تحسين الأداء للمعالجات متعددة النوى

د. م. جمال الياسين[1] ود. م. وسيم السمارة[2]

الملخص

اسْتُخْدِمَتِ المعالجات متعددة النوى مدة طويلة في أنظمة المخدمات، ثم انتقلت فيما بعد فأصبحت مستخدمة في أنظمة الحواسيب الشخصية. تتجلى المشكلة المشتركة بين العملاء في الشركات الصغيرة أو المتوسطة أنهم يتوقعون حلولاً فورية ومجربة من حيث تعزيز الأداء وكفاءة العمل عند شرائهم للتجهيزات العتادية الجديدة.

يعدُّ التفاعل بين أداء التطبيقات والمعالجات متعددة النوى معقداً، ممّا يؤدي إلى خفض كفاءة الأداء على العديد من المستويات سواء في التطبيقات أو أنظمة التشغيل.

من الجدير بالذكر أن المعالجات متعددة النوى لا تحسّن أداء بعض التطبيقات، بل على النقيض تؤثر سلباً فيها.

أدركت مراكز البحوث هذه المشكلة واتبعت العديد من الأساليب والطرائق لمعالجتها.

الكلمات المفتاحية: المعالجات متعددة النوى، تعدد المهام، التطبيقات التفرعية، الحوسبة عالية الأداء.

[1]أستاذ مساعد– قسم هندسة الحواسيب والاتمتة– كلية الهندسة الميكانيكية والكهربائية– جامعة دمشق.

[2]مدرس– قسم هندسة الحواسيب والأتمتة– كلية الهندسة الميكانيكية والكهربائية– جامعة دمشق.

Improve Performance of Multi-Core Processors

Dr. Eng. Jamal Al Yasin[(1)] and Dr. Eng. Wasim Al Samara[(2)]

Abstract

Multi-core processors, long used on high-end servers, have now moved into the PC server and desktop system space. The problem is that corporate buyers at small to medium sized companies will expect, when they -buy new hardware, solution that boost performance abd efficiency immediately.

However, the interaction between application performance and multi-core processors is far more complex, affecting numerous layers of the application and operating systems.

For some business applications, multi-core processors do not boost performance; to the contrary, such processors actually slow down certain classes of applications.

Research centers have become aware of this problem and have examined ways to address it.

Keywords: Multi-Core Processors, Multithreading, Parallel Applications, High-Performance Computing.

[(1)] Assistant Professor –Department of Computer& Automation Engineering - Faculty of Mechanical & Electrical Engineering Damascus University.

[(2)] Lecturer –Department of Computer& Automation Engineering - Faculty of Mechanical & Electrical Engineering Damascus University

المقدمة:

استخدّمت المعالجات متعددة النوى مدة طويلة في أنظمة المخدمات، ثم انتقلت فيما بعد فأصبحت مستخدمة في أنظمة الحواسيب الشخصية. تتجلى المشكلة المشتركة بين العملاء في الشركات الصغيرة أو المتوسطة أنهم يتوقعون حلولاً فورية ومجربة من حيث تعزيز الأداء وكفاءة العمل عند شرائهم للتجهيزات العتادية الجديدة.

يعدّ التفاعل بين أداء التطبيقات والمعالجات متعددة النوى معقداً، ممّا يؤدي إلى خفض كفاءة الأداء على العديد من المستويات سواء في التطبيقات أو أنظمة التشغيل.

من الجدير بالذكر أن المعالجات متعددة النوى لا تحسّن أداء بعض التطبيقات بل على العكس تؤثر سلباً فيها.

أدركت مراكز الأبحاث هذه المشكلة وقامت واتبعت العديد من الأساليب والطرائق لمعالجتها.

تشرح هذه المقالة كيف أن للمعالجات متعددة النوى تأثيراً سلبياً في أداء بعض التطبيقات، وتقوم بعرض الخيارات المتاحة لدى مطوري هذه التطبيقات كي يستطيعوا التعامل معها.

تحتوي هذه المقالة على الموضوعات الآتية:

- ما المعالجات متعددة النوى؟
- لماذا توجد تطبيقات تعمل بسرعة أقل على المعالجات متعددة النوى؟
- ما خصائص التطبيقات التي تعمل أبطأ على المعالجات متعددة النوى؟
- ماذا عن التطبيقات التي تعمل بسرعة أكبر في المعالجات متعددة النوى؟
- ما تعريف التطبيقات متعددة المهام والتطبيقات التفرعية؟
- لماذا التطبيقات متعددة المهام غالباً لا تعمل عملاً أسرع على المعالجات متعددة النوى؟
- ما الآثار المترتبة على استخدام المعالجات متعددة النوى بالنسبة الى المطورين؟
- ما الخيارات المتاحة للمطورين للمساعدة في تحسين الأداء لمعالجات متعددة النوى؟

– تطور العتاد المادي:

في البداية قام المبرمجون بكتابة برامجهم معتمدين على تعليمات المستوى العتادي باستخدام مستوى لغة التجميع، فقد كانت بنية المعالج المادية واضحة في أذهان أولئك المبرمجين.

ثم تطور الأمر حين صنعت أنظمة التشغيل وارتكزت على اللغات عالية المستوى، وتمكن المطورون من كتابة برامجهم اعتماداً على لغات برمجة مثل (BASIC – COBOL– C-PASCAL K) وأخرى غيرها من اللغات التي مهمتها تحويل شفرة البرنامج إلى تعليمات مفهومة من قبل الآلة.

كان مبرمج أنظمة التشغيل بحاجة لجعل أنظمتهم تفرعية، في حين بقي مبرمجو التطبيقات على حالهم، ولم يطوروا برامجهم لتعمل في البيئة التفرعية.[5][1]

بات الإقبال على الحوسبة سريعاً جداً ولا ينتهي، مما دفع المطوّرين مثل (AMD-INTEL) للسعي من أجل الوصول لأعلى الحدود الممكنة للسرعة (إضافة المزيد من السرعة سوف يكون في جوهر صناعة الرقاقات الحرارية)، وإن إضافة النوى المتعددة الى رقاقة واحدة أصبح هو أسهل طريقة لزيادة القدرة الحاسوبية، هذا ما يدفعنا للسؤال المحوري لدينا: "إذا كانت زيادة السرعة تفيد دوماً في تسريع التطبيقات، فهل زيادة عدد النوى سوف يقدم النتائج ذاتها؟".

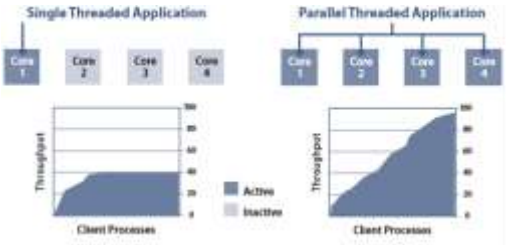
– تعقيد تقنية تعدد النوى وتأثيره في الأداء:

في الحواسيب المحمولة متعددة النوى يجد المستخدمون أنهم قادرون على تشغيل برامج متعددة في اللحظة نفسها

بتأخير قليل عند التنقل بينها، لكن عند استخدام برنامج وحيد لا يكون الأداء على مستوى عالٍ كما يجب.[2]

لذلك حتى يكون الأداء قوياً عند تشغيل برنامج وحيد، يجب على المطورين جعل شفرات البرامج تفرعية لكي تفيد من هذه الميزة المتوافرة في المعالجات متعددة النوى.

في هذه الأيام أصبحت معظم التطبيقات متعددة المهام (Multithreaded)، لكن عادة ماتكون هذه المهام مقتصرة على مناطق منفصلة تماماً، لذلك وكبداية يجب أن يكون هنالك قدر من التفرعية موجوداً داخل التطبيقات الى لتحسين من أدائها حين تستخدم ضمن البيئة التفرعية للافادة من هذه التقنية قدر المستطاع.



الشكل (1) مقارنة بين التطبيقات وحيدة المهمة والمتعددة المهام

الحاجز الثاني هو المهارة المطلوبة، إذ يعدّ جعل البرمجة فعّالة بين المعالجات متعددة النوى أمراً دقيقاً وصعباً على المبرمجين بسبب التحديات التي ستواجههم في البرمجة التفرعية التي هي:

- يتطلب هذا الأمر إعادة توليد شفرات برامجهم، وإعادة بنائها من جديد اعتماداً على منصات برمجية صممت خصّيصى لهذا الغرض.
- تحقيق التبادلية بين مكونات التطبيق أمر صعب، كمثال: وجود قاعدة بيانات تتفاعل معها بقية المكونات الموجودة ضمن التطبيق.

وإذا تم التغلب على هاتين المشكلتين فإن قوة الأداء سوف تتحقق بلا شك وسيفيد الجميع من ميزة المعالجات متعددة النوى.[6]

إحدى التحديات الموجودة عند تحليل إشكالية تقنية تعدد النوى هي أنّ إعادة بناء تطبيق معين أو أي جزء منه يعدّ أمراً صعباً اذ يجب فهم العديد من الأمور، منها أبعاد التطبيق المتعددة، وبنية الرقاقة، ونظام التشغيل:

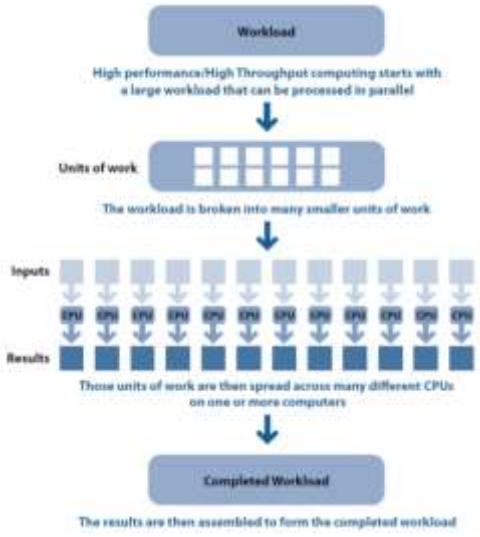
- المعالجات متعددة النوى ليست متطابقة، شركتا (Intel – AMD) تتبعان طرائق مختلفة في صناعة هذه المعالجات، ومن الممكن تشغيل التطبيقات على المعالجات كلّها، وهذا يعني احتمال تأثر أداء هذه التطبيقات إن لم تكن متوافقة مع بنية أحدها.
- تختلف أنظمة التشغيل عن بعضها بعضاً في قدرتها على التعامل مع المهام المتسايرة(concurrent threads).
- من الممكن أن تعمل التطبيقات بشكل أبطأ أو أسرع في تقنية المعالجات متعددة النوى اعتماداً على متطلبات التطبيق.[5][4]

- أين تساعد تقنية تعدد النوى؟؟

يمكن للمعالجات متعددة النوى أن تعزز الأداء وتقويه في حالات محددة، وفي هذه الحالات يعدّ تأثير المعالجات متعددة النوى ذا تأثير إيجابي قوي.

إذا قسّم التطبيق إلى أجزاء تعمل على التوازي وفي النهاية تُجمع نتائج أعمال الأجزاء، فإنّ مثل هذه البرامج تفيد بقوة من تقنية المعالجات متعددة النوى، مثال (تطبيقات معالجة الصوت والصورة، وتطبيقات النمذجة العلمية).[2]

يمثل الشكل الآتي البنية الأساسية للحوسبة عالية الأداء:



الشكل (2) البنية الأساسية للحوسبة عالية الأداء

- أين تضر تقنية تعدد النوى؟؟

من الواضح أنّ هناك أمراً غير مفهوم لدى مطوري البرمجيات، وهو أنّ تقنية المعالجات متعددة النوى من الممكن أن تضر الأداء في مثل الحالات الآتية:

- عندما لا يمكن تجزيء عمليات التطبيق تجزيئاً جيداً، اذ انّه يتطلب توضع كمية كبيرة من البيانات في الذاكرة في الوقت نفسه، مثال (البرامج المالية الديناميكية التي تنتظر إدخالات المستخدم اللحظية في الزمن الحقيقي).
- عندما يكون هناك أكثر من مستخدم للتطبيق، وجميعهم يؤدون العمل ذاته، مثال على ذلك (في المتاجر على شبكة الإنترنت ينفذ المستخدمون إلى قاعدة البيانات الخاصة بالمتجر بشكل متزامن ويتعاملون معها جميعاً في لحظة واحدة، ممّا يسبب مشكلات في الأداء وهذا يؤدي إلى عدم رضا الزبائن).
- استخدام قاعدة بيانات والاعتماد على حالة البيانات لإعطاء النتائج (عندما يكون هناك عمليات بحاجة لانتظار عمليات أخرى، كي تتابع التنفيذ، فهذا يشكل تحدياً من: حيث جعلها تعمل عملاً متوازياً).
- حالة مشاركة الملفات والبيانات.

نبين في الجدول (1) أداء التطبيقات وخصائصها في المعالجات متعددة النوى:

الجدول (1) أداء التطبيقات وخصائصها في المعالجات متعددة النوى

الأداء باستخدام تقنية المعالجات متعددة النوى	خصائص التطبيق
أداء أفضل	عدة تطبيقات تعمل بشكل متزامن
أداء أفضل أو الأداء نفسه	تطبيقات يمكن تقسيمها إلى خطوات معالجة مستقلة (ألعاب، محاكاة، نمذجة)
الأداء ينخفض	تطبيقات تستخدم قاعدة معطيات
الأداء ينخفض	تطبيقات تقوم بمشاركة البيانات وتملك مستخدمين عدّة

- التحديات التي تواجه تقنية تعدد النوى:

ذُكرت فيما سبق المواضع التي تكون فيها هذه التقنية مضرة وغير مفيدة، وتم التعرف اليها بوضوح ،فهذه المعالجات تقوم بتسريع العمل في مواضع معينة وتبطئه في مواضع أخرى، تغيرت حالياً منهجية التطوير وسوف نتعرف الآن الأمور التي تجري للحدّ من هذه الإشكالية.

تقدم تقنية تعدد النوى فرصة للافادة من ميزاتها بالشكل الصحيح، ويمكن للبرمجيات أن تأخذ محاسن تقنية المعالجات المتعددة النوى، وتقوم باختبار الأداء وتحسينه.[1]

في بعض الحالات، إن تحقيق التبادلية بين أجزاء التطبيق (مثل قواعد البيانات) يمكن من حل المشكلة دون الحاجة لتطوير كبير من قبل مبرمجي التطبيق.

ينبغي للمطوّرين أن ينظروا إلى هذا الأمر على أنّه طريق لتقليل المخاطر، ممّا يمكنهم من كسب الوقت ريثما يعالجون المشكلة الأهم، وهي إعادة كتابة شفرات برامجهم.[4]

يمثل الجدول (2) الخيارات المتاحة للمبرمجين للانتقال بالتطبيقات إلى البرمجة التفرعية:

الجدول (2) الخيارات المتاحة للمبرمجين للانتقال بالتطبيقات إلى البرمجة التفرعية

درجة التعقيد	العلاج	الحالة
أبسط حل	تطوير نظام التشغيل إلى نسخة أحدث تدعم تقنية المعالجات متعددة النوى	إذا كان التطبيق يعمل على نسخة من نظام تشغيل لا تدعم تقنية المعالجات متعددة النوى
التحديث بسيط نسبياً	تطوير قاعدة البيانات أو المكونات الأخرى إلى نسخ أحدث تدعم تقنية المعالجات متعددة النوى	إذا كان التطبيق يستخدم قاعدة معطيات أو مكونات أخرى لا تدعم تقنية المعالجات متعددة النوى
يحتاج إلى جهد كبير	إعادة كتابة البرنامج	إذا كان من الممكن جعل التطبيق ينفذ بشكل

– لماذا تبطئ التطبيقات متعددة المهام في النظم متعددة النوى؟؟

ليس من المتوقع كبداية أن تعمل التطبيقات متعددة المهام ببطء في النظم متعددة النوى وإنّ أسباب هذا الأمر معقدة وبحاجة لتوضيح:

في التطبيقات متعددة المهام التي تتشارك البيانات، يستهلك التزامن بين المهام كثيراً من مصادر النظام في الآلات متعددة النوى.

إذا نفذ أكثر من مسرى إلى البيانات ذاتها فإن هذا النفاذ يجب أن يتزامن، وعندما نقوم بجعل عملية النفاذ إلى البيانات متزامنة فإنّ هذا الجزء من البرنامج لن يعود قادراً على التنفيذ من قبل أكثر من مسرى ،وهذا يعني حدوث مشكلة في تتابع العمل.[3]

إن عملية استخدام ذاكرة كاش لا تعالج هذه المشكلة بتاتاً بل تزيدها سوءاً، لأنّه إذا كان لدينا أكثر من معالج، وكل معالج له ذاكرة كاش خاصة، وكانت ذواكر كاش جميعها تشير إلى البيانات نفسها فإنّ أي تعديل من قبل أحد هذه المعالجات على تلك البيانات سيجعل من هذه الأخيرة غير صالحة بالنسبة الى باقي المعالجات، ومن ثمّ لحل هذه الإشكالية يجب أن نضبط التزامن أيضاً بين ذواكر كاش.

إن التكلفة الإجمالية لعملية التزامن مرتفعة، وذلك يفيد بأنّ الأداء في النظم متعددة النوى سيكون أقل فعالية من الأداء في النظم أحادية النواة اذ لسنا بحاجة هنا لعملية تزامن.[6]

– المشكلات الشائعة في التطبيقات متعددة المهام:

الصعوبات الأخرى التي تواجه المبرمجين في هذه المرحلة الانتقالية هي مشكلات قديمة من البرمجة التفرعية القائمة على الأنظمة متعددة النوى، ومنها:

- فشل مشاركة الملفات.
- اختلاف الذاكرة .

– الخيارات المقدمة للمطورين:

لدينا العديد من الأدوات المتوافرة التي قدمت المساعدة في حل المشكلات السابقة ومن هذه الأدوات:

- مكتبة مايكروسوفت التفرعية (Microsoft's parallel library).
- منصة إنتل التفرعية (Intel parallel studio).
- منصة MPI المفتوحة (Open MPI).

- تحسين تحويل فوريه السريع على بنى المعالجات متعددة النوى:

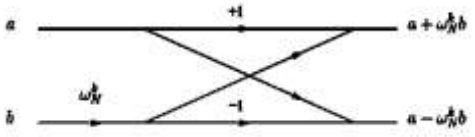
تحويل فوريه السريعFFT: هو خوارزمية سريعة لحساب تحويل فوريه المتقطعDFT. وهناك العديد من الخوارزميات لتحويل فوريه السريع سنركز على الأكثر شهرة بينها، وهي Cooley-Tukey اذ تحسب من العلاقة:

$$X(k) = F_1(k) + w_N^k F_2(k), 0 \le k \le \frac{N}{2} - 1$$

$$X\left(k + \frac{N}{2}\right) = F_1(k) + w_N^k F_2(k), 0 \le k \le \frac{N}{2} - 1$$

اذ w هو عامل العبث الخاص بهذه الخوارزمية، و F1 و F2 هي نقاط N/2 للتحويل فوريه المتقطع DFT.

تـدعى المعادلــة الســابقة بعمليــة الفراشــة Cooley-Tukey، وهي مبّينة بالشكل (3).



الشكل (3) عملية الفراشة Cooley-Tukey

حلّلنا بداية مشكلة تحويل فوريه السريع ذي بعد واحد 1D FFT، وطبقنا عدة تقنيات عليها تحتاج إلى معالجات متعددة النوى للقيام بها سنوردها فيما يأتي:

1- تطبيق الأساس التفرعي:

وهي باعتبار عملية الفراشة وحدة عمل التي تتضمن:

1) قـراءة معلومــات 2-point، وعامــل العبــث مــن الــذاكرة الرئيسة.

2)، ثـم تطبيـق عمليـة الفراشـة علـيهم. 3)، ثـم كتابـة نتيجـة 2-point في الذاكرة الرئيسة.

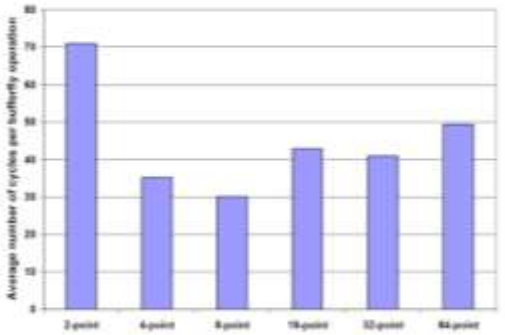
ولاحظنا أنَّ هذه التقنية لم تحقق أي تحسن وإنَّما هي تطبيق لمبدأ الخوارزمية على المعالجات متعددة النوى دون تحسين.

2- وحدة العمل المثالية:

في الطريقة السابقة توجد حواجز تستخدم للتحكم في النفاذ للمعطيات المشتركة، وبهذه التقنية طبّقنا

عمليـات مزامنــة علــى العمليـات بحيـث نقـوم بتقليـل الحواجز في حال عدم الحاجة لها ولا حظنا في هذه التقنية تحسناً بمقدار 40% على 4-point وكلما زدنا نقاط العمل لاحظنا تحسناً أكبر(وصل في بعض الأحيان) إلى 80-100%، وفي حال تمكننا من إيجاد جهاز يملك عدداً غير منته من المسجلات سيكون هناك تحسن لانهائي ولكن لا يمكن إيجاد ذلك، وفي الوقت الراهن تحقق نسبة تحسن 100% في معالجات 64-bit متعددة النوى.

وبين الشكل (4) عدد الدورات من أجل كل عملية فراشة مقابل حجم وحدة العمل مقيسة بـ x-point.



الشكل (4) عدد الدورات من أجل كل عملية فراشة مقابل حجم وحدة العمل مقيسة بـ x-point.

3- تطبيق طريقة خاصة على المرحلة الأولى:

استخدمنا في هذه التقنية في المرحلة الأولى عامل العبث الخاص بالنقطة الأولى نفسه، للنقاط كلَّها، أمَّا في المرحلة الثانية استخدمنا عامل العبث الخاص بالنقطة الثانيــة للنقــاط كلّهــا وهكــذا... ولاحظنــا تحســناً بمقــدار 28.4%.

4- التخلص من عمليات الذاكرة غير الضرورية:

في كل مرحلة يستخدم نصف عدد عامل العبث للمرحلة السابقة حسب عدد النقاط المطبق عليها، ولكن تبقى محجوزة في سجلات الذاكرة، فإذا تخلصنا منها يمكن أن يقوم ذلك بعمليات تسريع بسبب وجود سجلات فارغة أكثر، وحقّق ذلك تحسناً بمقدار 6.2% فقط لأنّه لم يؤثر تعدد النوى فيها إنما مساحة الذاكرة.

5-تقنية تسمية المسجلات وجدولة التعليمات يدوياً:

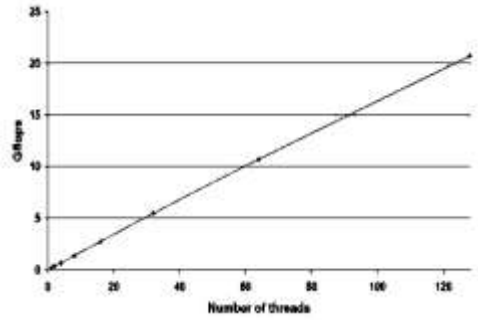
قمنا بعملية مشابهة للتي يقوم بها الحاسب تلقائياً ،ولكن بطريقة يدوية مؤتمتة للعمل على عدة أنوية محسوبة مسبقاً، مّما يقلّل مرات الوصول إلى ذاكرة الكاش، ويزيد cache hit، ويقلل cache miss وتحسن الأداء بمقدار 13.7% ولكنها طريقة غير عملية.

فيما يأتي جدول (3) يبيّن التقنيات المستخدمة، ومدى التحسين في كل منها على تحويل فوريـة السريع باستخدام معالجات متعددة النوى.

الجدول (3) التقنيات المستخدمة ومدى التحسين في كل منها على تحويل فوريه السريع باستخدام معالجات متعددة النوى.

التقنية المستخدمة	نسبة التسريع
الأساس التفرعي	%0
وحدة العمل المثالية	%101.5
طريقة خاصة	%28.4
التخلص من عمليات الذاكرة	%6.2
المسجلات والتعليمات	%13.7

وفيما يأتي الشكل (5) يبيّن تسريع تحويل فوريه السريع أحادي البعد باستخدام عدة مسار حتى الوصول إلى 128 مسرى. ونلاحظ التسارع الكبير في الأداء عند استخدام مسار أكثر.



الشكل (5) يبيّن تسريع تحويل فوريه السريع أحادي البعد باستخدام عدة مسار حتى الوصول إلى 128 مسرى.

REFERENCES المراجع

[1] د.م. عمار زقزوق، " نظم المعالجات التفرعية والبنى المتقدمة للحاسوب "، منشورات جامعة البعث، 2014

[2] Jadhav, S. S [2009]. "Advanced Computer Architecture and Computing, 2nd Edition, Technical Publications Pune, India.

[3] Kal, H., and N. Jotwani [2010]. Advanced Computer Architecture: Parallelism, Scalability, Programmability., 2nd Edition, Tata McGraw-Hill, New Delhi.

[4] Buchanan, W. [2000]. "Computer Busses: Design and Application", Arnold, a member of the Hodder Headline Group, London, 502-503.

[5] Andrews, J. [2009]. Guide to Managing and Maintaining your pc, Edition, Course Technology Cengage Learning, Boston, 243-246.

[6] Ghoshal, S. [2011]. "Computer Architecture and Orginzation: From 8085 To Core2Due and Beyond", Dorling Kindersley, India.

Received	2017/05/29	إيداع البحث
Accepted for Publ.	2017/11/2	قبول البحث للنشر