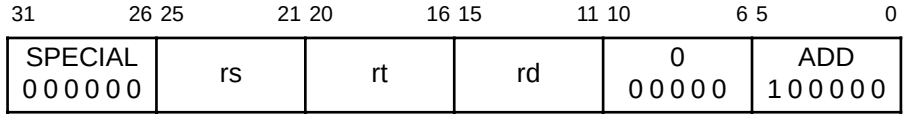
## ADD

Add Word




**Format** : ADD rd, rs, rt

**Fonction** : To add 32-bit integers. If an overflow occurs, then trap.

Description : rd ← rs + rt

The 32-bit word value in GPR rt is added to the 32-bit value in GPR rs to produce a 32-bit result. If the addition results in 32-bit 2's complement arithmetic overflow, the destination register is not modified and an Integer Overflow exception occurs. If the addition does not overflow, the 32-bit result is placed into GPR rd.

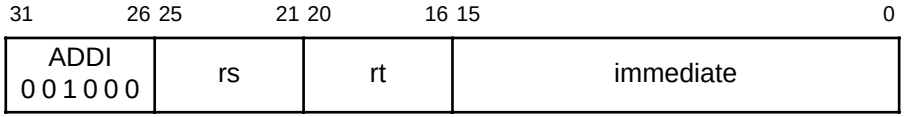
**Restrictions** : None

**Exceptions** : Integer Overflow

**Notes** : ADDU performs the same arithmetic operation but does not trap on overflow.

## ADDI

Add Immediate Word




**Format** : ADDI rt, rs, immediate

**Fonction** : To add a constant to a 32-bit integer. If overflow occurs, then trap.

**Description** : rt ← rs + immediate

The 16-bit signed immediate is added to the 32-bit value in GPR rs to produce a 32-bit result. If the addition results in 32-bit 2's complement arithmetic overflow, the destination register is not modified and an Integer Overflow exception occurs. If the addition does not overflow, the 32-bit result is placed into GPR rt.

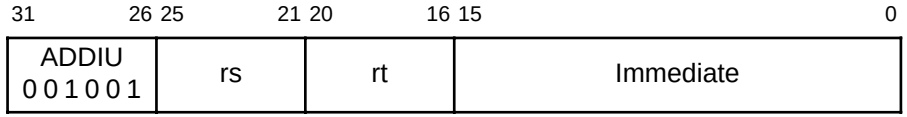
**Restrictions** : None

**Exceptions** : Integer Overflow

**Notes** : ADDIU performs the same arithmetic operation but does not trap on overflow.

## ADDIU

Add Immediate Unsigned Word




**Format** : ADDIU rt, rs, immediate

**Fonction** : To add a constant to a 32-bit integer

**Description** : rt ← rs + immediate

The 16-bit signed immediate is added to the 32-bit value in GPR rs and the 32-bit arithmetic result is placed into GPR rt. No Integer Overflow exception occurs under any circumstances.

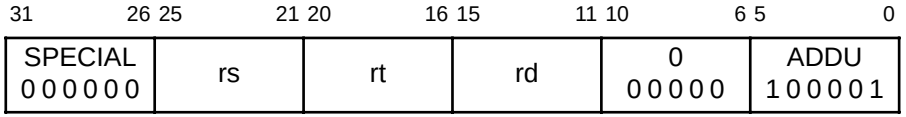
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## ADDU

Add Unsigned Word




**Format** : ADDU rd, rs, rt

**Fonction** : To add 32-bit integers

**Description** : rd ← rs + rt

The 32-bit word value in GPR rt is added to the 32-bit value in GPR rs and the 32-bit arithmetic result is placed into GPR rd. No Integer Overflow exception occurs under any circumstances.

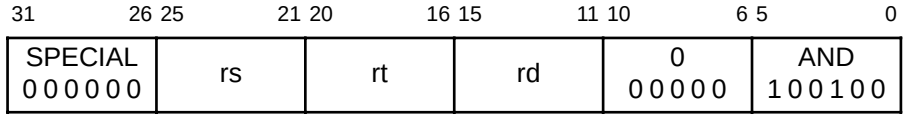
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## AND

And




**Format** : AND rd, rs, rt

**Fonction** : To do a bitwise logical AND

**Description** : rd ← rs AND rt

The contents of GPR rs are combined with the contents of GPR rt in a bit-wise logical AND operation. The result is placed into GPR rd.

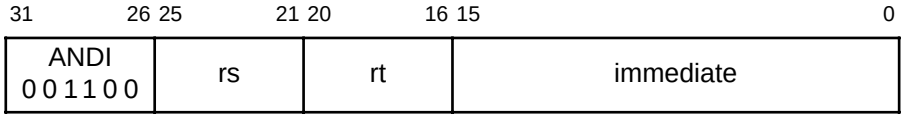
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## ANDI

And Immediate




**Format** : ANDI rt, rs, immediate

**Fonction** : To do a bitwise logical AND with a constant

**Description** : rt ← rs AND immediate

The 16-bit immediate is zero-extended to the left and combined with the contents of GPR rs in a bitwise logical AND operation. The result is placed into GPR rt.

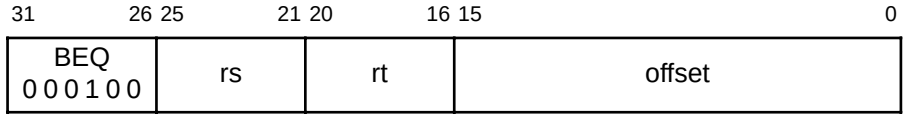
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## BEQ

Branch on Equal




**Format** : BEQ rs, rt, offset

**Fonction** : To compare GPRs then do a PC-relative conditional branch

**Description** : if rs = rt then branch

An 18-bit signed offset (the 16-bit offset field shifted left 2 bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address. If the contents of GPR rs and GPR rt are equal, branch to the effective target address after the instruction in the delay slot is executed.

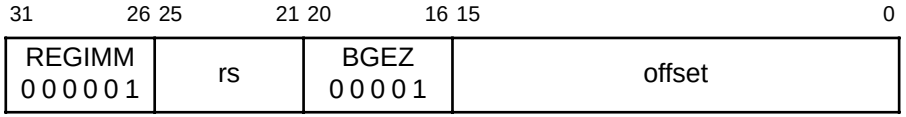
**Restrictions** : None

**Exceptions** : None

**Notes** : With the 18-bit signed instruction offset, the conditional branch range is ± 128 Kbytes. Use jump (J) or jump register (JR) instructions to branch to addresses outside this range.

## BGEZ

Branch on Greater Than or Equal to Zero




**Format** : BGEZ rs, offset

**Fonction** : To test a GPR then do a PC-relative conditional branch

**Description** : if rs >= 0 then branch
An 18-bit signed offset (the 16-bit offset field shifted left 2 bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address. If the contents of GPR rs are greater than or equal to zero (sign bit is 0), branch to the effective target address after the instruction in the delay slot is executed.

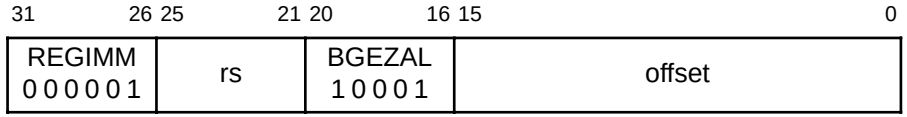
**Restrictions** : None

**Exceptions** : None

**Notes** : With the 18-bit signed instruction offset, the conditional branch range is ± 128 KBytes. Use jump (J) or jump register (JR) instructions to branch to addresses outside this range.

## BGEZAL

Branch on Greater than or Equal to Zero And Link




**Format** : BGEZAL rs, offset

**Fonction** : To test a GPR then do a PC-relative conditional procedure call

**Description** : if rs >= 0 then procedure_call

Place the return address link in GPR 31. The return link is the address of the second instruction following the branch, where execution continues after a procedure call.
An 18-bit signed offset (the 16-bit offset field shifted left 2 bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address.
If the contents of GPR rs are greater than or equal to zero (sign bit is 0), branch to the effective target address after the instruction in the delay slot is executed.

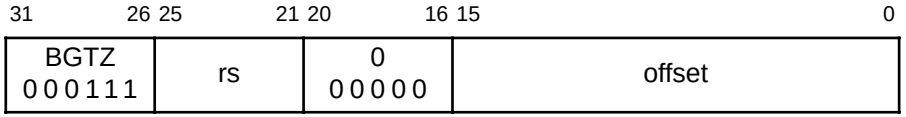
**Restrictions** : GPR 31 must not be used for the source register rs, because such an instruction does not have the same effect when reexecuted. The result of executing such an instruction is UNPREDICTABLE. This restriction permits an exception handler to resume execution by reexecuting the branch when an exception occurs in the branch delay slot.

**Exceptions** : None

**Notes** : With the 18-bit signed instruction offset, the conditional branch range is ± 128 KBytes. Use jump and link (JAL) or jump and link register (JALR) instructions for procedure calls to addresses outside this range.

## BGTZ

Branch on Greater Than Zero




**Format** : BGTZ rs, offset

**Fonction** : To test a GPR then do a PC-relative conditional branch

**Description** : if rs > 0 then branch

An 18-bit signed offset (the 16-bit offset field shifted left 2 bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address. If the contents of GPR rs are greater than zero (sign bit is 0 but value not zero), branch to the effective target address after the instruction in the delay slot is executed.

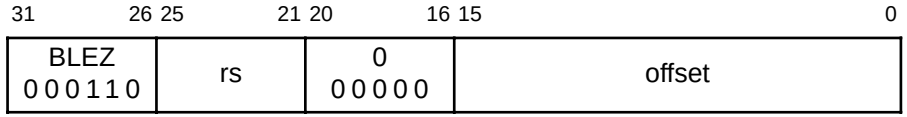
**Restrictions** : None

**Exceptions** : None

**Notes** : With the 18-bit signed instruction offset, the conditional branch range is ± 128 KBytes. Use jump (J) or jump register (JR) instructions to branch to addresses outside this range.

## BLEZ

Branch on Less Than or Equal to Zero




**Format** : BLEZ rs, offset

**Fonction** : To test a GPR then do a PC-relative conditional branch

**Description** : if rs <= 0 then branch

An 18-bit signed offset (the 16-bit offset field shifted left 2 bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address. If the contents of GPR rs are less than or equal to zero (sign bit is 1 or value is zero), branch to the effective target address after the instruction in the delay slot is executed.

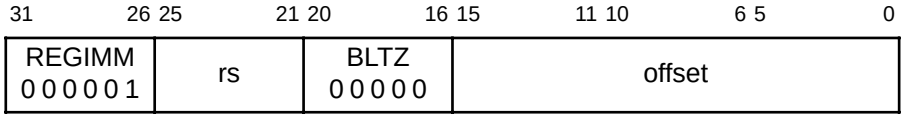
**Restrictions** : None

**Exceptions** : None

**Notes** : With the 18-bit signed instruction offset, the conditional branch range is ± 128 KBytes. Use jump (J) or jump register (JR) instructions to branch to addresses outside this range.

## BLTZ

Branch on Less Than Zero




**Format** : BLTZ rs, offset

**Fonction** : To test a GPR then do a PC-relative conditional branch

**Description** : if rs < 0 then branch

An 18-bit signed offset (the 16-bit offset field shifted left 2 bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address. If the contents of GPR rs are less than zero (sign bit is 1), branch to the effective target address after the instruction in
the delay slot is executed.

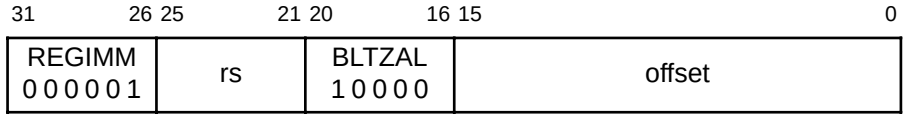
**Restrictions** : None

**Exceptions** : None

**Notes** : With the 18-bit signed instruction offset, the conditional branch range is ± 128 KBytes. Use jump and link (JAL) or jump and link register (JALR) instructions for procedure calls to addresses outside this range.

## BLTZAL

Branch on Less Than Zero And Link




**Format** : BLTZAL rs, offset

**Fonction** : To test a GPR then do a PC-relative conditional procedure call

**Description** : if (rs < 0) then procedure_call

Place the return address link in GPR 31. The return link is the address of the second instruction following the branch, where execution continues after a procedure call. An 18-bit signed offset (the 16-bit offset field shifted left 2 bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address. If the contents of GPR rs are less than zero (sign bit is 1), branch to the effective target address after the instruction in the delay slot is executed.

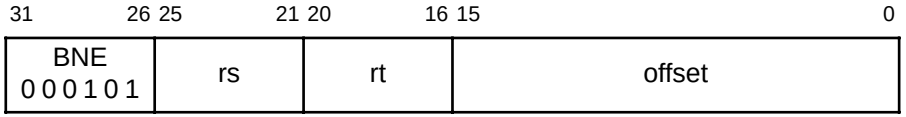
**Restrictions** : GPR 31 must not be used for the source register rs, because such an instruction does not have the same effect when reexecuted. The result of executing such an instruction is UNPREDICTABLE. This restriction permits an exception handler to resume execution by reexecuting the branch when an exception occurs in the branch delay slot.

**Exceptions** : None

**Notes** : With the 18-bit signed instruction offset, the conditional branch range is ± 128 KBytes. Use jump and link (JAL) or jump and link register (JALR) instructions for procedure calls to addresses outside this range.

## BNE

Branch on Not Equal




**Format** : BNE rs, rt, offset

**Fonction** : To compare GPRs then do a PC-relative conditional branch

**Description** : if rs != rt then branch

An 18-bit signed offset (the 16-bit offset field shifted left 2 bits) is added to the address of the instruction following the branch (not the branch itself), in the branch delay slot, to form a PC-relative effective target address. If the contents of GPR rs and GPR rt are not equal, branch to the effective target address after the instruction in the delay slot is executed.

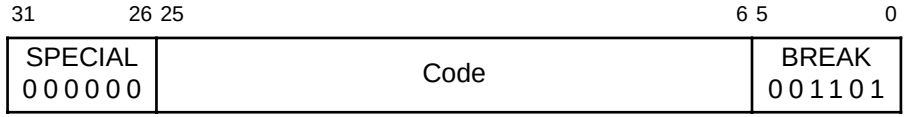
**Restrictions** : None

**Exceptions** : None

**Notes** : With the 18-bit signed instruction offset, the conditional branch range is ± 128 KBytes. Use jump (J) or jump register (JR) instructions to branch to addresses outside this range.

## BREAK

Break Point




**Format** : BREAK

**Fonction** : To cause a Breakpoint exception

**Description** :

A breakpoint exception occurs, immediately and unconditionally transferring control to the exception handler. The code field is available for use as software parameters, but is retrieved by the exception handler only by loading the contents of the memory word containing the instruction.

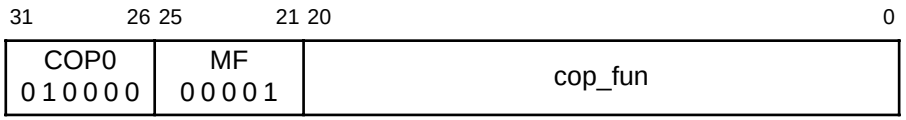
**Restrictions** : None

**Exceptions** : Breakpoint

**Notes** : None

## COP0

Coprocessor system operation




**Format** : COP0 cop_fun

**Fonction** : Execute a coprocessor system function

**Description** :

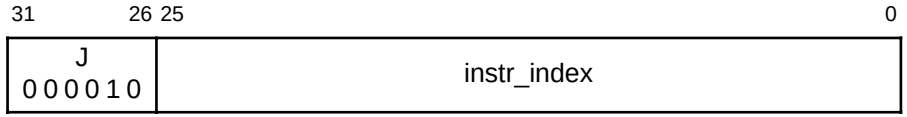
The coprocessor instruction defined by the cop_fun field is done by the coprocessor system. The different instructions are detailed in the coprocessor specifications.

**Restrictions** : None

**Exceptions** : None

**Notes** : None

## J — Jump




**Format** : J target

**Fonction** : To branch within the current 256 MB-aligned region

**Description** :

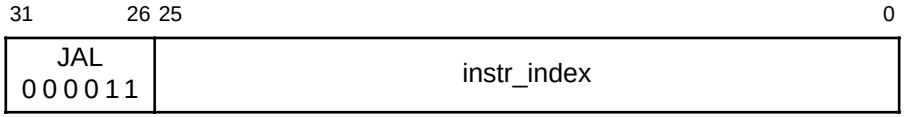
This is a PC-region branch (not PC-relative); the effective target address is in the “current” 256 MB-aligned region. The low 28 bits of the target address is the instr_index field shifted left 2 bits. The remaining upper bits are the corresponding bits of the address of the instruction in the delay slot (not the branch itself).

**Restrictions** : None

**Exceptions** : None

**Notes** : Forming the branch target address by catenating PC and index bits rather than adding a signed offset to the PC is an advantage if all program code addresses fit into a 256 MB region aligned on a 256 MB boundary. It allows a branch from anywhere in the region to anywhere in the region, an action not allowed by a signed relative offset.

## JAL — Jump And Link




**Format** : JAL target

**Fonction** : To execute a procedure call within the current 256 MB-aligned region

**Description** :

Place the return address link in GPR 31. The return link is the address of the second instruction following the branch, at which location execution continues after a procedure call. This is a PC-region branch (not PC-relative); the effective target address is in the “current” 256 MB-aligned region. The low 28 bits of the target address is the instr_index field shifted left 2 bits. The remaining upper bits are the corresponding bits of the address of the instruction in the delay slot (not the branch itself).

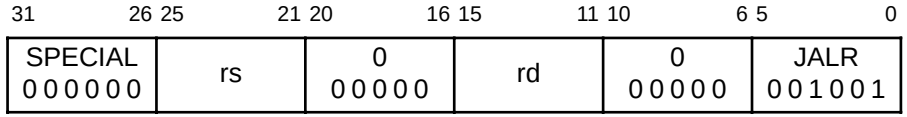
**Restrictions** : None

**Exceptions** : None

**Notes** : Forming the branch target address by catenating PC and index bits rather than adding a signed offset to the PC is an advantage if all program code addresses fit into a 256 MB region aligned on a 256 MB boundary. It allows a branch from anywhere in the region to anywhere in the region, an action not allowed by a signed relative offset.

## JALR

Jump And Link Register




**Format** : JALR rs (rd = 31 implicite)
JALR rd, rs

**Fonction** : To execute a procedure call to an instruction address in a register

**Description** : rd ← return_addr, PC ← rs

Place the return address link in GPR rd. The return link is the address of the second instruction following the branch, where execution continues after a procedure call.

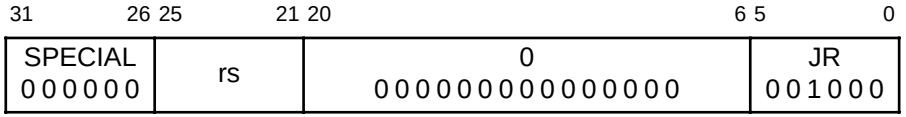
**Restrictions** : Register specifiers rs and rd must not be equal, because such an instruction does not have the same effect when reexecuted. The result of executing such an instruction is UNPREDICTABLE. This restriction permits an exception handler to resume execution by re-executing the branch when an exception occurs in the branch delay slot.

**Exceptions** : None

**Notes** : This is the only branch-and-link instruction that can select a register for the return link; all other link instructions use GPR 31. The default register for GPR rd, if omitted in the assembly language instruction, is GPR 31.

## JR

Jump Register




**Format** : JR rs

**Fonction** : To execute a branch to an instruction address in a register

**Description** : PC ← rs

Jump to the effective target address in GPR rs.

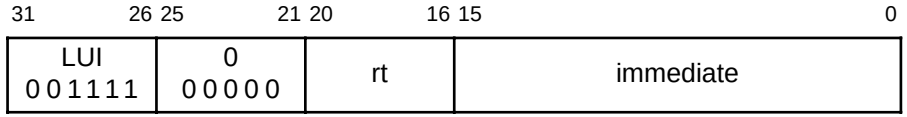
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## LUI

Load Upper Immediate




**Format** : LUI rt, immediate

**Fonction** : To load a constant into the upper half of a word

**Description** : rt ← immediate || $0^{16}$

The 16-bit immediate is shifted left 16 bits and concatenated with 16 bits of low-order zeros. The 32-bit result is placed into GPR rt.

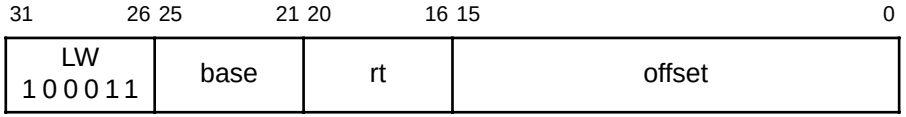
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## LW

Load Word




**Format** : LW rt, offset(base)

**Fonction** : To load a word from memory as a signed value

**Description** : rt ← memory[base+offset]

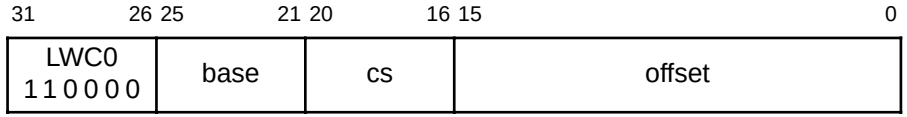
The contents of the 32-bit word at the memory location specified by the aligned effective address are fetched, sign-extended to the GPR register length if necessary, and placed in GPR rt. The 16-bit signed offset is added to the contents of GPR base to form the effective address.

**Restrictions** : None

**Exceptions** : None

**Notes** : None

## LWC0 — Load Word to Coprocessor System




**Format** : LWC0 cs, offset(base)

**Fonction** : To load a word from memory to an coprocessor system register.

**Description** : cs ← memory[base+offset]

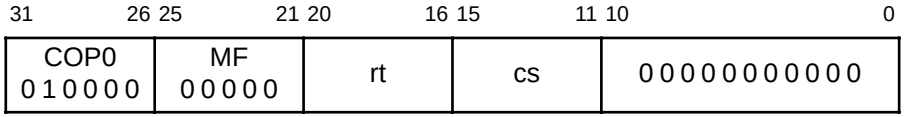
The contents of the 32-bit word at the memory location specified by the aligned effective address are fetched and placed into the coprocessor 0 general register cs. The 16-bit signed offset is added to the contents of GPR base to form the effective address.

**Restrictions** : None

**Exceptions** : None

**Notes** : None

## MFC0 — Move From Coprocessor System




**Format** : MFC0 rt, cs

**Fonction** : To move the contents of a coprocessor 0 register to a general register.

**Description** : rt ← cs

The contents of the coprocessor 0 register cs is loaded into general register rt.

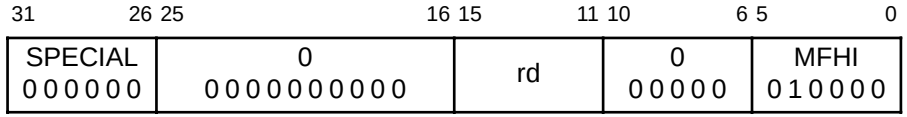
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## MFHI

Move From HI register




**Format** : MFHI rd

**Fonction** : To copy the special purpose HI register to a GPR

**Description** : rd ← HI

The contents of special register HI are loaded into GPR rd.

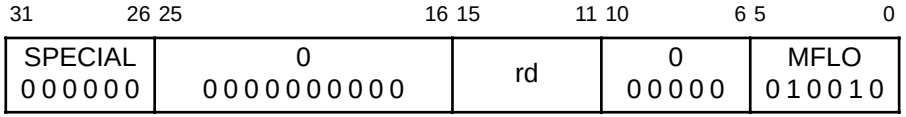
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## MFLO

Move From LO register




**Format** : MFLO rd

**Fonction** : To copy the special purpose LO register to a GPR

**Description** : rd ← LO

The contents of special register LO are loaded into GPR rd.

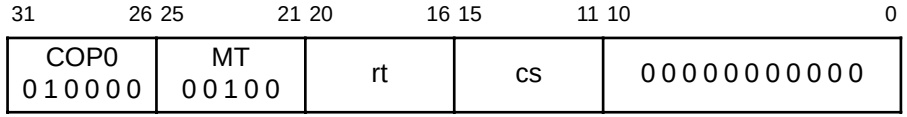
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## MTC0

Move To Coprocessor System




**Format** : MTC0 rt, cs

**Fonction** : To move the contents of a general register to a coprocessor 0 register.

**Description** : cs ← rt

The contents of general register rt are loaded into the coprocessor 0 register rd

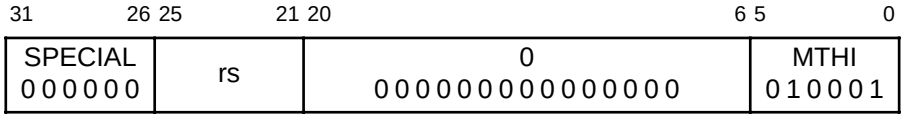
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## MTHI

Move To HI register




**Format** : MTHI rs

**Fonction** : To copy a GPR to the special purpose HI register

**Description** : HI ← rs

The contents of GPR rs are loaded into special register HI.

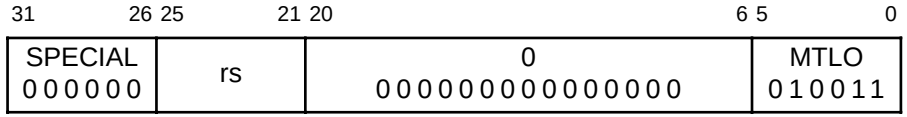
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## MTLO

Move To LO register




**Format** : MTLO rs

**Fonction** : To copy a GPR to the special purpose LO register

**Description** : HI ← rs

The contents of GPR rs are loaded into special register LO.

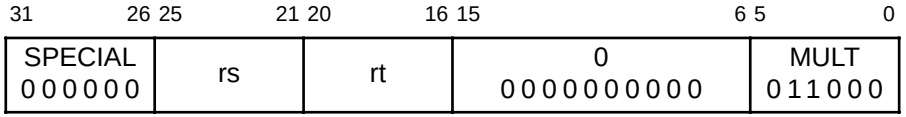
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## MULT

Multiply Word




**Format** : MULT rs, rt

**Fonction** : To multiply 32-bit signed integers

**Description** : (LO, HI) ← rs x rt

The 32-bit word value in GPR rt is multiplied by the 32-bit value in GPR rs, treating both operands as signed values, to produce a 64-bit result. The low-order 32-bit word of the result is placed into special register LO, and the high-order 32-bit word is splaced into special register HI. No arithmetic exception occurs under any circumstances.

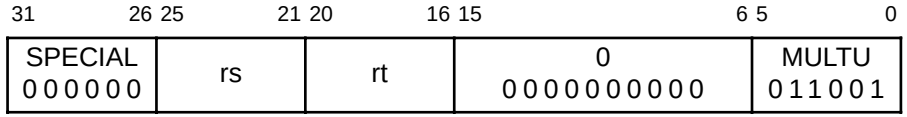
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## MULTU

Multiply Unsigned Word




**Format** : MULTU rs, rt

**Fonction** : To multiply 32-bit unsigned integers

**Description** : (LO, HI) ← rs x rt

The 32-bit word value in GPR rt is multiplied by the 32-bit value in GPR rs, treating both operands as unsigned values, to produce a 64-bit result. The low-order 32-bit word of the result is placed into special register LO, and the high-order 32-bit word is placed into special register HI. No arithmetic exception occurs under any circumstances.

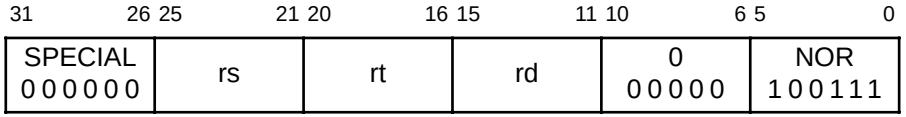
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## NOR

Not Or




**Format** : NOR rd, rs, rt

**Fonction** : To do a bitwise logical NOT OR

**Description** : rd ← rs NOR rt

The contents of GPR rs are combined with the contents of GPR rt in a bit-wise logical NOR operation. The result is placed into GPR rd.

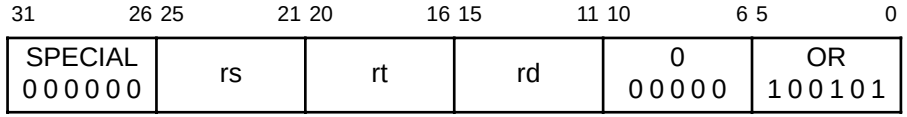
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## OR

Or




**Format** : OR rd, rs, rt

**Fonction** : To do a bitwise logical OR

**Description** : rd ← rs OR rt

The contents of GPR rs are combined with the contents of GPR rt in a bit-wise logical OR operation. The result is placed into GPR rd.

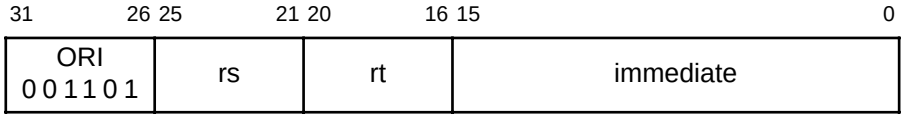
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## ORI

Or Immediate




**Format** : ORI rt, rs, immediate

**Fonction** : To do a bitwise logical OR with a constant

**Description** : rt ← rs OR immediate

The 16-bit immediate is zero-extended to the left and combined with the contents of GPR rs in a bitwise logical OR operation. The result is placed into GPR rt.

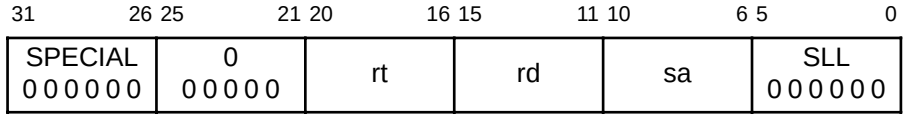
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SLL

Shift Word Left Logical




**Format** : SLL rd, rt, sa

**Fonction** : To left-shift a word by a fixed number of bits

**Description** : rd ← rt << sa

The contents of the low-order 32-bit word of GPR rt are shifted left, inserting zeros into the emptied bits; the word result is placed in GPR rd. The bit-shift amount is specified by sa.

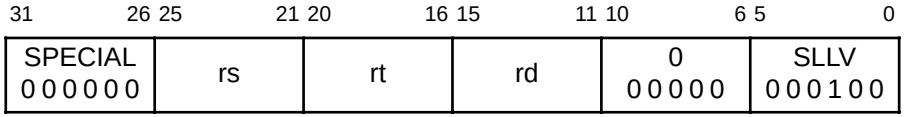
**Restrictions** : None

**Exceptions** : None

**Notes** : SLL r0, r0, 0, expressed as NOP, is the assembly idiom used to denote no operation.

## SLLV

Shift Word Left Logical Variable




**Format** : SLLV rd, rt, rs

**Fonction** :To left-shift a word by a variable number of bits

**Description** : rd ← rt << rs

The contents of the low-order 32-bit word of GPR rt are shifted left, inserting zeros into the emptied bits; the result word is placed in GPR rd. The bit-shift amount is specified by the low-order 5 bits of GPR rs.

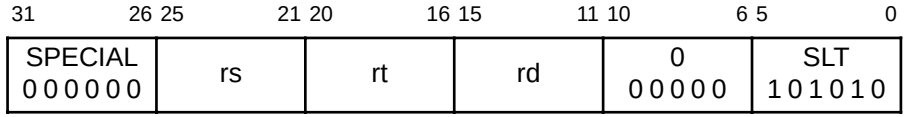
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SLT

Set On Less Than




**Format** : SLT rd, rs, rt

**Fonction** : To record the result of a less-than comparison

**Description** : rd ← (rs < rt)

Compare the contents of GPR rs and GPR rt as signed integers and record the Boolean result of the comparison in GPR rd. If GPR rs is less than GPR rt, the result is 1 (true); otherwise, it is 0 (false). The arithmetic comparison does not cause an Integer Overflow exception.

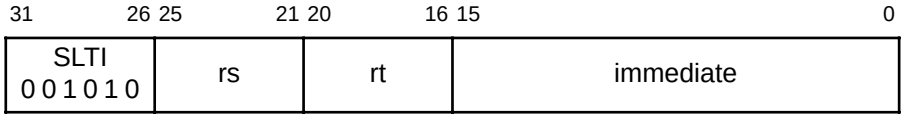
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SLTI

Set on Less Than Immediate




**Format** : SLTI rt, rs, immediate

**Fonction** : To record the result of a less-than comparison with a constant

**Description** : rt ← (rs < immediate)

Compare the contents of GPR rs and the 16-bit signed immediate as signed integers and record the Boolean result of the comparison in GPR rt. If GPR rs is less than immediate, the result is 1 (true); otherwise, it is 0 (false). The arithmetic comparison does not cause an Integer Overflow exception.

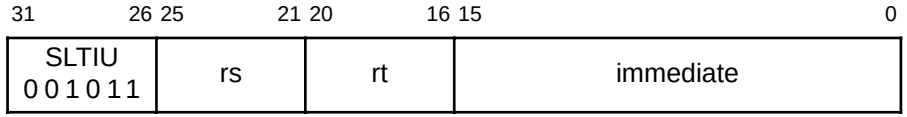
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SLTIU

Set on Less Than Immediate Unsigned




**Format** : SLTIU rt, rs, immediate

**Fonction** : To record the result of an unsigned less-than comparison with a constant

**Description** : rt ← (rs < immediate)

Compare the contents of GPR rs and the sign-extended 16-bit immediate as unsigned integers and record the Boolean result of the comparison in GPR rt. If GPR rs is less than immediate, the result is 1 (true); otherwise, it is 0 (false). The arithmetic comparison does not cause an Integer Overflow exception.

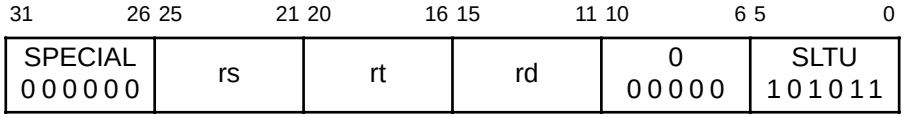
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SLTU

Set on Less Than Unsigned




**Format** : SLTU rd, rs, rt

**Fonction** : To record the result of an unsigned less-than comparison

**Description** : rd ← (rs < rt)

Compare the contents of GPR rs and GPR rt as unsigned integers and record the Boolean result of the comparison in GPR rd. If GPR rs is less than GPR rt, the result is 1 (true); otherwise, it is 0 (false). The arithmetic comparison does not cause an Integer Overflow exception.

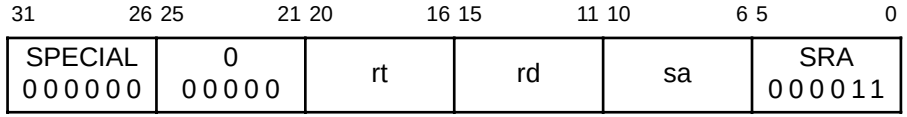
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SRA

Shift Word Right Arithmetic




**Format** : SRA rd, rt, sa

**Fonction** : To execute an arithmetic right-shift of a word by a fixed number of bits

**Description** : rd ← rt >> sa (arithmetic)

The contents of the low-order 32-bit word of GPR rt are shifted right, duplicating the sign-bit (bit 31) in the emptied bits; the word result is placed in GPR rd. The bit-shift amount is specified by sa.

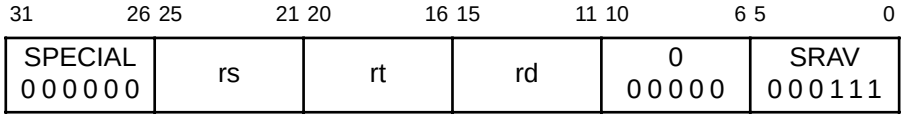
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SRAV

Shift Word Right Arithmetic Variable




**Format** : SRAV rd, rt, rs

**Fonction** : To execute an arithmetic right-shift of a word by a variable number of bits

**Description** : rd ← rt >> rs (arithmetic)

The contents of the low-order 32-bit word of GPR rt are shifted right, duplicating the sign-bit (bit 31) in the emptied bits; the word result is placed in GPR rd. The bit-shift amount is specified by the low-order 5 bits of GPR rs.

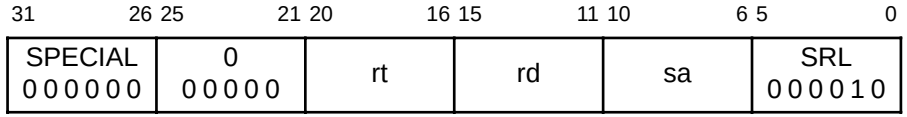
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SRL

Shift Word Right Logical




**Format** : SRL rd, rt, sa

**Fonction** : To execute a logical right-shift of a word by a fixed number of bits

**Description** : rd ← rt >> sa

The contents of the low-order 32-bit word of GPR rt are shifted right, inserting zeros into the emptied bits; the word result is placed in GPR rd. The bit-shift amount is specified by sa.

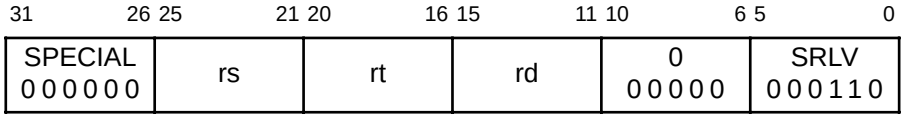
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SRLV

Shift Word Right Logical Variable




**Format** : SRLV rd, rt, rs

**Fonction** : To execute a logical right-shift of a word by a variable number of bits

**Description** : rd ← rt >> rs

The contents of the low-order 32-bit word of GPR rt are shifted right, inserting zeros into the emptied bits; the word result is placed in GPR rd. The bit-shift amount is specified by the low-order 5 bits of GPR rs.

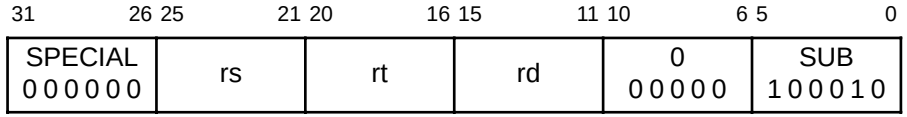
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SUB

Substract Word




**Format** : SUB rd, rs, rt

**Fonction** : To subtract 32-bit integers. If overflow occurs, then trap

**Description** : rd ← rs - rt

The 32-bit word value in GPR rt is subtracted from the 32-bit value in GPR rs to produce a 32-bit result. If the subtraction results in 32-bit 2's complement arithmetic overflow, then the destination register is not modified and an Integer Overflow exception occurs. If it does not overflow, the 32-bit result is placed into GPR rd.

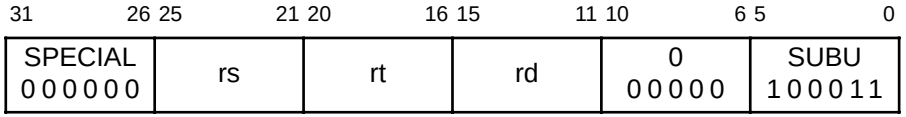
**Restrictions** : None

**Exceptions** : Integer Overflow

**Notes** : SUBU performs the same arithmetic operation but does not trap on overflow.

## SUBU

Substract Unsigned Word




**Format** : SUBU rd, rs, rt

**Fonction** : To subtract unsigned 32-bit integers

**Description** : rd ← rs - rt

The 32-bit word value in GPR rt is subtracted from the 32-bit value in GPR rs and the 32-bit arithmetic result is and placed into GPR rd. No integer overflow exception occurs under any circumstances.

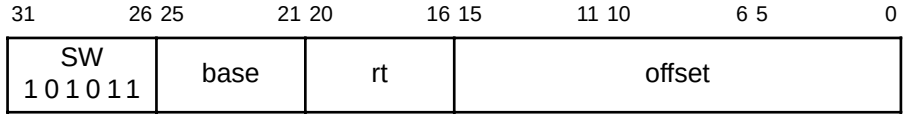
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SW

Store Word




**Format** : SW rt, offset(base)

**Fonction** : To store a word to memory

**Description** : memory[base+offset] ← rt

The least-significant 32-bit word of register rt is stored in memory at the location specified by the aligned effective address. The 16-bit signed offset is added to the contents of GPR base to form the effective address.

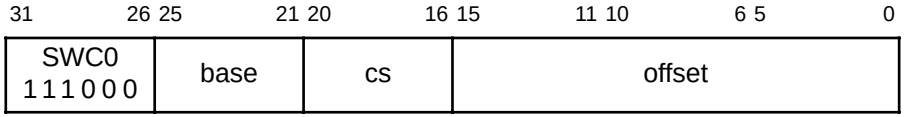
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SWC0

Store Word From Coprocessor System




**Format** : SW cs, offset(base)

**Fonction** : To store a word from an COP0 register to memory

**Description** : memory[base+offset] ← cs

The word from COP0 cs is stored in memory at the location specified by the aligned effective address. The 16-bit signed offset is added to the contents of GPR base to form the effective address.

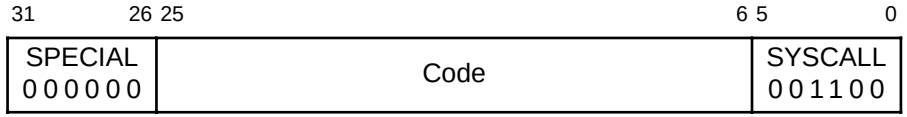
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## SYSCALL

System Call




**Format** : SYSCALL

**Fonction** : To cause a System Call exception

**Description** :

A system call exception occurs, immediately and unconditionally transferring control to the exception handler. The code field is available for use as software parameters, but is retrieved by the exception handler only by loading the contents of the memory word containing the instruction.

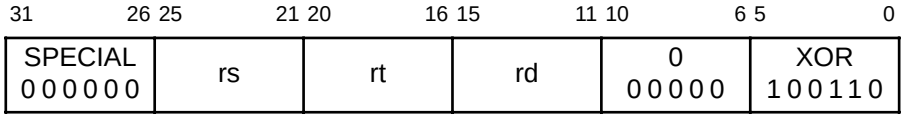
**Restrictions** : None

**Exceptions** : System Call

**Notes** : None

## XOR

Exclusive Or




**Format** : XOR rd, rs, rt

**Fonction** : To do a bitwise logical Exclusive OR

**Description** : rd ← rs XOR rt

LCombine the contents of GPR rs and GPR rt in a bitwise logical Exclusive OR operation and place the result into GPR rd.

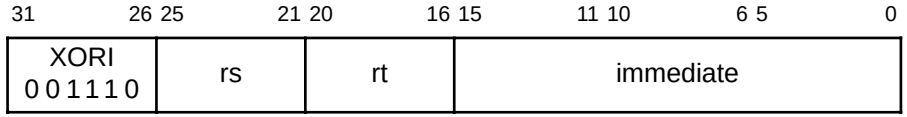
**Restrictions** : None

**Exceptions** : None

**Notes** : None

## XORI

Exclusive Or Immediate




**Format** : XORI rt, rs, immediate

**Fonction** : To do a bitwise logical Exclusive OR with a constant

**Description** : rt ← rs XOR immediate

Combine the contents of GPR rs and the 16-bit zero-extended immediate in a bitwise logical Exclusive OR operation and place the result into GPR rt.

**Restrictions** : None

**Exceptions** : None

**Notes** : None