

# IFT 1227 – Architecture des ordinateurs

## Devoir 4

- À faire en groupe de **deux** étudiants.
- Remise : Le **26 avril** 2018 avant minuit.
- Il y aura une pénalité de **10%** par jour de retard

### I. Extension d'un processeur MIPS version un cycle

Il s'agit d'ajouter une implémentation de quelques instructions au processeur MIPS version un cycle vu en cours. Le schéma donné représente le processeur MIPS version un cycle 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.

- a) (8 pts) Modifier un module ALU comportemental en VHDL (fichier `alu.vhd`) pour qu'il implémente la table de vérité suivante :

F5:0	Function
000000	B sll shamt
100000	A + B
100010	A - B
100100	A and B
100101	A or B
100110	A xor B
100111	A nor B
101011	SLTU (non signé)
101010	SLT (signé)

- b) (7 pts) Écrire un testbench auto vérificateur pour tester ALU sur les données suivantes : `shamt = "00001"` et `B = X "00000001"` pour une instruction `sll` et `A = X "80000000"` `B = X "7FFFFFFF"` pour toutes les autres.
- c) (25 pts) Étendre l'implémentation du chemin de données et du contrôleur pour traiter les instructions **jr, andi, jal, lui, ori, sll, sltu, slt** et **indexIntAdr** (solution papier).
- Les modifications nécessaires du chemin de données doivent être faites d'abord sur papier (à remettre). Utilisez le schéma donné ou redessinez votre schéma.

- L'implémentation des instructions **jr**, **andi** sera montrée lors de séance de démonstration. L'intégration de ces instructions dans votre solution (papier + VHDL) sera notée (total - 10 pts).
- La solution papier des modifications du chemin de données pour les instructions **jal**, **lui**, **ori**, **indexIntAdr** (4\*3=12 pts), **sll** (1 pts), **sltu** (1 pts) et **slt** (1 pts).

Pour la partie contrôleur (solution papier):

- (25 pts) Compléter la table de vérité de l'unité de contrôle pour incorporer les nouvelles instructions.

### **Notations:**

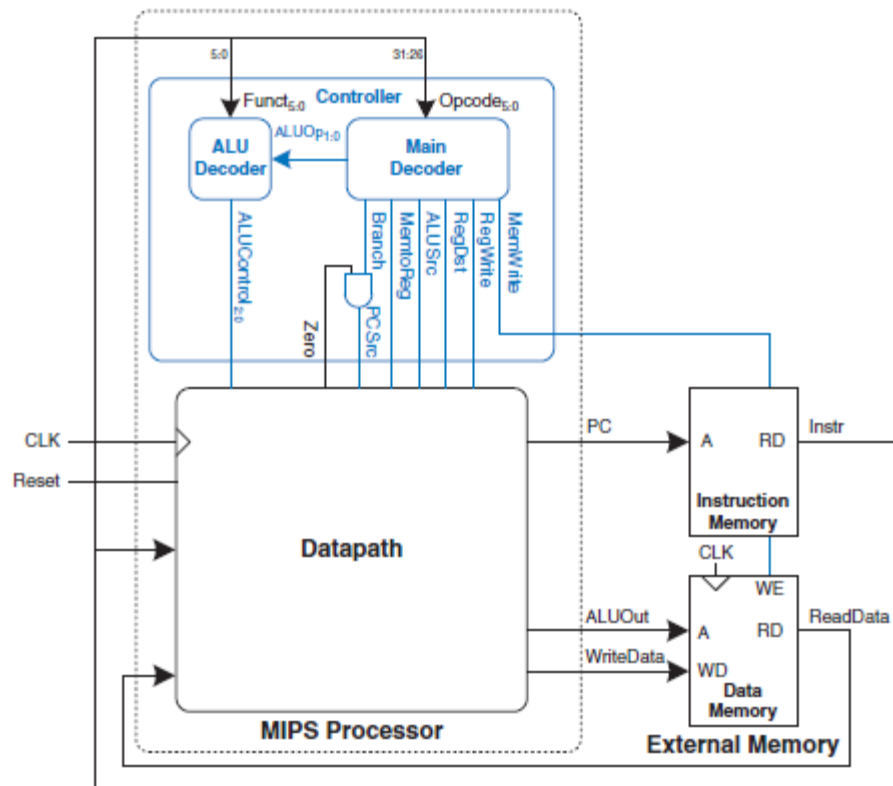
- `[reg]` – Contenu du registre
- `imm` – Le champ Immediate de 16 bits d'instructions de type I
- `addr` – Le champ représentant une partie de l'adresse sur 26 bits de l'instruction de type J
- `SignImm` – la valeur `imm` étendue par le bit le plus significatif de `imm` représentant une signe sur 32-bits (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 = [rs] + SignImm`
- `Mem[Address]` – contenu de la mémoire à l'adresse `Address`
- `BTA - branch target address = PC + 4 + (SignImm << 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 et 2 bits zéros (positions 1 et 0)
- `Label` – le texte qui indique une localisation de l'instruction

### Les détails des instructions à implémenter :

Code d'opération	Nom	Description	Opération	funct
000000 (0)	sll rd, rt, shamt	Shift left logical	$[rd] = [rt] \ll \text{shamt}$	000000 (0)
000000 (0)	slt rd, rs, rt	Set Less Than	$[rs] < [rt] ? [rd] = 1 : [rd] = 0$	101010 (2A)
000000 (0)	sltu rd, rs, rt	Set Less Than Unsigned	$[rs] < [rt] ? [rd] = 1 : [rd] = 0$	101011 (2B)
000000 (0)	jr rs	Jump register	$PC = [rs]$	001000 (08)
001100 (C)	andi rt, rs, imm	And Immediate	$[rt] = [rs] \& \text{ZeroImm}$	-
000011 (3)	jal label	jump and link	$[\$ra] = PC + 4, PC = JTA$	-
001101 (D)	ori rt, rs, imm	Or Immediate	$[rt] = [rs] \mid \text{ZeroImm}$	-
001111 (F)	lui rt, rs, imm	Load upper immediate	$[rt] = \{\text{imm}, 16'b0\}$	-
010001 (17)	indexIntAdr rd, rs, rt	Index to Address	$[rd] = [rs] * 4 + [rt]$	-

- d) (20 pts) Modifier le code VHDL du processeur MIPS initial (disponible sur le site du cours) pour incorporer les neuf instructions spécifiées plus haut et 2 instructions présentées en classe (andi et jr). Remplacer le module **alu** initial par votre version de l'**alu** et adapter le code du module **aludec**.

Le schéma structurel du code VHDL est présenté plus bas.



- e) (10 pts) Remplacer le code test en langage machine MIPS (module `imem`) par le code machine du programme calculant `factorial(6)` en sauvegardant le résultat de cet appel à l'adresse `0xFFFF0010` (vu qu'on considère seulement les 8 derniers bits pour une adresse, l'écriture aura lieu à l'adresse `0x10` dans notre mémoire de données). Essayez d'implémenter le code en utilisant les 8 instructions nouvellement ajoutées pour pouvoir les tester. Il ne faut pas utiliser les instructions non implémentées.

Votre programme doit implémenter les fonctionnalités suivantes :

- Fonction `main` :

Fait l'appel `factorial(6)` et sauvegarde le résultat de cet appel à l'adresse `0xFFFF0010`

Simuler votre processeur en observant le résultat d'exécution des instructions spécifiées plus haut.

Présentation : 5 pts

### À remettre.

- Le rapport : le schéma, la table de vérité de l'unité de contrôle, le programme test contenant le code assembleur et le code machine (voir page 437 (428) de votre livre pour un exemple).

Pour la remise papier du rapport: scanner les copies papier et faire la remise électronique des documents scannés par StudiUM ou faites la remise papier au démonstrateur/professeur.

- Le projet compressé dans un seul fichier contenant tous les fichiers sources VHDL dans le répertoire **files** pour qu'on puisse simuler et vérifier votre processeur.

[illegible]

