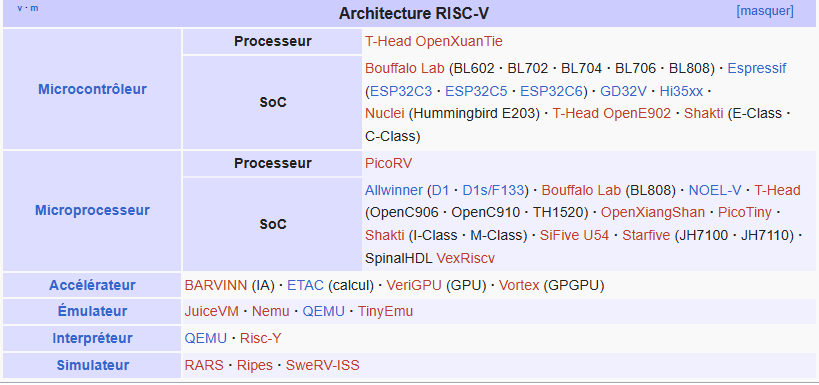
**Introduction à l’architecture RISC-V**

# 1. Présentation de l’architecture RISC-V

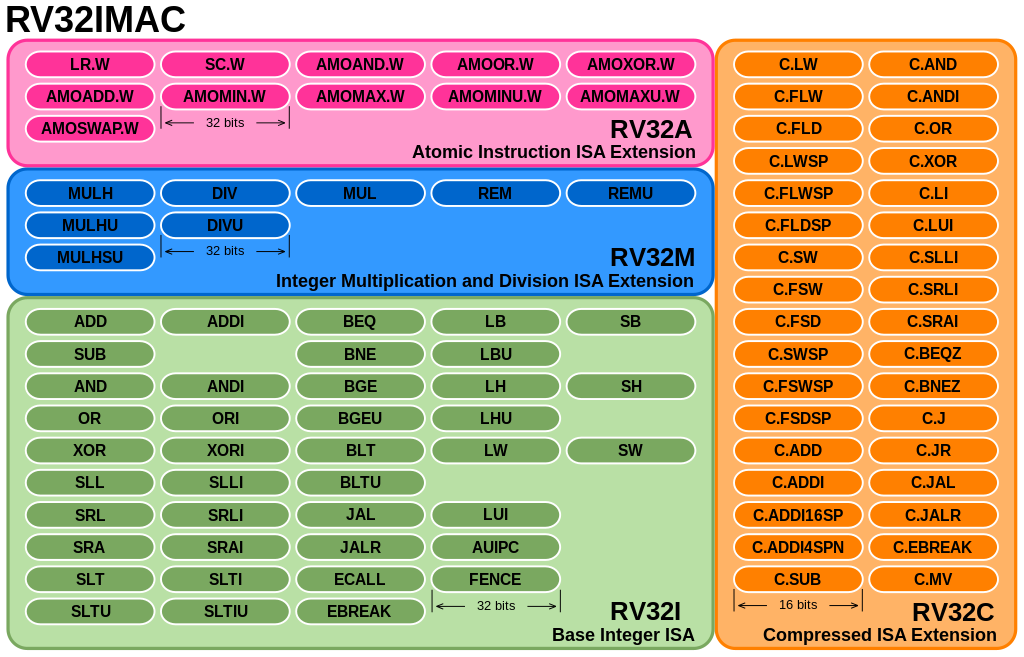
L’architecture RISC-V est une architecture de jeu d’instructions (ISA) développée à l’Université de Berkeley en 2010. Son objectif est de fournir une architecture ouverte, simple, modulaire et extensible qui puisse être utilisée pour des applications allant des microcontrôleurs aux supercalculateurs. RISC-V vise à démocratiser l’accès à l’architecture matérielle en offrant une alternative libre aux architectures propriétaires telles que x86 ou ARM.

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.



# 2. Formats des instructions RISC-V

Les instructions RISC-V sont codées sur différents formats selon leur type :  
- 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.



Jeux d’instructions

# 3. Comparaison avec d’autres architectures

RISC-V se distingue des architectures classiques comme x86 (CISC) ou ARM (RISC propriétaire) par son ouverture et sa simplicité :  
- x86 est une architecture complexe (CISC) avec un grand nombre d’instructions variées et une rétrocompatibilité lourde.  
- ARM est une architecture RISC très utilisée dans les appareils mobiles, mais propriétaire.  
- RISC-V offre un standard ouvert et modulaire, facilitant la recherche, l’éducation et le développement de processeurs personnalisés.

# 4. Cycle de vie d’une instruction RISC-V

Une instruction RISC-V suit plusieurs étapes dans un processeur typique :  
1. Fetch (Récupération) : récupération de l’instruction en mémoire.  
2. Decode (Décodage) : interprétation de l’instruction et préparation des opérandes.  
3. Execute (Exécution) : exécution de l’opération dans l’ALU ou autre unité.  
4. Memory Access (Accès mémoire) : lecture ou écriture en mémoire si nécessaire.  
5. Write Back (Écriture retour) : stockage du résultat dans un registre.  
6. Update PC (Mise à jour compteur) : mise à jour de l’adresse de l’instruction suivante.

Memory

Acces

Execute

(Exéc. )

Decode

Update

(Décod .)

pc

Fetch

Write

(Récup .)

back

+---------+ +---------+ +---------+ +---------+ +---------+ +---------+  
| Fetch | ----> | Decode | ----> | Execute | ----> | Memory | ----> | Write | ----> | Update |  
| (Récup.)| | (Décod.)| | (Exéc.) | | Access | | Back | | PC |  
+---------+ +---------+ +---------+ +---------+ +---------+ +---------+  
  
1. Récupération de l’instruction  
2. Décodage de l’instruction  
3. Exécution de l’opération  
4. Accès mémoire si nécessaire  
5. Écriture du résultat  
6. Mise à jour du compteur de programme

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

Cette partie présente un exemple de programme en assembleur RISC-V pour calculer la factorielle d’un entier (ici 5). Le programme utilise une boucle pour multiplier successivement les nombres de 1 à n.

## Programme assembleur : calcul de la factorielle de 5

# Calcul de la factorielle de 5

.data

message\_saisie: .asciiz "Entrez un nombre pour calculer sa factorielle : "

message\_resultat: .asciiz "La factorielle est : "

.text

.globl main

main:

# Afficher le message de saisie

li a7, 4 # Code pour afficher une chaîne de caractères

la a0, message\_saisie # Charger l'adresse du message dans a0

ecall

# Lire l'entier saisi par l'utilisateur

li a7, 5 # Code pour lire un entier

ecall

mv t0, a0 # Stocker le nombre saisi dans t0 (n)

# Initialisation

li t1, 1 # t1 = résultat (factorielle)

li t2, 1 # t2 = compteur i = 1

boucle:

mul t1, t1, t2 # résultat = résultat \* i

addi t2, t2, 1 # i = i + 1

ble t2, t0, boucle # si i <= n, continuer la boucle

# Afficher le message de résultat

li a7, 4

la a0, message\_resultat

ecall

# Afficher le résultat (t1)

mv a0, t1

li a7, 1

ecall

# Terminer le programme

li a7, 10

ecall

# résultat final   
 li a7, 4 + la a0, message\_saisie + ecall  
→ Affiche le message demandant à l'utilisateur d’entrer un entier.

li a7, 5 + ecall  
→ Lit un entier saisi au clavier et le place dans le registre a0.  
Ce nombre est ensuite copié dans t0 (il représente **n**, le nombre dont on veut la factorielle).

li t1, 1  
→ Initialise le registre t1 à 1 : c’est là que le résultat de la factorielle sera calculé.

li t2, 1  
→ Initialise un compteur i à 1 dans le registre t2.

mul t1, t1, t2  
→ Multiplie le résultat courant par i. C’est le cœur de la factorielle.

addi t2, t2, 1  
→ Incrémente le compteur i de 1 à chaque tour de boucle.

ble t2, t0, boucle  
→ Si le compteur i est inférieur ou égal à n, on continue la boucle.

la a0, message\_resultat + li a7, 4 + ecall  
→ Affiche un message avant d’afficher le résultat.

mv a0, t1 + li a7, 1 + ecall  
→ Affiche le résultat de la factorielle (stocké dans t1).

li a7, 10 + ecall  
→ Termine le programme proprement.

[Début de la boucle]

│

▼

Multiplier résultat (t1)

par le compteur (t2)

t1 = t1 \* t2

│

▼

Incrémenter t2 (i)

t2 = t2 + 1

│

▼

Est-ce que t2 <= t0 ? (encore à multiplier ?)

┌───────┴───────┐

│ │

Oui Non

│ │

▼ ▼

[boucle] Affichage résultat

Conclusion

Le projet permet de comprendre les bases de l’architecture RISC-V, notamment son jeu d’instructions simple et ses formats variés. La programmation en assembleur permet de saisir la gestion fine des registres et le déroulement des instructions dans un processeur. RISC-V, par sa simplicité et son ouverture, est une architecture d’avenir, largement adoptée dans la recherche et l’industrie.