BEQ `Branch on Equal` p68

31-26	25-21	20-16	15-0
BEQ(000100)	rs	rt	offset
6	5	5	16

Format:

```
BEQ rs, rt, offset
```

Purpose:

To compare GPR[rs] and GPR[rt] then do a PC-relative conditional branch

Description:

if GPR[rs] == GPR[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 2 GPRs are equal then branch to the effective target address after the instruction in the delay slot is executed

BEQL `Branch on Equal Likely` p69

31-26	25-21	20-16	15-0
BEQL(010100)	rs	rt	offset
6	5	5	16

Format:

```
BEQL rs, rt, offset
```

Purpose:

To compare GPR[rs] and GPR[rt] then do a PC-relative conditional branch; execute the delay slot only if the branch is taken

Description:

if GPR[rs] == GPR[rt] then branch likely

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 branch is not taken, the instruction in the delay slot is not executed

BGEZ Branch on Greater Than or Equal to Zero p70

31-26	25-21	20-16	15-0
REGIMM(000001)	rs	BGEZ(00001)	offset
6	5	5	16

Format:

```
BGEZ rs, offset
```

Purpose:

To test a GPR then do a PC-relative conditional branch

Description:

if GPR[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 2 GPRs are greater than or equal to zero, then branch to the effective target address after the instruction in the delay slot is executed

BGEZAL Branch on Greater Than or Equal to Zero and Link p71

31-26	25-21	20-16	15-0
REGIMM(000001)	rs	BGEZAL(10001)	offset
6	5	5	16

Format:

```
BGEZAL rs, offset
```

Purpose:

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

Description:

if GPR[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

BGTZ `Branch on Greater Than Zero` p75

31-26	25-21	20-16	15-0
BGTZ(000111)	rs	0(00000)	offset
6	5	5	16

Format:

```
BGTZ rs, offset
```

Purpose:

To test a GPR then do a PC-relative conditional branch

Description:

if GPR[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 2 GPRs are greater than zero, then branch to the effective target address after the instruction in the delay slot is executed

BLEZ Branch on Less Than or Equal to Zero p77

31-26	25-21	20-16	15-0
BLEZ(000110)	rs	0(00000)	offset
6	5	5	16

Format:

```
BLEZ rs, offset
```

Purpose:

To test a GPR then do a PC-relative conditional branch

Description:

if GPR[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, then branch to the effective target address after the instruction in the delay slot is executed

BLTZ Branch on Less Than Zero p79

31-26	25-21	20-16	15-0
REGIMM(000001)	rs	BLTZ(00000)	offset
6	5	5	16

Format:

```
BLTZ rs, offset
```

Purpose:

To test a GPR then do a PC-relative conditional branch

Description:

if GPR[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, then branch to the effective target address after the instruction in the delay slot is executed

BLTZAL `Branch on Less Than Zero and Link` p80

31-26	25-21	20-16	15-0
REGIMM(000001)	rs	BLTZAL(10000)	offset
6	5	5	16

Format:

```
BLTZAL rs, offset
```

Purpose:

To test GPR[rs] then do a PC-relative conditional branch

Description:

if GPR[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

BNE `Branch on Not Equal` p84

31-26	25-21	20-16	15-0
BNE(000101)	rs	rt	offset
6	5	5	16

Format:

```
BNE rs, rt, offset
```

Purpose:

To compare GPRs then do a PC-relative conditional branch

Description:

if GPR[rs] != GPR[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, then branch to the effective target address after the instruction in the delay slot is executed

J `Jump` p129

31-26	25-0
J(000010)	instr_index
6	26

Format:

```
J target
```

Purpose:

To branch within the current 256MB-aligned region

Description:

This is a PC-region branch(not PC-relative); the effective target address is in the "current" 256MB-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)

JAL `Jump and Link` p130

31-26	25-0
JAL(000011)	instr_index
6	26

Format:

```
JAL target
```

Purpose:

To execute a procedure call within the current 256MB-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

JALR `Jump and Link Register` p131

31-26	25-21	20-16	15-11	10-6	5-0
SPECIAL(000000)	rs	0(00000)	rd	hint	JALR(001001)
6	5	5	5	5	6

Format:

```
JALR rs (rd = 31 implied)
JALR rd, rs
```

Purpose:

To execute a procedure call to an instruction address in a register

Description:

GPR[rd] ← return_addr, PC ← GPR[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.

JR `Jump Register` p138

31-26	25-21	20-11	10-6	5-0
SPECIAL(000000)	rs	0(0000000000)	hint	JALR(001001)
6	5	10	5	6

Format:

```
JR rs
```

Purpose:

To execute a branch to an instruction address in a register

Description:

PC ← GPR[rs]

Jump to the effective target address in GPR[rs]. Execute the instruction following the jump, in the branch delay slot, before jumping

LWL `Load Word Left` p155

31-26	25-21	20-16	15-0
REGIMM(000001)	rs	BGEZAL(10001)	offset
6	5	5	16

Format:

```
LWL rt, offset(base)
```

Purpose:

To load the most-significant part of a word(GPR[rt]) as a signed value from an unaligned memory address

Description:

Different from MIPS Official Document, this description is aiming at the hardware implemention level

1. EffAddr = SignedExtended(offset) + GPR[base]
   The address where CPU intended to load from
2. AliAddr = EffAddr - EffAddr[1:0]
   Because the CPU must load a word through an aligned address, so CPU need to find out the aligned address of this word
3. Save the lower (4 - EffAddr[1:0]) bytes to the left of the GPR[rt]

LL `Load Linked Word` p149

31-26	25-21	20-16	15-0
LL(110000)	base	rt	offset
6	5	5	16

Format:

```
LL rt, offset(base)
```

Purpose:

To load a word from memory for a atomic read-modify-write

Description:

GPR[rt] ← memory[GPR[base] + offset]

The LL and SC instructions provide the primitives to implement atomic read-modify-write(RMW) operations for synchronizable memory locations.

TEQ `Trap if Equal` p262

31-26	25-21	20-16	15-6	5-0
SPECIAL(000000)	rs	rt	code	TEQ(110100)
6	5	5	10	6

Format:

```
TEQ rs, rt
```

Purpose:

To compare GPRs and do a conditional trap

Description:

if GPR[rs] == GPR[rt] then Trap

Compare the contents of GPR[rs] and GPR[rt] as signed integers, if GPR[rs] is equal to GPR[rt], then take a Trap exception

TEQI `Trap if Equal Immediate` p263

31-26	25-21	20-16	15-0
REGIMM(00001)	rs	TEQI(01100)	immediate
6	5	5	16

Format:

> TEQI rs, immediate

Purpose:

To compare a GPR[rs] to a constant(imm) and do a conditional trap

Description:

if GPR[rs] == immediate then Trap

Compare the contents of GPR[rs] and the 16-bit signed *immediate* as signed integers, if GPR[rs] is equal to *immediate*, then take a Trap exception

TGE `Trap if Greater or Equal` p264

31-26	25-21	20-16	15-6	5-0
SPECIAL(000000)	rs	rt	code	TGE(110000)
6	5	5	10	6

Format:

```
TGE rs, rt
```

Purpose:

To compare GPRs and do a conditional trap

Description:

if GPR[rs] >= GPR[rt] then Trap

Compare the contents of GPR[rs] and GPR[rt] as signed integers, if GPR[rs] is greater than or equal to GPR[rt], then take a Trap exception

TGEI `Trap if Greater or Equal Immediate` p265

31-26	25-21	20-16	15-0
REGIMM(00001)	rs	TGEI(01000)	immediate
6	5	5	16

Format:

```
TGEI rs, immediate
```

Purpose:

To compare a GPR[rs] to a constant(imm) and do a conditional trap

Description:

if GPR[rs] >= immediate then Trap

Compare the contents of GPR[rs] and the 16-bit signed *immediate* as signed integers, if GPR[rs] is greater than or equal to *immediate*, then take a Trap exception

TGEU `Trap if Greater or Equal Unsigned` p267

31-26	25-21	20-16	15-6	5-0
SPECIAL(000000)	rs	rt	code	TGEU(110001)
6	5	5	10	6

Format:

```
TGEU rs, rt
```

Purpose:

To compare GPRs and do a conditional trap

Description:

if GPR[rs] >= GPR[rt] then Trap

Compare the contents of GPR[rs] and GPR[rt] as unsigned integers, if GPR[rs] is greater than or equal to GPR[rt], then take a Trap exception

TGEIU `Trap if Greater or Equal Immediate Unsigned` p266

31-26	25-21	20-16	15-0
REGIMM(00001)	rs	TGEIU(01001)	immediate
6	5	5	16

Format:

```
TGEIU rs, immediate
```

Purpose:

To compare GPR[rs] to a constant and do a conditional trap

Description:

if GPR[rs] >= immediate then Trap

Compare the contents of GPR[rs] and the 16-bit signed *immediate* as unsigned integers, if GPR[rs] is greater than or equal to *immediate*, then take a Trap exception

TLT `Trap if Less Than` p274

31-26	25-21	20-16	15-6	5-0
SPECIAL(000000)	rs	rt	code	TLT(110010)
6	5	5	10	6

Format:

```
TLT rs, rt
```

Purpose:

To compare GPRs and do a conditional trap

Description:

if GPR[rs] < GPR[rt] then Trap

Compare the contents of GPR[rs] and GPR[rt] as signed integers, if GPR[rs] is less than GPR[rt], then take a Trap exception

TLTI `Trap if Less Than Immediate` p275

31-26	25-21	20-16	15-0
REGIMM(00001)	rs	TLTI(01010)	immediate
6	5	5	16

Format:

```
TLTI rs, immediate
```

Purpose:

To compare GPR[rs] to a constant and do a conditional trap

Description:

if GPR[rs] < immediate then Trap

Compare the contents of GPR[rs] and the 16-bit signed *immediate* as signed integers, if GPR[rs] is less than *immediate*, then take a Trap exception

TLTIU `Trap if Less Than Immediate` p276

31-26	25-21	20-16	15-0
REGIMM(00001)	rs	TLTIU(01011)	immediate
6	5	5	16

Format:

```
TLTIU rs, immediate
```

Purpose:

To compare GPR[rs] to a constant and do a conditional trap

Description:

if GPR[rs] < immediate then Trap

Compare the contents of GPR[rs] and the 16-bit signed *immediate* as unsigned integers, if GPR[rs] is less than *immediate*, then take a Trap exception

TLTU `Trap if Less Than Unsigned` p277

31-26	25-21	20-16	15-6	5-0
SPECIAL(000000)	rs	rt	code	TLT(110011)
6	5	5	10	6

Format:

```
TLTU rs, rt
```

Purpose:

To compare GPRs and do a conditional trap

Description:

if GPR[rs] < GPR[rt] then Trap

Compare the contents of GPR[rs] and GPR[rt] as unsigned integers, if GPR[rs] is less than GPR[rt], then take a Trap exception

TNE `Trap if Not Equal` p278

31-26	25-21	20-16	15-6	5-0
SPECIAL(000000)	rs	rt	code	TLT(110110)
6	5	5	10	6

Format:

```
TNE rs, rt
```

Purpose:

To compare GPRs and do a conditional trap

Description:

if GPR[rs] != GPR[rt] then Trap

Compare the contents of GPR[rs] and GPR[rt] as signed integers, if GPR[rs] is not equal to GPR[rt], then take a Trap exception