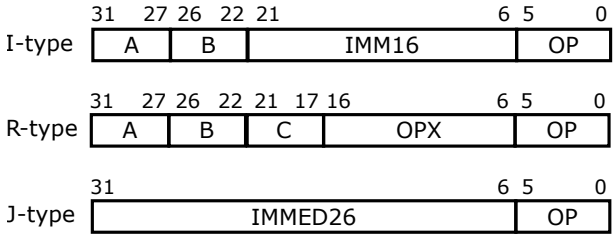
1 Registers

The following table lists the conventional function and names[1] of the 32 registers of the **Register File**.

Register	Name	Function
r0	zero	0x00000000
r1	at	Assembler Temporary
r2	v0	Return Value
r3	v1	Return Value
r4	a0	Register Arguments
r5	a1	Register Arguments
r6	a2	Register Arguments
r7	a3	Register Arguments
r8	t0	Temporary Register
r9	t1	Temporary Register
r10	t2	Temporary Register
r11	t3	Temporary Register
r12	t4	Temporary Register
r13	t5	Temporary Register
r14	t6	Temporary Register
r15	t7	Temporary Register

Register	Name	Function
r16	s0	Saved Register
r17	s1	Saved Register
r18	s2	Saved Register
r19	s3	Saved Register
r20	s4	Saved Register
r21	s5	Saved Register
r22	s6	Saved Register
r23	s7	Saved Register
r24	et	Exception Temporary
r25	bt	Breakpoint Temporary
r26	gp	Global Pointer
r27	sp	Stack Pointer
r28	fp	Frame Pointer
r29	ea	Exception Return Address
r30	ba	Breakpoint Return Address
r31	ra	Return Address

2 Instruction Encoding



3 Instructions

The following table lists the instructions implemented by a Nios II processor. We write "$\texttt{rA}_s$" to consider the rA as signed, and "$\texttt{rA}_u$" when it is unsigned. We write "$\texttt{imm}_s$", when the immediate value is signed extend, and "$\texttt{imm}_u$", when the immediate value is unsigned (i.e., extended with zeros).

[1] To improve the readability of the code, we extend the official Nios II registers naming: we added names to the registers r2 to r23, which were unnamed. These names are only supported by the **Nios2Sim** simulator.

Category	OP/OPX/TYPE	Instruction				Meaning
Arithmetic	0x04/-/I	**addi**	rB,	rA,	imm	rB ← rA + imm_s
	0x3a/0x31/R	**add**	rC,	rA,	rB	rC ← rA + rB
	0x3a/0x39/R	**sub**	rC,	rA,	rB	rC ← rA − rB
Logical	0x3a/0x0e/R	**and**	rC,	rA,	rB	rC ← rA *and* rB
	0x0c/-/I	**andi**	rB,	rA,	imm	rB ← rA *and* imm_u
	0x3a/0x16/R	**or**	rC,	rA,	rB	rC ← rA *or* rB
	0x14/-/I	**ori**	rB,	rA,	imm	rB ← rA *or* imm_u
	0x3a/0x1e/R	**xor**	rC,	rA,	rB	rC ← rA *xor* rB
	0x1c/-/I	**xori**	rB,	rA,	imm	rB ← rA *xor* imm_u
	0x3a/0x06/R	**nor**	rC,	rA,	rB	rC ← rA *nor* rB
Comparator	0x08/-/R	**cmpgei**	rB,	rA,	imm	rB ← (rA ≥ imm_s)? 1 : 0
	0x10/-/I	**cmplti**	rB,	rA,	imm	rB ← (rA < imm_s)? 1 : 0
	0x18/-/I	**cmpnei**	rB,	rA,	imm	rB ← (rA ≠ imm_s)? 1 : 0
	0x20/-/I	**cmpeqi**	rB,	rA,	imm	rB ← (rA = imm_s)? 1 : 0
	0x28/-/I	**cmpgeui**	rB,	rA,	imm	rB ← (rA_u ≥ imm_u)? 1 : 0
	0x30/-/I	**cmpltui**	rB,	rA,	imm	rB ← (rA_u < imm_u)? 1 : 0
	0x3a/0x08/R	**cmpge**	rC,	rA,	rB	rC ← (rA ≥ rB)? 1 : 0
	0x3a/0x10/R	**cmplt**	rC,	rA,	rB	rC ← (rA < rB)? 1 : 0
	0x3a/0x18/R	**cmpne**	rC,	rA,	rB	rC ← (rA ≠ rB)? 1 : 0
	0x3a/0x20/R	**cmpeq**	rC,	rA,	rB	rC ← (rA = rB)? 1 : 0
	0x3a/0x28/R	**cmpgeu**	rC,	rA,	rB	rC ← (rA_u ≥ rB_u)? 1 : 0
	0x3a/0x30/R	**cmpltu**	rC,	rA,	rB	rC ← (rA_u < rB_u)? 1 : 0
Shift	0x3a/0x13/R	**sll**	rC,	rA,	rB	rC ← rA ≪ $\text{rB}_{4..0}$
	0x3a/0x12/R	**slli**	rC,	rA,	imm	rC ← rA ≪ $\text{imm}_{4..0}$
	0x3a/0x1b/R	**srl**	rC,	rA,	rB	rC ← rA_u ≫ $\text{rB}_{4..0}$
	0x3a/0x1a/R	**srli**	rC,	rA,	imm	rC ← rA_u ≫ $\text{imm}_{4..0}$
	0x3a/0x3b/R	**sra**	rC,	rA,	rB	rC ← rA_s ≫ $\text{rB}_{4..0}$
	0x3a/0x3a/R	**srai**	rC,	rA,	imm	rC ← rA_s ≫ $\text{imm}_{4..0}$
	0x3a/0x03/R	**rol**	rC,	rA,	rB	rC ← rA *rol* $\text{rB}_{4..0}$
	0x3a/0x0b/R	**ror**	rC,	rA,	rB	rC ← rA *ror* $\text{rB}_{4..0}$
	0x3a/0x02/R	**roli**	rC,	rA,	imm	rC ← rA *rol* $\text{imm}_{4..0}$
Memory	0x07/-/I	**ldb**	rB,	imm	(rA)	$\text{rB}_{7..0}$ ← MEM[imm_s+rA]$_{7..0}$
	0x17/-/I	**ldw**	rB,	imm	(rA)	rB ← MEM[imm_s+rA]
	0x05/-/I	**stb**	rB,	imm	(rA)	MEM[imm_s+rA]$_{7..0}$ ← $\text{rB}_{7..0}$
	0x15/-/I	**stw**	rB,	imm	(rA)	MEM[imm_s+rA] ← rB
Branch	0x06/-/I	**br**	imm			goto PC+4+imm_s
	0x0e/-/I	**bge**	rA,	rB,	imm	if (rA ≥ rB) goto PC+4+imm_s
	0x16/-/I	**blt**	rA,	rB,	imm	if (rA < rB) goto PC+4+imm_s
	0x1e/-/I	**bne**	rA,	rB,	imm	if (rA ≠ rB) goto PC+4+imm_s
	0x26/-/I	**beq**	rA,	rB,	imm	if (rA = rB) goto PC+4+imm_s
	0x2e/-/I	**bgeu**	rA,	rB,	imm	if (rA_u ≥ rB_u) goto PC+4+imm_s
	0x36/-/I	**bltu**	rA,	rB,	imm	if (rA_u < rB_u) goto PC+4+imm_s
Jump	0x00/-/J	**call**	imm			goto imm ≪ 2; ra ← PC+4
	0x3a/0x1d/R	**callr**	rA			goto rA; ra ← PC+4
	0x3a/0x05/R	**ret**				goto ra
	0x3a/0x0d/R	**jmp**	rA			goto rA
	0x01/-/J	**jmpi**	imm			goto imm ≪ 2