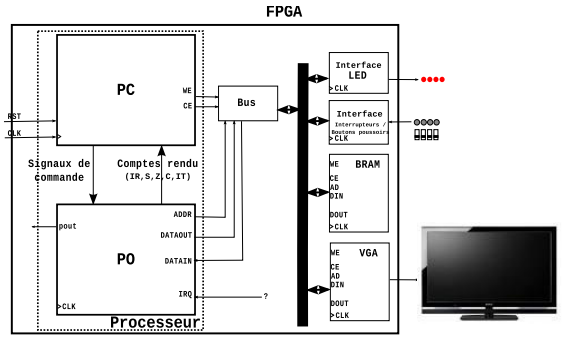
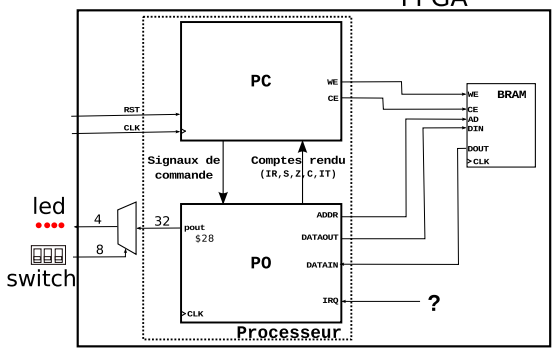
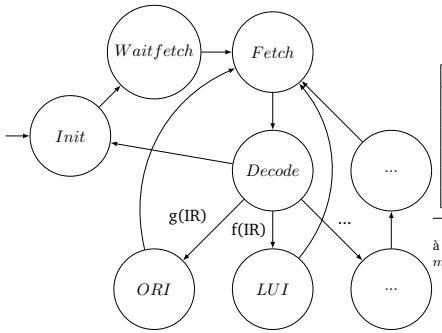
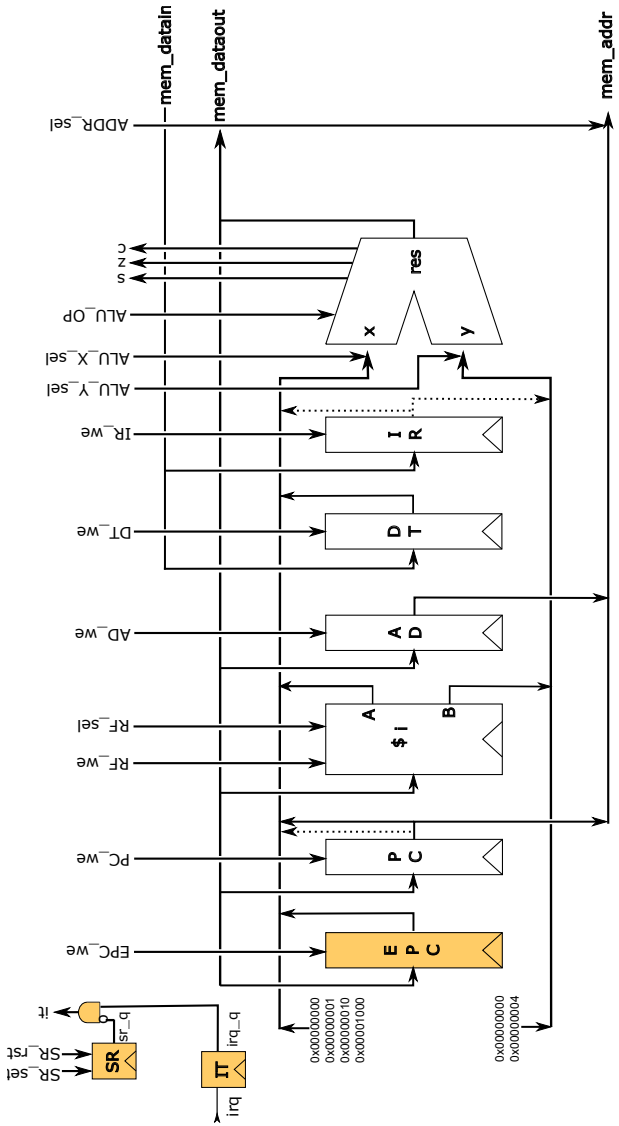






États	Opérations entre registres (RTL)
Init	$PC \leftarrow 0x0$
Waitfetch	$mem[PC]$[a]
Fetch	$IR \leftarrow mem\_datain$
Decode	$PC \leftarrow PC + 4$
ORI	$RT \leftarrow 0^{16} \| IR_{15...0}$ OR RS ; $mem[PC]$
LUI	$RT \leftarrow IR_{15...0} \| 0^{16}$; $mem[PC]$

[a] Un accès mémoire nécessite un cycle. On demande un accès à la valeur $mem[PC]$ qui sera fournit au cycle suivant sur le bus mem_datain.



0.1 Interface à la PO

Les signaux de commandes de la PO sont regroupés dans une structure de type `MMIPS_PO_cmd`, définie dans le fichier `MMIPS_pkg.vhd`. Voici les différents champs de cette structure :

Champ	Type VHDL	Rôle
ALU_X_Sel	UXS_type	Sélection de l'opérande X sur l'ALU
ALU_Y_Sel	UYS_type	Sélection de l'opérande Y sur l'ALU
ALU_OP	AO_type	Sélection de l'opération effectuée par l'ALU
ALU_extension_signe	std_logic	'1' si les opérations arithmétiques sont effectuées avec extension de signe sur 33 bits, '0' sinon.
RF_Sel	RF_sel_type	Sélection du numéro de registre destination
RF_we	boolean	Valide l'écriture dans RF
EPC_we	boolean	Valide l'écriture dans EPC
PC_we	boolean	Valide l'écriture dans PC
AD_we	boolean	Valide l'écriture dans AD
DT_we	boolean	Valide l'écriture dans DT
IR_we	boolean	Valide l'écriture dans IR
ADDR_sel	ADDR_select	Sélection de l'adresse vers la mémoire
mem_we	boolean	Valide une écriture dans la mémoire
mem_ce	boolean	Valide une transaction vers la mémoire (lecture ou écriture)

Les types utilisés dans cette structure sont également définis dans le fichier `MMIPS_pkg.vhd` comme spécifié ci-dessous :

`UXS_type` est le type énuméré utilisé pour sélectionner la valeur à fournir sur l'opérande X de l'UAL.

Valeur	Sémantique
UXS_RF_RS	Port A du banc de registre pointé par $IR_{25...21}$ (RS)
UXS_PC	Registre PC
UXS_EPC	Registre EPC
UXS_DT	Registre DT
UXS_cst_x00	Constante 0x00000000
UXS_cst_x01	Constante 0x00000001
UXS_cst_x10	Constante 0x00000010
UXS_IT_vec	Constante 0x00001FFC
UXS_PC_up	$PC_{31...28} \parallel 0^{28}$
UXS_IR_SH	$0^{27} \parallel IR_{10...6}$

`UYS_type` est le type énuméré utilisé pour sélectionner la valeur à fournir sur l'opérande Y de l'UAL.

Valeur	Sémantique
UYS_IR_imm16	$0^{16} \parallel IR_{15...0}$
UYS_IR_imm16_ext	$IR_{15}^{16} \parallel IR_{15...0}$
UYS_IR_imm16_ext_up	$IR_{15}^{14} \parallel IR_{15...0} \parallel 0^2$
UYS_IR_imm26	$0^4 \parallel IR_{25...0} \parallel 0^2$
UYS_RF_RT	Port B du banc de registre pointé par $IR_{20...16}$ (RT)
UYS_cst_x00	Constante 0x00000000
UYS_cst_x04	Constante 0x00000004

`RF_sel_type` est le type énuméré utilisé pour sélectionner le registre de destination.

Valeur	Sémantique
RFS_RD	$IR_{15...11}$ (RD)
RFS_RT	$IR_{20...16}$ (RT)
RFS_31	registre R31

`AO_type` est le type énuméré utilisé pour sélectionner l'opération à réaliser par l'UAL.

Valeur	Sémantique
AO_plus	$RES \Leftarrow (ext(X) \parallel X) + (ext(Y) \parallel Y)$
AO_moins	$RES \Leftarrow (ext(X) \parallel X) - (ext(Y) \parallel Y)$
AO_and	$RES \Leftarrow X and Y$
AO_or	$RES \Leftarrow X or Y$
AO_xor	$RES \Leftarrow X \oplus Y$
AO_nor	$RES \Leftarrow X nor Y$
AO_SLL	$RES \Leftarrow Y << X_{4...0}$ (logique)
AO_SRL	$RES \Leftarrow Y >> X_{4...0}$ (logique)
AO_SRA	$RES \Leftarrow Y >>> X_{4...0}$ (arithmétique)

La fonction $ext(a)$ étend le bit de signe ou non selon le signal `ALU_extension_signe`.

`ADDR_select` est le type énuméré utilisé pour la sélection de l'origine de l'adresse vers la mémoire.

Valeur	Sémantique
ADDR_from_PC	Registre PC
ADDR_from_AD	Registre AD

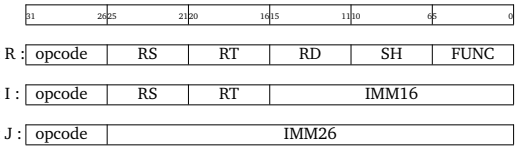
La PO retourne un ensemble de signaux d'états (*status*), regroupés dans la structure `status` de type `MMIPS_PO_status`. Les différents champs sont les suivants :

Champ	Type VHDL	Valeur
IR	w32	L'instruction en cours
s	boolean	Le bit de signe du résultat de l'ALU (bit 31)
c	boolean	Le bit de retenue du résultat de l'ALU (bit 32)
z	boolean	true si le résultat de l'ALU vaut 0, false sinon

Le type `w32` est un vecteur de 32 bits.

0.2 Codage des instructions

Les 3 formats d'instruction



Champ **opcode**

31...29 \ 28...26	000	001	010	011	100	101	110	111
000	**special**	**regimm**	J	JAL	BEQ	BNE	BLEZ	BGTZ
001	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
010	**cop0**	-	-	-	-	-	-	-
011	-	-	-	-	-	-	-	-
100	LB	LH	-	LW	LBU	LHU	-	-
101	SB	SH	-	SW	-	-	-	-
11X	-	-	-	-	-	-	-	-

Champ **FUNC**, lorsque l'**opcode** vaut **special**.

5...3 \ 2...0	000	001	010	011	100	101	110	111
000	SLL	-	SRL	SRA	SLLV	-	SRLV	SRAV
001	JR	JALR	-	-	SYSCALL	BREAK	-	-
010	MFHI	MTHI	MFLO	MTLO	-	-	-	-
011	MULT	MULTU	DIV	DIVU	-	-	-	-
100	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
101	-	-	SLT	SLTU	-	-	-	-
11X	-	-	-	-	-	-	-	-

Champ **RT**, lorsque l'**opcode** vaut **regimm**.

20...19 \ 18...16	000	001	010	011	100	101	110	111
00	BLTZ	BGEZ	-	-	-	-	-	-
10	BLTZAL	BGEZAL	-	-	-	-	-	-
X1	-	-	-	-	-	-	-	-

Champ **RS**, lorsque l'**opcode** vaut **cop0**.
00000 : MFC0 , 00100 : MTC0
Champ **FUNC**, lorsque l'**opcode** vaut **cop0**.
011000 : ERET