SE201 : Projet 1

Lucie Molinié, Isaïe Muron, Florian Tarazona

Partie 1 - Jeu d'instruction RISC-V

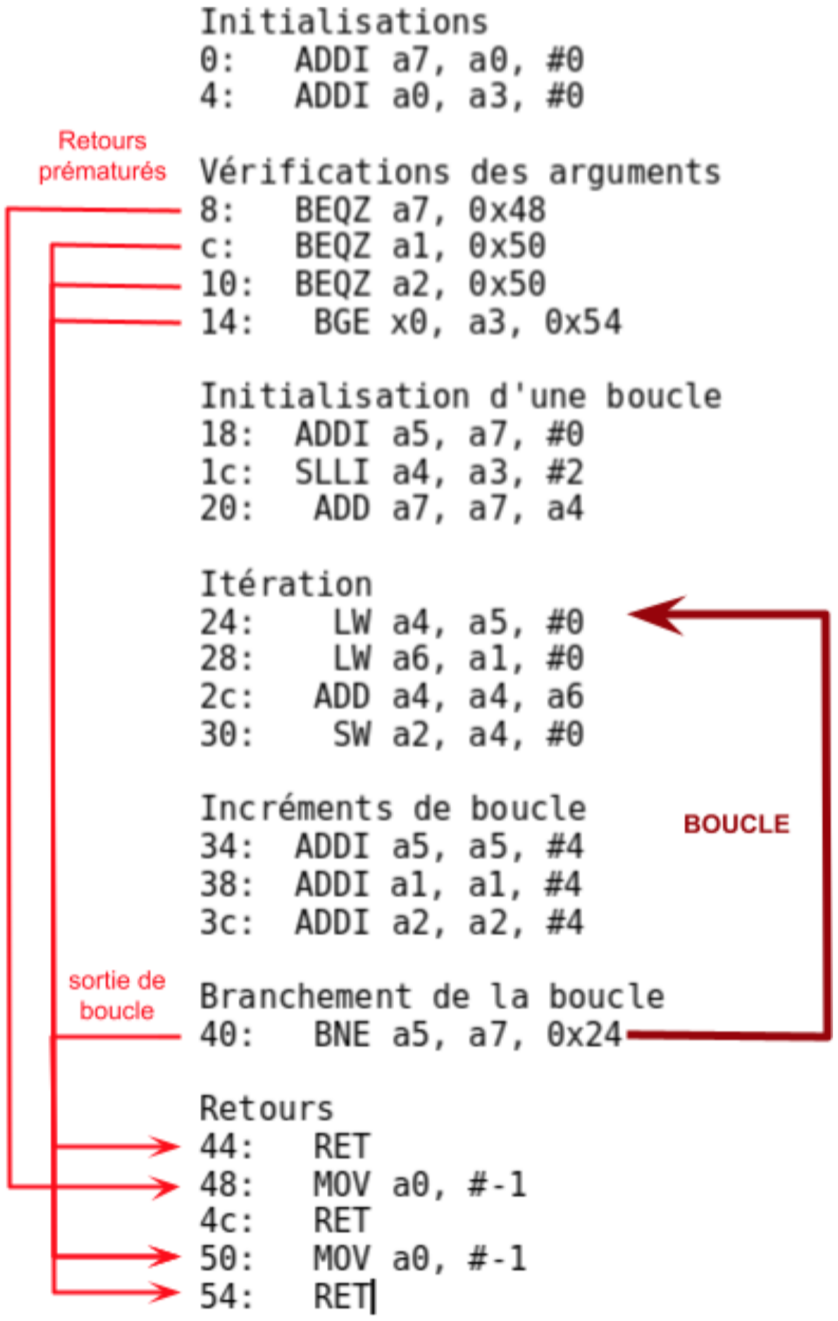
Le programme

Pour traduire ce programme : - On traduit d'abord les instructions hexadécimales en binaires - On identifie le format des instructions et leur mnémonique à l'aide de la documentation *RISC-V* (p.130) et de la dernière page du sujet

On obtient le résultat suivant :

```
0:   0000 0000 0000 0101 0000 1000 1001 0011 ADDI a7, a0, #0   (I)
4:   0000 0000 0000 0110 1000 0101 0001 0011 ADDI a0, a3, #0   (I)
8:   0000 0100 0000 1000 1000 0000 0110 0011  BEQ a7, x0, #64  (SB)
c:   0000 0100 0000 0101 1000 0010 0110 0011  BEQ a1, x0, #68  (SB)
10:  0000 0100 0000 0110 0000 0000 0110 0011  BEQ a2, x0, #64  (SB)
14:  0000 0100 1101 0000 0101 0000 0110 0011  BGE x0, a3, #64  (SB)
18:  0000 0000 0000 1000 1000 0111 1001 0011 ADDI a5, a7, #0   (I)
1c:  0000 0000 0010 0110 1001 0111 0001 0011 SLLI a4, a3, #2   (I)
20:  0000 0000 1110 1000 1000 1000 1011 0011  ADD a7, a7, a4   (R)
24:  0000 0000 0000 0111 1010 0111 0000 0011   LW a4, a5, #0   (I)
28:  0000 0000 0000 0101 1010 1000 0000 0011   LW a6, a1, #0   (I)
2c:  0000 0001 0000 0111 0000 0111 0011 0011  ADD a4, a4, a6   (R)
30:  0000 0000 1110 0110 0010 0000 0010 0011   SW a2, a4, #0   (S)
34:  0000 0000 0100 0111 1000 0111 1001 0011 ADDI a5, a5, #4   (I)
38:  0000 0000 0100 0101 1000 0101 1001 0011 ADDI a1, a1, #4   (I)
3c:  0000 0000 0100 0110 0000 0110 0001 0011 ADDI a2, a2, #4   (I)
40:  1111 1111 0001 0111 1001 0010 1110 0011  BNE a5, a7, #-28 (SB)
44:  0000 0000 0000 0000 1000 0000 0110 0111 JALR x0, x1, #0   (I)
48:  1111 1111 1111 0000 0000 0101 0001 0011 ADDI a0, x0, #-1  (I)
4c:  0000 0000 0000 0000 1000 0000 0110 0111 JALR x0, x1, #0   (I)
50:  1111 1111 1111 0000 0000 0101 0001 0011 ADDI a0, x0, #-1  (I)
54:  0000 0000 0000 0000 1000 0000 0110 0111 JALR x0, x1, #0   (I)
```

On peut organiser le code en plusieurs parties et l'annoter pour mettre en avant les branchements :



Les incréments multiples de 4 pour `a1`, `a2` et `a5`suggèrent que ces registres contiennent des adresses. Cela est confirmé par leurs utilisations dans les instructions d'accès mémoire.

Le corps de la boucle charge deux valeurs, les additionne puis stocke le résultats dans une autre partie de la mémoire : **la fonction est un additionneur vectoriel**. Donnons finalement l'usage de la fonction :

```
int add_vector(sc1_array(a0), // premier vecteur de à additionner                sc2_array(a1),
// second vecteur de à additionner                  dst_array(a2), // vecteur résultat
size(a3) // taille des vecteurs                  );
```
La fonction renvoie `-1` si l'un des vecteurs `dst`, `sc1`, `sc2` a une adresse nulle, `size` sinon.

Les Branch Delay Slots

Dans tous les processeurs dont l'architecture est basée sur un pipeline, les instructions de branchement impliquent de casser le pipeline.
En effet, prenons l'exemple d'un processeur RISC-V, basé sur un pipeline à 5 étages comme vu en cours, qui exécute l'instruction

```
bge a0, a1, #20
```

Supposons que `a0 < a1`. Le processeur ne pourra s'en rendre compte qu'à l'étage d'EXÉCUTION du pipeline. À ce moment, l'instruction suivante dans la mémoire sera déjà dans l'étage de DÉCODAGE. Au coup d'horloge suivant, cette dernière, alors qu'elle vient d'être décodée, sera tuée par le processeur. Cela fait perdre au processeur un coup d'horloge.

Dans des architectures assez anciennes comme *MIPS*, la technique des branch delay slots était employée. Elle consiste simplement à exécuter l'instruction suivant une instruction de branchement. **C'était au programmeur de prêter attention à ce qu'il faisait.** Il pouvait toutefois profiter de cette instruction afin de prévoir par exemple une instruction pour aller chercher une donnée. Il pouvait concevoir des optimisation très fines, mais le code devenait moins lisible et les compilateurs plus difficiles à optimiser.

RISC-V, au contraire, prend le parti de ne pas nécessairement exécuter une instruction suivant un branchement. Cela permet au programmeur de gérer son code de manière plus naturelle. Toutefois, il est possible d'implémenter au sein du processeur un système de prédiction de branchement qui permettra de tenter de charger directement la bonne instruction.