ALLOCATION DE REGISTRE

CONCEPTION

CodeOptimizer.java

```
private static class InstructionBlock
```

PLAN

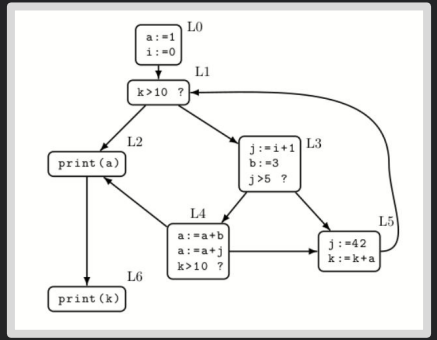
- *Conception*
- *Graphe de contrôle*
- *Vivacité*
- *Graphe de conflit*
- *Allocation*
- *Focus technique*
- *Tests*

CONCEPTION

- générer le graphe de contrôle
- calculer les ensembles LV_{entry} et LV_{exit}
- générer le graphe de conflits
- le colorer
- modifier le code fourni par le groupe 2

```
public Program optimize(Program program)
    this.program = program;

    buildControlGraph();

    computeLiveness();

    buildConflictGraph();

    this.colorSize = conflictGraph.color

    return applyAllocation();
```

GRAPHE DE CONTRÔLE




```
private void buildControlGraph()
```

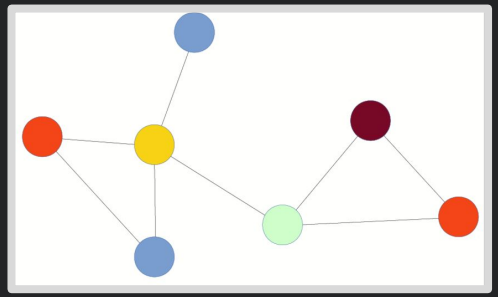
- `createBlocks();`
- `linkBlocks();`

VIVACITÉ

LV_{entry} et LV_{exit}

- `private void computeLiveness()`
  - `computeGenKill()`
  - `getReadRegisters(instruction)`
  - `getWrittenRegister(instruction)`

GRAPHE DE CONFLIT



```
private void buildConflictGraph()
```

- `this.conflictGraph = new UnorientedGraph<Integer>();`
- `conflictGraph.addVertex(write);`
- `conflictGraph.addEdge(write, liveReg)`

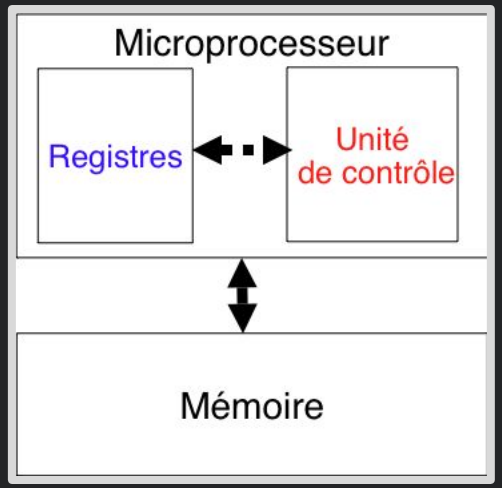
ALLOCATION

```
1 LD R0 SP
2 XOR R1 R1 R1
3 ADDi R1 R1 1
4 ADD R2 R0 R1
5 LD R3 SP
```

- `private Program applyAllocation()`
  - `getPhysicalRegister(ual.getSr1())`
  - `storeSpill(ual.getDest(), TMP_REG_2, dest)`
  - `loadSpill(ual.getSr1(), TMP_REG_1)`

FOCUS TECHNIQUE

- *Spilling*




```
private Program storeSpill(int virtualRegister, int regAddr, int regVal){
```

```
private Program loadSpill(int virtualRegister, int regAddr){
```

TESTS EFFECTUÉS

```java
public class AssertDemo1 {
    public static void main(String[] args) {
        int x = 5;
        assert x == 5 : "x should be 5";
        System.out.println(x);
    }
}
```

```java
// Test fonction facto
ArrayList<ArrayList<Integer>> factoResult = testCode( path: "test_facto");

// Test fonction fibo
ArrayList<ArrayList<Integer>> fiboResult = testCode( path: "test_fibo");

// Test des tableaux
ArrayList<ArrayList<Integer>> tabResult = testCode( path: "test_tab");

// Test du Spill
ArrayList<ArrayList<Integer>> spillResult = testCode( path: "test_spill");
```

CONCLUSION

```
CodeOptimizer codeOpt = new CodeOptimizer(32);
Program program = codeOpt.optimize(linearProgram)
```