Sharwes Scientific Journal (SSJ)
Volume 7, Issue 2, 2026, pp 350-370.
ISSN (Print): 3079-6040 • ISSN (Online): 3135-4343

Architectural Study and Simulation of RISC and CISC

Wisam Elwalda

The Higher Institute of Science and Technology, Misurata, Libya.

*Corresponding author email: Wisam Elwalda | w_elwalda@yahoo.com

Received: 09-02-2026 | Accepted: 25-07-2026 | Available online: 10-08-2026 | DOI:10.5281/zenodo.21864448

ABSTRACT

The microprocessor unit (MPU) or central processing unit (CPU) constitutes the fundamental component that determines the performance of computing systems, as it executes instructions and coordinates the operation of system components according to a defined architecture. Advances in integrated circuit technologies have led to increasingly complex designs, making the evaluation of instruction set architectures (ISA) essential for understanding processor operational characteristics. Two major paradigms have emerged: Complex Instruction Set Computer (CISC), which provides multifunctional instructions within a complex structure that allows high-level operations to be executed with a single instruction but requires more intricate design; and Reduced Instruction Set Computer (RISC), which relies on a limited set of simple instructions, typically executed in a single clock cycle, thereby enhancing efficiency and speed while simplifying design. This study employs MATLAB to simulate selected instructions in both RISC and CISC architectures, serving as a practical example of distinctive instructions in each paradigm. The results were compared in terms of instruction count, clock cycles, code size, and execution efficiency. The simulation outcomes demonstrated the superiority of RISC architecture over CISC in terms of execution time and overall efficiency, despite requiring a larger number of instructions and greater program size. Conversely, the CISC architecture relied on a single complex instruction that reduced program size but resulted in longer execution time and lower efficiency. The study recommends expanding the scope of simulation by incorporating additional instructions such as addition, subtraction, and branching to assess architectural behavior under more complex conditions. Furthermore, integrating additional metrics such as power consumption would provide a more comprehensive perspective on performance.

Keywords: RISC, CISC, CPU, architecture.

دراسة ومحاكاة معمارية المعالجات RISC وCISC

وسام الولده

قسم تقنيات الحاسوب، المعهد العالي للعلوم والتقنية، مصراته، ليبيا

*المؤلف المراسل: وسام الولده | w_elwalda@yahoo.com

استقبلت: 09-02-2026م | قبلت: 25-07-2026م | متوفرة على الانترنت | 10-08-2026م | DOI:10.5281/zenodo.21864448

ملخـــــص البحـــــث

تُعد وحدة المعالج الدقيق (Microprocessor Unit – MPU) أو وحدة المعالجة المركزية (Central Processing Unit – CPU) المكوّن الأساسي الذي يحدد أداء الأنظمة الحاسوبية، إذ تنفذ التعليمات وتنسق عمل المكونات وفقًا لمعمارية محددة، وقد أدى التطور في تقنيات الدوائر المتكاملة إلى ظهور تصاميم أكثر تعقيدًا، مما جعل تقييم معماريات مجموعة التعليمات (Instruction Set Architecture – ISA) أمرًا جوهريًا لفهم الخصائص التشغيلية للمعالج، حيث برز اتجاهان رئيسيان هما الحوسبة ذات مجموعة التعليمات المعقدة (CISC) Complex Instruction Set Computer التي توفر تعليمات متعددة الوظائف ضمن بنية معقدة تسمح بتنفيذ عمليات عالية المستوى بتعليمة واحدة لكنها تتطلب تصميمًا أكثر تعقيدًا، والحوسبة ذات مجموعة التعليمات المبسطة Reduced Instruction Set

"Articles published in *Sharwes Sci J* are licensed under a Creative Commons Attribution-Non-Commercial 4.0 International License."

Computer (RISC) التي تعتمد على تعليمات محدودة وبسيطة تُنفذ غالبًا في دورة ساعة واحدة مما يعزز الكفاءة والسرعة ويبسّط التصميم ؛ تعتمد هذه الدراسة على محاكاة تعليمات محددة في كل من RISC و CISC باستخدام بيئة MATLAB كمثال عملي للتعليمات المميزة في كلا النمطين، حيث تمت مقارنة النتائج من حيث عدد التعليمات، وعدد دورات الساعة، وحجم الكود، وكفاءة التنفيذ، وقد أظهرت نتائج المحاكاة تفوق بنية RISC على بنية CISC من حيث زمن التنفيذ والكفاءة الإجمالية على الرغم من حاجتها إلى عدد أكبر من التعليمات وحجم أكبر للبرنامج، في حين أن بنية CISC اعتمدت على تعليمة واحدة معقدة قللت حجم البرنامج لكنها أدت إلى زيادة زمن التنفيذ وانخفاض الكفاءة، كما وتوصي الدراسة بتوسيع نطاق المحاكاة بإضافة تعليمات جديدة مثل الجمع والطرح والقفز لتقييم سلوك البنية في ظروف أكثر تعقيدًا، ودمج مقاييس إضافية مثل استهلاك الطاقة لتوفير رؤية شاملة حول الأداء.

الكلمات المفتاحية: تعليمات RISC ، تعليمات CISC ، معالج CPU ، معمارية.

1. مقدمة

نظراً لتنوع المتطلبات والمهام التي نواجهها في حياتنا اليومية، تُعدّ الحواسيب ذات القدرات المختلفة ضرورية لتلبية هذه المتطلبات بكفاءة. فعلى سبيل المثال، تختلف الحواسيب المستخدمة في البحث العلمي اختلافاً كبيراً عن الحواسيب الشخصية الموجودة في المنازل والمكاتب. وغالباً ما تتطلب التطبيقات العلمية أنظمة حوسبة عالية الأداء، بينما يمكن إنجاز المهام اليومية – مثل عرض الصور ومشاهدة الأفلام وتصفح الإنترنت – بواسطة أجهزة أقل قوة.

ولتلبية هذا التنوع في التطبيقات، تُستخدم أنواع مختلفة من المعالجات الدقيقة. يعتمد أداء المعالج الدقيق على عدة عوامل، منها سرعة الساعة (clock)، وعرض ناقل البيانات، وعدد الأنوية (أحادية النواة أو متعددة الأنوية)، مع ذلك، هناك عامل حاسم غالباً ما يُغفل عنه في الاستخدام اليومي، وهو بنية المعالج – التصميم الهيكلي للمعالج ومكوناته–، والذي يُحدد كيفية جلب البيانات ومعالجتها وتنفيذها.

في هذه الورقة، نلقي الضوء على نموذجين معماريين رئيسيين في تصميم المعالجات الدقيقة:

حاسوب مجموعة التعليمات المعقدة.(Complex Instruction Set Computer) CISC و حاسوب مجموعة التعليمات المختصرة(Reduced Instruction Set Computer) RISC .

قبل الخوض في التفاصيل المعمارية لكل منهما، من المهم فهم البنية الداخلية لوحدة المعالجة المركزية (CPU) وكيف تؤثر الخيارات المعمارية على الأداء العام.[1] [2]

2. منهجية البحث

يعتمد هذا البحث على محاكاة معماريتي RISC و CISC لمقارنة أدائهما في تنفيذ المهمة نفسها من حيث عدد التعليمات، وعدد دورات الساعة، وحجم الكود، وكفاءة التنفيذ. وقد أُجريت المحاكاة باستخدام لغة

MATLAB، التي توفر بيئة مرنة لإنشاء سيناريوهات متنوعة لتحليل أداء كل معمارية.

3. الخطوات المتبعة في بناء البحث

1.3 البرمجيات

البرنامج المستخدم في هذا البحث هو لغة الماتلاب MATLAB ، وهو بيئة برمجة سهلة الاستخدام ومتوافقة مع معظم أنظمة التشغيل. يوفر هذا البرنامج المرونة والكفاءة في محاكاة بنى المعالجات المختفة.

2.3 الجزء العملي

تم استخدام المحاكاة في لغة الماتلاب لمحاكاة تعليمات محددة في RISC و CISC كمثال للتعليمات المستخدمة في هذين النوعين من التعليمات ثم قورنت النتائج لتقييم اختلافات الأداء.

3.3 أسلوب البحث والعمليات التي تمت فيه

يعتمد كل نوع من المعالجات على بنية محددة لأداء مهامه بكفاءة ودقة. وتُعدّ السرعة والكفاءة وحجم التعليمات من أهم التحديات في تصميم المعالجات.

في هذه الورقة، تمّت دراسة ومحاكاة كلٍّ من بنيتي RISC و CISC لاستكشاف خصائصهما ومزاياهما وحدود امكانياتهما على حدة.

4. بنية المعالج الدقيق

يُعدّ المعالج الدقيق بمثابة العقل المركزي لنظام الحوسبة، فهو مسؤول عن استقبال التعليمات والعناوين، ومعالجة البيانات، وتقديم النتائج على شكل معلومات ذات معنى. من الناحية المادية، يتكون المعالج عادةً من دائرة متكاملة خفيفة الوزن مربعة الشكل، مزودة بشبكة من الدبابيس على سطحها السفلي. تُسهّل هذه الدبابيس الاتصال بين المعالج ومقبس اللوحة الأم، مما يُمكّن من تبادل البيانات بسلاسة بين مكونات النظام. داخليًا، يتألف المعالج من ملايين الترانزستورات المُكدّسة بكثافة على شريحة رقيقة من السيليكون، مُشكّلةً أساس قدراته الحسابية.

تتحدد بنية المعالج الدقيق بعدة وحدات وظيفية أساسية، تُساهم كل منها في أدائه العام وسلوكه التشغيلي.

1.4. وحدة التحكم Control Unit (CU)

تتولى وحدة التحكم تنسيق تدفق البيانات داخل المعالج. فهي تدير التنسيق بين المكونات الداخلية، وتوجه تنفيذ التعليمات، وتنظم توقيت العمليات. وبصفتها العنصر الإشرافي للمعالج، تضمن وحدة التحكم عمل جميع الأنظمة الفرعية بتناغم، مما يجعلها مكونًا أساسيًا في بنية المعالج.

2.4 وحدة واجهة الناقل(BIU) Bus Interface Unit

تتولى وحدة واجهة الناقل مسؤولية نقل البيانات بين المعالج والمكونات الخارجية، وخاصة الذاكرة الرئيسية للنظام (RAM). فهي تنظم وتدير مسارات البيانات، مما يضمن اتصالاً فعالاً بين المعالج وعناصر الأجهزة الأخرى مثل أجهزة الإدخال/الإخراج ووحدات الذاكرة.

3.4 وحدة الحساب والمنطق Arithmetic and Logic Unit (ALU)

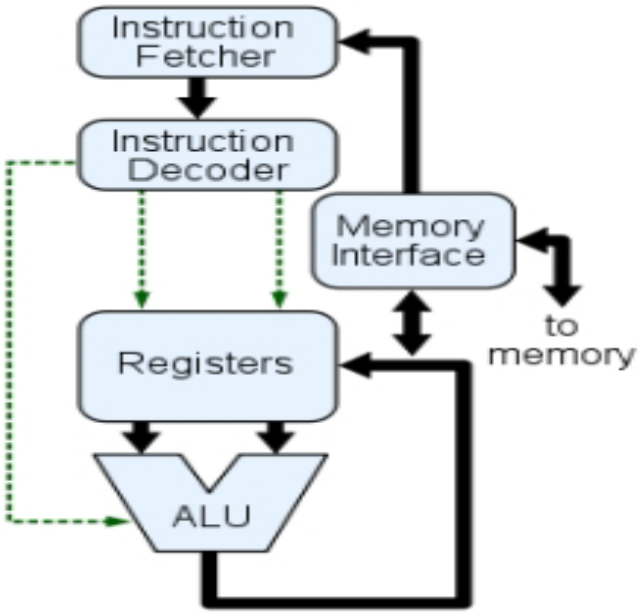
تتولى وحدة الحساب والمنطق (ALU) مسؤولية تنفيذ العمليات الحسابية والمقارنات المنطقية. وهي مقسمة إلى وحدتين متخصصتين:

•وحدة الأعداد الصحيحة Integer Unit : تتولى هذه الوحدة العمليات الحسابية التي تتضمن أعدادًا صحيحة بدون فواصل عشرية. وتُستخدم بكثرة في التطبيقات التي تعتمد على النظام الثنائي، مثل معالجة النصوص، وبرامج العروض التقديمية، والبرامج العامة.

• وحدة الفاصلة Floating Point Unit (FPU): تُنفذ هذه الوحدة العمليات الحسابية التي تتضمن أعدادًا عشرية. وهي بالغة الأهمية للتطبيقات التي تتطلب حسابات عالية الدقة، مثل ألعاب الفيديو ثلاثية الأبعاد، وبرامج التصميم مثل أوتوكاد.

4.4 المسجلات Registers

المسجلات هي وحدات ذاكرة فائقة السرعة وصغيرة السعة مُدمجة داخل المعالج. تقوم بتخزين البيانات والتعليمات مؤقتًا أثناء التنفيذ، مما يسمح بالوصول السريع إليها من قِبل وحدة الحساب والمنطق ووحدة التحكم. تُنفذ هذه المسجلات عادةً باستخدام تقنية ذاكرة الوصول العشوائي الثابتة (SRAM)، التي توفر أداءً عالي السرعة و ضروريًا لمهام المعالجة في الوقت الحقيقي كما هو موضح في الشكل1 [4][3].



شكل 1 : مخطط وحدة المعالجة المركزية

5. خطوات تشغيل المعالج

يتبع تشغيل المعالج الدقيق تسلسلًا منظمًا من الخطوات التي تُمكّنه من تنفيذ التعليمات وإدارة تدفق البيانات بكفاءة. تُعد هذه الخطوات أساسية لوظائف المعالج، وعادةً ما تُصنّف كما يلي:

1. جلب التعليمات: يسترجع المعالج التعليمات المخزنة في ذاكرة الوصول العشوائي Random Access Memory (RAM) للنظام. تتضمن هذه الخطوة الوصول إلى موقع الذاكرة الذي توجد فيه التعليمات وتحميلها في سجل تعليمات المعالج.

2. فك التشفير: بمجرد جلب التعليمات، يُفسّرها المعالج أو يفك تشفيرها لتحديد العملية المطلوبة والبيانات المرتبطة بها. تتضمن هذه الخطوة تحديد نوع التعليمات وإعداد المعاملات اللازمة.

3. التنفيذ: يُنفّذ المعالج العملية المحددة، مثل العمليات الحسابية أو نقل البيانات. عند الانتهاء، تُكتب النتائج مرة أخرى إلى الذاكرة أو تُخزّن في السجلات لاستخدامها لاحقًا.

تتكرر دورة الجلب و فك التشفير و التنفيذ هذه باستمرار، حيث تشكل أساس جميع العمليات الحسابية داخل وحدة المعالجة المركزية (CPU).[4]

1.5 سرعة المعالج

تُعدّ سرعة المعالج عاملاً حاسماً يؤثر على أداء النظام واستجابته. تُقاس عادةً بالميغاهرتز (MHz) أو الغيغاهرتز (GHz) ، وهو ما يُمثل عدد دورات الساعة التي يُمكن للمعالج تنفيذها في الثانية الواحدة.

تُقسم سرعة المعالج عادةً إلى فئتين:

- سرعة الساعة الداخلية:

تشير الساعة الداخلية إلى التردد الذي تُجرى به العمليات داخل المكونات الأساسية للمعالج. على سبيل المثال، يُمكن لمعالج بسرعة ساعة داخلية تبلغ 500 ميغاهرتز تنفيذ 500 مليون دورة في الثانية. تُمكّن سرعات الساعة الداخلية الأعلى المعالج من تنفيذ المزيد من التعليمات في وحدة الزمن، مما يُحسّن الإنتاجية الحسابية.

- سرعة الساعة الخارجية:

تُعرف أيضاً بسرعة ناقل النظام، وتتحكم الساعة الخارجية في معدل تبادل البيانات بين المعالج ومكونات النظام الأخرى، مثل وحدة الذاكرة أو مجموعة الشرائح الأخرى. على سبيل المثال، يُمكن لمعالج بنتيوم 3 بسرعة ساعة خارجية تبلغ 133 ميغاهرتز نقل البيانات بمعدل 133 مليون نبضة في الثانية عبر الناقل. إذا كانت سرعة الساعة الخارجية أقل بكثير من السرعة الداخلية، فقد يواجه المعالج اختناقات بسبب محدودية إدخال البيانات، مما يحول دون الاستفادة الكاملة من قدراته الداخلية.

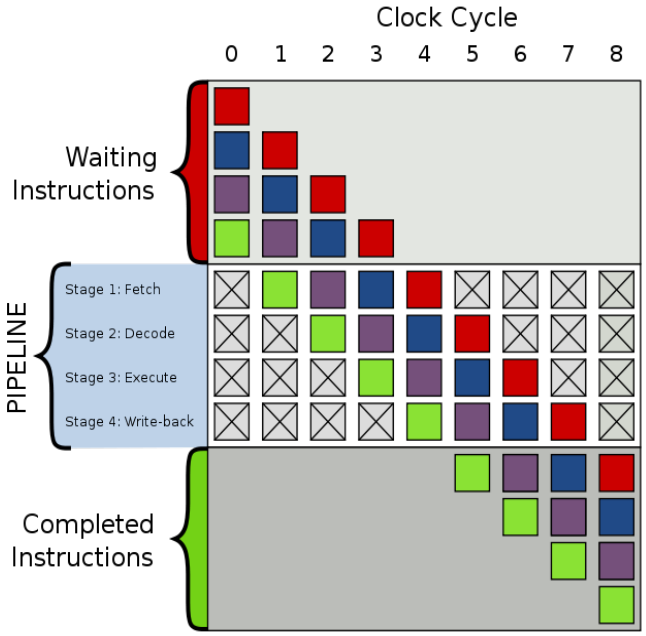
1.1.5 ميزات تحسين الأداء

تتضمن المعالجات الحديثة تحسينات معمارية لرفع سرعة التنفيذ وكفاءته، ومن هذه الميزات:

- بنية المعالجة الفائقة Superscalar Architecture : يحتوي المعالج الفائق على مسارات تنفيذ متعددة، مما يسمح له بمعالجة أكثر من تعليمة في وقت واحد. على سبيل المثال، إذا وصلت تعليمتان في وقت واحد، وكان المعالج مزودًا بوحدتي تنفيذ، فيمكن معالجتهما بالتوازي، مما ينتج عنه إخراج أسرع وأداء أفضل.
- خطوط الأنابيب Pipelining : خطوط الأنابيب هي تقنية لتحسين الأداء تُستخدم في تصميم المعالجات الدقيقة الحديثة لزيادة إنتاجية التعليمات، وهي تتضمن تقسيم دورة معالجة التعليمات إلى مراحل متسلسلة متعددة، حيث تكون كل مرحلة مسؤولة عن جزء محدد من عملية التنفيذ، مثل جلب البيانات، وفك تشفيرها، وتنفيذها، وكتابة النتائج.

بدلاً من انتظار اكتمال تعليمة واحدة قبل بدء التالية، تسمح تقنية خطوط الأنابيب بمعالجة عدة تعليمات في وقت واحد، حيث تشغل كل تعليمة مرحلة مختلفة من خط الأنابيب. يُحسّن هذا التداخل في العمليات كفاءة التنفيذ بشكل ملحوظ ويقلل وقت الخمول داخل المعالج.

على سبيل المثال، أثناء فك تشفير تعليمة ما، يمكن جلب تعليمة أخرى، وتنفيذ تعليمة ثالثة ، كل ذلك خلال دورة الساعة نفسها. والنتيجة هي تدفق أكثر سلاسة وانسيابية لمعالجة التعليمات، مما يؤدي إلى أداء عام أعلى. كما هو موضح في الشكل 2.[6][5][4]



شكل 2 : خطوط الأنابيب

الان بعد محاولة البسط لبنية المعالجات الدقيقة وآلياتها التشغيلية، ننتقل الآن إلى نموذجين معماريين رئيسيين شكّلا تصميم المعالجات الحديثة: مجموعة التعليمات المعقدة Complex Instruction Set Computer (CISC) و مجموعة التعليمات المختصرة Reduced Instruction Set Computer (RISC) يمثل هذان النموذجان نهجين مختلفين جذريًا لبنية مجموعة التعليمات(ISA) ، من حيث تعقيد المكونات المادية، وكفاءة التنفيذ. تهدف بنية CISC إلى تقليل عدد التعليمات في كل برنامج من خلال دمج العمليات المعقدة متعددة الخطوات في تعليمات منفردة، في المقابل، تُبسّط بنية RISC مجموعة التعليمات، مُفضّلةً عددًا أكبر من التعليمات المُبسّطة التي تُنفّذ بسرعة وبشكل موحد.

فيما يلي نحاول توضيح الخصائص الهيكلية، وفلسفات التصميم، وتأثيرات الأداء لكل من بنيتي CISC وRISC، عن طريق تقديم تحليلًا مُقارنًا قائمًا على المبادئ النظرية والمعايير العملية.

6. مجموعة التعليمات المعقدة (CISC)

1.6 الخلفية التاريخية

يمكن تتبع ظهور بنية حاسوب مجموعة التعليمات المعقدة (CISC) إلى تطوير نظام IBM System/360

عام 1964، والذي شكّل علامة فارقة في تطور بنية الحاسوب، قدّم هذا النظام مفهوم التحكم المُبرمج جزئيًا، مما أتاح تنفيذًا مرنًا للتعليمات المعقدة. ومع تزايد متطلبات الحوسبة خلال ستينيات القرن العشرين، أصبحت بنية CISC أكثر انتشارًا، وتميزت بتصاميم أجهزة معقدة ومجموعات تعليمات واسعة.

استمر هذا التوجه المعماري عبر الأجيال المتعاقبة من المعالجات، بما في ذلك Intel 80486 و Pentium و MMX و Pentium III، مع ذلك، بحلول منتصف سبعينيات القرن العشرين، بدأ المستخدمون والمطورون يدركون قيود تصاميم CISC. فقد أدى تعقيد فك تشفير التعليمات وتنفيذها إلى اختناقات في الأداء، كما أن العبء الإضافي المرتبط بدعم لغات البرمجة عالية المستوى جعل تحسين المُترجم أكثر صعوبة، في كثير من الحالات، لم تتمكن المُترجمات من الاستفادة الكاملة من مجموعات التعليمات الغنية التي توفرها معالجات CISC. تتمثل الفلسفة الأساسية وراء معمارية CISC في تقليل عدد التعليمات اللازمة لإنجاز مهمة ما، وبالتالي تقليل عدد أسطر كود التجميع، ويتحقق ذلك من خلال تصميم معالجات قادرة على تنفيذ عمليات متعددة الخطوات عبر تعليمة واحدة معقدة. تدعم معمارية CISC عادةً مئات التعليمات بأطوال وتنسيقات متنوعة، بعضها يسمح بالمعالجة المباشرة لمعاملات الذاكرة. ونتيجةً لذلك، يجب أن تستوعب أنظمة عنونة الذاكرة ترميزات (encodings) عناوين كبيرة ومرنة. [2][5][7]

2.6 الخصائص المعمارية

يتطلب تصميم مجموعة تعليمات CISC موازنة احتياجات لغات البرمجة على مستوى لغة الآلة ولغات البرمجة عالية المستوى. أحد الدوافع الرئيسية لاعتماد مجموعات التعليمات المعقدة هو تحسين الأداء من خلال تبسيط عملية الترجمة من التعليمات البرمجية عالية المستوى إلى تعليمات الآلة. في أنظمة CISC، غالبًا ما تستطيع المترجمات توليد تعليمة آلة واحدة لكل عبارة من عبارات لغة البرمجة عالية المستوى، مما يُبسط عملية الترجمة.

من السمات المميزة لبنية CISC تنسيق التعليمات ذي الطول المتغير، على سبيل المثال، قد تشغل تعليمة تتطلب معاملات مسجلات بايتين أو ثلاثة بايتات فقط، بينما قد تتطلب تعليمة تتضمن عنواني ذاكرة خمسة بايتات. في نظام ذي كلمات 32 بت (أربعة بايتات)، قد تستخدم تعليمة قصيرة نصف كلمة، بينما قد تمتد تعليمة أطول على كلمة كاملة بالإضافة إلى بايت إضافي، يتطلب استيعاب هذه التنسيقات المتغيرة ضمن كلمات ذاكرة ثابتة الطول دوائر فك تشفير متخصصة قادرة على تحليل التعليمات ومحاذاتها بناءً على عدد البايتات.[1][7][8][9]

3.6 مثال على التعليمات

لنفترض التعليمة التالية:

MULT 0x100, 0x101

تجري هذه التعليمة عملية ضرب لقيمتين مخزنتين في عنواني الذاكرة 0x100 و 0x101 ، أثناء التنفيذ، يقوم المعالج بتحميل القيمتين في مسجلات داخلية، ثم يُجري عملية الضرب، ويخزن النتيجة في مسجل مُخصص، تُغلف هذه التعليمة الواحدة عدة خطوات (استرجاع البيانات، والحساب، وتخزين النتيجة) مما يُبرز الطبيعة المُدمجة والقوية لتصميم تعليمات CISC.

4.6 مميزات وعيوب بنية CISC

تُقدم بنية حاسوب مجموعة التعليمات المعقدة (CISC) العديد من المزايا والقيود التي تؤثر على ملاءمتها لبيئات الحوسبة المختلفة. تتبع هذه الخصائص من فلسفة تصميمها، التي تُركز على مجموعات التعليمات الغنية والتعقيد على مستوى الأجهزة، فيما يلي ملخص لأهم المزايا والعيوب:

- **المميزات:**

– التوافق مع لغات البرمجة عالية المستوى: صُممت بنية CISC لتتوافق بشكل وثيق مع بنى البرمجة عالية المستوى، مما يسمح للمترجمات بترجمة العبارات المعقدة إلى تعليمات آلية واحدة.

– كفاءة التعليمات: يمكن لبعض التعليمات المعقدة تنفيذ عمليات متعددة الخطوات في دورة تعليمات واحدة، مما يقلل عدد التعليمات المطلوبة لكل برنامج.

– تقليل حجم الكود: من خلال دمج العمليات في عدد أقل من التعليمات، تميل برامج CISC إلى شغل مساحة ذاكرة أقل مقارنةً بنظيراتها من RISC.

- **العيوب :**

–استهلاك عالٍ للطاقة: يؤدي تعقيد دوائر فك تشفير التعليمات وتنفيذها إلى زيادة استهلاك الطاقة.

– أطوال تعليمات متغيرة: تختلف التعليمات في حجمها، مما يتطلب آليات فك تشفير متطورة ويزيد من تعقيد عملية محاذاة الذاكرة.

– وقت تنفيذ طويل: تتطلب العديد من التعليمات دورات ساعة متعددة لإكمالها، مما قد يؤثر على الأداء في التطبيقات الحساسة للوقت.

– مساحة محدودة للمسجلات: غالبًا ما يؤدي التركيز على العمليات القائمة على الذاكرة إلى تقليل عدد

المسجلات، مما يقلل من كفاءة معالجة البيانات.

– زيادة تعقيد المكونات المادية: يتطلب دعم نطاق واسع من التعليمات عددًا أكبر من الترانزستورات، مما يجعل معالجات CISC أكثر تكلفة في التصنيع.

– قيود على استخدام الذاكرة: قد تفرض التعليمات المعقدة قيودًا على عنونة الذاكرة وأنماط الوصول إليها، مما يؤثر على مرونة النظام بشكل عام. [1][2][8][9][10]

7. مجموعة التعليمات المختصرة

1.7 الخلفية التاريخية

تطورت بنية حاسوب مجموعة التعليمات المختصرة (RISC) استجابةً لتزايد التعقيد وانخفاض الكفاءة المرتبطين بمعالجات CISC. مع انخفاض تكاليف الذاكرة وتحسن تقنيات المترجمات خلال سبعينيات القرن الماضي، أصبح من الممكن إعادة النظر في فلسفة تصميم المعالجات الدقيقة من خلال تبسيط مجموعات التعليمات.

قدّم جون كوك المفاهيم الأساسية لـ RISC في مركز أبحاث توماس جيه واتسون التابع لشركة IBM في سبعينيات القرن الماضي. أدى عمله إلى ابتكار الحاسوب المصغر IBM 801 عام 1971، والذي صُمم في الأصل للاستخدام في أنظمة تحويل المكالمات الهاتفية. تميّز هذا المعالج بعدة ابتكارات رئيسية، بما في ذلك تنسيق تعليمات ثابت والقدرة على تنفيذ التعليمات في دورة ساعة واحدة، وهي سمات مميزة لتصميم RISC.

على عكس تركيبة CISC، التي تعتمد على تعليمات معقدة ومتعددة الوظائف، تتميز تركيبة RISC بمجموعة محدودة ومُحسّنة للغاية من التعليمات. تتطلب البرامج المصممة لمعالجات RISC عادةً عددًا أكبر من التعليمات البسيطة لإنجاز المهام، ولكن يتم تنفيذ هذه التعليمات بكفاءة أعلى، كما تتضمن معالجات RISC عددًا كبيرًا من المسجلات العامة للأغراض الحسابية، مما يقلل الاعتماد على الوصول إلى الذاكرة. علاوة على ذلك، تقتصر عمليات الذاكرة على تعليمات التحميل والتخزين، مما يبسط معالجة البيانات ويعزز فعالية تسلسل التعليمات.[2][5][7][9]

2.7 الخصائص المعمارية

تتميز بنية RISC بعدة خصائص فريدة تُسهم في أدائها وكفاءتها:

• عمليات مباشرة بين المسجلات: تعمل معظم التعليمات مباشرةً على المسجلات، مما يقلل من الوصول إلى الذاكرة ويُحسّن سرعة التنفيذ.

• طول ثابت للتعليمات: جميع التعليمات ذات حجم موحد، مما يُبسط عملية فك التشفير ويُسرّع جلب التعليمات وتنفيذها.

• تنفيذ في دورة واحدة: صُممت معالجات RISC لتنفيذ تعليمة واحدة في كل دورة ساعة، مما يُتيح أداءً عالي السرعة وقابلًا للتنبؤ.

• معالجة متوازية فعّالة: تُتيح بساطة التعليمات وتجانسها، معالجة متوازية مُبسطة، حيث تتم معالجة تعليمات متعددة في وقت واحد على مراحل مختلفة.

• مجموعة مسجلات موسعة: يُتيح العدد الأكبر من المسجلات نقل البيانات بشكل أسرع ويُقلل من الحاجة إلى الوصول المتكرر إلى الذاكرة، مما يُحسّن الإنتاجية الإجمالية.

تجعل هذه الخصائص بنية RISC مناسبة بشكل خاص للتطبيقات التي تتطلب معالجة عالية السرعة، مثل الأنظمة المُدمجة والأجهزة المحمولة وبيئات الحوسبة الآنية.[1][6]

3.7 مثال على التعليمات في بنية RISC

في الأنظمة القائمة على بنية RISC، لا تتوفر التعليمات المعقدة الموجودة في بنية CISC. بدلاً من ذلك، تُقسّم المهام إلى سلسلة من التعليمات البسيطة والموحدة. على سبيل المثال، لإجراء عملية ضرب بين قيمتين مخزنتين في الذاكرة، يحتاج معالج RISC إلى عدة تعليمات لإتمام المهمة. تتضمن هذه العملية تحميل المعاملات في المسجلات، وتنفيذ عملية الضرب، وتخزين النتيجة في مسجل مُخصص.

يوضح تسلسل التعليمات التالي الموضح في الجدول 1 هذا النهج:

الجدول 1: مثال على تسلسل التعليمات في معمارية RISC

الخطوة	التعليمة البرمجية	الشرح
1	**LOAD Reg A, 0x100**	تحميل القيمة من عنوان الذاكرة 100x0 إلى المسجل A.
2	**LOAD Reg B, 0x101**	تحميل القيمة من عنوان الذاكرة 101x0 إلى المسجل B.
3	**MUL Reg A, Reg B**	ضرب القيم في المسجلين A وB، وتخزين النتيجة في A.
4	**MOV Reg C, Reg A**	نقل النتيجة من المسجل A إلى المسجل C.

في هذا المثال، تقوم التعليمات بتنفيذ عملية الضرب وتخزين النتيجة في المسجل A. ثم تقوم التعليمات بنقل النتيجة إلى المسجل C، يعكس هذا التنفيذ خطوة بخطوة فلسفة RISC المتمثلة في البساطة والكفاءة من خلال تقليل تعقيد التعليمات.

4.7 مميزات وعيوب بنية RISC

توفر بنية حاسوب مجموعة التعليمات المختصرة (RISC) مجموعة من المزايا والتنازلات التي تميزها عن نظيرتها CISC. نلخص هذه الخصائص فيما يلي:

- **المزايا:**

• كفاءة وسرعة عاليتان: تسمح التعليمات البسيطة بتنفيذ أسرع ومعالجة أكثر سلاسة.

• سهولة التصميم وفعالية التكلفة: تقلل البنية المبسطة من تعقيد التصميم وتكاليف التصنيع.

• دورات تطوير أقصر: يمكن للمهندسين تصميم معالجات RISC وتنفيذها بسرعة أكبر.

• استهلاك منخفض للطاقة: يساهم عدد أقل من الترانزستورات ومنطق تحكم أبسط في كفاءة استهلاك الطاقة.

• طول تعليمات موحد: تبسط التعليمات ذات الطول الثابت عملية فك التشفير وتدعم التنفيذ في دورة واحدة.

• توازي مُحسَّن: يُمكّن التنفيذ في دورة واحدة من استخدام خطوط الأنابيب بكفاءة ومعالجة المهام المتزامنة.

• سرعات ساعة أعلى: يسمح انخفاض التعقيد بسرعات ساعة أعلى.

• مجموعة مسجلات موسعة: يقلل العدد الكبير من المسجلات من الوصول إلى الذاكرة ويحسن الأداء.

• يقتصر الوصول إلى الذاكرة على عمليتي التحميل والتخزين: يُبسط هذا التقييد إدارة الذاكرة ويعزز كفاءة خط الأنابيب.

• عدد أقل من الترانزستورات المطلوبة: يقلل تصميم الأجهزة الأبسط من عدد الترانزستورات اللازمة.

- **العيوب:**

• زيادة حجم الكود: تتطلب العمليات المعقدة تعليمات متعددة، مما قد يؤدي إلى زيادة حجم البرنامج.

• محدودية وظائف التعليمات: قد يتطلب غياب التعليمات المعقدة جهدًا أكبر من المترجمات والمبرمجين لتحقيق نفس الوظائف.[1][2][10]

الجدول 2: مقارنة بين معمارية CISC ومعمارية RISC

الميزة	معمارية CISC	معمارية RISC
التصميم	مركّز على المكونات المادية	مُركّز على البرمجيات
تعقيد التعليمات	تعليمات معقدة متعددة الساعات	تعليمات مُبسّطة تعمل بنبضة ساعة واحدة
نموذج الوصول إلى الذاكرة	عمليات نقل البيانات بين الذاكرة مُدمجة في التعليمات	عمليات بين السجلات؛ الوصول إلى الذاكرة عبر التحميل/التخزين فقط
حجم الكود ودورات التنفيذ	حجم كود أصغر؛ دورات معالجة أعلى لكل تعليمة	حجم كود أكبر؛ دورات أقل لكل تعليمة
تخصيص الترانزستورات	استخدام عدد أكبر من الترانزستورات لفك تشفير التعليمات ومنطق التحكم	عدد أكبر من الترانزستورات مُخصّص لتخزين السجلات
تنسيق التعليمات	تعليمات متغيرة الطول	تعليمات ثابتة الطول

كما هو موضح في الجدول 2، فإن الفرق بين البنيتين واضح: تتعامل CISC مع التعليمات المعقدة من خلال آليات التحكم المشفرة بدقة، بينما تعالج RISC التعليمات البسيطة باستخدام نهج مبسط وثابت التنسيق.

8. أمثلة على معمارية RISC وCISC

لتوضيح التطبيق العملي لهذين النموذجين المعماريين، نورد فيما يلي أمثلة بارزة للمعالجات وعائلات مجموعات التعليمات التي تجسد مبادئ تصميم CISC وRISC، كما هو موضح في الشكل 3.

1.8 أمثلة على معمارية CISC

تتميز المعالجات القائمة على معمارية CISC بمجموعات تعليماتها المعقدة وتصميمها الذي يركز على المكونات المادية. ومن الأمثلة البارزة عليها:

• PDP-11: طُوِّر بواسطة شركة Digital Equipment Corporation (DEC)، ويُعرف بمجموعة تعليماته المؤثرة.

• VAX: معمارية أخرى من DEC، وتتميز بمجموعة تعليماتها الغنية والمتكاملة.

• Motorola 68k: استُخدم على نطاق واسع في أجهزة الكمبيوتر الشخصية المبكرة والأنظمة المدمجة.

• Intel x86: إحدى أكثر معمارية CISC انتشارًا، وتشكل أساس معظم وحدات المعالجة المركزية لأجهزة الكمبيوتر المكتبية والمحمولة.

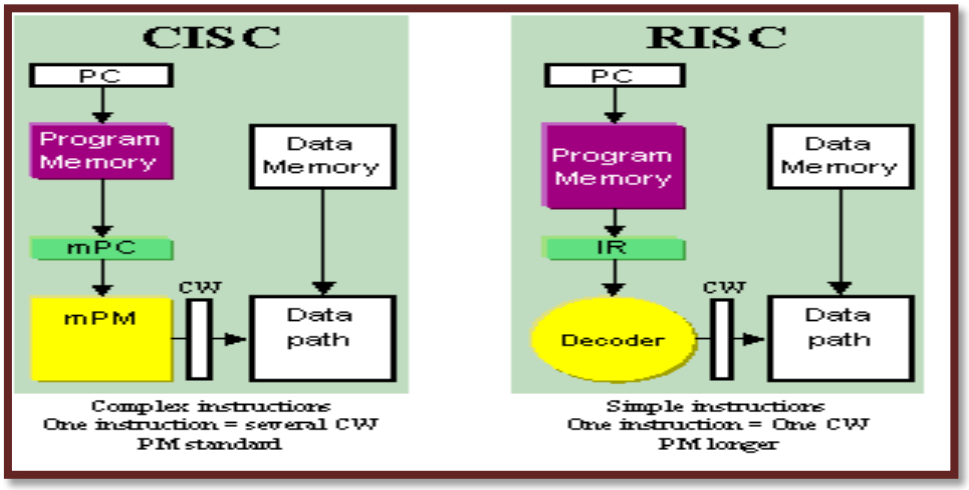
تركز هذه المعماريات على تقليل عدد التعليمات لكل برنامج من خلال دمج العمليات متعددة الخطوات في تعليمات واحدة.

2.8 أمثلة على بنية RISC

تُعطي بنية RISC الأولوية للبساطة والسرعة وكفاءة معالجة البيانات المتسلسلة. ومن أبرز عائلاتها:

- DEC Alpha، AMD 29k، ARC، Atmel AVR، Blackfin
- Intel i860 و i960
- MIPS، Motorola 88000، PA-RISC
- بنية Power (بما في ذلك PowerPC)
- SuperH، SPARC، و ARM

تُستخدم هذه المعالجات على نطاق واسع في الأنظمة المدمجة والأجهزة المحمولة والحوسبة عالية الأداء نظرًا لمجموعات التعليمات المُبسطة والتصاميم الموفرة للطاقة. [1][2][10]



شكل 3 : معالجات RISC و CISC

9. معايير تصميم بنية الحاسوب

عند تصميم بنى الحاسوب، غالبًا ما يتناول المهندسون ومصممو الأنظمة هذه العملية من زوايا نظر مختلفة تبعًا لمتطلبات التطبيق والقيود التقنية ومتطلبات السوق. ومع ذلك، هناك معياران أساسيان يوجهان قرارات التصميم المعماري باستمرار:

- زيادة سرعة التشغيل و تقليل وقت التنفيذ :

الهدف الرئيسي هو تحسين الأداء عن طريق تقليل الوقت اللازم لتنفيذ التعليمات، ويشمل ذلك تحسين مسارات التعليمات، والوصول إلى الذاكرة، وإنتاجية المعالج لتحقيق سرعة حسابية أكبر.

• تقليل تكلفة التطوير وسعر البيع:

في الأنظمة التجارية والمدمجة، تُعد الكفاءة الاقتصادية أمرًا بالغ الأهمية، ويسعى المصممون إلى تقليل تعقيد الأجهزة، وتبسيط عمليات التصنيع، وتقصير دورات التطوير لإنتاج أنظمة بأسعار معقولة دون المساس بالوظائف الأساسية.

10. المحاكاة

تساعد المحاكاة في تحليل أداء المعالجات وسلوكها، في هذه البحث، أُجريت المحاكاة باستخدام لغة الماتلاب MATLAB للأسباب التالية:

• يوفر أدوات فعّالة لتحليل أداء التعليمات، مثل أداة profile.

• يسمح بمحاكاة مرنة لمجموعات تعليمات RISC وCISC.

• يدعم التصور البياني والمقارنات الزمنية بسهولة.

• مناسب تمامًا للباحثين والطلاب في مجال هندسة الحاسوب وهندسة المعالجات.

1.10 بيئة المحاكاة

صُممت بيئة المحاكاة لمقارنة أداء معماريتي RISC وCISC من خلال الخطوات التالية:

• محاكاة تعليمة معقدة واحدة في معمارية CISC.

• محاكاة سلسلة من التعليمات البسيطة في معمارية RISC لأداء المهمة نفسها.

• قياس مؤشرات الأداء الرئيسية مثل عدد التعليمات، ووقت التنفيذ، وحجم الكود.

• استخدام أدوات تحليل الأداء المدمجة في MATLAB (مثل profile) لتحليل الأداء بدقة عالية.

• عرض النتائج في نافذة الأوامر، مع إمكانية تصور المقارنات بيانيًا باستخدام الرسوم البيانية.

2.10 تهيئة الذاكرة

تم إنشاء نموذج ذاكرة مبسط باستخدام بنية containers.Map في MATLAB، والتي تسمح بتخزين أزواج المفاتيح والقيم. في هذه المحاكاة، تعمل عناوين الذاكرة (مثل x1000 وx1010) كمفاتيح، وتمثل القيم العددية (مثل 5 و7) البيانات المخزنة في تلك العناوين، يحاكي هذا الإعداد كيفية تحميل البيانات من الذاكرة إلى المسجلات أثناء تنفيذ التعليمات.

فعند تخزين الرقمين في عنواني الذاكرة يكون الأمر كما يلي :

```
memory = containers.Map({'0x100', '0x101'}, {5, 7});
```

3.10 تطبيق نموذج CISC

• تم تنفيذ تعليمة معقدة واحدة (مثل MULT)، تشمل عمليات تحميل الذاكرة ومعالجتها وتخزينها في خطوة واحدة.

• قياس زمن التنفيذ باستخدام دالتي tic و toc في MATLAB.

• افتراض أن التعليمة المعقدة تتطلب أربع دورات ساعة لإكمالها.

• تم حساب حجم الكود بناءً على عدد التعليمات، بافتراض أن كل تعليمة تحتوي على 4 بايت.

• تم حساب كفاءة التنفيذ كنسبة بين ناتج الإخراج وحاصل ضرب زمن التنفيذ في عدد دورات الساعة.

4.10 تطبيق نموذج RISC

• حتى تتم عملية المقارنة تم تنفيذ المهمة نفسها باستخدام سلسلة من التعليمات البسيطة، وهي:

LOAD: لاسترجاع البيانات من الذاكرة إلى المسجلات.

MUL: لإجراء عملية الضرب.

MOV: لنقل النتيجة إلى مسجل الوجهة.

• تم قياس زمن التنفيذ وعدد التعليمات وعدد دورات الساعة، بافتراض أن كل تعليمة تتطلب دورة ساعة واحدة.

• تم حساب حجم الكود وكفاءة التنفيذ باستخدام نفس الطريقة المطبقة في CISC.

5.10 عرض و تحليل النتائج

عُرضت النتائج في نافذة أوامر MATLAB باستخدام الدالة fprintf. تضمنت المخرجات مؤشرات الأداء التالية لكل بنية:

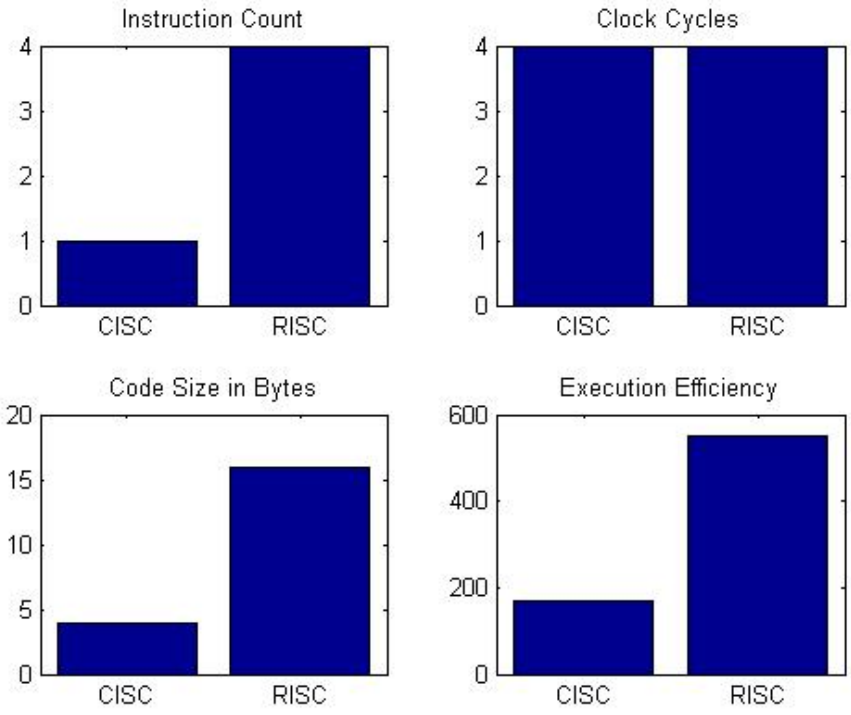
1. عدد التعليمات لكل من RISC وCISC.
2. دورات الساعة اللازمة للتنفيذ.
3. حجم الكود بالبايت.
4. زمن التنفيذ بالثواني.
5. كفاءة التنفيذ، محسوبة كنسبة بين نتيجة المخرجات وحاصل ضرب زمن التنفيذ في دورات الساعة.

بعد إجراء المحاكاة، تحصلنا على النتائج التالية والموضحة في الجدول 3:

الجدول 3: التحليل المقارن لـ RISC مقابل CISC

التحليل	RISC	CISC	المقياس
تستخدم معمارية CISC تعليمة واحدة معقدة، بينما تستخدم معمارية RISC عدة تعليمات بسيطة.	4	1	عدد التعليمات
تتطلب كلتا المعماريتين نفس عدد الدورات في هذا السيناريو.	4	4	دورات الساعة
تتميز معمارية RISC بحجم كود أكبر نظرًا لكثرة التعليمات المستخدمة.	16	4	حجم الكود (بايت)
تُنفذ معمارية RISC بسرعة أكبر بكثير	0.002163	0.029844	زمن التنفيذ (ثانية)
تُظهر معمارية RISC كفاءة أعلى بكثير في هذه الحالة.	4045.4945	293.1893	كفاءة التنفيذ

6. توليد الرسوم البيانية :



شكل 4 : نتائج محاكاة RISC و CISC

من خلال شكل 4 والذي يظهر فيه المقارنة بين RISC و CISC من حيث حجم الكود والكفاءة وعدد التعليمات وعدد دورات الساعة ، تظهر نتائج المحاكاة بأن على الرغم من استخدام معالجات CISC عددًا أقل من التعليمات، إلا أن وقت تنفيذها كان أطول، مما يؤدي إلى انخفاض في كفاءتها.

أما معالجات RISC، واستخدامها عددًا أكبر من التعليمات، إلا أنها حققت سرعة تنفيذ أعلى وكفاءة أكبر بكثير من معالجات CISC.

يُعدّ حجم الكود الأكبر في معالجات RISC عيبًا معروفًا، ولكنه قد يكون مقبولًا في الأنظمة ذات الذاكرة الكافية. معالجات CISC تستخدم عددًا أقل من التعليمات، إلا أن وقت تنفيذها أطول، مما يؤدي إلى انخفاض كفاءتها. وفي الوقت نفسه فإن معالجات RISC، استخدمت عددًا أكبر من التعليمات، إلا أنها حققت سرعة تنفيذ أعلى وكفاءة أكبر بكثير.

6.10 مناقشة النتائج

أظهرت نتائج المحاكاة تفوق بنية RISC على بنية CISC من حيث زمن التنفيذ والكفاءة الإجمالية، على الرغم من حاجتها إلى عدد أكبر من التعليمات وحجم أكبر للبرنامج.

في المقابل، استخدمت CISC تعليمة واحدة معقدة، مما قلل حجم البرنامج ولكنه أدى إلى زيادة زمن التنفيذ، ما أثر سلبًا على الكفاءة.

تشير النتائج إلى أن RISC حققت كفاءة تنفيذ أعلى بأكثر من 13 مرة من CISC في هذا السيناريو، مما يُبرز ميزة التعليمات البسيطة والسريعة في تقديم أداء فائق، لا سيما في المهام الحسابية المباشرة. لا يعني انخفاض عدد التعليمات بالضرورة أداءً أفضل، إذ قد تتطلب كل تعليمة في CISC وقتًا أطول للتنفيذ.

وكذلك قد تستخدم RISC عددًا أكبر من التعليمات، لكنها تُنفذ بشكل أسرع نظرًا لبساطتها ومدة دورتها المنتظمة.

في CISC، يمكن أن تكون التعليمة الواحدة معقدة وتستغرق عدة دورات، على الرغم من انخفاض عدد التعليمات، ولكن في RISC، تُنفذ كل تعليمة في دورة ساعة واحدة، مما ينتج عنه أداء أكثر قابلية للتنبؤ وأكثر اتساقًا.

دورات الساعة المتساوية في هذه المحاكاة لا تشترط وجود أداءً متساوياً، إذ يختلف زمن التنفيذ الفعلي (بالثواني).

ومن ناحية حجم الكود يتميز معالج CISC بحجم كود أصغر، مما يجعله مناسباً للبيئات التي تتطلب تقليل استهلاك الذاكرة إلى أدنى حد، بينما ينتج معالج RISC حجم كود أكبر نسبياً نظراً لتعدد التعليمات، ولكنه

يعوض ذلك بسرعة تنفيذ أعلى وكفاءة أكبر. (في بعض التطبيقات قد يكون حجم الكود عاملاً حاسماً في اختيار بنية المعالج).

كفاءة التنفيذ هي مقياس يُستخدم لتقييم مدى فعالية المعالج في أداء مهمة معينة، وذلك بمقارنة النتيجة المحققة بالوقت والموارد الحاسوبية المستهلكة. في هذا السيناريو الذي تم، تُحسب الكفاءة باستخدام الصيغة التالية: كفاءة التنفيذ = (النتيجة / زمن التنفيذ × دورات الساعة).

على الرغم من أن كلا البنيتين استخدمتا نفس عدد دورات الساعة في هذا السيناريو، إلا أن معمارية RISC حققت زمن تنفيذ أقل بكثير، مما أدى إلى كفاءة أعلى بأكثر من 13 ضعفًا مقارنةً بمعمارية CISC. تعكس الكفاءة العالية في معمارية RISC بساطة وسرعة تعليماتها، في المقابل، على الرغم من أن معمارية CISC تستخدم تعليمة واحدة، إلا أن تعقيدها يؤدي إلى زمن تنفيذ أطول وكفاءة أقل.

يُعد هذا المقياس ذا قيمة خاصة عند مقارنة البنيات في التطبيقات التي تتطلب أداءً عاليًا وزمن استجابة منخفضًا.

11. قابلية التوسع

يتميز سيناريو المحاكاة المقترح بقابلية توسع عالية، ويمكن توسيعه ليشمل جوانب أوسع من تحليل بنية الحاسوب، تشمل التحسينات المحتملة ما يلي:

• إضافة تعليمات جديدة مثل الجمع والطرح والقفز لتقييم سلوك البنية في ظل ظروف أكثر تعقيدًا.

• تطوير واجهة رسومية تفاعلية باستخدام مصمم التطبيقات في MATLAB، مما يسمح للمستخدمين باختيار البنى وعرض النتائج بشكل ديناميكي.

• دمج مقاييس إضافية مثل استهلاك الطاقة.

• دمج سيناريوهات متعددة في تقرير شامل يقارن الأداء عبر مهام متنوعة، مما يعزز القيمة التعليمية والبحثية للنموذج.

12. الخلاصة

أظهرت هذه المحاكاة تحليلًا مقارنًا بين معماريتي RISC وCISC باستخدام MATLAB، مع التركيز على مؤشرات الأداء الرئيسية: عدد التعليمات، وعدد دورات الساعة، وحجم الكود، ووقت التنفيذ، وكفاءة التنفيذ.

حيث : حققت معمارية RISC كفاءة تنفيذ أعلى بكثير ووقت تنفيذ أسرع، على الرغم من حاجتها إلى عدد أكبر من التعليمات وحجم كود أكبر، وعلى الرغم من استخدام معمارية CISC عددًا أقل من التعليمات

وإنتاجها حجم كود أصغر، إلا أنها استغرقت وقتًا أطول للتنفيذ نظرًا لتعقيد تعليماتها.

•أيضا كانت كفاءة تنفيذ RISC أعلى بأكثر من 13 مرة من كفاءة تنفيذ CISC في السيناريو المختبر، مما يُبرز ميزة التعليمات البسيطة والموحدة في التطبيقات الحساسة للأداء.

تُؤكد هذه النتائج على أهمية تقييم كل من المقاييس الكمية (مثل عدد التعليمات، وعدد الدورات، والحجم) والعوامل النوعية (مثل تعقيد التعليمات، وإمكانية التنبؤ) عند اختيار أو تصميم معمارية المعالجات.

توفر هذه المحاكاة إطارًا أساسيًا لمزيد من الاستكشاف، بما في ذلك دمج أنواع إضافية من التعليمات، وتحديد خصائص الطاقة والذاكرة، وتطوير أدوات تعليمية تفاعلية.

من التوصيات التي نود تقديمها في هذا المجال هي محاولة تحليل أداء هاتين البنيتين تحت ظروف مختلفة مثل عدد تعليمات أكثر (حجم برنامج كبير) وتحديد المعالج الأنسب للتطبيق. كذلك إمكانية استخدام أدوات أخرى للمحاكاة للحصول على نتائج أفضل ودراسة مؤشرات الأداء بين هذه المعالجات وتحديد أيهم الأفضل والأنسب لكل بيئة عمل.

المراجع

[1]. Vivek Dubey M. Tech Schohlar Shri Ram Institute of Technology, Jabalpur (M.P.), [INDIA],Prof. Ravi Mohan Head of the Department & Professor Department of Electronics & Communication Engg. Jabalpur (M.P.), [INDIA] (March 2014) 'A New Architecture of RISC cum CISC Processor Architecture', International Journal of Modern Engineering & Management Research, 2 (1), pp. 42-49.

[2]. Yi Gao, Shilang Tang, Zhongli Ding (n.d.) 'Comparison between CISC and RISC', , (), pp. .

[3]. Usama Arshad, Dr. Ghulam Abbas Superscalar Processors Registration Number: CS2201 May 22, (2022), Ghulam Ishaq Khan Institute. Registration Number: CS2201

[4]. Saiprathyusha, P., & Chandrasekhar, C. (2025). Implementation of RISC-V Processor. ITMWebConferences, 74,02006. https://doi.org/10.1051/itmconf/20257402006

[5]. Pamela Smallwood, Mohamed Lotfy, and Mark Sanders Computer Science Department School of Computer and Information Sciences Regis University Denver, CO 80221 (2013) 'INTRODUCING THE CPU AND ASSEMBLY LANGUAGE CONCEPTS VIA A MODEL INSTRUCTION SET ARCHITECTURE', pp. 35-43.

[6]. ianming Liu1, Yunjie Zhang2 1:Computer Department Weifang Medical University Weifang, China 2:Electronic & Information Engineering Department Tianjin Institute of Urban Construction Tianjin, China, Lili Xu3, Pengtao Liu1 3:Weifang Vocational College Weifang, China (2013) 'Design of Simple CPU Based on Hardware Description Language', Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013), (), pp.

[7]. William Stallings (Copyright © 2013, 2010, 2006) COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE, NINTH EDITION edn., Manufactured in the United States of America.

[8]. Jain, N. (2022). RTL Design of CISC CPU IP Core. International Journal for Research in Applied Science and Engineering Technology, 10(6), pp.4620–4624. https://doi.org/10.22214/ijraset.2022.45067

[9]. Danijela Jakimovska, Aristotel Tentov, Goran Jakimovski, Sashka Gjorgjievska and Maja Malenko. (2023). EPIC Architecture Data Flow Processors and their Execution [Review of EPIC Architecture Data Flow Processors and their Execution]. Journal of Information Organization, Volume13 Number4 December2023. https://doi.org/10.6025/jio/2023/13/4/112-117

[10]. Chevtchenko, S. F.1; Vale, R. F.2 Department of Statistics and Informatics UFRPE Recife, Brazil (2013) 'A Comparison of RISC and CISC Architectures'