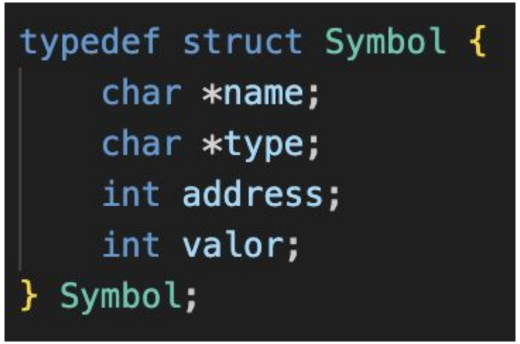
Projet système informatique

Cristian MARTINEZ
Oussama ELJID

1. COMPILATEUR

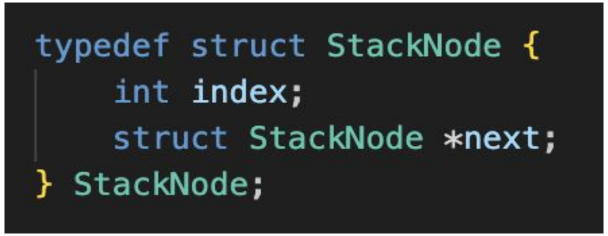
STRUCTURES IMPORTANTES

Symbol Table



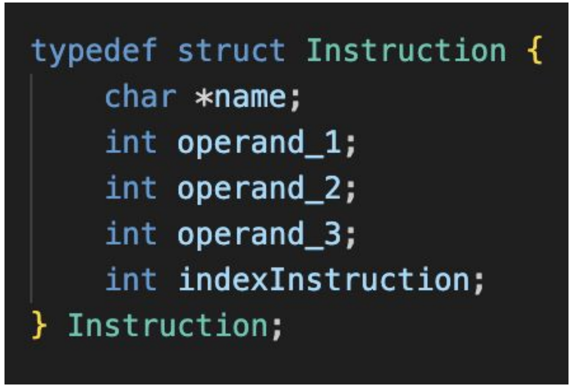
```c
typedef struct Symbol {
    char *name;
    char *type;
    int address;
    int valor;
} Symbol;
```

Stock IF - WHILE index



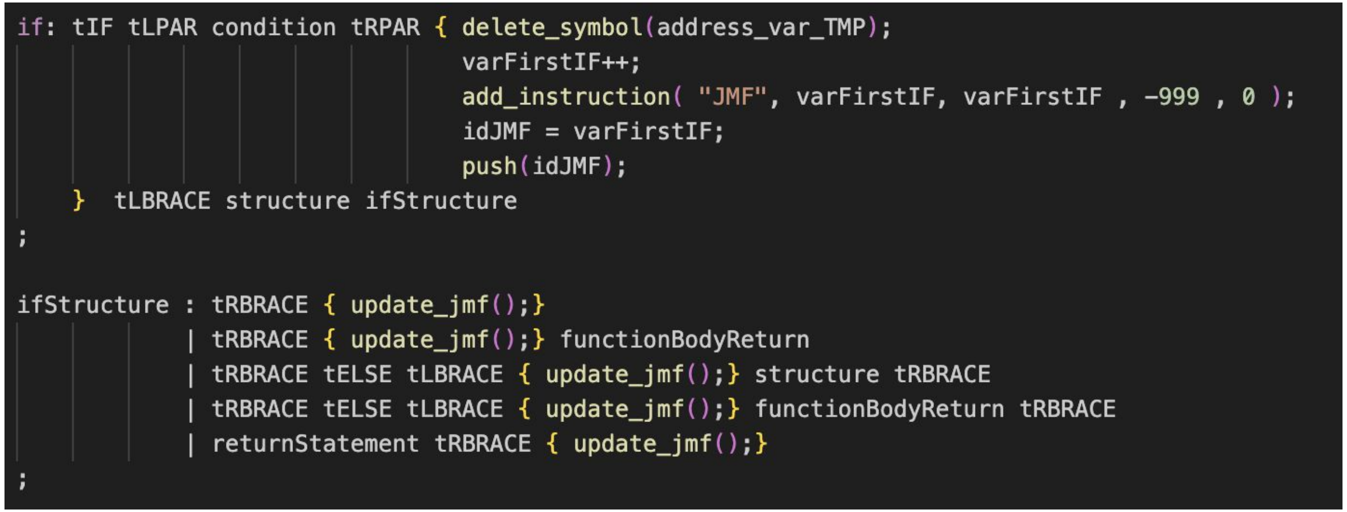
```c
typedef struct StackNode {
    int index;
    struct StackNode *next;
} StackNode;
```

Instruction Table

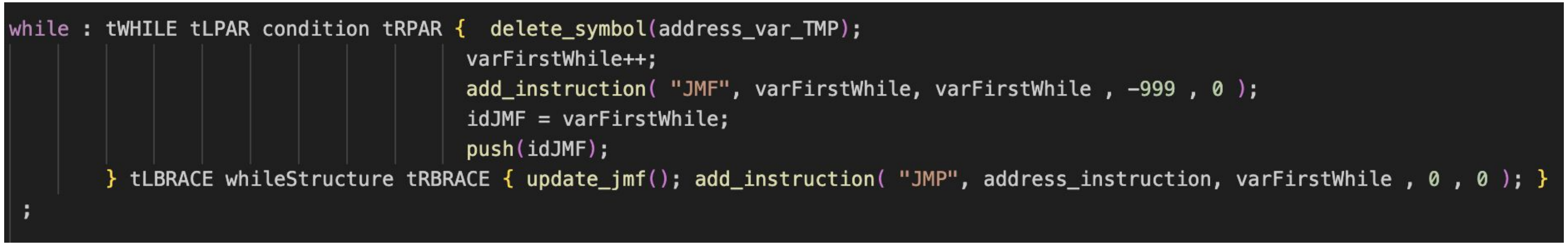


```c
typedef struct Instruction {
    char *name;
    int operand_1;
    int operand_2;
    int operand_3;
    int indexInstruction;
} Instruction;
```

IF - WHILE -> JMF



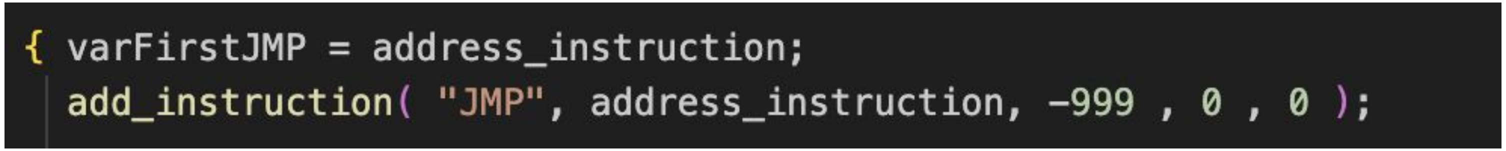
```
if: tIF tLPAR condition tRPAR { delete_symbol(address_var_TMP);
                                varFirstIF++;
                                add_instruction( "JMF", varFirstIF, varFirstIF , -999 , 0 );
                                idJMF = varFirstIF;
                                push(idJMF);
    }  tLBRACE structure ifStructure
;

ifStructure : tRBRACE { update_jmf();}
            | tRBRACE { update_jmf();} functionBodyReturn
            | tRBRACE tELSE tLBRACE { update_jmf();} structure tRBRACE
            | tRBRACE tELSE tLBRACE { update_jmf();} functionBodyReturn tRBRACE
            | returnStatement tRBRACE { update_jmf();}
;
```



```
while : tWHILE tLPAR condition tRPAR {  delete_symbol(address_var_TMP);
                                       varFirstWhile++;
                                       add_instruction( "JMF", varFirstWhile, varFirstWhile , -999 , 0 );
                                       idJMF = varFirstWhile;
                                       push(idJMF);
        } tLBRACE whileStructure tRBRACE { update_jmf(); add_instruction( "JMP", address_instruction, varFirstWhile , 0 , 0 ); }
 ;
```

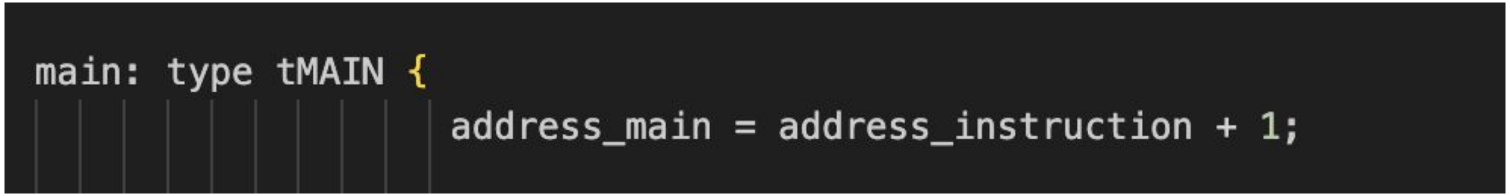
MAIN -> JMP

1. Function Found - JMP Added



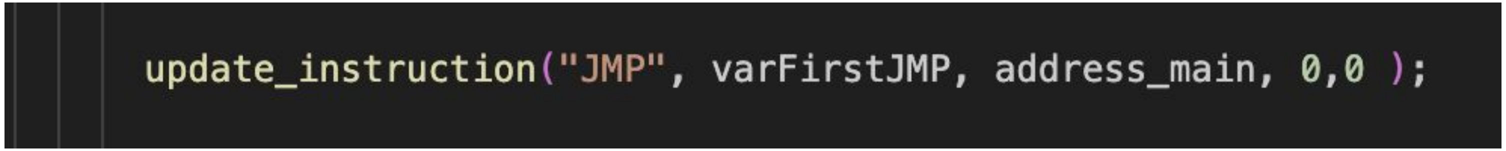
```
{ varFirstJMP = address_instruction;
  add_instruction( "JMP", address_instruction, -999 , 0 , 0 );
```

2. Main Found - Address main saved



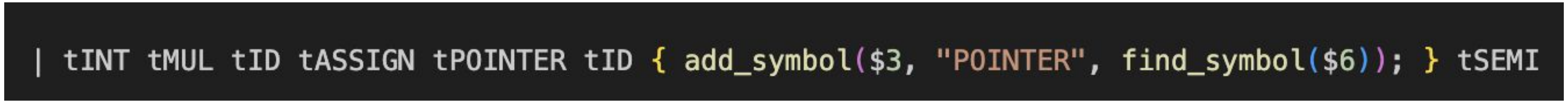
```
main: type tMAIN {
                  address_main = address_instruction + 1;
```

3. End Main - JMP updated

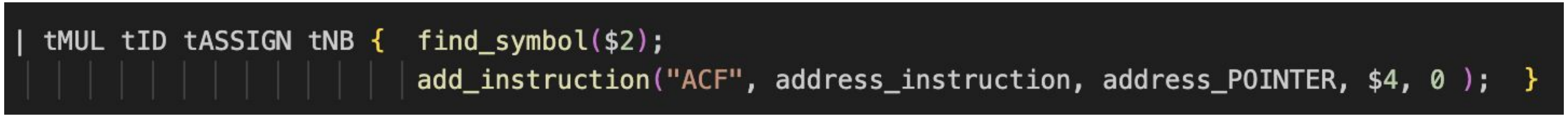


```
      update_instruction("JMP", varFirstJMP, address_main, 0,0 );
```

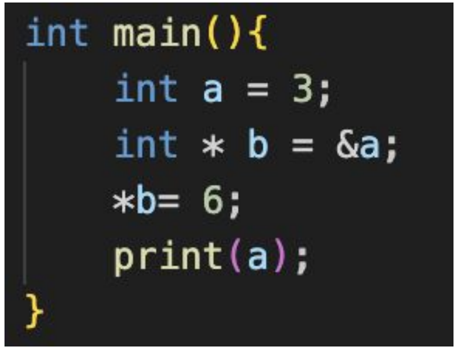
POINTERS



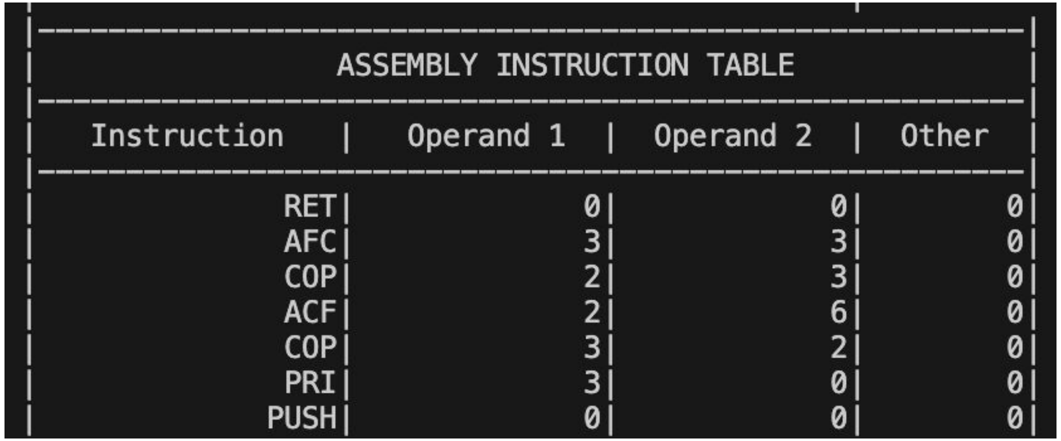
```
| tINT tMUL tID tASSIGN tPOINTER tID { add_symbol($3, "POINTER", find_symbol($6)); } tSEMI
```



```
| tMUL tID tASSIGN tNB {  find_symbol($2);
                          add_instruction("ACF", address_instruction, address_POINTER, $4, 0 );  }
```

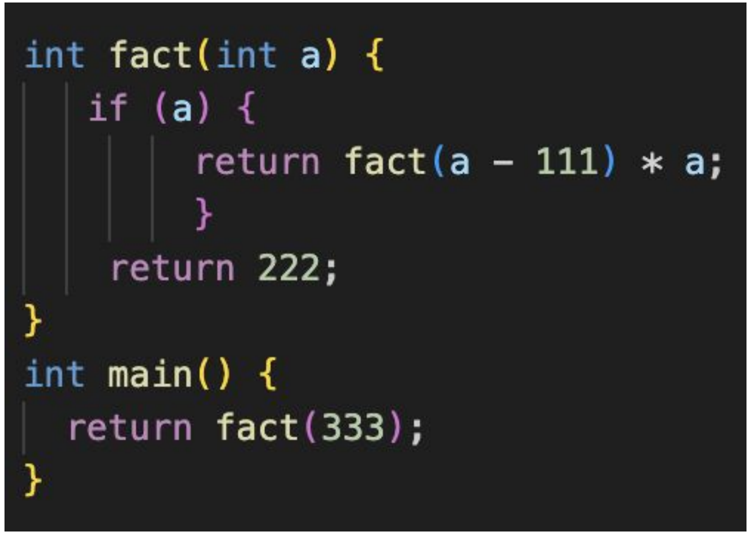


```c
int main(){
    int a = 3;
    int * b = &a;
    *b= 6;
    print(a);
}
```

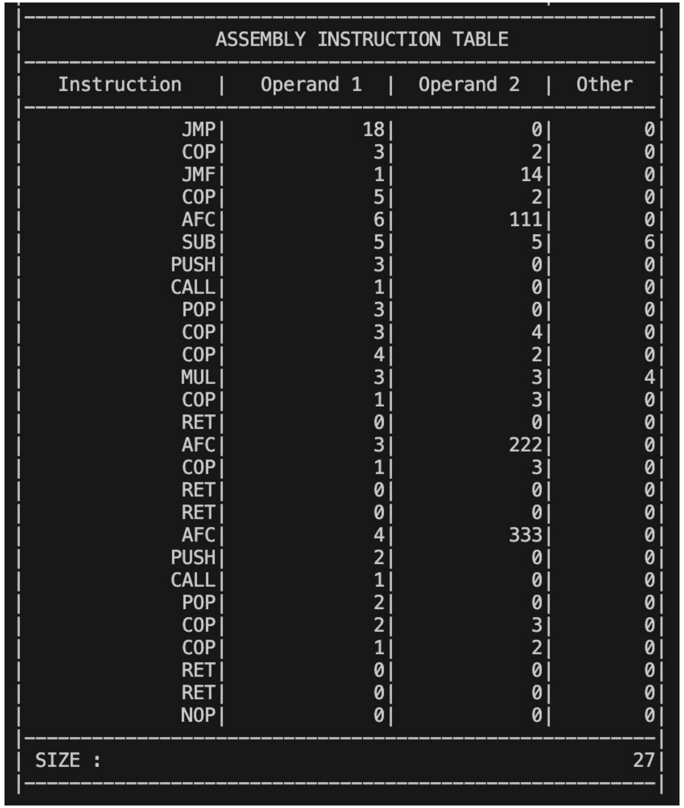


```
|-----------------------------------------------------------|
|               ASSEMBLY INSTRUCTION TABLE                  |
|-----------------------------------------------------------|
|   Instruction     |   Operand 1  |   Operand 2  |  Other  |
|-----------------------------------------------------------|
|                RET|             0|             0|        0|
|                AFC|             3|             3|        0|
|                COP|             2|             3|        0|
|                ACF|             2|             6|        0|
|                COP|             3|             2|        0|
|                PRI|             3|             0|        0|
|               PUSH|             0|             0|        0|
```

EXAMPLE EXECUTION

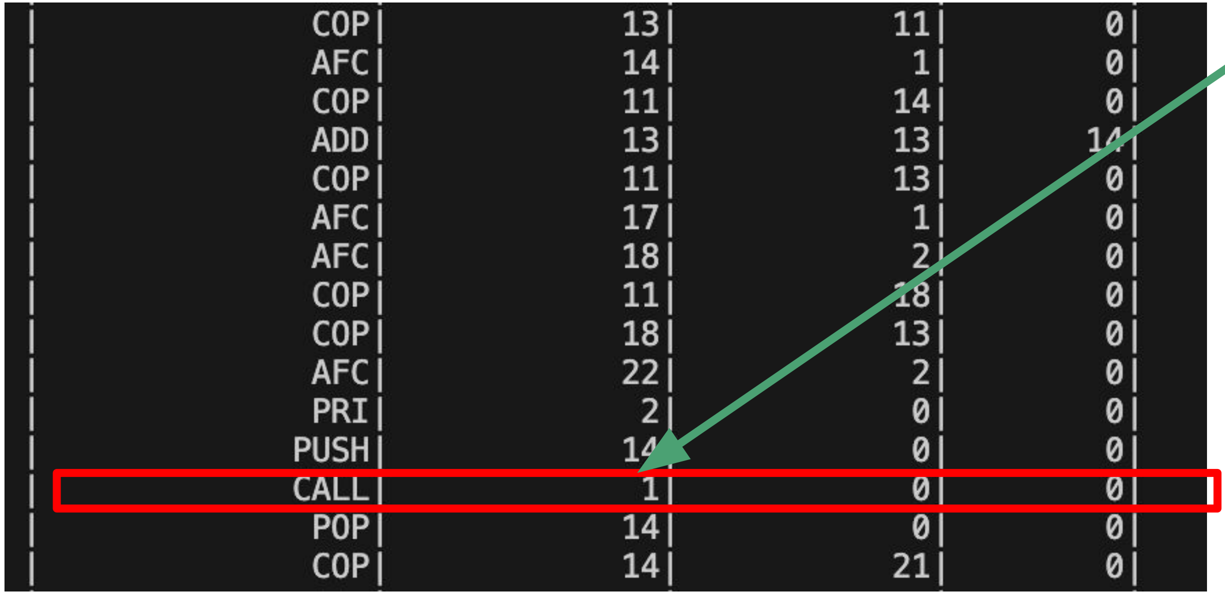


```c
int fact(int a) {
    if (a) {
        return fact(a - 111) * a;
    }
    return 222;
}
int main() {
  return fact(333);
}
```



```
|------------------------------------------------------------|
|                 ASSEMBLY INSTRUCTION TABLE                  |
|------------------------------------------------------------|
|   Instruction     |  Operand 1  |  Operand 2  |  Other     |
|------------------------------------------------------------|
|                JMP|           18|            0|           0|
|                COP|            3|            2|           0|
|                JMF|            1|           14|           0|
|                COP|            5|            2|           0|
|                AFC|            6|          111|           0|
|                SUB|            5|            5|           6|
|               PUSH|            3|            0|           0|
|               CALL|            1|            0|           0|
|                POP|            3|            0|           0|
|                COP|            3|            4|           0|
|                COP|            4|            2|           0|
|                MUL|            3|            3|           4|
|                COP|            1|            3|           0|
|                RET|            0|            0|           0|
|                AFC|            3|          222|           0|
|                COP|            1|            3|           0|
|                RET|            0|            0|           0|
|                RET|            0|            0|           0|
|                AFC|            4|          333|           0|
|               PUSH|            2|            0|           0|
|               CALL|            1|            0|           0|
|                POP|            2|            0|           0|
|                COP|            2|            3|           0|
|                COP|            1|            2|           0|
|                RET|            0|            0|           0|
|                RET|            0|            0|           0|
|                NOP|            0|            0|           0|
|------------------------------------------------------------|
| SIZE :                                                   27|
|------------------------------------------------------------|
```

CALL FUNCTION



```
|          COP|        13|        11|         0|
|          AFC|        14|         1|         0|
|          COP|        11|        14|         0|
|          ADD|        13|        13|        14|
|          COP|        11|        13|         0|
|          AFC|        17|         1|         0|
|          AFC|        18|         2|         0|
|          COP|        11|        18|         0|
|          COP|        18|        13|         0|
|          AFC|        22|         2|         0|
|          PRI|         2|         0|         0|
|         PUSH|        14|         0|         0|
|         CALL|         1|         0|         0|
|          POP|        14|         0|         0|
|          COP|        14|        21|         0|
```

@Instruction Number

2. MICROPROCESSEUR

VHDL : Création du pipeline

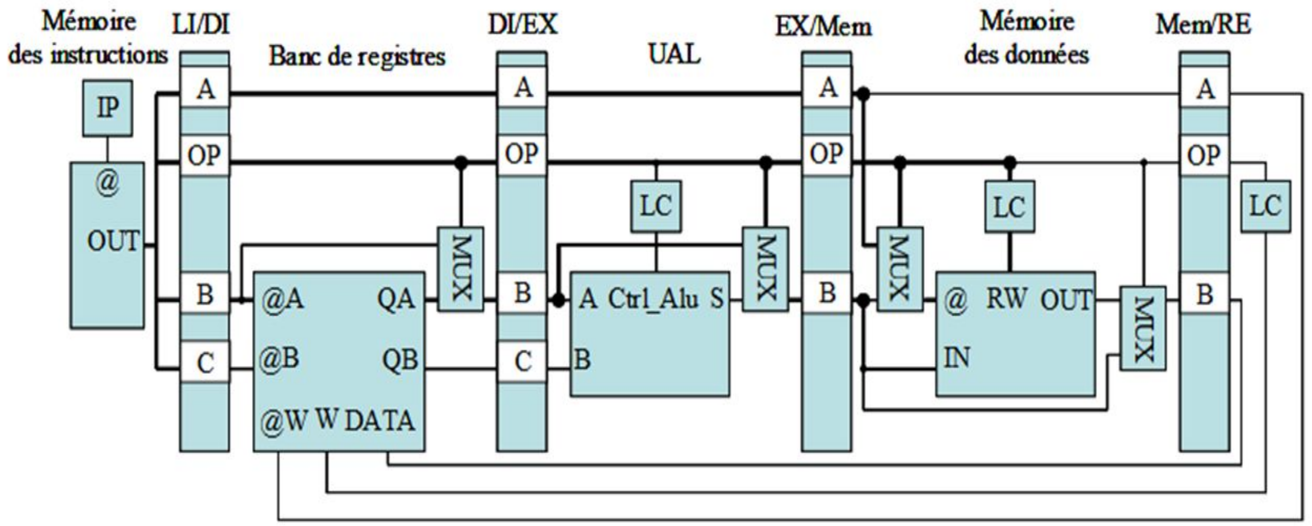
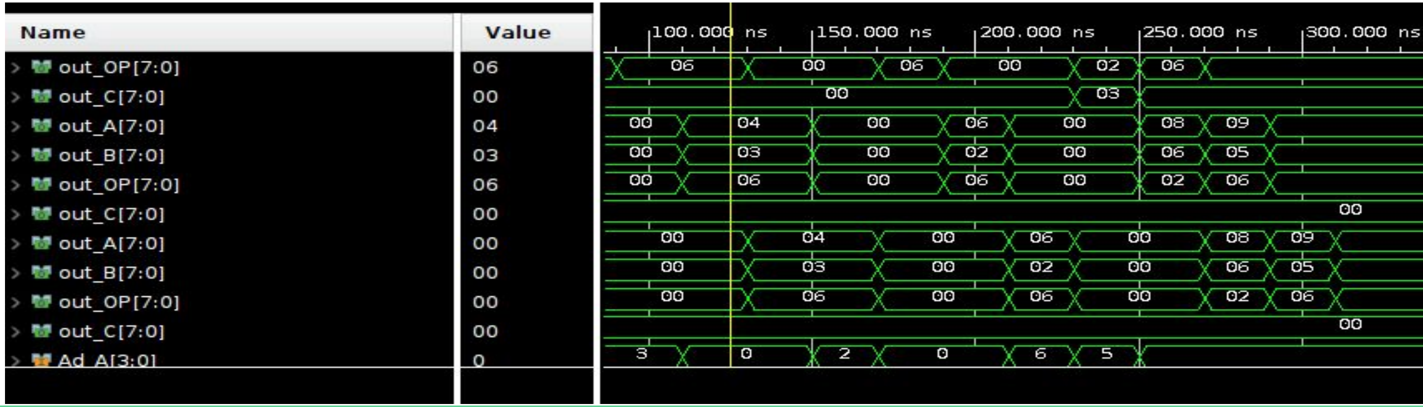


Illustration 5 : Instructions AFC, COP, ADD, MUL, DIV, SOU, LOAD, STORE



Name	Value
out_OP[7:0]	06
out_C[7:0]	00
out_A[7:0]	04
out_B[7:0]	03
out_OP[7:0]	06
out_C[7:0]	00
out_A[7:0]	00
out_B[7:0]	00
out_OP[7:0]	00
out_C[7:0]	00
Ad_A[3:0]	0

VHDL : gestion des aléas

-Aléas entre étage 1 et 2.

-Aléas entre étage 1 et 3 .

Vérification:

-A_2 = B_1 or A_2 = C_1

-A_3 = B_1 or A_3 = C_1

…

-AFC R1 3

-ADD R2 R1 R3

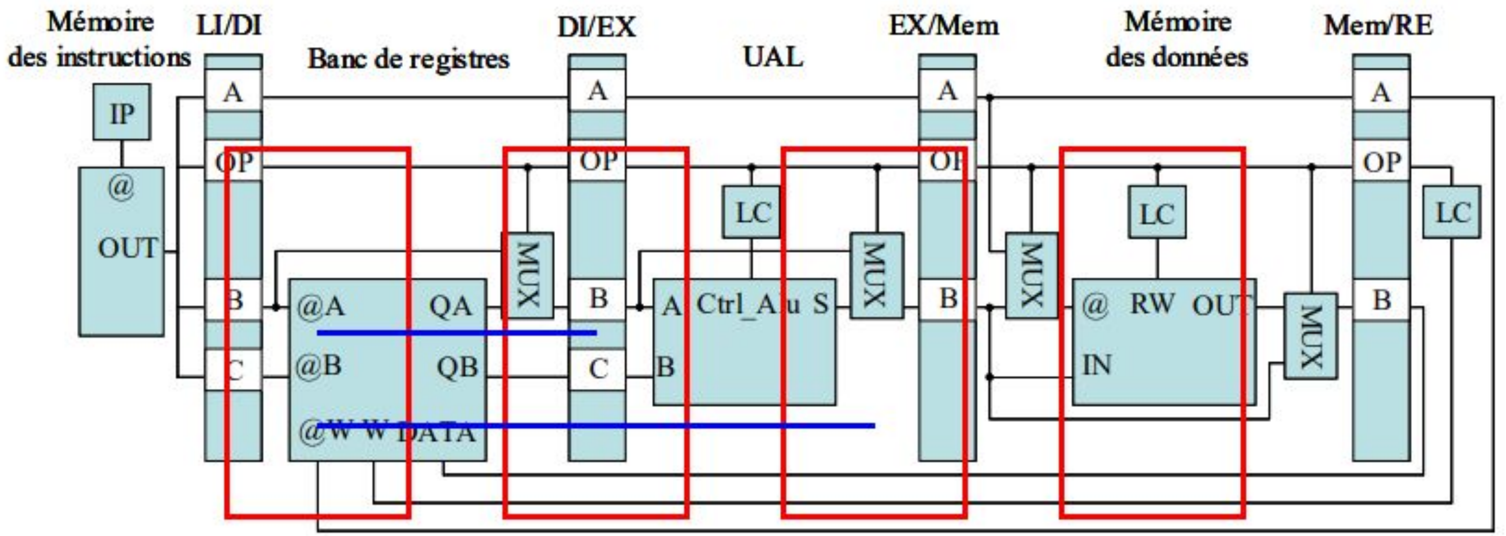


Illustration 6 : Chemin des données

MERCI