Exemple de tests

Pour s'assurer de la bonne implémentation de la ZAM et de son fonctionnement, quelques exemples ont été créé pour tester au mieux les offres attendues par le simulateur. Ils sont à placer dans le main après avoir instancié un simulateur de la manière suivante :

```
def main(args: Array[String]) { ...

       val Simulateur = new ZAM.Simulator()
```

Premier test :

Ce premier test construit un code mettant l'entier 2 puis 4 dans la pile pour ensuite mettre un entier 50 en variable globale. Il est ensuite exécuté par une thread qui avance et recule dans son exécution.

```
/**
        * Test implantation
        * 1) Un thread et instruction simple
        */
       Simulateur.AST.add(new ZAM.Const(2))
       Simulateur.AST.add(new ZAM.Push)
       Simulateur.AST.add(new ZAM.Const(4))
       Simulateur.AST.add(new ZAM.Push)
       Simulateur.AST.add(new ZAM.Constint(50))
       Simulateur.AST.add(new ZAM.Setglobal(3))

       Simulateur.Preparer()
       //print
       print(Simulateur.toString())

       Simulateur.Env.pushthread(new ZAM.ThreadState)
       //execution
       Simulateur.Avancer(0, 5)

       println(Simulateur.printthread(0))

       Simulateur.Revenir(0, 2)

        //print
       println(Simulateur.printthread(0))
       println(Simulateur.printenv())

       Simulateur.Avancer(0, 3)

         //print
       println(Simulateur.printthread(0))
       println(Simulateur.printenv())
```

Deuxième test :

Dans ce deuxième test les entiers 10 et 55 sont aussi ajouté dans l'environnement globale. L'entier 55 étant la même variable globale (elle écrase donc le 50). Ce code est ensuite exécuté par deux threads avançant ou reculant indépendamment.

```
/**
        * Test implantation
        * 2) Deux threads et instruction simple
        */
       Simulateur.AST.add(new ZAM.Const(2))
       Simulateur.AST.add(new ZAM.Push)
       Simulateur.AST.add(new ZAM.Const(4))
       Simulateur.AST.add(new ZAM.Push)
       Simulateur.AST.add(new ZAM.Constint(50))
       Simulateur.AST.add(new ZAM.Setglobal(3))
       Simulateur.AST.add(new ZAM.Constint(10))
       Simulateur.AST.add(new ZAM.Setglobal(2))
       Simulateur.AST.add(new ZAM.Constint(55))
       Simulateur.AST.add(new ZAM.Setglobal(3))

       Simulateur.Preparer()
       //print
       print(Simulateur.toString())

       //deux threads
       Simulateur.Env.pushthread(new ZAM.ThreadState)
       Simulateur.Env.pushthread(new ZAM.ThreadState)

       //execution
       Simulateur.Avancer(0, 8)
       println("**********************\n")
       println(Simulateur.printthread(0))
       println(Simulateur.printenv())

        //execution
       Simulateur.Avancer(1, 10)
       println("**********************\n")
       println(Simulateur.printthread(1))
       println(Simulateur.printenv())

       Simulateur.Revenir(1, 4)

        //print
       println("**********************\n")
       println(Simulateur.printthread(0))
       println(Simulateur.printthread(1))
       println(Simulateur.printenv())

       Simulateur.Avancer(0, 2)

         //print
       println("**********************\n")
       println(Simulateur.printthread(0))
       println(Simulateur.printenv())
```

Troisième test :

Ce troisième test utilise les blocs, et ses méthodes de gestions. Ici on crée un string et l'on change la variable globale stockant cette chaine par sa 6° lettre (le W). Trois threads exécutent ce code.

```
/**
       * Test implantation
       * 3) Trois threads et blocs (string)
       */
      Simulateur.AST.add(new ZAM.Const('e'))
      Simulateur.AST.add(new ZAM.Pushconst('l'))
      Simulateur.AST.add(new ZAM.Pushconst('l'))
      Simulateur.AST.add(new ZAM.Pushconst('o'))
      Simulateur.AST.add(new ZAM.Pushconst('W'))
      Simulateur.AST.add(new ZAM.Pushconst('o'))
      Simulateur.AST.add(new ZAM.Pushconst('r'))
      Simulateur.AST.add(new ZAM.Pushconst('l'))
      Simulateur.AST.add(new ZAM.Pushconst('d'))

      Simulateur.AST.add(new ZAM.Pushconst('h'))

      Simulateur.AST.add(new ZAM.Makeblock(10, ZAM.BlockT.string_t.id))

      Simulateur.AST.add(new ZAM.Setglobal(3))

      Simulateur.AST.add(new ZAM.Getglobalfield(3,5))
      Simulateur.AST.add(new ZAM.Setglobal(3))


      Simulateur.Preparer()
      //print
      print(Simulateur.toString())

      //trois threads
      Simulateur.Env.pushthread(new ZAM.ThreadState)
      Simulateur.Env.pushthread(new ZAM.ThreadState)
      Simulateur.Env.pushthread(new ZAM.ThreadState)

      //execution
      Simulateur.Avancer(0, 11)
      println("**********************\n")
      println(Simulateur.printthread(0))
      println(Simulateur.printenv())

       //execution
      Simulateur.Avancer(1, 12)
      println("**********************\n")
      println(Simulateur.printthread(1))
      println(Simulateur.printenv())

      Simulateur.Revenir(1, 4)

       //print
      println("**********************\n")
      println(Simulateur.printthread(0))
      println(Simulateur.printthread(1))
      println(Simulateur.printenv())
```

```
Simulateur.Avancer(0, 2)
Simulateur.Avancer(2, 14)

  //print
println("**********************\n")
println(Simulateur.printthread(0))
 println(Simulateur.printthread(2))
println(Simulateur.printenv())
```

Quatrième test :

Dans ce test on crée un tableau de double. Un double étant lui même un bloc cela test une grande partie des instructions concernant la gestion de block. Ce code exécuté par un seul thread crée 5 blocs de double comportant chacun deux entiers, puis un tableau de double avec ces 5 blocs

```
/**
        * Test implantation
        * 4) thread et blocs (Double Array)
        */

       Simulateur.AST.add(new ZAM.Const(156))
       Simulateur.AST.add(new ZAM.Pushconst(123))

       Simulateur.AST.add(new ZAM.Makeblock(2, ZAM.BlockT.double_t.id))
       Simulateur.AST.add(new ZAM.Setglobal(0))

       Simulateur.AST.add(new ZAM.Const(256))
       Simulateur.AST.add(new ZAM.Pushconst(223))
       Simulateur.AST.add(new ZAM.Makeblock(2, ZAM.BlockT.double_t.id))

       Simulateur.AST.add(new ZAM.Setglobal(1))
       Simulateur.AST.add(new ZAM.Const(356))
       Simulateur.AST.add(new ZAM.Pushconst(323))
       Simulateur.AST.add(new ZAM.Makeblock(2, ZAM.BlockT.double_t.id))

       Simulateur.AST.add(new ZAM.Setglobal(2))
       Simulateur.AST.add(new ZAM.Const(456))
       Simulateur.AST.add(new ZAM.Pushconst(423))
       Simulateur.AST.add(new ZAM.Makeblock(2, ZAM.BlockT.double_t.id))

       Simulateur.AST.add(new ZAM.Setglobal(3))
       Simulateur.AST.add(new ZAM.Const(556))
       Simulateur.AST.add(new ZAM.Pushconst(523))
       Simulateur.AST.add(new ZAM.Makeblock(2, ZAM.BlockT.double_t.id))


       Simulateur.AST.add(new ZAM.Pushgetglobal(3))
       Simulateur.AST.add(new ZAM.Pushgetglobal(2))
       Simulateur.AST.add(new ZAM.Pushgetglobal(1))
       Simulateur.AST.add(new ZAM.Pushgetglobal(0))

       Simulateur.AST.add(new ZAM.Makefloatblock(5))

       Simulateur.AST.add(new ZAM.Setglobal(4))

       Simulateur.AST.add(new ZAM.Getglobal(4))

       Simulateur.AST.add(new ZAM.Getfloatfield(2))

       Simulateur.AST.add(new ZAM.Pushgetglobal(4))

       Simulateur.AST.add(new ZAM.Setfloatfield(1))

       Simulateur.Preparer()
       //print
       print(Simulateur.toString())

       //un thread
       Simulateur.Env.pushthread(new ZAM.ThreadState)
```

```
//execution
Simulateur.Avancer(0, 2)
println("**********************\n")
println(Simulateur.printthread(0))
println(Simulateur.printenv())

 Simulateur.Avancer(0, 2)
println("**********************\n")
println(Simulateur.printthread(0))
println(Simulateur.printenv())

 Simulateur.Avancer(0, 2)
println("**********************\n")
println(Simulateur.printthread(0))
println(Simulateur.printenv())

 Simulateur.Avancer(0, 18)
println("**********************\n")
println(Simulateur.printthread(0))
println(Simulateur.printenv())
```

Cinquème test :

Dans ce test les opérations d'entiers, booléens et les branchements. Chaque thread test ses opérations et se retrouve à l'instruction switch. Les instructions de vecteurs sont ensuite testées.

```
      /**
         * Test implantation
         * 5) thread, arithmetic et sauts
         */

        Simulateur.AST.add(new ZAM.Const(4))
        Simulateur.AST.add(new ZAM.Setglobal(0))

        Simulateur.AST.add(new ZAM.Const(2))
        Simulateur.AST.add(new ZAM.Pushgetglobal(0))

        Simulateur.AST.add(new ZAM.Integer_comparision("EQ"))
        Simulateur.AST.add(new ZAM.Branchif(26)) //if varglob(0) == 2 -> pc +20

        // {
        Simulateur.AST.add(new ZAM.Const(1))
        Simulateur.AST.add(new ZAM.Pushgetglobal(0))
        Simulateur.AST.add(new ZAM.Subint) // var --
        Simulateur.AST.add(new ZAM.Setglobal(0))

        Simulateur.AST.add(new ZAM.Getglobal(0))
        Simulateur.AST.add(new ZAM.Pushconst(3))
        Simulateur.AST.add(new ZAM.Integer_comparision("LEINT"))
        Simulateur.AST.add(new ZAM.Branchifnot(9)) //if! ( 3 <=  varglob(0)) ->
pc + 20

        // {
        Simulateur.AST.add(new ZAM.Const(2))
        Simulateur.AST.add(new ZAM.Pushconst(2))
        Simulateur.AST.add(new ZAM.Addint) //2+2

        Simulateur.AST.add(new ZAM.Pushconst(6))
        Simulateur.AST.add(new ZAM.Pushconst(6))
        Simulateur.AST.add(new ZAM.Mulint) //6*6

        Simulateur.AST.add(new ZAM.Pushconst(2))
        Simulateur.AST.add(new ZAM.Branch(22))
        //}
        //{ ifnot
        Simulateur.AST.add(new ZAM.Const(1))
        Simulateur.AST.add(new ZAM.Negint) //~1

        Simulateur.AST.add(new ZAM.Pushconst(3))
        Simulateur.AST.add(new ZAM.Pushconst(9))
        Simulateur.AST.add(new ZAM.Divint) // 9/3

        Simulateur.AST.add(new ZAM.Pushconst(2))
        Simulateur.AST.add(new ZAM.Pushconst(10))
        Simulateur.AST.add(new ZAM.Modint) // 10%2

        Simulateur.AST.add(new ZAM.Integer_branch_comparision(0, 13, "BEQ"))
        //}}
        // if {
      Simulateur.AST.add(new ZAM.Const(2))
      Simulateur.AST.add(new ZAM.Pushconst(6))
      Simulateur.AST.add(new ZAM.Andint) // 6 & 2
```

```
Simulateur.AST.add(new ZAM.Pushconst(2))
Simulateur.AST.add(new ZAM.Pushconst(4))
Simulateur.AST.add(new ZAM.Orint) // 4 | 2

Simulateur.AST.add(new ZAM.Pushconst(2))
Simulateur.AST.add(new ZAM.Pushconst(6))
Simulateur.AST.add(new ZAM.Xorint) // 6 ^ 2

Simulateur.AST.add(new ZAM.Integer_comparision("GTINT")) // 4 > 6
Simulateur.AST.add(new ZAM.Boolnot) //true
Simulateur.AST.add(new ZAM.Branchifnot(205)) //not jump
//}

Simulateur.AST.add(new ZAM.Switch(0, Array(1, 6, 11)))

Simulateur.AST.add(new ZAM.Const(2))
Simulateur.AST.add(new ZAM.Pushconst(2))
Simulateur.AST.add(new ZAM.Lslint) // 2 << 2
Simulateur.AST.add(new ZAM.Makeblock(3, ZAM.BlockT.double_t.id))
Simulateur.AST.add(new ZAM.Setglobal(0))

Simulateur.AST.add(new ZAM.Const(2))
Simulateur.AST.add(new ZAM.Pushconst(2))
Simulateur.AST.add(new ZAM.Lsrint) // 2 >> 2
Simulateur.AST.add(new ZAM.Getglobal(0))
Simulateur.AST.add(new ZAM.Vectlength)

Simulateur.AST.add(new ZAM.Const(2))
Simulateur.AST.add(new ZAM.Pushconst(2))
Simulateur.AST.add(new ZAM.Asrint) // 2 >> 2
Simulateur.AST.add(new ZAM.Const(2))
Simulateur.AST.add(new ZAM.Push)
Simulateur.AST.add(new ZAM.Getglobal(0))
Simulateur.AST.add(new ZAM.Setvectitem)

 Simulateur.Preparer()
 //print
 print(Simulateur.toString())

 //un thread
 Simulateur.Env.pushthread(new ZAM.ThreadState)
 Simulateur.Env.pushthread(new ZAM.ThreadState)
 Simulateur.Env.pushthread(new ZAM.ThreadState)

 Simulateur.Avancer(0, 2)
 Simulateur.Avancer(1, 2)
 Simulateur.Avancer(2, 2)

 Simulateur.Avancer(0, 20)
 println("**********************\n")
 println(Simulateur.printthread(0))
 println(Simulateur.printenv())

 Simulateur.Avancer(1, 21)
 println("**********************\n")
 println(Simulateur.printthread(1))
 println(Simulateur.printenv())

 Simulateur.Avancer(2, 16)
 println("**********************\n")
 println(Simulateur.printthread(2))
```

```
println(Simulateur.printenv())

println("**********************\n")
println("**      apres switch  **\n")

Simulateur.Avancer(1, 6)
println("**********************\n")
println(Simulateur.printthread(1))
println(Simulateur.printenv())

Simulateur.Avancer(2, 6)
println("**********************\n")
println(Simulateur.printthread(2))
println(Simulateur.printenv())

Simulateur.Avancer(0, 8)
println("**********************\n")
println(Simulateur.printthread(0))
println(Simulateur.printenv())
```