوز سعتها بضع مئات كيلوبايت) ولكنها من أسرع أنواع الذاكرة المستخدمة في الحاسوب بعد المسجلات.

***تستخدم ذاكرة كاشي** لتخزين البيانات والتعليمات الاكثر استعمالاً في وقت ما. توضع ذاكرة كاشي عادةً بين المعالج والذاكرة الرئيسية (الاطلاع على شكل ١٤ صفحة ٣٤).

***الذاكرة الارتباطية :** هي ذاكرة صغيرة الحجم أيضا يتم البحث والتخزين فيها بسرعة عالية.

***من اهم خصائصها :** ان عملية البحث فيها تتم حسب المحتويات وليس حسب العنوان كما في الذاكرة الرئيسية .

*تتم عملية البحث في جميع مواقع الذاكرة الارتباطية في نفس الوقت.

***يتكون نظام الادخال والاخراج في حواسيب IBM 360/370 من الأجزاء التالية :**

- وحدات الادخال والاخراج.
- وحدات التحكم بوحدات الادخال والاخراج. Control units
- معالجات الادخال والاخراج) قنوات. Input-Output Processors IOP (Channels

***تحتوي حواسيب IBM 360/370 على عدة معالجات:**

1.معالجات الإدخال والاخراج بالاضافة الى المعالج الرئيسي.

***مهام المعالج الرئيسي:** تنفيذ جميع أنواع التعليمات بإستثناء تعليمات الادخال والاخراج التي أصبحت من مهام معالجات الإدخال والاخراج .

*ان حواسيب IBM 360/370 تحتوي مصدر واحد للتعليمات وعدة معالجات لتنفيذها.

*يمكن أن يؤدي هذا إلى حدوث خلافات conflicts بين المعالجات.

*يمكن تجنب حدوث مثل هذه الخلافات بتوجيه التعليمات إلى المعالج الرئيسي والذي يقوم بدوره بتحديد نوع التعليمة.

*إذا كانت من اختصاصه او يرسلها الى معالج الإدخال والإخراج المختص. وهذا يتطلب ان يحتوي طاقم تعليمات المعالج الرئيسي على تعليمات خاصة لتنظيم عملية الاتصال بينه و بين معالجات الإدخال والإخراج .

***ومن أمثلة هذه التعليمات:**

١ .تعليمة إبدأ الإدخال / الاخراج"SIO"Start input/output

٢.تعليمة أو قف الإدخال / الاخراج"HI0"Halt input/output

٣ .تعليمة إفحص الإدخال / الاخراج"TIO"Test input/output

***ومن الأمثلة على تعليمات معالجات الإدخال والإخراج ما يلي:**

١ .تعليمة إقرأread

٢ .تعليمة خزنWrite

٣ .تعليمة إقلب الصفحه. eject

****سؤال: لماذا استخدمت شركةIBM معالجات خاصه لتنفيذ عمليات الإدخال و الاخراج ؟ وهل المعالج الرئيسي لا يستطيع تنفيذ عمليات الإدخال والإخراج ؟**

***الإجابة :** هي أن معظم وحدات الإدخال و الاخراج في طبيعتها كهروميكانيكيه بطيئة السرعة بالمقارنة مع المعالجات الالكترونية ذات السرعة العالية . وذلك لأن الحركة الميكانيكية في وحدات الإدخال و الاخراج تتطلب وقتا اكبر من المعالجة الالكترونية في المعالج. فإذا احتوى الحاسوب معالجا واحدا ذا سرعة عالية وعدة وحدات للإدخال والإخراج ذات سرعة قليلة ، فإن المعالج سيقضي معظم وقته متوقفا ينتظر انتهاء عمليات الإدخال والإخراج.

*لذا فالحواسيب التي تحتوي معالجات خاصة للإدخال و الاخراج تعمل اسرع بكثير من تلك التي تحتوي معالجة واحدة.

***ويمكنك تفسير ذلك كمايلي:** فبعد إحضار التعليمة من الذاكرة الرئيسية يقوم المعالج الرئيسي بتحديد نوع العمليه فيها. فإذا كانت من غير عمليات الإدخال و الاخراج يقوم المعالج الرئيسي بتنفيذها بنفسه .أما إذا كانت من عمليات الإدخال و الاخراج يصدر المعالج الرئيسي اوامره إلى وحدة الإدخال و الاخراج المعنية لتنفيذ العملية، ومن ثم ينتقل المعالج الرئيسي الى حالة التوقف في انتظار انتهاء العملية في وحدة الإدخال و الاخراج، التي تقوم عندئذ بإصدار اشارة خاصة الى المعالج الرئيسي تعلمه بانتهاء العملية. ثم يستأنف المعالج الرئيسي عمله بتنفيذ التعليمه التالية . **هذا في حالة وجود معالج واحد**

*** اما إذا احتوى الحاسوب على معالجات**

خاصه للإدخال و الاخراج، فإن المعالج الرئيسي، بدلا من التوقف وانتظار انتهاء العملية في وحدة الإدخال والإخراج، يصدر اوامره إلى معالج الإدخال و الاخراج للإشراف على تنفيذ العملية المطلوبة. وعندما يبدأ معالج الإدخال و الاخراج بتنفيذ العملية ينتقل المعالج الرئيسي فوراً إلى تنفيذ التعليمة اللاحقة .

*وهكذا فإن المعالج الرئيسي ومعالجات الإدخال والاخراج تعمل على التوازي مما يؤدي إلى زيادة سرعة وإنتاجية الحاسوب . يبين الشكل ١٥ صفحة ٣٦ المخطط الصندوقي لحواسيب IBM 360/370 صفحة ٣٦

**هيكليات الحاسوب الميكروية

أولاً: انواع البيانات في حواسيب 286(الرقم المختصر من رقم المعالج الدقيق 80286)

*تستطيع حواسيب 286 معالجة جميع انواع البيانات بغض النظر عن اطوالها تماما كما في الحواسيب الكبيرة

*غير ان سرعة المعالجة في حواسيب 286 تكون عادة اقل منها في الحواسيب الميكروية . وذلك لأن حواسيب286 تتضمن تعليمات يمكنها التعامل في نفس اللحظة مع البيانات التي لا يزيد طولها عن 16 ثنائية

*لذا لمعالجة مثل هذه البيانات(البيانات التي يزيد طولها عن 16 ثنائية) يلزمنا تجزئتها الى مجموعات طول كل منها يساوي 16 او8 ثنائيات، ومن ثم تنفيذ التعليمات اللازمة عدة مرات حسب عدد المجموعات

* في حين ان الحواسيب الكبيرة تستطيع معالجة البيانات الطويلة بتعليمة واحدة او بعدة تعليمات (عددها يكون اقل منه في حواسي (286))

*فمثلا لنفرض اننا نريد اضافة رقمين طول كل منهما 64 ثنائية بواسطة حاسوب كبير يحتوي على تعليمات تتعامل في نفس اللحظة مع البيانات التي يمثل طولها 64 ثنائية ففي هذا الحاسوب تتم عملية إضافة الرقمين بواسطة تعليمة واحدة فقط .

*اما لو اردنا اضافة نفس الرقمين بواسطة حواسيب 286 فيلزمنا تجزئة كل من الرقمين إلى مجموعتين طول كل منهما 16 ثنائية، ومن ثم جمع كل مجموعة على حدة بواسطة تعليمة واحدة .لذا يلزمنا تعليمتان اثنتان لجمع الرقمين

* ومن اجل تمكين حواسيب 286 من معالجة البيانات الطويلة بفعالية عالية يلزمنا استخدام معالجا رياضيا من نوع 80287 بالاضافة الى معالج 286يمكننا الاستنتاج انه لمعالجة البيانات الطويلة بفعالية في الحواسيب المصغرة يجب ان تحتوي هذه الحواسيب على معالجا رياضيا وتجدر الاشارة هنا الى أن المعالج الرياضي ((Math co-processor معالج خاص يستخدم في الحواسيب الميكروية لزيادة سرعة تنفيذ العمليات على البيانات الممثلة بصيغة الفاصلة العائمة))ضروري ايضا في حالة معالجة البيانات الممثلة بصيغة الفاصلة العائمة .

***تصنف البيانات في حواسيب 286 الى الانواع التالية:**

1- البيانات الرمزية

تمثل القيم الرمزية في حواسيب 286 حسب نظام ASCII .ويمكن ان يصل طول القيمة الرمزية الى 65536 رمزا .

2- البيانات العددية

***تصنف البيانات العددية حسب طريقة تمثيلها الى:**

- الاعداد الصحيحة :تتضمن حواسيب 286 تعليمات لمعالجة الاعداد التي يكون طولها 8 او16 ثنائية .اما في حالة استخدامنا لمعالج 287 فيمكن التعامل مع البيانات التي يصل طولها 32 او64 ثنائية .

-الاعداد العشرية :تمثل الاعداد العشرية حسب نظام BCD تماما كما في الحواسيب الكبيرة.

-الاعداد الممثلة بصيغة الفاصلة العائمة floating-point number تستطيع حواسيب 286 التعامل مع هذا النوع من الاعداد في حال احتوائها على معالج رياضي 287فقط .

ثانياً: بنية المعالج الميكروي 286

*ان اهم ما يمتاز به المعالج 286 هو استخدامه البنية الانبوبية pipelined architecture في تركيبه الداخلي مما زاد في قدرته في معالجة البيانات * البنية الانبوبية pipelined architecture تعني تقسيم المعالج الى وحدات وظيفية مستقلة بحيث تستطيع هذه الوحدات اداء مهامها على التوازي دون أن تعيق بعضها بعضاً

***يتكون المعالج الميكروي 286 من الناحية الوظيفية من 4 وحدات اساسية :**

1- وحدة التنفيذ Execution unit

*تتولى وحدة التنفيذ مهام تنفيذ التعليمات على المعطيات من اجل الحصول على النتائج .

*ونظرا لوجود التعليمات المشفرة في وحدة التعليمة فان تنفيذ التعليمات تتم بسرعة عالية .

*تتكون وحدة التنفيذ من :وحدة الحاسب والمنطق،وحدة التحكم تحتوي ذاكرة ROM، و8 مسجلات عامة الاغراض.

2- وحدة التعليمة Instruction unit

المهمة الاساسية لوحدة التعليمة هي تشفير التعليمات القادمة من وحدة الناقلة وتخزين التعليمات المشفرة في طابور خاص يتسع لثلاث تعليمات فقط . لذا تتكون وحدة التعليمات من محلل التعليمة instruction decoder ، وطابور التعليمات المشفرة.

3-وحدة الناقلة Bus Unit

تعتبر وحدة الناقلة بمثابة الوسيط بين المعالج 286 من جهه ، والذاكرة الرئيسة وحدات الادخال والاخراج من جهة اخرى. لذا فإن وحدة الناقلة تتولى مهمة الإشراف على عمليات تبادل البيانات بين المعالج ووحدات الحاسوب الاخرى .

4-وحدة العنوان Address unit

تتولى وحدة العنوان مهمة حساب العناوين الفعلية physical addresses لموقع الذاكرة الرئيسة المشاركة في التعليمات.لذا تحتوي على 4 مسجلات خاصة (تسمى مسجلات القطاعات Segment Registers)، ودائرة جمع الازاحة offset adder، ودائرة جمع العنوان الفعلي physical address adder.

***يتضح من الشكل رقم 17 صفحة 41 ان النموذج البرامجي للمعالج 286يتكون من :**

-مسجلات البيانات (AX ،BX ،CX ،DX)

- مسجلات القطاعات (ES،SS،DS ،CS)

-مسجلات العناوين والتأشير(DI،SI،SP ،BP)

-مسجلات خاصة(FLAGS،IP)

ثالثاً:صيغ التعليمات وطرق العنونة

تصنف تعليمات المعالج 286 حسب عدد عناوين المعلمات إلى: تعليمات صفرية العنوان ، وتعليمات أحادية العنوان ، وتعليمات ثنائية العنوان . ونتيجة لاختلاف عدد العناوين فإن طول التعليمة بين 1-6 بايت.

***ويمكن تصنيف تعليمات المعالج 286 تماما كما هو الحال في الحواسيب الكبيرة حسب نوع العملية الممثلة في التعليمة إلى المجموعات التالية:**

-1تعليمات نقل البيانات -2التعليمات المنطقية -3تعليمات معالجة سلاسل الرموز

-4تعليمات التحكم بالمعالج -5التعليمات الحسابية -6تعليمات الازاحة والتدوير -7تعليمات التحكم بالبرنامج

***اما بالنسبة لطرق العنونة ففي حواسيب 286 تستخدم طرق مشابهه لتلك المستعلمة في الحواسيب الكبيرة مثل:**

العنونة الفورية ، العنونة المباشرة ، العنونة الغير مباشرة ،العنونة النسبية باستخدام الأساس ، والعنونة النسبية باستخدام الفهرس ، وغيرها

رابعاً:الذاكرة الرئيسية ونظام الادخال والاخراج

****يعمل المعالج 286بأحد اسلوبي التشغيل التاليين:**

-1**أسلوب العنوان الحقيقي Real-address mode** حيث يعمل المعالج 286 بشكل مشابه للمعالجات 8086 و 8088 .وعندما يعمل المعالج 286 في هذا الأسلوب يمكنه التعامل مع ذاكرة رئيسية تصل سعتها لغاية 1 ميغابايت

-2**أسلوب العنوان التخيلي المحمي Protected Virtual address mode** يمتاز هذا الأسلوب بكون المعالج 286 يستطيع العمل بشكل متطور بالنسبة للمعالجات 8086 و 8088 وغير متوافق معها فمثلا عندما يعمل المعالج 286 في هذا الأسلوب يمكنه التعامل مع ذاكرة رئيسية تصل سعتها لغاية 1 غيغابايت غير انه تجدر الإشارة هنا الى ان سعة الذاكرة الرئيسية الفعلية تصل الى 16 ميغابايت فقط

**اما بالنسبة لنظام الادخال والإخراج المستخدم في حواسيب 286 وفي الحواسيب الصغيرة بشكل عام فهو لا يختلف عن نظام الادخال والإخراج المستخدم في الحواسيب الكبيرة سوى في قدرة وسرعة مكونات النظام مثل : وحدات الادخال ، ووحدات الإخراج ، ومعالجات الادخال والإخراج

*إن نوع الهيكلية المستخدمة في الحاسوب تحدد معظم خصائصه مثل: سرعة تنفيذ العمليات وصيغ التعليمات وانواعها .

*ان المعالج يحتوي على عدد محدود من المسجلات السريعة جدا، والتي تستخدم عادة لتخزين البيانات بشكل مؤقت .

*إن عملية نقل البيانات بين هذه المسجلات تتم بسرعة عالية مقارنة مع سرعة نقل البيانات بين المعالج والذاكرة الرئيسة لذا فإن التعليمات التي تكون معاملاتها مخزنة في الذاكرة الرئيسية تنفذ بسرعه اقل من تلك التعليمات التي تكون معاملاتها مخزنة في المسجلات وذلك لان عملية نقل المعاملات من الذاكرة الرئيسية الى المعالج تتطلب وقتا اضافية.

***تصنف الهيكليات المستخدمة في الحواسيب الى ثلاثة أنواع:**

أولاً: الهيكليات المركمية: Accumulator-based computer :هي الهيكلية التي تعتمد بشكل رئيسي في تنفيذ العمليات على المركم ،حيث تنفذ جميع التعليمات باستخدام المركم.

*تحتوي أبسط انواع المعالجات على مسجل خاص (أو مسجلين اثنين) يسمى مركما .ACC

*وفي كثير من الحواسيب يسمي المسجل الرئيسي نظرا لانه يستخدم في تنفيذ جميع التعليمات تقريبا .

*فمثلا، المعالج الميكروي المستخدم في الحاسوب الميكروي من نوع MCS48-، من انتاج شركة Intel ، يحتوي مركما واحدا، في حين أن المعالج الميكروي MC6809 ، من انتاج شركة Motorolla ،يحتوي على مر كمين اثنين . يبين الشكل 18 صفحة 45هيكلية المعالج ميكروي يحتوي مركماً واحداً.

*تمتاز المعالجات ذات الهيكليات المركمية بأن معظم تعليماتها أحادية العنوان، اي تحتوي عنوانا واحدا لتحديد المعاملات. غير أن من معظم التعليمات تحتاج الى معاملتين او ثلاث .

*لذا ففي الحواسيب ذات الهيكليات المركمية يفترض وجود احدى المعاملات في المركم في حين أن المعاملة الأخرى توجد في التعليمة نفسها. أما النتيجة فتخزن في المركم ثانية .

*فمثلا، إذا كانت X ترمز لعنوان المعاملة في الذاكرة الرئيسية، فإن التعليمة ADD X تؤدي الى جمع محتويات المركم مع المعاملة المخزنة في الموقع X وتخزين المجموع في المركم. ويمكن التعبير عن هذه العملية كما يلي: ACC = ACC + (X)

*ملاحظة :(X) تعني البيانات المخزنة في الذاكرة الرئيسة في العنوان.X

*يبين الشكل ١٩ صفحة ٤٦ مخطط سير العمليات ومراحل تنفيذ التعليمات في الحواسيب ذات الهيكليات المركبة.

*يستخدم مسجل البيانات DR لتخزين البيانات القادمة والمرسلة الى ذاكرة الرئيسية .اما المسجلة ARفيستخدم للتخزين.

*يحتوي مسجل التعليمة IR على code العملية للتعليمة الحالية. وبعد إحضار التعليمة وتخزين code العملية في المسجل IR ، يحدد نوع العملية(جمع) ، يرسل عنوان المعاملة X إلى الذاكرة الرئيسية عن طريق مسجل العنوان . ترسل الذاكرة الرئيسية المعاملة نفسها إلى مسجل البيانات في المعالج. وعندها تقوم وحدة الحساب والمنطق بتنفيذ العملية بجمع محتويات المركم محتويات مسجل البيانات وتخزين المجموع في المركم.

ثانياً:- الهيكليات ذات المسجلات العامة General-register computer

وهي الهيكلية التي تعتمد بشكل رئيسي في تنفيذ التعليمات على المسجلات، حيث تنفذ جميع التعليمات بإستخدام المسجلات .

*من أهم مساوئ الهيكليات المركمة قلة عدد المسجلات المسموح استخدامها من قبل المبرمج أثناء كتابة البرامج بلغة أسمبلي

*ومن أجل تخفيف حدة المشاكل تم زيادة عدد المسجلات العامة في المعالج ،وهكذا تم الانتقال الى الهيكليات ذات المسجلات العامة .

*من الامثلة على المعالجات التي تستخدم الهيكليات ذات المسجلات العامة . VAX-11 ,IBM 370, M68000 , Z8000 :

*يتراوح عدد المسجلات العامة في هذه المعالجات بين 8 و 16.

*الاطلاع على شكل ٢٠ الذي يمثل نموذج الهيكلية ذات المسجلات العامة صفحة ٤٧

*تمتاز الحواسيب التي تستخدم الهيكليات ذات المسجلات العامة بأن معظم تعليماتها ذات عنوانين أو ثلاثة عناوين . فمثلاً، لجمع القيمتين y ، xالمخزنتين في الذاكرة الرئيسية وتخزين المجموع في الموقعZ ، فإننا نستخدم تعليمة الجمع التالية ADD : Z,Y,X

***نلاحظ أن هذه التعليمة تحتوي على 3 عناوين :** يخصص الأول والثاني منها لتخزين القيم المراد جمعها ويخصص العنوان الثالث لتخزين النتيجة . ويمكن توضيح هذه التعليمة كما يلي (X) + (Y) = (Z):

والشكل ٢١ صفحة ٤٩ يبين مراحل تنفيذ تعليمة الجمع ثلاثية العنوان.

*أما في حالة جمع القيمتين X و Y وتخزين المجموع في X أو Y فإننا سوف نستخدم تعليمة الجمع ذات العنوانين التالية ADD X,Yوالتي تعني (X) + (Y) = (X)

ثالثاً :- الهيكليات المكدسيةStack computer

وهي الهيكلية التي تعتمد بشكل رئيسي في تنفيذ التعليمات على المكدس، حيث تنفذ جميع التعليمات بإستخدام المكدس.

*ان الحواسيب ذات الهيكلية المكدسية (stack) لا تحتوي على مسجلات عامة ولا تحتوي على مركمات ان العنصر الاساسي في تركيب معالجات هذا النوع من الحواسيب يسمى مكدساً.

***يعرف المكدس : Stack** ذاكرة تعمل حسب طريقة LIFO :Last-in First-out أي من يدخل أخيراً يخرج أولاً وهناك المكدس البرامجي والمكدس المادي. شكل 22 عمليات الدفع والسحب في المكدس

**ويمكن تشبيه المكدس بمخزن الطلقات في المسدس حيث يتم ادخال الطلقات الى المخزن واحدة تلو الأخرى فعند اطلاق طلقة فإنها تضغط سابقتها الى الداخل وتحل محلها اما عند الخروج في عملية اطلاق النار فإن اخر طلقة دخلت الى المخزن هي التي ستخرج أولا

***ويمكن اجراء عمليتين على المكدس**

1- ادخال القيم الى المكدس نسميها "الدفع" push

2- اخراج القيم من المكدس نسميها "سحب"pop

***حسب طريقه بناء المكدس يمكن تصنيفها الى نوعين**

1) المكدس المادي Hardware stack

ومن اجل بناء مكدس بسعة n قيمة يلزم استخدام m من مسجلات الازاحة

*ويمتاز المكدس المادي بسرعته العالية في تنفيذ عمليات الدفع والسحب التي تلزم في الحواسيب بشكل واسع لمعالجة الاعتراضات. *ومن أمثلة الحواسيب التي تستخدم المكدس المادي: IBM 370

2)**المكدس البرامجي Software stack** يبنى المكدس البرامجي عادة بتخصيص منطقة في الذاكرة الرئيسية بسعة n قيمة ويخصص كذلك احد مسجلات المعالج ويسمى مؤشر المكدس SP لتخزين عنوان اخر قيمة تم دفعها الى المكدس يقال ان مؤشر المكدس SP يشير دائما الى قمة المكدس

تمتاز الحواسيب ذات الهيكليات المكدسة بان معظم تعليماتها لا تحتوي اية عناوين أي العملية (صفرية) العنوان تتكون التعليمة من كود العملية فقط اما المعلمات المشاركة في التعليمة فيجب على المبرمج تخزينها في المكدس قبل البدء بتنفيذ العملية **ولتوضيح ذلك نشرح خطوات جمع قيمتين مخزنتين في مواقع الذاكرة الرئيسية X و Y

1- نقل محتويات الموقع X الى المكدس ؛ PUSH X

2- نقل محتويات الموقع Y الى المكدس ؛ PUSH Y

3- جمع القيمتين وتخزين المجموع في قمة المكدس ؛ ADD

*وتلاحظ ان تعليمة الجمع لا تحتوي اية عناوين وحسب هذه التعليمة تسحب اعلى قيمتين في قمة المكدس الى وحدة الحساب والمنطق حيث تجمعها هناك ثم يدفع المجموع الى قمة المكدس

*وكثيرا من الحواسيب لا يمكن تصنيفها الى احد أنواع الهيكليات السالفة الذكر وذلك لكونها تجمع بين خصائص عدة أنواع من الهيكليات فمثلا ، يحتوي المعالج الميكروي 8080 على 7 مسجلات عامة ويشكل المسجل A المركم لذا فالمعالج 8080 يجمع بين خصائص الهيكليات المكدسية والهيكليات ذات المسجلات العامة فمعظم التعليمات الحسابية والمنطقية تستخدم المركم A لذا فهي أحادية العنوان اما تعليمات نقل البيانات بين المسجلات فهي ثنائية العنوان وعلى الرغم من احتواء المعالج 8080 على مؤشر المكدس SP وتعليمات لدفع وسحب القيم من والى المكدس فهو لا يحتوي على تعليمات صفرية العنوان والتي تتميز بها الهيكليات المكدسية

*لكي يتمكن الحاسوب من انجاز المهمات المحددة في التعليمات يجب أن تحتوي التعليمة على كافة المعلومات اللازمة لذلك.

***تتكون التعليمة من حلقين اساسيين:**

أولاً:حقل كود العملية :operation code filed (op-code)

*يخصص حقل كود العملية لتحدد نوع العملية المراد انجازها . يعتمد طول الحقل على عدد تعليمات المعالج المستخدمة وأسلوب تشفير كود العمليات. Op-code encoding

*وتجدر الاشارة إلى ضرورة تخصيص كود لكل عملية بحيث يختلف عن باقي العمليات .

****هناك عدة أساليب لتشفير كود العملية في التعليمات وأهمها:**

١.اسلوب الكود الثابت الطول Block code

٢.اسلوب الكود الممتد Expanding op-code

٣.اسلوب كود هوفمانHuffman code

س: كيف تتم عملية تشفير العمليات حسب اسلوب الكود الثابت الطول ؟

يعتمد اسلوب الكود الثابت الطول على اعطاء شيفرة ذات طول ثابت لكل عملية .فإذا كان عدد العمليات التي يستطيع المعالج تنفيذها هو N ، فانه لتمثيل كل عملية يلزم حقل طوله m بت ويحدد حسب العلاقة في المثال صفحة 54.

ثانياً :حقل المعاملات:operand filed

يستخدم في بعض التعليمات لتحديد المعاملات المشاركة في العمليات .حيث يمكن للتعليمو ان تحتوي على المعاملات نفسها مباشرة او عنوان المعاملات اذا كانت مخزنة في الذاكرة الرئيسية أو رقم (اسم)احد المسجلات في المعالج .اي انه يمكن
***تخزين المعاملات في احد الأماكن التالية:**

١ .التعليمة نفسها ٢.المسجلات عامة الاغراض في المعالج ٣.مواقع الذاكرة الرئيسية ٤.منافذ الادخال

****يعتمد طول التعليمة على عدة عوامل منها:**

١)عددالحقول (عدد العناوين)في التعليمة، ٢)وطريقة العنونة المستخدمة للتعبير عن المعاملات

٣)اما عدد العناوين في التعليمة فيعتمد بدوره على نوع الهيكلية المستخدمة في المعالج.

***وعلى هذا الأساس يمكن تصنيف التعليمات حسب عدد العناوين فيها الى الانوا التالية:**

1) تعليمات صفرية العنوان 2) تعليمات احادية العنوان 3) تعليمات ثنائية العنوان 4) تعليمات ثلاثية العنوان

****انواع التعليمات هذه.**

١ **) تتكون التعليمة صفرية العنوان من حقل كود العملية فقط،** اي هذا النوع من التعليمات لا يحتوي حقولا للعناوين. لذا فإن التعليمات صفرية العنوان تستخدم بشكل رئيسي في الحواسيب ذات الهيكليات المكدسية.

٢ **)تتكون التعليمة احادية العنوان من حقلين اثنين** يخصص الاول منهما لكود العملية ويخصص الثاني للمعاملة نفسها او عنوانها. تستخدم التعليمات احادية العنوان غالبا في الحواسيب ذات الهيكليات المركمية.

٣ **)تتكون التعليمة ثنائية العنوان من٣ حقول هي**: حقل كود العملية. حقل المعاملة الاولى. حقل المعاملة الثانية.

تستخدم التعليمات ثنائية العنوان في الحواسيب التي تحتوي مسجلات عامة الاغراض.

٤ **)تتكون التعليمات ثلاثية العنوان من ٤ حقول هي:** - حقل كود العملية. - حقل المعاملة الاولى. - حقل المعاملة الثانية.

- حقل النتيجة.

*تستخدم التعليمات ثلاثية العنوان في الحواسيب التي تحتوي مسجلات عامة الاغراض ايضا غير انه لوحظ في الاونة الاخيرة عدم استخدام التعليمات ثلاثية العنوان واستبدالها بالتعليمات ثنائية العنوان مما ادى الى تقصير طول التعليمات.

*وتجدر الاشارة هنا الى انه في الحواسيب القديمة استخدمت تعليمات رباعية العنوان. فبالإضافة الى الحقول الاربعة الواردة في التعليمات ثلاثية العنوان تحتوي التعليمات رباعية العنوان على حقل خامس لتحديد عنوان التعليمة اللاحقةفي التنفيذ وقد استغني عن هذا الحقل الخامس نظرا لتنفيذ التعليمات بشكل متسلسل واستخدام عداد البرنامج لتخزين عنوان التعليمة اللاحقة في التنفيذ.

مثال :ولتوضيح تأثير عدد العناوين على طول البرنامج سوف نكتب برامجا بإستخدام انواع التعليمات الآنفة الذكر لحساب العلاقه الرياضية التالية:X=A*B+C*D **الحل صفحة58**

**نوضح بعض التعليمات والرموز التي سنستخدمها في البرامج.

- تعليمة الجمع ADD - تعليمة الضرب MUL - التحميل (نقل القيم من الذاكرة الى المركم) LOAD

- التخزين (نقل القيم من المركم الى الذاكرة) STORE - الدفع PUSH - السحب POP - التحريك MOVE

-المركم Acc -مسجات عامة الأغراض R1,R2 -قمة المكدس TOS -مواقع الذاكرة الرئيسية التي تحتوي القيم

وبعد دراسة المثال صفحة 57 يمكنك التوصل بسهولة الى الاستنتاجات التالية.

** في التعليمات صفرية العنوان تنجز جميع التعليمات باستخدام المكدس، لذا يلزم تخزين القيم المشاركة في العملية قبل البدء بتنفيذها فإذا اردت اجراء عملية معينة على قيمتين يتوجب عليك ان تخزن الاولى في قمة المكدس والثانيه في الموقع التالي بعد القمة في المكدس. اما النتيجة فتخزن في قمة المكدس.

-**في التعليمات احادية العنوان تنجز جميع التعليمات باستخدام المركم، حيث تخزن احدى القيم المشاركة في العملية في المركم مسبقا، وتخزن القيمة الاخرى في احد مواقع الذاكرة الرئيسة. اما النتيجة فتخزن في قمة المركم.

**في التعليمات ثنائية العنوان لايجوز اجراء العمليات على القيم المخزنة في مواقع الذاكرة الرئيسة مباشرة. بل يجب ان تكون الأولى في احد المسجلات، والثانية- في الذاكرة الرئيسة او في احد المسجلات. اما النتيجة فتخزن في محل إحدى القيمتين.

- يزداد عدد التعليمات في البرنامج بتقليل عدد العناوين في التعليمات. غير ان طول التعليمة يزداد بزيادة عدد العناوين فيها.

الوحدة الثانية : بنية المعالج الدقيق

**** التركيب الوظيفي للمعالج الدقيق (الميكروي)**

** يعرف المعالج بأنه عبارة عن مجموعه من الألكترونية المبنية على رقاقة (شريحة) مطبوعة من مواد اشباه الموصلات مثل السيليكون . تحاط هذه الرقاقة بغلاف بلاستيكي او سيراميكي يخرج منه اطراف معدنية يعتمد عددها من نوع المعالج .

* تستخدم هذه الأطراف عادة لتثبيت الرقاقة على اللوحات المطبوعه . و كذلك لتوفير امكانية نقل المعلومات واشارات التحكم بين رقاقة المعالج والرقاقات الاخرى المكونة لنظام الحاسوب . * فمثلا تحتوي رقاقة المعالج 8088 على 40 طرف .

*تقسم اطراف المعالج حسب الاشارات التي تمر من خلالها الى المجموعات التالية :

1 - أطراف العنوان (عددها 20) تستخدم لنقل العناوين من المعالج الى رقاقات الذاكرة الرئيسية و رقاقات التوسط المتصلة مع وحدات الادخال والاخراج من جهة اخرى

2 - اطراف البيانات (عددها 8) تستخدم لنقل البيانات والتعليمات بين رقاقة المعالج من جهة و رقاقات الذاكرة الرئيسية والرقاقات المسانه (Suppere dips)

3 - اطراف التحكم (عددها 15)تستخدم لنقل اشارات التحكم بين المعالج و باقي مكونات الحاسوب من جهة اخرى

4 - اطراف التغذية الكهربائية والتوقيت (عددها 4) وتستخدم لتزويد الدوائر الالكترونية المكونه للمعالج بالطاقة الكهربائية وكذلك لتحديد مواعيد البدء بتنفيذ جميع العمليات في الحاسوب عن طريق اصدار اشارات التوقيت اللازمة في الوقت المناسب .

*لو قمت بجمع عدد الاطراف لوجدت ان مجموعها (47) اكثر من 40 طرفا خلافا لما ذكر اعلاه ولحل هذه المشكلة يلزمنا استخدام بعض الاطراف لاداء اكثر من وظيفة واحدة .

*يتم تنظيم استخدام الأطراف متعددة المهام عن طريق تنفيذ كل مهمة في وقت محدد

*اي ان هناك اطراف تم تخصيصها لاداء مهمة واحدة وهناك اطراف اخرى تستخدم لاداء اكثر من مهمة واحدة ويتم تنظيم استخدام الاطراف متعددة المهام عن طريق تنفيذ كل مهمة في وقت محدد فمثلا هناك اطراف تستحدم لنقل البيانات والعناوين حيث تحمل هذه الاطراف العنوان في وقت معين ثم تحمل البيانات في وقت اخر .

**المعالجات 8088 و 8086 فالتركيب الداخلي لهذين المعالجين يسمح بتنفيذ العمليات على البيانات التي طولها 8 ثنائيات او 16 ثنائية في نفس اللحظة . غير ان عدد الثنائيات التي يتم تبادلها بين المعالج من جهة و باقي وحدات الحاسوب من جهة اخرى .

**يختلف من معالج الى اخر فوحدة نقل البيانات في المعالج 8088 تساوي 8 ثنائيات في حين ان وحدة نقل البيانات في المعالج 8086 تساوي 16 ثنائية وهذا يفسر ان عدد اطراف البيانات في المعالج 8086 هو 16 طرفا في حين ان عدد اطراف البيانات في المعالج 8088 يساوي 8 ثنائيات

*وهذا يشكل فرقا اساسيا بين المعالجين بالطبع يؤدي الى ان سرعة نقل البيانات في الانظمة التي تستخدم المعالج 8086 تكون اكبر منها في الانظمة التي تحتوي المعالج 8088 .

**واخيرا نشير الى ان كل من المعالج 8086 و 8088 يستطيع العمل باحد اسلوبي التشغيل :

1- اسلوب النظام الصغير Minimum mode : يستخدم عادة في الانظمة الصغيرة مثل ماكنات الغسيل الاوتوماتيكية حيث يمكن تشغيل مثل هذه الماكنات بواسطة المعالج 8088 الذي يعمل في اسلوب النظام الصغير

2 - اسلوب النظام الكبير Maximum Mode : يستخدم عادة في التطبيقات الاكثر تعقيدا مثل الحواسيب المصغرة و الحواسيب متعددة المعالجات

**** مكونات المعالج الدقيق (الميكروي)8086/8088**

*تستخدم المعالجات عادة لتنفيذ تعليمات البرنامج الواحدة تلو الاخرى مرورا بمرحلة الاحضار ومرحلة التنفيذ واللتين يمكن تلخيصهما كما يلي :

1- احضار التعليمة التالية من الذاكرة الرئيسية

2- احضار المعاملات اللازمة لتنفيذ التعليمة ان لزم ذلك

3- - تنفيذ التعليمة على المعاملات

4- تخزين النتائج في المكان المناسب ان لزم ذلك

*فالمعالجات التي سبقت المعالج 8086 صممت بحيث تنفذ هذه الخطوات على التوالي

*ومن اجل زيادة سرعه معالجة التعليمات يمكن زيادة تردد الساعه clock او تطوير البنية الداخلية للمعالج بحيث يستطيع تنفيذ هذه الخطوات على التوازي . ان هذه البنية الجديدة اصبحت معروفة باسم البنية الانبوبية pipline

*** ان تقسيم دورة التعليمية الى مرحلتين قد حذا بالمهندسين والمصممين الى تقسين المعالج 8086/8088 الى وحدتين منفصلتين و مستقلتين بعضهما عن بعض هما :**

1 - وحدة المواجهة البينية Bus Interface Unit "BIU"

2 - وحدة التنفيذ Excution Unit "EU"

*فوحدة التوسط تتولى مهمة تنفيذ الخطوات (4.2.1) اما وحدة التنفيذ تتولى مهمة تنفيذ الخطوة الثالثة فقط .

****من الشكلين (شكل 1و2ص84) يمكنك الوصول إلى الاستنتاجيات التاليه:**

1.تبلغ فعالية استخدام المعالج 8080حوالي 50٪ حيث يقضي معظم وقته في إنتظار انتهاء عمليات تبادل البيانات والتعليمات بين المعالج والناقله.

2.تصل فعالية الاستخدام الناقلة في نظام 8080الى حوالي 50٪ايضا بسبب إنتظار تنفيذ العمليات في المعالج

3.تصل فعالية استخدام المعالج والناقلة في نظام 8088الى حوالي 100٪حيث يعمل كل من المعالج والناقلة بدون انقطاع ،ويعود السبب في ذلك إلى تقسيم المعالج إلى وحدتي التوسط والتنفيذ اللتين تعملان بشكل منفصل ومستقل .

*ففي نظام 8080عندما ينهي المعالج تنفيذ التعليمه التاليه ينتقل إلى حالة الانتظار انتهاء عملية إحضار التعليمه الحاليه ،وبعد ذلك يستأنف المعالج 8080عملية التنفيذ .

* وهذا يعني أنه في نظام 8080في أية لحظة زمنية يعمل المعالج أو الناقلة بشكل متناوب .

*أما في 8088ففي الفترة التي ينفذ فيها المعالج التعليمه الحاليه فإن الناقله تعمل على إحضار التعليمه اللاحقة . فعندما ينتهي المعالج من تنفيذ التعليمه الحاليه تكون الناقلة قد أتمت عملية إحضار التعليمه اللاحقة والتي يبدأ المعالج فوراً بتنفيذها .

*وهذا يعني أنه في نظام 8088في لحظة زمنية يعمل المعالج والناقلة معا على التوازي وليس على التناوب كما هو الحال في نظام 8080.

***وحدة التنفيذ EU.**

***تتكون وحدة التنفيذ من اربع اجزاء أساسية هي:**

1.وحدة الحساب والمنطق

2.وحدة التحكم بوحدة التنفيذ

3.مسجلات عامة الاغراض

4.مسجل الحالة

تستخدم وحدة الحساب والمنطق لتنفيذ مختلف أنواع العمليات على البيانات المدخلة إليها عن طريق الناقلة الداخلية وكذلك نقل نتائج تنفيذ العمليات إلى الأماكن المناسبة مروراً بالناقله الداخلية أيضاً.

*وكما هو موضح في الشكل 3ص86 تتصل عناصر وحده التنفيذ بعضها مع بعض بواسطة الناقلة الداخلية ولا يوجد وسائل اتصال مباشرة بين وحدة التنفيذ والعالم الخارجي . فالوسيلة الوحيدة للاتصال بينهما تمر من خلال وحدة التوسط.

وحدة المواجهة البينية BIU

*تتكون وحدة التوسط من الأجزاء التاليه:

1.دائرة الجمعadder

2.مجموعة المسجلات peen

3.طابور التعليمات

4.وحدة التحكم بوحدة التوسط.

*ذكرنا سابقاً أن وحدة التوسط تتولى مسؤولية عمليات تبادل البيانات بين وحدة التنفيذ وباقي أجزاء الحاسوب. لذا تستخدم دائرة الجمع والمسجلات (مسجلات القطاعات)لحساب عناوين البيانات .

* أما طابور التعليمات فيستخدم لتخزين التعليمات (أو أجزاء التعليمات)التي تم إحضارها مسبقاً.

*ويختلف طول طابور التعليمات في المعالج 8086 عنه في المعالج8088.ففي المعالج 8088طول طابور التعليمات يساوي 4بايت.اما في المعالج 8086فإن طول طابور التعليمات يساوي 6بايت

***يستخدم طابور التعليمات على النحو التالي :** عندما تكون وحدة التنفيذ مشغولة بتنفيذ التعليمة الحالية فإن وحدة المتوسط تتولى مهمة إحضار التعليمة التالية وتخزينها في طابور التعليمات لحين الانتهاء من تنفيذ التعليمة الحالية . فعندما تفرغ وحدة التنفيذ من التعليمة الحالية فإنها تنتقل فوراً الي تنفيذ التعليمة التالية الموجودة في طابور التعليمات وهذا بدوره يلغي زمن انتظار وحدة التنفيذ اللازم لإحضار التعليمة التالية من الذاكرة الرئيسية .

*إن هذا ينطبق على الحالات التي تكون فيها التعليمة الحالية والتعليمة التالية مخزنه في مواقع متتالية في الذاكرة الرئيسية ، غير انه في حالة تعليمات نقل التحكم تكون التعليمة التالية مخزنه في مواقع بعيدة من التعليمة الحالية ، لذا في مثل هذه الحالات فإن طابور التعليمات لا يحتوي التعليمة التالية في التنفيذ . ولحين تقوم وحدة التوسط بإحضار التعليمة التالية من موقعها فإن وحدة التنفيذ تبقى في حالة إنتظار . وبعد ذلك تبدأ وحدة التنفيذ بتنفيذ التعليمة التي تم إحضارها . وخلال هذه الفترة تعمل وحدة التوسط على إحضار التعليمة التالية من الموقع الجديد .

*وتجدر الإشارة هنا إلى أن طول جميع المسجلات في وحدتي التنفيذ والتوسط يساوي 16 ثنائية ، وكذلك فإن طول وحدة الحساب والمنطق يساوي 16 ثنائية .

****نظام النواقل BUS**: تعتبر الناقلة وسيلة لربط الوحدات المكونة للحاسوب بعضها مع بعض بالإضافة لكونها الوسيلة التي من خلالها يرتبط المعالج الدقيق مع العالم الخارجي .

*يتكون نظام النواقل 8086/8088 من ثلاث أنواع :

1. خطوط التحكم : تستخدم لنقل إشارات التحكم في المعالج الي وحدات التحكم الأخرى وبالإتجاه المعاكس أيضا تبعا لنوع الإشارة. ومن الأمثلة على خطوط التحكم : خطوط الاعتراض INTA, INTR , NMI ,..... وغيرها وهناك أيضا مجموعة من الخطوط للتغذيه الكهربائية وإشارات الساعة .

1. ناقل العنوان 2.ناقل البيانات :

*تستخدم خطوط العنوان لنقل عناوين مواقع الذاكرة الرئيسية أو عناوين وحدات الادخال والإخراج

* يصل عدد خطوط العنوان في المعالج 8086/8088 إلي 20 خطا

*وتجدر الإشارة هنا الي أن خطوط العنوان تنقل العناوين باتجاه واحد ، من المعالج الي الذاكرة الرئيسية أو وحدات الادخال والإخراج وليس العكس وبما أن عدد خطوط العنوان هو 20 خط فإن سعة الذاكرة الرئيسية التي يمكن عنونتها تصل لغاية 1 ميغا بايت (أي 220).

* يستخدم ناقل البيانات لنقل البيانات والتعليمات بين المعالج الدقيق من جهة ، والذاكرة الرئيسية والرقاقات المساندة ووحدات الادخال والإخراج من جهة أخرى وهذا يعني أن خطوط البيانات تعمل بإتجاهين ويصل عدد خطوط البيانات في المعالج 8086 الي 16 خطا ، في حين أن عدد خطوط البيانات في المعالج 8088 يساوي 8 خطوط

*وتجدر الإشارة إلي أن عدد خطوط البيانات يحدد سرعة نقل البيانات

*ولغايات التوفير في عدد الخطوط الواصلة بين المعالج 8086 والعالم الخارجي فإن العناوين والبيانات تشترك في نفس الخطوط : من AD0 إلي AD15 . اي ان هناك فقط 16 خط تستخدم لنقل العناوين والبيانات ، حيث تحمل هذه الخطوط العناوين في أوقات محددة ، وتحمل البيانات في أوقات أخرى

*أما في المعالج 8088 فإن خطوط (من AD0 الي AD7) تكون مشتركة نظراً لكون عدد خطوط ناقلة البيانات يساوي 8 فقط .

* أما بالنسبة لخطوط ناقلة العناوين الباقية (من A16 إلىA19) فهي تستخدم لنقل المنازل الأربعة في أقصى يسار العنوان في لحظات محددة ، وتحمل معلومات تحدد مسجل القطاع المستخدم حاليا في لحظات اخرى .

**المسجلات Registers

*يحتوي المعالج الميكروي 8086/8088 على مجموعة من المسجلات التي تستخدم لأغراض الحاسوب ولا يستطيع المبرمج استخدامها في برامجه .

* من الأمثلة على هذه المسجلات : مسجل التعليمة IR والمسجلات المرقنة ، بالإضافة إلى هذه المسجلات يحتوي المعالج 8086/8088 على مجموعة من المسجلات التي يستطيع المبرمج استخدامها في برامجه .

*** تصنف هذه المسجلات إلى المجموعات التالية :**

1- مسجلات المعطيات Data registers : المسجلات DX,CX,BX,AX ، طول كل منهما 16 ثنائية أو المسجلات AL,DH,DL,CH,CL,BH,BL,AH طول كل منهما 8 ثنائيات . تستخدم مسجلات المعطيات لتخزين كافة أنواع البيانات .

2- مسجلات الفهرسة والتأشير Index and Pointer Registers :

▪**مسجلات الفهرسة:** المسجلات SI , DI طول كل منهما 16 ثنائية تستخدم لعنونة البيانات في قطاع الإضافة أو قطاع البيانات

▪**مسجلات التأشير :** المسجلات BP,SP طول كل منهما 16 ثنائية تستخدم للتأشير إلى البيانات المخزنة في قطاع المكدس .

3- مسجلات القطاعات Segment Registers : المسجلات ES,SS,DS,CS طول كل منهما 16 ثنائية تستخدم لتخزين عنوان الأساس لقطاعات الذاكرة المنشطة .

4- **مسجلات حالة البرنامج Program Status Register :** المسجلات IP و Flags طول كل منهما 16 ثنائية تستخدم لتحديد حالة البرنامج والحاسوب بعد تنفيذ كل تعليمة.

* الثنائيات المكونة للمسجل ترقّم من اليمين الى اليسار ابتداءً من الصفر . وأن الثنائية الواقعة في أقصى يسار المسجل (الثنائية رقم 7 إذا كان طول المسجل يساوي 8 ، الثنائية رقم 15 إذا كان طول المسجل يساوي 16 ثنائية) ، يستخدم لتمثيل إشارة الرقم .

* يستطيع المبرمج إستخدام المسجلات في برامجه المكتوبة بلغة الأسمبلي ، ولتحقيق هذه الإمكانية زوّد كل مسجل باسم خاص به يميزه عن غيره من المسجلات (باستثناء مسجل الحالة الذي لا يحمل اسم ولهذا لا يستطيع المبرمج التعامل معه كوحدة واحدة ، بل يمكن للمبرمج أن يتعامل مع بعض الثنائيات المكوّنة لمسجل الرايات) ' الشكل4 صفحة 89 يبين النموذج البرامجي للمعالج 8086/8088 والذي يتكون من 14 مسجلاً .

أولا: المسجلات عامة الأغراض General purpose Reg

* تستخدم مسجلات المعطيات لتخزين مختلف أنواع البيانات بالإضافة إلى الإستخدامات الخاصة لكل مسجل منها .

* يحتوي المعالج 8086/8088 على أربعة مسجلات للمعطيات هي : CX,BX,AX,DX وطول كل واحدة منها يساوي 16 ثنائية .

* مسجلات المعطيات تستخدم بشكل عام لتخزين البيانات التي يمكن تمثيلها بواسطة 16 ثنائية .

* ما يميز مسجلات المعطيات أنه يمكن تقسيم كل مسجل إلى نصفين طول كل منهما يساوي 8 ثنائيات .

* وبهذا ينتج 8 مسجلات هي : BH,BL,AH,AL,DH,DL,CH,CL انظر الشكل 4 .

* ان النصف الأيسر في كل طل مسجل من مسجلات المعطيات يحمل الرمز H في اسمه (مأخوذة من High-order half) وأن النصف الأيمن يحمل الرمز L في اسمه (مأخوذة من Low-order half) .

* من هنا يمكن الاستنتاج أن مسجلات المعطيات يمكن استخدامها كمسجلات طولها 16 ثنائية أو كمسجلات طولها 8 ثنائيات حسب اسم المسجل المستخدم . فإذا استعملت مثلا ، الإسم BX فإنك تشير إلى المسجل كاملاً بطول 16 ثنائية أما إذا استعملت الإسم BL فإنك تشير إلى النصف الأيمن من المسجل BX .

* الاستخدامات الخاصة لمسجلات المعطيات : تتطلب بعض العمليات استخدام مسجلات معينة، فإذا استخدمت مسجلات غير تلك المحددة في هذه العمليات فإنك تكون قد وقعت في خطأ يبلغك الحاسوب عنه .

*** من الأمثلة على الإستخدامات الخاصة للمسجلات :-**

▪ يستخدم المسجل AX في عمليات ضرب وقسمة البيانات التي طولها 16 ثنائية ، ويجب أيضاً استخدام المسجل AX في عمليات ادخال أو إخراج البيانات التي طولها 16 ثنائية (البيانات التي طولها 16 ثنائية تسمي الكلمات Words).

▪ يستخدم المسجل AL في عمليات ضرب وقسمة البيانات التي طولها 8 ثنائيات ، (البيانات التي طولها 8 ثنائيات تسمي بايتات Bytes) ،ويستخدم المسجل AL في إدخال وإخراج البيانات والعمليات على البيانات العشرية.

▪ يستخدم المسجل AH في عمليات ضرب وقسمة البايتات .

▪ يستخدم المسجل BX لتخزين عنوان بداية القوائم في عمليات البحث في القوائم.

▪ يستخدم المسجل CX في عمليات معالجة سلاسل الرموز String operations ،وكذلك لتخزين عدد مرات تكرار الدورانات Loops .

▪ يستخدم المسجل CL في عمليات الإزاحة والتدوير Shift and Rotate .

▪ يستخدم المسجل DX في عمليات ضرب وقسمة الكلمات وكذلك في عمليات الإدخال والإخراج غير المباشر لتخزين رقم الموانئ المستخدمة.

ثانيا: مسجلات التأشير والفهرسة (Index Reg)

* يحتوي المعالج 8086/8088 أربعة مسجلات تسمى مسجلات التأشير والفهرسة وهي : DI,SI,BP,SP .ونظراً لأن هذه المسجلات تستخدم غالباً لتخزين و حساب العناوين فإنها أيضاً تسمى مسجلات العناوين.

* يبلغ طول كل مسجل من مسجلات التأشير والفهرسة 16 ثنائية. يستخدم كل مسجل من هذه المسجلات كوحدة واحدة ولا يمكن تجزئتها كما هو الحال في مسجلات المعطيات.

* يمكن استخدام المسجلات SI(Source Index) , DI (Destination Index) لتخزين مختلف أنواع البيانات بالإضافة إلى استخداماتها الأساسية في تعليمات معالجة سلاسل الرموز .

* يستخدم المسجل Stack Pointer SP لتخزين قمة المكدس، لذا فهو يشارك في عمليات المكدس.

* يستخدم المسجل Base Pointer BP في حساب العنوان الفعلي للبيانات المخزنة في قطاع المكدس . وسيتم توضيح هذا الموضوع لاحقاً .

ثالثا: مسجلات القطاعات Segment Reg

* ان سعة الذاكرة الرئيسة في الحواسيب المبينة باستخدام المعالج 8086/8088 تصل لغاية 1 ميغابايت .

* من أجل زيادة فعالية استخدام هذه السعة تقسم الذاكرة إلى قطاعات منطقية يصل طول القطاع الواحد لغاية 64 كيلو بايت . أي أن عدد القطاعات لا يقل عن ١٦ قطاع

*غير أن المعالج 8086/88088 يستطيع التعامل في نفس الوقت مع ٤ قطاعات فقط. وهذا يعني أن هناك أربعة قطاعات نشيطة في أية لحظة . ويمكن دائما تحويل القطاع من حالة نشطة إلى حالة خاملة والعكس صحيح .

*يعرف القطاع بأنه عبارة عن مجموعة من مواقع الذاكرة الرئيسية (يصل عددها لغاية ٦٤ كيلو بايت) والتي تبدأ بموقع يكون عنوانه من مضاعفات الرقم ١٦ مثل العناوين: 0.16.32

*** وتصنف القطاعات حسب نوع محتوياتها إلى اربع انواع:**

١. قطاعات الكود (التعليمات) Code segment يحتوي على تعليمات فقط

٢. قطاع البيانات data segment يحتوي على بيانات

٣. قطاع المكدس Stack segment فيحتوي أية بيانات تخزن من قبل الحاسوب أو المبرمج لحين الحاجة .

٤. القطاع الإضافي Extra segment يحتوي قطاع الاضافي أيضا بيانات ويمكن الاستغناء عنه إذا كان حجم البيانات المستخدمة بالبرنامج قليلا.

*ومن هنا يمكن الاستنتاج انه في البرامج المكتوبة في لغة اسمبلي يجب الفصل بين الكود والبيانات.

* إي أن البرنامج المكتوب بلغة الاسمبلي يجب أن تكون من 3قطاعات على الاقل: قطاع للكود واخر للبيانات وثالث للمكدس وطبعا يمكن للبرنامج أن يحتوي أكثر من قطاع واحد من الأنواع الآنفة للذكر

***ومن أجل تحديد القطاعات الاربعة النشطة في أية لحظة يحتوي المعالج 8086/88088 أربعة مسجلات تسمي مسجلات القطاعات وهي:**

-مسجل قطاع الكود Cs code segment register : يشير المسجل cs إلى قطاع الكود النشط اي القطاع الذي تنفذ تعليماته في الوقت الحالي.

- مسجل قطاع البيانات DS data segment register : فيشيرالمسجل ds الى قطاع البيانات النشط اي القطاع الذي يحتوي متغيرات البرنامج الذي ينفذ حاليا.

- مسجل قطاع المكدس ss Stack segment register : * يشير المسجل ss الى قطاع المكدس النشط اي القطاع الذي تنفذ عمليات المكدس على محتوياته حاليا .

- مسجل القطاع الاضافي es extra segment register : يشير المسجل ES إلى القطاع الاضافي النشط اي القطاع الذي يحتوي متغيرات البرنامج أثناء تنفيد بعض التعليمات التي تتطلب استخدام القطاع الاضافي.

*تستخدم مسجلات القطاعات لتخزين عنوان بداية القطاع المقابل . وبما أن حجم القطاع يصل لغاية ٦٤ كيلوبايت فإن طول مسجل القطاع يجيب أن يكون مساويا ١٦ ثانية .

**ومن الجدير بالذكر هنا أن المبرمج يستطيع تغير محتويات مسجلات القطاعات من داخل برنامجه ولكن هذا قد يؤدي إلى الوقوع في اخطاء لذا يجب أن يتعامل المبرمج بحذر شديد مع التعليمات التي تؤثر على محتويات مسجلات القطاعات.

****مسجلات حالة البرنامج Status Reg :**

تستخدم مسجلات حالة البرنامج لتحديد حالة البرنامج والحاسوب بعد تنفيذ كل تعليمة من التعليمات .

***تتكون هذه المجموعة من مسجلين طول كل منهما ١٦ ثنائية وهما :**

1. مؤشر التعليمة (IP (instructions point فمؤشر التعليمة IP يشير دائما إلى التعليمة التالية في التنفيذ بعد الانتهاء من تنفيذ التعليمة الحالية . يشبه مؤشر التعليمة عداد برنامج PC في المعالجات التي استخدمت من قبل المعالج8086/8088.

***تقوم وحدة المتوسط BIU** بتعديل قيمة مؤشر التعليمة Ip بحيث يحتوي دائما عنوان التعليمية اللاحقة بالنسبة لعنوان بداية قطاع الكود النشط .

* وتجدر الإشارة إلى أن المبرمج لا يستطيع تغيير قيمة مؤشر التعليمة بواسطة تعليمات خاصة . وانما تتغير ip اتوماتيكيا حسب طول التعليمات المكونة للبرنامج ونوع تلك التعليمات.

2,**مسجل الرايات Flags register** يتكون مسجل الرايات من 16 ثنائية يستخدم من 9 فقط أما باقي الثنائيات السبعة فهي لا تستخدم في المعالج 8086/8088.

*ولقد تركت هذه الرايات للتطويرات المستقبلية أو لاستخدمات المبرمجين ان لزم ذلك.

***تقسم الرايات إلى مجموعتين:** رايات الحالة Status Reg وعددها ستة ورايات التحكم control flags وعددها ثلاثة.

*تستخدم رايات الحالة للتعبير عن خصائص نتائج العمليات الحسابية والمنطقية فطاقم التعليمات المعالج 8086/8088 يتضمن تعليمات للتحكم في تسلسل تعليمات البرنامج بالاعتماد على حالة هذه الرايات . اي الاعتماد على نتيجة العملية السابقة
* ليس كل التعليمات لها تأثير على الرايات فلكل تعليمة تأثير يختلف عن التعليمات الأخرى .شكل (5): مسجل الرايات

****رايات الحالة :**

- راية الحمل (cary) : تحتل الثنائية رقم صفر ويرمز لها بحرف .C تعكس راية الحمل وجود (أو عدم وجود) حمل من الثنائية في اقصى اليسار أو حصول استقراض منه بعد تنفيذ بعض التعليمات : الجمع، والطرح، والتدوير، ... وغيرها، فحصول حمل يخول راية الحمل إلى حالة "1"، وإلا - إلى حالة "0" .

-راية التطابق (Parity) : تحتل الثنائية رقم "2" ويرمز لها بحرف P. تعكس هذه الراية التطابق الزوجي ، والذي يعني أن عدد الثنائيات في النتيجة التي تحتوي "1" زوجيا. (فإذا كان عدد هذه الثنائيات زوجية فإن راية التطابق تحتوي "1"، وإلا فتحتوي "0". يستخدم مفهوم التطابق بشكل عام للتأكد من صحة عمليات تخزين ونقل البيانات .

-راية الحمل المساعد: تحتل الثنائية رقم "4" ويرمز لها بحرف A تعكس هذه الراية وجود او عدم وجود حمل من الثنائية رقم من الثنائية رقم" 3" الى الثنائية رقم"4" أو حصول استقراض من الثنائية رقم "4" الى الثنائية رقم "3" أثناء تنفيذ التعليمات على البيانات العشرية. حصول مثل هذا الحمل او الاستقراض يحوّل راية المحمل المساعد الى حالة "1" والا الى حالة "0".

-راية الصفر (Zero) : تحتل الثنائية رقم "6" ويرمز لها بحرف Z، تعكس هذه الراية مساواة نتيجة العملية المنفذة الى الصفر. فإذا كانت نتيجة العملية تساوي صفراًفإن راية الصفر تتحول إلى حالة "1"، وإلا إلى حالة "0" .

-راية الاشارة : تحتل الثنائية رقم "7" ويرمز لها بحرف S. تعكس هذه الراية اشارة نتيجة العملية المنفذة حيث تتحول راية الاشارة الى حالة "1" إذا كانت النتيجة سالبة اما اذا كانت موجبة او مساوية للصفر فإن راية الاشارة تتحول الى "0".

-راية الفيض (Overflow): تحتل الثنائية رقم "11" ويرمز لها بحرفO . تعكس هذه الراية امكانية تخزين نتيجة العملية المنفذة في المكان المخصص لها. حيث تتحول راية الفيض الى حالة "1" إذا كان المكان المخصص لتخزين النتيجة لا يتسع لتخزينها ، وإلا فإن هذه الراية تكون في حالة "0" . ونشير هنا أن حصول فيض يسبب فقدان جزء من النتيجة وبالتالي

الحصول على نتائج خاطئة. لذا فعند حصول فيض بعد تنفيذ اية عملية فإن الحاسوب يضع راية الفيض في حالة "1" مبلغ المبرمج عن ذلك ، وبعد ذلك ينفذ الحاسوب برنامج خاص للمعالجة هذه الحالة وإنهاء تنفيذ البرنامج الذي ادى الى حصول فيض .

*أما رايات التحكم فتستخدم عادة للتأثير على سير عمل المعالج اثناء تنفيذ البرامج .توضح معنى رايات التحكم كل واحدة على حدة.

-راية المصيدة (Trap): تحتل الثنائية رقم"8" ويرمز لها بحرف T. تعكس هذه الراية اسلوب تشغيل المعالج، فوضع راية لمصيدة في حالة "1" يحول المعالج إلى اسلوب «الخطوة خطوه»، أي التوقف بعد تنفيذ كل تعليمه وإظهار تأثير المصيدة) التعليمة المنفذه على الرايات ومسجلات المعالج ومواقع الذاكره ذات العلاقة. أما إذا وضعت راية المصيدة في حالة "0" فإن المعالج ينفذ تعليمات البرنامج مرة واحده بدون توقف بعد تنفيذ كل تعليمه. إن استخدام راية الأثر في اسلوب الخطوه خطوه يساعد في اكتشاف الأخطاء في البرامج بالاضافة الى توضيح عمل كل تعليمه من تعليمات المعالج.

-راية الاعتراض (Interrupt): تحتل الثنائية رقم "9" ويرمز لها بحرف I. تحدد هذه الراية إمكانية قبول (أو عدم قبول) الاعتراضات القادمة من وحدات الحاسوب الى المعالج، فإذا وضعت راية الاعتراض في حالة "1" فإن المعالج يتعرف على الاعتراضات الخارجية ويوقف تنفيذ البرنامج الذي يشغله حاليا وينتقل إلى تنفيذ برنامج معالجة الاعتراض المناسب. أما إذا وضعت راية الاعتراض في حالة "0' فإن المعالج لا يتعرف على الاعتراضات مما يؤدي إلى إهمالها . سنوضح الاعتراضات بالتفصيل لاحقا ان شاء الله .

-راية الاتجاه (Direction) : تحتل الثنائية رقم "10" ويرمز لها بالحرف D. تستخدم راية الاتجاه لتحديد إتجاه معالجة البيانات في تعليمات سلاسل الرموز (String Instructions). فإذا وضع راية الاتجاه في حالة "0" فإن سلاسل الرموز تعالج من اليسار الى اليمين. أما إذا وضعت راية الاتجاه في حالة "1"فإن سلاسل الرموز تعالج من اليمين الى اليسار .

****مصطلحات مهمة :**

****تعددية البرامجMultiprogramming:** اسلوب تنفيذ عدة برامج مخزنة في الذاكره الرئيسية حسب سياسة أولويات معينة.

****القطاع الإضافي Extra segment** قطاع يستخدم لتخزين البيانات في حالات خاصة مثل تعليمات معالجة سلاسل الرموز

****قطاع البيانات Data segment** قطاع يستخدم لتخزين البيانات بشكل عام

***قطاع الذاكرة Memory segment** مجموعة مواقع في الذاكره الرئيسية تبدأ بعنوان من مضاعفات الرقم 16 وتصل سعة القطاع لغاية 64 كيلوبايت .

***قطاع الكود Code segment** قطاع يستخدم لتخزين تعليمات البرنامج.

***قطاع المكدس Stack segment** قطاع يستخدم لتخزين معلومات خاصة من قبل الحاسوب أو المستخدم .

****مؤشر التعليمة Instruction pointer:** مسجل (IP) يستخدم لتخزين مقدار ازاحة التعليمة التالية في التنمية بالنسبة لبداية قطاع الكود المنشط .

****ناقلة التحكم Control bus** مجموعة من الخطوط المستخدمة لنقل إشارات التحكم من وإلى المعالج

****ناقلة العنوان Address bus** مجموعة من الخطوط (20 خطا) المستخدمة لنقل عناوين مواقع الذاكرة الرئيسية او عناوين وحدات الادخال والإخراج من المعالج الى تلك الوحدات

****ناقلة المعطيات Data bus** :مجموعة من الخطوط (8 خطوط في المعالج 8088 و 16 خط في المعالج 8086) المستخدمة لنقل البيانات من والى المعالج

****النموذج البرمجي للمعالج Processor software model** :مجموعة المسجلات في المعالج التي يستطيع المبرمج استخدامها في برامجه المكتوبة بلغة أسمبلي .

****وحدة التنفيذ Execution unit** :الوحدة التي تتولى مهمة تنفيذ التعليمات في المعالج وتحتوي وحدة الحساب والمنطق ومجموعة من المسجلات

****وحدة المتوسط :Bus interface unit** : الوحدة التي تتولى مهمة تبادل البيانات والتعليمات بين المعالج من جهة وباقي وحدات الحاسوب من جهة اخرى

الوحدة الرابعة : مقدمة إلى لغة الاسمبلي

*تمثل اللغات الطبيعية وسيلة مهمة للتعامل والاتصال بين افراد البشر فمن خلالها يتم نقل الحضارات والتعبير عن الافكار وكذلك فان لغات البرمجة تمثل ايضا وسيلة الاتصال بين الانسان وجهاز الحاسوب .

*ولقد واكبت لغات البرمجة التطور النوعي الذي حصل في تكنولوجيا الحواسب حيث مرت في مراحل مختلفة ان الفرق الاساسي بين اللغات والطبيعة ولغات البرمجة هو ان الاخيرة يمكن وصفها بشكل دقيق جدا لا يدع احتمالا لتعدد التفسيرات لنفس العبارة كما هو الحال في اللغات الطبيعية ان ميزة لغات البرمجة هذه قد وفرت الوسائل اللازمة لتعليم جهاز الحاسوب هذه اللغات والقدرة على استخدامها لوصف الحلول لمختلف المسائل في معظم مجالات الحياة

تصنف لغات البرمجة الى اربع انواع هي

1. لغة الالة Machine Language
2. لغة التجميع الاسمبلي Assembly Language
3. اللغات عالية المستوى High Level Language HLL
4. اللغات عالية المستوى جدا Very High Level Language VHLL

*لقد كانت لغة الالة هي اللغة الوحيدة التي استخدمت في حواسيب الجيل الاول كوسيلة اتصال بين المبرمج الحاسوب فالبرامج المكتوبة في بلغة الالة تتكون من سلسة من الاصفار والواحدات وذلك لان الحواسيب تبنى من دارات الكترونية قادرة على التعرف غلى الاصفار والواحدات

* لهذا فان عملية البرمجة لحل أي مسالة في هذه اللغات تتطلب من المبرمج ما يلي :

1. تمثيل جميع الارقام اللازمة لحل المسألة بواسطة الاصفار والواحدات
2. تمثيل تعليمة البرنامج بواسطة الاصفار الواحدات ايضا
3. ادارة مواقع الذاكرة وتخصيص مواقع محددوة لجميع الارقام والتعليمات
4. ادخال الاصفار والواحدات الى مواقع الذاكرة المحددة
5. اصدار امر تنفيذ البرنامج

*تمتاز لغة الالة بالخصائص التالية

1. صعوبة عملية البرمجة
2. صعوبة اكتشاف وتصحيح الاخطاء
3. تتطلب من المبرمجج معرفة دقيقة بجميع مكونات الحاسوب
4. لا تمتلك البرامج المكتوبة في لغات الالة امكانية نقلها الى حواسيب اخرى أي ان هذه اللغات مرتبطة بالحواسيب المعدة لها
5. تمتاز البرامج المكتوبة بلغة الالة بفعالية عالية من حيث سرعة التنفيذ وحجم الذاكرة اللازم لتخزينها
6. تمتاز البرامج المكتوبة بلغة الالة بجاهزيتها للتنفيذ أي انها لا تحتاج بعد ادخالها الى الحاسوب لمراحل معالجة مسبقة

*ان هذه الخصائص السلبية التي امتازت بها لغات الالة قد دعت المختصين الى البحث عن لغات اخرى لا تمثل هذه العيوب وسرعان ما تم تطوير اللغات الرمزية التي وفرت امكانية استخدام الحروف الابجدية والارقام العشرية والرموز الخاصة بالإضافة الى بعض الاختصارات الماخوذة من اللغات الانسانية والانجليزية ومن الامثلة على هذه اللغات لغة اسمبلي .

*تتكون البرامج المكتوبة بلغة اسمبلي من ارقام وحروف ورموز بالإضافة الى بعض الاختصارات بحيث تعبر هذه العناصر بشكل او باخر عن تعليمات لغة الالة وتجد الاشارة الى ان كل تعليمة في لغة اسمبلي يقابلها تعليمة في لغة الالة

**تمتاز لغة اسمبلي بالخصائص الاتية

1. عملية البرمجة واكتشاف وتصحيح الأخطاء أصبحت أسهل .
2. ما زالت هذه اللغة مرتبطة بالحواسيب المعدة لها.
3. - تمتاز برامج هذه اللغة بفعالية عالية .
4. -تمتاز برامجها بعدم جاهزيتها للتنفيذ الفوري بسبب مرورها على مترجم الأسمبلر .تستخدم في برمجة التطبيقات التي تتطلب سرعة فائقة مثل تطبيقات الزمن الحقيقي، وبرامج التحكم في حركات الأنظمة الربوتية (الإنسان الآلي)، كذلك في أنظمة الحواسيب ذات الذاكرة المحدودة جدا

**أما بالنسبة للغات البرمجة عالية المستوى فتتكون برامجها من كلمات وتعابير من اللغات الإنسانية بالإضافة إلى التعابير الرياضية.

*تمتاز لغات البرمجة عالية المستوى بالخصائص التالية:

1. سهولة كتابة البرامج واكتشاف وتصحيح الأخطاء .
2. تمتاز برامجها بإمكانية نقلها بين الحواسيب المختلفة
3. بلغات الأسمبلي أو لغات الآلة - تدني فعاليتها قياسا
4. تحتاج برامجها إلى مراحل معالجة إضافية تسبق عملية التنفيذ عبر المترجمات compilers

*ونظرا لمزايا لغات البرمجة عالية المستوى فانها الاكثر استخداما في الوقت الحالي ومن الامثلة على هذه اللغات COBOL,FORTRAN,BASIC,PROLOG,C,2 MODULA-PASCAL.....وغيرها

*يسمى البرنامج المكتوب بإحدى لغات البرمجة عالية المستوى أو لغات الأسمبلي برنامجا مصدريا source program في حين ان البرنامج الناتج عن عملية الترجمة بواسطة المترجمات والاسمبلر يسمى برنامجا هدفيا object program

*ويسمى البرنامج المستخدم لتحويل البرنامج المصدري المكتوب باحدى لغات البرمجة عالية المستوى الى برنامج هدفي بالمترجم اما البرنامج المستخدم لتحويل البرنامج المصدري المكتوب بلغة الاسمبلي الى برنامج هدفي يسمى برنامج الاسمبلر انظر الشكل 1 صفحة 199

* إن استخدام لغات البرمجة عالية المستوى في حل أية مسألة، تتطلب من المبرمج أن يصف خطوات الحل بشكل مفصل بحيث يستطيع الحاسوب فهم كل خطوة.

*أما لغات البرمجة العالية جدا VHLLفيكفي تحديد الأوامر اللازمة للحل التي توضح للحاسوب ماذا يجب أن يفعل، وليس كيف يفعل.

*ومن الامثلة على لغات VHLL

لغات سؤال –جواب المسماه QUESTION-ANSWER(QA)

ولغات توليد التقارير REPORT GENRTATOR LAVGUAGES

*وتجدر الاشارة الى ان لغات البرمجة ذات المستوى العالي جدا تسمى ايضا لغات الجيل الرابع 4G FOURTH GENRATOR LANGUGES

*نفرض انه لدينا برنامج مكتوب بلغة اسمبلي لحاسوب يستخدم المعالج الدقيق 8086\8088

*يبين الششكل 2 صفحة 201 برنامجا لجمع رقمين هما FIRST ,SECوتخزين المجموع في SUM

*بغض النظر عن محتويات البرنامج نتعرف على مراحل التي يمر بها هذا البرنامج باستخدام نظام الاسمبلي علما بان هذه المرحلة شاملة لاي برنامج بلغة الاسمبلي

المرحلة الأولى: إدخال البرنامجProgram Input

*تهدف الى نقل البرنامج إلى الحاسوب وتخزينه على الذاكرة المساعدةولتحقيق هذا الهدف نستخدم احد برامج معالجة النصوص او محرر النصوص .

*وتجدر الإشارة إلى أن الضغط على المفتاح المقابل لأي رمز يؤدي إلى إدخال كود الأسكي المقابل لذلك الرمز إلى البايت التالي في الذاكرة الرئيسية .

*وبعد الانتهاء من عملية إدخال البرنامج تحتوي الذاكرة الرئيسية نسخة من البرنامج ممثلة في كود الأسكي

*ونظراً لكون الذاكرة الرئيسية مؤقتة لذا يلزم نقل البرنامج إلى الذاكرة المساعدة(انظر شكل3 ص(203) وهنا ينصح عند دخول البرنامج عدم تأجيل نقل البرنامج إلى الذاكرة المساعدة لحين الانتهاء من إدخال جميع رموز البرنامج ،بل ينصح بإجراء عملية النقل هذه بشكل دوري بعد الانتهاء من إدخال كل سطرين او ثلاثة سطور، وذلك لأنه في حالة عدم الأخذ بهذه النصيحة يتطلب إعادة إدخال البرنامج كاملاً عند انقطاع التيار الكهربائي أو حدوث خلل في جهاز الحاسوب قبل إجراء عملية النقل إلى الذاكرة المساعدة

*ان المعلومات تخزن في الذاكرة المساعدة في ملفات،لذا فإن البرنامج المكتوب بلغة الأسمبلي يجب أن يخزن أيضا في ملف

*يتم اختيار أسماء الملفات التي تحتوي برامج لغة الأسمبلي حسب قواعد نظام التشغيل (DOS)

*بحيث يحتوي الشق الثاني دائما الرموز (ASM)

*على سبيل المثال نفرض أن البرنامج الذي يتم إدخاله من قبل قد خزن في ملف اسمه(SOURCI.ASM)

***المرحلة الثانية:** ترجمة البرنامج (program Compilation) ****تهدف مرحلة الترجمة إلى:**

-اكتشاف وتصحيح الأخطاء المطبعية والقواعدية،أي الاخطاء الناجمة عن مخالفة قواعد اللغة

-تحويل البرنامج المصدري إلى برنامج هدفي

*تنفذ مرحلة الترجمة بواسطة مترجم الاسمبلر الذي يحمل اسم (MASM)

*كما هو مبين في الشكل(3)ص(203) فإن الاسمبلر لا يكون البرنامج الهدفي إلا في حالة خلو البرنامج المصدري من كافة الأخطاء المطبعية والقواعدية

*وفي حالة وجود مثل هذه الأخطاء يصدر الأسمبلر قائمة تتضمن رقم الخطأ،ورقم السطر الذي يحتوي الخطأ،ووصف قصير للخطأ،

*ان وجود اخطاء قواعدية يتطلب من المبرمج الرجوع إلى مرحلة إدخال البرنامج لإجراء التصحيحات والتعديلات اللازمة وتستمر عملية الانتقال بين مرحلتي الإدخال والترجمة لحين الحصول على البرنامج الهدفي

*ان تنقل البرنامج إلى الذاكرة المساعدة بعد إجراء اي تغيير عليه ،فالتغيير الذي تجريه عن طريق لوحة المفاتيح يتم على البرنامج المخزن في الذاكرة الرئيسة في حين أن الأسمبلر يعالج البرنامج المخزن في الذاكرة المساعدة.

***المرحلة الثالثة:الربط والتحرير (program Linking)**

*أن البرنامج الهدفي الناتج من مرحلة الترجمة لا يكون جاهزاً للتنفيذ بسبب احتوائه على رموز لا يستطيع برنامج الأسمبلر ترجمتها إلى لغة الآلة في مرحلة الترجمة.

*فمثلاً يمكن أن يحتوي البرنامج على استدعاء لبرامج مكتبية أو برامج أخرى للمستخدم تمت ترجمتها سابقاً بشكل منفصل عن البرنامج الحالي

*أن ترجمة مثل هذه الاستدعاءات تتم في مرحلة الربط والتحرير وتتطلب البحث عن هذه البرامج ودمجها مع البرنامج الحالي لتكوين برنامج تنفيذي واحد

*تنفذ مرحلة الربط والتحرير بواسطة برنامج الربط والتحرير Linker ، الذي يستقبل برنامجا هدفيا واحدا (أو مجموعة برامج هدفية)مخزن على الذاكرة المساعدة ويحوله إلى برنامج تنفيذي

*يلاحظ ان اسم البرنامج الهدفي يتضمن عبارة . (Obj)في حين أن اسم البرنامج التنفيذي يتضمن عبارة .(EXE)

*يتكون البرنامج التنفيذي فقط في حالة خلو البرنامج الهدفي(البرامج الهدفية) من الاخطاء

*وفي حالة وجود اخطاء في عملية الربط والتحرير يجب الرجوع الى مرحلة الادخال لاجراء التعديلات اللازمة ،واجراء عملية الترجمة،ومن ثم تنفيذ عملية الربط والتحرير .

*وتستمر عملية الانتقال بين مراحل الادخال والترجمة والربط والتحرير لحين تكوين البرنامج التنفيذي

*وتجدر الإشارة إلى أن الضغط على المفتاح المقابل لأي رمز يؤدي إلى إدخال كود الأسكي المقابل لذلك الرمز إلى البايت التالي في الذاكرة الرئيسية.

*وبعد الانتهاء من عملية إدخال البرنامج تحتوي الذاكرة الرئيسية نسخة من البرنامج ممثلة في كود الأسكي

*ونظراً لكون الذاكرة الرئيسية مؤقتة لذا يلزم نقل البرنامج إلى الذاكرة المساعدة(انظر شكل3 ص(203) وهنا ينصح عند دخول البرنامج عدم تأجيل نقل البرنامج إلى الذاكرة المساعدة لحين الانتهاء من إدخال جميع رموز البرنامج ،بل ينصح بإجراء عملية النقل هذه بشكل دوري بعد الانتهاء من إدخال كل سطرين او ثلاثة سطور، وذلك لأنه في حالة عدم الأخذ بهذه النصيحة يتطلب إعادة إدخال البرنامج كاملاً عند انقطاع التيار الكهربائي أو حدوث خلل في جهاز الحاسوب قبل إجراء عملية النقل إلى الذاكرة المساعدة

*ان المعلومات تخزن في الذاكرة المساعدة في ملفات،لذا فإن البرنامج المكتوب بلغة الأسمبلي يجب أن يخزن أيضا في ملف

*يتم اختيار أسماء الملفات التي تحتوي برامج لغة الأسمبلي حسب قواعد نظام التشغيل(DOS)

*بحيث يحتوي الشق الثاني دائما الرموز(ASM)

*على سبيل المثال نفرض أن البرنامج الذي يتم إدخاله من قبل قد خزن في ملف اسمه (SOURCI.ASM)

***المرحلة الرابعة:تنفيذ البرنامج (program Execution)**

*ان البرنامج الناتج بعد مرحلة الربط والتحرير يصبح جاهزاً للتنفيذ. تهدف مرحلة التنفيذ إلى الحصول على النتائج المطلوبة

***يوجد طريقتان لتنفيذ البرامج وهما :**

-الطريقة المباشرة من خلال نظام (DOS)

-استخدام برنامج "مكتشف الأخطاء"(Debug)

*تستخدم الطريقة المباشرة عادة في الحالات التي يتضمن فيها البرنامج وسائل لادخال المعطيات واخراج النتائج الى وحدات الاخراج المناسبة

*تستخدم الطريقة الثانية في الحالات التي لا تتضمن البرنامج فيها مثل هذه الوسائل أو الحالات التي تؤدي إلى نتائج خاطئة أو لا تعطي أي نتائج بسبب وجود اخطاء منطقية في البرنامج نفسه

*وتبعا للطريقة المباشرة يتم تنفيذ البرنامج بكتابة اسم البرنامج فقط(SOURCEI) بعد الحصول على إشارة جاهزية الحاسوب لاستقبال الامر التالي. وهنا يبدا الحاسوب فوراً بتنفيذ البرنامج طبقاً لتسلسل التعليمات المكونة له

*وتبعاً للطريقة الثانية يتم تنفيذ البرنامج من خلال استدعاء برنامج (Debug)الذي بدوره يطلب من المستخدم طباعة الأوامر عن طريق لوحة المفاتيح

***يستخدم برنامج (Debug) لتنفيذ العمليات الآتية :**

-تنفيذ البرنامج دفعة واحدة أو تعليمية بعد تعليمه

-عرض محتويات المسجلات ومواقع الذاكرة الرئيسة حب الطلب

-تعديل محتويات المسجلات ومواقع الذاكرة الرئيسة حسب الحاجة

-تحويل التعليمات المصدرية(Source Instructions) إلى لغة الآلة وبالعكس

-البحث في الذاكرة الرئيسة،مليء مجموعة مواقع بقيمة معينة،...غيرها

****تعليمات استخدام نظام الأسمبلي**

* تتم عملية ادخال نص البرنامج المصدري بوساطة أحد معالجات النصوص أو محررات النصوص شائعة الاستخدام شريطة ان يتم تمثيل رموز البرنامج في شيفرة الأسكي .

*هذا ويمكن استخدام محرر النصوص المتوفر في نظام الأسمبلي نفسه مثل (M.EXE أو PWB.EXE).

*يعرف محرر النصوص (معالج النصوص) بأنه برنامج مخصص لمعالجة النصوص من خلال تنفيذ عمليات على النص نفسه.

***يتضمن اي معالج نصوص العمليات الاساسية الآتية :**

-خلق أو توليد النص Text creation

-عرض النص Text viewing

-تعديل وتحديث النص Text editing

-تخزين وطباعة النصSaving printing&

*تنفذ هذه العمليات عادة بواسطة تعليمات خاصة. يختلف شكل وصيغة هذه التعليمات من برنامج إلى آخر .

*فالقرص المرافق لهذه الوحدة يحتوي برنامج تحرير النصوص المسمى (SK).

* يحتوي ملحق استخدام نظام الأسمبلي شرحا للتعليمات الأساسية لجميع مكونات نظام الأسمبلي، لذا عليك عزيزي الدارس الرجوع إلى برنامج (SK) والتدرب على إستخدام تعليماته الأساسية من خلال اعداد البرنامج السابق وتخزينه في ملف إسمه (SOURCE1.ASM).

*يستخدم برنامج الأسمبر (MASM) لتحويل البرنامج المصدري إلى برنامج هدفي بواسطة الأمر التالي : A> MASM

*وهنا يبدأ الحاسوب تنفيذ برنامج الأسمبلي الذي يطلب فوراً : اسم البرنامج المصدري، واسم البرنامج الهدفي، واسم ملف تقرير الأسمبلر، واسم ملف مرجعيات الأسماء على النحو التالي:

Source filename[.ASM]:SOURCE1

Object filename (SOURCE1.OBJ]: SOURCE1

Source listing [NULLST]: SOURCE1

Cross-reference [NUL.CRF): SOURCE1

*حيث ان الاسماء إلى يمين النقطتين الرأسيتين يتم ادخالها من قبل المبرمج، ويمكنك عزيزي الدارس الرجوع إلى ملحق استخدام نظام الأسمبلي للتعرف أكثر على إستخدام هذا الأمر وتأثيره على محتويات تقرير الأسمبلر .

*وبعد إكمال عملية الترجمة يلزم تحويل البرنامج الهدفي الناتج إلى برنامج تنفيذي بوساطة برنامج الربط والتحريك الذي يستدعى من خلال الامر : A> LINK

* وهنا يبدأ الحاسوب تنفيذ برنامج الربط والتحرير الذي يطلب فورا: اسم البرنامج

الهدفي (أو اسماء البرامج الهدفية)، وإسم البرنامج التنفيذي، واسم الملف الذي يحتوي توزيع القطاعات في الذاكرة الرئيسة، واخيرا اسم الملف الذي يحتوي البرامج المكتبية كما هو موضح على النحو التالي

Object Modules [.OBJ]: SOURCE1

Run File [SOURCEL.EXE): SOURCE1

List File (NUL.MAP]: SOURCE1

:Libraries [.LIB]

*وبعد الانتهاء من عملية التحرير والربط يتكون البرنامج التنفيذي(SOURCEI.EXE يتم تنفيذ هذا البرنامج بواسطة مكتشف الاخطاء ((Debug الذي يستدعى على النحو التالي: A> Debug

*عند تنفيذ هذا الأمر يصدر البرنامج إشارة "-" التي تعني جاهزية مكتشف الاخطاء لاستقبال الأمر التالي. تدريب 2و3 و1 صفحة 208

**انواع الأخطاء في برامج لغة الأسمبلي:

*من المعروف لدى المختصين في علوم الحاسوب والبرمجة أنه من الصعب كتابة برنامج لحل أية مسألة والحصول على نتائج صحيحة وبالشكل المطلوب من خلال عملية التنفيذ الأولى .

* وسبب ذلك يعود إلى عدم شمولية الحل لجميع أشكال وأنواع البيانات أو بسبب صعوبة تحديد خصائص المسائل وصعوبة التعبير عنها في لغات البرمجة المحدودة الامكانيات . وهذا ينطبق أيضا على البرمجة بلغة الأسمبلي ، فالبرامج تحتوي عادة على أخطاء مطبعية او منطقية .

تصنّف الأخطاء في برامج الأسمبلي إلى الأنواع الآتية :

- اخطاء مرحلة الترجمة Assembly - time errors

-اخطاء مرحلة الربط والتحرير Link - time error

-اخطاء مرحلة التنفيذ Run - timeerrors

*تكتشف اخطاء مرحلة الترجمة بواسطة برنامج الأسمبلر MASN ، وتنشأ هذه الاخطاء بسبب مخالفة قواعد اللغة ، لذا يسميها بعض المختصين بالأخطاء القواعدية (Syntax erros)

*عندما يكتشف الأسمبلر خطأ من هذا النوع يبلغ المبرمج عن ذلك من خلال اصدار رسالة (errors message) تتضمن رقم السطر ، ورقم الخطأ ، ووصف لا يزيد عن سطرين عن الخطأ الحاصل .

*عندما ينهي الأسمبلر عملية الترجمة يصدر تقرير يتضمن قائمة الاخطاء القواعدية التي اكتشفت في البرنامج . وعلى المبرمج في هذه الحالة أن يصحح هذه الأخطاء ويعيد عملية الترجمة .

*وفي حالة خلو البرنامج من الاخطاء القواعدية يبلغ الأسمبلر المبرمج عن ذلك باصدار رسالة يحدد فيها أن عدد الاخطاء يساوي صفر.

و حسب تأثير الأخطاء على عملية الترجمة تصنف الاخطاء القواعدية إلى مستويين اثنين هما :

-اخطاء تحذيرية Warnings : تتسبب عادة عن بعض الأخطاء المطبعية التي يحاول الأسمبلر تصحيحها لاغراض استكمال عملية الترجمة والوصول إلى نهاية البرنامج . إن وجود مثل هذه الأخطاء لا يمنع تكوين البرنامج الهدفي ، غير انه قد يؤدي إلى الحصول على نتائج خاطئة . ومن الأمثلة على الاخطاء التحذيرية : عدم تحديد نقطة بداية تنفيذ البرنامج في جملة (END) .

- اخطاء جدية Severe errors : تتسبب عن مخالفة قواعد اللغة مثل زيادة عدد المعاملات (Operands) عن الحد المطلوب في احدى تعليمات لغة الأسمبلي ، وعادة يقوم الأسمبلر بشطب الجملة أو جزء من الجملة التي تحتوي هذا الخطأ ، وهذا بدوره يمكن أن يسبب اخطاء أخرى وبالتالي يؤدي إلى عدم تكوين البرنامج الهدفي.

** تكتشف أخطاء مرحلة الربط والتحرير بواسطة برنامج تحرير الوصل . LINKER

*لايستطيع برنامج الأسمبلر اكتشاف مثل هذه الأخطاء بسبب عدم توفر المعلومات اللازمة لاكتشافها في مرحلة الترجمة .

*من الأمثلة الشائعة عن على مرحلة التحرير كتابة اسم البرنامج المكتبي بشكل خاطئ ، فمثلا عند استدعاء برنامج حساب جيب الزاوية (SIN) تستخدم الاسم (SINE) .

* فالأسمبلر لا يستطيع الوصول إلى ملف الاقترانات المكتبية ويترك هذا الأمر لبرنامج التحرير الذي يبحث بدوره عن اقتران باسم (SINE) ، ونظرا لعدم وجود اقتران يحمل هذا الاسم يصدر برنامج التحرير رسالة يبلغ فيها عن عدم مقدرته تحديد عنوان الاقتران المسمى (SINE) والرسالة هي(Unresolved external reference)

*ان وجود مثل هذه الأخطاء لا يمنع تكوين البرنامج التنفيذي ، وفي حالة الانتقال الى مرحلة التنفيذ فإن ذلك يؤدي إلى حصول اخطاء في النتائج أو عدم الحصول على أية نتائج ، لذلك ينصح عادة باجراء التصحيحات اللازمة قبل الانتقال إلى تنفيذ البرنامج الذي يحتوي مثل هذه الأخطاء .

*وتجدر الإشارة إلى أن الاخطاء القواعدية واخطاء مرحلة الربط والتحرير يتم اكتشافها بطريقة أتوماتيكية بوساطة برامج الأسمبلر ومحرر الوصل ولا يوجد صعوبات في تصحيحها .

* أما بالنسبة إلى اخطاء مرحلة التنفيذ فالأمر يختلف . هنا اخطاء يتم اكتشافها بطريقة اتوماتيكية أثناء مرحلة التنفيذ ، مثل القسمة على صفر ، ويتم الإبلاغ عنها .

*وهناك اخطاء من نوع آخر تسبب تعليقة في البرنامج، مثل الدوران اللانهائي هذا بالاضافة إلى الاخطاء البرمجية التي تؤدي إلى نتائج خاطئة .

*ان اخطاء مرحلة التنفيذ تعتبر من أخطر انواع الاخطاء واكثرها تعقيدا من من حيث صعوبة اكتشافها وتأثيرها المباشر على النتائج التي تستخدم في اتخاذ القرارات .

*وفي كثير من المراجع تسمى اخطاء مرحلة التنفيذ بالأخطاء المنطقية (Logical errors) لارتباطها المباشر بمنطق البرنامج .

**** عناصر ومكونات تقرير الأسمبلر Assembly Listing**

** يتكون تقرير الأسمبلر من البرنامج المصدري ، والبرنامج الهدفي ، ومجموعة من الجداول التي تحتوي معلومات حول عناصر البرنامج المختلفة مثل أسماء القطاعات وحجمها ، أسماء الماكرو وحجمها .... وغيرها كما هو مبين في ملف التقرير (Listing File) الموضح في شكل (4) صفحة 214

**يبدأ تقرير الأسمبلر بتقديم معلومات حول الأسمبلر نفسه وتاريخ إجراء عملية الترجمة ، ويلي ذلك البرنامج المصدري (من اليمين) والبرنامج الهدفي (إلى اليسار) .

* تلاحظ ع أن هناك جملا في البرنامج المصدري لا يقابلها أي كود هدفي (وهي معظم التوجيهات) . أما تعليمات البرنامج المصدري فتقابل كل منها إلى اليسار الكود الهدفي المكافئ لها .

*** وبهذا يتكون السطر من ثلاثة عناصر هي :**

1 [الإزاحة :

فالإزاحة هي مقدار المسافة بالبايت بين كل جملة وبداية القطاع الذي يحتوي تلك الجملة. أما الكود فهو عبارة عن ارقام في النظام السادس عشري تمثل العملية المتضمنة في التعليمة وعناوين المعاملات المشاركة في العملية نفسها .

2 [الكود : ويمكن للكود أن يحتوي بعض الرموز مثل (R) أو (E) التي تعني أن برنامج الأسمبلر لم يستطع تحويل عنصر البرنامج المقابل إلى رقم بسبب عدم توفر المعلومات اللازمة لعملية التحويل في مرحلة الترجمة . وهذه الرموز يتم فهمها من قبل برنامج تحرير الوصل والربط (Linker) الذي يقوم بتحويلها إلى قيم عددية . وإلى يمين الكود تقع الجملة المصدرية المكتوبة بلغة الأسمبلي .

3 [الجملة المصدرية

وفي حالة حدوث اخطاء في ترجمة أية جملة تظهر فورا بعد الجملة الخاطئة رسالة الخطأ التي تحتوي رقم السطر ، ورقم الخطأ ، ووصف للخطأ بما لا نريد عن سطر أو سطرين كما هو موضح في المثال الآتي :

E8 001 CR CALL DOIT test . s (46) : error A 2071 : Forward needs override or FAR 0012 71

. فالرقم (46) هو رقم الجملة في البرنامج المصدري ، الرقم (2071A) فهو رقم الخط أما الرقم (71) فهو رقم الجملة في ملف التقرير .

ويلي البرامج المصدري والهدفي دائما مجموعة من الجداول تظهر في ملف تقرير الأسمبل حسب حاجة المبرمج ، حيث يستطيع المبرمج إظهار بعض الجداول أو جميعها بواسطة تحديد الخيارات المناسبة ، ولمزيد من المعلومات عليك الرجوع إلى ملحق استخدام نظام الأسمبلي . نتعرف الآن عزيزي الدارس على بعض هذه الجداول .

* يحتوي جدول الماكرو (Macro Table) على أسماء وأحجام جميع الماكرو المستدعاة أو المعرفة في البرنامج المصدري . يحدد حجم الماكرو بعدد السطور المكونة له . تظهر اسماء الماكرو في الجدول مرتبة ابجديا.

*يحتوي الجدول التالي (جدول القطاعات والمجموعات (Segment and Group table على أسماء المجموعات والقطاعات وخصائصها ابتداءا بالمجموعات ويلي ذلك القطاعات . يحتوي جدول الرموز (Symbol Table) جميع الأسماء الواردة في البرنامج المصدري باستثناء اسماء الماكرو والمجموعات والقطاعات .

*يحتوي جدول الرموز على خصائص الأسم مثل : نوع الأسم وقيمته ومعلومات أخرى . فمثلا هل الأسم يمثل اسم لأجراء قريب أو بعيد أم اسم مطلق لوسام . ويلي هذه الجداول عادة معلومات إحصائية حول البرنامج مثل : عدد سطور البرنامج المصدري ، وعدد الرموز المستخدمة في البرنامج ، وعدد الأخطاء من الأنواع التحذيرية أو الجدية إن وجدت .

**** عناصر لغة الأسمبلي**

يتكون البرنامج المصدري المكتوب بلغة الاسمبلي من قطاعات تتكون بدورها من جمل. والجملة تتكون من كلمات وأعداد مبنية بإستخدام الحروف والأرقام والرموز لذا قبل التعرف على كتابة البرامج يلزمنا التعرف على العناصر التي تكون البرنامج .

و نبدا بطاقم الرموز (Character Set) . يعرف طاقم الرموز(الأبجدية) بأنه مجموعة الرموز التى باستخدامها في بناء البرامج المصدرية. **يحتوي طاقم رموز لغة الأسمبلي أنواع الرموز التالية :**

الحروف الأبجدية الأنجليزية وتضم الحروف الكبيرة)A-Z والحروف الصغيرة(a-z) ، عدد هذه الرموز يساوي 52.

-الأرقام العربية وتشمل الأرقام (0-9) وعدها (10).

-الرموزالخاصةوعددها)22(وتشمل الرموز التالية :

% v ..: # =\.1 O[I)< > { }

1 الثوابت constants

تصنف الثوابت إلى نوعين هما: الثوابت العد دية والثوابت الرمزية.

يعرّف الثابت الرمزي بأنه رمز واحد أو سلسلة من الرموز المكتوبة بين حاصرتين اثنتين على النحو التالي : 'Characters.

يعرف طول الثابت الرمزي بأنه عدد الرموز المكونة له ولا يشمل الحاصرتين .

*تمثل الرموز المكونة للثابت في نظام الأسكي ASCI I بحيث يخصص بايت واحد لكل رمز.

*وبما أن الحاصرة تستخدم لتحديد بداية ونهاية الثابت الرمزي فأن بناء ثابت يحتوي الحاصرة كجزء منه يتطلب كتابة الحاصرة مرتين كما في الثابت التالى : 'an ."Ma. وهذا بالطبع لا يزيد طول الثابت الرمزي (طول الثابت = 5 رموز). **من الأمثلة على الثوابت الرمزية الصحيحة ما يلي:** "AMMAN' , 'Lotus 123', Jordan, Jerusalem'

*تستخدم الثوابت الصحيحة لتمثيل الأعداد الصحيحه وتكتب في نظام العدد الثنائي، أو الثماني، أو العشري، أو النظام السداعشري .

*يكتب الثابت الثنائي كسلسلة من الأصفار والواحدات متبوعة بالحرف (B) أو (b) كماموضح في الأمثلة : B(101، 1(( ) .1010010b

يحتوي الثابت الثماني الأرقام (0-7)ويكون متبوعا بالحرف (Q) أو (O)كما في الامثلة : (6305Q)، (37Q).

أما الثابت العشري فيحتوي الأرقام (0-9) ويكون متبوعا بالحرف (D)أو (d)، ويمكن للثابت العشرى أن لا يحتوي الرمز (D) كما في الأمثلة : (101 1994)، (D) .

يكتب الثابت السداعشري كسلسلة من الأرقام (A-F و 0-9) متبوعة بالحرف (H) أو (h) كما هو موضح في الامثلة : (CH3، A79H0). ويلاحظ أن هذا الثابت السداعشري يجب أن يسبق بالرقم صفر في حالة ابتدائه بأحد الحروف (A-F) لتمييزه عن أسماء المتغيرات.

2.الأسماء Names

*تستخدم الاسماء في لغة الأسمبلي بشكل واسع وتعبر عن : اسماء المتغيرات ، وأسماء القطاعات، وأسماء الإجراءات، وأسماء الوسامات، وأسماء الماكرو.

****يختارالمبرمج أسماء ل لتعبير عن هذه العناصر مع مراعاة ما يلي :**

-يصل طول الأسم لغاية (31) رمز .

-يتكون الأسم من الحروف، والأرقام، وبعض الرموز الخاصة مثل : علامةالاستفهام؟، وإشارة الدولار $، وإشارة @، وإشارة المد ،%.

-الرمز الأول في الأسم يجب أن يكون حرفا أو أحد الرموز الخاصة المذكورة أعلاه أو النقطة

-لا يجوز اختيار أسماء مطابقة لأسماء الكلمات المحجوزة.

***تعرف الكلمة المحجوزة** بأنها أي اسم يحمل معنى خاص ومحدد مسبقا من قبل الأسمبلر

*من الأمثلة على الكلمات المحجوزة : أسماء المسجلات، وأسماء التعليمات، وأسماء التوجيهات بالإضافة إلى الأسماء الواردة في الجدول ١.

3. ادوات العمليات operators

*تستخدم الثوابت وبعض الأسماء لبناء التعابير بواسطة أدوات العمليات.

تصنف العمليات التي يمكن أن تستخدم في تعابير لغة الأسمبلي إلى 5 مجموعات هي :

1-Arithmetic Operators (العملياات الحسابيه)

2-Logical Operators (العمليات المنطقيه)

3-Relational Operators (العمليات النسبيه)

4 - عمليات ارجاع القيمه Value- Returning Operators

5-عمليات الصفات Attribute Operators

جدول (١) : الأسماء المحجوزة في لغة الاسمبلي صفحة 220

*تستخدم العمليات الحسابية في بناء التعابير الحسابية، أي التعابير التي ت تكون من ثوابت وأسماء تعبّر عن قيم عددية وتكون نتيجتها قيمة عددية أيضا وتشمل العمليات الحسابيه ما يلي صفحة 221

*تستخدم العمليات المنطقيه في بناء التعابير المنطقيه التي تتكون من ثوابت واسماء تعبر عن قيم عددية.

وتشمل العمليات الآتية صفحة 222

**تستخدم العمليات النسبية لمقارنة قيمتين لتحديد العلاقة بينهما، وتكون صيغة العمليات النسبية استفهامية م ما يؤدي إلى الحصول على إجابة بصيغة صح أو خطأ. تشمل هذه العمليات : GE, LE,GT,LT,NE,EQ. فمثلا تكون نتيجة العملية (XLT20) تساوي 0 إذا كانت (*) أكبر أو تساوي (20)، وتكون قيمة العملية تساوي1 إذا كانت قيمة Xاقل من20.

اما نتيجة التعبير (OFFFFH AND 7) OR (0 AND 8).فهي (7)

*** تستخدم عمليات ارجاع القيم لتحديد بعض المعلومات التي تخص المتغيرات او الوسامات المستخدمة في البرامج وتضم هذه المجموعة العمليات الآتية :**

عمليات-SEG& OFFSET.

*تستخدم هذه العمليات لإرجاع مقدار الإزاحة أو الأساس ل لمتغير أو الوسام كما في **الأمثلة التالية :

SEG TI : تحسب قيمة الأساس ل لمتغير T1.

OF FSET: TIتحسب قيمة الإزاحة ل لمتغير T1 .

عملية TYPE

تستخدم هذه العملية لتحديد طول المتغير أو نوع الوسام. فمثلا إذا كان الأسم)T2(يعبر عن متغير طوله بايت واحد فإن العملية)TYPE T2(يؤدي إلى ارجاع القيمة)1(. أما إذا كان طول المتغير)T2(يساوي)2(بايت، فإن العملية T2( )TYPEترجع)2(.

بالنسبة للوسام فهناك نوعان : الوسام القريب Near Label والوسام البعيد FarLabel.

فإذا كان الأسم T3 يمثل وساما فإن العملية T3( )TYPEترجع القيمة)-1(إذا كان الوسام قريباً أو القيمة)-2(إذا كان الوسام بعيدا .

(LENGTH) عملية

*تستخدم هذه العملية لتحديد عد د وحدات الذاكرة)عد د البايتات أو عدد الكلمات التي يحتلها المتغير المعرف بواسطة العملية)dup(فمثلا إذا كان المتغير)T4(معرفا حسب الجملة التالية : T4 DW 100 DUP (3)

فالعملية T4( )LENGTHتؤدي إلى الحصول على القيمة 100 . أن استعمال عمليةLENGTH مع متغير غير معرّف بواسطة)DUP(يؤدي إلى ارجاع القيمة 1.

SIZE عملية

*تستخدم لتحديد البايتات التي يحتلها المتغير المعرف بواسطة DUP، فالعملية تؤدي (Size4) الي الحصول على القيمة 200. فنتيجة عملية SIZEهي حاصل ضرب نتيجة عملية)LENGTH(ونتيجة عملية)TYPE.

تستخدم عمليات الصفات لتخطى الصفات الاعتيادية المفهومة ضمنا لبعض عناصرالبرنامج مثل الوسام أو نوع القطاع المستخدم. **تضم هذه المجموعة العمليات الآتية:**

-عملية ال PTR

تستخدم هذه العملية لتخطى صفة المعاملة)بايت أو كلمة(أو صفة المسافة ل لمعاملة)قريب أو بعيد(. فمثلا يمكنك التعامل مع أي بايت في جدول يتكون من كلمات كما في المثال التالي:

() WORD TABLE DW 100 DUP

FIRST

EQU BYTE PTR WORD TABLE

حيث تستخدم الجملة الأولى لتعريف جدول يتكون من 100 كلمة، وتستخدم الجملة الثانية لتخزين قيمة البايت الأول في المتغير FIRST

CS: DS: SS: ES: عملية تخطي نوع القطاع

لأغراض الوصول إلى أي بايت يلزم حساب العنوان الطبيعي له. وهذا يتطلب معرفة مقدار الإزاحة وقيمة الأساس التي تعتمد على نوع القطاع الذي يحتوى البابت المطلوب. وفي حالة عدم تحديد نوع القطاع يتم استخدام القطاع الضمني. وفي حالة وجود البايت في قطاع يختلف عن القطاع الضمني يلزم تحديد ذلك القطاع مباشرة بواسطة عملية تخطي نوع القطاع. فمثلا تستخدم التعليمة MOV AL, [BP] البايت المشار اليه بواسطة المسجل BP والموجود ضمنيا في قطاع المكدس، فإذا أردنا استخدام البايت المشار اليه بواسطة المسجل BP والموجود في قطاع المعطيات فإن التعليمة MOVAL, DS: [[BP]] تؤدي المطلوب .

SHORT عملية

تستخدم هذه العملية لتحديد نوع المعاملة في بعض التعليمات بحيث يكون مدى هذه المعاملة محصورا بين)-128، +127(بايت. فمثلا تستخدم التعليمة JMPSHORT L1 لجعل المسافة بين هذه التعليمة والتعليمة التي تحمل الوسام L1 تقع في المجال)-128 ، +107(بايت. في حين أن التعليمة JMP L1 تجعل المسافة بين التعليمة والتعليمة التي تحمل الوسام L1 تقع في المجال)-32768 ، +32767(.

HIGH LOW عملية

تستخدم هذه العمليات لاستنباط البايت الأيمن)LOW(أو البايت الأيسر)HIGH(من قيمة طولها كلمة واحدة)2 بايت(كما في المثال التالي:

MOV AH, LOW WORD 1

WORD 1

DW

CL, HIGH WORD_1

حيث أن الجملة الأولى تستخدم لتخزين القيمة ABC9H في موقع الذاكرة WORD_1 الذي طوله يساوي 2 بايت. تستخدم جملة MOV الأولي لنقل البايت الي المسجل AH (C9)WORD أما جملة MOV الثانيه فتستخدم لنقل الثابت الايسر في WORD-.CL الي المسجل(AB

عملية THIS

تستخدم هذه العملية لتعريف معاملة طولها بايت أو كلمة واعطائها قيم الإزاحه والأساس للقيمة كما في المثال صفحة 255.

** الصيغة العامة لجملة لغة الاسمبلي

***يتكون البرنامج الصدري في لغة الأسمبلي من قطاعات التي بدورها تتكون من:**

جمل STATEMENTS . تتكون الجملة من اربعة حقول هي:

1. **حقل الوسام أر الأسم:** يستخدم هذا الحفل لتحديد. علامه (وسام) لإحدى جمل البرنامج، يمثل هذا الحفل عنوان البايت الأول في الجملة .

يستخدم هذا الحقل أو بشكل اختياري حسب منطق PROC بشكل اجباري في بعض الحمل مثل جملة البرنامج كما هو الحال في معظم التعليمات. يعتمد طول حقل الوسام على طول الأسم ولكنه لا يزيد عن 31 رمزا، يفصل حقل الوسام عن الحقل التالي بواسطة الفراغات أو النقطتين الرأسيتن :

2. **حقل العملية :** يستخدم حقل العملية لتحديد نوع العملية الممثلة في الجملة . تمثل العملية عادة احدى تعليمات المعالج الدقيق أو أحد أوامر الأسمبلر نفسه . يصل طول هذا الحفل لغاية 20 رمز .

3. **حقل المعاملات :** يستخدم حقل المعاملات لتحديد مواقع تخزين القيم المشاركة في العملية والقيم الناتجة بعد تنفيذ العملية . يختلف عدد المعاملات من جملة إلى أخرى ويتراوح بين صفر و2 في التعليمات وأكثر من ذلك في التوجيهات .

4. **حقل الملاحظات** : يستخدم حقل الملاحظات ، كما هو الحال في لغات البرمجة عالية المستوى مثل لغة بيسك، لزيادة استيعابية البرنامج لدي المستخدمين. تكتب الملاحظات في لغة الأسمبلي بعد حقل المعاملات (أن وجد) أو بشكل مستقل في سطر واحد. تستخدم الفاصلة المنقوطة " ؛ " لتحديد بداية حقل الملاحظات .

*تفصل حقول العملية، والمعاملات ، والملاحظات بعضها عن بعض بواسطة أي عدد من الفراغات، (الاطلاع على شكل ا5 صفحة 226 والذي يبين الصيغة العامة لجملة لغة الأسمبلي) .

**** جمل برنامج الأسمبلي**

*** تصنف جمل برنامج الأسمبلي إلى ثلاثة أنواع هي :**

- التوجيهات Directives

- التعليمات Instructions

- التعليمات المتسعة Macrocalls

*** التوجيهات :** هي عبارة عن أموامر تنفذ من قبل برنامج الأسمبلر في مرحلة الترجمة .

**** استخدامات التوجيهات :-***

• بناء وتحديد قطاعات الذاكرة والإجراءات .

• تعريف الأسماء المستخدمة في البرنامج .

• حجز مواقع في الذاكرة للمعطيات والنتائج .

• التحكم بعملية الترجمة.

• تحديد عدد السطور في صفحات تقرير الأسمبلر .

* إن معظم التوجيهات لا تؤدي إلى تكوين كود هدفي بسبب كونها جمل إخبارية ضرورية في عملية الترجمة .

* تحتوي لغة الأسمبلي للمعالج 8088/8086 على حوالي 60 توجيهة .

* التعليمات : هي أموامر تنفذ من قبل المعالج الدقيق في مرحلة تنفيذ البرنامج ،ونتيجة لتنفيذ التعليمات يتم الحصول على نتائج .

*** تصنف التعليمات حسب نوع العملية التي تنفذها إلى عدة مجموعات :-**

• تعليمات نقل (تحريك) البيانات. • التعليمات الحسابية.

• تعليمات معالجة البتات (الثنائيات) • تعليمات نقل التحكم.

• تعليمات سلاسل الرموز. • تعليمات الإعتراضات. • تعليمات التحكم بالمعالج.

*** طاقم التعليمات :** هو مجموعة التعليمات التي يستطيع المعالج تنفيذها.

* يتكون طاقم تعليمات المعالج 8088/8086 من حوالي 150 تعليمة ، يختلف شكل التعليمة عادة حسب عدد ونوع المعاملات فيها .

* هناك تعليمات لا تحتوي أية معلومات تسمى تعليمات صفرية العنوان .

* هناك تعليمات تحتوي معاملة واحدة تسمى تعليمات أحادية العنوان .

* هناك تعليمات تحتوي معاملتين واحدة تسمى تعليمات ثنائية العنوان .

* تؤدي عملية ترجمة التعليمات بواسطة الأسمبلر إلى توليد كود هدفي لكل تعليمة في البرنامج على عكس التوجيهات.

*** التعليمات المتسعة :** هي عبارة عن جملة واحدة تؤدي إلى توليد مجموعة من الجمل عند استدعائها، وهذا يتطلب من المبرمج تحديد الجمل التي تقابل كل تعليمة متسعة بواسطة تعريف الماكرو .

**** هيكل برنامج لغة الأسمبلي**

* أن برنامج الأسمبلي يتكون من مجموعة من القطاعات لا يقل عدها عن ثلاثة .

* أن هناك اربعة أنواع من القطاعات هي : قطاع الكود، و قطاع المعطيات، وقطاع المكدس، وقطاع الإضافة.

* يمكن لبرنامج الأسمبلي أن يحتل ملف واحدة أو عدة ملفات في الذاكرة المساعدة.

* انظر الشكل رقم (6) صفحة ٢٣٠ الذي يبين النموذج الرئيسي لبرنامج الأسمبلي الذي يحتل ملف واحد.

* أما في حالة إذا كان برنامج الأسمبلي يحتل عدة ملفات، فإن أحد هذه الملفات يجب أن يحتوي النموذج الرئيسي (انظر الشكل(6) صفحة ٢٣٠) .

أما باقي الملفات يجب أن تحتوي النموذج الثانوي المبين في الشكل (7) انظر صفحة ٢٣٠ **يمكنك ملاحظة ما يلي :**

١) يتكون كل من النموذج الرئيسي والنموذج الثانوي من أجزاء ثابتة وأخرى متغيرة من برنامج إلى آخر. فالأجزاء المتغيرة هي تلك المكتوبة داخل اطارات، وباقي الأجزاء هي ثابتة ولا تتغير من برنامج إلى آخر. لذا يمكن الاحتفاظ بنسخة عن هذه النماذج على القرص المغناطيسي . وعند الحاجة إلى كتابة برنامج جديد يمكنك عمل نسخة من هذا البرنامج و كتابة المعطيات والتعليمات اللازمة في الاماكن المحددة .

2) يحتوي قطاع الكود في النموذج الرئيسي على جبل لبناء المكدس اللازم لتوفير امكانية العودة إلى نطام DOS بعد الإنتهاء من تنفيذ البرنامج (انظر الجمل PUSH ,SUB,PUSH) ويحتوي ايضا جملا لإعطاء قيمة ابتدائية لمسجل المعطيات (انظر جملتي MOV) .

3)إن تلجمل الخمسة المذكورة في شكل ٧ ثفحة ١٣٢ مكتوبة في النموذج الرئيسي فقط بسبب أن قطاعي المعطيات في النموذجين يحملان نفس الاسم DSEG .أما اذا كانت أسماء قطاعات المعطيات مختلفة فيلزم اضافة جملتي MOV في النوذج الثانوي أيضاً .

4)يلاحظ ان قطاعي الكود في النموذجين يحملان نفس الاسم مما أدى إلى اعطاء الاجراء P2 صيغة NEAR.وفي حالة اختلاف اسماء قطاعات الكود فيلزم اعطاء الاجراء P2 صيغة FAR

5) يجب أن تحتوي جملة END في نهاية النموذج الرئيسي على اسم يحدد بداية تنفيذ البرنامج .أما جميل END في النوذج الثانوي فيجب أن لا تحتوي على اسم .

الأسمبلر: tasembler. البرنامج المستخدم لتحميل البرنامج المصدري المكتوب بلغة الأسمبلي إلى برنامج هدفي مكافى

البرنامج التنفيدي Executable hugram البرنامج الناتج من عملية الربط والوصل بين العدة برامج هادفية (أو برنامج واحد) بواسطة برنامج تحرير الريد والوصل Linker

البرنامج المصادر Source Peetana البرنامج المكتوب بلغة الأسمبلي أو احدى لغات البرمجة عالية المستوى ،

البرنامج الهدفي Object Prograntt البرنامج الناتج من عملية الترجمة بواسطة برنامج الأسمبلر أو برنامج Catalpoiler

التعليمات Instructions الأوامر التي تنفذ من قبل المعالج في مرحلة التنفيذ .

التوجيهات Directives أوامر تنفذ من قبل الأسمبلر في مرحلة الترجمة وتحمل معلومات حول عناصر البرنامج أو حول عملية الترجمة نفسها

محرر الربط والو حصل Linker البرنامج المستخدم لتحويل البرنامج البرامج) الهدفي إلى برنامج تنقيادي

محرر النصوص Text Editor البرنامج المتسخدم لأعداد النصوص، وادخالها واجراء التعديلات عليها

كشف الأخطاء Debugger البرنامج المستخدم لتنفيذ تعلميات البرنامج دفعة واحدة أو واحدة تلو الأخرى وله وظائف عديدة منها عرض محتويات المسجلات ومواقع الذاكرة وتعديلها ، تنفية، تعليمات البرنامج ... وغيرها .

الوحدة الخامسة: التوجيهات Directives OR Pseudo-Operation

*سناخذ جميع الامثلة على المايكرو أسمبلر (Macro-Assembler) والذي يطلق عليه احيانا المجمع المتسع المستخدم مع اجهزة اي بي ام الميكروية وموافقتها .

ولبيان اهمية التوجيهات في برنامج لغة اسمبلي ، اعود بك قليلا الي لغات البرمجة عالية السمتوى مثل لغة بيسك وباسكال، ففي مثل هذه اللغات تجد نوعين رئيسي من الجمل هما جمل تنفيذية واخرى غير تنفيذية .

من الجمل الغير تنفيذية (والتي يناظرها التوجيهات في لغة اسمبلي) جمل DATA, ENDفي لغة بيسك وجميع الاعلانات التعريفية في اقسام الاعلانات المختلفة في لغة باسكال بالاضافة الي جمل تعريف البيانات الاخرى مثل السجلات وغيرها في باسكال .

* وعلى سبيل المثال ، فإن جملة END هي بمثابة اشعار للمترجم بنهاية البرنامج المصدري لكي تتوقف عملية الترجمة . وبدون ذلك من الصعب ان يعرف المترجم اين يتوقف عند مروره على البرنامج المصدري في ذاكره الحاسوب بغرض ترجمته الي لغة الالة .

*ولذلك نجد ان التوجيهات هي توجيهات وأوامر للمايكرو واسمبلر توجهه في عملية ترجمة البرنامج المصدري الي لغة الالة وهي ضرورية ولا يكاد يخلو منها برنامج واحد مثلما ان البيانات والاسماء التعريفية ضرورية لبرنامج اللغات العالية المستوى .

*بالرجوع الي التوجيهات يمكن للماكرواسمبلر ان يحدد عدد المواقع المطلوب حجزها للبيانات في ذاكرة الحاسوب واللازمة لتنفيذ البرنامج عليها وكذلك حجز المواقع اللازمة لتخزين النتائج الناتجة عن عملية المعالجة هلى هذه البيانات ويمكن ايضا استخدام التوجيهات لتحديد بداية ونهاية البرنامج الفرعية في برنامج لغة اسمبلي وكذلك تحديد القطاعات المختلفة المكونة للبرنامج مثل قطاع البرنامج وقطاع البيانات وقطاع المكدس والقطاع الاضافي .

* اهمية جمل التحكم حيث تحدد جملة ORG عل سبيل المثال بداية البرنامج الهدفي عند ترجمة البرنامج المصدريفي حين تحدد جملة END نهاية البرنامج المصدري وبذلك يتوقف الماكرو اسمبلر اثناء عملية الترجمة من لغة اسمبلي الى لغة الالة عند مصادفته لمثل هذا التوجيه .

*اما توجيهات الطباعة فتبرز اهميتها في التحكم بشكل صفحات الطباعة من حيث الطول وعرض الصفحة وكذلك التحكم في طباعة العناوين الرئيسية والفرعية .

*وأخيرا فهناك اهمية لتوجيهات الترجمة المشروطة بانواعها حيث ان مثل هذه التوجيهات يمكن استخدامها عند وجود انماط من البرنامج او في حالة اختلاف أجزاء البرامج عن بعضها البعض قليلا .

ففي هذة الحالة يمكن استخدام جمل التوجيهات المشروطة لاختيار النمط المطلوب ترجمته ومن ثم تنفيذه وعلى سبيل المثال لا الحصر يمكن تضمين البرنامج المصدري في لغة اسمبلي جزئين بحيث يعالج الجزء الاول موضوع ارسال النتائج الى نوع معين من الات الطباعة الموصولة على منفذ المتوازي (parallel poet) والجزء الثاني يعالج موضوع ارسال النتائج الي نوع اخر من الآت الطباعة الموصولة على النوع الاخر من ميناء المخارج (output port) وهو منفذ التوالي (Serial port) وفي مثل هذه الحالة يمكن استخدام توجيهات الترجمة المشروطة لإختيار الجزء الاول او الجزء الثاني بدلا من كتابة برنامجين منفصلين لهذا الغرض.

*يمكن لجمل التوجيهات ان تحتوي على اربعة حقول ، بعضها اجباري وبعضها اختياري كما يلي :

[Name] pseudo-po [operand][;comment]

حيث ان الاقواس المربعة هنا تعني ان هذه الحقول ويمكن تواجدها في الجملة اذا دعت الحاجة لها ويمكن كذلك الاستغناء عنها في بعض الاحوال

*اما الحقل الثاني وهو Pseudo-op والذي يطلق عليه في بعض الاحيان Directive فهو اجباري ويجب تواجده في كل جملة من جمل التوجيهات .وتجدر الاشارة هنا الي ان هذا الحقل سمي بهذا الاسم لقرب الشبه قواعديا بين التوجيهات وتعليمات لغة اسمبلي حيث ان pseudo-op وتعني باللغة العربية شبه تعليمية و op وهنا اختصار لoperation والتي تعني عمليه.

*وبالنسبة للحقل الاول من الشكل العام للتوجيهة فهو اسم Name ، ويحدده المبرمج وينطبق عليه قواعد تكوين الاسماء في لغة اسمبلي ، وهذا الخقل ضروري مع يعض التوجيهات ولا يجوز استخدامه مع بعض الانواع من التوجيهات وهو اختياري مع ما تبقى من التوجيهات. مثال 1و2 و3 صفحة 246.

**التوجيهات الخاصة بالبيانات :

يمكن ان نحدد تحت هذا النوع من انواع التوجيهات خمس مجموعات رئيسية تؤدي كل منها وظائف مميزة ، ويمكن تلخصيها بما يلي :

1-توجيهات تعريف اسماء الثوابت Symbol Definition Directives:

*تستخدم هذه التوجيهات لإسناد قيمة ثابت او قيمة تعبير حسابي او نص من النصوص الي اسم تعريفي وبعد تعريف الاسم التعريفي يمكن الاستغناء عن استخدام مثل هذة الثوابت والتعابير والنصوص داخل البرامج واستخدام الاسماء التعريفية مكانها .

توجد اكثر من فائدة لاستخدام الاسماء التعريفية بدلا من استخدام القيم والتعابير والنصوص منها نذكر الاتي :

أ) بإستخدام اسماء تعريفية بدلا من الثوابت يمكن اختيار الاسماء الهادفة والتي تدل على المقصود منها وبذلك فإن استخدام مثل هذه الاسماء يساعد في جعل البرنامج اكثر وضوحا وأسهل للفهم مقارنة مع الحالة عندما تكون القيم والتعابير منتشرة هنا وهناك في البرنامج

ب) يمكن تجميع الاسماء التعريفية خاصة الثابت منها والتي لا تتغير قيمتها اثناء تنفيذ البرنامج في جزء واحد في بداية البرنامج . بعد ذلك يمكن استخدام هذه الاسماء التعريفية في اي جزء واحد من البرنامج بدلا من القيم التي تعرفها تلك الاسماء وفي حالة الحاجة الي إجراء اي تعديل على اي من القيم يمكن تعديلها مرة واحدة في قسم التعريفات عوضاً عن إجراء التعديل على القيم الكثيرة المبعثرة هنا وهناك في البرنامج .لذلك فإن استخدام الاسماء التعريفية يقلل من إمكانية الوقوع في الاخطاء مثل نسيان تعديل قيمة من القيم عند الحاجة لإجراء مثل هذا التعديل على قيمة من القيم ، وكذلك فإن عملية تعديل قيمة واحدة في موقع واحد اسهل من تعديل القيمة في كل موقع ترد فيه في البرنامج .

ومن التوجيهات المستخدمة بشكل اوسع لتعريف الاسماء التالي :

1) توجيهة EQU

*يمكن عن طريق هذه التوجيهة اقتران اسم تعريفي بقيمة ثابتة حيث يصبح من غير الممكن اعادة تعريف هذا الاسم التعريفي عن طريق تحديد قيمة اخرى له . وفي البرنامج يمكن للمبرمج استخدام الإسم التعريفي في كل مكان يحتاج فيه إلى القيمة الثابتة التي اسندت الي هذا الاسم .

*ولمقارنة عمل توجيهة EQU مع ما يناظرها في لغات البرمجة العالية المستوى ، نجد ان عمل هذه التوجيهة يشبه تعريف أسماء تعريفية للثوابت في قسم تعريف الثوابت constant section والذي يبدأ بالكلمة المحجوزة Const في برنامج لغة باسكال .

*وتنبع اهمية هذا التوجيه ، بتجميع جميع قيم الثوابت التي يستخدمها البرنامج في جزء واحد ، وعادة ما يكون ذلك في بداية البرنامج لذلك فإنه يصبح من السهل أجراء التعديلات على هذه القيم ومن غير الممكن نسيان بعضها خاصة ان الثابت الواحد يرد تعريفه في البرنامج مرة واحدة فقط .فعلى سبيل المثال يمكن للبرنامج حساب قيمة الضريبة ان يعتمد على قيمة ثابته قد تستخدم في عشرات الجمل ، وفي حالة تغيير قيمة الثابت حيث ان المعطيات قد تتغير ، يحتاج المبرمج ان يجري تعديلا على عشرات الجمل بدلا من تغيير قسمة الثابت مرة واحدة في توجيهة EQU

*والكلمةEQUتعتبر من الكلمات المحجوزة Reserved Words في لغة اسمبلي واما بالنسبة للصيغة العامة لتوجيهة EQU فهي على النحو التالي :Const_Name EQU expression

*يخضع الاسم التعريفي Const_Name لقواعد تعريف الاسماء المستخدمة في لغة اسمبلي والتي تعتمدعلى نوع المجمع .

*وأما بالنسبة للثابت (expression) فمن الجائز ان يكون هذا التعبير من النوع العددي الصحيح أو من النوع الرمزي . ففي الحالة الاولى غالبا ما يطلق على توجيهة EQU توجيهة المساواة العددية (Numeric equate) وفي الحالة الثانية يطلق على هذه التوجيهه اسم توجيهة مساواه السلاسل الرمزيه (String Equate)

*وتجد الإشارة الى ان توجيهة EQU لا تحجز مواقع في الذاكره للإسم التعريفي كما هو الحال مع المتغيرات حيث يتم حجز مواقع لها في ذاكرة الحاسوب

**ولمقارنة عمل التوجيهية EQU مع ما يناظرها في لغات البرمجة العالية المستوى، نجد أن العالم التوجيه يشبه تعريف أسماء تعريفية للثوابت في قسم تعريف الثوابت Constant Section والذي يبدأ بالكلمة المحجوزة Const في برنامج لغة باسكال.

*وتنبع أهمية هذا التوجيه، بتجميع جميع قيم الثوابت التي يستخدمها البرنامج في جزء واحد، وعادة ما يكون ذلك في بداية البرنامج. لذلك فإنه يصبح من السهل اجراء التعديلات على هذه القيم ومن غير الممكن نسيان بعضها خاصة أن الثابت الواحد يرد تعريفه في البرنامج مرة واحدة فقط. *فعلى سبيل المثال يمكن لبرنامج حساب قيمة الضريبة أن يعتمد على قيمة ثابتة قد تستخدم في عشرات الجمال، وفي حالة تغيير قيمة الثابت حيث أن المعطيات قد تتغير.

*يحتاج المبرمج أن يجري تعديلا على عشرات الجمل بدلا من تغيير قيمة الثابت مرة واحدة في توجيهة EQU.

*والكلمة EQU تعتبر من الكلمات المحجوزة Reserved Words في لغة أسمبلي، وأما بالنسبة للصيغة العامة التوجيهة EQU فهي على النحو الآتي: Const_Name EQU expression

*حيث يخضع الإسم التعريفي Cons_Name لقواعد تعريف الأسماء المستخدمة في لغة أسمبلي و التي تعتمد على نوع المجمع .

*وأما بالنسبة للثابت (expression)، فمن الجائز أن يكون هذا التعبير من النوع العددي الصحيح أو من النوع الرمزي. في الحالة الأولى غالبا ما يطلق على توجيهة EQU توجيهة المساواة العددية (Numeric equatec) وفي الحالة الثانية يطلق على هذه التوجيهة اسم توجيهة مساواة السلاسل الرمزية (String Equate) .

وتجدر الإشارة إلى أن توجيهة EQU لا تحجز مواقع في الذاكرة الجسم التعريفي كما هو الحال مع المتغيرات حيث يتم حجز مواقع لها في ذاكرة الحاسوب . مثالل و5و6 4 صفحة 250

*أن الموقع الطبيعي العادي للتوجيهات الخاصة بالبيانات هو بداية البرنامج وذلك قبل استخدامها في الجمل الأخرى للبرنامج .

*وبالنسبة للتعبير الوارد في توجيهات تعريف الأسماء فيمكن أن يكون متغيرة أو اشارة (Label) أو قيمة ثابته أخرى أو تعبير يحوي على عنوان متغير وما إلى ذلك.

* ننتقل إلى الصنف الثاني من توجيهات تعريف الأسماء .

ثانيا: توجيهة المساواة العددية القابلة لإعادة التعريف (=):

تستخدم هذه التوجيهة لإسناد قيمة ثابت عددي إلى اسم تعريفي حيث أنه من الممكن تغير القيمة المسنده إلى الإسم التعريفي في أي مكان في البرنامج ويتم إسناد هذه القيم بواسطة الماكرو أسمبلر أثناء عملية تحويل البرنامج المصدري من لغة التجميع إلى لغة الألة. وأود أن أنوه إلى أنه وبالرغم من إمكانية إعادة تعريف القيمة المسندة إلى الإسم التعريفي في أي مكان في البرنامج المصدري، إلا أنه تستخدم قيمة ثابتة واحدة لأي من التطبيقات وتبقي عادة القيمة ثابتة طيلة تنفيذ البرنامج .

*ومن هذا المنطلق، فإن الإستخدام الأمثل لهذه التوجيهة هو في بعض الماكرو والقطاعات التي يتكرر تنفيذها والتي يعتمد تنفيذها أو عدمه على بعض القيم التي يتم حسابها أثناء ترجمة البرنامج بواسطة الماكرو أسمبلر من لغة المصدر إلى لغة الألة.

*وفي مثل هذه الحالة يمكن إسناد قيم مختلفة للإسم التعريفي وحسب الضرورة .

*والمثال ٧صفحة ٢٥٤ يوضح صيغة استخدام هذه التوجيهة وبعض المفاهيم الأنفة الذكر . تدريب 4 صفحة255

***توجيهات تعريف البيانات**

*تعمل هذه المجموعة من التوجيهات على تعريف المتغيرات وحجز مواقع لها في الذاكرة الرئيسية للحاسوب مع امكانية تحديد قيم ابتدائية لتخزينها في تلك المواقع التخزينية التي تم حجزها.

*ويمكن للقيم (البيانات) التي يتم تخزينها في تلك المواقع ان تكون قيما عددية, او سلسلة رمزية (رمز او اكثر) , او تعابير تؤول قيمها الى ثوابت وما الى ذلك . ويقوم الماكرو واسمبلر بتحويل جميع قيم الثوابت الى قيم ثنائية على شكل وحدات تخزينية (مثل بايت ,او كلمة ((والتي تعادل 2 بايت)وما الى ذلك من وحدات تخزينية) يتم تحديدها باستخدام توجيهات تعريف البيانات . وتعتبر عملية تخزين القيم الثنائية في المواقع التخزينية المخصصة من احدى وظائف الماكرو واسمبلر ويتم ذلك اثناء مرحلة ترجمة البرنامج من لغة المصدر (اي لغة التجميع) الى لغة الالة.

*وبالنسبة للصيغة العامة لتوجيهة تعريف البيانات فهي على النحو التالي: [... ,] expression Dn [Name]

*حيث نجد اسم المتغير بين قوسين مربعين ويعني ذلك ان الاسم هو اختياري ويظهر عند الضرورة لذلك (اي عندما تكون هناك ضرورة للرجوع الى المواقع التخزينية باسماءها).

*واما بالنسبة للتوجيهة التي تلي اسم المتغير والتي تعمل على حجز المواقع فتاخذ احد الاشكال التالية DT:او DQ اوDDاو

حيث أن:

DB:تعني Define Byte وهي توجيهة لحجز بايت , ويمكن للموقع الواحد من هذا النوع ان يخزن قيمة صحيحة بدون

اشارة تتراوح من 0 الى 255 او تتراوح بين 128 – و 127 في حالة اعتبار الاعداد باشارة.

DW: وتعني Define Word,وهي توجيهة تستخدم لحجز وحدات تخزينية مكونة من 2 بايت (16 خانة ثنائية)

*ويمكن للوحده التخزينية الواحدة من هذا النوع تخزين قيم عددية صحيحة تتراوح بين 0 و 65,535 في حالة الاعداد بدون اشارة او قيم تتراوح بين 32768 –و32767 في حالة الاعداد باشارة.

*عند طباعة محتويات الموقع الواحد من هذا النوع من الوحدات التخزينية يتم ذلك بالطرية المعتادة لكتابة الرقم (اي ان الجزء الاكبر وزنا الى يسار الجزء الاقل وزنا) . على سبيل المثال فان القيمة 1990 والتي بالنظام السادس عشر يتم طباعتها , من قبل معظم الانظمة المستخدمة مع الماكرو واسمبلر للطباعةC607تعادل القيمة C607.بنفس الصورة ,مع ان تخزين هذه القيمة الداخلي يتم على الشكل

(يخزن الى يسار البايت الاكثر قيمة (وقيمته 07 في هذا المثالC6.(اي ان البايت الاقل قيمة (وهو

DD:وتعني Define Doubleword وهي توجيهة تستخدم لحجز وحدات تخزينية حيث تتكون الوحدة التخزينية الواحدة

. من 4 بايت (اي كلمتين حيث ان الكلمة الواحدة تتكون من 2 بايت) , او ما يعادل 32 ثنائية

وهذا يعني ان الوحدة التخزينية الواحدة من هذا النوع لها القدرة على استيعاب ارقام عددية صحيحة تتراوح بين صفر و 295 , 967 , 294 , 4 في حالة الاعداد بدون اشارة ,اما في حالة اعتبار الاشارة فان مدى الاعداد التي يمكن تمثيلها يتراوح بين 648 , 483 , 147 , -2 و 647 , 438 , 147 , 2

DQ:وتعني Define Quadword وهي توجيهة تستخدم لحجز وحدات تخزينية طول الواحدة منها 4 كلمات (اي 8 بايت)

او ما يعادل 64 خانة ثنائية.

DT:ويعني Define Ten Bytes وهي توجيهة تستخدم لحجز وحدات تخزينية طول الواحد منها 10 بايت . والاستخدام , BCDالمضغوط (packed Binary Coded Decimal)الضمني لهذا النوع من التوجيهات هو مع الاعداد الممثلة بنظام اما بالنسبة للتعبير (expression) يمكن ان يكون ثابتاعدديا او رمزيا او اشارة استفهام او يمكن ان يستخدم رمز التكرار DPU : والذي ياخذ الصيغة التالية

[Name] Dn value DPU (expression)…

مثال:(8)

توضح الجمل الواردة في هذا المثال الكثير من الامور المهمة والمفاهيم التي تحكم استخدام توجيهات تعريف المتغيرات وحجز مواضع لها واعطائها قيم ابتدائية نورد منها الاتي:

1 -يمكن استخدام التوجيهات لتعريف متغيرات مع امكانية اسناد قيم ابتدائية لهذه المتغيرات.

2 -يمكن للتعبير الوارد في توجيهة تعريف البيانات ان يكون تعبيرا تؤول قيمته الى قيمة ثابتة كما هو الحال في الجملة الاولى وهي:

VAR 1 DB 45

او ان تكون علامة استفهام لتشير الى ان الموقع او المواقع التي تم حجزها لم يتم تحديد قيمتها الابتدائيةو كما هو الحال في الجملة الثانية وهي: VAR 2 DB ?

دون تحديد قيمة ابتدائية لهذا الموقع . يمكن للمتغير ان يتكون من عدة قيم يفصلها -23 VARحيث يتم حجز موقع للمتغير

عن بعضها البعض وتحدد عدد هذه القيم بطول السطر الواحد كما هو الحال في الجملة التالية (Comma): الفارزة

VAR 3 DB 10 , 11 , 12 , 20 , 25

يعني VAR3وعند تعريف المتغيرات بهذه الطريقة يقوم الاسمبلر بحجز مواقع متجاورة لهذه القيم وان اي استخدام للمتغير

استخدام القيمة الاولى وهي 10 , وللوصول الى القيمة الثانية (وهي 11 في هذا المثال) يكون عن طريق استخدام التعبير

وهكذا,)VAR3+2). والوصول الى القيمة التي تليها يكون عن طريق استخدام التعبير(1+ VAR3)

فقط ويتم تخزين هذه الرموز بالطريقة -4 DBيمكن حجز سلسلة رمزية طولها يزيد على رمزين اثنين باستخدام الصيغة

المعتادة (اي من اليسار الى اليمين) , كما هو مبين في الجملة الرابعة في المثال السابق, ولتميز الثابت الرمزي تستخدم

.ويقوم الاسمبلر بتخزين رمز الثابت "January" او حاصرتين علويتين كما في'January' الحاصرة العلوية كما في

الرمزي على شكل شيفرة اسكي . اما الاشكال الاخرى لتوجيهات حجز البيانات فلا تسمح بتعريف سلسلة رمزية يزيد طولها على اكثر من رمزين.

5 -يمكن ان يكون الثابت ممثلا بالنظام لسادس عشر كما هو مبين في الجملة الخامسة وهي: VAR5 DW 0FF0H

(اي ان العدد ممثل بالنظام السادس عشر) واود ان ابين على ان الاسمبلر يقومHexadeximalيعنيH حيث ان الحرف

بتحويل جميع الثوابت العددية الى ما يكافئها بالنظام السادس عشر, ومن ثم يتم تخزين الرقم على شكل بايتات وبترتيب معكوس كما تم شرحه سابقا . اما بالنسبة للاشكال المختلفة التي يمكن استخدامها لتمثيل الثوابت العددية فيتم التطرق اليها في مكان اخر من هذه الوحدة.

6 -يمكن توزيع العناصر التي تم حجزها للمتغير اكثر من سطر واحد , وفي مثل هذه الحالة لا يتكرر ذكر المتغير بل نكتفي

في المثال السابق Tabel1. تكرار توجيهة الحجز كما هو الحال عن تعريف

في هذاVAR8في توجيهات تعريف المتغيرات كما هو الحال في تعريف -DUP7 يمكن استخدام رمز الحجز المتكرر

المثال وكما هو موضح في الجمل الاخرى الاتية:

DB 10 DUP (?)

DB 5 DUP (12)

DB 2 DUP (3 DUP (1,2))

حيث ان الجملة الاولى تعمل على حجز 10 بايت دون اسناد اي قيم ابتدائية الى اي من هذه المواقع , وان الجملة الثانية تعمل على حجز 5 بايت يحتوي كل منها على القيمة 12 بالنظام العشري

بالنظام السادس عشر) .0) cوالتي تكافئ

وتعمل الثالثة على حجز 12 بايت تحتوي على القيم الابتدائية كما يلي:

رقم البايت : 1 2 3 4 5 6 7 8 9 10 11 12

محتوياته : 1 2 1 2 1 2 1 2 1 2 1 2

8 -يمكن المزج بين البيانات العددية والرمزية عند استخدام توجيهات نعريف البيانات كما هو مبين في السطر الرابع من هذا المثال.

بالاضافة الى ما تقدم اورد بيان الامور التالية حول توجيهات تعريف البيانات:

*يمكن للتعبير المستخدم في توجيهات تعريف البيانات ان يكون عنوانا كما في المثال صفحة 260:

يقوم الاسمبلر بتحويل جميع الثوابت العددية الى النظام السادس * DT,DO,DO,DW عند التعامل مع التوجيهات

عشر ومن ثم تخزن القيم بترتيب معكوس .وعلى سبيل المثال فان القيمة (12345)10 يتم تحويلها الى القيمة (00003039)16

يقوم الاسمبلر بتحويل تعريف متغير باستخدام توجيهة DT,DQ,DD,DW عند التعامل مع التوجيهات

التي يحجز 4 بايت للعنصر الواحد وان كل بايت يمكن ان يستوعب رقمي بالنظام السادس عشر ولذلك فان الرقم00003039 يتكون من ثمانية مواقع بالنظام السادس عشر . وعند تخزين الرقم عند عملية الترجمة , يتم تخزينه بالشكل 3930 00 00.

,من غير الممكن ان يزيد عدد رموز السلسلة الرمزية مع التوجيهات الاخرى لتعريف البيانات * DB باستثناء التوجيهة

على رمزين . وكذلك فانه يتم تخزين الرمزين بترتيب معكوس لظهورهما مع التوجيهة

.("وبالاضافة الى عكس الرمزعند تخزينه يقوم الاسمبلر بوضع رمزين في "CP يصبح)"PC" اي ان الثابت الرمزي

الموقعين التخزينين الاول والثاني ويتم ملئ المواقع المتبقية اصفارا . وعلى سبيل المثال اذا اعطينا التعريف التالي: VAR DD AB

بنظام اسكي)B (ويمثل قيمة الرمز حيث يتم حجز 4 بايت , ويخزن في البايت الاول القيمة 42 بانظام السادس عشر

بنظام اسكي) . اما المواقع المتبقية Aوفي البايت الثاني تخزن القيمة 41 بالنظام السادس عشر (وتمثل هذه القيمة الرمز

فان عددDTبDD فيتم ملأها اصفارا, وعدد المواقع في هذا المثال هو موقعين طول كل منهما بايت. وفي حال استبدال

. المواقع التي سيتم ملئها اصفارا يكون ثمانية مواقع

LABEL **توجيهة** : تستخدم هذه التوجيهة لاعادة تعريف خواص بعض الاسماء التي تم تعريفها مسبقا في ابرنامج , حيث يعطي ذلك مرونة اكثر للمبرمج للتحكم في الاسماء وخواصها .والشكل العام لهذه التوجيهة هو: Name LABEL type

حيث ان Name *هو الاسم الجديد بالخواص الجديدة والذي يمكن استخدامه بالاضافة الى الاسم الذي تم تعريفه مسبقا وبخواص مختلفة

اما Type: فيمكن ان يكون احد القيم التالية:

TBYTE , QWORD , DWORD , WORD , BYTE ,FAR , NEAR

حيث انFAR , NEAR . تستخدم مع الاجراءات والشارات وبقية القيم تستخدم مع المتغيرات

حيث يمكنFAR الى NEAR لتغيير خاصية جملة من الجمل منLABELوعلى سبيل المثال يمكن استخدام توجيهة

بعد ذلك استخدام تعيلمة نقل التحكم (والتي يمكن ان تكون معرفة في قطاع اخر) للانتقال الى هذه التعليمة . ومن الامثلة على ذلك:

```
XX      LABEL     FAR
MOVE    AX,0
```

حيث انXX تستخدم لتعليم الامرMOVويمكن عندئذ استخدام جملة نقل التحكم والتي يمكن ان تكون معرفة في قطاع اخر .

للانتقال الى التعليمة MOV.ويمكن استخدام توجيهة LABEL في تحديد نقطة عبور اخرى الى الاجراء الفرعي غير بداية الاجراء كما في المثال9

***توجيهات تعريف القطاعات والبرامج الفرعية**

****يمكن تقسيم برنامج لغة التجميع إلى عدد من القطاعات،حيث أن أي من هذه القطاعات يندرج تحت نوع من أربعة أنواع هي على الآتي.**

1.قطاع البيانات (data segment) وتتم عنونته بواسطة مسجل قطاع البياناتDS.

2. قطاع المكدس (stack segment) وتتم عنونته بواسطة مسجل المكدس SS.

3. القطاع الإضافي (Extra segment) وتتم عنونته بواسطة مسجل القطاع الإضافيES.

4.التعليمات (code segment) وتتم عنونته بواسطة مسجل قطاع التعليماتCS.

*كذلك يمكن احتواء البرامج على عدد من البرامج الفرعية أيضاً داخل قطاع (قطاعات) التعليمات.

***ويمكن تحديد أبرز توجيهات تعريف القطاعات والبرامج الفرعية كما يلي:**

1.توجيهة بداية القطاع(Segment). 2. توجيهة نهاية القطاع (Ends).

3.توجيهة اخبار المترجم باسم مسجل القطاع الذي يحوي عنوان القطاع (Assume).

4.توجيهة بداية البرنامج الفرعي (الإجراء)Proc.

5.توجيهة نهاية البرنامج الفرعي (الإجراء)(Endp).

6.توجيهة تجميع القطاعات (Group). 7.توجيهة Public. 8.توجيهة Extrn.

***أولا توجيهة SEGMENT**

*إن العمل الأساس لهذه التوجيهة هو تحديد بداية القطاع،حيث يمكن للبرنامج الواحد أن يحتوي على أي عدد من القطاعات , وأن هذه القطاعات تندرج تحت نوع من الأنواع الانفة الذكر.

*يمكن بيان الصيغه العامة لتوجيهة Segmentعلى النحو التالي: Name Segment[align][combine][class]

*حيث أن:Name يبين إسم القطاع ويمكن لهذا الإسم أن يكون فريدا أو مكررا في البرنامج وفي هذه الحالة فإن القطاعات التي تحمل نفس الاسم تعامل على أنها قطاع واحد مميز.*والمميز الوحيد لهذا التكرار هو عند استخدام جزئين مختلفين من نفس القطاع في منطقتين مختلفتين من البرنامج المصدري(Defferent source Modules) كما تم شرحه في التوجيهات اللاحقة .وفي مثل هذه الحالة يمكن تكرار اسم القطاع.

*وبالنسبة للأنواع الواردة بين أقواس مربعه فهي اختياريه وهي عبارة عن توجيهات الاسمبلر والبرامج المساعده الأخرى التي تأتي مع الاسمبلر مثلlinker.*وفي حالة ورودها كليا أو جزئيا يجب أن تظهر بنفس الترتيب وتعمل هذه التعليمات على إخبار الاسمبلر وبرنامج الربط(linker) بكيفية تجهيز وربط القطاعات مع بعضها البعض في البرنامج عند تحميلها إلى ذاكرة الحاسوب واود أن أبين أن معظم المبرمجين المبتدئين يستخدمون هذه التوجهة في أبسط صورها دون استخدام الانواع الاختياريه حيث يترك للاسمبلر استخدام القيم الضمنية(Default values) ،التي تم تحديدها مسبقاً من قبل النظام المستخدم.

وبالنسبة للنوع [align] فهو يحدد للاسمبلر مدى العناوين المحتملة والتي يمكن استخدامها كقيمه ابتدائيه لعنوان القطاع (أي عنوان بداية القطاع) وتجدر الإشارة إلى وجود عدة قيم لهذا الخيار يمكن تلخيصها بما يلي:

Byte: حيث يحدد البايت الآتي المتوفر سيتم استخدامه لتحديد القطاع الذي سيتم تعريفه.

Word: ويعني استخدام الكلمة الآتيه المتوفرة على أنها عنوانا لبداية القطاع.

Dword: ويعني استخدام بداية الكلمة المضاعفة التالية بداية لعنوان القطاع (الكلمة المضاعفة تساوي 4 بايت .)

Para: ويعني استخدام بداية الفقرة الآتي (حيث أن الفقرة تساوي 16بايت

Page: ويعني استخدام بداية الصفحه التاليه بداية لعنوان القطاع (حيث أن الصفحه الواحدةتساوي256بايت .)

وهذا يعني أن المترجم في هذه الحالة يعتبر أن الذاكرة مقسمة الى صفحات طول الصفحه 256بايت.

*وفي حالة عدم تحديد أي من هذه الأنواع عند تعريف القطاع (أي عند استخدام توجيهة segment), يعني أن النوع paraهو الذي يستخدم ضمنيا وتستخدم هذه الأنواع الانفة الذكر من قبل البرنامج (linker) وذلك لتحديد عنوان البدايه النسبي للقطاع المعرف ،وكذلك من قبل نظام التشغيل Dos لحساب بداية العنوان الفعلي للقطاع عند تحميل البرنامج في ذاكرة الحاسوب.

*أما بالنسبة للنوع[combine] فهو يحدد كيف يتم تجميع القطاعات التي تحمل نفس الاسم ويوجد عدة قيم للنوعcombine هي :

puplic: وتستخدم لتجميع جميع القطاعات التي تحمل نفس الاسم لتكوين قطاع واحد حيث تصبح عناوين جميع التعليمات والبيانات في القطاع الجديد نسبيه إلى العناوين للجزء في احد مسجلات القطاعات ويتم تعديل ابعاد التعليمات والبيانات لكي تصبح نسبة إلى بداية القطاع الجديد بدلاً من القطاع الذي كانت معرفة فيه قبل استخدام puplic

stack:وهذا النوع يشبه النوع السابق باستثناء أن جميع العناوين في القطاع الجديد تصبح نسبة إلى مسجل قطاع المكدس(ss) ويتم تخزين طول القطاع في مسجل مؤشر المكدس(sp) ويجب التنويه إلى أنه عند تعريف قطاع المكدس compbineيجب استخدام هذا النوع من أنواع القيم حيث أنه في مثل هذه الحالة يتم تخزين القيمة الابتدائيه لعنوان قطاع المكدس في مسجل قطاع تلقائياً وفي حالة تعريف قطاع المكدس دون استخدام النوع فإن عملية تهيئة المسجلين تقع على عاتق المبرمج وفي حالة عدم القيام بذلك ينتج عن ذلك خطأ برمجي.

common:يختلف هذا النوع من الأنواع السابقة حيث يتم إنشاء عدة قطاعات متراكمةoverlapping حيث تبدأ جميع القطاعات التي تحمل نفس الاسم من نفس العنوان وبالنسبة لطول القطاع الناتج فهو مساو لطول أطول هذه القطاعات.

memory:عند استخدام هذه القيمة يتم تجميع جميع جميع القطاعات التي تحمل نفس الاسم في قطاع واحد متتابع ويتم معالجة جميع القطاعات من هذا النوع بنفس الأسلوب الذي تتم به معالجة commonالقطاعات التي تحمل النوع السابق والهدف من استخدام هذا النوع هو التوافق بين الأنواع المختلفة من برامج الربطlinkers مثل البرامج المعدة من قبل شركة مايكروسوفت وتلك المعدة من قبل شركة إينتل (الشركة الصانعه للمعالج الدقيق المستخدم.)

At address:وعند استخدام هذا النوع فإن ذلك يؤدي إلى جعل جميع عناوين الشارات والمتغيرات المعرفة في ذلك Atالقطاع نسبيه إلى العنوان الوارد بعد .

وفي حالة عدم استخدام أي من أنواع[combine] المذكورة أعلاه مع توجيهةsegment فهذا يعني أن كل قطاع له نوعه الخاص . وفي مثل هذه الحالة لا يتم تجميع القطاعات التي تحمل نفس الاسم إلى قطاع واحد.

*عند استخدام أكثر من قطاع واحد بنفس الاسم ،يجب أن تحمل هذه القطاعات (التي تحمل نفس الاسم في البرنامج الواحد) نفس الخواص ،او خواص غير متعارضة على الأقل .لذلك فإنه ينصح في حالة وجود عدة قطاعات بنفس الإسم أن يحدد نوع القطاع الاول منها ،ومن ثم لا توجد ضرورة لتعريف أنواع القطاعات اللاحقة التي تحمل نفس الاسم.

*وفي معظم البرامج (إن لم يكن في جميعا) يجب تعريف قطاع واحد على الأقل يحمل النوع stack.

*وفي حالة عدم وجود مثل هذا النوع من القطاعات قد يؤدي ذلك إلى إعطاء إشعار إلى المبرمج من البرنامج linker.*ويمكن إهمال هذل التحذير دون تأثير على البرنامج في حالة وجود مبرر لعدم إعطاء هذا النوع لأي من قطاعات البرنامج،كما هو الحال عند تعريف برنامج يقصد بتحويله إلى نوع (com),حيث أنه في مثل هذه الحالة من غير الضروري وجود قطاع منفصل من نوعstack

*فإن الهدف الحقيقي من استخدام النوع classهو ربط قطاعات ذات اسماء مختلفة ولكنها تتشابه في العرض الذي أنشأت من أجله.*وفي حالة تحديد النوع للقطاع،يجب أن يظهر النوع بين حاصرات علويه كما في classاو data وفي حالة عدم تحديد النوع للقطاع فيفهم ضمنا أن نوع هذا القطاع هو null.

*حالة استخدام أسماء لنوع القطاع(class)،يجب أن لا تستخدم هذا الاسماء للمتغيرات أو الشارات في البرنامج المصدري.

***ومن الانواع المستخدمه الانواع التاليه:**

Stack: وتستخدم مع قطاع المكدس

Data: وتستخدم مع قطاع البيانات

Code: وتستخدم مع قطاع التعليمات حيث يعامل البرنامج linker جميع القطاعات التي تحمل هذا النوع على أنها قطاع التعليمات.

والأمثلة التاليه تبين استخدام الخيارات المختلفة مع توجيهة segment مثال 10 و11 وتدريب 6 صفحة 270

ثانياً توجيهة ENDS

* تستخدم هذه التوجيهة لإخبار الأسمبلر بنهاية القطاع،حيث أن جميع الجمل الواردة بين توجيهة SEGMENT و أول ENDS تعتبر جملاً خاصة بنفس القطاع، حيث تستخدم هذه المعلومات بواسطة الأسمبلر وبرنامج الربط Linker أثناء عملية ترجمته البرنامج المصدري إلى لغة الآلة

ثالثاً: توجيهة ASSUME

*تستخدم هذه التوجيهة لإخبار الأسمبلر عن إقران القطاعات المعرفة في البرنامج مع المسجلات الخاصة بالقطاعات وهي (ES,SS,DS,SS)

***توجد عدة صيغ لهذه التوجيهة كما يلي:**

Name [.Segment Register Name.] ASSUME Segment Register

ASSUME Segment Register: NOTHING

ASSUME NOTHING

حيث:

*:NAME يعني اسم القطاع او مجموعة من القطاعات (التي يمكن تكوينها بواسطة توجيهةGROUP)التي سيتم إقرائها مع احد مسجلات القطاعات

*Segment Register :يعني أحد مسجلات القطاعات الاربعة المذكورة أعلاه

*NOTHING تستخدم لإلغاء اي اقتران سابق للقطاع مع اي من مسجلات القطاعات وعلى سبيل المثال،عند استخدام الصيغة الاخيرة من الصيغ لثلاث أعلاه،فانه يتم إلغاء اي اختيار سابق لمسجلات القطاعات تتم عن طريق توجيهة ASSUME سابقه

رابعاً توجيهة بداية البرنامج الفرعي (الاجراء)PROC

*تحدد هذه التوجيهة بداية الاجراء.والمقصود بالبرنامج الفرعي (الاجراء) هو مجموعة من الجمل تكتب مرة واحدة في البرنامج وتنفذ من مواقع مختلفة في البرنامج عن طريق جملة استدعاء خاصة بذلك.لذلك فإن تعريف الإجراء لا يعني بأي حال من الأحوال تنفيذه،بل إن عملية التنفيذ تكون مقرونة مع عملية استدعائه،لذلك فإن من فوائد استخدام البرانج الفرعية ان يصبح البرنامج قصيرا وهذا يوفر جهداً في عملية اعداده

*ويمكن للبرنامج الفرعي ان يحتوي على التعليمة RET، وبعد تنفيذها ينتقل المعالج الدقيق لتتفيذ الجملة التالية لجملة الاستدعاء مباشرة

ويتميز الاجراء بان يكون إما من النوع NEAR(النوع القريب) أو النوع FAR(النوع البعيد).ويرد هذا التوصيف NEAR او FAR عند استخدامه بعد توجيهة PROC مباشرة على نفس السطر كما في المثال الآتي AA PROC FAR

*وفي حالة عدم تحديد نوع الإجراء يفهم ضمناً انه من النوع NEAR يمكن استدعاءه من داخل القطاع الذي يتم تعريف الاجراء فيه فقط ولا يمكن استدعاءه من داخل اي قطاع آخر،بينما يمكن استدعاء الاجراء من النوع FAR من اي قطاع في البرنامج ويمكن ان يشتمل القطاع الواحد على اكثر من اجراء واحد

خامساً... توجيهة نهاية البرنامج الفرعي(الاجراء)ENDP

تعتبر هذه التوجيهة بمثابة اشعار للمترجم بنهاية الاجراء حيث يفيد ذلك في عملية ترجمة الاجراء بتحديد نهايته.ويجب ملاحظة انه يوجد جملة ENDP لكل جملة PROC وعند تعريف توجيهة ENDP يجب ان يستخدم نفس الاسم المستخدم عند تعريف الاجراء .وعلى سبيل المثال،فان تعريف توجيهة ENDP للبرنامج الفرعي السابق يكون على النحو التالي: AA ENDP

سادسا... توجيهة تجميع القطاعات GROUP

*تستخدم هذه التوجيهة لتجميع عدد من قطاعات البرنامج حيث تبدا جميعها في نفس الموقع في ذاكرة الحاسوب الرئيسة. ومن هذا المنطلق يمكن استخدام مسجل قطاع واحد عند عنونة أي موقع في هذه القطاعات وذلك عن طريق تحميل عنوان هذه القطاعات إلى مسجل القطاع.

*وبالنسبة للصيغة العامة لهذه التوجيهة فهي على النحو التالي: Name GROUP Segment[,Segment]...

حيث ان: Name: يمثل الاسم للمشترك الجديد لهذه القطاعات ويعني عنوان البداية للقطاعات وبذلك فإن جميع العناوين المعرفة في هذه القطاعات سوف تحول إلى عناوين نسبية وذلك نسبة لبداية القطاع الجديد والمشار إليه ب Name بدلا من القطاع الذي عرفت فيه في البرنامج لمصدري.ومثال ذلك كما هو مبين في المثال 13 صفحة 274

**وبما أننا مازلنا في صدد الحديث عن القطاعات والبرامج الفرعية فقد تتطلب بعض التطبيقات بأن تتم برمجة ذلك التطبيق على شكل وحدات منفصلة (modules)، حيث يتم الربط بين هذه الوحدات في مرحلة متقدمة لتكوين برنامج واحد متكامل يؤدي الوظيفة المطلوبة. وقد تؤدي كل وحدة من هذه الوحدات وظيفة خاصة، حيث أن عملية تجزئة البرنامج الكبير إلى عدة وحدات له عدة فوائد، من أبرزها أن عملية تجزئة المسألة المعقدة إلى عدة مسائل سهلة يؤدي إلى حل تلك المسألة بسهولة، وكذلك فإن عملية اعداد البرنامج على شكل وحدات منفصلة يزيد من الإستفادة من تلك الوحدات، حيث يمكن استخدامها في عدة برامج.

* ولتسهيل مهمة تجزئة البرنامج إلى وحدات منفصلة يوجد توجيهتين مهمتين لهذه الغاية هما PUBLIC و EXTRN. ووجدت عزيزي الدارس أنه من الأنسب درج هاتين التوجيهتين ضمن توجيهات القطاعات والبرامج الفرعية ، مع أن بعض المؤلفين قد يخالفني الرأي . وتعالج هاتين التوجيهتين إما تحت عنوان منفصل أو مع أي من أنواع التوجيهات الأخرى. وعلى أي حال، فالمهم هو معرفتهما ومعرفة كيفية عملهما .

7. توجيهة PUBLIC:

لا تستخدم هذه التوجيهية للإعلان عن الأسماء (سواء كانت المتغيرات أو اشارات، أو الإجراءات وما إلى ذلك) على أنها عامة (PUBLIC) بالنسبة للوحدات الأخرى (أي أنه يمكن استخدامها من قبل وحدات أخرى في البرنامج غير الوحدة التي عرفت فيها).

في حالة عدم تعريف الإسم على أنه من النوع PUBLIC فتصبح تلك الأسماء معرفة للوحدة التي عرفت بها فقط. وفي حالة تعريف اسم معين على أنه من النوع PUBLIC فإنه يتم تجميع جميع الأسماء التعريفية التي تحمل نفس الاسم في الوحدات المختلفة وإعطاءها عنوان واحدا من قبل البرنامج (Linker) الذي يقوم بتحديد مواقع في الذاكرة لجميع الأسماء والقطاعات والإجراءات والوحدات المكونة للبرنامج والشكل العام لتوجيهة PUBLIC هو: PUBLIC Name [.name .....]

حيث أن :Name يمثل اسم المتغير أو اسم شارة (Label) (بما في ذلك اسماء الإجراءات) أوالأسماء التي تعرف باستخدام توجيهة EQU . مثال 14 صفحة 275

8 توجيهة EXTRN:

تستخدم هذه التوجيهة للدلالة على الأسماء (من متغيرات وشارات واسماء إجراءات)المعرفة في وحدات أخرى غير تلك التي ستستخدم بها . وكما تعلم عزيزي الدارس فإنه عنداستخدام الأسماء (مثل اسماء المتغيرات والشارات) في البرنامج يجب أن تكون تلك الأسماء معرفة للأسمبلر .

* وتعمل توجيهة EXTRN على إخبار الأسمبلر بعدم إصدار اشعار بحدوث خطأ عند استخدام اسم من الأسماء غير المعرفة في الوحدة الحالية، والذي تم الإعلان عنه بواسطة توجيهة EXTRN على أنه معرف في وحدة أخرى من وحدات البرنامج، حيث يفترض الأسمبلر على أن ذلك الإسم قد تم تعريفه في وحدة أخرى. وعلى أي حال فإن تلك الأسماء يجب أن تعرف في وحدات البرنامج ويجب أن يتم ذلك عن طريق توجيهة PUBLIC،

وإلا فإنه سيتم إشعار المبرمج بعدم تعريف الإسم في وحدات البرنامج وهذا يؤدي إلى -برمجي الأسماء التي تم الإعلان عنها على أنها EXTRN (أي معرفة في وحدات من وحدات البرنامج غير تلك التي تستخدمها) يجب أن يتم الإعلان عنها كذلك باستخدام توجيهة PUBLIC في وحدات أخرى من وحدات البرنامج .

والشكل العام لتوجيهة EXTRN هو: EXTRN name: type[, name: type] ...

وهناك عدة أنواع تستخدم مع توجيهة EXTRN تعتمد على نوع الإسم المعرف. فإذا كان الإسم المعرف من المتغيرات فيكون النوع (type) أحد الأنواع التالية :TBYTE (QWORD, DWORD, WORD, BYTE,)

أما إذا كان الأسم يمثل اسم اجراء أو شارة (Label) في البرنامج فيمكن للنوع (أيtype) أن يكون NEAR أو FAR أو PROC

أما في حالة كون الإسم معرفة باستخدام EQU أو = فيكون النوع (أي type) مساوية القيمة ABS ..

مثال 16و16 تدريب7 صفحة 278.

6. جمل التحكم Control statements (Directives)

- تستخدم هذه الجمل بقصد التحكم بالأسمبلر وتوجيهه أثناء عملية ترجمة البرنامج المصدري .
- من أبرز جمل التحكم :

a) جملة نهاية البرنامج المصدري END

- تشكل هذه الجملة إشعاراً للأسمبلر لإيقاف عملية الترجمة بسبب الوصول إلى نهاية البرنامج .
- لابد من وجود مثل هذه الجملة في كل برنامج مصدري حتى تتوقف عملية الترجمة , في حال عدم وجودها فإن ذلك يؤدي إلى وجود خطأ برمجي .
- أنواع توجيهات النهاية : ENDSحيث تعني نهاية القطاع و ENDP وتعني نهاية الإجراء و END وتعني نهاية البرنامج بكامله .
- الصيغة العامة لتوجيهة END : END [Name] , حيث أن Name تعني اسم البرنامج أو نقطة بداية البرنامج . حيث أن اسم البرنامج الذي يلي توجيهة END هو اختياري
- أما بقية البرامج الثانوية فتحتوي على توجيهة END دون إتباعها باسم البرنامج .
- البرنامج الواحد ال279.من برنامج رئيس وعدة برامج ثانوية قد يحتوي على عدة توجيهات نهاية END لكن واحدة فقط من هذه التوجيهات تكون متبوعة بإسم البرنامج.
- انظر مثال 17 صفحة 279 .

b) توجيهة ORG

- تستخدم هذه التوجيهة لإخبار المترجم (Assembler) عن عنوان بداية منطقة الذاكرة المخصصة لتخزين البرنامج الهدفي الناتج من عملية الترجمة للبرنامج المصدري.
- يتم ذلك بتخصيص عنوان البداية إلى مسجل خاص بعنوان الموقع يطلق عليه اسم Location Counter (أي عداد المواقع أو العناوين)
- الشكل العام لتوجيهة ORG : ORG expression , حيث أن expression يمثل عنوان منطقة التخزين ' ويجب أن تؤول قيمته إلى ثابت عددي صحيح , ويمكن أن يحتوي expression على $.
- مثال: التوجيهة .ORG280 تعمل على تخزين البرنامج الهدفي الناتج من عملية الترجمة ابتداءً من عنوان 100H .أي 100 بالنظام السادس عشر.
- انظر مثال 18 صفحة 280 .
- انظر مثال 19 صفحة 281.

c) توجيهة EVEN

- تعتبر من التوجيهات نادرة الاستخدام
- عملها هو جعل محتويات عداد المواقع Location Counter قيمة زوجية في حالة احتواء هذا العداد على قيمة فردية .
- مثلاً لو فرضنا أن عداد المواقع يحتوي على القيمة 121H (أي أنه يشير إلى الموقع الذي يحمل عنوان 121H)، فعند استخدام توجيهة EVEN يتم إخبار الأسمبلر بأن يقوم بتخزين البيانات اللاحقة على العنوان 122H بدلاً من العنوان 121H .
- تكمن فائدة هذه التوجيهة مع أنواع الحواسيب التي تستخدم معالجات ميكروية 8086 أو 80286 والتي تستخدم 16 خطاً لنقل البيانات من و إلى الذاكرة , فاستخدام هذه التوجيهة مع مثل هذه الأنواع من المعالجات يزيد من كفاءة تنفيذ البرنامج حيث يمكن نقل محتويات 16 موقعاً ثنائياً من وإلى الذاكرة في عملية واحدة .

- هذا يتحقق عندما تكون البيانات التي سيتم نقلها تبدأ من عنوان زوجي .
- في حال نقل بيانات من مواقع ذات عناوين فردية فإن ذلك يستغرق وقت أطول .
- ينصح عند تخزين البيانات أن يتم ذلك على النحو التالي :
  استخدم DT أولاً ثم DD ثم DW وأخيراً DB .
- ينصح بتخزين البيانات في مواقع ذات عناوين زوجية عند التعامل مع معالجات من النوع 8086 أو 80286 أو 80386 وخاصة عند استخدام هذه البيانات في برامج خاصة بتطبيقات تحتاج إلى سرعة التنفيذ .
- في حالة احتواء القطاع على بيانات من النوع بايت فقط , ينصح باستخدام هذه التوجيهة قبل كل جملة من جمل تعريف البيانات باستثناء جملة التعريف الأولى كما في مثال 20 صفحة 283 .

d) توجيهة INCLUDE

- تستخدم هذه التوجيهة عند الحاجة لتضمين مجموعة من الجمل في أكثر من برنامج مصدري واحد .
- في مثل هذه الحالة تكتب مجموعة الجمل هذه منفصلة و تخزن على القرص المغناطيسي تحت اسم معين , بعد ذلك يمكن تضمين هذه الجمل إلى أي من البرامج المصدرية باستخدام توجيهة INCLUDE .
- فمثلاً: لو فرضنا أنه تم تخزين مجموعة من الجمل تحت الاسم Statements.Inc فإن الجملة INCLUDE Statements.Inc تشعر الأسمبلر بأن يقوم بنقل مجموعة الجمل المكونة للملف Statements.Inc إلى البرنامج المصدري الذي يحتوي توجيهة INCLUDE ، حيث يتم استبدال توجيهة INCLUDE بهذه الجمل في البرنامج المصدري .
- أما بالنسبة لعملية الترجمة فإنها تتم كما لو لم تكن توجيهة INCLUDE في البرنامج المصدري ، وتعامل مجموعة الجمل كأنها جزء من البرنامج المصدري .

**جمل الطباعة listing psudo-operatins

تستخدم للتحكم بشكل طباعة البرنامج

***جمل التحكم بطباعة البرنامج التوجيهات الاتية :**

1 **توجيهة تنظيم الصفحة (page):** تستخدم لتحديد عدد الاسطر وعدد الاعمدة في الصفحة تستخدم بالشكل الاتي :

عدد الرموز,عدد السطورpage (page 66,80)(page)توجيهتين متكافئتين

عند استخدام page عداد الاسطر ياخذ القيمة الدالة على عدد الاسطر ويتناقص بمقدار 1 الى ان يساوي الصفر ثم يتم الانتقال الى صفحة جديدة

2 **توجيهة +PAGE**

تستخدم للدلالة على بداية فصل (Chapter)جديد تؤدي الى تعديل قيمة عدادات الصفحات وزيادة عداد الفصول بمقدار 1

3 **توجيهة موضع البرنامج TITLE**

تستخدم لتحديد موضع البرنامج وطباعته في السطر الثاني من كل صفحة يجب ان يشمل موضع البرنامج على القرص المغناطيسي ومختصر عن طبيعة البرنامج ولا يزيد طول النص الوارد مع TITLE عن 60 رمز

4 **توجيهة الموضع الفرعي SUBTTL**

يتم عند استخدامها طباعة النص على السطر الثالث وتستخدم للدلالة على البرامج الثانوية SUBTIL SYNTAX ANALYSIS ROUTINES يتم عن هذه التوجيهة تعريف موضوع فرعي في وسط السطر الثالث

5 توجيهات .LIST و .XLIST و % OUT

عند مصادفة توجيهة (.XLIST)يتوقف الاسمبلر عن الكتابة الى ملف الطباعة تتم عملية الكتابة الى الملف عند وجود (.LIST) توجيهة % OUT تستخدم لاظهار اشعارات معينة على شاشة العرض تدريب 11 صفحة 287

****جمل الترجمة المشروطة Conditional Assembly Directives (pseudo-oper**

تهدف الى توجيه الاسمبلر الى استثناء جملة او مجموعة جمل في الترجمة وتحديد نهاية الجمل التي يمكن شمولها في حالة تحقق الشرط الوارد في التوجيهة او عدم تحققه

*تستخدم عبارة ENDIF وعدم استخدامها يؤدي الى خطأ برمجي يمكن استخدام ELSE مع جمل الترجمة المشروطة تحقق الشرط يؤدي الى ترجمة الجمل بين توجيهة الترجمة المشروطة وكلمة ELSEفي حالة عدم تحقق الشرط يتم معالجة الجمل الواقعة بين ELSE و ENDIF واستثناء الجمل الواردة بين توجيهة الترجمة المشروطة و ELSE من المعالجة بواسطة الاسمبلر في حالة عدم استخدام ELSE فان تحقق شرط الترجمة يؤدي الى معالجة جميع الجمل الواردة بين توجيهة الترجمة المشروطة و ENDIF

****انواع جمل الترجمة المشروطة :**

اولا: **توجيهة IF**

تاخذ الشكل العام :

IF expression ; condition

[ELSE] ; optional

ENDIF ; End Of IF

حيث اذا آلت قيمة expression الى قيمة غير الصفر فإن ذلك يعنى تحقق جواب جملة if,وبذلك يتم معالجة الجمل الواقعة بين if و Else في حالة استخدام else ويتم استثناء جميع الجمل المتبقية والواقعة بين ELSE و ENDIF واستثناء الجمل الواقعة بين IF و ELSE من المعالجة بواسطة الأسمبلر.

وفي حالة عدم استخدام ELSE يصبح شكل التوجيه على النحو IF expression

ENDIF

وفي مثل هذه الحالة فإن تحقق شرط الترجمة يؤدي الى معالجة جميع الجمل الواقعة بين IF و ENDIF من قبل الاسمبلر؛ وأن عدم تحقق الشرط(اي عندما تؤول قيمته الى الصفر أو False) فيتم استثناء الجمل الواقعة بينIF و ENDIF في المعالجة بواسطة الاسمبلر بالطبع اذا لم يتم استخدام عبارة ELSE ومثال 21 صفحة 290 يوضح ذلك

ثانيا:توجيهة IFE (IF Equal to Zero)

تؤدي هذه التوجيهة الى شمول الجمل الواقعة بين IFE و ENDIF (في حالة عدم استخدام الخيار ELSE) أو الجمل الواقعة بين IFE وELSE (في حالة استخدام الخيار ELSE) بالمعالجة من قبل الأسمبلر؛ عندما تؤول قسمة expression الى الصفر تدريب 22

ثالثاً: توجيهة IFI

تعطي هذة الجملة القيمة True اذا كان الاسمبلر يعمل في الجولة الاولى(first pass)؛ وإلا فإن نتيجة هذه التوجيهة تعطي القيمة False (اي ان الشرط غير محقق)

*وتبرز فائدة هذة التوجيهة عند استخدام بعض الماكرو تتم عادة خلال الجولة الاولى(first pass) من جولات الاسمبلر أثناء عملية الترجمة

*حيث يتم استخدام مكتبة الماكرو والمسماه MylipMacومعالجتها خلال الجولة الاولى للأسمبلر؛ وفي الجولة الثانية (Second pass) لا توجد أي حاجة لإستدعاء مكتبة الماكرو مرة اخرى؛ وهذا بالطبع يوفر الكثير من الوقت أثناء عملية ترجمة البرنامج المصدري إلى برنامج هدفي.

رابعاً: توجيهة IF2

تشبه هذه الجملة IFI باستثناء أن تحقق الشرط يتم في الجولة الثانية من جولات الاسمبلر أما في الجولة الاولى فإن نتيجة استخدام هذه التوجيهة يعطي القيمة False ولذلك يتم استثناء الجمل الواقعة بين IF2 و ENDIF من عملية المعالجة بواسطة الاسمبلر

خامساً: توجيهة IFDEF (IF symbol Defined)

تستخدم هذه التوجيهة لفحص ما اذا كان اسم معين قد عرف في البرنامج أم لا.

*حيث تؤدي نتيجة الفحص إما إلى أن يقوم الاسمبلر بمعالجة الجمل الواردة بين IFDEF وENDIF (وفي حالة عدم

* استخدام الخيار ELSE) او تلك الواردة بين IFDEF وELSE فقط (في حالة استخدام الخيار ELSE) في حالة تحقق الشرط (اي عندما يكوم الإسم قد عرف)أو عدم المعالجة في حالة عدم تحقق الشرط

وفي حالة كون الإسم قد عرف مسبقا تؤدي نتيجة الفحص إلى تحقق الشرط.(اما في حالة عدم تعريف الإسم فإن نتيجة فحص الشرط لن تحقق أي النتيجة هي القيمة المنطقية false. مثال 23 صفحة 293 وتدريب 12 صفحة 293

سادساً: توجيهة IF SYmbol Not Defibd(IFNDEF)

تشبه هذه الجملة الجملة السابقة باستثناء أن عدم تعريف الاسم يؤدي الى تحقق الشرط وبذلك يتم شمول الجمل الواقعة بين IFNDEF وENDIF للمعالجة من قبل الأسمبلر. اما اذا كاه الاسم معرفا فان نتيجة هذه التوجيهة تكون القيمة المنطقية False وهذا يعني استثناء الجمل الواردة بين IFNDEF و ENDIF من المعالجة من قبل الأسمبلر. بالطبع يتم ذلك عند عدم استخدام الخيار ELSE. اما اذا استخدم الخيار ELSE فإن تحقق الشرط(إي عندما يكون الاسم غير معروف)يؤدي الى معالجة الجمل الواقعة بين IFNDEF و ELSE واستثناء الجمل الواقعة بين ELSE و ENDIF من عملية المعالجة بواسطة الاسمبلر.

سابعاً : توجيهة IFIDN (If Identical)

*حيث ان الأقواس على شكل زاوية ضرورية , وتستخدم الفارزة للفصل بين معاملي التوجيهة <Arg2> و <Arg1> وممكن ان يكونا اسماء أو اعداد أو تعابير.

*وتستخدم هذه التوجيهة داخل الماكرو , وفي حالة تساوي المعاملين Arg1 و Arg2 فإن الأسمبلر يقوم بمعالجة الجمل الواقعة بين IFIDN و ENDIF (حيث يكون شرط التشابه بين المعاملين قد تحقق) أما اذا لما يتساوى المعاملين فإن الأسمبلر سوف يستثني هذه الجمل من المعالجة. (مثال 24 ص294)

ثامناً : توجيهة IFDIF (If Different)

حيث ان الأقواس والفارزة كما في التوجيهة السابقة ومعنى هذه التوجيهة انه عند اختلاف المعاملين Arg1 و Arg2 يقوم الأسمبلر بمعالجة الجمل الواقعة بين IFDIF و ENDIF , أما اذا تشابه Arg1 و Arg2 فستستثنى الجمل الوقعة بين IFDIF و ENDIF من المعالجة بواسطة الأسمبلر أثناء عملية ترجمة البرنامج المصدري الى برنامج هدفي.

وتستخدم ايضاً مع الماكرو وذلك لمقارنة معاملات الماكرو مع أسماء أو قيم معينة للتأكد فيما اذا كانت هذه القيم مساوية لمعاملات الماكرو (Macro Parameters)

تاسعاً : توجيهة IFB هناك توجيهات أخرى للترجمة المشروطة والمستخدمة خاصة مع الماكرو ,أبرزها:

IFB <Argument> - والتي تعني اذا كان المعامل (Argument) فارغاً (Blank) يقوم المترجم بمعالجة الجمل الواقعة بين IFB و ENDIF والا سيتم استثناء هذه الجمل من الترجمة .

IFNM < Argument> - والتي تعني اذا كان المعامل (Argument) فارغاً فإنه سيتم معالجة الجمل الواقعة بين IFNB و ENDIF والا سيتم استثناء هذه الجمل من المعالجة.

عاشراً : توجيهة EXITM EXITM – وتعمل هذه التوجيهة على إشعار المترجم بالتوقف عن الإستمرار بعملية معالجة الماكرو والتي يطلق عليها (Macro expansion) عند مصادفة هذه التوجيهة في البرنامج وينتقل التحكم إلى الجملة التالية لنهاية الماكرو, حيث ان توجيهة ENDM تعني نهاية الماكرو كما أن ENDP تعني نهاية الإجراء (Procedure).

إجراء Procedure مجموعة من اجمل لغة اسمبلي تعرف مرة واحدة في البرنامج وتستخدم عدة مرات. وتعدد توجيهة PROC بداية الاجراء و تحدد تو جبهة ENDP نهايته . بوفر استخدام الإجراء الجهد والوقت في عملية البرمجة ، والإجراء هو نوع من انواع البرامج الفرعية الي تشمل الإقترانات و الماكرو أيضا .

برنامج الربط Linker وهو برنامج مساعد باني مع المترجم يجب تنفيذه على البرنامج الهدفي الناتج من المترجم, حيث أنه بدون تنفيذ هذا البرنامج يصبح من غير الممكن تنفيذ البرنامج الهدفي الناتج من المترجم مباشرة، وبالنسبة لوظائف هذا البرنامج وطبيعة عمله تعالج في مكان اخر من هذا المقرر .

Directive (Pseudo-operation) توجيهة نوع من جمل لغة التجميع (أسمبلي) وظيفتها إخبار المترجم (الأسمبلر) بما يجب عمله أثناء ترجمة البرنامج المصادري، وهي موجهة للمترجم وليس للمعالج.

قطاع أو مقطع Segment جزء من الذاكره يصل حجمه إلى 64 كيلو بايت ، ويوجد أربعة أنواع من القطاعات التي يتكون منها برنامج لغة اسمبلي في قطاع التعليمات وقطاع البيانات وقت المكدس والقطاع الإضافي .

المترجم (الأسمبلر) (Assembler)Translator يطلق على البرنامج الذي يقوم بترجمة برنامج لغة اسمبلى من لغة المصدر إلى الألة وغالبا ما يؤدي عمله على مرحلتين أو جولتين (Two passes) .

الوحدة السادسة :طرق العنونة وطاقم التعليمات الأساسية

*تخزن البيانات الثنائية المستخدمة في الحاسوب إما في ذاكرته أو في مسجلات المعالج.

*فالبيانات الثنائية المخزنة في المسجلات قد تمثل بينات يحتاجها المعالج لإجراء العمليات الحسابية عليها , أو تستخدم بغرض التحكم بسير عمل المعالج نفسه .

*وبالنسبة لبيانات التحكم بالمعالج فهي عبارة عن ثنائية أو مجموعة من الثنائيات تستخدم بغرض تحديد مجموعة الأوامر اللازمة (حسب تتابع معين) لمالجة البيانات المخزنة في المسجلات الأخرى.

*سنركز جل همنا علي البيانات الرقمية وغير الرقمية الممثلة داخل ذاكرة الحاسوب أو مسجلاته باستخدام النظام الثنائي , ويتم معالجة هذه البيانات عن طريق وحدة الحساب والمنطق للحصول على النتائج المرجوة من هذه المعالجة.

****يمكن تقسيم انواع البيانت في ذاكرة الحاسوب إلي ثلاثة أقسام رئيسة هي:**

1.البيانات الرقمية المستخدمة في العمليات الحسابية

2.الرموز الأبجدية العددية(Alphanumeric characters)

3.الرموز الأخرى المستخدمة لأغراض خاصة مثل رموز العمليات الحسابية والاقواس وما إلى ذلك.

*ويتم تمثيل هذه الأنواع ,باستثناء البيانات الرقمية في ذاكرة الحاسوب ومسجلاته باستخدام إحرى طرق الترميز المختلفة , حيث يُمثل كل رمز من هذه الرموز أو حرف من الحروف بمجموعة من الثنائيات تعتمد على نظام التمثيل المستخدم وعلى الرمز أو الحرف نفسه.
والسبب الرئيس في التمثيل الثنائي للبيانات هو أن المسجل يتكون من مجموعة من النطاطات او القلابات (Flip-Flops) وان القلابة الواحدة لها القدرة على تخزين الخانة الثنائية 0 او 1

*مع ان النظام الثنائي يعتبر الامثل للاستخدام بواسطة الحاسوب , الا ان الكثير من المستخدمين الذين اعتادوا على التعامل مع النظام العشري يفضلون التعامل مع هذا النظام هذا النظام العددي على تعلم نظام عددي جديد (مثل النظام الثنائي).

*لذلك نجد ان معظم الحواسيب يمكنها التعامل مع مجموعة مختلفة من الانظمة العددية اضافة الى النظام الثنائي مثل النظام الثنائي والثماني والسادس عشري.

*فمن ميزات النظام الثماني انه يمكن تمثيل كل ثلاثة ثنائيات من العدد الممثل بالنظام الثنائي بخانة واحدة باستخدام النظام الثماني.

والسبب في ذلك ان(8=2^3)

*اما بالنسبة للنظام السادس عشري فهو ايضا من الانظمة المنتشرة , والسبب في ذلك أن الخانة الواحدة في النظام السادس عشري يمكن ان تمثل أربع خانات متجاورة من النظام الثنائي بسبب ان.(16=2^4)

***طرق العنونة Addressing Modes**

* لتوضيح مبدأ عمل هذه الطرق سنستخدم التعليمة MOV وذلك لسهولتها ، والصيغة العامة لهذه التعليمة هي : MOV dest , source ، حيث أن :

- source يمثل الحقل الخاص ببيانات المصدر (أي البيانات التي سيتم عمل نسخة عنها وتخزينها في الموقع المعنون بالعامل dest).

-dest : يمثل الحقل المستقبل لهذه النسخة من البيانات (أي يمثل عنوان موقع استقبال البيانات التي سيتم عمل نسخة عنها والمعنونة بالعامل source). * انظر مثال 14 صفحة 341

(1) العنونة الفورية Immediate Addressing

* حسب هذه الطريقة من طرق العنونة تكون البيانات المطلوب اجراء العمليات عليها مخزنة في نفس التعليمة .

* ويمكن أن يكون طول هذه البيانات 8 ثنائيات (بايت) أو 16 ثنائية (كلمة). * انظر مثال 15 & 16 صفحة 342 تدريب 16

(2) العنونة باستخدام المسجلات Register Addressing

* باستخدام هذه الطريقة لعنونة المصدر فإن المعالج يبحث عن البيانات في أحد مسجلاته ،أما في حالة استخدام هذه الطريقة لعنونة الموقع المستقبل للبيانات فهذا يعني أن هذا الموقع سيكون أحد مسجلاته المبينة في التعليمة .

* وفي مثل هذه الحالات ليس ضرورياً الرجوع إلى الذاكرة للبحث عن البيانات أو تخزينها . * انظر مثال 17 صفحة 343

(3) العنوان الفعال للبيانات Data Effective Address

* تختلف طرق العنونة عن بعضها البعض في طريقة حساب عنوان الموقع المطلوب .

* يتكون البرنامج المصدري في لغة أسمبلي من عدة قطاعات هي :

- قطاع البرنامج Code Segment
- قطاع البيانات Data Segment
- قطاع المكدس Stack Segment
- القطاع الاضافي Extra Segment

*العنوان الفعال للبيانات : هو بعد موقع الذاكرة الذي يحتوي على هذه البيانات من بداية القطاع الذي يحتوي على هذا الموقع .

* يستخدم العنوان الفعال في حساب العنوان الحقيقي Physical Address للبيانات .

* تختلف طريقة حساب العنوان الفعال باختلاف أسلوب العنونة المستخدم .

* لبيان طريقة حساب العنوان الحقيقي : نفرض أن البيانات مخزنة في قطاع البيانات وأن عنوان بداية هذا القطاع هو 034FH ، حيث يخزن هذا العنوان مخزناً في مسجل قطاع البيانات DS . كذلك نفرض أن العنوان الفعال للبيانات هو 32 بايت من بداية القطاع .

* من هذه المعطيات يتم حساب قيمة العنوان الحقيقي بإضافة قيمة العنوان الفعال إلى القيمة المخزنة في مسجل قطاع البيانات DS بعد ضربه بالعدد 16 وذلك للحصول على عنوان مكون من 20 ثنائية كي يتطابق هذا العدد مع عدد خطوط ناقلة العنوان ال 20 ، لذلك يضاف الرقم 0 إلى أقصى يمين القيمة 034FH المخزنة في المسجل DS وهذا مكافئ لضرب القيمة بالعدد 16 ومن ثم تتم العملية على هذا النحو : $(03522)_{16}$ = 00032 + 034F0 والناتج هو قيمة العنوان الحقيقي للبيانات المطلوبة تدريب 344

(4) عنونة البيانات المخزونة في الذاكرة

* طرق عنونة البيانات المخزونة في الذاكرة تختلف عن بعضها البعض في طريقة حساب العنوان الفعال ، ومن هذه الطرق :

1) العنونة المباشرة Direct Addressing
   - في هذا النوع يقع العنوان الفعال لموقع الذاكرة المطلوب الوصول اليه ضمن التعليمة نفسها .

- يكتب العنوان في التعليمة على شكل اسم لموقع الذاكرة الذي نرغب في الوصول إليه .
- عند ترجمة التعليمة التي تستخدم هذا النوع من العنونة ،يقوم المترجم بحساب العنوان الفعال على شكل قيمة صحيحة تمثل بعد هذا الموقع من بداية القطاع الذي يحتوي على الموقع .
- يطلق على هذه القيمة مقدار الإزاحة offset .
- في العنونة المباشرة يكون العنوان الفعال مساوياً لقيمة الإزاحة. *انظر مثال 18 صفحة 345 .تدريب 18

2) العنونة غير المباشرة باستخدام المسجلات Regeister Indirect Addressing

- في هذا النوع يكون العنوان الفعال مخزناً في أحد المسجلات التالية : مسجل الأساس BX، مسجل مؤشر الأساس BP، ومسجل فهرسة المصدر SI، ومسجل فهرسة حقل البيانات DI.
- لتمييز العنونة غير المباشرة عن غيرها يوضع اسم المسجل الذي يحتوي على العنوان الفعال داخل أقواس مربعة "[]" انظر مثال 19 صفحة 347 . انظر مثال 21 صفحة 349 . انظر مثال 20 صفحة 348 . تدريب 19
- يمكن استخدام العنونة غير المباشرة مع الحقل الاول (أي الجهة المستقبلة للبيانات) .

3) العنونة باستخدام الأساس Based Addressing

- يتم حساب قيمة العنوان الفعال عن طريق جمع محتوى مسجل الأساس BX أو BP ومقدار الإزاحة (displacement) المخزنة ضمن التعليمة نفسها .
- تستخدم 8 خانات أو 16 خانة ثنائية لتخزين قيمة الإزاحة ، وتمثل هذه القيمة على أنها ثنائية بدون إشارة أي أن قيمة الإزاحة قد تصل إلى $2^{16} - 1$ أي إلى 65535 .
- عند استخدام المسجل BP لتخزين عنوان الأساس فهذا يعني أن القطاع المعني هو قطاع المكدس (STACK) وليس قطاع البيانات كما هو الحال عند استخدام المسجل BX لهذا الغرض .تدريب 20 صفحة 350
- من فوائد هذه الطريقة من طرق العنونة استخدامها في معالجة مجموعة من البيانات المخزونة في الذاكرة على شكل سجل، حيث يخزن عنوان الأساس أي بداية السجل في مسجل الأساس، ومن ثم يمكن الوصول إلى أي عنصر من عناصر السجل بإضافة بُعد هذا العنصر من بداية السجل إلى محتوى الأساس . انظر شكل 3 صفحة 350 .

4) العنونة باستخدام الفهرس(Indexed Addressing)

*يوجد تشابه الى حد كبير بين هذه الطريقة والطريقة السابقة عند حساب العنوان الفعال والإختلاف في استخدام احد مسجلات الفهرسة SI أو DI بدلا من المسجل الأساس BX ومؤشر الأساسBP .والتعليمة الآتية تمثل النونة باستخدلم الفهرس في حقل المصدر: MOV AL,[SI]ARRAY

*في هذا المثال يمثل الجزء ARRAY من عنوان حقل المصدر إزاحة مباشرة (Direct Disaplacement)

*يمثل السجل SI مسجل الفهرسة

* يتم في هذه الحالةحساب العنوان الفعال بإضافة بعد الموقع المعنون بالإسم ARRAYإلى محتوى المسجل SI .

*لذلك نجد أن اسم مسجل الفهرسة يجب أن يظهر داخل قوسين مربعين .

*وبالنسبة للعنوان الحقيقي في هذا المثال فيتم حسب العلاقة الاتية: العنوان الحقيقي=(DS)*16+(SI)+Array

Array :يمثل إزاحة الموقع المعنون بالإسم ARRAY من بداية قطاع البيانات

،(SI): محتوى مسجل فهرسة المصدر،(DS):محتوى مسجل قطاع البيانات *مثال صفحة 351

*ان وضع إسم المسجل داخل قوسين يعني أن المقصود هو محتوى المسجل **شكل(4)العنونة باستخدام الفهرس

*وتجدر الإشارة هنا أن هذه الطريقة من أكثر الطرق استخداما مع المصفوفة ذات البعد الواحد *تدريب 21 صفحة353.

*اذا رغبنا بالوصول الى جميع عناصر المصفوفة يبدأ المسجل SI بالقيمة 0 كي نبدأ مع العنصر الأول من عناصر المجموعة

*للانتقال للعنصر التالي يتم زيادة محتوى المسجل SI بمقدار 1 اذا كانت المصفوفة من النوع Byte وبمقدار 2 اذا كانت من النوعWord

5) العنونة باستخدام الأساس والفهرس(Based Indexed Addressing)

هذه الطريقة هي نتيجة ضم العنونة باستخدام الأساس والعنونة باستخدام الفهرس ولذلك تعتبر اكثر قوة من الطريقتين السابقتين مثال صفحة 353

*حساب العنوان الفعال يتم عن طريق جمع محتوى مسجل الأساس (BXأوBP) ومحتوى مسجل الفهرسة (DIأوSI)ومقدار الإزاحة إن وجدت ضمن التعليمات

*يمكن تمثيل حساب العنوان الفعال والعنوان الحقيقي من التعليمة كما في الشكل 5 صفحة 354

*أما في حالة عدم وجود إزاحة في التعليمة , فيتم إيجاد قيمة العنوان الفعال بجمع محتوى السجل الأساس مع محتوى مسجل الفهرس فقط

*يستفاد من هذا النوع من العنونة في معالجة الجداول ذات البعدين(two dimensional arrays) حيث يستخدم مسجل الأساس لحفظ عنوان بداية الجدول في الذاكرة ويستعمل كل من مسجل الفهرس والإزاحة للدلالة على عناوين الأسطر والأعمدة في الجدول أثناء المعالجة *مثال22 صفحة 354 مثال 32. تدريب 22 صفحة 255.

(4) عنونة سلاسل الرموز (String Addressing)

*تستعمل مسجل فهرسة المصدر ومسجل فهرسة ضمناً دون ذكر هذين المسجلين في التعليمة .

*سمي المسجل SI بهذا الاسم لإستخدامه ضمنا للدلالة على إزاحة العنصر التالي في السلسلة الرمزية المصدرية

*سمي المسجل DI بهذا الاسم لاستخدامه ضمنا للدلالة على إزاحة العنصر التالي في السلسلة الرمزية المستقبلة للبيانات

*لذلك فإن التعليمة MOVSB يستخدم المسجلين SI و DI ضمنا دون أن يعلن عنهما في التعليمة

(5)عنونة منافذ الإدخال والإخراج(Input/Output Port Addressing)

*تستخدم مع تعليمة الإدخال (IN) وتعليمة الإخراج(OUT) حيث يحدد عنوان منفذ وحدة الإدخال أو الإخراج ضمن التعليمة نفسها أو يشار إليها بواسطة المسجل DX

في الحالة الأولى يطلق عليها العنونة المباشرة ويمكن لهذه الطريقة عنونة 256 منفذ (أي من 0 إلى 255)

أما في الحالة الثانية فيمكن عنونة 16^2 (أي من 0 إلى 65535) وذلك بتخزين عنوان المنفذ في المسجل DX قبل تنفيذ العملية مثال 24 صفحة 356

*طاقم التعليماتInstruction Set :*لقد تم تقسيم طاقم التعليمات إلى ستة أنواع رئيسية:

1)الصيغة العامة لتعليمات لغة اسمبلي (Instruction Format)

*نود بيان الصيغة العامة لتعليمات هذه اللغة كي يتسنى لك معرفة مدلول اي تعليمة من التعليمات وتصبح قادراً على فهم بعض التعليمات المختلفة ومن ثم استخدامها في إعداد برامج الحل مسائل متنوعه.

*تتكون صيغة التعليمية من تعليمات لغة اسمبلي من أربعة حقول على الأكثر كما هو موضح في الشكل العام:

[Label:] Mnemonic [Operand(s)] [;Comment]

حيث أن:

Lable: تمثل حقل وسم التعليمة،والنقطتين دوماً تلي العلامه،ووجود هذا الحقل داخل قوسين مربعين يعني أن استخدامه اختياريا وحسب الحاجة إليه.

Mnemonic: ويمثل رمز العملية المطلوبة ووجوده في التعليمه دوماً اجباريا وإلا فإنه لن يكون لدينا تعليمة.

Operand (s): ويمثل حقل أو حقول البيانات ووجوده ضمن التعليمة أو عدمه سواء وكذلك عدد المعاملات إن وجدت يعتمد دوماً على نوع العملية.

Comments: وتمثل حقل الملاحظات وتأتي بعد فارزة منقوطة وهو حقل اختياري يعتمد وجوده أو عدمه على مدى حاجة المبرمج لذلك.

*وتفصل الحقول عن بعضها البعض بفراغ أو أكثر. والمثال التالي يبين هذه الحقول:

حقل الملاحظات حقل البيانات رمز العملية حقل الوسم Get-Count Mov Cx, si ;Initialize Count

*توضيح أكثر لكل جزء من الأجزاء المكونة لتعليمة لغة اسمبلي:

1. حقل الوسم(Lable Field)

*يهدف هذا الحقل إلى تمييز التعليمة الموسومه عن التعليمات الأخرى ، كما هو الحال مع رقم السطر في لغة بيسك، حيث يتم وضع وسم خاص للتعليمة كي يتم الانتقال إليها عند الحاجة من قبل تعليمات أخرى داخل البرنامج .*ومن غير الجائز تكرار نفس الوسم في البرنامج حيث أن ذلك يؤدي إلى الالتباس وعدم معرفة أي منها هو المقصود .*يعتمد عدد الرموز المسموح بها ونوعها عند تكوين الوسم على نمط الاسمبلر المستخدم. وفي .*بعض أنواع الاسمبلر قد يصل طول الوسم إلى 31حرفا مع ملاحظة أن الوسم يجب أن يكون متبوعا بإشارة شارحه(colon)":" ***ويمكن استخدام الآتي في تكوين الوسم:**

-الحروف الأبجدية (A-Z) أو(a-z)

-الارقام الحسابية عربية الأصل(0-9)

-بعض الرموز الخاصة مثل؟ ,.,_,$,...الخ

*ومن غير الجائز أن يبدأ الوسم برقم.وفي حالة استخدام النقطه».» ضمن الوسم يجب أن تكون الحرف الأول من أحرف الوسم .*عدم صحة وجود فراغ

الفراغ ضمن أحرف الوسم وعدم جواز استخدام أي من التعليمات أو التوجيهات أو الكلمات المحجوزة المستخدمة في لغة الاسمبلر كوسما

2.حقل العملية Mnemonic Filed:

يمثل هذا الحقل نوع العملية التي سينفذها الحاسوب حيث يطلق عليه في بعض الأحيان رمز العملية (Operation code)

*ويتكون من الحروف الأبجدية الانجليزيه فقط حيث يتراوح عدد الاحرف الممثله لهذا الحقل بين حرفين وستة أحرف . لقد ورد معنا في سياق طرق عنونة البيانات رمز العملية mov حيث يتكون هذا الرمز من ثلاثة أحرف . بالإضافة إلى هذا الرمز توجد مجموعة كبيرة من التعليمات وتشكل بمجملها موضوع الأقسام اللاحقة.

*فإن نوع التعليمة يحدد عدد معاملاتها واللازمه لتنفيذها .فعلى سبيل المثال تحتاج التعليمة mov الى معاملين.

3. حقل البيانات (المعاملات)(Operand)(s)(Field)

*يبين هذا الحقل للمعالج مكان وجود البيانات اللازمة لإجراء العمليات، حيث يقوم المعالج بدوره بإحضار هذه البيانات تمهيدا لإجراء العملية اللازمة . وعلى سبيل المثال فإن التعليمه: MOV AL;15

*تعني أن العملية هي عملية تحريك للبيانات وأن مثل هذه العملية بحاجة إلى معاملين (أو قيمتين) .فالمعامل الأول يمثل المعامل المستقبل للبيانات (destination operand) وهو المسجل AL ويستخدم طريقة العنونة باستخدام المسجل AL ،والمعامل الثاني يمثل معامل المصدر (source operand) ويستخدم اسلوب العنونة الفوريه.

*يعتمد وجود مثل هذه المعاملات على طبيعة التعليمة ، لذلك نجد أن بعض التعليمات بحاجة إلى معاملات وبعضها الآخر يكون فيها المعامل مضمنا ومن غير الجائز التصريح باسمه علنا ضمن التعليمة.*وفي حالة استخدام حقل البيانات في التعليمة فإما تكون الحاجة إلى معامل واحد أو معاملين على الأكثر.*وفي حالة حاجة التعليمة إلى معاملين يتم الفصل بيهما باستخدام الفارزة "," . ويمثل المعامل الأول (والواقع الى جوار رمز التعلية) الحقل المستقبل للبيانات أو النتائج (destination operand)لكثير من التعليمات ،والمعامل الثاني يمثل معامل المصدر.(source operand)

*كذلك يلاحظ في مثل هذه الحالة أن قيمة معامل المصدر أن تتغير بينما نجد أن قيمة معامل المصدر لن تتغير بينما نجد أن قيمة المعامل المستقبل قد تتغير نتيجة لتنفيذ التعليمة وحسب طبيعة التعليمة المستخدمة.

4.*حقل الملاحظةComment Field

*يستخدم هذا الحقل لتوضيح عمل البرنامج حيث يمكن كتابة عدة أسطر من الملاحظات في بداية البرنامج تبين هدف البرنامج وكاتبه وتاريخ إعداده والتعديلات التي تمت عليه وتاريخها وما إلى ذلك من معلومات ضرورية للمبرمج .*كذلك يمكن استخدام حقل الملاحظة إلى ما جوار ما هي عمل التعليمة وليس على كيفية عملها التعليمات لبيان ما تؤديه هذه التعليمات عند كتابة الملاحظة .*وفي هذه الحالة يجب التركيز على ماذا تعمل التعليمة وليس على كيف تعمل.*يعتبر جزء الملاحظات غير تنفيذي ولا يتم معالجته من قبل المعالج، فوجوده أو عدمه لن يؤثر اطلاقا على عمل المعالج الدقيق .*ولكي يتعرف الاسمبلر على ملاحظات البرنامج يجب أن تسبق الملاحظة إشارة الفارزة المنقوطة;

2)تعليمات نقل البيانات :

تهدف هذه التعليمات الي نقل البيانات اما بين المسجلات نفسها ابو بين المسجلات ومواقع الذاكره ، وكذلك نقل العنوان الي احد مسجلات العنوان مثل DS وES، وتقسم هذه التعليمات الى اربع مجموعات :

1-التعليمات عامة الغرض : وتشمل خمس تعليمات

١-**تعليمة تحريك البيانات** MOV :تهدف هذه التعليمات الي نقل بيانات المصدر الي الحقل المستقبل لها وتستخدم لنقل محتوى مسجل الي مسجل اخر او نقل بايت او كلمة من مسجل الى موقع في الذاكره او العكس او نقل قيمة فورية الي مسجل او موقع في الذاكره . وتجد الاشارة الي ان طول حقل المصدر يجب ان يكون مسآويا لطول حقل استقبال البيانات . وهذا الطول يمكن ان يكون من النوعByte او wordوالصيغة العامة التعليمية هي: Mov dest , source

وفيما يلي بعض الامثلة على استعمال هذه التعليمة :

هذه التعليمة تنقل محتوى المسجل CL وطوله بايت الي المسجل AL وطوله بايتAL,CL;MOV

تنقل محتوى المسجل BX الي موقع الذاكره المعنون بالمسجلAX [AX] , BX; MOV

تنقل محتوى المسجل AX الي موقع الذاكره المعنون باسم TABLE TABLE ,AX; MOV

تنقل مختوى موقع الذاكره المعنون بالاسمTABLE الى المسجل AX AX, TABEL ; MOVE

تنقل القيمة الثابته 20 بالنظام العشري الي المسجل20 ,SI MOV SI ;

وتنقل القيمة الثابته 16 (20) الى موقع الذاكره المعنون بالاسمTABEL MOV TABLE , 20H;

**وعند استعمال هذه التعليمات يجب مراعاه ان تكون حقول مصدر واستقبال البيانات من نفس الطول . لذلك تعتبر التعليمية التالية غير سليمةMOV AL , BX

*والسبب في ذلك ان طول مصدر البيانات) اي المسجل(BX مساويا كامة وطول حقل مستقبل البيانات) اي المسجل (AL مساويا بايت.

اضافة الي ذلك يستثنى من تعليمة تحريك البيانات الحالات الاتية:

أ. نقل قيمه فوريه الى مسجلات القطاعات مباشره حيث يمكن انجاز ذلك باستخدام مسجل عام الغرض توسيط ومن ثم انقل محتويات هذا المسجل الي مسجل القطاع. وعلى سبيل المثال من غير الجائز نقل بدايه قطاع البيانات المعرف على النحو الاتي:

DSEG DATA

DSEG ENDS

بإستخدام التعليمة: MOVE DS, DSEG

ويمكن انجاز ذلك باستخدام التعليمتين التالتين MOV AX, DSEG

MOV DS, AX

حيث استخدم المسجل AX عاملا وسيطا.

ب. نقل محتوى مسجل من مسجلات القطاع الي مسجل اخر مباشره ويمكن انجاز ذلك باستخدام احد مسجلات عامه الغرض كوسيط كما في الحاله السابقه. على سبيل المثال من غير الجائز نقل محتويات المسجلES الى محتوى المسجلDS عن طريق تعليمة تحريك البيانات مباشره على النحو: MOV DX, ES

*وتعتبر هذه التعليمات غير سليمه ويمكن انجاز ذلك باستخدام المسجل AXعلى سبيل المثال على النحو الاتي MOV AX, ES
MOV DX, AX

*وتأثير هاتين الجملتين جعل مسجل القطع الاضافي ESومسجل قطع البيانات ويشير الى نفس قطع الذاكره

ج. نقل محتوى موقع الذاكره الرئيسيه الى موقع اخر مباشره فعلى سبيل المثال من غير الجهاز استعمال التعليمه: MOV A , B

لنقل محتويات موقع الذاكره المعنون بالاسم Bالى الموقع الاخر المعنون بالاسم A ويمكن انجاز مثل هذا العمل عن طريق استخدام مسجل عام الغرض كوسيط لذلك على النحو التالي:

MOV AX, A

MOV B, AX

*يتم تخزين محتويات الموقع المعنون بالاسم Aفي المسجل AX ومن ثم يتم تحديد محتويات المسجلAX الى موقع الذاكره المعنون بالرمزB

. 2القيود المفروضة على استخدام التعليمة MOV وكيفية التغلب على بعضها.

(2تعليمة دفع البيانات الي المكدس PUSH

تستخدم هذه التعليمه لتخزين محتوى بعض المسجلات ومواقع في الذاكره في قطاع المكدس تخزين مؤقتا الصيغه العامه لهذه التعليم هي:

POUH source

حيث يمثل source عنوان مصدر البيانات ويجب ان يكون من المسجلات او المواقع التي طولها كلمه اي 2 بايت

*في الامثله الاتيه توضح استخدام هذه التعليمات:

دفع متحوى المسجل AX على المكدس POUH AX

دفع محتوى الموقع A من النوع word على المكدس ;POUH A

تتلخص عمل التعليمه POSHفي الخطوات التاليه علي فرض ان محتوى مصدر البيانات هو 16(AABB)

-ينقص محتوى مؤشر المكدس (Stack pointer)SPبمقدار 1

-يدفع بالبايت ذو العنوان الاعلى اي قيمه16 (AA)الى موقع قطاع المكدس المعنون بالمؤشر spنسبه الى بدايه قطاع المكدس

-ينقص محتوى مؤشر المكدس مره ثانيه بمقدار 1

-يدفع بالبايت ذو العنوان الادنى من مصدر البيانات اي القيمه)16)BBالى موقع المعنون بالمؤشرSP نسبه الى بدايه قطاع المكدس

لذا نرى ان مؤشر المكدس دوما يشير الى اخر عنصر تمت اضافته الي المكدس والمهم بالنسبه للمبرمج هو تعليمهPOUH تنقص قيمه مؤشر المكدس بمقدار 2 اولا ثم تدفع بالمحتوى مصدر البيانات الى الموقع (نسبه الى بدايه قطع المكدس)

*من غير الجائز دفع قيمة فورية الى المكدس.اي ان التعليمة الاتية غير سليمة: POUH 105 ويلاحظ كذلك ان تعليمة POUH لا تؤثر على محتوى مصدر البيانات.

*على فرض ان محتوى sp قبل التفيذ هو (02BE) وان محتوى المسجل AX هو16(AABB)

3)تعليمة استرجاع البيانات على المكدس POP

تعمل هذه التعليمة عكس عمل التعليمة POUH فبدلا من الخزن على المكدس تسترجع العنصر مخزن في قيمه المكدس المشار اليه بالمسجل SPنسبه الى بدايه القطاع

ويخزن العنصر المسترجع في حقل مستقبل البيانات ومن ثم يتم تعديل قيمة المسجل SP بحيث تضاف اليه القيمة 2 بدلا من تنقيص القيمة 2 كما في الحال مع التعليمه POUHفيما عدا ذلك ان ما ينطبق على التعليمة POUH ينطبق ايضا على هذه التعليمة

والصيغة العامة لهذه التعليمة هي POP destination :

حيث ان destination يمثل حق المستقبل البيانات وبعد تنفيذ هذه التعليمات قد يطرا تغيير على محتوى حقل مستقبل البيانات كما هو الحال مع حقل المستقبل البيانات في التعليمات الاخرى

إفرض على سبيل المثال ان التعليمة POP BX :

قد تم تنفيذها بعد تنفيذ التعليمة PUSH AX :

لذلك فإن المسجل BX سوف يخزن القيمة 16(AABB)

ومن ثم يتم اضافة القيمة 2 الى محتوى المسجل SP حيث تعود محتوياته الي ما كانت عليه قبل تنفيذ التعليمة POUH AX :

اما بالنسبة لمحتوى الموقع SS: 02BC فلن يطرأ عليه اي تعديل تدريب 23 صفحة 264

4)**تعليميه المبادلة XCHG** الصيغه العامه لهذه التعليمية هي: XCHG operand1operand2,

وتعمل هذه التعليم على مبادلة قيم المعاملينoperand2 ، operand1مع بعضهم البعض لذلك يعتبر كلها زين المعامله بين حقل مرسلا للبيانات ومستقبل لها في نفس الوقت.

لذلك من غير الجائز ان يكون اي من هذين الحقلين قيمه فوريه او ان يكون كليهما عنوانا لموقع في الذاكره وتستخدم هذه التعليم الى تبديل محتوى مسجل مع اخر او مع محتوى موقع في الذاكره فقط ، يجب ان يكون المعاملين من نفس الطول(اي كليهما بايت او كلمه)كذلك من غير الجائز تبديل محتوى مسجلات القطاعات مع بعضها البعض عن هذه عن طريق هذه التعليمة.

***وهذه التعليمه في الحقيقه مكافئه لثلاث تعليمات فعلى سبيل المثال تعتبر التعليمة XCHG AX, BX مكافئه الخطوات الثلاث الاتية**

1مخزن محتوى المسجل AXمؤقتا في موقع ثالث

2حرك محتوى المسجل BXالى المسجلAX

3حرك محتوى الموقع المؤقت)اي محتوى المسجل AXقبل تنفيذ العمليه (الى المسجلBX

***و الامثله التاليه تبين استعمال هذه التعليمة:**

تؤدي هذه التعليم الى مبادلة محتوى المسجل BLمع محتوى المسجلAL XCHG BL

وتؤدي هذه التعليمة الى مبادلة محتوى المسجل DXومحتوى موقع الذاكره TABLE ;CCHG TABLEDX;

وتؤدي هذه التعليمة الي مبادلة محتوى المسجل DXومحتوى موقع الذاكره TABLE

5)**تعليميه معرفه قيمه عنصر بمصفوفهXLAT :**

تستخدم هذه التعليمة بهدف معرفه قيمه عنصر في المصفوفه . ويتم ذلك باستخدام موقع ذلك العنصر ضمن عناصر المصفوفه ومن ثم انقل محتوى ذلك العنصر الى المسجل ALلذلك و قبل تنفيذ التعليمة يجب تحميل عنوان بدايه المصفوفة في مسجل الاساس BX وتحميل قيمة ازاحة الموقع المطلوب (من بداية المصفوفة)في المسجلAL

*والصيغة العامة لهذه التعليمة هي: XLAT Source_Table

*حيث ان Source_Table يمثل عنوان المصفوفة المطلوب استرجاع محتوي عنصر من عناصر ها وتخزين هذا المحتوى في المسجل ALوعلى سبيل المثال يمكن البحث عن قيمه العنصر الخامس في المصفوفه ARRAY(حيث انه ازاحه العنصر الاول =0)وباستخدام التعليمات التاليه:

MOV AL,4

MOV BX,OFFSET ARRAY

XLAT ARRAY

*حيث انه ازاحة العنصر الخامس من بدايه المصفوفه=4 لذلك يتم تخزين هذه الازاحه في المسجلAL

تستخدم هذه التعليمه في عمليه تشفير وفك تشفير الرموز وذلك عن طريق استخدام جداول المناسبه لذلك، ويلاحظ في هذه التعليمة ان عمليه حساب العنوان الفعال للعنصر المطلوب تتم بإضافه محتوى المسجل ALالى المسجلBX وكذلك بما ان العنصر المطلوب سيتم احضاره الى المسجل ALيجب ان تعرفها المصفوفة على انها من النوع.Byte

2)تعليمات لادخال والاخراج:

تشكل هذه التعليمات القسم الرئيس الثاني من تعليمات نقل البيانات , **وعددها تعليمتين فقط هما :**

1) تعليمة الادخال IN

الصيغة العامة لهذه التعليمة هي : IN accumulator, port

عند نقل بايت من منفذ الادخال . ALعند نقل كلمة او AXيمثل accumulator-حيث ان يمثل عنوان منفذ الادخال , ويمكن ان يكون العنوان قيمة مباشرة (0 الى 255), - Port اذا زادت قيمته عن 255. بقد تم بيان ذلك عند معالجة عنونة منافذDX او مخزنا في المسجل الادخال والاخراج .

ويبين منفذ الادخال لجهاز الحاسوب المكان الذي عن طريقه سيتم ادخال البيانات الى الجهاز وذلك عن طريق ربط احد اجهزة الادخال على هذا المنفذ .

تعليمات نقل العنوان (Address Transfer Instructions)

يشمل هذا القسم ثلاث تعليمات هي :

1) تعليمة تحميل العنوان الفعال LEA (Load Effective Address)

تعمل هذه التعليمة على نقل العنوان الفعال لموقع من مواقع الذاكرة الى احد المسجلات المبينة ضمن التعليمة . وبما ان العنوان الفعال ممثل ب 16 خانة ثنائية يجب ان يكون طول المسجل المستخدم 16 خانة ثنائية ايضا .

الصيغة العامة لهذه التعليمة هي :

LEA reg16, mem

حيث ان - Reg16 يمثل احد المسجلات التي سينقل اليها العنوان الفعال للموقع المطلوب (Mem) واستخدام الرقم 16 للدلالة على ان المسجل المستخدم يجب ان يكون طوله 16 ثنائية .

- mem يمثل اسم موقع الذاكرة المطلوب معرفة عنوانه الفعال وترحيل هذا العنوان الى المسجل Reg16

والمسجلات التي يمكن ان تخدم غرض استقبال عنوان الازاحه هي المسجلات عامة الغرض AX, BX, CX, DXومسجلي الفهرسة SI وDIوالمسجلين SP وBP.

2) تعليمة تحميل مسجل قطاع البيانات (LDSRegister Load Data Segment)

تعمل هذه التعليمة على مواقع في الذاكرة معرفة باستخدام توجيهة حجز البيانات DD والتي تستخدم لحجز كلمة مزدوجة (او 4 بايت) للمتغير الواحد .

الصيغة العامة لهذه التعليمة هي :

LDS reg16, double_word_pointer

حيث ان :
reg16– يمثل مسجلا من المسجلات كما في تعليمة LEA
double_word_pointer يمثل اسما لعنوان موقع في الذاكرة تم تعريفه باستخدام DD حيث يعمل على حجز كلمة مزدوجة .

وتكمن فائدة هذه التعليمة عند الحاجة لتحميل مسجل قطاع البيانات DS بعنوان بداية قطاع البيانات وتحميل مسجل اخر مثل مسجلات عامة الغرض او مسجلات الفهرسة او SP او BP بعنوان (مقدار) الازاحة لذلك الموقع .

مـــثال (30) صـفحة 371

3) تعليمة تحميل مسجل القطاع الاضافي LES (Load Extra Segment register)

الصيغة العامة لهذه التعليمة هي :

LES reg16, double_word_pointer

تعمل هذه التعليمة مثل تعليمة LES مع الاختلاف الوحيد وهو ان مسجل القطاع الذي تم تحميله هو ES وليس DS .

4. التعليمات الخاصة بمسجل الرايات (Flag Transfer Instruction)

*يتكون هذا القسم من تعليمات نقل البيانات من التعليمات الأربع الاتية :

1 تعليمة LAHF (Load AH from Flags register)

تعمل هذه التعليمة على تحميل المسجل AH و بالتحديد الخانات 0 , 2 , 4 , 6 , 7 من الرايات ZF , AF , PF , CF , SF حسب الترتيب المبين . وهذه التعليمه لا تحتاج معاملات تأخذ ضمنينا على انها المسجل AH (و يمثل المعالم المستقبل) و الرايات (تمثل معامل المصدر) .

2 تعليمة SAHF (Store AH into Flags register)

و يؤدي تنفيذ هذه التعليمة الى نقل محتوى الخانات 0 , 2 , 4 , 6 , 7 من المسجل AH الى الرايات ZF , AF , PF , CF , SF في مسجل الرايات . حيث يخزن الموقع الاول من المسجل AH في الراية CF و هكذا . و هذه التعليمه ليست بحاجة الى معاملات حيث انها تؤخذ ضمنيا على انها المسجل AH و مسجل الرايات . و كما توى فهي تعمل بعكس التعليمه السابقه . و تجدر الإشاره الى ان الهدف من تصميم هذه التعليمتين هو تحقيق الموائمه أو التوافق (Compatibility) بين أنماط مختلفة من المعالجات مثل 8088 و 8085 و 8080 , حيث ان نفس الرايات موجودة في المعالجات الثلاثة .

3 تعليمه PUSHF (Push Flags onto Stack)

و تهدف هذه التعليمة الى نقل محتويات مسجل الرايات (Flag Register) و عددها 16 ثنائية الى مكدس . و الصيغة العامه لهذه التعليمه هي : PUSHF

وهذه التعليمة ليست بحاجة الى معاملات حيث تعتبر المعاملات ضمنا على انها مسجل الرايات و المكدس . و بعد تنفيذ هذه التعليمة يتم انقاص المسجل SP بمقدار 2 ليشر على الموقع التالي في المكدس .

4 تعليمة POPF (POP Flags off the Stack)

تعمل هذه التعليمة على تحميل محتوى موقع قمة مكدس (top of stack) و المشار إليه بالمسجل SP إلى مسجل الرايات . و غالبا ما تجن ان كل تعليمة PUSHF يناظر تعليمة POPF في البرنامج .

و في نهاية هذا البند اود الاشارة الى ان التعليمتين POPF و PUSHF فقط من بين تعليمات نقل البيانات تؤثر على محتوى مسجل الرايات , اما باقي التعليمات ليس لها أي تأثير . جدول 7 صفحة 373 يبين ملخص لتعليمات نقل البيانات .

3.4 التعليمات الحسابية (Arithmetic Instruction)

*تحتوى لغة التجميع على مجموعة من التعليمات المخصصة للقيام بالعمليات الحسابية الضرورية من جمع و طرح و ضرب و قسمة . ز بما ان جهاز الحاسوب يقوم بأداء هذه العمليات على الأعداد الممثلة بأحد طرق تمثيل البيانات الرقمية داخل الحاسوب (مثل نظام المكمل لإثنين) فبإمكانه تنفيذ العمليات الحيابية على الارقام الثنائية بإشارة و بدون إشارة و على الأرقام الممثلة بالنظام العشري الثناءي المضغوط (Packed BCD) و غير المضغوط (Unpacked BCD) .

*و الفرق بين النظامين الضغوط و غير المضغوط هو أن تمثيل الرقم في النظام المضغوط يحتاج إلى 4 ثنائيات بينما تمثيله في النظام غير المضغوط يحتاج الى 8 ثنائيات .

*يحدد طول الأرقام الثنائية المستعملة في التعليمات الحسابية ب 8او 16 ثنائية . لذا تتطلب معالجة الأعداد التي يزيد طولها عن ذلك معاملة خاصة عن طريق مجموعة من التعليمات بدل تعليمة واحدة.

*وتجدر الإشارة إلى أنه يمكن للمعالج معالجة بيانات رقمية غير ممثلة بالنظام الثنائي وهذا يتطلب تنفيذ بعض تعليمات التعديل على النتائج لتعديلها وتحويلها إلى قيم سليمة بالنظام المعني .

*تذكر دائماً عزيزي الدارس ، أن المعالج الميكروي لا يميز ما إذا كانت البيانات المشمولة في عملية حسابية تمثل بيانات ثنائية مع إشارة أو بدون إشارة أو بيانات ممثلة بالنظام الثنائي المضغوط او غير المضغوط .فالمعالج في جميع الأحوال يتعامل مع جميع المعاملات الحسابية على أنها قيم ثنائية .

***ويمكن تقسيم التعليمات الحسابية إلى خمس مجموعات رئيسة تتكون كل مجموعة من عدد من التعليمات وهذه المجموعات هي :**

- تعليمات الأضافة (الجمع) Addition Instructions
- تعليمات الطرح Subtraction Instructions
- تعليمات القسمة Division Instructions
- تعليمات مد خانة الإشارة Sign _Extension Instructions

و سنتناول هذه المجموعات بالتفصيل في البنود التالية .

1.3.4تعليمات الإضاف Addition Instructions

يوجد خمس تعليمات ضمن هذه المجموعة وهي كما ياتي:

1 تعليمة الإضافة ADD

الصيغة العامة لهذه التعليمة هي :

ADD dest, source

حيث أن : - source يمثل حقل بيانات المعامل الأول (معامل المصدر)

- dest يمثل حقل المعامل الثاني للعملية والمستقبل للنتيجة .

وتعمل هذه التعليمه على إضافة محتويات الحقل source إلى محتويات الحقل dest ومن ثم تخزين النتائج في dest أي أن :

dest= dest + source

ويمكن تمثل تنفيذ التعليمه بالعلاقة :

القيمة المخزنة في حقل مستقبل البيانات بعد تنفيذ التعليمة = القيمة المخزنة في حقل مستقبل البيانات قبل تنفيذ التعليمة + القيمة المخزنة في حقل بيانات المصدر.

*و من استخدامات هذه التعليمة انه يمكن إضافة محتويات مسجل إلى مسجل اخر أو إلى محتويات موقع في الذاكرة شريطة ان يكون المعاملين من نفس الطول (بايت أو كلمة) , ككذلك يمكن اضافة قيمة فورية إلى مسجل أو موقع في الذاكرة و الأمثلة الأتيه توضح ذلك

ADD ALL, Mem_byte ;

ADD Mem_word.AX ;

ADD ADD AX, BX ; NL, 20

*و تجدر الإشارة إلى أنه من غير الجائز إضافة موقع في الذاكرة إلى اخر مباشرة . و يمكن عمل ذلك عن طريق استخدام مسجل وسيط من مسجلات عامة الغرض . كذلك من غير الجائز استعمال قيمة فورية في حقل مستقبل البيانات .

2 تعليمة الاضافة مع الحمل (ADD with carry)

الصيغة العامة لهذه التعليمة : ADC dest, source

و يشبه عمل هذه التعليمة ADD مع الفارق الوحيد انه اثناء عمليه جمع محتوى حقل المصدر ومحتوى حقل مستقبل البيانات يضاف الى ذلك ايضا قيمه رايه الحمل (Carry Flag)

أي انه يمكن تمثيل عمل هذه التعليمة كما يلي :

القيمة المخزنة في حقل مستقبل البيانات بعد اتمام العملية = القيمة المخزنة في قبل بدء التنفيذ + القيمه المخزنه في حقل المصدر (source) + القيمه المخزنه في راية الحمل CF أي أن : dest = dest + source + CF

ولتوضيح مبدH الحمل نفرض اننا نود اضافه العدد 250 الى العدد 20 كما يلي :

يمثل العدد 250 بالنظام الثنائي على الشكل 11111010

كل ذلك يمثل العدد 20 بالنظام الثنائي على الشكل +00010100

يكون حاصل مجموع العددين هو 10 (270) 100001110

ولتمثيل النتيجه تمثيلا صحيحا نحتاج الى 9 خانات ثنائيه . اما اذا كنا نستخدم مسجلا او موقعا في الذاكره مكون من ثمانيه خانات فان الخانات الثمانيه الاولى ابتداء من اقصى اليمين (00001110 في هذا المثال) سيتم تخزينها في حقل مستقبل البيانات وان الخانه التاسعه من النتيجه (القيمة 1 في هذا المثال) سيتم تخزينها في رايه الحمل . اذا عليك اذا اخذنا القيمه المخزنه في موقع الذاكره او المسجل لوحدها فهي تمثل العدد 14 وليس العدد 270

من الممكن استخدام التعليمه ADC مع التعليمة ADD لجمع عددين طول كل منهما 32 ثنائية . لو فرضنا على سبيل المثال ان العدد الاول مخزنه في المسجلين DX و CX و العدد الثاني في المسجلين AXو BX ، يمكن اضافة محتوى المسجلين CXو DX إلى محتوى المسجلين AX و BX على النحو التالي :

ADD AX,CX

ADC BX,DX

فالتعليمة الأولى تعمل على إضافة محتوى المسجل CX إلى محتوى المسجل AX . وفي حال وجود حمل (باليد) في ناتج هذه العملية يتم اخذه بعين الإعتبار في التعليمة الثانية أي ADC BX , DX

والتي يؤدي تنفيذها الى إضافة محتوى المسجل DX إلى محتوى المسجل BX مضافاً إلى ذلك قيمة راية الناتجة عن التعليمة ADD السابقة .

3) تعليمة تعديل الآسكي للجمع ASCII Adjust for Addition) AAA)

الصيغة العامة لهذه التعليمة هي:

AAA

حيث تستخدم بدون معاملات وتأتي بعد تعليمات الجمع مباشرة . وتعمل هذه التعليمة على تعديل القيمة المخزنة في المسجل AL عند جمع الأعداد الممثلة بالنظام العشري الثنائي غير المضغوط ، أما طريقة عمل هذه التعليمة فيمكن تلخيصها بالآتي :

تفحص هذه التعليمة النصف الأيمن من المسجل AL ، فإذا احتوى على رقم يتراوح بين 0 و 9 تتم عملية تصفير النصف الأيسر من AL ووضع الرايات AF و CF في حالة الصفر . أما إذا كان النصف الأيمن من المسجل AL يحتوي على قيمة أكبر من 9 ، أو كانت الراية AF في حالة1 فهذا يعني أن محتوى النصف الأيمن من المسجل AL لا يحتوي رقما" ممثلا" بالنظام العشري الثنائي . لذلك يتم تحويل هذه القيمة المخزنة في AL إلى قيمة عشرية ثنائية غير مضغوطة .

ويتم ذلك حسب الخطوات التالية نتيجة تنفيذ التعليمة AAA بعد العملية الحسابية

- يضاف الرقم 6 إلى محتوى المسجل AL
- يضاف الرقم 1 إلى محتوى المسجل AH
- يصفر النصف الأيسر من المسجل AL
- توضح الرايات CF و AF في حالة 1

مثال 31 صفحه 377

4 تعليمه التعديل العشري للجمع(DDA (Decimal Adjust for Addition

الصيغه العامه لهذا التعليمه هي : DDA

وهي تحتاج الى معاملات وطريقه عملها يشبه الى حد كبير عمل تعليمه AAA الا ان هذه التعليمه تعالج جميع القيم المخزنه في المسجل AL وليس الرقم السادس عشر الاول فقط . لذلك فان هذه التعليمات ضروريه عند جمع رقمين ممثلين بالنظام العشري الثنائى المضغوط وذلك لتعديل النتيجه كي تصبح هذه النتيجه ايضا سليمه حسب هذه الطريقه للتنفيذ . ويؤدي تنفيذ هذه التعليم الى الخطوات التاليه وحسب الترتيب.

- اذا كانت قيمه النصف الايمن في المسجلAL اكبر من 9 او كانت الراية AF في الحاله "1" يضاف الرقم 6 الى النصف الايمن من المسجل AL وتوضع الرايه AF في الحالة "1" والا فيترك النصف الايمن للعدد كما هو . ومن ثم يتم فحص النصفي الايسر من AL

- إذا كانت القيمه المخزنه في النصف الايسر من المسجل AL وتوضع الرايه CF في الحالة "1"

مثال 32 صفحه 379

5 تعليمة الزياده بواحد INC (increment)

تستخدم هذه التعليمات عند الحاجه لاضافه القيمه 1 الى محتوى احد السجلات او مواقع الذاكره والصيغه العامه لهذه التعليمة هي :

INC destination

حيث ان : - Destination تمثل مسجلا او موقعا في الذاكره بطول 8 او 16 ثنائية .

وهذا التعليم لا تؤثر على رايه الحمل كما هو الحال مع تعليمتي ADC و ADD . تستخدم هذه التعليمه عند الحاجه لتعديل قيم العدادات (Counters) في جمل التكرار و كذلك لتعديل مسجلات الفهرس او مسجلات المؤشرات عند التعامل مع مجموعه من القيم المتجاوره في الذاكره . وفيما ياتي بعض الامثله على هذه التعليمة :

تؤدي هذه التعليم الى زياده محتوى المسجلAX بالقيمة 1

INC AX;

وتؤدي هذه التعليم الى زياده محتوى موقع الذاكره المعنون بالاسم Table بمقدار 1

INC Table;

ومن مميزات هذه التعليمات ان المعالج 8088 يحتاج لوقت اكثر عند استخدام هذه التعليمات مع مسجل طوله 8 ثنائيات منه عند استخدامها مع مسجل طوله 16 ثنائيه . والسبب في ذلك ان مصممي هذا المعالج قد اخذ بعين الاعتبار ان المبرمجين سوف يستعملون العدادات بطول 16ثنائية أكثر من تلك التي بطول 8 ثنائيات . هبه عليك سان المصممين قد تم وتعليما خاصه لهذا الغرض بطول بيت واحد زياده مسجل طوله 16 ثنائية

*بطول 16ثنائية اكثر من تلك التي طولها 8ثنائيات لذلك فإن المصممين قد صمموا تعليمة خاصة لهذا الغرض بطول بايت واحد لزيادة مسجل طوله 16ثنائية

تعليمات الطرح

*يقوم المعالج 8088وافراد عائلته بإجراء عمليات الطرح بواسطة وحدة الجمع (adder)وذلك بجمع المطروح منه مع المكمل لاثنين للعدد الثاني (اي المطروح)حيث أن هذا المعالج لا يحتوي على وحدة داخلية خاصة بتعليمات الطرح

*فعلى سبيل المثال فإن العملية الحسابية :A-B

يمكن تمثيلها على الشكل :A+(-B)

حيث تمثل القيمة (-B)باستخدام المكمل لاثنين وبذلك تحويل عملية الطرح إلى عملية جمع وفي مثل هذه الحالة فإنه يتم إهمال قيمة راية الحمل إذا ما زادت نتيجة عملية الإضافة على 8خانات ثنائية

***تتكون تعليمات الطرح من 7تعليمات سيتم معالجتها في البنود التالية :**

1)تعليمة الطرح (SUB (Subtract

*الصيغة العامة لهذه التعليمة هي:

SUB dest,source

*حيث أن :source-يمثل حقل العدد المطروح

dest-يمثل حقل العدد المطروح منه ويستخدم لتخزين ناتج العملية

وتعمل هذه التعليمة على طرح المعامل source من العامل dest وتخزين النتيجة في dest اي أن :(dest=dest -source)

*باستخدام هذه التعليمة يمكن طرح القيمة المخزنة في مسجل من المسجلات أو قيمة فورية من مسجل اخر أو من موقع في الذاكرة.كذلك يمكن طرح قيمة مخزنة في موقع في الذاكرة في مسجل

*ومن غير الجائز طرح موقع في الذاكر من موقع آخر مباشرة

*كذلك من غير الجائز استعمال قيمة فورية مكان الحقل dest يمكن لأي من الحلاقين source و deatان يتكون من 8ثنائيات أو 16ثنائية

*والأمثلة الآتية تبين استعمال هذه التعليمة

2)تعليمة الطرح مع الاستعارة (SBB (Subtract with borrow

*تؤدي هذه التعليمة عملا مشابها لعمل التعليمة SUBمع الفارق البسيط وهو أن هذه التعليمة تأخذ بعين الاعتبار راية الحمل أثناء عملية الطرح

*والصبغة العامة لهذه التعليمة هي :

SBB Dest,source

*حيث أن scource و dest كما في التعليمة SUB

*والمقصود بهذه التعليمة هو أن :

قيمة dest بعد إجراء التعليمة =قيمة dest قبل إجراء العملية - قيمة - source قيمة راية الحمل

اي ان :

dest =dest -source - CF

*يمكن استخدام التعليمة SBB مع التعليمة SUB لطرح عدد مكون من 32ثنائية مخزن في مسجلين طول كل منهما 16ثنائية من عدد آخر مماثل له في الطول ومخزن في مسجلين آخرين

مثال (34)ص 382]

3)تعليمة التعديل الاسكي للطرح (Asecii Adjust for Subtraction)

*يأتي موقع هذه التعليمة في الغالب بعد التعليمتين SUB و SBB في برنامج لغة اسمبلي وذلك لتحويل القيمة المخزنة في المسجل AL من الصيغة الثنائية إلى النظام العشري الثنائي

*والخطوات التي تتم نتيجة لتنفيذ هذه التعليمة تعتمد على القيمة المخزنة في النصف الأيمن من المسجل AL , حيث تفحص التعليمة AAS النصف الأيمن من المسجل AL

*فإذا كانت قيمة محتوياته تتراوح بين 0و 9 تؤدي هذه التعليمة إلى تصفير النصف الأيسر من المسجل AL ووضع الرايات CF و AF في حالة "0"

*اما إذا كان النصف الأيمن من ثنائيات المسجل AL يحتوي على قيمة اكبر من 9(وهو اكبر رقم بالنظام العشري الثنائي يمكن تمثيله في 4ثنائيات)أو كانت قيمة الراية AF تساوي *"1" تنفذ الخطوات الآتية :

_ يطرح الرقم 6 من محتوى المسجل AL

_يطرح الرقم 1من المسجل AH

_ توضع الرايات CF و AF في حالة "1"

_ يصفر النصف الأيسر من المسجل AL

_ بذلك يصبح المسجل AL محتويا على رقم عشري ثنائي غير مضغوط ، والصبغة العامة لهذه التعليمة هي : AAS

حيث أن المعامل يأخذ ضمنا على أنه المسجل AL مثال (35) ص 348

4) تعليمة التعديل العشري للطرح DAS (Decimal Adjust Substraction)

*الصيغة العامة لهذه التعليمة هي : DAS

اي ان التعليمة تكتب بدون معاملات حيث أنها ضمنا تعمل على محتوى المسجل AL . وتعالج هذه التعليمة الرقمين المخزنية في المسجل AL بدلا من رقم واحد كما هو الحال مع التعليمة السابقة . وتلي هذه التعليمة تعليمة SUB أو تعليمة SBB في برنامج لغة اسمبلي . أما بالنسبة لأداء هذه التعليمة فيمكن تلخيصه بالآتي:

* تقوم هذه التعليمة بفحص محتويات المسجل AL وبناء عليه :

1) اذا كان محتوى النصف الأيمن من المسجل AL يحمل قيمة اكبر من 9 أو كانت الراية AF في الحالة 1 يطرح الرقم 6 من المسجل AL وتوضع الراية AF في الحالة 1 .

2) بعد ذلك يتم فحص النصف الأيسر من المسجل AL حيث أنه إذا كان يحتوي على قيمة اكبر من 9 أو كانت الراية CF في حالة 1 يطرح الرقم H60 من المسجل AL وتوضع الراية AF في الحالة 1 .

لاحظ عزيزي الدارس أن التعليمتينDAA و DAS لا تؤثران على محتوى المسجل AH كما هو الحال في التعليمتين AAA و AAS . كذلك فإن راية الحمل المساعد AF توضع في الحالة 1 اذا كان هناك حمل من الرقم الاول إلى الرقم الثاني أو استعارة من الرقم الثاني للرقم الأول . وبالنسبة للراية CF فإنها توضع في الحالة 1 اذا كان هناك حمل من الرقم الثاني في المسجل AL الي بايت آخر ، أو هناك استعارة من بايت آخر الي هذا الرقم.

5) تعليمة النقص بواحد DEC (Decrement)

* الصيغة العامة لهذه التعليمة هي : DEC dest

حيث يمثل dest اسم مسجل أو عنوان موقع الذاكرة مكون من 8 أو 16 خانة ثنائية.

وتعمل هذه التعليمه على إنقاص محتوى المسجل أو موقع الذاكرة المقصود بالقيمة 1 وليس لهذه التعليمة اي تأثير على الرايات . كما هو الحال مع تعليمة INC فإن إنقاص محتوى مسجل طوله 16 ثنائية يستغرق وقتاً أقل من الوقت اللازم لانقاص محتوى مسجل طوله 8 ثنائيات ، كذلك فإن طول التعليمة اللازمة لإنقاص محتوى مسجل طوله 16ثنائية أقل من طول التعليمة اللازمة لإنقاص محتوى مسجل طولة 8 ثنائيات وذلك لنفس الأسباب التي ذكرناها بخصوص التعليمة INC .

6) تعليمة العكس NEG (Negate)

تستخدم هذه التعليمة لإيجاد المكمل لإثنين لمعامل معين والذي يمكن أن يكون اسم مسجل أو موقع من مواقع الذاكرة بطول 8او 16 خانة ثنائية

والصيغه العامة لهذه التعليمة هي : NEG dest .حيث يمثل dest اسم أحد المسجلات أو مواقع من مواقع الذاكرة . ويمكن لهذا المسجل أو موقع الذاكرة أن يكون بطول 8 ثنائيات أو 16 ثنائية. وتعمل التعليمة على إيجاد المكمل لإثنين للمعامل وتخزين الناتج فيه. لذلك يستخدم dest على أنه مصدر البيانات ومستقبل النتيجة . وبالنسبة لتأثير هذه التعليمة على مسجل الرايات فهو مماثل التعليمة SUB.

كما تعلم عزيزي الدارس فإنه من غير الممكن طرح مسجل أو مكان في الذاكرة من قيمة فورية في تعليمة SUB . فعلى سبيل المثال تعتبر التعليمة التالية غير صحيحة: SUB 200 , AL

7) تعليمة المقارنة CMP (Compare destination to source)

* الصيغه العامة لهذه التعليمة هي : CMP dest ,source

حيث يمثل كل من dest و source اسماء مسجلات أو مواقع في الذاكرة طولها أما 8 ثنائيات أو 16 ثنائية . ومن غير الجائز مقارنة موقع ذاكرة مع موقع ذاكرة في نفس التعليمة .

ويؤدي تنفيذ هذه التعليمة إلى طرح قيمة source في قيمة dest دون التأثير على المعامل dest وذلك خلافاً لتعليمة SUB التي يؤدي تنفيذها الي تخزين نتيجة الطرح في المعامل dest. قد تسأل عزيزي الدارس ما الفائدة إذا من هذه التعليمة؟ والجواب على ذلك هو أنها تؤثر على قيم الرايات SF , PF , OF , AF , CF , ZF حيث تستخدم هذه الرايات عند تنفيذ إحدى تعليمات نقل التحكم المشروط تأتي بعد تنفيذ التعليمة CMP بسبب أن تعليمات نقل التحكم تعمل على حالة الرايات ، وان تنفيذ التعليمة CMP يؤثر على حالة هذه الرايات فقط . والجدول (8) يبين تأثير تعليمة CMP على الرايات.

يوجد ثلاثة تعليمات ضمن هذه المجموعة هي :

1- تعليمة MUL

وتستعمل هذه التعليمة عند الحاجة لضرب عددين ممثلين بدون إشارة وتكون النتيجة كذلك بدون إشارة والصيغة العامة لهذه التعليمة هي MUL source

حيث يمثل source اما مسجلا او اسما لموقع من مواقع الذاكرة بطول 8 او 16 خانة ثنائية ويشكل احد العددين المطلوب إيجاد حاصل ضربهما ببعضهما البعض للحصول على النتيجة

اما بالنسبة للعدد الثاني اللازم لإنجاز عملية الضرب فانه يأخذ ضمنا محتوى المسجل AL اذا كان source من النوع Byte او AX اذا كان source من النوع Word

لذلك وقبل انجاز عملية الضرب يجب تخزين احد العددين اما في المسجل AL او المسجل AX وذلك حسب نوع source (بايت او كلمة) اما بالنسبة للنتيجة فتخزن على النحو التالي

a- اذا كان source من النوع Byte (أي طوله 8 ثنائيات) فان النتيجة سوف تخزن في المسجل AX حيث يخزن البايت الأقل أهمية في المسجل AL والبايت الأكثر أهمية في المسجل AH

والكلمة الأكثر أهمية سيتم تخزينها في المسجل DX

لاحظ انه عند تنفيذ عملية الضرب يجب ان يكون المعاملين من نفس الطول (بايت او كلمة) فاذا كان المعاملين من النوع Word اما اذا كان المعاملين من النوع Word فان النتيجة ستكون من النوع كلمة مزدوجة (Double word)

والمقصود هنا في البايت الأقل أهمية هو الخانات الثمانية من النتيجة مأخوذة من اقصى يمين الثنائي للنتيجة فاذا كانت النتيجة على سبيل المثال ABCDH0 فان البايت 16(CD)يمثل البايت الأقل أهمية (والذي سيخزن في المسجلAL) والبايت 16(AB) يمثل البايت الأكثر أهمية (والذي سيخزن في المسجل AHبعد انجاز تعليمة الضرب)

وبالنسبة لتاثير هذه التعليمة على الرايات فهو كما يلي

اذا كانت قيمة النصف اكثر أهمية (بايت او كلمة) للنتيجة لا تساوي 0 فان الرايات CF و OF سوف توضع في الحالة "1" والا فان هذه الرايات سوف توضع في الحالة "0"

ص: 389

وتجدر الإشارة الى انه من غير الجائز ان يكون source قيمة فورية فعلى سبيل المثال تعتبر التعليمة التالية غير سليمة : MUL 5

وذلك لكون source هنا قيمة فورية ويمكن انجاز عملية الضرب هذه عن طريق وضع القيمة 5 في مسجل او موقع في الذاكرة أولا ومن ثم استخدام هذا المسجل او موقع الذاكرة في مكان المعامل source

فيما يأتي بعض الأمثلة على تعليمة الضرب MUL ص 390

2- تعليمة ضرب الاعداد باشارة IMUL

ان الفارق الرئيس والوحيد بين هذه التعليمة والتعليمة MUL هو ان هذه التعليمة تستخدم عندما تكون الاعداد بالإشارة لذلك فان مدى الاعداد وقيمة النتيجة التي يمكن تمثيلها في حالة التعليمة IMUL اقل من مدى الاعداد وقيمة النتيجة التي يمكن تمثيلها في حالة التعليمة MUL لكون الثنائية الأخيرة من ثنائيات العدد تستخدم للدلالة على اشاراته والصيغة العامة للتعليمة هي :

IMUL source

حيث ان source كما هو الحال في تعليمة MUL وتجدر الإشارة الى انه اذا كان النصف الأكثر أهمية من النتيجة هو مجرد امتداد لاشارة النصف الأقل أهمية توضع كل من الراية CF والراية OF في الحالة "0" والا فيوضعان في الحالة "1"

ص:390

يمكن استخدام الأمثلة السابقة في تعليمة MUL كأمثلة هنا وذلك باستبدال MUL بالتعليمة IMUL

ان مدى الاعداد الصحيحة باشارة والتي يمكن تمثيلها في بايت هو -128 الى 127+ ومدى الاعداد الصحيحة التي يمكن تمثيلها في كلمة (أي 16 ثنائية) هو (215)- الى (215) +1) أي من -32768 الى + 32767 اخذين بعين الاعتبار ان التمثيل يتم باستخدام نظام المكمل لاثنين

كما في مثال ص 391

3- تعليمة تعديل الاسكي للضرب AAM والصيغة العامة للتعليمة هي AAM

تعمل هذه التعليمة على محتويات المسجل AX وذلك لتعديل حاصل الضرب في حالة ضرب الاعداد الممثلة بالنظام العشري الثنائي المضغوط عندما تكون المعاملات من النوع Byte وتؤدي هذه التعليمة الى تحويل حاصل الضرب في المسجل AX الى صيغة غير مضغوطة في المسجلين AL و AH

وتتم عملية التحويل عن طريق قسمة محتوى المسجل AL على الرقم 10 حيث يخزن ناتج القيمة في المسجل AH والباقي من عملية القسمة في المسجل AL وتؤثر هذه التعليمة على الرايات PF و ZF و SF

مثال 39 ص 392

تعليمات التقسيمة (Division Instruction)

توجد 3 تعلبمات لهذه المجموعة

تعلمية div(Unsigned Divide) تستخدم هذه التعليمة لقسمة الاعداد الممثلة بدون اشارة

والصيغة العامة لهذه التعليمة هي Div source

حيث يمثل source المقسوم عليه وهو اما اسم مسجل من المسجلات عامة الغرض عادة او اسما لموقع في الذاكرة ويكون طول source اما بايت او كلمة وبالنسبة للمقسوم عليه فياخذ ضمنا على انه اما المسجل ax او المسجلين axو dx وذلك كما ياتي

اذا كان طول المقسوم عليه بايت فان المقسوم ياخذ ضمنا على انه محتوى المسجل ax

اما اذا كان طول المقسوم عليه كلمة (أي 16 ثنائية)فان المقسوم عليه ياخذ ضمنا على انه المسجلين axو dxحيث يحتوي axعلى الكلمة الاقل اهمية ويحتوي المسجل dxعلى الكلمة الاكثر اهمية

اما بالنسبة للنتيجة فيتم تخزينها على النحو الاتي

اذا كان طول المقسوم عليه بايت فان تخزين ناتج عملية القسمة يتم في المسجل alويتم تخزين باقي عملية القسمة في المسجل ah كما هو مبين بالشكل صفحة 394

اما اذا كان طول المقسوم عليه كلمة فانه يتم تخزين ناتج عملية القسمة في المسجل axوالباقي في المسجل dxكما هو مبين في شكل 2 صفحة 394

إذا لم يكن بلامكان استيعاب ناتج القسمة من قبل المسجل alاو axفان المعالج يولد اعتراضا (interrupt) من النوع 0 وهذا الاعتراض مكافئ للاعتراض الناتج عن القسمة على 0وذلك لاخبار المبرمج بان نتيجة القسمة غير سليمة

وفيما ياتي بعض الامور التي يجب مراعاتها عند استخدام تعليمة div والتي تؤدي الى توليد فيض (overflow) في عملية القسمة

ا) اذا كانت قيمة المقسوم تساوي 0

ب) اذا كام المقسوم عليه من النوع byteوكان ناتج القسمة يزيد على 255 وذلك لان الناتج سيخزن في المسجل alوان اكبر قيمة بدون اشارة يمكن تخزينها في المسجل al هي 255

ج)اذا كان المقسوم عليه من النوع world وكان ناتج القسمة يزيد على 65536وذلك لان التاج سيخزن في المسجلax وان اكبر قيمة بدون اشارة يمكن تخزينها في المسجل axهي 65536

وكما هو الحال مع تعليمة mulمن غير الجائز استعمال قيمة فورية مع تعليمة divمكان المعامل source

بارجوع الى مثال 40 ص394 ومثال 41 ص 395 وتدريب 31ص359

2) تعليمة القسمة للاعداد باشارة idiv (signed divide)

تستعمل هذه التعليمة بطريقة مشابهة للتعليمة السابقة مع فارق وحيد هو ان الاعداد ممثلة مع اشارة لذلك فان مدى هذه الاعداد بدون اشارة من حيث المقسوم والمقسوم عليه وناتج عملية القسمة والصيغة العامة لهذه التعليمة هي

IDIV source

حيث ان source وطريقة تخزين النتيجة والمقسوم تشبه مثيلاتها في تعليمة div

العوامل التي تسبب فيضا في ناتج القسمة والتي تؤدي الى توليد اشارة مقاطعة (اعتراض) لسير عمل المعالج

أ)القسمة على0

ب)اذا كان طول المقسوم عليه بايت وزاد نات جالقسمة عن +32767او اقل من القيمة -32768وذلك لكون المعالج يستعمل المكمل لاثنين لتخزين الاعداد

وكما هو الحال مع تعليمة divمن غير الجاهز استخدام قيمة فورية مع تعليمة idivايضا مكان المعامل source

مثال 42 وتدريب 32 ص396

3)تعليمة تعديل الاسكي للقسمة add(**ASCII Adjust for Division**)

هذه التعليمة يجب أن تسبق عملية القسمة بدلا من ان تاتي بعدها كما هو الحال مع التعليمات aam,aas,aaa

لصيغة العامة:add

بالنسبة لعمل هذه التعليمة فيمكن تلخيصه بالتالي - :

يضرب محتوى المسجل AH بالعدد 10

-تضاف نتيجة الضرب إلى القيمة المخزنة في المسجلal

يصفر المسجلAH

طريقة استخدام هذه التعليمة :

حيث تعمل هذه التعليمة على تعديل قيمة المقسوم غير المضغوط ;ADD

بعد ذلك نفذ عملية القسمة;div bl

تعليمات مد خانة الاشارة (Sign Extension Instructions)

تعليمة تحويل بايت الى كلمةCBW (Convert Byte to Word)

تستعمل لمد خانة الإشارة في المسجل AL إلى جميع الخانات الثنائية في المسجلAH

لتوضيح عمل هذه التعليمة :إفرض أن المسجل AL يحتوي على القيمة الثنائية كما هو مبين بالشكل ص 397

وبغض النظر عن محتوى المسجل AH ،وبعد تنفيذ التعليمة CBWتصبح محتويات المسجل AX على النحو التالي:

AH AL

شكل ص 397

حيث أن خانة الإشارة في المسجل (AL أي الخانة الأخيرة) ومقدارها 1 يتم مدها إلى جميع خانات المسجلAH

مثال 43 وتدريب 33 ص 398

2)تعليمة تحويل كلمة إلى كلمة مزدوجة(Double to Word Convert)CWD

عمل على مد خانة إشارة العدد المخزن في المسجل AX إلى جميع خانات المسجل . DX وتكمن فائدة هذه التعليمة في تمكين المبرمج من تقسيم معامل حسابي طوله كلمة على معامل حسابي آخر طوله كلمة أيضا مع العلم أن تعليمات القسمة تتطلب أن يكون عدد خانات العدد المقسوم ضعف عدد خانات العدد المقسوم عليه.

مثال 44 ص398

ملاحظة:*تعني انه من الممكن حدوث تغيير في قيمة الراية بعداجراء العملية

؟ تعني ان قيمة الراية غير معرفة بعد اجراء العملية

Destination تعني حقل مستقبل نتيجة العملية

Sourceتعني حقل المصدر

تعليمات نقل التحكم (Control Transfer Instructions)

يمكن تصنيف تعليمات نقل التحكم إلى أربع مجموعات هي - :

1تعليمات القفز غير المشروط واستدعاء البرنامج الفرعي والعودة منه -(RET, CALL, JMP)

2تعليمات نقل التحكم المشروط (Conditional Transfer Instructions)

3تعليمات التكرار(Iteration Control Instructions)

4تعليمات الاعتراض (Interrupt Instructions)

تعليمات القفزغيرالمشروط واستدعاءالبرنامج الفرعي والجودةمنه وتشمل هذه المجموعةالتعليمات الثلاث الأتية:

١)تعليمة القفزغيرالمشروطJMP (UnconditionalJump)

تشبه هذه التعليمةجملةGOTOفي لغتي بيسك وباسكال وغيرهامن لغات البرمجة،حيث أن تنفيذهايؤدي إلى

الإنتقال إلى التعليمة الموسومة بمعامل هذه التعليمة.والصيغةالعامةلهذه التعليميةهي:

Target JMP

حيث ان Targetيمثل وسام التعليمةالتي سيتم الإنتقال إليها.

ويتم تكوين الإسمTargetبنفس الطريقةالتي تتكون بهاأسماء الأوسمة(labels)والتي تم بيانها عند شرح الصيغةالعامةلتعليمات لغة أسمبلي،أو بنفس طريقةتكوين الأسماء

المستخدمة مع التوجيهات والتي تم بيانها في الوحدة الخامسة في القسم الخاص بالصيغة العامة للتوجيهات.

وقد يكون القفز اماميا (Forward)عندما تقع التعليمة التي سيتم الإنتقال اليها والموسومة بالوسام Target

بعدتعليمةJMP،أوخلفيا(Backward)عندما تقع التعليمةالتي سيتم الإنتقال إليها

قبل تعليمةالقفز.JMPوفي كلتاالحالتين يتم حساب مقدارالفرق في المسافة(الإزاحة)في العنوان بين عنوان التعليمةJMPوعنوان التعليمة التي سينتقل التحكم اليها،حيث تضاف هذه الإزاحةإلى محتوى مسجل مؤشر

التعليمةIP (Instruction Pointer)وذلك كي

يتم الانتقال الى التعليمة المطلوبة.

وعند التعامل مع تعليمة JMP يجب مراعاة الآتي:

- إذا كان الموقع الذي سيتم الإنتقال إليه ضمن نفس القطاع الذي يحتوي على تعليمة القفز، فإن هذا النوع يطلق عليه القفز القريب (Near Jump) حيث يتم في هذه الحالة تخصيص 3 بايت للتعليمة والمعامل(Target).

- إذا كان الموقع الذي سيتم الإنتقال إليه يقع ضمن قطاع غير قطاع التعليمة JMP(حيث انه في هذه الحالة يجب استخدام الجملة "EXTRN Target FAR"

قبل القطاع الذي يحتوي على تعريف الوسام Target). ويطلق على هذا النوع من القفز بالقفز البعيد (Far Jump). وفي هذه الحالة يتم تخصيص 5 بايت (2 بايت لإزاحةTarget من بداية قطاعه و 2 بايت لعنوان القطاع الذي يحوي Target وبايت لشيفرة التعليمة) .

401

أما إذا كان الموقع الذي سيتم الإنتقال إليه يبعد عن التعليمةJMPبأقل من128بايت او يسبقه باقل من129بايت فبإمكاننا الإستفادة من ذلك وإجبار الأسمبلر على تخصيص2 بايت لتعليمة القفز وذلك باستخدام الصيغة التالية:JMP SHORT Target

ويطلق على هذا النوع من القفزبالقفز القصير.(SHORT Jump)

وفي حاله عدم إستخدام هذه الصيغة فإن الأسمبلر يعامل القفز على أنه من النوع القريب.يجب مراعاة نوعين من

معاملات تعليمية القفزJMP هما:

عندمايكون المعامل وسما(Label)،

وفي هذه الحالةيطلق على نوع القفز

المباشر وذلك كما في المثال الآتي :

..

..

..

MOV AL, BL

ADD DL, AL

JMP THERE HERE: MOV MEM_BYTE, DL

...

...

...

THERE: MOV SAVE_DL, DL

....

....

....

وهذه المجموعة من التعليمات تبين إستخدام التعليمة JMP للقفزعن مجموعة من التعليمات التي سيتم تنفيذها بالإنتقال إليها من مكان آخر في البرنامج.

- عندما يكون المعامل أحد مواقع الذاكرة أو المسجلات حيث يطلق على هذا النوع من القفز بالقفز غير المباشر (Indirect Jump). فعلى سبيل المثال يمكن استخدام

المسجل BX كما في التعليمة :

JPM BX

حيث يحتوي المسجل على إزاحة التعليمة التي سيتم الإنتقال إليها نسبة إلى قطاع البرنامج (Code Segment).

وعند تنفيذ هذه التعليمة يتم نقل محتوى المسجل BX إلى مؤشر التعليمة IP ومن ثم يتم الإنتقال إلى التعليمة المعنونة بالعنوان (CS:IP)

2) تعليمة إستدعاء الإجراء CALL

قد نحتاج أحيانا إلى تكرار مجموعة من التعليمات في أكثر من موقع في برنامج لغة أسمبلي. ففي مثل هذه الحالة يمكن تكرار هذه المجموعة من التعليمات حيثما دعت الحاجة.

لا شك أن مثل هذا الأسلوب يعتبر مضيعة لجهد المبرمج إضافة أن البرنامج يصبح طويلا بحيث تصعب عملية مراجعته وإجراء التعديلات عليه . كذلك فإن البرنامج الطويل يحتاج

إلى ذاكرة أكبر لتخزينه. ولتسهيل كتابة البرنامج ومتابعته يمكن تنظيم البرنامج وتقسيمه إلى أجزاء تدعي إجراءات (Procedures) يحتوي كل منها على مجموعة من العمليات

المخصصة لمعالجة موضوع معين. بعد تعريف هذه الإجراءات في البرنامج مرة واحدة، فإنه يمكن إستدعاؤها أي عدد من المرات عند الحاجة إليها .

وعند التعامل مع الإجراءات يجب مراعاة الآتي:

- يحدد مدخل البرنامج التنفيذي لاخبار الحاسوب بالبداية الفعلية للتعليمات باستخدام المعامل FA.

- يعني المعامل NEAR عند استخدامه مع تعريف الإجراء أن الإجراء هو فرعي ويقع ضمن قطاع التعليمات الحالي .

- يعطى كل إجراء إسما حسب قواعد تكوين الأسماء.

- ينتهي كل إجراء بالتوجيهة ENDP .

- يتم إستدعاء الإجراء باستخدام التعليمة CALL.

الصيغة العامة لإستدعاء الإجراء هي :

CALL Target

حيث أن : Target يمثل إسم الإجراء الذي سيتم إستدعائه ونقل التحكم إليه. ومايتطبق على Target في تعليمة القفز JMP ينطبق على Target هنا. يمكن لبرنامج لغة أسمبلي أن يحتوي على أي عدد من الإجراءات، وعلى أي عدد

من جمل الإستدعاء وذلك حسب الحاجة .

البرنامج الآتي يبين كيف يتم إستدعاء الإجراء :

CODE SEGMENT PARA

MAIN PROC FAR ; مدخل البرنامج التنفيذي

إستدعي البرنامج الفرعيCALL PROC 1; PROC 1

إستدعي البرنامج الفرعي CALL PROC 2; PROC 2

جملة الرجوع من البرنامج التنفيذي ; RET

;MAIN ENDP

بداية البرنامج الفرعي

:PROCI PROC; NEAR

.....

.....

....

PROC1 ;RET جملة الرجوع من

نهاية البرنامج الفرعي PROC1 PROC 1 ENDP;

PROC2 ;PROC NEAR PROC 2 بداية البرنامج الفرعي الثاني

....

.....

.....

PROC 2 RET; جملة الرجوع من

نهاية البرنامج الفرعي PROC 2 PROC2 ENDP;

CODE ENDS

عند مصادفته للتعليمة CALL وفي حالة أن البرنامج الفرعي

المستدعى معرف على أنه من النوع NEAR فإن المعالج يحتفظ بعنوان الأمر التالي (محتوى المسجل IP) على

الكومة وذلك عن طريق تنفيذ التعليمة PUSH IP. وعند مصادفته للتعليمة RET فان تنفيذها يؤدي إلى إسترجاع محتوى المسجل IP المخزنة على الكومة وذلك بتنفيذ المعالج للتعليمة POP IP.

أما إذا كان الإجراء المستدعي من النوع FAR فإن المعالج يدفع بمحتوى المسجلCS إلى الكومة ومن ثم محتوي المسجل IP عندمصادفة التعليمة CALL وذلك بتنفيذ التعليمتين

PUSH CS

PUSH IP

404

وعند مصادفة التعليمة RET يسترجع المعالج محتويات المسجلين CS و IP بتنفيذ التعليمتين

POP IP

POP CS

وذلك للإنتقال بالتحكم إلى التعليمةالتاليةلتعليمةالإستدعاءCALL .

3)تعليمة العودة من البرنامج الفرعي RET

الصيغة العامة لهذه التعليمة هي:

RET

حيث أن تنفيذها يعني العودة إلى الجملة التي تلي جملة الإستدعاء (Call) كي يستمر التنفيذ من هناك.

اذا كان الاجراء الذي يحتوي على التعليمة RET من النوعNEARفإن

تنفيذها يؤدي إلى إسترجاع محتوى الكومة إلى المسجل IP وأما إذا كان الإجراء من نوعFAR فإنه يتم إسترجاع محتوى IP ومحتوى CS من الكومة.

ب) تعليمات نقل التحكم المشروط Conditional Jump Instructions

تستخدم هذه التعليمات لنقل التحكم بسير تنفيذ البرنامج إلى تعليمة معينة عند تحقق

شرط من الشروط مثل أن تكون محتوى المسجل CX مساويا للصفر أو أن يكون راية أو أكثر من مسجل الرايات (Flags Register) في وضع معين. وبالنسبة لموقع الرايات في مسجل الرايات فهو على النحو :

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

- - - - O D I T S Z - A - P - C

حيث ان الاشارة - تعني عدم استخدام هذا الموقع من مواقع مسجل الرايات.

405

يوجد 31 تعليمة من تعليمات نقل التحكم المشروط في المعالج الميكروي 8088 وذلك كما هو مبين في الجدول رقم (10)

ملاحظة: * تعني ان التعليمة ملائمة بعد العمليات الحسابية مع الاعداد بإشارة.

والصيغة العامة لهذه التعليمات هي:

JX short_label

حيث ان:

-تتكون من حرف إلى ثلاثة أحرف كما هو مبين في الجدول (10.6).

- short_label تمثل عنوان التعليمة التي سيتم الإنتقال إليها ويجب أن لا يزيد بعدها

604

عن 127+ بايت في حالة القفز للامام وان لا يقل عن128- فيحالة القفز للخلف والسبب في ذلك انه خصص بايتا واحدا للدلالة على مقدار القفز وانه يمكن للبايت الواحد ان يستوعب بين 128– الى 127+

يلاحظ ان جميع تعليمات نقل التحم المشروط تستخدم القفز القصير (SHORT) ويلاحظ كذبك ان الاسمبلر يتعرف على بعض التعليمات باكثر من شكل واحد ويبين في الجدول 18(10.6)نوع مختلف من التعليمات حيث ان بعضها ملائم للاعداد باشاره وبعضها ملائم للاعداد من دون اشاره الانواع من التعليمات يستخدم عادة بعد تعليمات المقارنة والبعض الاخر من تعليمات مناسب للفحص الحسابي حيث يمكن استخدامها بعد اجراء عملية من العمليات الحسابية

وعند تحقق شروط الانتقال يتم اضافة مقدار الازاحة بين (128- و127+)الى محتوى مؤشر التعليمية IP كي يتم الانتقال الى التعليمية المناسبة

فيما ياتي بعض الامثلة على تعليمات قل التحكم

مثال 45 صفحه 407:

يؤدي تنفيذ التعليمتين

ADD AX,BX

JC Lable

الى نقل التحكم الى التعليمية المعنونة يالعنوانLable اذ ادت عملية الجمع الى نتيجة لايمكن استيعابها من قبل المسجل AXوالذي يؤدي بدوره الى رفع رايات الحمل في الحالة 1

ويؤدي تنفيذ التعليميه JC Lable الى فحص راية الحمل فاذا كانت في الحالة 1يتم الانتقال الى التعليمية المعنونه بالاسم Lable والا فان التنفيذ ينتقل الى التعليمية التالية

مثال46 صفحه 407:

يبين هذا المثال استخدام تعليمية القفز المشروط JZ أي اقفز اذا كانت راية الصفر في حالة الوضع 1بعد تعليمية الطرح كما ياتي :

SUB AL,BL

JZ LOS

ففي هذا المثال يتم اجراء عملية الطرح اولا فاذا كان محتوى المسجلAL مساوي BL فان حاصل عملية الطرح سيؤدي تاى تخزين النتيجة 0في المسجل ALوهذا بدوره يؤدي الى رفع راية الصفرZF في الحالة 1 والا فان راية الصفر توضع في الحالة 0 ومن ثم تؤدي الى تنفيذ التعليمية

JZ LOS الى الانتقال الى التعليمية المعنونة بالاسم LOS اذا كانت راية الصفر في حالة 1والا فان التنفيذ ينتقل الى التعليمية الثانية

الاطلاع على مثال 47 صفحة 408:

تدريب 34 صفحة 408:

بين نتيجة تنفيذ مجموعة التعليمات الاتية :

MOV AL,[BX]

CMP AL,[BX+1]

JNL LOS

.....

.....

...

LOS:.....

جدول 11صفحة 409 يوضح علاقة بعض تعليمات نقل التحكم المشروط مع تعليمة GMP

ج) تعليمة التكرار :Iteration or Loop Instruction

تستخدم هذه العملية لتكرار تنفيذ تعليميه او اكثر لحين تحقق شرط معين وعند التعامل مع تعليمات التكرار يجب مراعاة الامور التالية

اعطاء المسجل CX قيمة ابتدائية مساوية لعدد مرات التكرار

يؤدي تنفيذ تعليمية التكرار الى انقاص قيمة المسجل CX تلقائي بمقدار 1

يبقى التكرار قائما ما دامت قيمة المسجل CX غير مساوية للصفر اضافة الى شرط اخر في بعض تعليمات التكرار

يوجد خمسة انواع من تعليمات التكرار يمكن تقسيمها الى ثلاث مجموعات هي :

1تعليمة التكرار المشروط بمحتوى المسجل CX فقط LOOP

الصيغة العامه لهذه التعليمية :LOOP Loaction

حيث يؤدي تنفيذ هذه التعليمية الى ما يلي :

انقاص محتوى المسجل CX بالقيمة 1

فحص محتوى المسجل CX فاذا كانت هذه القيمة لا تساوي 0 فانه يتم انتقال التنفيذ الى التعليمية المعنونة بالاسم Location والا فانه سيتم الانتقال الى التعليمة التالية لتعليمية LOOP

الاطلاع على مثال 48 صفحة 410:

2 تعليمتي LOOPIN (LOOP whil Not Equal)

ومكافئتها LOOPNZ (LOOP Whil Not Zero flag)

الصيغة العامة لهاتين التعليمتين هي :

LOOPEN Loacation

او

LOOPNZ Loacatio

حيث يؤدي تنفيذ أي من هاتين التعليمتين الى ما ياتي :

انقاص محتوى المسجل CX بالقيمة 1

اذا بقيت القيمة المخزنة في المسجل CXلا تساوي صفر وكانت راية الصفر في الحالة 0 يتم الانتقال الى التعليمة المعنونة بالاسم Locaation والا فانه يتم الانتقال الى التعليمية التالية لتعليمة الدوران LOOPEN او LOOPNZ

وهذا يعني ان عملية التكرار سوف تتوقف عندما تصبح القيمة المخزنة في المسجل CX مساوية للصفر او ان راية الصفر في الحاله 1

3 تعليمتي LOOP (LOOP While Equal)ومكافئتها LOOPZ (LOOP If Zero Flag)

الصيغة العامة لهاتين التعليمتين هي :

LOOPNE Loacation

LOOPNZ Loacation

حيث تؤدي تنفيذ أي من هاتين التعليمتين الى :

انقاص محتوى المسجل CXبالقيمة 1

اذا بقيت القيمة المخزنة في المسجل CXلا تساوي صفر وكانت راية الصفر في الحالة 1 يتم الانتقال الى التعليمة المعنونة بالاسم Locaation والا فانه يتم الانتقال الى التعليمية التالية لتعليمة الدوران LOOPEN او LOOPNZ وهذا يعني ان عملية التكرار سوف تتوقف عندما تصبح القيمة المخزنة في المسجل CX مساوية للصفر او ان راية الصفر في الحالة 0

تدريب 35 صفحة 411:

اكتب مجموعة التعليمات الازمة لايجاد اول عنصر يحتوي على القيمة 50 من مجموعة مكونة من 30 عنصر من النوع Byte حيث ان Table يمثل عنوان بداية هذه العناصر وفي حالة عدم وجود مثل هذا العنصر يجب القفز الى الموقع المعنون بالاسم Not -found اما ان كان العنصر موجود استخدم المسجل SI للاشارة علية ضمن مجموعة العناصر الثلاثين اما في حالة عدم وجود مثل هذا العنصر ضمن العناصر يجب ان يشير المسجل SI الى العنصر الاحق للعناصر الثلاثين

الاطلاع على اسئلة التقويم الذاتي 22 صفحة 411:

د) تعليمات الاعتراض :Intrrupt INSTRUCATION

يمكن تصنيف ثلاثة أنواع من تعليمات الاعتراض هي :

1_تعليمة الاعتراض (interrupt) INT الصيغة العامة لهذه التعليمة هي : INT Interrupt_Type

حيث يمثلInterrupt_Type

نوع الإعتراض وهو عبارة عن عدد صحيح تتراوح قيمته بين 0,255 .

وعند تنفيذ هذه التعليمة يقوم المعالج بما يأتي :

_ يتوقف تنفيذ البرنامج الحالي بعد الانتهاء من تنفيذ التعليمة الحالية والتي بدأ تنفيذها .

_ تخزين مسجل الرايات على المكدس(stack)

_ تصفير كل من رايات المتابعة (TF) والاعتراض (IF) وذلك لإبطال إمكانية التنفيذ خطوة خطوة وإبطال الاعتراضات المقنعة (maskable Interrpts) الآخرى .

_ تخزين مسجل قطاع التعليماتCS على المكدس .

_ إيجاد عنوان موجه الإعتراض. (Interrupt Vector) وذلك بضرب نوع الإعتراض بالقمية4 ، ويتكون موجه الإعتراض من كلمتين (4 بايت) .

_ تحميل محتويات الكلمة الثانية من عنوان موجه الإعتراض (الناتج من عميلة الضرب أعلاه) إلى مسجل قطاع التعليمات .

_ تخزين محتويات مؤشر التعليمة Ip على المكدس.

_ تحميل محتويات الكلمة الأولى من عنوان موجه الإعتراض إلى مؤشر التعليمة.

بعد ذلك ينتقل التنفيذ إلى الروتين الخاص بهذا النوع من الاعتراض.

وتجدر الإشارة إلى أن موجهات الإعتراض (Interrupt Vectors) مخزنة في مواقع الذاكرة من0 إلى 1023 .

تستخدم تعليمة الإعتراض عند الحاجة للتعامل مع وحدات الإدخال والإخراج بغرض ادخال البيانات أو طباعة النتائج وكذلك عند التعامل مع وحدات التخزين المساعدة، حيث يتطلب ذلك تنفيذ روتين معين يعالج العمليات الخاصة بهذه الوحدات.

الاطلاع على مثال 49 صفحة 412:

2) **تعليمية INOT (Interrupt On Overflow)**

تعتبر هذه التعليمية من تعليمات الاعتراض المشروط حيث تعمل هذه على استدعاء الاعتراض من نوع 4عندما تصبح راية الفيض OFفي الحالة 1 لذلك يمكن استخدام هذه التعليمية بعد التعليمات التي قد تؤدي الى حدوث فيض في النتيجة مثل القسمة على القيمة 0 والصيغة العامة لهذه التعليميةهي : INOT

2) **تعليمية العودة من الاعتراضIRET (InterruptReturn)**

تستخدم هذه التعليمية لاعادة المعالج الى حالتة قبل تنفيذ روتين الاعتراذ لذلك فان الموقع الطبيعي لهذه التعليمية هو اخر تعليمية من تعليمات روتين الاعتراض يؤدي هذه التعليمات الى تحميل قيم لكل من مؤشر التعليمية ومسجل قطاع التعليمات ومسجل الرايات المكدس وذلك بترتيب عكسي لما تم في تعليمة INT

والصيغة العامة لهذه التعليمة : IRET

وتجدر الاشاره الى وجود تعليمتين لمعالجة راية الاعتراض هما :

تعليمة تصفير راية الاعتراض CLL (Clear Interrupt flag)

وتعليمة وضع راية الاعتراض في الحالة 1 STI (Set Intrrupt flag)

وتقع هاتين التعليمتين ضمن المجموعة الاخيره من تعليمات المعالج 8088 أي تعليمات التحكم بالمعالج الميكروي

تعليمات معالجة البيانات الثنائية Bit Manipulaton Instructions ::

تعمل هذه التعليمات على ثنائيات المسجلات أو مواقع الذاكرة حيث يمكن تقسيم هذه التعليمات الى ثلاث مجموعات على النحو هي:

أ) التعليمات المنطقية Logical Instruction سميت هذه التعليمات بهذا الاسم لكونها تعمل حسب قواعد المنطق الشكلي (formal logic) وليس حسب القواعد الحسابية.

يوجد خمس تعليمات من هذا النوع هي : 1) تعليمة " و " (And Instruction) تحتاج هذه التعليمة إلى معاملين طول كل منهما إما بايت أو كلمة . ويمكن للمعاملين أن يكونا مسجلين أو مسجل ، موقع ذاكرة أو قيمة فورية مع مسجل أو مع موقع ذاكرة . والصيغه العامة لهذه التعليمة هي:

AND dest , source

حيث أن dest و source يمثلان معاملين هذه التعليمة . وعند تنفيذ هذه التعليمة يتم تطبيق القاعدة المنطقية

يتم تطبيق القاعدة المنطقية "و" على الخانات الثنائية المتناظرة من كلا المعاملين ويخزن الناتج في المعامل dest . ويكون الناتج حسب القواعد التالية :

AND 0 -----> 0 0

AND 1 ------> 0 0

AND 0 ------> 0 1

AND 1 ------> 1 1

مثال (50) صفحة 414

فيما يلي أمثلة على هذه التعليمة :

AND AX,BX

AND Mem_Loc,10010110B

حيث يؤدي تنفيذ التعليمة الاولى الي تطبيق القاعدة "و" على المسجلين AX و BX تخزين النتيجة في المسجل AX ، ويؤدي تنفيذ التعليمة الثانية الي تطبيق القاعدة "و" على محتوى موقع الذاكرة المسمى Mem_Loc والقيمة الفورية 2(10010110) وتخزين الناتج في موقع الذاكرة Mem_Loc .

ويلاحظ أن القاعدة المنطقية "و" تعني أن هناك حالة واحدة فقط تجعل قيمة الثنائية الناتجة في الحالة "1" وهي عندما تكون الثنائيتين المتناظرتين من المعاملين في الحالة "1" والإ فإن قيمة الثنائية الناتجة المناظرة لهاتين الثنائيتين تكون في الحالة "0" .

تعليمة " أو " (OR Instruction)

يتم تنفيذ هذه التعليمة على نفس نوع المعطيات كما هو الحال في التعليمة السابقة . والصيغه العامه لهذه التعليمة هي : OR dest, source

حيث أن dest و source يمثلان معاملي التعليمة.

ويؤدي تنفيذ هذه التعليمة الي تطبيق القاعدة المنطقية "أو" على الثنائيات المتناظرة للمعاملين وذلك حسب القواعد التالية وتخزين الناتج في المعامل dest .

OR 0 -----> 0 0

OR 1 -----> 1 0

OR 0 -----> 1 1

OR 1 -----> 1 1

وتعني قاعدة "أو" أن الثنائية الناتجة من تطبيق "أو" على ثنائيتين متناظرتين في المعاملين توضع في الحالة "1" اذا كانت إحدى هاتين الثنائيتيناو كليهما في الحالة "1" . والا فإن قيمة الثنائية الناتجة تكون في الحالة "0" .

مثال (51) صفحة 415

تعليمة " أو المقصورة" Instruction XOR (Exclusive OR) كما هو الحال مع التعليمتين السابقتين تحتاج هذه التعليمة الي معاملين ومايتطبق عليهما ينطبق أيضا على هذه التعليمة بالنسبة للمعاملين. والصيغة العامه لهذه التعليمة هي : XOR dest, source

حيث أن dest و source يمثلان معاملي هذه التعليمة . وتعمل هذه التعليمة حسب قاعدة " أو المقصورة " المنطقية والتي عند تطبيقها على ثنائيتين يكون الناتج القيمة "1" في حال اختلاف هاتين الثنائيتين . أما في حال تساو هاتين الثنائيات (اي كليهما "0" أو كليهما "1") فإن الناتج يكون مساويا للقيمة "0" وذلك طبقا لقواعد أو المقصورة وهي:

XOR 0 -----> 0 0

XOR 1 -----> 1 0

XOR 0 -----> 1 1

XOR 1 -----> 0 1

مثال (52) صفحة 416

تعليمة TEST :

تشبه هذه التعليمة تعليمة AND والفارق بينهما أن تعليمة TEST لا تؤثر على المعاملات وتأثيرها الوحيد هو على الرايات وبنفس الأسلوب الذي تؤثر فيه تعليمة AND .

وعند تنفيذ هذه التعليمة يتم وضع الرايات CF و OF في الحالة "0" كذلك قد تعدل حالة الرايات SF,ZF,PF . أما الراية AF فتبقى غير معرفة.

والصيغه العامة لهذه التعليمة هي : TEST operand1,operand2

يمكن استخدام هذه التعليمة قبل تعليمة نقل التحكم المشروط JNZ ، حيث يتحقق القفز عند وجود (على الأقل) موضعان ثنائيان متناظران من معاملي التعليمة TEST يحتوي كل منهما على القيمة "1" . حيث أن هذا يعني أن ناتج العمليه: TEST operand1, operand2 لا يساوي الصفر ولذلك فإن راية الصفر ZF توضع في الحالة "0".

تعليمة النفي NOT

بخلاف التعليمات السابقة فإن هذه التعليمة بحاجة الي معامل واحد يمكن أن يكون أحد المسجلات أو موقع من مواقع الذاكرة

والصيغه العامة لهذه التعليمة هي : NOT dest

حيث أن dest يمثل معامل التعليمة وطوله بايت أو 2بايت.

وتنفيذ هذه التعليمة يؤدي إلي تغيير حالة كل ثنائية من ثنائية المعامل dest حيث أن الحالة "1" تحول إلى الحالة "0" والعكس لجميع الثنائيات.

مثال (53) صفحة 418

ب) تعليمات الإزاحة Shift Instructions

تستخدم هذه التعليمات لإزاحة

محتويات مسجل أو موقع من مواقع الذاكرة طول كل منها بايت أو 2بايت . يوجد نوعان رئيسان من تعليمات الأزاحة يحتوي كل منهما على تعليمات حيث يمثل النوع الأول تعليمة الازاحة الحسابية لليمين SAR وتعليمة الازاحة الحسابية لليسار SAL, أما النوع الثاني فيمثل تعليمات الازاحة المنطقية لليمين SHR ولليسار SHL .

تحتاج هذه التعليمات الى معاملين حيث يمثل المعامل الاول dest اسم المسجل أو موقع الذاكرة المراد إزاحته ويمثل المعامل الثاني count مقدار الإزاحة . ويكون المعامل الثاني أما الرقم 1 (اذا كان المطلوب إزاحة الموقع خانه واحدة) أو المسجل CL اذا كانت قيمة الازاحة أكثر من موقع واحد حيث يخزن هذا العدد في المسجل CL قبل تنفيذ التعليمة.

وتستخدم تعليمات الازاحة الحسابية مع الاعداد بإشارة وتعليمات الازاحة المنطقية مع الاعداد بدون إشارة . وفيما يلي وصف لهذه التعليمات.

تعليمة الازاحة الحسابية لليمين SAR (Shift Arithmetic Right)

الصيغه العامة لهذه التعليمة هي : SAR dest,count

حيث تعمل هذه التعليمة على إزاحة محتوى المعامل dest الي اليمين بعدد من الخانات مساوي لقيمة count ويتم ملئ الخانات المفرغه من اليسار بإشارة العدد أما بالنسبة لقيمة الثنائية الخارجة من اقصى اليمين. فيتم الاحتفاظ بها في راية الحمل CF

مثال (54) صفحة 419

2) تعليمة الازاحة الحسابية لليسار SAL(Shift Arithmetic Left)

والصيغه العامه لهذه التعليمة هي :SAL dest, count

حيث يمثل dest المعامل المطلوب إزاحته ويمثل count مقدار الازاحة وهو إما يساوي 1 أو يمثل المسجل CF .

بما أن الإمامة لليسار فإن المواقع المفرغة من أقصى اليمين يتم صلتها بالقيمة "0" أما القيمة الخارجة من الموقع في أقصى اليسار فيتم الاحتفاظ بها في الراية CF. والمثال (55) يبين ذلك صفحة 420

3) تعليمة الإزاحة المنطقية لليمين SHR(Shift Logical Right)

والصيغة العامة لهذه التعليمة هي: SHR dest, count

حيث يتم إزاحة dest لليمين بمقدار count (اي مقدار 1 أو بمقدار محتوى CL اذا كان count يعني CL). وأما بالنسبة للخانات المفرغة من أقصى اليسار فيتم ملئها بالقيمة الثنائية 0 وكذلك فإنه يتم الاحتفاظ بالثنائيات الخارجة من أقصى اليمين من الراية CF

مثال (56) صفحة 420

4) تعليمة الازاحة المنطقية لليسار SHL(Shift Logical Left)

والصيغه العامه لهذه التعليمة هي :SHL dest,count

يشبه عملها عمل التعليمة SAL .

تجدر الإشارة إلى أنه يمكن استخدام تعليمات الازاحة لليسار لمضاعفة الرقم في حال عدم فقدان الرقم 1 في أقصى اليسار. كذلك تستخدم تعليمات الازاحة لليمين لقسمة العدد على 2 في حال فقدان الرقم 1 من أقصى اليمين.

وعلى سبيل المثال يمكن استخدام تعليمة الازاحة SHL مع التعليمة المنطقية OR لتحويل عدد مكون من رقمين ممثل بالنظام العشري الثنائي غير المضغوط (حيث يخزن AL الرقم الأقل أهمية ويخزن BL الرقم الأكثر أهمية) إلى عدد مكافئ ممثل بالنظام العشري الثنائي المضغوط حيث سيتم تخزينة في المسجل AL . والتعليمات الآتية تؤدي هذا الغرض:

MOV CL,4

SHL BL,CL

OR AL,BL

وبالنسبة لتأثير تعليمات الازاحة على الرايات فهو مشابه للتعليمات المنطقية.

ج) تعليمات الدوران Rotate Instructions

تشبه هذه التعليمات تعليمات الازاحة من حيث عدد المعاملات ونوعها . الا انها تختلف عنها في طريقة الأداء.

توجد اربع تعليمات من تعليمات الدوران حيث يمكن تقسيمها إلى نوعين رئيسين هما: -تعليمات الدوران بدون راية حمل -تعليمات الدوران مع راية الحمل ، حيث يشترك CF في حلقة الدوران والتعليمات الأربع هي :

1) تعليمة الدوران لليمين ROR(Rotate Right)

والصيغه العامة لهذه التعليمة هي:ROR dest, count

حيث أن dest وcount كما في تعليمات الازاحة . والمثال (57) يبين عمل هذه التعليمة صفحة 422

2) تعليمة الدوران لليسار ROL(Rotate Left)

الصيغه العامة لهذه التعليمة هي : ROL dest,count

حيث أن dest و count كما هو الحال مع تعليمات الازاحة والمثال (58) يبين عملها . صفحة 422

3) تعليمة الدوران لليمين مع راية الحمل RCR(Rolate Right through carry

والصيغة العامة لهذه التعليمة هي: RCR dest,count

والمثال (59) يبين عملها . صفحة 423

4) تعليمة الدوران لليسار مع راية الحملRCL(Rotate Left through carry)

الصيغة العامة لهذه التعليمة هي : RCT dest,count

حيث أن dest و count كما هو الحال مع التعليمات السابقة. والمثال (60) يبين عملها . صفحة 423

ملاحظة : صفحة 424 * تعني أن القيمة الثنائية للراية قد تتغير حسب نتيجة العملية . _ تعني أن قيمة الراية لا تتأثر. ؟ تعني أن قيمة الراية غير معرفة بعد التنفيذ.

جدول صفحة 424 ملخص تعليمات معالجة البيانات الثنائية وتأثيرها على الرايات المختلفة.

3) يمكن إستبدال التعليمة REP MOVSB

REP MOVS dest, source code

حيث أن كليهما يؤدي إلى نفس النتيجة، إلا أن إستخدام الصيغة الأولى أكثر كفاءة

حيث يوفر جهدا على الأسمبلر ويعفية من فحص نوع المعاملات source و dest لتحويل

التعليمة إلى الشكل الأول والذي يعرفه المعالج الدقيق .

ب) تعليمات المقارنة Compare String Instructions

يوجد ثلاثة أشكال لتعليمات المقارنة كما هو الحال مع تعليمات التحريك . والتعليمات

الثلاث هي CMPS

،

CMPSB

،

CMPSW حيث أن الشكل CMPS يحتاج إلى معاملين أما

الشكلين الآخرين فيستعملان بدون معاملات حيث أن المعاملات ضمنية كما هو الحال مع

تعليمات التحريك .

الصيغة العامة للتعليمة CMPS هي: CMPS dest

source

حيث أن source و dest كما هو الحال مع تعليمات التحريك . وعند الحاجة إلى مقارنة

مجموعة من العناصر في المعاملين مع بعضهما البعض يتطلب ذلك إستخدام إحدى بادئات

تكرار REPE (أو مكافئتها REPRZ) أو REPNE (أو مكافئتها REPNZ). أما بالنسبة

استخدام البادئة REP مع تعليمات المقارنة ليس له معنى منطقي لكون أن التكرار في هذه

الحالة يؤدي إلى إستمرارية المقارنة بغض النظر عن قيم العناصر المقارنة

ترمز باديات التكرار أعلاه إلى ما يلي:

Repeat while Equal..REPE

Repeat while Zero Flag is Set...REPZ

Repeat while Not Equal...REPNE

Repeat while Zero Flag is Not Set..REPNZ

وعند إستعمال البادئة REPE أو مكافئتها REPZ

تعليمات المقارنة فإن المعالج

يقارن العناصر المتناظرة من المعاملين source و dest (حيث يستخدم المسجل SI للتأشير على

عناصر المعامل source والمسجل DI للتأشير على عناصر المعامل dest) . وتتوقف عملية

المقارنة إما بعد أن تصبح قيمة المسجل CX مساوية للصفر أو عند مقارنة عنصرين متناظرين

من Source و dest غير متساويين .

وهذا يعني إستمرار المقارنة طالما أن قيمة المسجل CX لا تساوي الصفر وأن العناصر المتناظرة من source

dest عند مقارنتها كانت متكافئة ، وبالطبع عند مقارنة عنصرين متكافئين فإن راية الصفر توضع في الحالة "1"، لذلك فإن هذه البادئة مكافئة للبادئة REPZ. أما بالنسبة اللبادئة REPNE ومكافئتها REPNZ فتعمل خلافا للبادئة REPE، أي أن تكرار تنفيذ تعليمة

المقارنة سوف يستمر طالما أن قيمة المسجل CX لا تساوي الصفر وأن العناصر المتناظرة من source

و dest والتي تم مقارنتها لغاية الآن لم تكن متكافئة. لذلك فإن عملية المقارنة تتوقف إما عندما

تصبح قيمة المسجل CX مساوية للصفر، أو عندما يتم مقارنة عنصرين متكافئين. تجدر الإشارة

إلى أن تعليمات المقارنة الخاصة بسلسلة الرموز تختلف في طريقة عملها عن تعليمة CMP التي مر

ذكرها. إذ تعمل تعليمات مقارنة سلسلة الرموز بطرح قيمة dest من قيمة source بعكس تعليمة

CMP التي تطرح قيمة Source من قيمة dest. لذلك فإن تعليمات نقل التحكم المشروط المستخدمة

بعد تعليمات مقارنة سلاسل الرموز يجب أن تختلف عن تلك التي تتبع تعليمة CMP.

الرجوع الي مثال 62

بالرجوع إلى مجموعة التعليمات يرجى ملاحظة ما يأتي :

1) بما أنه تم وضع راية تحديد الإتجاه في الحالة "o" في التعليمة الأولى، لذلك فإنه في

كل مرة يتم تنفيذ التعليمة CMPS يتم زيادة قيمة المسجلين SI و DI إما بالقيمة "1"،

إذا كان source

dest من النوع Byte، أو بالقيمة "2"إذا كانا من النوع Word . 429

لذلك فإن SI و DI بعد إنتهاء التنفيذ سيؤشران على العنصر التالي للعنصر المطلوب،

فإذا أردنا SI و DI أن يؤشرا على العنصر المطلوب يجب إضافة التعليمتين

SUB SI, 1

SUB DI, 1

إلى نهاية التعليمات أعلاه إذا كان المعاملين source و dest من النوع Byte، أو إضافة

التعليمتين :

SUB SI, 2

SUB DI, 2

إذا كان المعاملين source و dest من النوع Word . بذلك يصبح كل من المسجلين SI و

DI يشير إلى موقع العنصر المطلوب .

2) إذا كان المعاملين source و dest من النوع Byte من المفضل إستخدام الشكل

CMPSB بدلا من CMPS ، أما إذا كان المعاملين من النوع Word فيجب إستخدام

الشكل CMPSW والسبب في ذلك أن إستخدامهما أكثر كفاءة حيث يوفر الجهد على الأسمبلر عند ترجمة البرنامج المصدري .

3) يمكننا معرفة فيما إذا كان شرط توقف عملية المقارنة (أي إيجاد عنصرين متناظرين

من عناصر source dest غير متساويين) قد تحقق عن طريق فحص محتوى المسجل CX. فإذا كان محتوى المسجل CX مساوية للصفر فهذا يعني أن عملية المقارنة قد استنفذت جميع عناصر المعاملين source و dest دون ايجاد عنصرين غير متكافئين ، أي أن جميع العناصر المتناظرة في كلا المعاملين متكافئة . أما إذا كان محتوى المسجل CX لا يساوي الصفر فهذا يعني أنه تم إيجاد عنصرين متناظرين غير متماثلين وهذا بدوره أدى إلى توقف عملية المقارنة .

4) لو فرضنا أننا نرغب في إيجاد أول عنصرين متكافئين بدلا من غير متكافئين فالتعديل الوحيد على مجموعة التعليمات السابقة هو إستبدال بادئة التكرار REPE بالبادئة REPNE.

ح) تعليمات إستقصاء سلسلة من الرموز Scan String Instructionsتستخدم هذه التعليمات الاستقصاء سلسلة من الرموز للبحث عن رمز معين . ويستخدم المعالج مسجل القطاع الإضافي لحساب عنوان سلسلة الرموز. ففي حال وجود سلسلة الرموز هذه في قطاع البيانات المسمى DSEG على سبيل المثال، يجب تخزين عنوان بداية هذا القطاع في المسجل ES وذلك باستخدام التعليمتين التاليتين :

MOV AX, DSEG

MOVES, AX

كذلك وقبل تنفيذ تعليمات الإستقصاء يجب أيضا تخزين إزاحة بداية سلسلة الرموز في المسجل DI لكون أن المعالج يستخدم هذا المسجل ضمنا عند تنفيذ تعليمات الإستقصاء .

كذلك وقبل البدء بعملية إستقصاء سلسلة الرموز يجب تخزين الرمز المطلوب الإستقصاء عنه إما في المسجل AL ، إذا كانت الرموز معرفة على أنها من النوع Byte، أو في المسجل AX، إذا كانت معرفة على أنها من النوع Word وذلك لمقارنته مع عناصر سلسة

الرموز. لذلك تعتبر هذه التعليمات وكأنها تعليمات مقارنة بين محتويات المسجل AL (أوAX) وعناصر سلسلة رمزية. ولهذا السبب فإن تأثير هذه التعليمات على الرايات مشابه التعليمات مقارنة سلاسل الرموز.

وكما هو الحال مع تعليمات المقارنة، يمكن إستخدام بادئات التكرار REPZ/REPE

REPNZ/ REPNE

مع تعليمات الإستقصاء كذلك يوجد ثلاثة أشكال التعليمات الإستقصاء هي:

SCAS dest

SCASB

SCASW

وتعمل هذه التعليمة عند تنفيذها على مقارنة محتويات AL (في حالة تعريف destعلى أنه من النوع Byte) أو AX (في حالة تعريف dest من النوع Word) مع العنصر المشار إليه بالمسجل D1 من عناصر dest. ولحساب عنوان هذا العنصر يستخدم المعالج مسجل

القطاع الإضافي ES لهذا الغرض .

يتضح أن تأثير هذه التعليمة يكون على راية الصفر، أي أنه في حالة تساوي القيمة المخزنة في AL (أو AX) مع محتوى العنصر المشار إليه بالمسجال DI توضع راية الصفر في الحالة "1" وإلا فإنها توضع في الحالة "O" .

وبعد تنفيذ هذه التعليمة تزداد قيمة المسجل DI مقدار 1 إذا كانت العناصر من النوعByte أو بمقدار 2 إذا كانت العناصر من النوع Word وكانت راية تحديد الإتجاه في الحالة 0.

أما إذا كانت راية تحديد الإتجاه في الحالة "1" فيتم تنقيص قيمة المسجل DI بدلا من زيادته . وكما أسلفنا يفضل إستخدام الصيغة SCASB في حال كون العناصر من النوع Byteأو إستخدام الصيغة SCASW في حال كون العناصر من النوع word.

431

الرجوع لمثال 63 +تدريب 38

د) تعليمات تحميل عنصر من عناصر سلسلة رمزية إلى المسجل AL أو AX(Load string Instructions)

كما هو الحال مع التعليمات السابقة تشمل هذه التعليمات على ثلاثة أشكال هي:

LODS (Load string Instruction) تعليمة

الصيغة العامة لهذه التعليمة هي:

LODS source

حيث أن Source يمثل سلسلة رمزية غالبا ما تكون معرفة في قطاع البيانات، حيث يستخدم المعالج الميكروي مسجل قطاع البيانات في عملية حساب العنوان الحقيقي للعناصر .

تستخدم هذه التعليمة لنقل محتويات العنصر المشار إليه بالمسجل SIعناصر

source إلى المسجل AL (إذا كانت السلسلة الرمزية معرفة على أنها من النوع Byte) أو

432

إلى المسجل AX (إذا كانت سلسلة الرموز معرفة على أنها من النوع Word) . بعد ذلك يتم

تعديل محتوى المسجل SI بالزيادة أو النقصان (حسب حالة راية تحديد الإتجاه) بمقدار 1 أو 0

وذلك حسب نوع السلسلة الرمزية أي من النوع Byte أو كلمة) .

LODSB (Load Byte String) تعليمة

وتستعمل مع سلسلة رمزية من النوع Byte. والصيغة العامة لهذه التعليمة هي:LODSB

وهذا الشكل لا يحتاج إلى معاملات حيث أن المعاملات معرفة ضمناLODSW (Load word String) تعليمة

وتستعمل مع سلسلة رمزية من النوع Word حيث لا تحتاج هذه التعليمة أيضا إلى

معاملات. والصيغة العامة لهذه التعليمة هي: LODSW

الرجوع ل مثال 64 433 الرجوع ل تدريب 39

ه) تعليمات التخزين إلى سلسلة رمزي Store String Instructions

تشمل هذه التعليمات على ثلاثة أشكال يمكن للأسمبلر التعرف عليها وهي :

STOS (Store into String) تعليمة

الصيغة العامة لهذه التعليمة هي: STOS dest

حيث أن : = Worddest أو Byte مثل سلسلة من الرموز من النوع وغالبا ما تكون معرفة في القطاع الإضافي .

يؤدي تنفيذ هذه التعليمة إلى نقل محتوى المسجل AL (إذا كانت سلسلة الرموز من النوعByte) أو محتوى المسجل AX (إذا كانت سلسلة الرموز من النوع Word) إلى موقع الذاكرة المشار إليه بالمسجل DI. وللتخزين في الموقع الأول من مواقع سلسلة الرموز المعنونة بالإسم desi يجب تخزين إزاحة هذا العنصر في المسجل DI قبل التنفيذ. ولحساب العنوان الحقيقي العنوان الموقع الذي سيتم التخزين فيه ، يستخدم المعالج مسجل القطاع الإضافي ES والمسجل DI .

وبعد تنفيذ هذه التعليمة يتم زيادة أو انقاص قيمة المسجل DI (حسب حالة راية تحديد الإتجاه إذا كانت "0" أو "1") بمقدار 1 أو 2 حسب نوع سلسلة الرموز (Byte أو Word) .

يمكن إستخدام هذه التعليمة مع بادئة التكرار REP وذلك لتخزين محتوى المسجلAL أو AX إلى مجموعة من عناصر dest. وفي هذه الحالة يجب تخزين عدد هذه العناصر في المسجل CX.

الرجوع ل مثال 65 434

STOS (Store AL into String of type Byte) تعليمة

الصيغة العامة لهذه التعليمة هي : STOSB

اذ تعتبر هذه التعليمة مكافئة للتعليمة السابقة عندما تكون السلسلة الرمزية من النوعByte ويفضل إستخدامها كما هو الحال مع التعليمات السابقة .

STOSW (Store AX into String of type word) تعليمة

تعتبر هذه التعليمة مكافئة للتعليمة STOS عندما تكون عناصر السلسلة الرمزية

من النوع Word، ويفضل إستخدامها على التعليمة STOS وذلك لأن الأسمبلر يعمل

على تحويل التعليمة STOS إلى STOSW في حال كون عناصر السلسلة الرمزية من النوع

Word . أما في حال كون العناصر من النوع Byte فيقوم الأسمبلر بتحويل التعليمة STOS

إلى التعليمة STOSB . 435

الصيغة العامة لهذه التعليمة هي: stows

جدول(13) صفحة 436 تدريب(40)

****تعليمات التحكم بالمعالج processor control instructions**

*يحتوي طاقم تعليمات لغة اسمبلي على مجموعة من التعليمات يستطيع المبرمج عن طريقها التحكم بالمعالج 8088/8086 وتنظيم اسلوب عمله وذلك بتضمين هذه التعليمات في برنامجه المكتوب بلغة اسمبلي.

***يمكن تقسيم تعليمات التحكم بالمعالج الدقيق الى ثلاث مجموعات رئيسية هي:**

1. تعليمات خاصة بمعالجة الرايات flags manipulation instructions

*ويوجد ضمن هذا النوع سبع قيمات تؤثر على قيم الرايات هي: جدول رقم 14

*يتضح من هذه التعليمات أنه بامكان المبرمج التحكم بالمعالج بواسطة تعديل حالة رايات الحمل CF تحديد الاتجاه DF, تحديد الاعتراض IF, *فعلى سبيل المثال يمكن استخدام التعليمات الخاصة براية الحمل لاعداد هذه الراية تمهيدا لتنفيذ احدى تعليمات الدوران أو أي تعليمات أخرى قد تتطلب وضع راية الحمل في حال معينة صفر أو واحد.

*أما بالنسبة ل التعليمتين CLD و STD والمتعلقة براية تحديد الاتجاه فيمكن استخدامها لوضع راية تحديد الاتجاه في الحالة (0) أو الحالة(1) حيث تحدد هذه الحالة اتجاه معالجة سلاسل الرموزز (من أقصى اليسار الى اليمين أو من أقصى اليمين الى اليسار) كما هو مبين في مثال 6 صفحة 438 .

*أما بالنسبة ل التعليمتين CLD و STDفيستطيع المبرمج استخدامها للتحكم بالمعالج لتحديد امكانية الاعتراضات أو اهمالها.

فعند وضع راية الاعتراض في الحالة صفر فان المعالج سوف يهمل أو يؤجل معالجة الاعتراضات المقنعة (maskable interrupts)

*اما في الحالة (1(فان المعالج سوف يقوم بمعالجة الاعتراضات باعنواعها.

*أما بالنسبة للاعتراضات غير المقنعة فلا يمكن اهمالها أو تأجيلها حتى ولو كانت راية الاعتراض في الحالة 0

2. تعليمات التزامن الخارجي extremal synchronization instructions

*تستخدم لتنظيم عمل المعالج بمزامنته مع الاحداث الناتجة عن ملحقات الحاسب الخارجية مثل أجهزة الادخال والاخراج على سبيل المثال لا الحصر.

***وتعتبر التعليمات الأربعة التالية من اهم تعليمات التزامن الخارجي**

1. تعليمة التوقف الخامل HLT

*يؤدي تنفيذ هذه التعليمة الى وضع المعالج الدقيق 8086/8088 في حالة التوقف الحامل دون أي عمل. ويمكن اخراج المعالج من هذه الحالة باحدى طريقتين وذلك اما باعادة المعالج الى حالة البدء (الوضع المبدئي) عن طريق مفتاح خاص بذلك يطلق عليه (reset switch) , أو في حالة وصول اعتراض خارجي غير مقنع أو مقنع (شريطة أن تكون راية الاعتراض في الحالة 1) .

2. **تعليمة الانتظار WAIT** يؤدي تنفيذ هذه التعليمة لوضع المعالج في حالة التوقف النشط وسمي هذا التوقف نشطا لان المعالج يقوم أثناء توقفه بفحص خط من خطوط المعالج المسمى TESt على فترات منتظمة طول كل منها 5 نبضات, *ويخرج المعالج من حالة التوقف عندما يصبح هذا الخط في الحالة النشطة *ومن العوامل التي تؤدي الى تنشيط هذا الخط انهاء أحد الأجهزة الخارجية مثل وحدات الادخال او الاخراج او المعالج 8087 في حال وجوده

*ففي هذه الحالة يتم ارسال نبضة الى المعالج على الخط وبذلك يخرج المعالج من حالة التوقف لمتابعة التنفيذ من حيث توقف.

3. **تعليمة الافلات (Escape) ESC** *تستخدم هذه التعليمة في انظمة الحاسوب متعددة المعالجات مثل 8086 مع المعالج المساعد 8087 بتضمينها في برنامج لغة اسمبلي لتمكين المبرمج من تنفيذ تعليمات معالج اخر. *الصيغة العامة لهذه التعليمة هي ESC opcode source *حيث أن opcode يمثل رمز التعليمة المرسلة للمعالج المعني مثل المعالج 8087

Source يمثل معامل التعليمة المطلوبة حيث يمكن أحد المسجلات أو مواقع الذاكرة.

4. **تعليمة الحجز LOCK** *تسبق هذه البادئة احدى التعليمات بقصد حجز خطوط الناقلة لفترة زمنية مساوية لمدة تنفيذ هذه التعليمة

*وفي أثناء هذه الفترة أي قبل الانتهاء من تنفيذ التعليمة من غير الممكن لأي معالج اخر استعمال الناقلة.

ج) التعليمة NOP *يتبين من اسم هذه التعليمة من انها عملية لا تؤدي الى حدوث أي فعل في الحاسوب سوى استهلاك الوقت.

*والمسجل الوحيد الذي يتأثر من تنفيذ هذه التعليمة هو مؤشر التعليمة IP.

***يمكن استخدام التعليمة NOP للاغراض التالية: 1**. برمجة الفترات الزمنية. Taime Delays

2. تعديل البرنامج الهدفي باستبدال تعليمة أو أكثر بشيفرة التعليمة NOP أي H90 ثم تنفيذه دون اعادة الترجمة

الإعتراض Interrupt يطلق على إعتراض سير تنفيذ برنامج أو روتين معين بواسطة أي عامل خارجي وقد يكون الإعتراض مقصودا بقصد إعلام المعالج بحدث معين الاتحاد الإجراء المناسب تبعا لذلك . وعند الانتهاء من معالجة الإعتراض يتابع البرنامج أو الروتين تنفيذه من نفس النقطة التي تم الاعتراض عندها .

التعليمة Instruction مجموعة رموز تعرف عملية من العمليات التي يقوم بها الحاسوب ومن الممكن أن تكون هذه التعليمة بلغة الآلة أو بلغة البرمجة

شيفرة إبسيدك EBCDIC نظام آخر من أنظمة التشفير حيث يمثل الرمز الواحد ب 8 ثنائيات .

شيفرة أسكي ASCI Code النظام الشيفري المعياري الأمريكي لتبادل المعلومات حيث تمثل كل رمز باستخدام سبعة أرقام ثنائية .

العنوان الحقيقي Physical Address يطلق على العنوان المطلق لموقع من مواقع الذاكرة ويتم حسابه عن طريق إضافة

الإزاحة (أو العنوان الفعال لهذا الموقع إلى أحد مسجلات القطاع بعد ضرب محتويات القطاع المعني بالعدد 16.

العنوان الفعال Effective Address يطلق علي بعد موقع من مواقع الذاكرة من بداية القطاع المعرف فيه هذا الموقع . وغالبا ما يدعى بالإزاحة ويحدد بعدد البايتات

العنونة Addressing يقصد بالعنونة الطريقة المستخدمة لتخزين البيانات اللازمة لإجراء العمليات . ويوجد عدة طرق للعنونة منها العنونة الفورية وباستخدام المسجلات وإستخدام مواقع الذاكرة .

نظام الإشارة والمقدار Sign Magnitude أحد الأنظمة المستخدمة لتمثيل البيانات الرقمية الصحيحة بإشارة داخل ذاكرة الحاسوب .

النظام العشري المرمز الثنائي Binary Coded Decimal نظام خاص بتمثيل الأرقام العشرية . حيث يمثل كل رقم عشري بأربعة أرقام ثنائية .

نظام المكمل لإثنين sComplement'2 نظام آخر يستخدم لتمثيل البيانات الرقمية الصحيحة (Integers) مع إشارتها داخل ذاكرة الحاسوب ومسجلاته، وهو من الانظمة الاكثر انتشارا .

نظام المكمل لواحد sComplement'1 نظام ثالث يستخدم لتمثيل البيانات الرقمية الصحيحة مع إشارتها داخل ذاكرة الحاسوب ومسجلاته.

الوحدة السابعة : برمجة عمليات الإدخال والإخراج

2. أساليب برمجة عمليات الإدخال/الإخراج

- ✓ في بداية ظهور الحواسيب تميزت التطبيقات البرمجية باحتوائها بشكل رئيسي على عمليات حسابية ونسبة قليلة من عمليات الإدخال والإخراج .
- ✓ العمليات الحسابية تنفذ في وحدة المعالجة المركزية CPU في وحدات الإدخال والإخراج .
- ✓ من تطبيقات عمليات الإدخال والإخراج: برامج حجز وإصدار تذاكر السفر, برامج معالجة النصوص, برامج الرسم والتصميم.
- ✓ من أساليب إدخال المعطيات إلى البرامج المكتوبة بلغات البرمجة المختلفة ومنها لغات الأسمبلي :

1)طريقة ستاتيكية

- ❖ حيث تعرف المعطيات في البرنامج بواسطة جمل خاصة مثل جمل الإسناد أو التخصيص ,وهي مرحلة تنفيذ البرنامج تتم معالجة المعطيات والحصول على النتائج دون أي تدخل من المبرمج أو المستخدم .
- ❖ من أهم مساوئ الطريقة الاستاتيكية أ، البرامج أن البرامج التي تستخدم هذا الأسلوب لا يمكن استعمالها لمعالجة معطيات أخرى بدون تغيير فيها وإعادة ترجمتها وتحريرها وذلك لأن المعطيات مثبتة في البرنامج.
- ❖ لمعالجة بيانات أخرى يلزم الرجوع إلى البرنامج وتعديل جمل تعريف المعطيات فيها بحيث تحتوي المعطيات الجديدة . ومن ثم ترجمة هذه البرامج وتحريرها وأخيراً تنفيذها .
- ❖ تمتاز البرامج التي تستخدم الأسلوب الاستاتيكي في تحديد المعطيات بعدم شموليتها .
- ❖ يستخدم الأسلوب الاستاتيكي في التطبيقات التي تعالج معطيات محددة .

2) طريقة ديناميكية

- ❖ هي الأكثر شيوعاً , حيث تسمح بإدخال المعطيات بشكل مستقل عن البرنامج.
- ❖ وهنا يكون البرنامج منفصل عن المعطيات ويتم إدخال المعطيات اللازم معالجتها إلى البرنامج في مرحلة التنفيذ.
- ❖ لمعالجة معطيات جديدة يلزم فقط إعادة تنفيذ البرنامج وإدخال تلك المعطيات في مرحلة التنفيذ عند الحاجة اليها وطلبها من قبل البرنامج .

2.1 تنظيم عمليات الإدخال /الإخراج

- تنفيذ العمليات في المعالج الميكروي تتم بسرعات فائقة جداً مقارنة مع سرعة تنفيذ العمليات في وحدات الإدخال والإخراج . وهذا يؤدي إلى عدم ربط وحدات الإدخال والإخراج مباشرة مع المعالج الدقيق .
- لتقليل الفرق الشاسع في السرعات بين المعالج الميكروي من جهة و وحدات الإدخال والإخراج من جهة أخرى تم استخدام وحدات خاصة تسمى منافذ التوسط Interface Ports .
- منافذ التوسط Interface Ports : هي عبارة عن مسجلات من نوع خاص يتم الوصول إليها بوساطة العناوين كما هو الحال مع مواقع الذاكرة الرئيسة .
- كل وحدة إدخال أو اخراج تمثل بواسطة عدد معين من المنافذ ، ومجموع المنافذ التي يستطيع المعالج الميكروي التعامل معها يسمى حيز عناوين الإدخال والإخراج I/O address space .
- يخصص ثلاثة منافذ للآلة الطابعة وهي:
  - منفذ المعطيات ويحمل الرقم 3BCH
  - منفذ التحكم ويحمل الرقم 3BEH
  - منفذ الحالة ويحمل الرقم 3BDH
- لوحة المفاتيح تمثل بواسطة ثلاثة منافذ :
  - منفذ المعطيات ويحمل الرقم 60H
  - منفذ التحكم ويحمل الرقم 61H

- منفذ الحالة ويحمل الرقم 62H

- تصل سعة حيز عناوين الادخال/الاخراج في حواسيب 8086/8088 إلى 64 كيلو بايت .
- يمكن الوصول الى أي منفذ من بواسطة عنوانه الخاص به
- يصل حجم حيّز عناوين مواقع الذاكرة إلى 1 M ميجا بايت .
- طرق تنظيم عناوين مواقع الذاكرة و عناوين الادخال/الاخراج :

1) أسلوب فصل الحيزين I/O-Mapped I/O address space

- هذه الطريقة يخصص لمنافذ حيز الادخال والاخراج مجالاً مستقلا عن المجال المخصص لمواقع الذاكرة الرئيسة ، انظر شكل 1 صفحة 468 .
- يستخدم اسلوب فصل الحيزين في العديد من الحواسيب مثل الحواسيب الشخصية المبنية باستخدام المعالجات من نوع إنتل 8086-80486 .

2) أسلوب دمج الحيزين Memory-mapped I/O address space

- يتم تخصيص منطقة في الذاكرة الرئيسية لأغراض حيز الادخال والاخراج ، انظر الشكل 2 صفحة 469 .
- يستخدم أسلوب دمج الحيزين في عدة حواسيب منها : حواسيب VAX والحواسيب الشخصية المبنية بإستخدام المعالجات من نوع موتورولا .

- من أهم الفروق بين أسلوبي تنظيم حيز الادخال والاخراج و حيز مواقع الذاكرة :

a. يسمح أسلوب دمج الحيزين عنونة عدد كبير من منافذ وحدات الادخال والاخراج بحيث يصل عدد المنافذ إلى سعة الذاكرة الرئيسية ،أي عدد المنافذ ممكن أن يساوي عدد مواقع الذاكرة الرئيسة .
b. يتطلب أسلوب فصل الحيزين استخدام تعليمات خاصة (مثل تعليمات IN, OUT) ' للتعامل مع وحدات الادخال والاخراج على عكس اسلوب دمج الحيزين الذي يوفر إمكانية استخدام تعليمات نقل البيانات للتعامل مع وحدات الإدخال والإخراج كما هو الحال في التعامل مع مواقع الذاكرة الرئيسية.
c. ونتيجة لذلك فإن أسلوب دمج الحيزين يمتاز بمرونة عالية بسبب إمكانية استخدام معظم طرق العنونة مع المنافذ في حين أن أسلوب فصل الحيزين يستخدم فقط طريقة واحدة لعنونة البيانات وهي العنونة باستخدام ارقام المنافذ.
d. يوفر أسلوب دمج الحيزين امكانية إجراء العمليات الحسابية والمنطقية على المعطيات بالاضافة إلى عمليات الإدخال والإخراج ، أي أن القيم المخزنة في المنافذ يمكن أن تشارك في معظم العمليات بنفس الطريقة كما هو الحال مع مواقع الذاكرة الرئيسية .

- في الحواسيب التي تستخدم أسلوب فصل الحيزين تستخدم إشارات التحكم التالية لتحديد نوع القيمة الموضوعة على خطوط العناوين هل هي رقم موقع في الذاكرة الرئيسية أم رقم منفذ إدخال أو إخراج , وهي :
  - IOR : إشارة قراءة محتويات منفذ الإدخال (READ) .
  - IOW : إشارة تخزين قيمة منفذ الإخراج (WRITE) .
  - MEMR : إشارة قراءة محتويات موقع في الذاكرة الرئيسة (READ) .
  - MEMW : إشارة تخزين قيمة في موقع في الذاكرة الرئيسة (WRITE) .
- أما في الحواسيب التي تستخدم أسلوب دمج الحيزين فلا يوجد ضرورة لوجود إشارات تحكم لتحديد هل القيمة الموضوعة على خطوط العناوين تمثل رقم موقع في الذاكرة الرئيسية أم رقم منفذ إدخال أو إخراج , وذلك بسبب تخصيص عناوين للمنافذ تختلف عن عناوين مواقع الذاكرة ,,, يلزم هنا فقط إشارة لتحديد نوع العملية: عملية قراءة أم عملية تخزين . وهذا يتم بواسطة التعليمة نفسها : IN تعليمة الادخال (تخزين) , تعليمة OUT تعليمة الإخراج (قراءة) .
- طرق برمجة عمليات الإدخال والإخراج في الحواسيب الشخصية المبنية بواسطة المعالجات 8086/8088 والتي تستخدم أسلوب فصل الحيزين :-

2.2عمليات الإدخال والإخراج المبرمجة Programmed I/O Operations

- يتولى المعالج حسب هذه الطريقة إصدار إشارات التحكم اللازمة لتنفيذ جميع مراحل عمليات الإدخال والإخراج والتي تشمل ما يلي :
  - إشارة بدء العملية .
  - إشارات التحكم اللازمة لتنفيذ خطوات العملية .
  - إشارة انتهاء العملية .

- وهذا يعني أن عملية الإدخال والإخراج تنفذ بإشراف كامل ومباشر من قبل المعالج , وهذا يتطلب استخدام المكونات التالية :
  - مسجل الحالة Status register
  - مسجل البيانات Data register
  - عداد البيانات Data counter
  - مؤشر البيانات Data pointer
- يستخدم مسجل الحالة لتخزين الوضع الراهن لوحدة الإدخال أو الإخراج ووضع البيانات المراد نقلها و يستخدم أيضا مسجل الحالة لتحديد حالة البيانات حيث تخصص كل ثنائية فيه لأداء مهمة معينة
- يستخدم مسجل البيانات لتخزين البيانات بشكل مؤقت .
- يستخدم عداد البيانات لتخزين عدد الرموز المطلوب نقلها , حيث يتم طرح 1 من القيمة المخزنة في العداد بعد نقل كل رمز .
- يؤشر مؤشر البيانات إلى موقع الذاكرة الرئيسية المطلوب قراءة محتوياته أو إلى موقع الذاكرة المطلوب التخزين فيه .
- انظر شكل (3) صفحة 472 والذي يبين مخطط سير العمليات لعمليات الإدخال والإخراج المبرمجة .
- أسلوب عمليات الإدخال والإخراج المبرمجة يقلل من فعاليّة استخدام المعالج ووحدات الإدخال والإخراج وذلك لأنه يقضي وقتاً كبيراً في فحص حالة وحدة الإدخال أو الإخراج وانتظار جاهزية هذه الوحدة .
- انظر شكل (4) صفحة 473 والذي يبين إرسال الحرف 'A' إلى الآلة الطابعة حسب أسلوب الإدخال والإخراج المبرمج .

2.3عمليات الإدخال والإخراج الموجهة بالاعتراضات Interrupt-driven I/O Operations

- يهدف أسلوب الإدخال/الإخراج الموجهة بالاعتراضات إلى تحرير المعالج من مسؤولية الإشراف المباشر على جميع مراحل تنفيذ عمليات الإدخال/الإخراج .
- في أسلوب الإدخال/الإخراج المبرمج حيث يبقى المعالج في حالة انتظار لحين انتهاء العملية ووصول إشارة جاهزية وحدة الإدخال/الإخراج , أما في أسلوب الموجهة بالاعتراضات فإن المعالج لا يدخل في حالة انتظار بل يستغل معظم الوقت في تنفيذ عمليات أخرى لحين وصول جاهزية وحدات حيث ترسل هذه الوحدة إشارة اعتراض إلى المعالج تخبره بذلك .
- انظر شكل (5) والذي يبين مخطط سير العمليات لعمليات الإدخال والإخراج الموجهة بالاعتراضات .
- يستخدم المعالج أسلوبين لتحديد وحدة الإدخال/الإخراج التي أصدرت اشارة الاعتراض وهما :
  1. أسلوب الإستفتاء Polled-interrupt technique
  2. أسلوب المتجهات Vectored-interrupt technique
  - في اسلوب الاستفتاء تمر إشارة الاعتراض الصادرة عن وحدات الادخال/الاخراج عبر دائرة الجمع المنطقي OR الى مدخل الاعتراضات في المعالج (مدخل INTR) .
  - ويخصص ثنائية واحدة لتحديد حالة كل وحدة من وحدات الإدخال والإخراج من حيث هل أصدرت هذه إشارة اعتراض ام لا ؟ وتسمى هذه الثنائية ثنائية الحالة Status bit.
  - انظر شكل 6 صفحة 476
  - ملاحظات عن استدعاء الاعتراضات :
    1. يتم استدعاء الاعتراض بواسطة التعليمة INT
    2. الصيغة العامة لتعليمة INT هي : INT interrupt_code
    3. تتراوح قيمة كود الاعتراض بين صفر و 255
    4. يستخدم كود الاعتراض لتحديد عنوان برنامج خدمة الاعتراض المطلوب
    5. يستخدم برنامج خدمة الاعتراض لتنفيذ مهمة واحدة أو عدة مهام .
    6. يتم تحديد المهمة المطلوبة في الاعتراض بواسطة رقم المهمة الذي يجب أن يخزن في المسجل AH
    7. تحتاج بعض الاعتراضات إلى إعداد وتحضير مسبق وذلك بتخزين قيم معينة في مسجلات محددة .
  - انظر جدول (1) صفحة 477 والذي يبين الاعتراضات المستخدمة في الحواسيب الشخصية المبنية بواسطة المعالج 8088/8086

- من أهم الاعتراضات التي تستخدم في برمجة عمليات الادخال/الاخراج وهي :
10H , 16H , 17H , 21H .

3.استخدام الاعتراضات في برمجة عمليات الادخال والاخراج

1.3الاعتراض 21H INT

- يعتبر الاعتراض 21H من اهم الاعتراضات وذلك لكونه يقدم عدداً كبيراً من المهمات لنظام الحاسوب والمستخدم على السواء .
- مهمات الاعتراض 21H :
  - المهمات ذات العلاقة بلوحة المفاتيح (قراءة رمز أو سلسلة رموز).
  - المهمات ذات العلاقة بعرض الرموز على الشاشة (عرض رمز واحد أو سلسلة رموز على الشاشة).
  - المهمات ذات العلاقة بالآلة الطابعة (إرسال رموز أو سلسلة رموز إلى الآلة الطابعة).
  - مهمات إدارة الملفات (فتح ملف, إغلاق ملف, خلق ملف, شطب ملف, تغيير اسم ملف, إعطاء مواصفات ملف وتحديثها, تحديد مشغل الاقراص الضمني, حماية ملف, إلغاء حماية ملف,....).
  - المهمات ذات العلاقة بالأدلة (خلق أو شطب دليل التنقل بين الأدلة,... وغيرها).

1.1.3 عرض سلسلة رموز على الشاشة

- ✓ لنفرض أنه لدينا سلسلة من الرموز في قطاع المعطيات تبدأ بالبايت المسمّى MES وتنتهي بالبايت الذي يحتوي رمز إشارة الدولار $

```
MES  DB  'In God We trust$'
```

- ✓ بالرجوع للجدول نجد أن عرض سلسلة رموز الشاشة يمكن أن تنفذ باستخدام المهمة رقم 9 في الاعتراض 21H كما في سلسلة التعليمات الآتية :

```
MOV   AH,9
LEA    DX,MES
INT    21H
```

- ✓ تستخدم التعليمة MOV لتخزين رقم المهمة (9) في المسجل AH
- ✓ تستخدم التعليمة LEA لتخزين قيمة الازاحة لسلسلة الرموز MES في المسجل DX
- ✓ تستخدم التعليمة INT لتنفيذ مهمة عرض الرموز
- ✓ من الضروري احتواء السلسلة الرمزية على إشارة $

2.1.3 إدخال سلسلة رموز بواسطة لوحة المفاتيح

- ☒ إدخال سلسلة رموز من خلال لوحة المفاتيح تؤدي إلى تخزين شيفرة الآسكي داخل منطقة خاصة تسمى منطقة الادخال , والتي تتكون من 3 حقول وهي :
  - حقل الطول الكلي لمنطقة الإدخال (1بايت).
  - حقل الطول الفعلي لسلسلة الرموز المطلوب إدخالها (1بايت)
  - حقل سلسلة الرموز ,أي الحقل المستخدم لتخزين شيفرة الآسكي للرموز المطلوب إدخالها
  - تستخدم الجمل التالية لتعريف منطقة لإدخال سلسلة رموز تمثل إسم طالب بطول لا يزيد عن 30 حرفاً , يحتوي البايت الأول MLEN في المنطقة علة القيمة 32 , أما قيمة الحقل الثاني ALEN فهي غير معروفة

وتحدد لاحقاً بعد الانتهاء من عملية الادخال ,أما الحقل الثالث FIELD فيحتوي على 30 فراغاً يستخدم لتخزين شيفرة الآسكي للرموز عند ادخالها

```
INPUT   LABEL   BYTE
MLEN    DB      32
ALEN    DB      ?
FILD    DB      30 DUP(20H)
```

وبالرجوع إلى جدول المراجع نجد أن مهمة إدخال سلسلة رموز من لوحة المفاتيح تنفذ باستخدام المهمة ذات الرقم 0AH في الاعتراض رقم 21H كما في سلسلة التعليمات التالية :

```
MOV   AH,0AH
LEA   DX,INPUT
INT21H
```

- تستخدم تعليمة MOV لتخزين رقم المهمة 0AH في المسجل AH
- تستخدم تعليمة LEA لتخزين قيمة الازاحة لمنطقة INPUT في المسجل DX
- تستخدم تعليمة INT لتنفيذ مهمة إدخال سلسلة الرموز

2.1.3 إرسال سلسلة رموز إلى الآلة الطابعة

- توصل الآلة الطابعة مع منفذ LPT1 الذي يستطيع تخزين شيفرة الآسكي لرمز واحد فقط
- من أجل إرسال رمز إلى الآلة الطابعة تستخدم المهمة رقم 5 في الاعتراض رقم 21H

لنفرض اننا نود طباعة سلسلة الرموز التالية المعرفة في قطاع المعطيات على النحو التالي :

```
PRT     DB      'JERUSALEM UNIVERSITY'
```

ولارسال هذه الرموز الى الالة الطابعة تستخدم التعليمات الاتية :

الموضحة صفحـــة 483

) الى الالة الطابعة , فانه يلزمDL تستخدم لارسال رمز واحد (محتويات المسجل 21H نظرا لان المهمة رقم 5 ف يالاعتراض رقم

التي تتكون من 20 رمزا . نوضح الان التعليمات السبعة المذكورة سابقا PRTبرمجة دوران يتكرر 20 مرة لطباعة سلسلة الرموز

. لقد تم ترقيم الجمل لاغراض الشرح والتوضيح فقط . PRTواللازمة لطباعة سلسلة الرموز

1CL.) تستخدم الجملة الاولى لتخزين عدد الرموز المراد طباعتها (20) في المسجل

2BX . تستخدم الجملة الثانية لتخزين رقم الازاحة (العنوان) لسلسة الرموز في المسجل

3AH . تستخدم الجملة الثالثة لتخزين رقم المهمة (5) في المسجل

DL.في المسجل 4 PRT) تستخدم الجملة الرابعة لتخزين الرمز التالي من سلسلة

5) تستخدم الجملة الخامسة لتنفيذ مهمة ارسال رمز الى الالة الطابعة عن طريق الاعتراض .

PRT.بمقدار 1 بحيث يؤشر الى الرمز التالي من سلسلة الرموز BX 6) تستخدم الجملة السادسة لزيادة محتويات المسجل

7) تستخدم الجملة الاخيرة لتنظيم الاستمرار في الدوران او الخروج منه .

يتضح من هذا المثال ما يلي :

لارسال رمز واحد الى الالة الطابعة .21Hرقم _ تستخدم المهمة رقم 5 في الاعتراض

. DL_ يخزن الرمز المراد طباعته في المسجل

وللتعرف على بقية المهمات يمكنك عزيزي الدارس الرجوع الى جداول الاعتراضات في المراجع ودراسة امثلة اخرى على استعمال هذه المهمات .

(INT 10H)الاعتراض

بشكل رئيسي لتنفيذ العمليات الاساسية على الشاشة . يمكن تلخيص المهمات الاساسية للاعتراض 10Hيستخدم الاعتراض

على النحو الاتي : 10H

, مسح الشاشة, Resolution_ تحديد وتغيير اسلوب تشغيل الشاشة (تحديد عدد السطور والاعمدة , تحديد درجة حساسية الشاشة

في المكان المطلوب , ...) . (Cursor)وضع المؤشر

_ مهمات معالجة الرموز (قراءة الرموز , تحديد صفات الرموز , ...) .

(قراءة وعرض عناصر الصورة , تغيير الالوان ,..). Graphics_ المهمات ذات العلاقة بالرسومات

_ المهمات ذات العلاقة بتحديد مواصفات الشاشة (تحديد اسلوب عمل الشاشة , تحديد عرض الشاشة , تحديد صفحات الشاشة).

, وتشرح اهم العمليات التحضيرية لبعض المهمات . نتعرف عزيزي10Hتحتوي المراجع عادة على جداول تبين مهمات الاعتراض

. 10Hالدارس الان على المهمات الاساسية لاعتراض

في موقع معين على الشاشة . (Cursor)وضع المؤشر

يلزم في معظم التطبيقات عرض البيانات على الشاشة في مكان معين , ولتحقيق ذلك يجب وضع المؤشر في نقطة معينة على الشاشة.

كما في سلسلة التعليمات الاتية : 10Hولهذا الغرض يمكن استخدام المهمة رقم 2 في الاعتراض رقم

الموضحـــة صفحـــة 485

حيث يؤدي تنفيذ هذه التعليمات الى وضع المؤشر على الشاشة في نقطة تقاطع السطر رقم 13 والعمود رقم 37 .

تصنف الشاشات حسب طريقة تشكيل الرموز التي يمكن عرضها عليها الى نوعين هما :

. Character Screens_ الشاشات الرمزية

. Bit-mapped Screens_ الشاشات النقطية

تستخدم الشاشات الرمزية لعرض الرموز المختلفة مثل : الحروف , والارقام , والرموز الخاصة . ولذلك تسمى هذه الشاشات بالشاشات الابجدية_الرقمية . ونظرا لكون هذه الشاشات تستطيع عرض الرموز فقط فانها تكون مقسمة الى سطور واعمدة كما هو مبين في الشكل (7) صفحة 486 تحتوي الشاشات الاكثر استعمالا على 25 سطرو 80 عمود . ترقم السطور والاعمدة ابتداءً من الرقم صفر.

وتجدر الاشارة هنا الى ان الرموز المعروضة على الشاشة تكون مخزنة في منطقة خاصة في الذاكرة الرئيسية تسمى ذاكرة الفيديو

. تبلغ سعة ذاكرة الفيديو 16 كيلوبايت. يخصص في ذاكرة الفيديو 2 بايت لكل رمز : يحتوي البايت الاول على شيفرة Video RAM

الاسكي للرمز , ويحتوي البايت الثاني على صفات الرمز مثل : لون الرمز, ولون الخلفية , وشدة الاضاءة ,... وغيرها من الصفات . وبما ان الشاشة تتسع ل 80*25 =2000 رمز في نفس الوقت , فانه يلزم 4000 بايت في ذاكرة الفيديو لتخزين شاشة واحدة .

غير ان ذاكرة الفيديو تخصص لتخزين 4 شاشات . ترقم الشاشات ابتداءً من الصفر .

, ورقم العمود يخزن DHوبالرجوع الان الى تعليمات وضع المؤشر في نقطة معينة نجد ان رقم السطر يجب ان يخزن في المسجل

. BH , ورقم الشاشة يخزن في المسجل DLفي المسجل

تستخدم الشاشات النقطية عادة لعرض جميع انواع البيانات من رموز , وصور, ومخططات , ورسومات . ولهذا فهي تسمى بشاشات

. Graphics Screensالرسومات

. تعتمد جودة الصورة المعروضة على الشاشة على عددPexilتتكون الشاشات النقطية من عدد كبير جدا من النقاط التي تسمى

البكسيلات المكونة للشاشة . فكلما زاد عدد البكسيلات في الشاشة كلما زادت جودة الصورة المعروضة . يحدد عدد البكسيلات درجة

والذي يحدد بواسطة بطاقة الفيديو . من بطاقات الفيديو الاكثر استخداما في الوقت الحالي : Resolutionحساسية الشاشة

التي تحدد درجة الحساسية 1024 * 768 بكسيل . VGA , SVGA

وكما هو الحال في الشاشات الرمزية فان البيانات المعروضة على الشاشة تكون مخزنة في ذاكرة الفيديو , حيث يخصص عدد معين من الثنائيات (البتات) لكل بكسيل . ولهذا السبب فان حجم ذاكرة الفيديو في الشاشات النقطية اكبر بكثير من حجم ذاكرة الفيديو في الشاشات الرمزية . فمثلا عند تخصيص 8 ثنائيات لكل بكسيل فان حجم ذاكرة الفيديو يساوي 1024 * 768 * 8 ثنائية, اي حوالي 98 كيلوبايت .

ان استخدام الشاشات النقطية يتطلب من المبرمج اهتمام خاص , حيث يستطيع المبرمج التحكم بكل بكسيل على الشاشة وعليه ايضا ان يختار احد اسلوبي تشغيل الشاشة :

. Aalphanumerical mode_ الاسلوب الابجدي-العددي

. Graphics mode_ اسلوب الرسومات

. وبالرجوع الى المهمة رقم صفر نجد انها 10Hيتم اختيار اسلوب تشغيل الشاشة النقطية بواسطة المهمة رقم 0 في الاعتراض رقم

تحدد ما يلي :

_ اسلوب تشغيل الشاشة (الرسومات او الابجدي العددي) .

_ عدد سطور واعمدة الشاشة .

. Resolution_ درجة حساسية الشاشة

_ عدد الالوان التي يمكن عرضها على الشاشة .

فمثلا تستخدم التعليمات التالية لوضع الشاشة في اسلوب الرسومات , بدرجة حساسية 640 * 480 , وعدد الالوان 16 لون .

كما في صفحة رقم 487

مسح الشاشة :

تستخدم عدة طرق لمسح الشاشة . نتعرف على طريقتين منها :

_ تمسح الشاشة بارسال فراغ الى جميع مواقع الشاشة كما في سلسلة التعليمات التالية :

يوضع المؤشر في السطر صفر والعمود صفر :

كما في صفحـــــة 488

مسح الشاشة بارسال فراغ الى كل موقع في الشاشة :

كمافيصفحـــــة 488

, ولتحقيق هذا الغرض تستخدمScroll down او الى الاسفل Scroll up _ تمسح الشاشة بتحريك جميع سطور الشاشة الى الاعلى

كما في سلسلة التعليمات الاتية : 10Hالمهمة رقم 6 او 7 في الاعتراض

الموضحة صفحـــة 488

لتحريك سطر او اكثر الى الاعلى (الاسفل) , وهذا يتطلب من المبرمج تحديد 10Hتستخدم المهمة رقم 6 او 7 في الاعتراض رقم

احداثيات نقطة بداية ونقطة نهاية السطر (السطور) المراد تحريكها . يقصد بالاحداثيات رقم السطر ورقم العمود لنقطة البداية ونقطة

. DX ,وتخزن احداثيات نقطة النهاية في المسجل CXالنهاية. تخزن احداثيات نقطة البداية في المسجل

. عدد السطور يتراوح بين 1 و 25 . لذا فان تخزين الرقم صفر في المسجل ALاما عدد السطور المراد تحريكها فيخزن في المسجل

يؤدي الى تحريك جميع سطور الشاشة وبالتالي الى مسح الشاشة . AL

ان تحريك سطور الشاشة الى اعلى او الى اسفل يؤدي الى اظهار سطور فارغة ذات مواصفات معينة تحدد بواسطة القيمة المخزنة

. وهذا يعني انه يتوجب على المبرمج تحديد صفات السطور الفارغة التي تظهر في اسفل او اعلى الشاشة عند BHفي المسجل

الرقم 07 يعني سطور بيضاء على خلفية سوداء . وللتعرف على هذه الصفات BH,07 MOVتحريك السطور. فمثلا في الجملة

يمكنك عزيزي الدارس الرجوع الى المراجع . 10Hوعلى بقية المهمات التي يقدمها الاعتراض رقم

الاعتراضات(INT 17H, INT 16H)

للتحكم بجميع عمليات لوحة المفاتيح من خلال المهام 0 او 1 او 2 . تستخدم المهمة رقم صفر لقراءة 16Hيستخدم الاعتراض رقم

رمز واحد بواسطة لوحة لمفاتيح . تستخدم المهمة رقم 1 للحصول على حالة لوحة المفاتيح. اما المهمة رقم 2 فتستخدم للحصول على
الرايات الخاصة بلوحة المفاتيح .

وتجدر الاشارة الى انه عند الضغط على اي مفتاح في لوحة المفاتيح فان شيفرة الاسكي للرمز ورقم المفتاح تخزن في مخزن لوحة

. لذا فان التعليمات : صفحـــة 490 Keyboard bufferالمفاتيح

, ورقم AL بحيث يخزن الاسكي كود للرمز في المسجلAH, ALتؤدي الى نقل رمز واحد من مخزن لوحة المفاتيح الى المسجلات

. وفي حالة عدم وجود رموز في مخزن لوحة المفاتيح فان هذه المهمة تؤدي الى الدخول في دوران لحين AHالمفتاح الى المسجل

ظهور رمز في ذلك المخزن .

تستخدم المهمة رقم 1 لتحديد وجود رمز في مخزن لوحة المفاتيح . ففي حالة وجود رمز في المخزن فان هذه المهمة تؤدي الى نقل

كما هو الحال في المهمة رقم صفر . غير ان عملية النقل هذه تؤدي الى شطب الرمز AL,AHالرمز ورقم المفتاح الى المسجلات

ورقم المفتاح في المخزن على عكس المهمة رقم صفر.

ونتيجة لتنفيذ المهمة رقم 1 فان راية الصفر في مسجل الرايات توضع في حالة "1" اذا كان مخزن لوحة المفاتيح خاليا من الرموز ,

وتوضع راية الصفر في حالة "0" اذا احتوى مخزن لوحة المفاتيح على رموز .

تستخدم سلسلة التعليمات التالية لمسح محتويات مخزن لوحة المفاتيح .

كما موضح في صفحـــة 491

تستخدم التعليمات (3-1) لفحص وجود رمز في مخزن لوحة المفاتيح . وفي حالة وجود رموز يتم الانتقال الى التعليمة رقم 5 والتعليمات التي تليها والتي تؤدي الى شطب رمز واحد من مخزن لوحة المفاتيح والانتقال بعد ذلك الى التعليمة رقم 1 . تستمر هذه العملية لحين مسح جميع الرموز من مخزن المفاتيح .

. تعتمد قيم هذه الرايات على وضع ALتستخدم المهمة رقم 2 للحصول على الرايات الخاصة بلوحة المفاتيح ونقلها الى المسجل

. NUM LOCK , SCROLL LOCK , INSERT SHIFT , CTL , CAPSالمفاتيح :

لتنفيذ المهمات التالية : 17Hيستخدم الاعتراض رقم

الى الالة الطابعة . AL_ المهمة رقم 0: ارسال الرمز المخزن في المسجل

_ المهمة رقم 1: تجهيز الالة الطابعة لاستقبال الرموز .

_ المهمة رقم 2: الحصول على حالة الالة الطابعة .

تحدد الالة الطابعة بواسطة بايت خاص حيث تخصص كل ثنائية فيه لاداء وظيفة معينة. فمثلا : تستخدم الثنائية رقم 5 لتحديد وجود ورق على الالة الطابعة , وتسنخدم الثنائية رقم 7 لتحديد جاهزية الالة الطابعة .

كما هو موضح صفحــة 492

. وهنا يمكن لهذه المهمة التعامل مع ثلاثة الات طابعة ارقامها DX_ تستخدم التعليمة الاولى لتخزين رقم الالة الطابعة في المسجل هي 2 , 1 , 0 .

الى الرمز BX.وبهذا يشير المسجل BX_ تستخدم التعليمة الثانية لتخزين مقدار ازاحة الرمز الول في سلسلة الرموز في المسجل التالي .

. CX_ تستخدم التعليمة الثالثة لتخزين عدد رموز السلسلة المراد طباعتها في المسجل

. AL_ تستخدم التعليمة الرابعه لنقل الرمز التالي في الطباعة الى المسجل

_ تستخدم التعليمة الخامسة لطباعة الرمز التالي .

_ تستخدم التعليمات السادسة والسابعة لتنظيم دوران يكفل طباعة جميع رموز السلسلة .

وتجدر الاشارة هنا الى انه بعد تنفيذ طباعة اي رمز فان حالة الالة الطابعة وحالة عملية الطباعة تخزن في المسجل

AH . لذا بعد التعليمة INT 17H يمكن كتابة تعليمات لفحص حالة الالة الطابعة وحالة الطباعة للتتاكد من نجاحها .تدريب 4 صفحة492

*تطبيقات عملية صفحة 494

**الادخال/ الاخراج المبرمج -Programmed

اسلوب برمجة عمليات الادخال الاخراج الذي يتطلب اشراف مباشر وكامل من قبل المعالج على جميع مراحل تنفيذ العمليات .

**الادخال/ الاخراج الموجهة بالاعتراضات -Interrput – driven اسلوب برمجة عمليات الادخال الاخراج الذي تتولى فيه وحدة الادخال/ الاخراج المسؤولية الكاملة في تنفيذ جميع مراحل العمليات بعد اصدار اشارة اعتراض إلى المعالج.

**اسلوب الاستفتاء Polled- interrupt technique اسلوب تحديد الوحدة المعترضة بطريقة فحص ثنائية الحالة لجميع الوحدات حسب نظام اولويات محدد مسبقا .

Vectored-interrupt technique ** أسلوب المتجهات

اسلوب تحديد الوحدة المعترضة من خلال كود الاعتراض الذي تقدمه الوحدة المعترضة بعد تأكيد وصول اشارة الاعتراض من قبل المعالج.

**برنامج استاتيكي Static program البرنامج الذي يحتوي معطياته بداخله ويلزم تعديله في كل مرة يستخدم فيها لمعالجة معطيات جديدة

**برنامج ديناميكي Dynamic program البرنامج الذي تفصل فيه المعطيات عن الكود، بحيث يتم ادخال المعطيات اثناء مرحلة التنفيذ.

**حيز عناوين الذاكرة الرئيسية Memory address space الحيز الذي يتكون من جميع مواقع الذاكرة الرئيسية وتصل سعته في حواسيب 8088/8086 لغاية M1 ميغابايت .

**منافذ الادخال الاخراج IO address space الحيز الذي يتكون من جميع منافذ وحدات الادخال/ الاخراج وتصل سعته في حواسيب 8088/8086 لغاية K64 بايت .

**شاشة رمزية Character screen الشاشة التي تستطيع عرض الرموز فقط ولا تستطيع عرض الرسومات أو الصور

أو المخططات

**شاشة نقطية Bit -mapped screens الشاشة التي يتم فيها تشكيل الرموز من بكسيلات ويمكنها عرض مختلف اشكال البيانات من صور ومخططات ورسومات .

**مسجل حالة وحدة الادخال/ الاخراج IO status register المسجل المستخدم لتخزين حالة وحدة الادخال/ الاخراج بحيث تخصص كل ثنائية للتعبير عن صفة معينة مثل جاهزية الوحدة .

**منافذ الادخال/ الاخراج IO ports المسجلات التي تخزن البيانات والاشارات اللازمة لعملية الادخال/ الاخراج مثل : منفذ البيانات، ومنفذ الحالة ، ومنفذ التحكم.

الوحدة الثامنة : الماكرو واستخدامها

قد يتطلب الأمر في بعض الأحيان تكرار مجموعة من التعليمات ل لقيام بوظيفة محد دة داخل برنامج لغة أسمبلي. وهذا بدوره يؤدي إلى زيادة عد د التعليمات في البرنامج ما ينتج عنه زيادة إحتمال الوقوع في الأخطاء إضافة إلى زيادة في الجهد ال لازم لإدخال هذه التعليمات إلى البرنامج قبل تنفيذه. لذلك توفر لغة أسمبلي إمكانية تعريف مجموعة التعليمات المتكررة مرة واحدة على شكل ماكرو)برنامج جزئي(يكن إستدعائها عد د من المرات عند الحاجة إليها ومن أي مكان في البرنامج .

بهذا يمكن تعريف الماكرو على أنها مجموعة من جمل لغة أسمبلي يتم تعريفها عادة في بداية البرنامج(من قبل المبرمج مرة واحدة، ويمكن الرجوع إليها عن طريق جملة الإستدعاء من أي مكان في البرنامج الرئيس أو حتى من داخل الماكرو نفسها الستدعاء الذاتي ل لماكرو(أو من داخل ماكرو آخر، بدون الحاجة إلى تكرار مجموعة الجمل أو التعليمات .

ونظرا لأهمية ومكانة الماكرو في لغة أسمبلي فقد سمي الأسميلر الذي يعالج الماكر و باسم الماكرو أسمبلر Macro Assembler وذلك لكونه يعطي المبرمج إمكانية كتابة برنامج لغة أسمبلى على شكل مجموعة من الماكرو والبرامج الفرعية تؤدي بمجملها إلى الآتي:

تقليل عد د التعليمات المدخلة تجزئة البرنامج إلى مجموعة من الماكرو والإجراءات)procedure(.

تقليل الجهد ال لازم لإدخال التعليمات لكونها قد قلت .

تقليل إمكانية الوقوع في الخطأ الناتج عن عملية إدخال التعليمات بسب ب إدخالهاعلى شكل ماكرو .

ومن أهم خصائص البرامج المكتوبة على شكل مجموعة من الماكرو والإجراءات أنها سهلة التتبع والفهم والتعديل . فإذا أعطت الماكرو نتائج صحيحة فإن النتائج تبقى صحيحة وبدون أخطاء مهما بلغت عد د مرات إستدعائها.

وتجدر الإشارة إلى أنه عند إستدعاء الماكرو بإستخدام جملة الإستدعاء والتى ستم شرحها فيما بعد، فإن الأسمبلر يقوم بإستبدال هذه الجملة بمجموعة الجمل المكونة لمتن الماكرو. ويطلق على هذه العملية عملية إنتشار الماكرو Macro()Expansion

الأمثلة على إستخدام الماكرو : عمليات الإدخال والإخراج لتخزين قيم إبتدائية في المسجلات تمهيدا لتنفيذ عمليات الإعتراض (Inter rupts)، تحويل البيانات بين النظامين الأسكي والثنائي، وفي العمليات الحسابية عند إجرائها على كلمات عدة بدلا من بايت أوكلمة، وفي الروتينات الخاصة بمعالجة سلاسل الرموز وكذلك في إنجاز عملية القسمة عن طريق الطرح

ويجب التنويه هنا إلى أن تعريف الماكرو يسبق عادة تعريف أي من قطاعات برنامج لغة أسميلي.

أسئلة التقويم الذاتي صفحه 524

مقارنة بين الماكرو والبرنامج الفرعي)الإجراء Procelure(

مع أن الفكرة العامة ل لماكرو والإجراء هي واحدة أي كتابة مجموية التعليمات

المتكرة في البرنامج مرة واحدة ومن ثم إستدعائها عند الحاجة ، إلا أنه يوجد اختلاف بياها في أموركثيرة نوردها بما يلي :

- بالنسبة لطريقة التعريف يبدا تعريف الماكرو بتوجيهة Macro بينما يبدا تعريف الإجراء بتوجيهة PROC. كذلك فإن تعريف الماكرو قد يتطلب إستخدام معاملات)Parameters(. وفي مثل هذه الحالة يجب الإعلان عن هذه المعاملات ضمن جملة بداية الماكرو. أما عند تعريف الإجراء فمن غير الجائز إستخدام المعاملات مع توجيهة PROC ,

بالنسبة ل لإجراء فإن شيفرة الهدف Code( )Target المكونة لجسم الإجراء تظهر مرة واحدة في البرنامج وأن هناك جملة في شيفرة الهدف مقابل كل جملة إستدعاء لإجراء في لغة المصدر، وعلى خلاف ذلك فإن الأسمبلر يستبدل كل جملة إستدعاء الماكرو بالتعليمات

المكونة لجسم الماكرو. لذلك فإن شيفرة الهدف المكونة لجسم الماكرو تتكر ر في البرنامج الهدفي بعد د مرات الإستدعاء وذلك نتيجة معالجة جملة إستدعاء الماكرو من قبل المعالج. كذلك لا توجد جملة في شيفرة الهدف مقابل كل جملة إستدعاء ل لماكرو في لغة المصدر كما هو الحال في الإجراء. نستنتج ما تقدم عن عملية إنتشار الماكرو هي من خصائص الماكرو ولا يوجد ما يقابلها في الإجراء ت تم معالجة جملة إستدعاء الماكرو من قبل الأسمبلر وذلك بانتشار الجمل المكونة لجسم الماكرو مكان كل جملة إستدعاء، لذلك تعتبر جملة إستدعاء الماكرو من توجيهات الأسمبلر وليس من تعليمات المعالج. بينما تعتبر جملة إستدعاء الإجراء من التعليمات ولذلك فتتم معالجتها من قبل المعالج.

تستخدم تعليمة العودة من البرنامج الفرعي RET والتي ترد في نهايته ولا يوجد ما يكافئها في حالة الماكرو .

وقبل الإنتهاء من المقارنة بين الماكرو والإجراء لا بد من ذكر بعض الميزات التي يمتاز بها الماكرو على الإجراء :

يعتبر استخدام الماكرو أكثر كفاءة عندما يكون جسم الماكرو مكونا من عد د قليل من التعليمات. لذلك يحتاج برنامج اسمبلي الذي يستخدم الماكرو بدلا من الإجراء الى وقت أقل ل لتنفيذ لكون عملية إستدعاء الإجراء والعودة منه)أثناء التنفيذ(تحتاج إلى وقت ملحوظ مقارنة مع وقت تنفيذ متن الماكرو. ففي حال تكرار إستدعاء الإجراء فإن ذلك يؤدي إلى إستهلاك جزء لا باس به من وقت المعالج . يمكن تخزين الماكرو في ملف مكتبة الماكرو Library( )Macro حيث يمكن إستخدام هذه المكتبة أو جزء منها في برنامج أسمبلي مما يساعد المبر مج على بناء برنامج أكثر تعقيدا. يعتبر الماكرو أكثر مرونة من الإجراء حيث يمكن تعديل طريقة عمله)ليس فقط تغي ير النتائج عن طريق تغيير المعاملات كما هو الحال في الإجراء(عن طريق تعديل الجمل المكونة لتنه بإستخدام المعاملات. وهذا غير م مكن الحدوث في الإجراء، حيث أن أي تعديل على الجمل الملكونة لمتن الإجراء يتطلب ترجمته من جديد.

**. مكونات الماكرو

كما أسلفنا يتم تعريف الماكرو قبل تعريف قطاعات البرنامج المختلفة حيث يتكون تعريف الماكرو من جملة البداية وجملة النهاية والجمل الأخرى المشكلة لمتن الماكرو . وهذه الجمل إما تكون على شكل توجيهات للأسمبلر أو على شكل تعليمات كما تم بيانها في الوحدة السادسة من هذا المقرر. وسنبين في الأقسام التالية أجزاء الماكرو الثلاث بالتفصيل .

1.4 جملة بداية الماكرو Macro Definition Statement

تعتبر هذه الحملة من توجيهات الماكرو حيث تستعمل لتحديد بدايته ، وتشكل مجموعة جمل أسمبلي الواقعة بين جملة بداية الماكرو وجملة نهايته متن (جسم) الماكرو (Macro Body

والصيغة العامة لتوجيهة بداية الماكرو

Name MACRO | Parameter 1, Paramater]...

حيث أن : Name- هو إسم فريد مميز البرنامج للدلالة على الماكرو حيث يستخدم هذا الإسم عند إستدعاء الماكرو عند الحاجة إليه في البرنامج ويجب أن يكون الإسم خاضعة لقواعد تكوين الأسماء .

MACRO - مثل توجيهة بداية الماكرو .

(Formal تمثل المعاملات [Parameter I, Parameter)...]

الشكلية الوسطيه - (Paratneter List . بين جملة الاستدعاء والماكرو ، وهذه المعاملات إختيارية يحددها المبرمج

حسب الحاجة إليها وستستخدم من الآن فصاعدا الأقواس المربعة في حالة التوجيهات لتعني أن ما بداخلها هو إختياري. وبالنسبة للمعاملات فلا يوجد أي تحديد لعددها. والقيد الوحيد في بعض أنماط الأسمبلر أنها يجب أن تقع جميعا في سطر واحد مع اسم الماكرو . ويفضل عند إختيار إسم الماكرو و أن يكون معبرا عن العمل الذي يؤديه ولا يجوز تشابه الإسم مع الأسماء الأخرى المستخدمة في البرنامج، وعند مصادفة جملة بداية الماكرو من قبل الأسمبلر يتم إدخال إسم الماكرو إلى جدول خاص بأسماء المنكر و المستخدمة في البرنامج يدعى جدول أسماء الماكرو Macro Name Table) حيث يستخدم اسم الماكرو في الوصول إلى متن الماكرو أثناء عملية

انتشار و الماكرو جملة إستدعائه عند مصادفتها من قبل الأسمبلر

527

**جملة نهاية الماكرو ENDM

وتعتبر هذه الجملة من التوجيهات الخاصة بالأسمبلر حيث تعني أن الجمل الواردة بينها وبين جملة بداية الماكرو MACRO تمثل متن الماكرو والتي سيتم معالجتها من قبل الأسمبلر عند إستدعاء الماكروء

والصيغة العامة لهذه الجملة هي : ENDM

وكما يدل إسم هذه الجملة فهي الأخيرة في جمل الماكرو و تحدد نهايته .

3.4 متن (جسم) الماكرو Macro Body

يتكون جسم الماكرو من مجموعة من الحمل والتي تتكون من تعليمات وتوجيهات .

ويمكن لمتن الماكرو أن يتضمن أي من التعليمات والتوجيهات التي مر ذكرها في الوحدتين الخامسة والسادسة في هذا المقرر. إضافة إلى ذلك يوجد عدة توجيهات خاصة بالماكرو بالإضافة إلى توجيهة بداية الماكرو MACRO وتوجيهة نهاية الماكرو ENDM. ويمكن تقسيم

التوجيهات الخاصة بالماكرو إلى خمسة مجموعات رئيسة على النحو التالي:

أ) التوجيهات عامة الغرض General Purpose Pseudo - OPS

إضافة إلى توجيهة بداية الماكرو MACRO ونهايته ENDM توجد توجيهة ثالثة تحت هذا النوع هي التوجيهة LOCAL

تستخدم هذه التوجيهة داخل جسم الماكرو لتعريف الرموز المستخدمة داخله. ومثال ذلك عند إحتواء الماكرو على جمل موسومة (Labeled Statements). ففي مثل هذه الحالة يجب إخبار الأسمبلر عن طريق توجيهة LOCAL على أنه في كل مرة يتم فيها إستدعاء

الماكرو ونشره مكان جميلة الإستدعاء يجب أن يستعمل إسما جديدة لهذا الوسم (العلامة) وإلا فإن هذه الأوسمة ستتكرر وتشابه في أكثر من موقع داخل برنامج أسمبلي مما يؤدي إلى حدوث خطأ إعادة إستخدام نفس الرمز أو الوسم (العلامة) في أكثر من موقع». وهذا

مخالف لقواعد لغة التجميع ، والصيغة العامة لهذه التوجيهة هي:

... LOCAL symbol [, symbol]

حيث أن : symbol - يمثل إسم الرمز الموضعي المؤقت والذي سيتم استبداله بإسم رمز فريد في كل مرة يتم فيها نشر الماكرو مكان إحدى جمل الإستدعاء .

يجب أن يستخدم على الأقل رمز (Symbol) واحد مع كل توجيهة LOCAL أما في حال إستخدام أكثر من إسم رمز واحد كما تم التعبير عنه في الصيغة«. [symbol ,"] . .، فيجب فصل هذه الأسماء عن بعضها البعض بإستخدام الفارزة)comma(. يعتبرإستخدام

أسماء الرموز مع التوجيهة LOCAL بمثابة إعلان عن أن هذه الرموز هي بمثابة رموز موضعية للماكرو حيث يمكن إستخدامها من قبل أي جملة أخرى داخل متن الماكرو. لذلك عند إستخدامها في الماكرو تأتي التوجيهة LOCAL بعد جملة تعريف الماكرو مباشرة .

وأن إستخدام أي جملة قبلها حتى ولو كانت جملة ملاحظات يؤدي ذلك إلى توليد خطأ قبل الأسمبلر .

الرجوع الي مثال ١ صفحه 529 و 530

ب) توجيهات التكرار

تشمل هذه التوجيهات ثلاثة أنواع رئيسة هي :

1- توجيهة REPT

تستخدم هذه التوجيهة لاعادة تكرار مجموعة من الجمل عدة مرات تحدد بقية المعامل الحقيقي عند استدعاء الماكرو. والمقصود بالمعامل الحقيقي هو المعامل المستخدم مع جملة استدعاء الماكرو بينما نعني بالمعامل الشكلي بالمعامل المستخدم عند تعريف الماكرو والمستخدم ضمن متن الماكرو

والجمل التي يتكرر تنفيذها تقع بين التوجيهةREPT وتوجيهة نهاية التكرار وهي مشابهه لتوجيهة نهاية الماكرو في الصيغة (ENDM) والصيغة العامة لهذه التوجيهة هي

REPT expression

Statements

ENDM

حيث ان expression - يمثل ان المعامل الشكلي ويجب ان تؤول قيمته الى ثابت عددي (مكون من 16 ثنائية ويعامل على انه بدون اشارة). ويحدد هذا العدد عدد مرات تكرار تنفيذ مجموعة الجمل (Statements) ويمكن لهذه الجمل ان تشمل أي من جمل لغة الاسمبلي التي يفهمها الاسمبلر وتجدر الإشارة الى انه يمكن استخدام توجيهات التكرار منفصلة او داخل ماكرو

كما في مثال 2 ص 531

وتدريب 1 واسئلة التقويم الذاتي 4 ص 532

2- توجيهة IRP

تستخدم هذه التوجيهه عند الحاجة لتكرار جملة او اكثر من جمل لغة الاسمبلي حيث ان عدد التكرارات والمعاملات لكل واحدة من التكرارات تحددها مجموعة من القيم تسمى Arguments وترد جميعا ضمن جملة بداية التكرار . الصيغة العامة لهذه التوجيهة هي

<....IRP Parameter,<arguments[,arguments]

Statements

ENDM

كما هو مبين في هذه الصيغة فان توجيهة IRP تأخذ عاملين :الأول Parameter يمثل المعامل الشكلي والثاني يمثل مجموعة القيم التي سيتم تعويضها واحدة تلو الأخرى لكل مرة يتم فيها تكرار مجموعة الجمل Statmentsالمكونة لمتن التوجيهة IRP

ومبدا عمل هذه التوجيهة هو ان الجمل الواقعة بين IRP و ENDM تكرر مرة لكل قيمة من القيم ويعتبر المعامل الأول بمثابة اسم سيتم استبداله بالقيمة الحالية ويمكن للقيم ان تكون على شكل نص داخل حاصرات علوية او رمز او ثابت عددي وعند استخدام اكثر من قيمة يجب فصلها عن بعضها البعض باستخدام الفارزة (comma) كذلك فان الاقواس على شكل زاوية (<>) حول هذه القيم تعتبر اجبارية ويجب استخدامها حول القيم . وكذلك بالنسبة للمعامل فيمكن استخدامه أي عدد من الممرات ضمن الجمل المكونة لمتن التوجيهة IRP

وعند مصادفة الاسمبلر لتوجيهة IPR فانه يعمل نسخة من الجمل المكونة لمتن التوجيهة لكل قيمة من القيم الواردة في المعامل الثاني Arguments والواردة داخل قوسين على شكل زاوية واثناء عملية النسخ يتم استبدال القيمة الحالية مكان المعامل أينما ورد في جمل متن التوجيهة IRP

وفي حالة عدم استخدام قيم داخل القوسين(<>) فان المعامل الشكلي يستبدل بالقيمة الخالية أي يستعاض عنه ب لا شيء اما في حال عدم استخدام اقواس على شكل زاوية بداخلها قيم فانه يتم اهمال التوجيهة IRP وكانها لن تكن

كما في مثال 3 وتدريب 2 و 3 ص 533 واسئلة التقويم ص 534

3- توجيهة IRPC

حيث يتم تكرار مجموعة الجمل بعدد الاحرف المكونة للنص String ففي المرة الأولى يستبدل الحرف الأول من النص مكان المعامل الشكلي في الجمل وفي التكرار الثاني يستبدل الحرف الثاني وهكذا حتى التكرار الأخير حيث يحل الحرف الأخير من الحروف مكان المعامل الشكلي في الجمل

ويمكن للمعامل String ان يحتوي على حروف وأرقام ورموز أخرى وفي حال احتواء النص على فراغات او فارزات او أي رموز تستخدم لفصل الاحرف عن بعضها ويجب وضع النص داخل قوسين على شكل زاوية (<>) وكذلك كما هو الحال مع التوجيهات السابقة فانه يمكن استخدام المعامل الشكلي أي عدد من المرات داخل الجمل المكونة لمتن التكرار

وعند مصادفة الاسمبلر لتوجيهة IRPC فانه يعمل نسخة من الجمل المكونة لمتن التكرار لكل رمز من رموز النص بحيث يتم اثناء عملية النسخ استبدال الرمز الحالي مكان المعامل الشكلي في الجمل كما في مثال 4 وتدريب 4 ص 535

ج) التوجيهات الشرطية :

كما هو الحال مع توجيهات التكرار يمكن استخدام هذه التوجيهات منفصلة او ضمن ماكرو ففي هه التوجيهات يقوم الاسمبلر بفحص شرط معين ففي حال تحقق الشرط يقوم الاسمبلر بمعالجة جمل اسمبلي الواقعة بين بداية التوجيهة الشرطية وحتى جملة النهاية وفي حال عدم تحقق االشرط يقفز الاسمبلر عن هذه الجمل (يمكنك الرجوع الى الوحدة الخامسة ب عنوان جمل الترجمة المشروطة)

د) توجيهة EXITM

تستخدم هذه التوجيهة لجعل الاسمبلر ينهي عملية انتشار الماكرو مكان جملة استدعائه بحيث يستثنى من عملية الانتشار هذه جميع الجمل الواقعة بين توجيهة EXITM وتوجيهة نهاية ماكرو ENDM عند معالجة الاسمبلر لجملة الاستدعاء

ويمكن استخدام هذه التوجيهة مع التوجيهات الشرطية الأخرى وذلك للقفز عن بعض جمل الاسمبلي في نهاية متن الماكرو عند حدوث شرط معين

الرجوع الى مثال 5 ص 536 وتدريب 5 ص 537

هـ) توجيهات الطباعة

تستخدم هذه التوجيهات للتحكم بكمية الجمل التي ينتجها الاسمبلر من الماكرو في البرنامج الرئيس نتيجة عملية التجميع والتي تخزن في ملف من النوع LST وفي حالة عدم استخدام أي من هذه التوجيهات نلاحظ انه عند طباعة البرنامج بعد عملية الترجمة فان جمل الماكرو تتكرر في البرنامج الرئيس نتيجة نشر الماكرو مقابل كل تعليمة من تعليمات استدعائها وعن طريق استخدام توجيهات الطباعة هذه يمكن عرض او عدم عرض جمل الماكرو وعند استدعائها في البرنامج الرئيس وتشمل توجيهات الطباعة ثلاثة توجيهات وهي
LALL,SALL,XALL حيث ان النقطة هي جزء من التوجيهة وبدونها يعطي الاسمبلر خطا على هذه التوجيهات عند تشغيل الاسمبلر والقيام بعملية الترجمة

وتجدر الإشارة الى ان استخدام احدى هذه التوجيهات يبقي مفعولها قائما على كل ماكرو مستخدم ما لم تستخدم توجيهة أخرى كذلك وفي حالة عدم استخدام أي من هذه التوجيهات في البرنامج فان الاسمبلر يفترض ضمنا ان التوجيهة XALL هي المستخدمة .

1- توجيهة XALL

تستخدم هذه التوجيهة لعرض التعليمات التنفيذية فقط والتي تولد شيفرة هدفية حيث ان جمل الملاحظات الواردة في الماكرو والتوجيهات الأخرى التي لا تحجز مواقع للبيانات يتم حذفها وعدم عرضها في الملف من النوع (.LST) والناتج عن تنفيذ الاسمبلر MASM

حيث ان النقطة جزء من التوجيهة وعدم وضعها يؤدي الى خطا لغوي

2- توجيهة LALL

تستخدم هذه التوجيهة لعرض كافة جمل ماكرو (بما فيها جمل الملاحظة والتعليق) عند انتشار مكان جملة الاستدعاء ويجب في هذه الحالة ان تسبق هذه التوجيهة توجيهة استدعاء الماكرو

3- توجيهة SALL

عند استخدام هذه التوجيهة في برنامج لغة الاسمبلي فانه يؤدي ذلك الى عدم نشر الجمل المكونة لمتن الماكرو عند استدعائها من داخل البرنامج الرئيس ومعنى ذلك ان الملف من نوع (.LST) والناتج من عملية تنفيذ الاسمبلر MASM لا يحتوي على الجمل المكونة لمتن الماكرو

ومثال 6 يوضح كيفية استخدام هذه التوجيهات واثرها على جمل العرض ص 539 و ص 540

** جملة استدعاء الماكرو (Macro call Statment)

قبل استدعاء الماكرو يجب تعريفها أولا. وعند تعريفها يجب مراعاة الآتي:

-يحدد إسم الماكرو بحيث يكون الإسم معبرا عن وظيفتها .

-تحدد جمل اسمبلي المطلوب استخدامها في الماكرو (أي متن الماكرو)

-انهاء الماكرو بتوجيهة النهاية ENDM

أما بالنسبة للكيفية التي تستدعي بها الماكرو فيتم ذلك بذكر إسمها في المكان المطلوب فيه استدعائها متبوعا بسلسلة من المعاملات إن كان الماكرو قد عرف بمعاملات parameters. والبرنامج التالي يوضح كيفية الاعلان عن الماكرو واستدعائه من داخل البرنامج الرئيس . وفي هذا البرنامج تم استخدام برنامجين فرعيين من نوع ماكرو الأول بإسم INIT1 لتهيئة المسجلات اللازمة لعمل هذا البرنامج ، والثاني بإسم DISPM وذلك لعرض نص معين على الشاشة مثال 7 صفحه 542-543 مثال 8صفحه 544-545

**أدوات الماكرو (Macro Operators)

إضافة إلى التوجيهات السابقة والتي يمكن استخدامها مع الماكرو توجد أيضاً مجموعه من الادوات الخاصه والتي تستخدم مع هذه التوجيهات ومع الماكرو نفسه.وبالنسبه إلى عدد المعاملات ونوعها فيعتمد ذلك على نمط الاسمبلر المستخدم.ومن أبرز هذه الأدوات ما يلي:

1)أداة التعويض &

تستخدم هذه الاداة لأخبار الاسمبلر بأن يستبدل معاملا شكلياً بالمعامل الحقيقي المناظر له عند استدعاء الماكرو.

والصيغة العامة لهذه الأداة هي &parametar

تستخدم هذه الاداة في حالة كون أحد المعاملات يسبق أو يتبع نص معين ، أو عندما يظهر المعامل في نص داخل حاصرات علوية ،والمثال التالي يوضح استخدام هذه الاداة: مثال (9) صفحة 547

2- أداة النص الحرفي (Literal Text Operation) < >:

تستخدم أداة النص الحرفي وهي على شكل قوسين < > لإخبار الأسمبلر بأن يعامل مجموعة العناصر الواردة بينها على أنها نص متكامل وليس مجموعة قيم منفصلة.

الصيغة العامة < text >

حيث يمثل text على انه عنصر حرفي متكامل يؤخذ باحرفه لا بمعناه حتى لو احتوى على فواصل وفراغات واي رموز اخرى تحمل منفصلة معنى معين يمكن استخدام هذه الاداة لاخبار المعالج بان عمل بعض الرموز الخاصة مثل & والفارزة المنقوطة ; على انها حروف كما هي لاتحمل أي معنى . ويمكن استخدام هذه الاداة مع توجيهات تكرار مثل توجيهة ipr كما هو مبين في مثال 10 ص 548

3- اداة الرمز الحرفي ! (Literal – Character Operator)

تستخدم هذه الأداة لإخبار الأسمبلر بأن يعامل الحرف التالي مباشرة إلى جوار هذه الأداة على أنه رمزا من رموز الآسكي وليس رمزا من رموز لغة أسمبلي يحمل معنا معينا

الصيغة العامة :!Character

حيث أن Character يمثل رمزا من رموز شيفرة اسكي مثال 11 ص550

4- أداة التعبير (Expression) %

تستخدم هذه الأداة لإخبار الأسمبلر بأن يعامل ما بعد هذه الأداة على أنه تعبيرا Expression وليس نصا text

الصيغة العامة % text

حيث يقوم المعالج عند مصادفة هذه الأداة بحساب قيمة text ومن ثم يستبدل المعامل text بالقيمة المحسوبة. ويمكن للمعامل text أن يكون إما تعبيرا عدديا أو نصيا معرفا باستخدام التوجيهة EQU مثال 12 ص551

-5أداة الملاحظة داخل الماكرو ;;

تستخدم هذه الأداة لإخبار الأسمبلر بأن لا يقوم بنسخ جمل الملاحظات والتعليقات الموجودة في الماكرو عند نشر الماكرو مقابل كل جملة من جمل الاستدعاء.

الصيغة العامة ;; text

حيث أن text يمثل تعليقا أو ملاحظات تظهر فقط داخل الماكرو

6- مكتبة الماكرو Macro Library

الهدف من مكتبة الماكرو هو حفظ جميع البرامج الفرعية المكونة بأسلوب الماكرو بحيث يتم تعريفها مرة واحدة واستخدامها في أي برنامج يحتاج إليها المبرمج فبدلا من تعريف برامج الماكرو في بداية كل برنامج رئيس يمكن تخزين جميع برامج الماكرو في مكتبة الماكرو حيث يمكن الرجوع إليها باستخدام توجيهة INCLUDE

في حال كانت مكتبة الماكرو مخزنة على وحدة القرص الصلب C: باسم MACRO.LIB فانه يمكن استخدام جميع الماكرو في هذه المكتبة عن طريق توجيهة INCLUDE C: MACRO.LIB

وعند استخدام مكتبة الماكرو بهذا الأسلوب فإن طباعة الملف من النوع LST الناتج عن معالجة الأسمبلر للبرنامج الرئيس يؤدي أيضا إلى طباعة جمل الماكرو وللتخلص من هذه الطباعة ولتوفير الوقت والورق اللازم لذلك تستخدم توجيهة INCLUDE

على النحو التالي:

IF 1

INCLUDE C:MACRO.LIB

ENDIF

قد يلزم الأمر في بعض الأحيان فك ارتباط بعض برامج الماكرو الموجودة في مكتبة الماكرو بالبرنامج الرئيس وذلك لعدم الحاجة إليها. ويمكن لهذه الغاية استخدام توجيهة PURGE

على سبيل المثال أن مكتبة الماكرو تحتوي على برامج الماكرو MACRO1, MACRO2 ,MACRO3, MACRO4

وأن البرنامج الرئيس يحتاج فقط إلى الماكرو MACRO3 .لذلك يمكن حذف برامج الماكرو المتبقية باستخدام التالي في البرنامج الرئيس

IF 1

INCLUDE C: MACRO.LIB

ENDIF

PURGE MACRO1, MACRO2, MACRO4

مع ملاحظة أن برامج الماكرو التي حذفت من البرنامج الرئيس لم تحذف من مكتبة الماكرو حيث يمكن استخدامها من قبل برامج رئيسة أخرى.

7 - ربط برنامج لغة أسمبلي مع برنامج لغات أخرى عالية المستوى :

نظرا لسهولة برمجة بعض العمليات في لغة أسمبلي وصعوبتها في اللغات عالية المستوى وسهولة برمجة بعض العمليات الأخرى في اللغات عالية المستوى وصعوبتها في لغة أسمبلي، رأى مصممو لغات البرمجة الاستفادة من ميزات كل لغة من اللغات. لذلك نجد أنه بالإمكان ربط برامج لغة أسمبلي مع برامج لغات عالية المستوى مثل لغة باسكال ولغة بيسك ولغة C

مثال 13 ص 555

لربط برنامج الاسمبلي مع برنامج أي بي ام مايكروسوفت باسكال فيتم على النحو الاتي

1- يترجم برنامج اسمبلي للحصول على برنامج هدفي من النوع obj

2- يترجم برنامج باسكال للحصول على برنامج هدفي من النوع objايضا

3-يستخدم برنامج الربط linkللربط بين هذين البرنامجيين الناتجيين عن عملية الترجمة المنفصلة لكل منهما للحصول غلى برنامج واحد قابل للتنفيذ من النوع exe

يجب استخدام توجيهة exten في نهاية جملة تعريف اسمبلي داخل برنامج باسكال

واستخدام توجيهة public مع اسم اججراء اسمبلي داخل برنامج لغة اسمبلي من داخل برنامج باسكال

يجب مراعاة الامور التالية عند ربط برامج لغة التجميع مع برامج لغة c

1- تمييز بعض انماط لغة cبين الاحرف الصغيرة والاحرف الكبيرة اسمبلي من نفس نوع الاحرف

2- تتطلب بعض انماط لغة c انه في حالة تاثير برنامج لغة اسمبلي على المسجلين si ,di يجب ان يخزن هذين المسجلين على المكدس عن طريق استخدام التعليمة push عند الدخول الى برنام اسمبلي وذلك باستخدام تعليمة pop في نهاية برنامج اسمبلي

3- اذا كان مطلوب من برنام لغة اسمبلي اعادة القيم فانه يجب اعادة كلمة واحدة في المسجل ax او كلمتين في المسجليين ax ,dx

4- تتطلب بعض انماط لغة c انه في حالة ان برنامج اسمبلي قد غير راية تحديد الاتجاه الى الحالة 1 يجب تغيير حالة راية الاتجاه الى الحالة صفر عن طريق التعليمة cld وذلك قبل العودة من برام لغة التجيمع

5- عند تمرير المعاملات من قبل برنامج لغة c على المكدس فان معظم انماط لغة c تمرر المعاملات بشكل مخالف لمعظم اللغات الاخرى

**أدوات الماكرو Macro Operators: هي عبارة عن مجموعة من العمليات الخاصة بالماكرو وتقوم بوظائف محددة خاصة بالماكرو

**جملة إستدعاء الماكرو Macro Call جملة أسمبلي بواسطتها يمكن إستدعاء الماكرو من داخل برنامج أسمبلي الرئيس

أو من داخل ماكرو آخر أو من داخل الماكرو نفسه (في حالة الإستدعاء الذاتي)

**ماكرو Macro مجموعة من الجمل يتم تعريفها بشكل محدد مرة واحدة ويمكن إستدعائها أي عدد من المرات . ويهدف الماكرو إلى عدم تكرار تعريف هذه الجمل في البرنامج

**متن الماكرو Macro Body يعني مجموعة الجمل المكونة لجسم الماكرو والواقعة بين جملة تعريف بداية الماكرو وجملة نهايته .

**معامل حقيقي Actual Parameter متغير يرد مع جملة إستدعاء الماكرو وتعرف قيمه قبل استدعاء .

**معامل شكلي Formal Parameter :يقصد به المتغير الوارد في جملة تعريف بداية الماكرو .

**مكتبة الماكرو Macro Library هي مكتبة تجمع جميع الماكرو التي يحتاجها المبرمج في برامج لغة أسمبلي وتهدف إلى عدم تكرار هذه الماكرو عند الحاجة إليها في برامج مختلفة

نشر الماكرو Macro Expansion: يقصد بها استبدال جملة إستدعاء الماكرو بالجمل المكونة لمتنه .