مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (37) العدد (1) 2015
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (37) No. (1) 2015

بناء محاكي لبنيتي المعالجات فائقة التدرج والمعالجات الشعاعية ومقارنة أدائهما في معالجة التفرع على مستوى البيانات

الدكتور حسن الأحمد*
شيرين حبيب**

(تاريخ الإيداع 16 / 9 / 2014. قُبِل للنشر في 15/ 1 / 2015)

□ ملخّص □

تم في هذه الورقة عرض لبنى المعالجات المتوازية والتركيز على بنيتين أساسيتين من هذه البنى وهي بنية المعالج فائق التدرج (Superscalar Processor) وبنية المعالج الشعاعي (Vector Processor)، وبالإعتماد على الخصائص الأساسية لكل منها تم بناء محاكي لهذه البنى يحاكي آلية عملها برمجياً بهدف المقارنة بين أدائها فيما يخص التوازي على مستوى البيانات (Data Level Parallelism DLP) والتوازي على مستوى التعليمات (Instruction Level Parallelism ILP).

تبين النتائج أن فعالية تنفيذ التعليمات على التوازي تعتمد بشكل كبير وأساسي على اختيار بنية المعالج المناسبة للتنفيذ وفق نوع التوازي الممكن تطبيقه على التعليمات، وأن ميزات الشعاع في البنية الشعاعية تحقق تحسين ملحوظ في الأداء لايمكن إغفاله في تنفيذ عمليات DLP وتبسيط للكود البرمجي وتقليل لعدد التعليمات، ويشكل المحاكي المقدم نواة جيدة يمكن تطويرها والإضافة عليها خاصة فيما يخص المجال التعليمي لطلاب علوم وهندسة الحاسب والمجال البحثي.

الكلمات المفتاحية : التوازي، الهندسة الأنبوبية، المعالج فائق التدرج، المعالج الشعاعي، المحاكي، التوازي على مستوى التعليمة (ILP)، التوازي على مستوى البيانات (DLP).

* مدرس – قسم الحاسبات والتحكم الآلي كلية الهندسة الميكانيكية & الكهربائية_ جامعة تشرين_ اللاذقية_ سورية.
** طالبة دراسات عليا(ماجستير) – قسم الحاسبات_ كلية الهندسة الميكانيكية & الكهربائية_جامعة تشرين_اللاذقية_سورية.

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (37) العدد (1) 2015
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (37) No. (1) 2015

Building a simulator for Superscalar processors and Vector processors and comparing their performance in processing parallelism at data level

Dr. Hassan Alahmad[*]
Shereen Habib[**]

(Received 16 / 9 / 2014. Accepted 15 / 1 / 2015)

□ ABSTRACT □

This paper presents parallel computers architectures especially Superscalar processors and Vector processors, building a simulator depending on the basic characteristics for each architecture, the simulator simulates their mechanism of work programmatically at the aim of comparing the performance of the two architectures in executing Data Level Parallelism (DLP) and Instruction Level Parallelism ILP.

The results shows that the effectiveness of executing instructions in parallel depends significantly on choosing the appropriate architecture for execution, according to the type of parallelism that can be applied to instructions, and the vector features in the vector architecture achieve remarkable improvement in performance that cannot be ignored in execution of DLP, simplify the code and reduce the number of instruction. The provided simulator is a good core that can be developed and modified especially in the field of education for the students of Computer Science and Engineering and the research field.

Key words: Parallelism, Pipelining, Superscalar processor, Vector processor, simulator, Instruction Level Parallelism (ILP), Data Level Parallelism (DLP).

[*]**Assistant Professor, Department of Computer and Control Faculty of Mechanical and Electrical Engineering, Tishreen University, Lattakia, Syria.**
[**]**Postgraduate Student, Depart of Computer and Control Faculty of Mechanical and Electrical Engineering, Tishreen University, Lattakia, Syria.**

مقدمة:

صممت الحواسيب الرقمية بدايةً لإنجاز المهام الحسابية بسرعات كبيرة لايمكن للإنسان بلوغها وبقي الهدف الأساسي منها دوماً تسريع الأداء مما أدى إلى التوجه للمعالجة المتوازية Parallel Processing على اعتبار أن زيادة سرعة المعالج لوحدها لم تعد ناجعة. يقصد بالمعالجة المتوازية معالجة عدة مهام في وقت واحد وبالتالي تقليل زمن إتمام العملية أو إنجاز عدة عمليات في وقت واحد. يعد سايمور كراي Seymor Cray الأب الروحي للمعالجات فائقة الأداء حيث قام بتصميم بنيتين أساسيتين عدتا النواة لبنى المعالجات الحديثة حيث نفذ تصاميم عديدة تضمنت معالجات فائقة التدرج وحواسيب متعددة المعالجات معتبراً أن زيادة سرعة التنفيذ تحتاج لأكثر من زيادة سرعة الساعة، وأخيراً توصل إلى أن الأداء العالي يمكن أن يأتي من معالج وحيد ولكن بتصميم مختلف فصمم المعالج الشعاعي الذي ضم ميزات الشعاع ولكنه لم يلق رواجاً في سوق الحواسيب، توجه بعده العلماء إلى مناح عديدة لتحقيق الأداء العالي منها ماهو برمجي متعلق بالمترجمات وتقنيات الجدولة وتوقع التفرع، ومنها ماهو بنيوي باستخدام معماريات مختلفة منها الهندسة الأنبوبية Pipelining وتعدد النوى Multi-Core والمعالجات فائقة التدرج التي ميزت البنية الرئيسية في الحواسيب الحديثة، وقد استوجب ذلك تطوير في الكيان الصلب وفي الكيان البرمجي مما أدى إلى زيادة في حجم الكيان المادي وتعقيده، ومع التطور الكبير والسريع الذي شهدته كافة المجالات (مجالات التنبؤ بالطقس، معالجة الصورة، الملتيميديا، إلخ ....) ظهرت الحاجة إلى معالجة كميات أكبر من البيانات وبسرعات عالية جداً، مما جعل استغلال ميزات الشعاع والبنية الشعاعية تعود للواجهة. تم التركيز في هذا البحث على إظهار أسباب عودة البنى الشعاعية لتأخذ حيزاً في المعمارية الحاسوبية من خلال مقارنتها مع معمارية الحواسيب فائقة التدرج عبر محاكاة عمل كل منهما وإظهار آلية ونتائج تنفيذها لعمليات نوع DLP.

أهمية البحث وأهدافه:

يهدف البحث إلى دراسة البنى الحديثة للمعالجات المتوازية وتقويم أدائها من خلال بناء بيئة محاكاة (محاكي) لعمل بنيتين أساسيتين من بنى الحاسب الأولى هي المعالجات فائقة التدرج، والبنية الثانية هي المعالجات الشعاعية والتي بقيت في الظل لوقت طويل إلى أن ظهرت الحاجة لها مع تقدم العلوم والحاجة لمعالجة كميات ضخمة من البيانات أو مايعرف بالتوازي على مستوى البيانات (DLP)، وإظهار الفروقات البنيوية وآلية معالجة كلا البنيتين للعمليات نوع (DLP)، وإظهار أهمية استخدام ميزات الشعاع في مثل هذا النمط من العمليات.

طرائق البحث ومواده:

1- مستويات التوازي Parallelism levels:

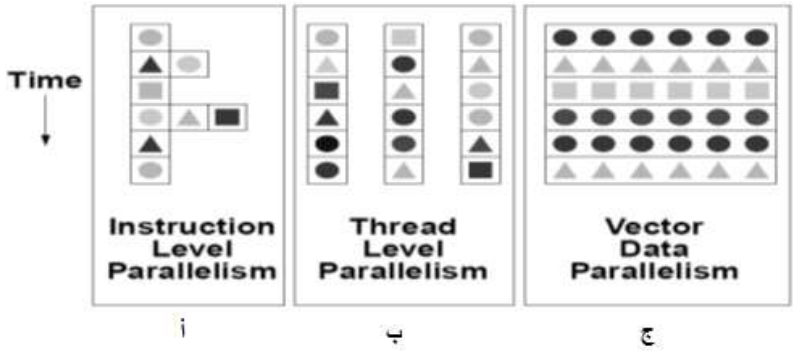
يمكن تمييز ثلاثة مستويات (أنماط) رئيسية للمعالجة على التوازي سيتم ذكرها بصورة مختصرة [1]:

أولاً: التوازي بمستوى التعليمة Instruction Level Parallelism (ILP): تنفذ عدة تعليمات من مسار وحيد بشكل متزامن، مثل المعالجات فائقة التدرج.

ثانياً: التوازي بمستوى المساق Thread Level Parallelism (TLP): تنفذ عدة مساقات من التعليمات بشكل متزامن، مثل تعدد المعالجات multiprocessor machines.

ثالثاً: التوازي بمستوى البيانات Data Level Parallelism (DLP): تنفذ تعليمة وحيدة على مصفوفة من العناصر بشكل متزامن، مثل المعالجات الشعاعية.

يوضح الشكل(1) هذه المستويات حيث يبدو في الشكل(1–أ) إمكانية تنفيذ تعليمة أو عدة تعليمات مختلفة (تدل الأشكال الهندسية المختلفة على وجود عمليات مختلفة) خلال دورة ساعة على مسار من المعطيات في التوازي بمستوى التعليمة، ويظهر الشكل(1–ب) تنفيذ تعليمات مختلفة على مسارات متعددة من التعليمات بمستوى المساق ويظهر الشكل(1–ج) تنفيذ تعليمة وحيدة على مسار من المعطيات خلال كل دورة ساعة بالتوازي بمستوى البيانات.



الشكل (1) مستويات المعالجة على التوازي[1]

نلاحظ أن التوازي بمستوى التعليمة يتميز بإمكانية تنفيذ تعليمات عديدة على عناصر مختلفة في كل دورة ساعة بينما يتم تنفيذ تعليمة وحيدة على مجموعة من العناصر في التوازي بمستوى البيانات مما يستدعي التساؤل عن نجاعة تنفيذ التوازي بمستوى البيانات على طريقة التوازي بمستوى التعليمة أي تكرار تنفيذ التعليمة (حلقة) على العناصر المراد معالجتها وهذا مايهتم البحث به بشكل أساسي.

2– تصنيف الأنظمة الحاسوبية:

تطورت معمارية الأنظمة الحاسوبية مع ظهور المنظومات المتوازية، وبناء على ذلك يمكن استعراض أهم التصنيفات للأنظمة الحاسوبية:

أ– تصنيف Flynn:

صنف فلاين الأنظمة الحاسوبية لأربعة أصناف رئيسية [2]:

أولاً– تعليمة وحيدة ومسار معطيات وحيد Single Instruction single Data Stream (SISD): وهي توافق المعالج uniprocessor والذي يستطيع المبرمج التفكير به كحاسب تسلسلي قياسي ولكن بإمكانه استخدام تقنيات ILP، مثل المعالجات فائقة التدرج وحيد النواة .

ثانياً– تعليمة وحيدة وعدة مسارات معطيات Single Instruction Multiple Data Stream (SIMD): تنفذ تعليمة وحيدة على علبة من البيانات (استخدام DLP)، مثل المعالجات الشعاعية.

ثالثاً– عدة تعليمات ومسار معطيات وحيد Multiple Instruction Single Data Stream (MISD): لايوجد منها نموذج تجاري حتى الآن ولكنها تدخل ضمن التصنيف.

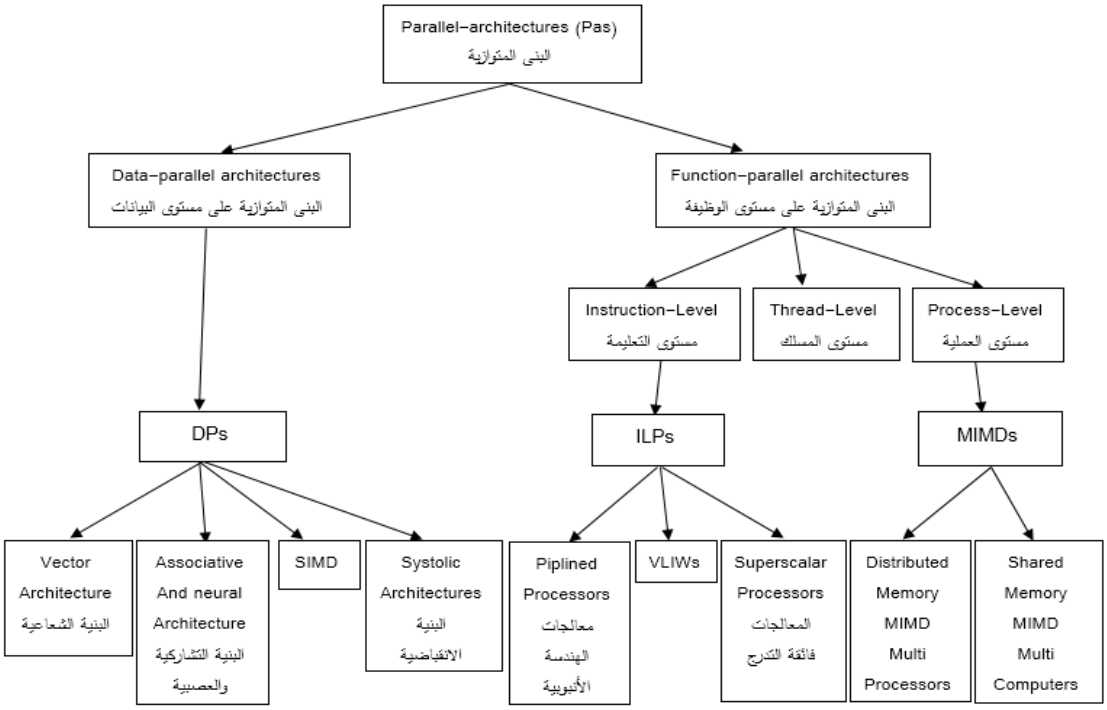
رابعاً– عدة تعليمات وعدة مسارات معطيات Multiple Instruction Multiple Data Stream (MIMD): كل معالج يجلب تعليماته الخاصة ويعمل على المعطيات الخاصة به، مثل المعالجات فائقة التدرج متعددة النوى.

ب- التصنيف الحديث:

تتقسم البنى المتوازية وفق التصنيف الحديث إلى صنفين أساسيين كما يظهر في الشكل(2):

أولاً- البنى ذات التوازي على مستوى البيانات Data Parallel architectures: وتتدرج تحتها أصناف عديدة من البنى، مثل البنى الشعاعية.

ثانياً- البنى ذات التوازي على المستوى الوظيفي Function Parallel architectures : والتي تتقسم بدورها لثلاث بنى متوازية: على مستوى التعليمة (ILP) وتتدرج تحتها المعالجات فائقة التدرج، على مستوى المسلك، وعلى مستوى العملية (MIMD).



الشكل(2) التصنيف الحديث للبنى الحاسوبية

نلاحظ من التصنيفات السابقة أن المعالجات فائقة التدرج تصنف على أنها معالجات مختصة بنوع التوازي ILP وبما أنها بنية المعالجات الرائجة في الحواسيب الحديثة سنظهر في هذا البحث كيف تنفذ هذه البنية التوازي من النوع DLP ولماذا تكون البنية الشعاعية ضرورية وبشدة في هذا النوع من التوازي بحيث لايمكن للمعالجات فائقة التدرج أن تغني عنها.

3- المعالجات فائقة التدرج:

يقوم المعالج فائق التدرج بتعزيز تنفيذ التعليمات العددية، يقصد بالقيمة العددية Scalar قيمة وحيدة صحيحة أو حقيقية يتم تخزينها في متحول في وقت ما، وإحدى طرق تسريع تنفيذ هذه العمليات هي القيام بجلب عدة تعليمات في وقت واحد وتنفيذها على التوازي وهذا مايميز المعالجات فائقة التدرج، تعتمد المعالجة فائقة التدرج على تعدد الأنابيب التي تعمل على التوازي وعلى عدد ونوع الوحدات الوظيفية المتاحة.

تطلق تسمية معمارية مجموعة التعليمات instruction set architecture (ISA) على (مجموعة التعليمات التي توصّف العمليات التي يقوم بها المعالج لإنجاز المهمة الموكلة إليه) والتي يجب على المبرمج أو المترجم أن

يفهمها حتى يتعامل مع المعالج بما يتناسب وبنيته الداخلية حيث تحدد آلية عمل المعالج والعمليات التي يدعمها وآلية التخزين وكيفية الحصول على البيانات من خلال هذه البنية، تم اعتماد MIPS ISA في هذا البحث.

تتألف MIPS ISA من مجموعة من المسجلات عددها 32 مسجل كل منها يخزن كلمة (32 bit) Word والذاكرة عبارة عن مصفوفة وحيدة البعد يحتوي الموقع منها بايت واحد حيث تخزن الكلمة في أربع مواقع متعاقبة في الذاكرة، وتعتمد على مجموعة من التعليمات تأخذ ثلاث صيغ مختلفة [3].

- محددات المعالجات فائقة التدرج:

تعاني الهندسة فائقة التدرج نتيجة خصوصيتها من بعض المحددات منها[4] [5]:

أولاً: تضارب المصدر resource conflicts:

تظهر عند تنافس تعليميتين أو أكثر على نفس المصدر (مسجل، ذاكرة، وحدة تنفيذية) في نفس الوقت، قد يشكل تعدد الوحدات الوظيفية عامل مخفف من حدة هذه التضاربات.

ثانياً: اعتمادية التحكم (الإجرائية) control (procedural) dependency:

تنجم هذه الاعتمادية عن وجود التفرعات وتم التخفيف من هذه الأمر بتقنيات عديدة لتوقع التفرع والحد من تأثير التفرع الخاطئ.

ثالثاً: تضارب (المعطيات) Data conflicts:

تحدث من خلال التنافس على المعطيات بين التعليمات في البرنامج المصدر ونميز منها ثلاث أنماط:

أ- اعتمادية المعطيات Read after Write (RAW): عندما يكون خرج تعليمة هو دخل لتعليمة لاحقة.

ب- اعتمادية الخرج Write after Write (WAW): عندما تكتب تعليمتان في نفس المكان.

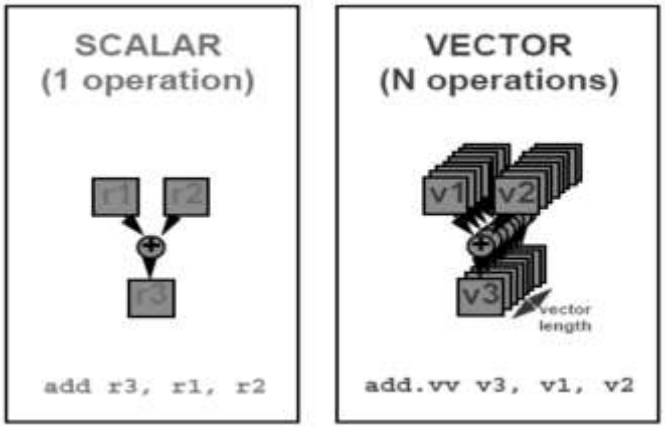
ج- Antidependency Write after Read (WAR): عندما تستخدم تعليمة معاملاً ستكتبه تعليمة لاحقة.

اعتمد البحث التأجيل stall عند ظهور الاعتمادية، والتمرير Forwarding حيث يتم تمرير المعطيات للمراحل اللاحقة حال جهوزها حتى وإن لم تستكمل مراحل الكتابة والتخزين بالإضافة لتوقع التفرع.

4- المعالجات الشعاعية :

تدعم البنية الشعاعية نمط معطيات شعاعي، حيث أن الشعاع عبارة عن مجموعة من الكلمات n بت.

يظهر الشكل(3) أن المعالجات الشعاعية تتعامل مع مسجلات شعاعية تحوي N عدد من الكلمات تطبق عليها تعليمة ما وكأنها عنصر واحد (جمع عناصر شعاع لعناصر شعاع آخر).



الشكل(3) نمط المعطيات في البنية الشعاعية مقارنة بالبنية العددية [1]

إن المعالجة الشعاعية تجلب مجموعة من الخصائص منها:

- تعليمة شعاعية وحيدة تمثل عدد كبير من التعليمات(حلقة).
- عمليات جلب أقل.
- كل نتيجة مستقلة تماما عما قبلها.
- يقلل التفرعات ومشاكلها في الأنابيب.

يجدر بالذكر أن استخدام الذاكرة المخبئية في المعالجات الشعاعية غير مجدي لا بل يأتي بنتائج عكسية لأن عمليات التحميل والتخزين من الذاكرة يتم بكميات كبيرة من البيانات (أشعة) وبالتالي من غير المجدي تحميلها إلى الذاكرة المخبئية أيضاً، ذلك سيعني وقت أطول والحاجة لذاكرة مخبئية كبيرة نسبياً وهذا يتعارض مع مفهوم الذاكرة المخبئية الأساسي [1].

عوض المعالج الشعاعي غياب الذاكرة المخبئية بهيكلية ولوج مختلفة للذاكرة الرئيسية تخفف التأخير الناجم عن النفاذ إلى الذاكرة (memory access penalty) تقوم على تقسيمها إلى علب (blocks) في حين تكون الذاكرة تداخلية (interleaving) في الأنماط العددية للمعالجات مثل المعالج فائق التدرج، ويمكن الولوج لأي من العلب حتى وإن كانت العلب الأخرى قد دخلت حالة التأخير أي كأن كل علبة هو ذاكرة مستقلة ضمن الذاكرة الأساسية وبالتالي ينفذ تأخير الولوج عند الولوج الأول لجلب شعاع من الذاكرة وبعدها يتم جلب عنصر شعاعي في كل دورة ساعة.

تم الاعتماد على مبادئ وصيغ تعليمات MIPS ISA في تصميم ISA بسيطة خاصة بالمعالج الشعاعي مبينة لاحقاً في التنفيذ.

- محددات المعالجة الشعاعية:

أهم نقاط ضعف المعالجات الشعاعية والتي تحد من فعالية أدائها هي:

- تعاني مشاكل الأنابيب التقليدية.
- أداء ضعيف مقارنة بالمعالجات العادية في معالجة القيم العددية.
- تكلفة عالية للذاكرة المخصصة رغم محدودية الولوج في الذاكرة التقليدية.
- تكلفة عالية للتصميم ومردود ضعيف مقارنة بالمعالجات فائقة التدرج.

5- تقييم الأداء Performance Evaluation:

اعتمد البحث في تقييم أداء البنى التي تمت محاكاتها على المعادلات التالية [2]:

أ- زمن المعالج CPU time:

نميز من العوامل المؤثرة في أداء المعالج ثلاثة عناصر أساسية: معدل دورات الساعة Clock Cycle(Rate) أو زمن دورة الساعة (Clock Cycle Time) وهو مقلوب معدل دورات الساعة، متوسط عدد الدورات بالتعليمة Cycle Per Instruction (CPI) وعدد التعليمات Instruction Count (IC). وإذا كنا نعلم عدد التعليمات وعدد دورات الساعة المستهلكة للتنفيذ Cycles يمكن عندها حساب متوسط عدد دورات الساعة اللازمة لتنفيذ تعليمة ما:

$$CPI = Cycles \div IC \qquad (1)$$

عندها يكون زمن المعالج:

$$CPUtime = CPI * Clock\ Cycle\ Time * IC \qquad (2)$$

ب- قانون Amdahal :

يعطى التحسن في الأداء الناجم عن تحسين بعض الأجزاء في حاسب ما بالعلاقة:

(3) زمن التنفيذ لكامل المهمة مع التحسين ÷ زمن التنفيذ لكامل المهمه بدون تحسين = التسارع Speed up

ج- معادلة SAXPY:

تم اختيار تطبيق حلقة معادلة SAXPY على كلا البنيتين لتبيان آلية معالجتهما لها، وتم اختيار معادلة SAXPY كونها الحلقة الداخلية لمعيار Linpack المستخدم في تقييم الأداء وهو عبارة عن مجموعة برامج للجبر الخطي وبرامج لتمثيل الاستبعاد الغاوصي [6].

معادلة SAXPY : Y= a*X + Y

حيث a : تمثل قيمة عددية.

X و Y : كل منها يمثل شعاع من العناصر

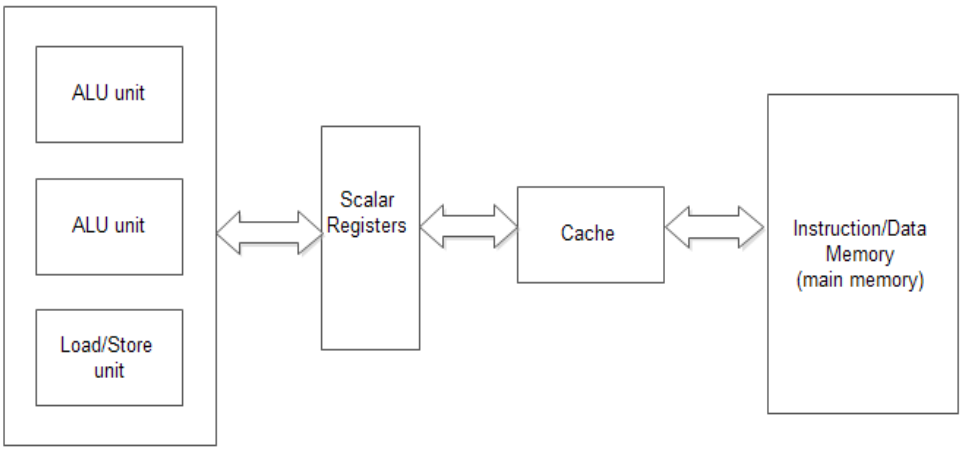
6- بناء المحاكي:

تم بناء المحاكي بواسطة اللغة البرمجية JAVA كونها لغة عملية وتعمل مع معظم أنظمة التشغيل وتحوي العديد مع المكتبات والطرائق المساعدة وذات واجهات أنيقة [7]، وتم اعتماد أبسط بنية ممكنة لكلا النوعين وذلك لتوضيح المزايا المجردة وغير المحسنة لكل طريقة وكيفية تأثيرها في عمليات DLP.

أ- محاكاة بنية المعالج فائق التدرج:

تم محاكاة بنية مبسطة للمعالجات فائقة التدرج تحوي أساسيات هذه البنية والتي تتكون من:

- ذاكرة رئيسية بحجم 40000 بايت.
- ذاكرة كاش بحجم 64 بايت.
- 32 مسجلاً للأغراض العامة كل منها بطول كلمة واحدة.
- وحدة تحميل/تخزين.
- وحدتي معالجة حسابية.
- أنبوبين يعملان على التوازي ويتم جلب تعليمتين ومعالجتهما في كل دورة ساعة. كما يظهر في الشكل(4).



الشكل(4) بنية المعالج فائق التدرج الذي تمت محاكاته

ب– محاكاة بنية المعالج الشعاعي:

تم محاكاة بنية مبسطة للمعالجات الشعاعية تحتوي أساسيات هذه البنية مكونة من:

– ذاكرة رئيسية.

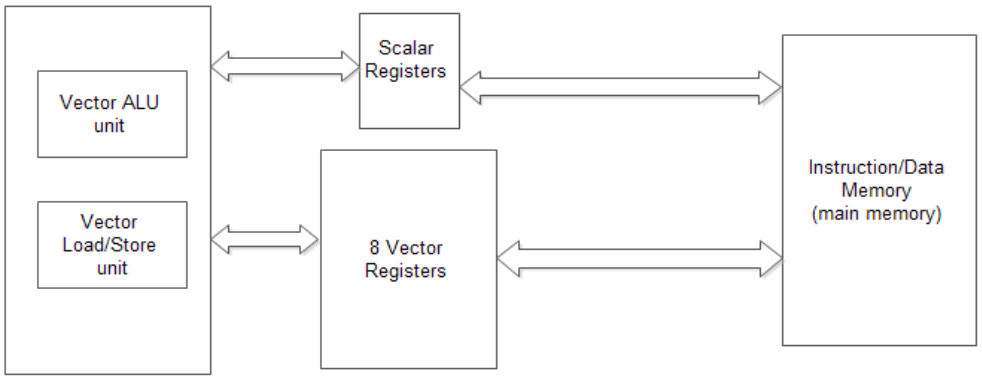
–32 مسجلا للأغراض العامة كل منها بطول كلمة واحدة.

– 8 مسجلات شعاعية طول كل منها 32 (أي كل منها مكون من 32 مسجل عددي وكل مسجل عددي بطول كلمة واحدة) .

– وحدة تحميل/تخزين شعاعية.

– وحدة حسابية شعاعية.

– أنبوب وحيد. كما يظهر في الشكل(5)

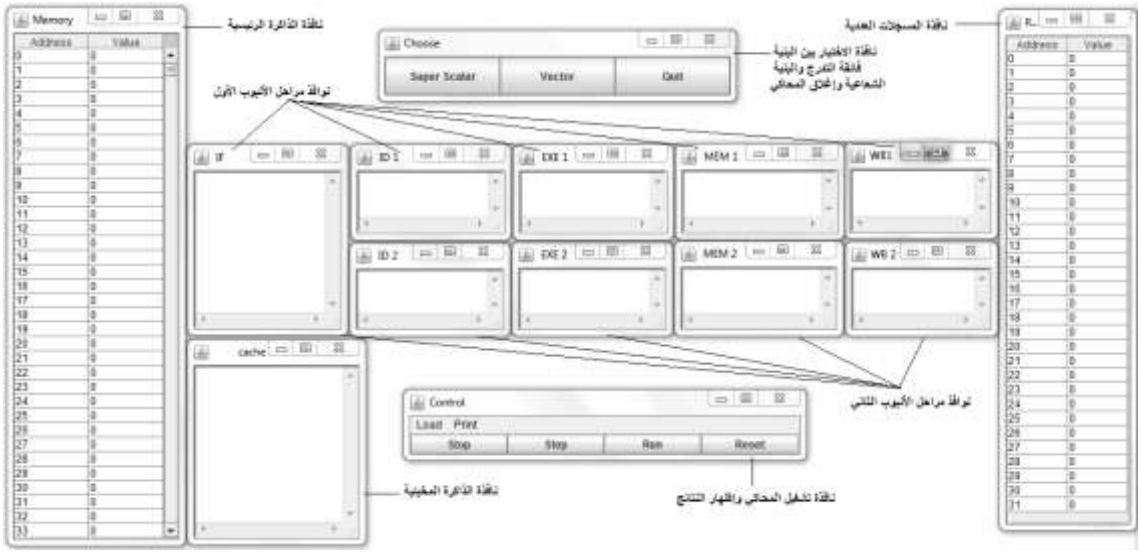


الشكل(5) بنية المعالج الشعاعي الذي تمت محاكاته

تم تصميم ISA بسيطة للمعالجات الشعاعية دعوناها VMIPS ISA حيث اعتمدنا MIPS ISA كقاعدة تمت الإضافة عليها، أهم الإضافات هي المسجلات الشعاعية (8 مسجلات شعاعية كل منها طوله 32 عنصراً عددياً) وإضافة العمليات الشعاعية مثل عملية ضرب قيمة عددية بمسجل شعاعي MULSV وجمع مسجلين شعاعيين لبعضهما ADDV.

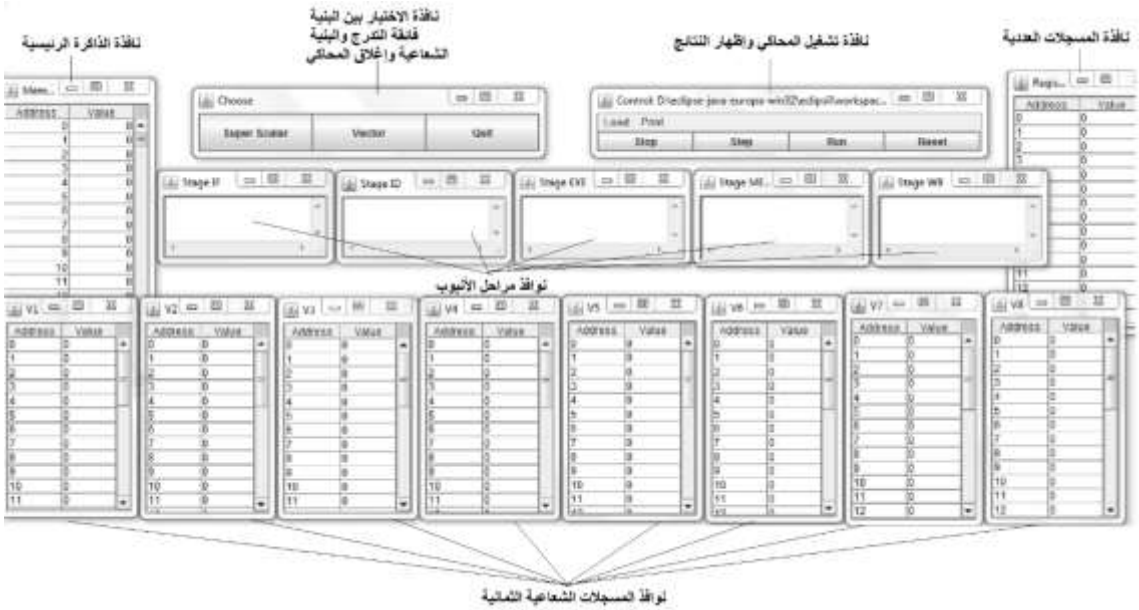
تم اعتبار تأخير الولوج للذاكرة واحد لكلا المعالجين ويساوي أربع دورات ساعة.

يظهر الشكل(6) واجهة المحاكي للمعالج فائق التدرج المكونة من نافذة التنقل بين البنيتين فائقة التدرج والشعاعية وعناصر البنية حيث تحدد خانة الذاكرة بالعنوان address والقيمة التي تحويها value وهي عبارة عن القيمة العشرية التي يمثلها بايت من البيانات وبالتالي تخزن الكلمة (32 بت) بأربع مواقع متعاقبة من الذاكرة، تحوي نافذة المسجلات رقم المسجل في الخانة address والقيمة التي يحويها في الخانة value وهي عبارة عن كلمة واحدة (32) بت، وستبين نافذة الذاكرة المخبئية كل العمليات التي ستجري ضمنها من قراءة وكتابة، يتم تحميل الملف الحاوي على البرنامج المراد تنفيذه محولاً من MIPS assembly إلى لغة الآلة من خلال نافذة التحكم بسير التنفيذ، وتظهر بوضوح مراحل الأنابيب التي ستبين سير المعالجة في كل خطوة.



الشكل(6) واجهة المحاكي للمعالج فائق التدرج

يظهر الشكل(7) واجهة المحاكي للمعالج الشعاعي حيث تظهر عناصر البنية من ذاكرة ومسجلات عددية وثمانية مسجلات شعاعية ومراحل الأنبوب التي ستبين سير المعالجة خطوة بخطوة بالإضافة لنافذة التحكم والاختيار التي سبق الحديث عنها، حيث سيتم تحميل الملف الحاوي على البرنامج المراد تنفيذه محولاً من VMIPS assembly إلى لغة الآلة ومراحل الأنبوب ستبين سير المعالجة في كل خطوة.



الشكل(7) واجهة المحاكي للمعالج الشعاعي

النتائج والمناقشة:

استخلصنا من خلال البحث أن أهم الفروقات بين المعالجات فائقة التدرج والمعالجات الشعاعية هي طريقة التعامل مع الذاكرة، ولتبيان ذلك تم تحميل شعاع من العناصر (32 عنصراً) لكلا المعالجين وتقييم أدائهما في تنفيذ عملية التحميل.

كود التحميل للمعالج فائق التدرج: كما هو مبين في الشكل(8)

```
n:   .word 32          تخزين القيمة 32 بالذاكرة
c:   .word 1           تخزين القيمة 1 بالذاكرة
s:   .word 4           تخزين القيمة 4 بالذاكرة
m:  .word 5 2 3 4 5 6 9 8 9 10 11 12 13 15 15 16 18 19 50 21 22 23 24 55 26 27 28 25 30 31 60
                       تخزين مجموعة القيم بالذاكرة
lw s8,n(s0)        تحميل المسجل رقم 8 بمحتوى موقع الذاكرة المعنون بـ(عنوان n + محتوى المسجل رقم 0 وهو دائماً 0)
lw s6,c(s0)        تحميل المسجل رقم 6 بمحتوى موقع الذاكرة المعنون بـ(عنوان c + محتوى المسجل رقم 0)
lw s5,s(s0)        تحميل المسجل رقم 8 بمحتوى موقع الذاكرة المعنون بـ(عنوان s + محتوى المسجل رقم 0)
loop:slt s9,s4,s8    وضع 1 في المسجل رقم 9 في حال قيمة المسجل رقم 4 أقل من قيمة المسجل رقم 8 ووضع 0 خلاف ذلك
    beq s9,s0,loopexit     الخروج من الحلقة في حال تساوي قيم المسجلين رقم 9 ورقم 0
    lw   s1,m(s7)      تحميل المسجل رقم 1 بمحتوى موقع الذاكرة المعنون بـ(عنوان m + محتوى المسجل رقم 7)
    add s4,s4,s6       إضافة محتوى المسجل رقم 6 إلى المسجل رقم 4 (زيادة الحلقة بمقدار 1)
    add s7,s7,s5       إضافة محتوى المسجل رقم 5 إلى المسجل رقم 7 (الانتقال للعنوان التالي من الذاكرة)
j loop                 القفز إلى بداية الحلقة
  Loopexit             الخروج من الحلقة
```

الشكل(8) كود تحميل 32 عنصراً للمعالج فائق التدرج

كود التحميل للمعالج الشعاعي: يمكن تحميل 32 عنصراً في مسجل شعاعي واحد، كما هو مبين في الشكل(9)

```
v:   .word  5 2 3 4 5 6 9 8 9 10 11 12 13 15 15 16 18 19 50 21 22 23 24 55 26 27 28 25 30 31 60
                                تخزين قيم الشعاع بالذاكرة
lv   v1,v(s0)          تحميل عناصر الشعاع إلى المسجل الشعاعي رقم 1 إبتداء من العنوان (عنوان v + محتوى المسجل رقم 0)
```

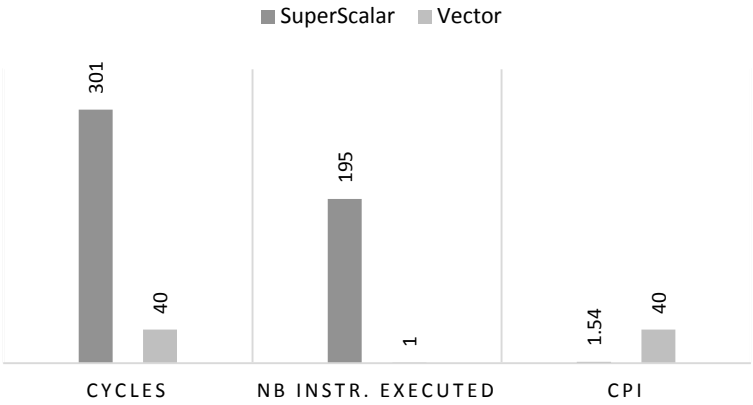
الشكل(9) كود تحميل 32 عنصراً للمعالج الشعاعي

بفرض زمن دورة الساعة يساوي 1ns لكلا المعالجين تظهر نتائج التنفيذ بواسطة المحاكي في الجدول(1).

الجدول(1) نتائج تنفيذ تحميل عدد من العناصر بواسطة المحاكي

نوع المعالج	SuperScalar Processor	Vector Processor
عدد دورات الساعة Cycles	301	40
عدد التعليمات المنفذةIC	195	1
متوسط التعليمات المنفذة في الدورة الواحدة CPI	1.54	40
CPU time	301 n.s	40 n.s

يظهر من الجدول(1) أن تحميل 32 قيمة بالمعالج فائق التدرج يتطلب حلقة لإدخال القيم ليصل مجموع التعليمات المنفذة إلى 195 تعليمة نفذّت بـ301 دورة ساعة بينما تم ذلك بتعليمة وحيدة في المعالج الشعاعي تمثل تحميل شعاع واحد نفذّت بـ40 دورة ساعة، كما يظهر بيانياً من خلال الشكل(10).



الشكل(10) نتائج تنفيذ عملية تحميل قيم بواسطة المحاكي

تكون نسبة التحسين التي قدمها المعالج الشعاعي وفق المعادلة(3):

Speedup = 301 ÷ 40 = 7.5%

نلاحظ أن المعالج الشعاعي يظهر نسبة تحسين كبيرة في عمليات التحميل فهو أسرع بأكثر من سبع مرات من المعالج فائق التدرج.

ثم تم تطبيق معادلة SAXPY على 32 عنصراً لتبيان فروقات التنفيذ فيما يخص عمليات DLP:

كود MIPS assembly المكافئ لحلقة معادلة SAXPY في المعالج فائق التدرج: كما هو مبين في الشكل(11)

```
n:   .word 32        تخزين القيمة 32 بالذاكرة
c:   .word 1         تخزين القيمة 1 بالذاكرة
s:   .word 4         تخزين القيمة 4 بالذاكرة
b:   .word 5         تخزين القيمة 5 بالذاكرة
t:   .word           حجز عنوان في الذاكرة لتعليمة تخزين
l:   .word 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
                     تخزين مجموعة القيم الأولى بالذاكرة
k:   .word 5 2 3 4 5 6 9 8 9 10 11 12 13 15 15 16 18 19 50 21 22 23 24 55 26 27 28 25 30 31 60
                     تخزين مجموعة القيم الثانية بالذاكرة
 lw s8,n(s0)         تحميل المسجل رقم 8 بمحتوى موقع الذاكرة المعنون بـ(عنوان n + محتوى المسجل رقم 0 وهو دائماً 0)
 lw s6,c(s0)         تحميل المسجل رقم 6 بمحتوى موقع الذاكرة المعنون بـ(عنوان c + محتوى المسجل رقم 0)
 lw s5,s(s0)         تحميل المسجل رقم 5 بمحتوى موقع الذاكرة المعنون بـ(عنوان s + محتوى المسجل رقم 0)
 lw s10,b(s0)        تحميل المسجل رقم 10 بمحتوى موقع الذاكرة المعنون بـ(عنوان b + محتوى المسجل رقم 0)
loop:slt s9,s4,s8    وضع 1 في المسجل رقم 9 في حال قيمة المسجل رقم 4 أقل من قيمة المسجل رقم 8 ووضع 0 خلاف ذلك
    beq s9,s0,loopexit   الخروج من الحلقة في حال تساوي قيم المسجلين رقم 9 ورقم 0
    lw  s1,l(s7)         تحميل المسجل رقم 1 بمحتوى موقع الذاكرة المعنون بـ(عنوان l + محتوى المسجل رقم 7)
    mul s1,s1,s10        ضرب محتوى المسجل رقم 10 بمحتوى المسجل رقم 1 وتخزين النتيجة في المسجل رقم 1
    lw  s2,k(s7)         تحميل المسجل رقم 2 بمحتوى موقع الذاكرة المعنون بـ(عنوان k + محتوى المسجل رقم 7)
    add s3,s1,s2         جمع قيم المسجلين 1 و2 ووضع النتيجة في المسجل رقم 3
    sw  s3,t(s7)         تخزين قيمة المسجل رقم 3 بموقع الذاكرة المعنون بـ(عنوان t + محتوى المسجل رقم 7)
    add s4,s4,s6         إضافة محتوى المسجل رقم 6 إلى المسجل رقم 4 (زيادة الحلقة بمقدار 1)
    add s7,s7,s5         إضافة محتوى المسجل رقم 5 إلى المسجل رقم 7 (الانتقال للعنوان التالي من الذاكرة)
j loop                   القفز إلى بداية الحلقة
 loopexit                الخروج من الحلقة
```

الشكل(11) كود SAXPY للمعالج فائق التدرج

كود VMIPS assembly المكافئ لحلقة معادلة SAXPY في المعالج الشعاعي: كما يظهر في الشكل(12)

```
b:   .word 5         تخزين القيمة 5 بالذاكرة
t:   .word           حجز عنوان في الذاكرة لتعليمة تخزين
x:   .word 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
                     تخزين قيم الشعاع الأول بالذاكرة
y:   .word 5 2 3 4 5 6 9 8 9 10 11 12 13 15 15 16 18 19 50 21 22 23 24 55 26 27 28 25 30 31 60
                     تخزين قيم الشعاع الثاني بالذاكرة
lw   s1,b(s0)        تحميل المسجل رقم 1 بمحتوى موقع الذاكرة المعنون بـ(عنوان b + محتوى المسجل رقم 0)
lv   v1,x(s0)        تحميل عناصر الشعاع إلى المسجل الشعاعي رقم 1 إبتداء من العنوان (عنوان x + محتوى المسجل رقم 0)
mulsv v1,s1,v1       ضرب القيمة العددية الموجودة بالمسجل 1 بعناصر الشعاع الأول
lv   v2,y(s0)        تحميل عناصر الشعاع إلى المسجل الشعاعي رقم 2 إبتداء من العنوان (عنوان y + محتوى المسجل رقم 0)
addv v3,v1,v2        جمع عناصرالمسجل الشعاعي رقم1 لعناصر المسجل الشعاعي رقم2 ووضع النتيجة في المسجل الشعاعي رقم3
sv   v3,t(s0)        تخزين الشعاع3 ابتداء من موقع الذاكرة المعنون بـ(عنوان t + محتوى المسجل رقم 0)
```

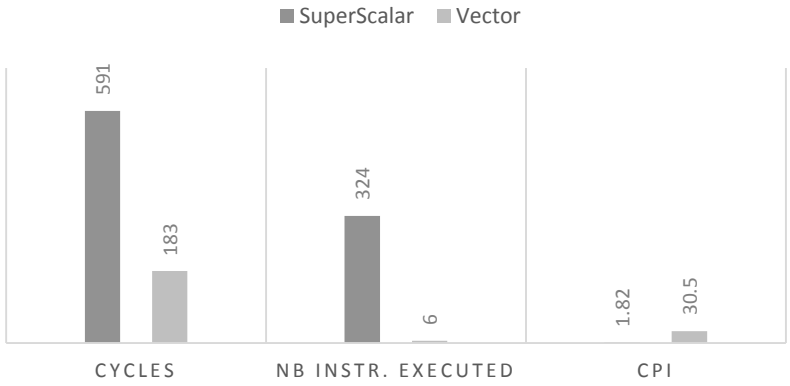
الشكل(12) كود SAXPY للمعالج الشعاعي

تظهر نتائج تنفيذ SAXPY بواسطة المحاكي في الجدول(2):

جدول(2) نتائج تنفيذ SAXPY بواسطة المحاكي

Vector Processor	SuperScalar Processor	نوع المعالج
183	591	عدد دورات الساعة Cycles
6	324	عدد التعليمات المنفذه IC
30.5	1.82	متوسط التعليمات المنفذة في الدورة الواحدة CPI
183 n.s	591 n.s	CPU time

يظهر من الجدول(2) أن المعالج الشعاعي نفذ SAXPY بعدد دورات أقل بكثير وبعدد تعليمات قليل جداً ولكن بمعدل CPI أكبر بكثير ويعود ذلك لكون العملية الشعاعية تكافئ عدد كبير من التعليمات العددية (حلقة كاملة)، كما يظهر بيانياً من خلال الشكل(13)



شكل (13) نتائج تنفيذ SAXPY بواسطة المحاكي

تكون نسبة التحسين التي قدمها المعالج الشعاعي وفق المعادلة(3):

$$Speedup = 591 \div 183 = 3.2\%$$

نلاحظ أن المعالج الشعاعي أظهر نسبة تحسين في تنفيذ SAXPY فهو أسرع بأكثر من ثلاث مرات من المعالج فائق التدرج، وبالتالي يمكننا القول إن المعالج الشعاعي أسرع بشكل ملحوظ من المعالج فائق التدرج في معالجة تعليمات التوازي على مستوى البيانات DLP.

الاستنتاجات والتوصيات:

بعد الانتهاء من البحث خلصنا الى مجموعة من الاستنتاجات، أهمها :

– تعتمد فعالية تنفيذ التعليمات على التوازي بشكل كبير وأساسي على اختيار بنية الحاسب المناسبة للتنفيذ وفق نوع التوازي الممكن تطبيقه على التعليمات.

- عندما يتعلق الأمر بمعالجة كمية كبيرة من المعطيات وفق تعليمة محددة فإن خواص الشعاع تأخذ أهمية كبيرة.
- يظهر الحاسب الشعاعي فعالية كبيرة في إنقاص زمن الولوج إلى الذاكرة.
- يظهر الحاسب الشعاعي فعالية كبيرة لايمكن إغفالها عند التفرع على مستوى البيانات.
- تطوير بيئة المحاكاة السابقة لتشمل أكبر قدر ممكن من بنى المعالجات لتساهم في فهم الباحثين والمطورين لها كون تنفيذها على أرض الواقع مكلف وغير متاح بصورة وافية.

المراجع:

[1] Asanovi´c,K.*Vector Microprocessors*. UNIVERSITY of CALIFORNIA, BERKELEY,1998, 6-8,21.

[2] Hennessy,J;Patterson,D.*Computer Architecture A Quantitative Approach*.Fifth Edition,University of California, Berkeley, 2012, 10-15,150-156.

[3] LEE,G.*Future Information Engineering*. WIT press, 2014, 666-674**.**

[4] Silc,J; Robic,B; Ungerer,T. *Processor Architecture: From Dataflow to Superscalar and Beyond*. Springer Science & Business Media, 1999, 18-32.

[5] GODSE,A,P;GODSE,D,A.Computer Architecture.fourth edition,Technical Publications, 2010, 55-60.

[6] Dongarra.G;Duff.I;Sorensen.D;Ven der vost.H.*Numerical Linear Algebra for High-Paerformance Computers*. Society for industrial and applied mathimatics, 1998,71-73.

[7] David J,Eck.*Introduction to Programming Using Java Version 5.1.2*. Hobart and William Smith Colleges, June 2010, 5.