Instructions MIPS

	Assembleur		Opération		Effet	Format
Arithmétiques/logiques	Add	Rd, Rs, Rt	Add	*Overflow detection*	Rd<-Rs+Rt	R
	Sub	Rd, Rs, Rt	Substract	*Overflow detection*	Rd<-Rs-Rt	R
	Addu	Rd, Rs, Rt	Add	*No Overflow*	Rd<-Rs+Rt	R
	Subu	Rd, Rs, Rt	Substract	*No Overflow*	Rd<-Rs-Rt	R
	Addi	Rt, Rs, I	Add Immediate	*Overflow detection*	Rt<-Rs+I	I
	Addiu	Rt, Rs, I	Add Immediate	*No Overflow*	Rt<-Rs+I	I
	Or	Rd, Rs, Rt	Logical Or		Rd<-Rs or Rt	R
	And	Rd, Rs, Rt	Logical And		Rd<-Rs and Rt	R
	Xor	Rd, Rs, Rt	Logical Exclusive-Or		Rd<-Rs xor Rt	R
	Nor	Rd, Rs, Rt	Logical Not Or		Rd<-Rs nor Rt	R
	Ori	Rt, Rs, I	Or Immediate	*Unsigned immediate*	Rt<-Rs or I	I
	Andi	Rt, Rs, I	And Immediate	*Unsigned immediate*	Rt<-Rs and I	I
	Xori	Rt, Rs, I	Exclusive-Or Immediate	*Unsigned immediate*	Rt<-Rs xor I	I
	Sllv	Rd, Rt, Rs	Shitf Left Logical Variable	*5 lsb of Rs is significant*	Rd<-Rt<<Rs	R
	Srlv	Rd, Rt, Rs	Shitf Right Logical Variable	*5 lsb of Rs is significant*	Rd<-Rt>>Rs	R
	Srav	Rd, Rt, Rs	Shitf Right Arithmetical Variable	*5 lsb of Rs is significant* **with sign extension*	Rd<-Rt>>*Rs	R
	Sll	Rd, Rt, sh	Shitf Left Logical		Rd<-Rt<<sh	R
	Srl	Rd, Rt, sh	Shitf Right Logical		Rd<-Rt>>sh	R
	Sra	Rd, Rt, sh	Shitf Right Arithmetical	**with sign extension*	Rd<-Rt>>*sh	R
	Lui	Rt, I	Load Upper Immediate	*16 lowers bits of Rt are set to zero*	Rt<-I||"0000"	I
	Slt	Rd, Rs, Rt	Set if Less Than		Rd<-1 if Rs<Rt else 0	R
	Sltu	Rd, Rs, Rt	Set if Less Than Unsigned		Rd<-1 if Rs<Rt else 0	R
	Slti	Rt, Rs, I	Set if Less Than Immediate	*Sign extended Immediate*	Rt<-1 if Rs<I else 0	I
	Sltiu	Rt, Rs, I	Set if Less Than Immediate	*Unsigned Immediate*	Rt<-1 if Rs<I else 0	I
	Mult	Rs, Rt	Multiply	*LO<-32 low significant bits* *HI<-32 high significant bits*	Rs*Rt	R
	Multu	Rs, Rt	Multiply Unsigned	*LO<-32 low significant bits* *HI<-32 high significant bits*	Rs*Rt	R
	Div	Rs, Rt	Divide	*LO<-Quotient* *HI<-Remainder*	Rs/Rt	R
	Divu	Rs, Rt	Divide Unsigned	*LO<-Quotient* *HI<-Remainder*	Rs/Rt	R
HI, LO	Mfhi	Rd	Move From HI		Rd<-HI	R
	Mflo	Rd	Move From LO		Rd<-LO	R
	Mthi	Rs	Move To HI		HI<-Rs	R
	Mtlo	Rs	Move To LO		LO<-Rs	R

Lecture/écriture mémoire	Lw	Rt, I(Rs)	Load Word	*Sign extended immediate*	Rt<-M(Rs+I)	I
	Sw	Rt, I(Rs)	Store Word	*Sign extended immediate*	M(Rs+I)<-Rt	I
	Lh	Rt, I(Rs)	Load Half Word	*Sign extended immediate. Two bytes from storage are located into the 2 less significant bytes of Rt. The sign of these 2 bytes is extended on the 2 most significant bytes.*	Rt<-M(Rs+I)	I
	Lhu	Rt, I(Rs)	Load Half Word Unsigned	*Sign extended immediate. Two bytes from storage are located into the 2 less significant bytes of Rt, others bytes are set to zero.*	Rt<-M(Rs+I)	I
	Sh	Rt, I(Rs)	Store Half Word	*Sign extended immediate/. The two less significant bytes of Rt are stored into the storage.*	M(Rs+I)<-Rt	I
	Lb	Rt, I(Rs)	Load Byte	*Sign extended immediate. One byte from storage is located into the less significant bytes of Rt. The sign of this byte is extended on the 3 most significant bytes.*	Rt<-M(Rs+I)	I
	Lbu	Rt, I(Rs)	Load Byte Unsigned	*Sign extended immediate. One byte from storage is located into the less significant bytes of Rt, others bytes are set to zero.*	Rt<-M(Rs+I)	I
	Sb	Rt, I(Rs)	Store Byte	*Sign extended immediate. The less significant byte of Rt is stored into the storage.*	M(Rs+I)<-Rt	I
Branchements	Beq	Rs, Rt, label	Branch if EQual		PC<-PC+4+(I*4) if Rs=Rt PC<-PC+4 if Rs!=Rt	I
	Bne	Rs, Rt, label	Branch if Not Equal		PC<-PC+4+(I*4) if Rs!=Rt PC<-PC+4 if Rs=Rt	I
	Bgez	Rs, label	Branch if Greater or Equal Zero		PC<-PC+4+(I*4) if Rs>=0 PC<-PC+4 if Rs<0	I
	Bgtz	Rs, label	Branch if Greater Than Zero		PC<-PC+4+(I*4) if Rs>0 PC<-PC+4 if Rs<=0	I
	Blez	Rs, label	Branch if Less or Equal Zero		PC<-PC+4+(I*4) if Rs<=0 PC<-PC+4 if Rs>0	I
	Bltz	Rs, label	Branch if Less Than Zero		PC<-PC+4+(I*4) if Rs<0 PC<-PC+4 if Rs>=0	I
	Bgezal	Rs, label	Branch if Greater or Equal Zero And Link		PC<-PC+4+(I*4) if Rs>=0 PC<-PC+4 if Rs<0 R31<-PC+4 in both cases	I
	Bltzal	Rs, label	Branch if Greater Than Zero And Link		PC<-PC+4+(I*4) if Rs<0 PC<-PC+4 if Rs>=0 R31<-PC+4 in both cases	I
	J	Label	Jump		PC<-PC 31:28||I*4	J
	Jal	Label	Jump and Link		R31<-PC+4 PC<-PC 31:28||I*4	J
	Jr	Rs	Jump Register		PC<-Rs	R
	Jalr	Rs	Jump and Link Register		R31<-PC+4 PC<-Rs	R
	Jalr	Rd, Rs	Jump and Link Register		Rd<-PC+4 PC<-Rs	R