Integer arithmetic

mnemonic	description	operands	notes
add	add 32 bit integers	dreg <= reg1, reg2	
addi		dreg <= reg, imm	
addiu			
addu		dreg <= reg1, reg2	
mulhi	multiply 32 bit integers	dreg <= intermediate	Upper 32 bits of a 64 bit result
muli		dreg <= reg, imm	
muls		dreg <= reg1, reg2	
mulu			
mulus			
muls_16	multiply 16 bit integers		
mulu_16			
mulus_16			
sub	subtract 32 bit integers		
subu			

Byte and bitfield manipulation

mnemonic	description	operands	notes
exb	extract byte from word	dreg <= reg, imm	
exbf	extract bitfield from word	dreg <= reg1, reg2	
exbfi		dreg <= reg, imm	32 bit version only Not allowed to be executed conditionally
exh	extract halfword from word		
lli	Load lower/upper halfword with immediate value	dreg <= imm	32 bit version only Not allowed to be executed conditionally
lui		dreg <= reg, imm	
sext	Sign extend an integer	dreg <= reg1, reg2	
sexti		dreg <= reg, imm	
conb	Concatenate bytes/halfwords	dreg <= reg1, reg2	
conh			

Boolean bitwise operations

mnemonic	description	operands	notes
and	bitwise and	dreg <= reg1, reg2	
andi		dreg <= reg, imm	
not	bitwise not	dreg <= reg	
or	bitwise or	dreg <= reg1, reg2	
ori		dreg <= reg, imm	
xor	bitwise xor	dreg <= reg1, reg2	

Conditional jumps (branches)

mnemonic	description	operands	notes
bc	Branch if condition is true.	pc <= pc, imm	Pre-evaluated flags from one of the eight condition registers are used to evaluate condition.
begt			
belt			
beq			
bgt			
blt			
bnc			
bne			

Other jumps

mnemonic	description	operands	notes
jal	jump and save link address	pc <= pc, imm dreg <= pc + increment	Not allowed to be executed conditionally.
jalr		pc <= reg dreg <= pc + increment	
jmp	jump	pc <= pc, imm	Not allowed to be executed conditionally
jmpr		pc <= reg	

Integer comparison

mnemonic	description	operands	notes
cmp	Compare and evaluate condition flags.	creg <= reg1, reg2	Not allowed to be executed conditionally.
cmpi		creg <= reg, imm	

Shifts

mnemonic	description	operands	notes
sll	logical shift left	dreg <= reg1, reg2	Only left shift produces flags
slli		dreg <= reg, imm	
sra	arithmetic shift right	dreg <= reg1, reg2	
srai		dreg <= reg, imm	
srl	logical shift right	dreg <= reg1, reg2	
srli		dreg <= reg, imm	

Memory load and store & data moving

mnemonic	description	operands	notes
ld	load a word from memory	dreg <= mem[reg + imm]	Address does not have to be aligned to word boundary. Usage of bits 0 to 1 depend on implementation.
st	store a word to memory	mem[reg1 + imm] <= reg2	
mov	move a word from register to register.	dreg <= reg	

Coprocessor instructions

mnemonic	description	operands	notes
cop	coprosessor instruction	cop <= imm	32 bit version only Not allowed to be executed conditionally
movfc	mov data from coprocessor	dreg <= cop, cpreg	
movtc	mov data to coprocessor	cop <= reg, cpreg	

Mode changing instructions

mnemonic	description	operands	notes
chrs	Change register set to operate with	psr <= imm	Not allowed to be executed conditionally. chrs, di, ei and retu available in super user mode only. Version 1.0 supports to decoding modes: 16 bit ISA and 32 bit ISA.
di	disable interrupts	psr <= IE <= ‘0’	
ei	enable interrupts	psr <= IE <= ‘1’	
swm	switch between decoding modes:	psr <= imm	
reti	return from an interrupt service routine	pc <= hw_stack_addr psr <= hw_stack_psr	
retu	return to user/SPSR defined mode.	pc <= lreg psr <= spsr	These instructions should be used to interface operating system or similar.
scall	system entry	psr <= sys_psr pc <= sys_entry_addr	

Miscellaneous

mnemonic	description	operands	notes
rcon	Restore all condition registers from general purpose register.	creg <= reg	Not allowed to be executed conditionally
scon	Move the contents of all condition registers to a general purpose register	dreg <= creg	
trap	software exception	psr <=	Should be used to catch software exceptions.
nop	no operation, idle		