**Introduction à l’architecture RISC-V**

Partie 1 – Étude théorique de l’architecture RISC-V

1. Présentation de l’architecture RISC-V

L’architecture RISC-V est une architecture de jeu d’instructions (ISA) développée à l’Université de Berkeley en 2010. Son objectif est de fournir une architecture ouverte, simple, modulaire et extensible qui puisse être utilisée pour des applications allant des microcontrôleurs aux supercalculateurs. RISC-V vise à démocratiser l’accès à l’architecture matérielle en offrant une alternative libre aux architectures propriétaires telles que x86 ou ARM. Sa flexibilité est d'ailleurs illustrée par sa présence dans des domaines variés, des microcontrôleurs comme ceux d'*Espressif* (*ESP32C3*) ou les *Nuclei* (*Hummingbird E203*), aux microprocesseurs intégrés tels que l'*Allwinner D1* ou le *SiFive U54*, en passant par des accélérateurs, des émulateurs et des simulateurs

Les caractéristiques principales de RISC-V sont :

* Une architecture RISC (*Reduced Instruction Set Computing*) simple avec un nombre limité d’instructions.
* Un format d’instructions fixe de 32 bits, avec des extensions pour 16, 64, 128 bits.
* Un jeu d’instructions de base minimaliste, extensible par des instructions optionnelles.
* Une conception modulaire facilitant l’implémentation pour différentes tailles et performances de processeurs.

1. Formats des instructions RISC-V

Afin d'optimiser le décodage et l'utilisation des ressources du processeur, les instructions RISC-V sont encodées selon plusieurs formats binaires distincts, chacun adapté à un type d'opération spécifique:

* Format R : instructions avec trois registres (ex : opérations arithmétiques).
* Format I : instructions avec un registre source, un registre destination et une valeur immédiate (ex : chargement, opérations immédiates).
* Format S : instructions de stockage mémoire, avec deux registres et une valeur immédiate pour l’adresse.
* Format B : instructions conditionnelles de branchement.
* Format U : instructions avec un grand immédiat utilisé pour le calcul d’adresses.
* Format J : instructions de saut avec adresse relative.

Ces différents formats s'inscrivent dans une architecture profondément modulaire, où le jeu d'instructions de base est complété par des extensions optionnelles. Cette modularité permet de répondre à une vaste gamme de besoins de performance et d'application, tout en gérant différentes longueurs d'instructions pour optimiser la taille du code.

Une image contenant texte, capture d’écran, Caractère coloré, conception

Le contenu généré par l’IA peut être incorrect.

Figure : Aperçu des extensions et formats d'instructions de RISC-V

1. Comparaison avec d’autres architectures

RISC-V se positionne comme une alternative révolutionnaire face aux architectures matérielles établies, se distinguant radicalement de l'approche complexe de *x86* et de la nature propriétaire d'*ARM*:

* ***x86* : L'héritage de la complexité (*CISC*)**

L'architecture *x86* est synonyme de *Complex Instruction Set Computing* (*CISC*), caractérisée par un nombre pléthorique d'instructions aux longueurs variables. Cette complexité est exacerbée par des décennies de rétrocompatibilité maintenue à tout prix, ce qui alourdit considérablement la conception des processeurs, rend leur vérification ardue et peut limiter leur efficacité énergétique.

* ***ARM* : La puissance mobile sous licence (*RISC* propriétaire)**

Quant à ARM, bien qu'étant une architecture *RISC* reconnue pour son efficacité énergétique et sa prédominance dans les appareils mobiles et embarqués, elle reste une architecture propriétaire. Cela signifie que son utilisation et sa personnalisation requièrent l'acquisition de licences onéreuses et restrictives, limitant la liberté d'innovation et l'accès à la conception de base pour la recherche ou les petites entreprises.

* ***RISC-*V : La liberté du standard ouvert et modulaire**

En contraste frappant, *RISC-V* incarne un standard ouvert et libre de droits. Sa simplicité et sa modularité intrinsèque facilitent grandement la recherche académique, l'éducation et le développement de processeurs hautement personnalisés (*ASIC*) sans les contraintes de coût ou de licence. Cette approche décentralisée favorise une innovation rapide, une transparence accrue et permet à quiconque de concevoir, vérifier et produire son propre processeur *RISC-V*, ouvrant ainsi la voie à une véritable démocratisation de l'architecture matérielle.

1. Cycle de vie d’une instruction RISC-V

L'exécution d'une instruction dans un processeur *RISC-V* typique s'effectue séquentiellement à travers un pipeline de plusieurs étapes distinctes, optimisant le débit d'instructions et l'utilisation des ressources matérielles :

1. **Fetch (Récupération de l'instruction) :** À cette étape, l'instruction est lue depuis la mémoire (généralement le cache d'instructions) à l'adresse spécifiée par le Compteur Ordinal (Program Counter - PC). Le PC est ensuite mis à jour pour pointer vers l'instruction suivante.
2. **Decode (Décodage de l'instruction) :** L'instruction récupérée est interprétée. L'unité de décodage identifie l'opération à effectuer (opcode), extrait les identifiants des registres sources et détermine si des valeurs immédiates sont présentes, préparant ainsi les opérandes pour l'étape suivante.
3. **Execute (Exécution) :** L'opération principale de l'instruction est réalisée. Cela peut impliquer une opération arithmétique ou logique effectuée par l'Unité Arithmétique et Logique (ALU), le calcul d'une adresse effective pour un accès mémoire, ou la détermination de la cible d'un saut ou d'une branche.
4. **Memory Access (Accès mémoire) :** Si l'instruction nécessite une interaction avec la mémoire de données (par exemple, une instruction de chargement : load ou de stockage : store), cette étape gère la lecture ou l'écriture des données vers ou depuis la mémoire principale ou le cache de données.
5. **Write Back (Écriture du résultat) :** Le résultat de l'opération (provenant de l'ALU ou de la mémoire de données) est écrit dans le registre de destination spécifié par l'instruction.
6. **Update PC (Mise à jour du Compteur Ordinal) :** Le Compteur Ordinal (PC) est mis à jour pour pointer vers l'adresse de l'instruction suivante à exécuter. Pour les instructions de branchement ou de saut, le PC est mis à jour avec l'adresse cible calculée, tandis que pour les instructions séquentielles, il est simplement incrémenté."

Figure : Représentation schématique du pipeline d'exécution RISC-V

Cette organisation en pipeline permet au processeur de traiter plusieurs instructions simultanément, chacune à une étape différente de son exécution, augmentant ainsi considérablement le débit global d'instructions.

Partie 2 – Pratique : Premiers pas en assembleur RISC-V

Pour cette exploration pratique de l'architecture RISC-V, le programme choisi est le calcul de la factorielle d'un entier. Cette implémentation en assembleur s'effectuera dans l'environnement de simulation RARS, offrant l'opportunité de manipuler concrètement les instructions de base, la gestion des boucles et les interactions utilisateur en assembleur RISC-V.

Le programme se compose d'une section de données pour les messages textuels et d'une section de texte pour le code exécutable. Après avoir affiché une invite, il lit un entier de l'utilisateur et le stocke dans le registre t0. Le calcul de la factorielle est effectué dans une boucle (boucle) qui multiplie de manière itérative le résultat courant (t1) par un compteur (t2) jusqu'à atteindre la valeur saisie (t0). Enfin, le programme affiche un message de résultat suivi de la factorielle calculée, puis se termine proprement via un appel système.

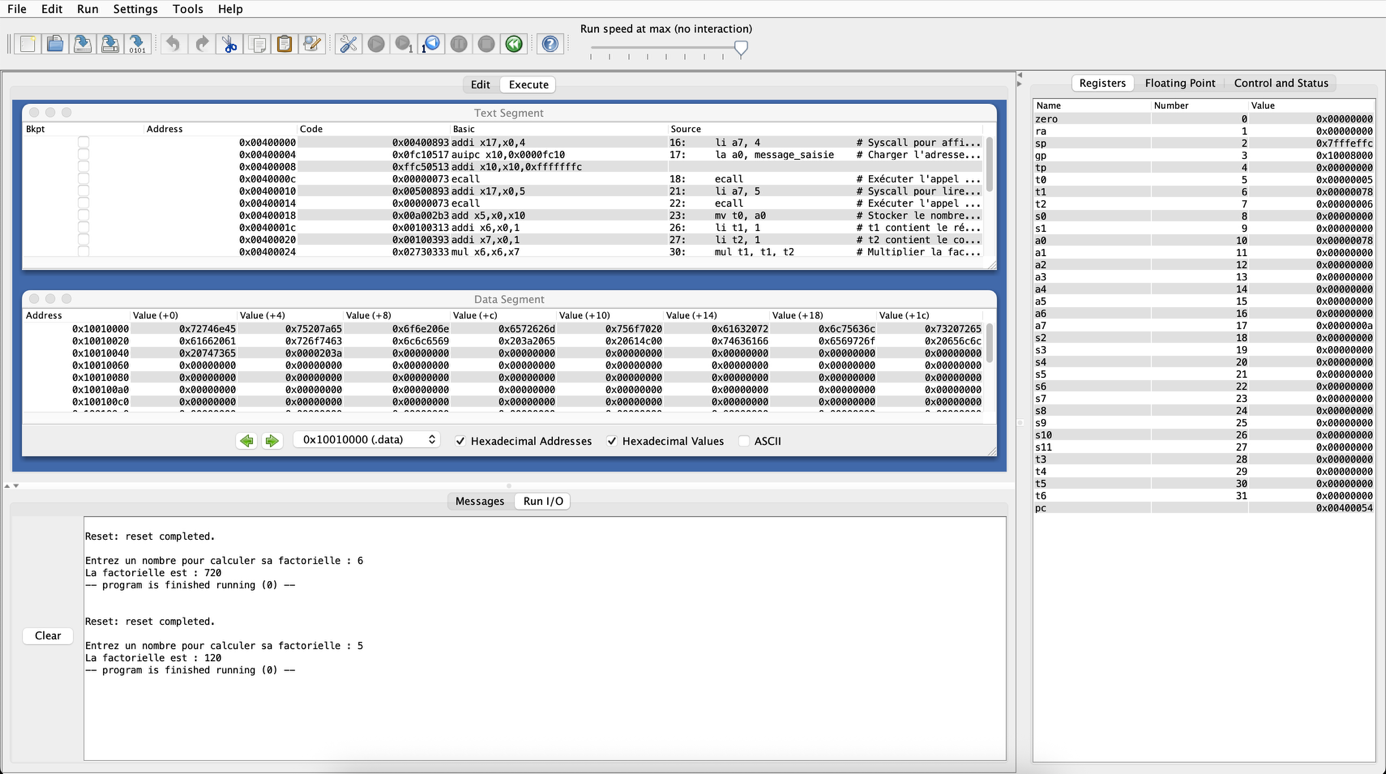


Figure : Exemple d'exécution du programme de calcul de factorielle dans l'environnement RARS

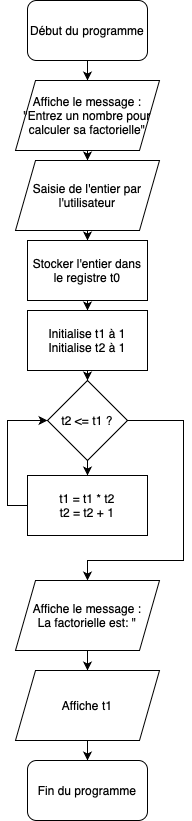


Figure 4: Représentation du fonctionnement du programme de calcul de factorielle en assembleur RISC-V

En conclusion de cette exploration, ce projet a permis d'acquérir une compréhension approfondie des fondements de l'architecture *RISC-V*. Il a notamment mis en lumière les atouts de son jeu d'instructions (*ISA*) épuré et modulaire, ainsi que la logique de ses formats d'instructions variés qui optimisent le décodage et l'utilisation des ressources du processeur. La mise en pratique par la programmation en assembleur a été essentielle pour saisir la gestion précise des registres et visualiser le déroulement séquentiel des instructions au sein d'un pipeline processeur. Grâce à sa simplicité inhérente et son modèle ouvert et libre de droits, *RISC-V* se positionne non seulement comme une architecture d'avenir prometteuse, mais aussi comme un catalyseur d'innovation de plus en plus adoptée et valorisée dans les domaines de la recherche académique et du développement industriel.