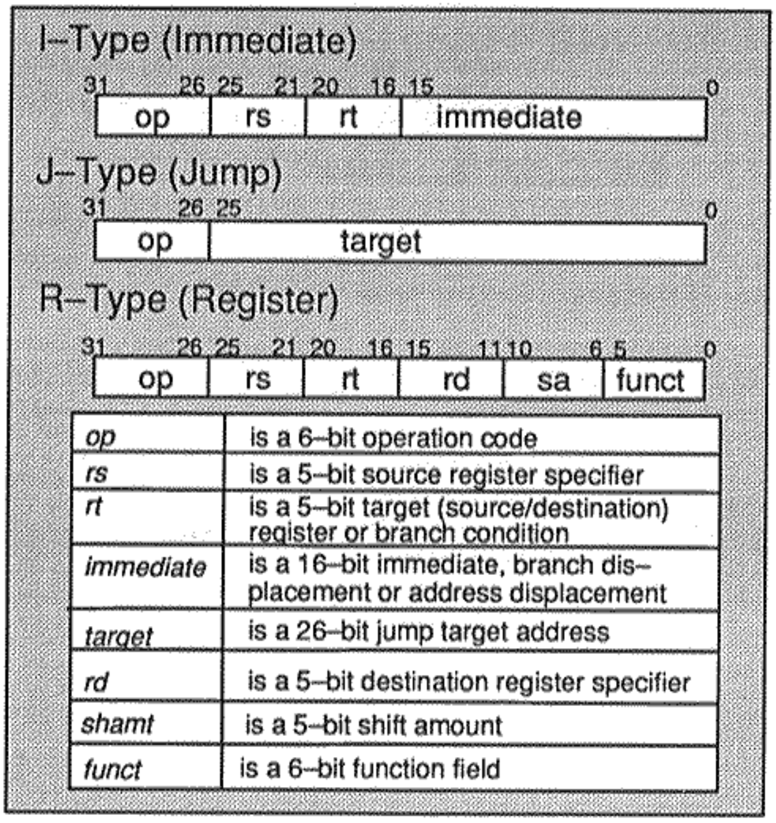
MIPS-Light Instruction Set Summary

The MIPS-Light ISA is a stripped down version of the MIPS R2000 ISA which is very close to the DLX ISA that is described in the textbook. The main difference between MIPS-Light ISA and DLX ISA concerns the branch instructions. MIPS-Light allows branches that have equal and not-equal comparisons between any two registers. When one of the registers is `r0`, the branch instruction is equivalent to a `beqz` or a `bnez` instruction in DLX. Please ignore references to the instructions `BLTZAL` and `BGEZAL`, which are not implemented in the MIPS-Lite Verilog model.

5.0.1 Instruction Formats

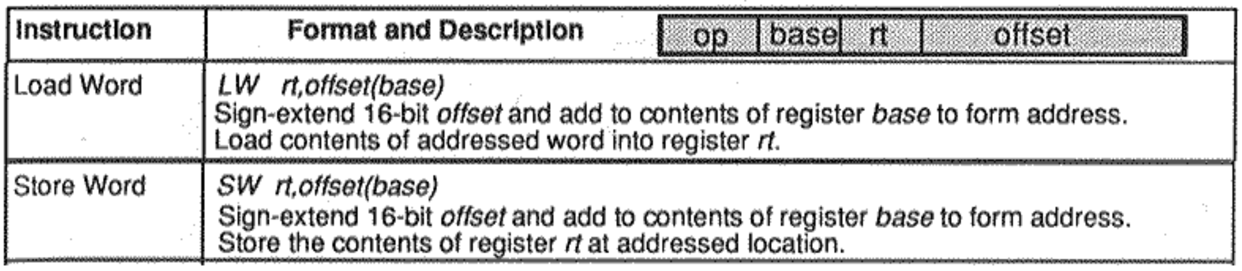


op	is a 6–bit operation code
rs	is a 5–bit source register specifier
rt	is a 5–bit target (source/destination) register or branch condition
immediate	is a 16–bit immediate, branch displacement or address displacement
target	is a 26–bit jump target address
rd	is a 5–bit destination register specifier
shamt	is a 5–bit shift amount
funct	is a 6–bit function field

In addition to the standard R2000 formats shown above, the MIPS-lite instruction set also has an additional format for the `bltz` and `bgez` instructions:

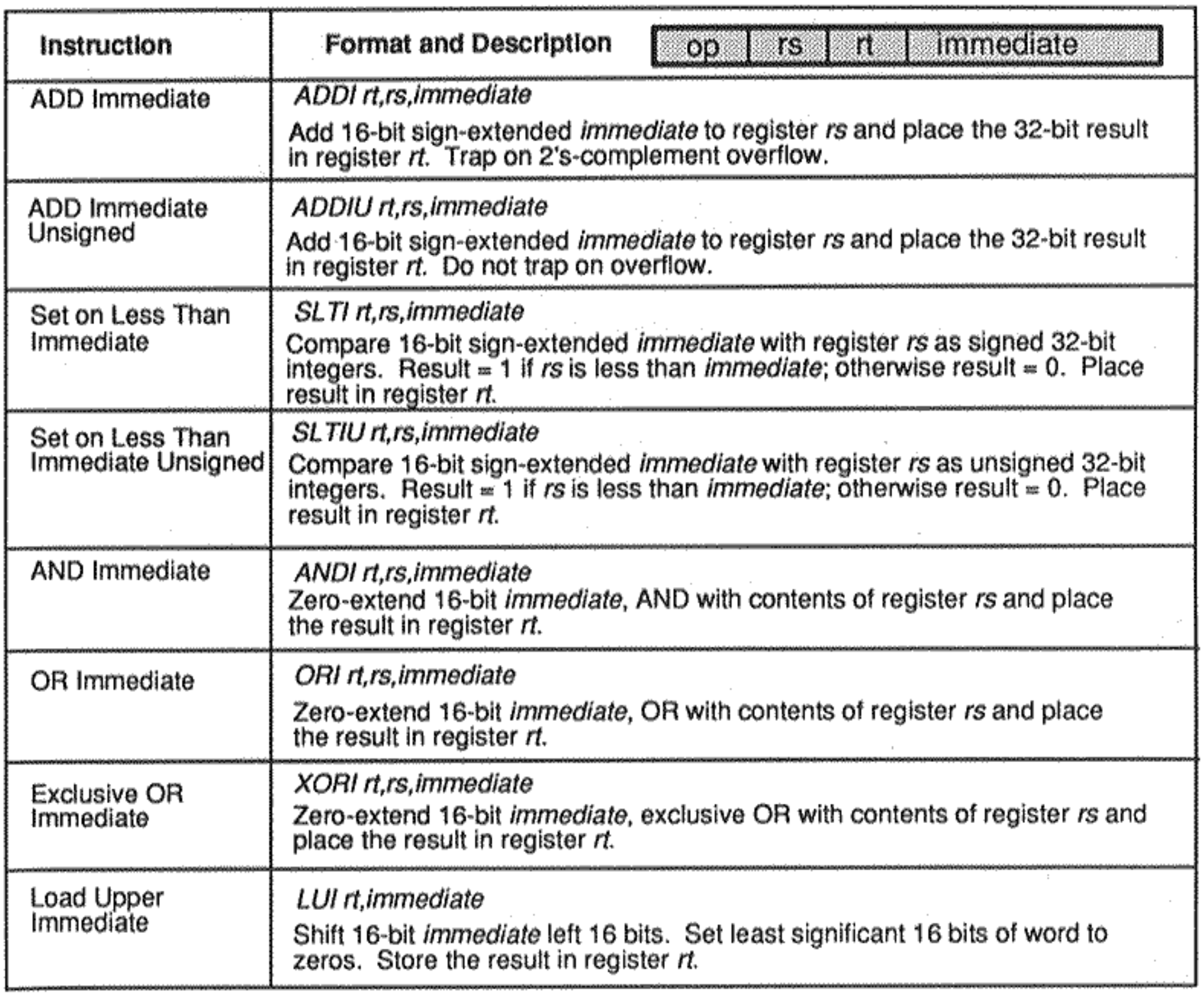
```
Bit:                [31-26]         [25-21]         [20-16]         [15-0]
Field:              op=REGIMM       rs              sub             offset
```

5.0.2 Load and Store Instructions

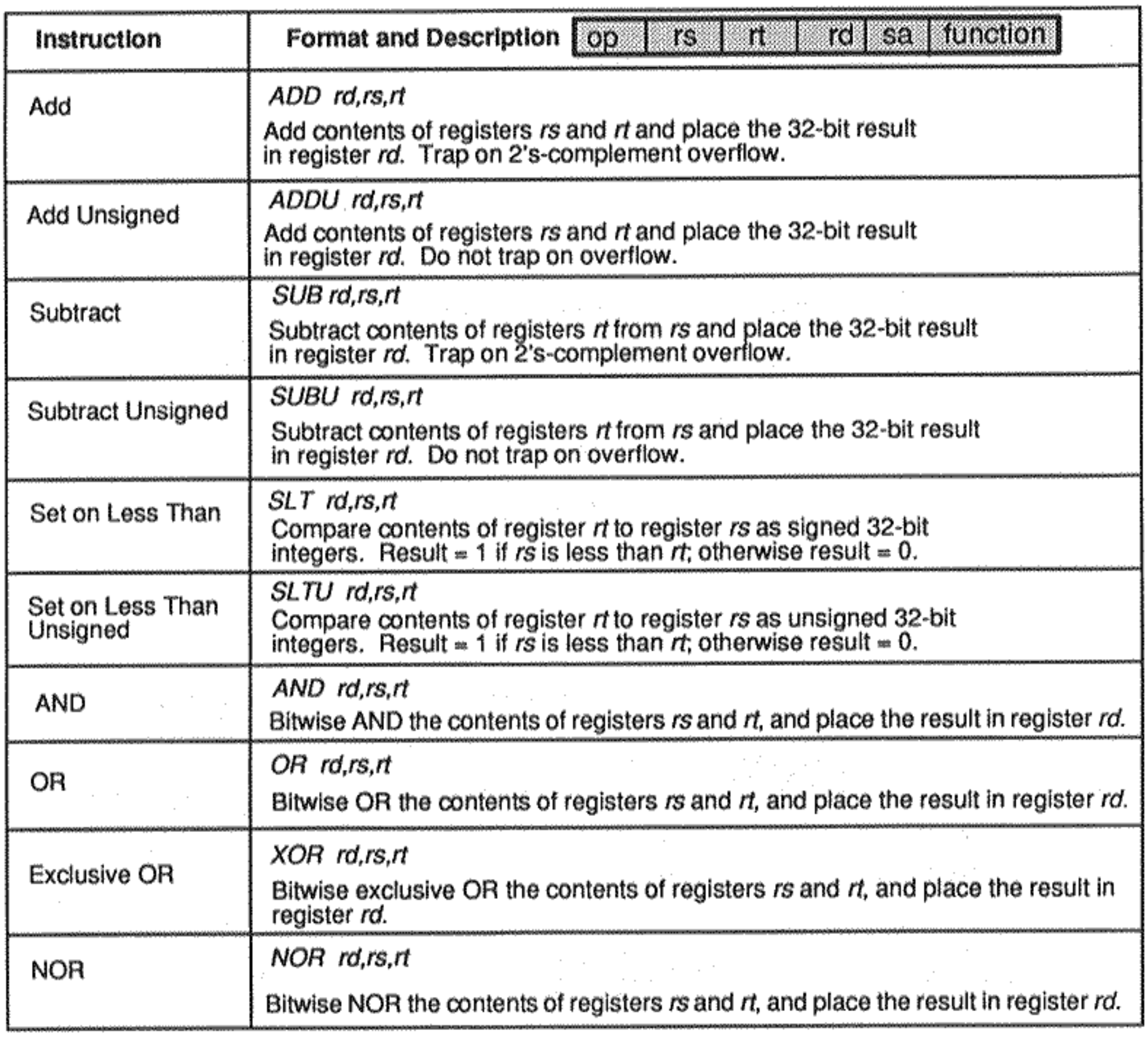


Instruction	Format and Description (op | base | rt | offset)
Load Word	*LW rt,offset(base)* Sign-extend 16-bit *offset* and add to contents of register *base* to form address. Load contents of addressed word into register *rt*.
Store Word	*SW rt,offset(base)* Sign-extend 16-bit *offset* and add to contents of register *base* to form address. Store the contents of register *rt* at addressed location.

5.0.3 ALU Instructions



Instruction	Format and Description | op | rs | rt | immediate |
ADD Immediate	*ADDI rt,rs,immediate* Add 16-bit sign-extended *immediate* to register *rs* and place the 32-bit result in register *rt*. Trap on 2's-complement overflow.
ADD Immediate Unsigned	*ADDIU rt,rs,immediate* Add 16-bit sign-extended *immediate* to register *rs* and place the 32-bit result in register *rt*. Do not trap on overflow.
Set on Less Than Immediate	*SLTI rt,rs,immediate* Compare 16-bit sign-extended *immediate* with register *rs* as signed 32-bit integers. Result = 1 if *rs* is less than *immediate*; otherwise result = 0. Place result in register *rt*.
Set on Less Than Immediate Unsigned	*SLTIU rt,rs,immediate* Compare 16-bit sign-extended *immediate* with register *rs* as unsigned 32-bit integers. Result = 1 if *rs* is less than *immediate*; otherwise result = 0. Place result in register *rt*.
AND Immediate	*ANDI rt,rs,immediate* Zero-extend 16-bit *immediate*, AND with contents of register *rs* and place the result in register *rt*.
OR Immediate	*ORI rt,rs,immediate* Zero-extend 16-bit *immediate*, OR with contents of register *rs* and place the result in register *rt*.
Exclusive OR Immediate	*XORI rt,rs,immediate* Zero-extend 16-bit *immediate*, exclusive OR with contents of register *rs* and place the result in register *rt*.
Load Upper Immediate	*LUI rt,immediate* Shift 16-bit *immediate* left 16 bits. Set least significant 16 bits of word to zeros. Store the result in register *rt*.



Instruction	Format and Description op | rs | rt | rd | sa | function
Add	*ADD rd,rs,rt* Add contents of registers *rs* and *rt* and place the 32-bit result in register *rd*. Trap on 2's-complement overflow.
Add Unsigned	*ADDU rd,rs,rt* Add contents of registers *rs* and *rt* and place the 32-bit result in register *rd*. Do not trap on overflow.
Subtract	*SUB rd,rs,rt* Subtract contents of registers *rt* from *rs* and place the 32-bit result in register *rd*. Trap on 2's-complement overflow.
Subtract Unsigned	*SUBU rd,rs,rt* Subtract contents of registers *rt* from *rs* and place the 32-bit result in register *rd*. Do not trap on overflow.
Set on Less Than	*SLT rd,rs,rt* Compare contents of register *rt* to register *rs* as signed 32-bit integers. Result = 1 if *rs* is less than *rt*; otherwise result = 0.
Set on Less Than Unsigned	*SLTU rd,rs,rt* Compare contents of register *rt* to register *rs* as unsigned 32-bit integers. Result = 1 if *rs* is less than *rt*; otherwise result = 0.
AND	*AND rd,rs,rt* Bitwise AND the contents of registers *rs* and *rt*, and place the result in register *rd*.
OR	*OR rd,rs,rt* Bitwise OR the contents of registers *rs* and *rt*, and place the result in register *rd*.
Exclusive OR	*XOR rd,rs,rt* Bitwise exclusive OR the contents of registers *rs* and *rt*, and place the result in register *rd*.
NOR	*NOR rd,rs,rt* Bitwise NOR the contents of registers *rs* and *rt*, and place the result in register *rd*.

Instruction	**Format and Description** op | rs | rt | rd | sa | function
Shift Left Logical	*SLL rd,rt,sa* Shift the contents of register *rt* left by *sa* bits, inserting zeros into the low order bits. Place the 32-bit result in register *rd*.
Shift Right Logical	*SRL rd,rt,sa* Shift the contents of register *rt* right by *sa* bits, inserting zeros into the high order bits. Place the 32-bit result in register *rd*.
Shift Right Arithmetic	*SRA rd,rt,sa* Shift the contents of register *rt* right by *sa* bits, sign-extending the high order bits. Place the 32-bit result in register *rd*.
Shift Left Logical Variable	*SLLV rd,rt,rs* Shift the contents of register *rt* left. The low order 5 bits of register *rs* specify the number of bits to shift left; insert zeros into the low order bits of *rt* and place the 32-bit result in register *rd*.
Shift Right Logical Variable	*SRLV rd,rt,rs* Shift the contents of register *rt* right. The low order 5 bits of register *rs* specify the number of bits to shift right; insert zeros into the high order bits of *rt* and place the 32-bit result in register *rd*.
Shift Right Arithmetic Variable	*SRAV rd,rt,rs* Shift the contents of register *rt* right. The low order 5 bits of register *rs* specify the number of bits to shift right; sign-extend the high order bits of *rt* and place the 32-bit result in register *rd*.

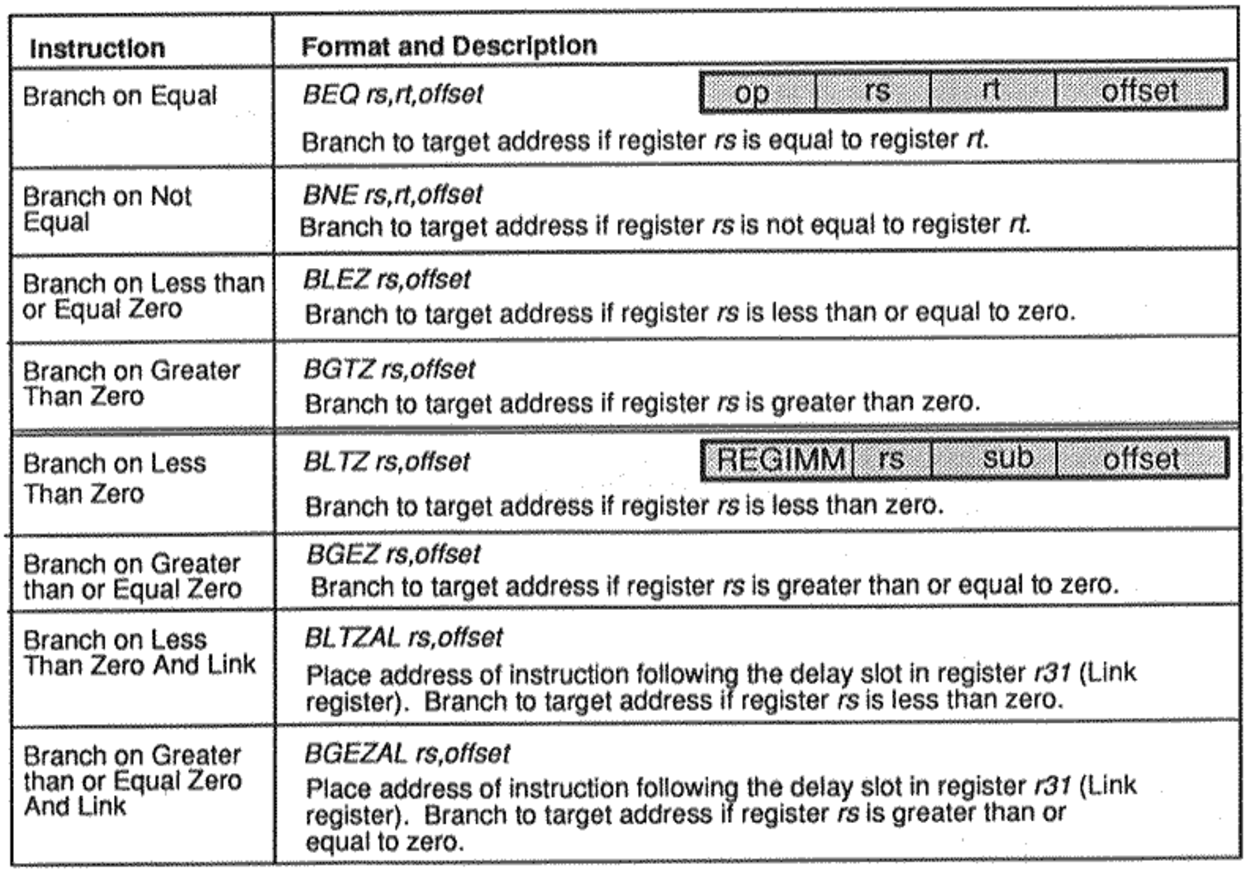
5.0.4 Jump and Branch Instructions

Instruction	**Format and Description** op | target
Jump	*J target* Shift the 26-bit *target* address left two bits, combine with high order four bits of the PC, and jump to the address with a 1-instruction delay.
Jump And Link	*JAL target* Shift the 26-bit *target* address left two bits, combine with high order four bits of the PC, and jump to the address with a 1-instruction delay. Place the address of the instruction following the delay slot in *r31* (*Link* register).

Instruction	**Format and Description** op | rs | rt | rd | sa | function
Jump Register	*JR rs* Jump to the address contained in register *rs*, with a 1-instruction delay.
Jump And Link Register	*JALR rs, rd* Jump to the address contained in register *rs*, with a 1-instruction delay. Place the address of the instruction following the delay slot in register *rd*.

ERRATA: The correct `JALR` instruction format is:

```
JALR rd, rs
```



Instruction	Format and Description
Branch on Equal	*BEQ rs,rt,offset* `op` `rs` `rt` `offset` Branch to target address if register *rs* is equal to register *rt.*
Branch on Not Equal	*BNE rs,rt,offset* Branch to target address if register *rs* is not equal to register *rt.*
Branch on Less than or Equal Zero	*BLEZ rs,offset* Branch to target address if register *rs* is less than or equal to zero.
Branch on Greater Than Zero	*BGTZ rs,offset* Branch to target address if register *rs* is greater than zero.
Branch on Less Than Zero	*BLTZ rs,offset* `REGIMM` `rs` `sub` `offset` Branch to target address if register *rs* is less than zero.
Branch on Greater than or Equal Zero	*BGEZ rs,offset* Branch to target address if register *rs* is greater than or equal to zero.
Branch on Less Than Zero And Link	*BLTZAL rs,offset* Place address of instruction following the delay slot in register *r31* (Link register). Branch to target address if register *rs* is less than zero.
Branch on Greater than or Equal Zero And Link	*BGEZAL rs,offset* Place address of instruction following the delay slot in register *r31* (Link register). Branch to target address if register *rs* is greater than or equal to zero.