

## Instruções do processador DLX

### *Instruções de transferência de dados*

LB	Rd, Adr	Load byte (sign extension)
LBU	Rd, Adr	Load byte (unsigned)
LH	Rd, Adr	Load halfword (sign extension)
LHU	Rd, Adr	Load halfword (unsigned)
LW	Rd, Adr	Load word
LF	Fd, Adr	Load single-precision Floating point
LD	Dd, Adr	Load double-precision Floating point
SB	Adr, Rs	Store byte
SH	Adr, Rs	Store halfword
SW	Adr, Rs	Store word
SF	Adr, Fs	Store single-precision Floating point
SD	Adr, Fs	Store double-precision Floating point
MOVI2FP	Fd, Rs	Move 32 bits from integer registers to FP registers
MOVI2FP	Rd, Fs	Move 32 bits from FP registers to integer registers
MOVF	Fd, Fs	Copy one Floating point register to another register
MOVD	Dd, Ds	Copy a double-precision pair to another pair
MOVI2S	SR, Rs	Copy a register to a special register (not impl. in WinDLX)
MOVS2I	Rs, SR	Copy a special register to a GPR (not impl. in WinDLX)

### *Operações aritméticas e lógicas*

ADD	Rd, Ra, Rb	Add
ADDI	Rd, Ra, Imm	Add immediate (all immediates are 16 bits)
ADDU	Rd, Ra, Rb	Add unsigned
ADDUI	Rd, Ra, Imm	Add unsigned immediate
SUB	Rd, Ra, Rb	Subtract
SUBI	Rd, Ra, Imm	Subtract immediate
SUBU	Rd, Ra, Rb	Subtract unsigned
SUBUI	Rd, Ra, Imm	Subtract unsigned immediate
MULT	Rd, Ra, Rb	Multiply signed
MULTU	Rd, Ra, Rb	Multiply unsigned
DIV	Rd, Ra, Rb	Divide signed
DIVU	Rd, Ra, Rb	Divide unsigned
AND	Rd, Ra, Rb	And
ANDI	Rd, Ra, Imm	And immediate
OR	Rd, Ra, Rb	Or
ORI	Rd, Ra, Imm	Or immediate
XOR	Rd, Ra, Rb	Xor
XORI	Rd, Ra, Imm	Xor immediate
LHI	Rd, Imm	LOAD high immediate - loads upper half of Rd with Imm
SLL	Rd, Rs, Rc	Shift left logical
SRL	Rd, Rs, Rc	Shift right logical

SRA	Rd, Rs, Rc	Shift right arithmetic
SLLI	Rd, Rs, Imm	Shift left logical 'immediate' bits
SRLI	Rd, Rs, Imm	Shift right logical 'immediate' bits
SRAI	Rd, Rs, Imm	Shift right arithmetic 'immediate' bits
S__	Rd, Ra, Rb	Set conditional: "__" may be EQ, NE, LT, GT, LE or GE
S__I	Rd, Ra, Imm	Set conditional immediate
S__U	Rd, Ra, Rb	Set conditional unsigned
S__UI	Rd, Ra, Imm	Set conditional unsigned immediate
NOP		No operation

### *Instruções de controlo*

BEQZ	Rt, Dest	Branch if GPR equal to zero; 16-bit offset from PC
BNEZ	Rt, Dest	Branch if GPR not equal to zero; 16-bit offset from PC
BFPT	Dest	Test comparison bit in the FP status register (true) and branch; 16-bit offset from PC
BFPF	Dest	Test comparison bit in the FP status register (false) and branch; 16-bit offset from PC
J	Dest	Jump: 26-bit offset from PC
JR	Rx	Jump: target in register
JAL	Dest	Jump and link: save PC+4 to R31; target is PC-relative
JALR	Rx	Jump and link: save PC+4 to R31; target is a register
TRAP	Imm	Transfer to operating system at a vectored address; see Traps.
RFE	Dest	Return to user code from an exception; restore user mode

### *Operações em vírgula flutuante*

ADDD	Dd, Da, Db	Add double-precision numbers
ADDF	Fd, Fa, Fb	Add single-precision numbers
SUBD	Dd, Da, Db	Subtract double-precision numbers
SUBF	Fd, Fa, Fb	Subtract single-precision numbers.
MULTD	Dd, Da, Db	Multiply double-precision Floating point numbers
MULTF	Fd, Fa, Fb	Multiply single-precision Floating point numbers
DIVD	Dd, Da, Db	Divide double-precision Floating point numbers
DIVF	Fd, Fa, Fb	Divide single-precision Floating point numbers
CVTF2D	Dd, Fs	Converts from type single-precision to type double-precision
CVTD2F	Fd, Ds	Converts from type double-precision to type single-precision
CVTF2I	Fd, Fs	Converts from type single-precision to type integer
CVTI2F	Fd, Fs	Converts from type integer to type single-precision
CVTD2I	Fd, Ds	Converts from type double-precision to type integer
CVTI2D	Dd, Fs	Converts from type integer to type double-precision
__D	Da, Db	Double-precision compares: "__" may be EQ, NE, LT, GT, LE or GE; sets comparison bit in FP status register
__F	Fa, Fb	Single-precision compares: "__" may be EQ, NE, LT, GT, LE or GE; sets comparison bit in FP status register