

# IFT 1227 – Architecture des ordinateurs

## Devoir 4

- Remise : Le 17.12.2023 avant minuit.

### I. Extension du processeur MIPS version un cycle

Il s'agit d'ajouter une implémentation de quelques instructions aux processeurs MIPS version un cycle et version multicycles vus en cours.

Les schémas donnés représentent les processeurs MIPS version un cycle et multicycles avec les instructions supportées **lw**, **sw**, **j**, **addi**, **or**, **and**, **add**, **sub**, **beq**. Les instructions **jr**, **andi** seront ajoutées lors de la séance de démonstration.

#### Les détails des instructions à implémenter :

Code d'opération	Nom	Description	Opération	funct
111111	<b>auipc</b> <i>rt,imm</i>	Add Upper Imm to PC	$R[rt] = PC + \{imm, 16'bits\ 0\}$	-
000011	<b>jal</b> <i>addr</i>	Saut et lien	$R[\$ra] = PC + 4, PC = JTA$	-

- Étendre les implémentations des chemins de données et des contrôleurs pour traiter les instructions **auipc** et **jal** (solutions papier schématiques).
- Intégrer votre solution schématique MIPS un cycle dans l'implémentation VHDL fournie.
- Ajuster le testbench (**testbench.vhd**) et le module mémoire d'instructions (**imem.vhd**) du processeur MIPS un cycle pour vérifier les nouvelles instructions. Fichier **imem.vhd** modélise la mémoire d'instructions (ROM) avec un programme enregistré contenant les instructions implémentées.

```
mem(0) := X"20020005"; --      addi $v0, $0, 5          # $v0(2) = 5
mem(1) := X"2003000c"; --      addi $v1, $0, 12          # $v1(3) = 12
mem(2) := X"2067fff7"; --      addi $a3, $v1, -9         # $a3(7) = $v1(3)(12) - 9 = 3
mem(3) := X"00e22025"; --      or  $a0, $a3, $v0         # $a0(4) = $a3(7) or $v0(2) = 3 or 5 = 7
mem(4) := X"00642824"; --      and  $a1, $v1, $a0         # $a1(5) = $v1(3) and $a0(4) = 12 and 7 = 4
mem(5) := X"00a42820"; --      add  $a1, $a1, $a0         # $a1(5) = $a1(5) + $a0(4) = 4 + 7 = 11
mem(6) := X"10a7000a"; --      beq  $a1, $a3, end         # $a1(5) == $a3(7) ? end: PC=PC+4; 11==3 ? PC=PC+4
mem(7) := X"0064202a"; --      slt  $a0, $v1, $a0         # $v1(3) < $a0(4) ? $a0 = 1 : $a0 = 0; 12 < 7 => $a0 = 0
mem(8) := X"10800001"; --      beq  $a0, $0, ar1          # $a0(4) == 0 ? ar1: PC = PC+4; 0==0 goto ar1
mem(9) := X"20050000"; --      addi $a1, $0, 0          #
```

```

mem(10) := X"00e2202a"; --ar1: slt  $a0, $a3, $v0      # $a3(7)<$v0(2)?$a0(4)=1:$a0=0; 3<5,$a0=1
mem(11) := X"00853820"; --      add  $a3, $a0, $a1      # $a3(7)=$a0(4)+$a1(5); 1+11=12
mem(12) := X"00e23822"; --      sub  $a3, $a3, $v0      # $a3(7)=$a3(7)-$v0(2); 12-5=7
mem(13) := X"ac670044"; --      sw   $a3, 68($v1)      # $a3(7)->M[68+$v1(3)]; 7->M[68+12=80] #test 1
mem(14) := X"8c020050"; --      lw   $v0, 80($0)       # $v0(2) = M[80+0]; $v0 = 7
mem(15) := X"08000011"; --      j     end              # goto end
mem(16) := X"20020001"; --      addi  $v0, $0, 1
mem(17) := X"ac02003C"; --end: sw   $v0, 60($0)       # $v0(2) write M[60]; M[60]=7; #test 2

```

### Notations:

- $R[reg]$  – Contenu du registre
- $imm$  – Le champ Immediate de 16 bits d'instructions de type I
- $addr$  – Le champ représentant une partie de l'adresse sur 26 bits de l'instruction de type J
- $SignImm$  – la valeur  $imm$  étendue par le bit le plus significatif de  $imm$  représentant un signe sur 32-bits (concaténer 16 fois le bit à la position 15 de  $imm$  et la valeur immédiate sur 16 bits  $imm$ ) =  $\{16\{imm[15]\}, imm\}$ ;
- $ZeroImm$  – la valeur  $imm$  étendue par zéros sur 32-bits =  $\{16'bits\ 0, imm\}$  (concaténation 16 zéros avec la valeur immédiate  $imm$ )
- $Address = R[rs] + SignImm$
- $Mem[Address]$  – contenu de la mémoire à l'adresse  $Address$
- $BTA - branch\ target\ address = PC + 4 + (SignImm \ll 2)$
- $JTA - jump\ target\ address = \{(PC + 4)[31:28], addr, 2'bits\ 0\}$  – concaténation de 4 bits les plus significatifs de 31 à 28 de la valeur  $(PC + 4)$  + le champs  $addr$  sur 26 bits + 2 bits zéros (positions 1 et 0)
- $Label$  – le texte qui indique une localisation de l'instruction

1. **Solution schématique du processeur MIPS un cycle (30 pts)**

a. **Composant chemin de données (10 pts)**

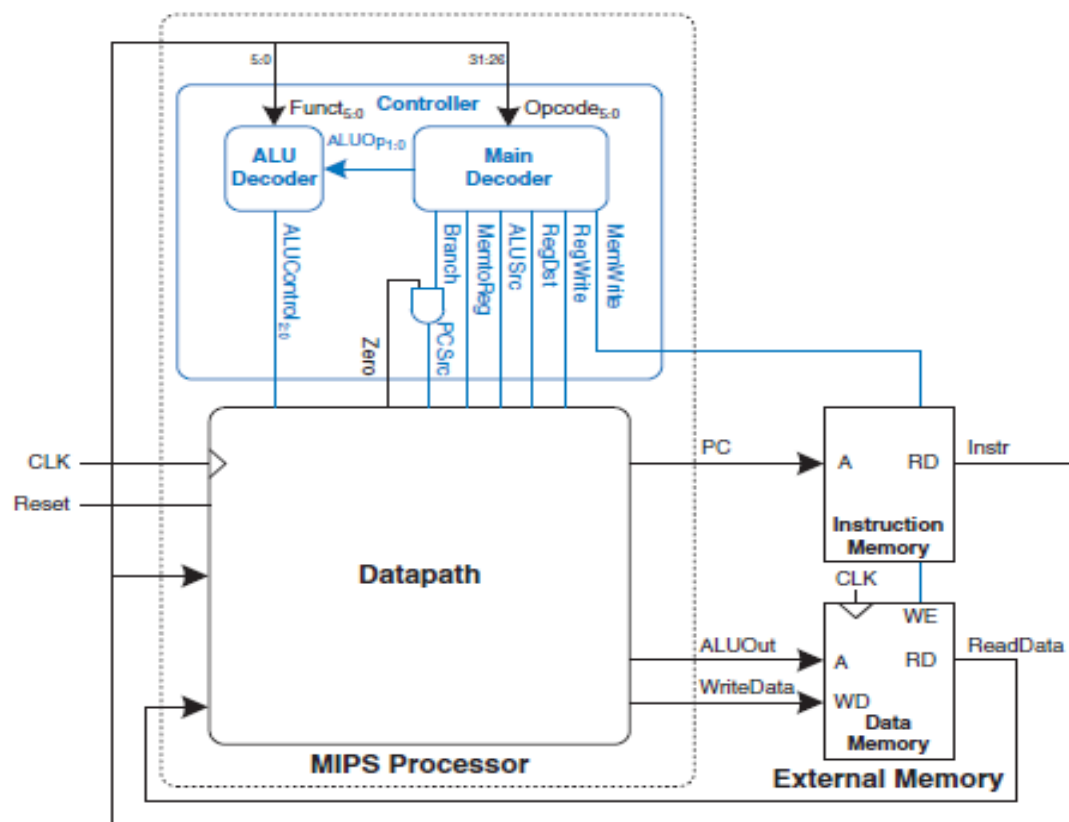
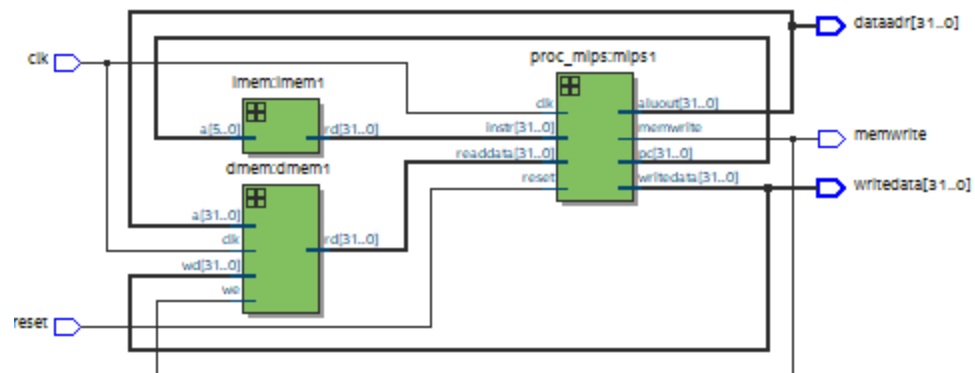
- La solution schéma des modifications du chemin de données pour les instructions **auipc** et **jal**.
- Les modifications nécessaires du chemin de données doivent être faites d'abord sur le schéma (inclure dans le rapport à remettre). Utilisez le schéma donné ou redessinez votre schéma.

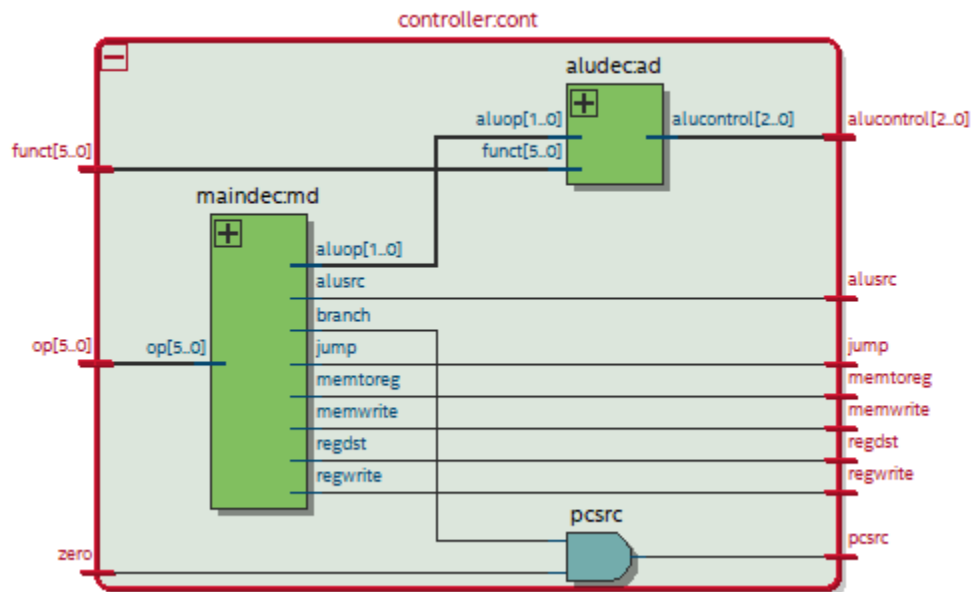
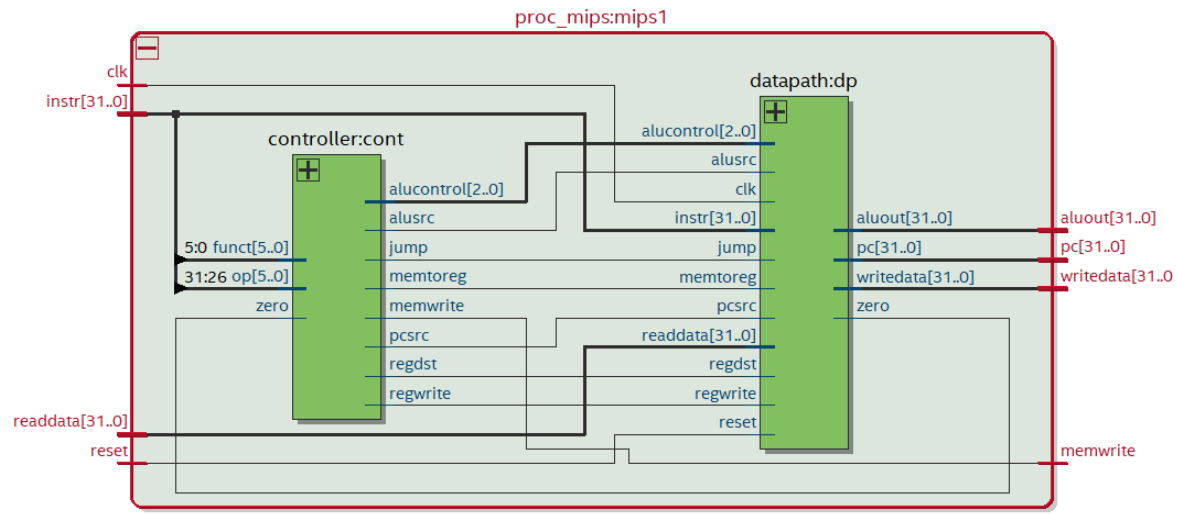
b. **Pour la partie contrôleur (tableau à compléter) (20 pts) :**

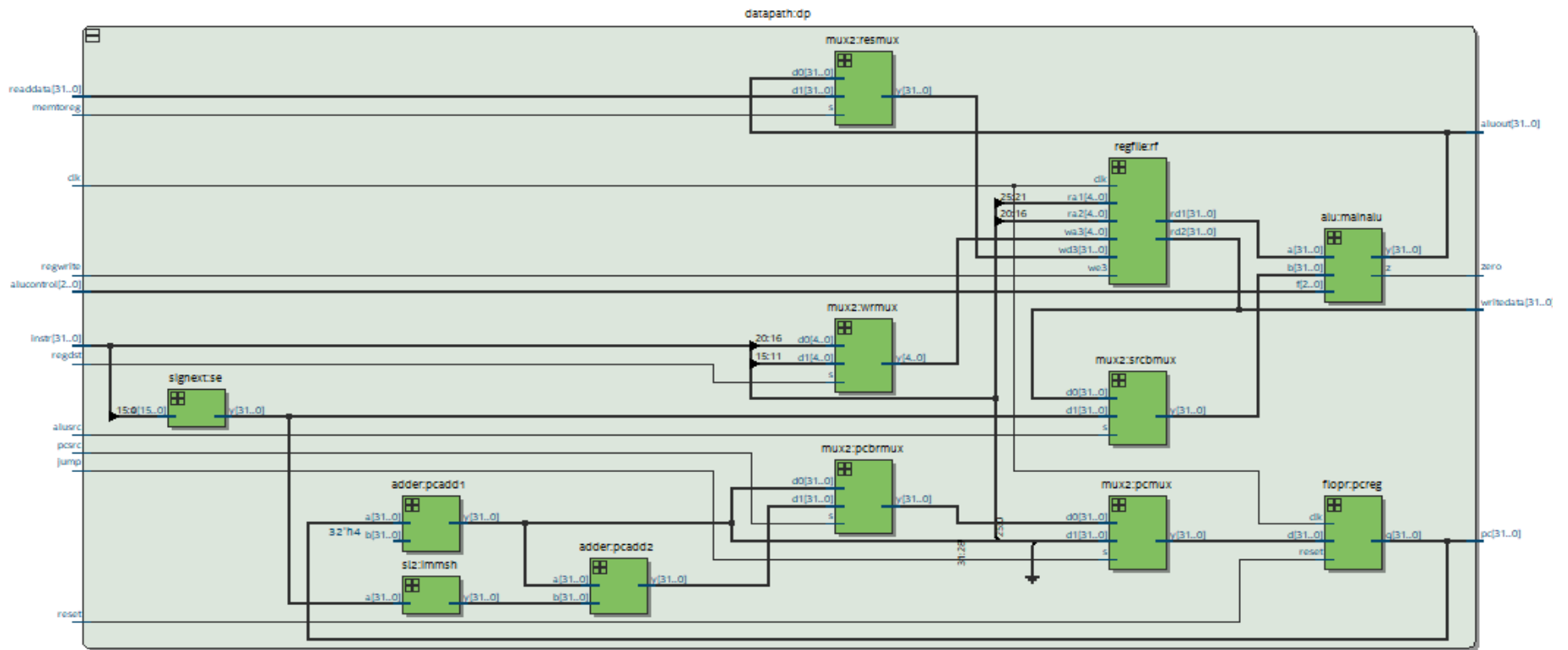
Compléter la table de vérité de l'unité de contrôle pour incorporer les nouvelles instructions.

2. **Solution VHDL du processeur MIPS un cycle (30 pts)**

- a) (20 pts) Modifier le code VHDL du processeur MIPS initial (disponible sur le site du cours) pour incorporer les 2 instructions spécifiées plus haut. Le schéma structurel du code VHDL est présenté plus bas.

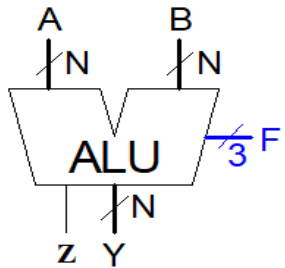






- b) (10 pts) Modifier le code du testbench et le programme dans le module imem pour tester toutes les instructions ajoutées . Simuler votre processeur en observant le résultat d'exécution des instructions spécifiées plus haut.

## ALU utilisée dans le devoir 4 :



$F_{2:0}$	Fonction
000	$A \& B$
001	$A   B$
010	$A + B$
011	Non utilisé
100	$A \& \sim B$
101	$A   \sim B$
110	$A - B$
111	SLT

```
entity alu is
    port (a, b: in STD_LOGIC_VECTOR(31 downto 0);
          f: in STD_LOGIC_VECTOR (2 downto 0);
          z: out STD_LOGIC;
          y: out STD_LOGIC_VECTOR(31 downto 0) := X"00000000");
end;
architecture behave of alu is
    signal diff, tmp: STD_LOGIC_VECTOR(31 downto 0);
begin
    diff <= a - b;
    process (a, b, f, diff, tmp) begin
        case f is
            when "000" => tmp <= a and b; --
            when "001" => tmp <= a or b; --
            when "010" => tmp <= a + b; --
            when "100" => tmp <= a and (not b); --
            when "101" => tmp <= a or (not b); --
            when "110" => tmp <= diff; --
            when "111" => --SLT
                tmp <= X"00000000" & "000" & diff(31);
            when others => tmp <= X"FFFFFFFF";
        end case;
    end process;
    z <= '1' when tmp = X"00000000" else '0';
    y <= tmp;
end;
```

### 3. Solution schématique du processeur MIPS version multicycles (30 pts)

#### a. Composant chemin de données (15 pts)

- La solution schéma des modifications du chemin de données pour les instructions `auipc` et `jal`.

#### b. Contrôleur (FSM) (15 pts) :

Incorporer les nouvelles instructions.

## **Barème**

- Q1(30 pts), Q2(30 pts), Q3(30 pts), Présentation (10 pts)

## **À remettre**

- Le rapport, format pdf : les 2 schémas, la table de vérité de l'unité de contrôle MIPS un cycle, FSM (MIPS multicycles), le programme test contenant le code assembleur et le code machine;
- Le projet doit contenir tous les fichiers sources VHDL dans le répertoire **files** pour qu'on puisse simuler et vérifier votre processeur.
- Comprimez le tout dans un seul fichier .zip.



[illegible]

