البرمجة التفرعية
Parallel Programming

References

- Peter S. Pacheco, An Introduction to Parallel Programming, Morgan Kaufmann Publishers is an imprint of Elsevier 2011

أساسيات البرمجة التفرعية:

- مقدمة:
  - نظراً لتزايد الحاجة إلى القدرات الحسابية يسعى مصمّمو الحواسيب باستمرار إلى رفع أداء بُنى حواسيبهم.
  - تحتاج بعض التطبيقات إلى سرعة حسابية فائقة مثل المحاكاة الرقمية للمسائل العلمية والهندسية، حيث تتطلب غالباً إجراء حساباتٍ تكرارية على حجم هائل من المعطيات قبل الحصول على النتائج المرجوَّة، والتي ومن المفترض أن تنتهي خلال زمن «معقول» نسبياً.
  - بعض المجالات تبدي تحدياتٍ كبيرة في مجال الحسابات مثل نمذجة التراكيب الكبيرة للأحماض النووية والتنبؤ بالأحوال الجوية.
    - وتُعدّ مسألةٌ ما ذات تحدٍ كبيرٍ بالنسبة إلى الحسابات إذا كان إنجازها غير ممكن خلال زمن معقول باستخدام الحواسيب المتاحة.
  - تُجري الحواسيب التقليدية العمليات المطلوبة بالبرنامج على معالج وحيد،
    - لزيادة القدرة الحسابية يتم استخدام حواسيب تتضمن عدة معالجات؛ أو عدة حواسيب تعمل بالتوازي لحل مسألة معرّفة،
      - وترتبط المعالجات في الحالة الأولى أو الحواسيب في الحالة الثانية بعضها ببعض بطريقة محدّدة لتشكل المنصَّة الحسابية أو الحاسوب التفرعي الذي يسمى «نظام متعدد المعالجات».
      - في كلا الحالتين تُقسم المسألة المستهدفة إلى أجزاء يجري تنفيذها تفرعياً، بحيث يُنفَّذ كلٌّ منها على معالج منفرد.
  - تُسمّى كتابة البرامج لنظام متعدد المعالجات بـ «البرمجة التفرعية» parallel programmingأو «الحوسبة التفرعية».
- إمكانات زيادة المقدَّرات الحسابية

- إن ما يهم بالدرجة الأولى عند تطوير خوارزمية معينة لتنفيذها على نظام متعدد المعالجات؛ هو مدى تسريع عملية الحساب الذي يمكن أن يقدّمه هذا النظام مقارنةً بتنفيذ الخوارزمية على نظام وحيد المعالج.
- يمكن حساب معامل التسريع S(p) وفق العلاقة التالية حيث p عدد المعالجات:

$$S(P)=\frac{\text{زمن تنفيذ الخوارزمية على معالج وحيد}}{\text{زمن تنفيذ الخوارزمية على نظام متعدد المعالجات باستخدام P معالج}}$$

- إن معامل التسريع الأعظمي النظري وفق العلاقة السابقة هو p بفرض أن زمن تنفيذ الخوارزمية قابل للتقسيم والتوزيع على نحو متساوٍ بين المعالجات،
- تفريع الخوارزمية لا يتطلب أعباء زمنيةً إضافية.
- لا يمكن الوصول عملياً إلى هذا الحدّ النظري للتسريع؛ إذ تُسهم عوامل متعددة في إضافة زمن في النسخة التفرعية للخوارزمية غير موجود في نسختها التسلسلية يَحدُّ من مقدار التسريع

- العوامل التي تحد من معامل التسريع:
  - الفترات الزمنية التي لا يمكن لكل المعالجات أن تنفّذ فيها جزءاً من البرنامج،
    - بهذه الحالة تكون هذه المعالجات بوضعية خمول idle وتوقّف عن العمل.
  - توفّر حسابات إضافية في النسخة التفرعية للبرنامج غير ظاهرة في نسختها التسلسلية
    - مثل إعادة حساب بعض المعاملات محلياً.
  - الزمن اللازم لتبادل المعطيات بين المعالجات.
  - إضافةً إلى ذلك فمن المنطقي ألاَّ تقبل بعض مراحل الحسابات التجزئة إلى مهام تفرعية، إذ ينبغي تنفيذها تتابعياً.
  - مثال:
    - إذا كان المطلوب هو تفريع خوارزمية ما على p معالجاً، وكان الجزء من الحسابات الَّذي لا يمكن تجزئته إلى مهام تفرعية هو f، وبإهمال أي زمن إضافي ناتج من تفريع البرنامج؛ فإن معامل تسريع البرنامج يُعطى بالعلاقة التالية والتي تُعرف بقانون «أمدال» :Amdahl's law

$$S(p)=\frac{p}{1+(p-1)f}$$

    - وبالتالي فإن معامل التسريع الأعظمي هو: $S(p)_{p\to\infty}=\frac{1}{f}$
      - أي بافتراض وجود 5 % من الحسابات ستُجرى تتابعياً في برنامج ما؛ فإن مقدار التسريع الأعظمي هو 20 مرة بغض النظر عن زيادة عدد المعالجات.

- أنواع الأنظمة المتعددة المعالجات
  - يمكن أن تصنّف الأنظمة المتعددة المعالجات من وجهة نظر توزيع الذاكرة في نوعين:
    - نظام متعدد المعالجات ذي ذاكرة مشتركة shared memory؛ وهي الأكثر استخداماً في الأنظمة الحديثة،

- نظام متعدد المعالجات ذي ذاكرة موزّعة distributed memory .
- طهر حديثا صنف جديد يستخدم مجموعة من الحواسيب الشخصية أو محطات العمل لتشكيل منصَّة عمل تسمى العنقود .cluster

- البرمجيات الداعمة للأنظمة المتعددة المعالجات
  - جرى تصميم لغات البرمجة التفرعية والبيئات الموافقة تبعاً للنموذج العتادي للنظام التفرعي؛ وتحديداً وفق هيكلية الذاكرة:
    - ذاكرة موزّعة
    - ذاكرة مشتركة
    - نظام عنقودي.
- الأنظمة المتعددة المعالجات ذات الذاكرة المشتركة
  - تدعم بيئة تعدد المعالجات المفتوحة Open Multi-Processing (OpenMP) البرمجة بلغة C أو C++ للأنظمة التفرعية ذات الذاكرة المشتركة بهيكليات ونظم تشغيل مختلفة.
  - تقدّم هذه البيئة للمبرمجين واجهات عمل لتطوير التطبيقات التفرعية؛
    - إن كانت على الحواسيب الشخصية المتضمِّنة معالجات متعددة النوى أو الحواسيب المخصَّصة العالية الأداء.
  - تعتمد OpenMP على تعدد النياسب multithreading، وهي طريقة تفريع parallelizing من خلالها يقوم نيسب thread رئيسي (مجموعة من التعليمات التي تُنفذ تسلسلياً) بالتشعب إلى عدد محدد من النياسب التابعة slaves
    - يجري تقسيم المهمة فيما بينها، بحيث تُنفذ تفرعياً كل منها على نواة أو معالج.
    - يجري تشكيل النياسب للجزء من البرنامج المعَد للتنفيذ التفرعي قبل التنفيذ.
    - يعطى كل نيسب رقماً تعريفياً خاصاً به، في حين يأخذ النيسب الرئيسي الرقم التعريفي صفراً.
    - بعد التنفيذ التفرعي لهذا الجزء من البرنامج، تعود النياسب التابعة لتنضم إلى النيسب الرئيسي الذي يُكمل تنفيذ البرنامج.
  - تتكون الوحدات الأساسية لـ OpenMP من العناصر التالية:
    - توليد النياسب
    - توزيع الأعباء
    - التزامن بين النياسب
    - إدارة معطيات البرنامج ومتحولاته.
  - بما أن OpenMP خاصة بالأنظمة ذات الذاكرة المشتركة، فإن متحولات البرنامج العامة تكون متاحة لجميع النياسب، إضافة إلى المتحولات الداخلية الخاصة بكل نيسب.

- توفر مكتبات مثل Pthreads وShmem العديد من البرامج الفرعية التي تتيح إمكان توليد النياسب وإدارتها، وهذا يؤدي إلى سهولة نسبية بالبرمجة.

- الأنظمة المتعددة المعالجات ذات الذاكرة الموزّعة
  - توفّر الواجهة البينيّة لتمرير الرسائل Message Passing Interface (MPI) مجموعة من البرامج الفرعية ضمن مكتبة معيارية تُتيح كتابة برامج فعالة بلغات برمجة - مثل C, C++, Fortran- لتبادل الرسائل بين عقد النظام التفرعي وتزامن المهام الموزّعة بينها الذي يعتمد على بنية الذاكرة الموزّعة،
    - مثل: الحواسيب التفرعية، والعناقيد، والشبكات غير المتجانسة.
  - اعتماداً على برامج المكتبة السابقة يمكن بناء برامج للتراسل في تلك الأنظمة التفرعية.
  - لا تُعدّ MPI لغة برمجة، وإنما يجري استدعاء برامجها الفرعية من قبل برامج النظام.
  - جرى تصميم MPI لتوفر طوبولوجيا افتراضية virtual، والتزامن، وآليات التراسل بين مجموعة من الإجرائيات على نحو مستقل عن لغة البرمجة،
    - يجري مقابلة كل إجرائية بعقدة معالجة (حاسب أو مخدم)،
    - للحصول على أداء مرتفع يجري عادة نسب إجرائية واحدة إلى كل عقدة معالجة.
  - تدعم جميع منصات عمل أنظمة الحسابات العالية الأداء الواجهة MPI مع قابلية للتوسع scalability.
  - تحتوي الهيكلية الأساسية لبرنامج يستخدم MPI على رابط communicator، وهو متحول variable يُعرِّف الإجرائيات المسموح بها بالتخاطب فيما بينها،
  - بالإضافة إلى ذلك تحتوي الهيكلية أيضاً على تهيئة لبيئة MPI قبل أن يجري استدعاء أي تابع من توابعها، وإغلاقها عند الانتهاء من تلك البيئة.
  - يشمل تراسل المعطيات في بيئة الـ MPI نوعين:
    - تراسل نقطة إلى نقطة، وفيه يتشارك معالجان فقط بتبادل المعطيات،
    - تراسل المعطيات الجمعي، وفيه تتشارك كل المعالجات المعرّفة داخل رابط بتبادل المعطيات.

- الأنظمة العنقودية
  - توفر الآلة الافتراضية التفرعية Parallel Virtual Machine (PVM) بيئة للمبرمجين لتطوير تطبيقات على أنظمة متعددة المعالجات - تحديداً الأنظمة العنقودية - والتحكم في المهام الموزّعة على العقد والتزامن بينها.
    - بالإضافة إلى ذلك تتيح تلك البيئة الأدوات البرمجية اللازمة لتبادل الرسائل بين عقد النظام التفرعي.
  - تُستخدم هذه البيئة على نطاقٍ واسع؛ إذ تضمن التشغيل البيني interoperability بين عقد النظام حتى غير المتجانسة وبوجود نظم تشغيل مختلفة.

- نظم التشغيل التفرعية

- تعتمد أنظمة التشغيل الحديثة على منحيين لتلبية الاهتمامات المتزايدة للتطبيقات التفرعية.
  - الأول اتخذته نظم التشغيل الحديثة من UNIX وWindows بدءاً من الإصدار NT والتي تهتم بالأنظمة المتعددة المعالجات المتناظرة symmetric multiprocessing،
  - الثاني اتخذته نوى نظم تشغيل مخصصة .custom kernels
- تهتم نظم التشغيل التفرعية بصورة أساسية بالجدولة، والتزامن، وتعدد النياسب، وإدارة الذاكرة وتحمل الخلل.
- يسهل تصميم نظام التشغيل في الأنظمة المتعددة المعالجات المتناظرة؛ إذ يشابه مثيله على معالج وحيد،
  - يقوم كل معالج بتنفيذ نسخة من نظام التشغيل؛ وهذا يتيح تنفيذ عدة إجرائيات في الوقت ذاته من دون تدني الأداء،
- في حين يصعب تصميم نظام تشغيل للأنظمة المتعددة المعالجات غير المتناظرة.

- موازنة الأعباء وتحمل الخلل
  - يهدف توزيع الأعباء (الإجرائيات) بصورة متوازنة بين المعالجات إلى الحصول على أعلى أداء ممكن من النظام التفرعي،
  - يوجد نوعان:
    - التوزيع السكوني؛ إذ تُوزّع إجرائيات البرنامج قبل البدء بتنفيذها ويُشار إليها على أنها مسألة جدولة ساكنة static scheduling،
    - والتوزيع الديناميكي dynamic حيث تُوزّع الإجرائيات في أثناء تنفيذ البرنامج.
      - تصنيف التوزيع الديناميكي:
        - مركزي- وهو المستخدم على نطاق واسع- حيث يجري توزيع الإجرائيات بواسطة برنامج مركزي يُنفَّذ على أحد المعالجات ويقوم بدور السيد master، في حين تقوم الإجرائيات الموزعة على المعالجات بدور التابع slave
        - غير المركزي: تعمل مجموعة من المعالجات على المسألة المعنية عن طريق إمرار الإجرائيات بينها.
      - تلجأ الأنظمة العالية الأداء إلى التوزيع الديناميكي المركزي المتعدد الطبقات؛ إذ تأخذ هيكلية المعالجات البنية الشجرية أو العنقودية، وفيها توزّع كل طبقة الإجرائيات على الطبقة الأدنى وفق علاقة السيد-التابع.
  - تُتيح البنى التفرعية وتوزيع المهام تحمل الخلل:
    - ففي حال اكتُشِف خلل بأحد المعالجات -عن طريق المراقبة المستمرة لأدائها- يجري إعادة توزيع المهام الموكلة إليه على معالجات أخرى عاملة،

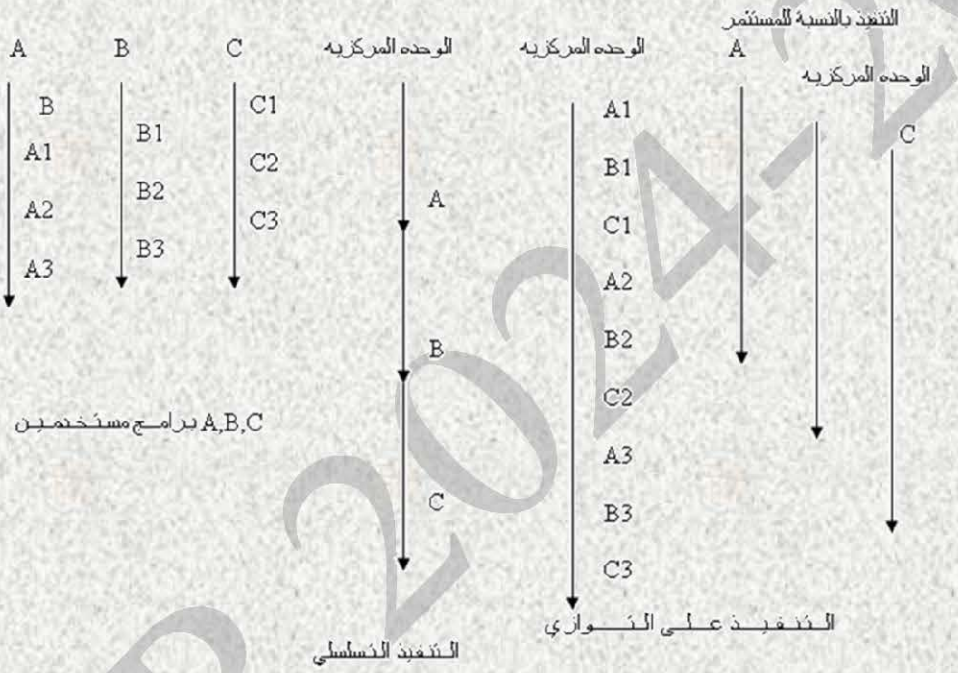
- لتجنب انخفاض الأدائية في النظام كله بهذه الحالات تُصمم الأنظمة بحيث تحتوي عادة على معالجات إضافية احتياطية تعمل بحالة تأهب hot-standby والتي تبدأ بتنفيذ المهام الموكلة فور إسنادها إليها،
    - ويسبق ذلك إعادة دمج المعالج وفق هيكلية النظام.

- تطبيقات وآفاق
    - برزت البرمجة والحوسبة التفرعية لتحقيق المسائل التي تحتاج إلى مُقدَّرات حسابية هائلة.
    - برزت الحاجة في مختلف مناحي الحياة إلى الإمكانات التي تتيحها الحوسبة التفرعية.
    - على سبيل المثال:
        - مجال الإنترنت
            - مع ازدياد الحاجة إلى الخدمات المتنوعة التي تقدّمها- مثل محرّكات البحث والتنقيب في المعطيات والخدمات التجارية المعتمدة على الوب ونحوها- يجري تصميم مخدّمات الوِبْ وتنفيذها اعتماداً على الأنظمة العنقودية؛ بهدف الاستجابة لطلبات المستخدمين خلال زمن معقول.
        - في مجالات الصحة
            - أسهمت الحوسبة التفرعية في رفع فاعلية الأجهزة الطبية مثل التشخيص عن طريق الصور الطبية الثلاثية الأبعاد، وتصميم التراكيب الدوائية. وفي قطاع الاقتصاد تتيح البرمجة التفرعية النمذجة المالية والاقتصادية بفعالية.
        - بالإضافة إلى ذلك فتحت البرمجة التفرعية آفاقاً جديدة في الصناعات الترفيهية بوساطة الرسومات البيانية المتقدّمة، والحقيقة الافتراضية وتقانات الوسائط المتعددة.
        - النمذجة والمحاكاة: تنبؤات الأرصاد الجوية، علم المحيطات، الفيزياء الفلكية.
        - الهندسة: ميكانيك الموائع، الهندسة النووية، الهندسة الكيميائية الروبوتية، الذكاء الاصطناعي، معالجة الصور، وغير ذلك.
        - البحث عن مصادر الطاقة: الكشف عن البترول والمعادن، الكشف الجيولوجي، البحث الطبي والعسكري

- أشكال معالجة المعطيات على التوازي
    - المعالجة المتوازية هي شكل من أشكال معالجة المعطيات تسمح بتنفيذ عدد من الأحداث المتزامنة بنفس الوقت . هذه الأحداث المتوازية يمكن أن تكون على مستويات مختلفة :
        - مستوى البرامج (Programs)
            - يتم تنفيذ عدد من البرامج المستقلة عن بعضها بنفس الوقت و تستخدم مبادئ تعددية البرمجيات (Multiprograms) و المشاركة الزمنية (Temporal Partcipation) وتعددية المعالجة (Multiteatment) من أجل تحقيق ذلك.

- يستثمر هذا المستوى على الحاسبات الكبيرة و تكون المعالجة على التوازي شفافة بالنسبة للمستخدم
- يتولى نظام التشغيل مهمة إدارتها و تطويرها .
- يمكن بالاعتماد على مبدأ تعددية البرمجيات ، تنفيذ عدة برامج لمستخدم واحد أو عدة برامج لمستخدمين مختلفين .
  - تعتمد إحدى طرق تعددية البرمجيات على تقسيم زمن الوحدة المركزية إلى مجالات زمنية متساوية بحيث تشغل الوحدة المركزية برامج مختلفة بشكل دوري خلال المجالات الزمنية

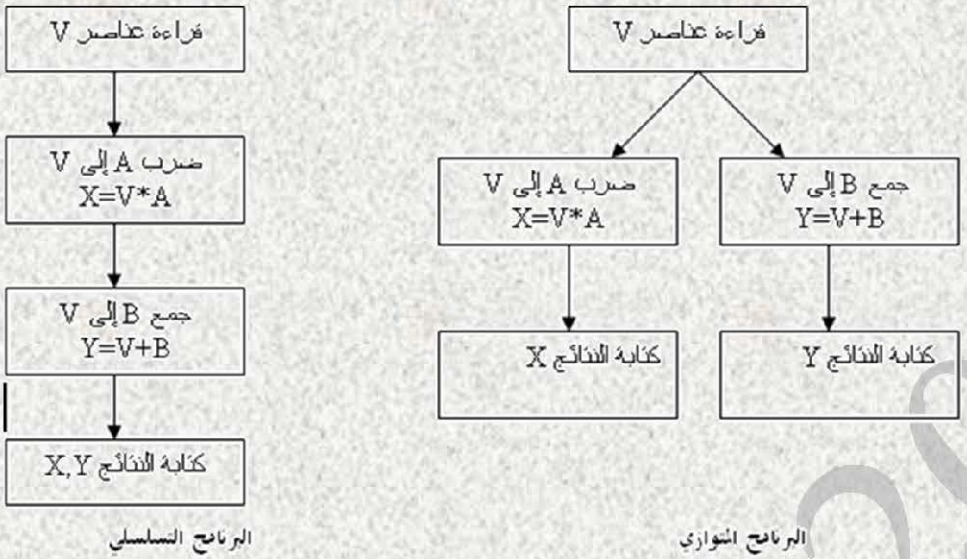
- مستويات المعالجة
  - مستوى البرامج (Programs)
    - مثال:



  - مستوى الإجرائية ((Procedure)
    - يتطلب هذا المستوى تقسيم البرنامج الواحد إلى عدة مهمات على التوازي.
      - إن تقسيم البرامج على هذا الشكل ليس بالعمل السهل فمهمات البرنامج الواحد مرتبطة فيما بينها و غالبًا ما يعتمد تنفيذ مهمة ما على نتائج المهمات الأخرى لذا لابد من البحث عن علاقات التبعية ومن ثم برمجة المهمات غير المرتبطة أو المرتبطة جزئيًا على التوازي.
    - يقوم نظام التشغيل بإدارة ومعالجة هذا المستوى من التوازي إذا تم استخدام أدوات آلية للتوازي أو مترجمات ذكية ولكن هذه الأدوات ما زالت في مجال البحث ولم تثبت فعاليتها بعد،

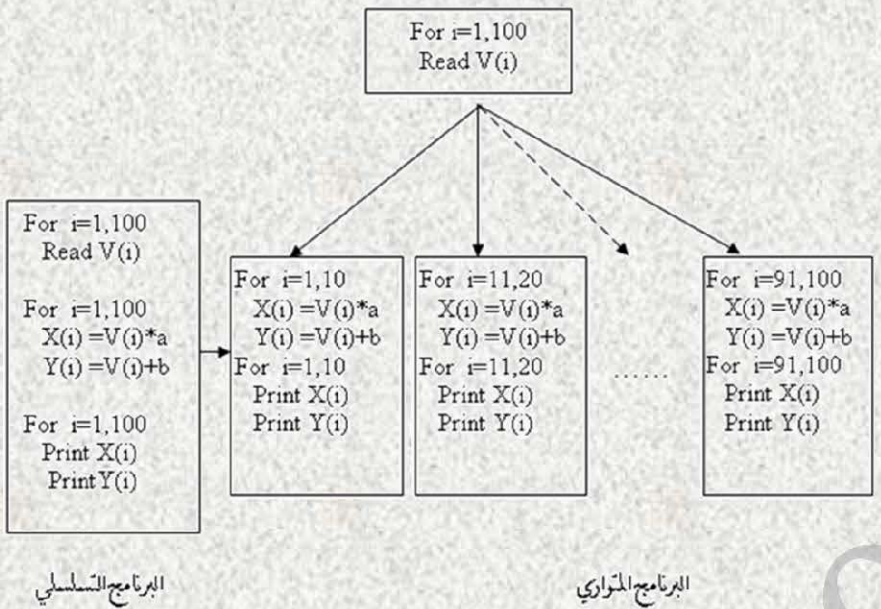
- غالبًا ما يقوم المستخدم بتحليل البرامج على مستوى الخوارزميات و تحويلها إلى برامج متوازية.

- مثال:



- مستوى التعليمات (Instructions)

- يوجد العديد من التقنيات التي تعتمد مبدأ تنفيذ عدة تعليمات مستقلة فيما بينها على التوازي و أشهرها تقنية معالجة الجداول التي تقوم بتنفيذ تعليمة وحيدة تعالج معطيات مختلفة على عدة معالجات بنفس الوقت .
- يعالج هذا النوع من التوازي على مستوى :
  - نظام التشغيل نظرًا لتوفر الأدوات الآلية المساعدة على تحويل البرامج .
  - اللغات البرمجية حيث تتوفر على الحاسبات الشعاعية لغات برمجية خاصة للبرمجة الشعاعية
    - مثال: FORTRAN Victories
- يستخدم هذا النوع من التوازي في برامج الحساب العملي بشكل خاص حيث تعالج الأشعة و المصفوفات
- مثال:



- تعتمد تقنيات العمل الضخي (Pipeline) مبدأ تقسيم التعليمة الواحدة إلى تعليمات جزئية متتالية بحيث يمكن تنفيذ هذه التعليمات الجزئية بنفس الوقت على معطيات مختلفة.
- يتم الاستفادة من هذا النوع من التوازي على مستوى العتاد، وهو شفاف بالنسبة للمستخدم في الحاسبات الحالية.
- تستخدم أغلب المعالجات والبطاقات المتخصصة ذات الأداء العالي في يومنا هذا مبدأ العمل الضخي في تصميمها.
- مثال:
  - يمكن تقسيم عملية ضرب عددين ممثلين بالفاصلة العائمة إلى العمليات الجزئية التالية :
    - مقارنة القوى.
    - وضع الرقمين بنفس القوة.
    - القيام بعملية جمع العددين.
    - كتابة النتيجة بشكل مضبوط.
  - باعتبار هذا التقسيم يمكن إجراء عدة عمليات ضرب على سلسلة من الأعداد بنفس الوقت، وذلك بإجراء المراحل المختلفة على ثنائيات أعداد مختلفة.
  - طريقة إجراء العملية على التوازي
  - مثال:
    - عملية ضرب عددين ممثلين بالفاصلة العائمة


مقارنة القوى
وضع الرقمين بنفس القوة
جمع العددين
ضبط النتيجة
المرحلة الأولى
المرحلة الثانية
المرحلة الثالث
المرحلة الرابعة
المرحلة الخامسة
A.B
C.D
E.F
G.H
I.J
A.B
C.D
A.B
C.D
E.F
A.B
C.D
E.F
G.H
A.B
C.D
E.F
G.H
I.J