**არითმეტიკული და ლოგიკური ინსტრუქციები**

ქვემოთ აღწერილი ყველა ინსტრუქციაში, **src1**, **src2**, და **dest** წარმოადგენენ საერთო დანიშნულების რეგისტრებს. **imm** არის 16-ბიტიანი უშუალო მონაცემი, რომელიც ჩადგმულია ინსტრუქციაში.

 add Rdest, Rsrc1, Rsrc2: შეკრება (გადავსებით)

 addi Rdest, Rsrc1, Imm: შეკრება უშუალო (გადავსებით)

 addu Rdest, Rsrc1, Rsrc2: შეკრება (გადავსების გარეშე)

 addiu Rdest, Rsrc1, Imm: შეკრება უშუალო (გადავსების გარეშე)

 and Rdest, Rsrc1, Src2: AND

 andi Rdest, Rsrc1, Imm: AND Immediate

 div Rsrc1, Rsrc2: გაყოფა (გადავსებით)

 divu Rsrc1, Rsrc2: გაყოფა (გადავსების გარეშე)

 mult Rsrc1, Rsrc2: გამრავლება

 multu Rsrc1, Rsrc2: უნიშნო გამრავლება

 nor Rdest, Rsrc1, Src2: NOR

 or Rdest, Rsrc1, Src2: OR

 ori Rdest, Rsrc1, Src2: OR Immediate

 sll Rdest, Rsrc1, Src2: Shift Left Logical

 sllv Rdest, Rsrc1, Rsrc2: Shift Left Logical Variable

 sra Rdest, Rsrc1, Src2: Shift Right Arithmetic

 srav Rdest, Rsrc1, Rsrc2: Shift Right Arithmetic Variable

 srl Rdest, Rsrc1, Src2: Shift Right Logical

 srlv Rdest, Rsrc1, Rsrc2: Shift Right Logical Variable

წანაცვლებები

 sub Rdest, Rsrc1, Rsrc2: გამოკლება (გადავსებით)

 subu Rdest, Rsrc1, Rsrc2: გამოკლება (გადავსების გარეშე)

 xor Rdest, Rsrc1, Src2: XOR

 xori Rdest, Rsrc1, Imm: XOR Immediate

**Constant-Manipulating Instructions**

 lui Rdest, imm: Load Upper უშუალო

**შედარების ინსტრუქციები**

ქვემოთ მოყვანილ ყველა ინსტრუქციაში, **Src2** შეიძლება იყოს რეგისტრი ან უშუალო რიცხვითი მონაცემი (16-ბიტიანი მთელი რიცხვი).

 slt Rdest, Rsrc1, Src2: ნაკლებია

 slti Rdest, Rsrc2, Imm: ნაკლებია უშუალო მონაცემზე

 sltu Rdest, Rsrc1, Src2: ნაკლებია ნიშნის გაუთვალისწინებლად

 sltiu Rdest, Rsrc1, Imm: ნაკლებია ნიშნის გაუთვალისწინებლად უშუალო მონაცემზე

რეგისტრი Rdest ხდება 1, თუ რეგისტრი Rsrc1 ნაკლებია Src2-ზე (ან Imm-ზე) და ხდება 0 წინააღმდეგ შემთხვევაში..

**განშტოების და გადასვლის ინსტრუქციები**

ქვემოთ მოყვანილ ყველა ინსტრუქციაში, **Src2** შეიძლება იყოს რეგისტრი ან უშუალო მნიშვნელობა (მთელი რიცხვი). გადასვლის ბრძანებები იყენებენ 16-თანრიგიან ნიშნის მქონე ველს. გადასვლა შესაძლებელია 2^15 -1 ინსტრუქციით (და არა ბაიტით) წინ და 2^15 ინსტრუქციით უკან. გადასვლის ბრძანება შეიცავს მისამართის 26-თანრიგიან ველს.

 bczt ჭდე: Branch Coprocessor z True

 bczf ჭდე: Branch Coprocessor z False

გადასვლა ინსტრუქციაზე შესაბამისი ჭდით, z კოპროცესორის პირობის ალამი არის true(false).

 beq Rsrc1, Rsrc2, offset: Branch on Equal

გადასვლა ინსტრუქციაზე შესაბამისი ჭდით, თუ Rsrc1 და Src2 რეგისტრები ტოლია.

 bgez Rsrc, offset: Branch on Greater Than Equal Zero

გადასვლა ინსტრუქციაზე შესაბამისი ჭდით, თუ Rsrc-ს მნიშვნელობა მეტია ან ტოლი 0-ზე.

 bgezal Rsrc, offset: Branch on Greater Than Equal Zero And Link

გადასვლა ინსტრუქციაზე შესაბამისი ჭდით, თუ Rsrc-ს მნიშვნელობა მეტია ან ტოლი 0-ზე. მომდევნო ინსტრუქციის მისამართი ინახება 31-ე რეგისტრში

 bgtz Rsrc, offset: Branch on Greater Than Zero

გადასვლა ინსტრუქციაზე შესაბამისი ჭდით, თუ Rsrc-ს მნიშვნელობა მეტია 0-ზე.

 blez Rsrc, offset: Branch on Less Than Equal Zero

გადასვლა ინსტრუქციაზე შესაბამისი ჭდით, თუ Rsrc-ს მნიშვნელობა ნაკლებია ან ტოლი 0-ზე.

 bgezal Rsrc, offset: Branch on Greater Than Equal Zero And Link

 bltzal Rsrc, offset: Branch on Less Than And Link

გადასვლა ინსტრუქციაზე შესაბამისი ჭდით, თუ Rsrc-ს მნიშვნელობა ნაკლებია ან ტოლი 0-ზე. მომდევნო ინსტრუქციის მისამართი ინახება 31-ე რეგისტრში.

 bltz Rsrc, offset: Branch on Less Than Zero

გადასვლა ინსტრუქციაზე შესაბამისი ჭდით, თუ Rsrc-ს მნიშვნელობა ნაკლებია 0-ზე.

 bne Rsrc1, Src2, offset: Branch on Not Equal

გადასვლა ინსტრუქციაზე შესაბამისი ჭდით, თუ Rsrc1 და Src2 რეგისტრები არ არის ტოლი.

 j ჭდე: Jump

უპირობო გადასვლა ინსტრუქციაზე მითითებული ჭდით

 jal ჭდე: Jump and Link

 jalr Rsrc: Jump and Link Register

უპირობო გადასვლა ინსტრუქციაზე მითითებული ჭდით ან Rsrc რეგისტრში ჩაწერილ მისამართზე. მომდევნო ინსტრუქციის მისამართი ინახება 31-ე რეგისტრში.

 jr Rsrc: Jump Register

უპირობო გადასვლა Rsrc რეგისტრში ჩაწერილ მისამართზე.

**ჩატვირთვის ინსტრუქციები**

 lb Rdest, imm(Rsrc): Load Byte

 lbu Rdest, imm(Rsrc): Load Unsigned Byte

ბაიტის ჩატვირთვა მეხსიერების Rsrc + imm მისამართიდან Rdest რეგისტრში.

 lh Rdest, imm(Rsrc): Load Halfword

 lhu Rdest, imm(Rsrc): Load Unsigned Halfword

16-ბიტიანი მონაცემის (ნახევარსიტყვის) ჩატვირთვა მეხსიერების Rsrc + imm მისამართიდან Rdest რეგისტრში.

 lw Rdest, imm(Rsrc): Load Word

32-ბიტიანი მონაცემის (სიტყვის) ჩატვირთვა მეხსიერების Rsrc + imm მისამართიდან Rdest რეგისტრში.

 lwcz Rdest, imm(Rsrc): Load Word

სიტყვის ჩატვირთვა მეხსიერების Rsrc + imm მისამართიდან კოპროცესორის Rdest რეგისტრში.

 lwl Rdest, imm(Rsrc): Load Word Left

 lwr Rdest, imm(Rsrc): Load Word Right

მარცხენა (მარჯვენა) სიტყვის ჩატვირთვა მეხსიერების Rsrc + imm მისამართიდან Rdest რეგისტრში.

**შStore Instructions**

 sb Rsrc1, imm(Rsrc2): Store Byte

რეგისტრ Rsrc1-ის უმცროსი ბაიტის შენახვა მეხსიერების Rsrc2 + imm მისამართზე.

 sh Rsrc1, imm(Rsrc2): Store Halfword

რეგისტრ Rsrc1-ში არსებული ნახევარსიტყვის შენახვა მეხსიერების Rsrc2 + imm მისამართზე.

 sw Rsrc1, imm(Rsrc2): Store Word

რეგისტრ Rsrc1-ში არსებული სიტყვის შენახვა მეხსიერების Rsrc2 + imm მისამართზე.

 swcz Rsrc1, imm(Rsrc2): Store Word

კოპროცესორის რეგისტრ Rsrc1-ში არსებული სიტყვის შენახვა მეხსიერების Rsrc2 + imm მისამართზე.

 swl Rsrc1, imm(Rsrc2): Store Word Left

 swr Rsrc1, imm(Rsrc2): Store Word Right

რეგისტრ Rsrc1-ში არსებული მარცხენა (მარჯვენა) სიტყვის მეხსიერების Rsrc2 + imm მისამართზე.

**სხვადასხვა ტიპის ინსტრუქციები**

გამრავლების და გაყოფის შედეგები ნაწილდება ორ დამატებით hi და lo რეგისტრში. მათი საშუალებით ხდება 0-ზე გაყოფის და გადავსების შეცდომების გამოვლენა გამრავლების, გაყოფის ან ნაშთის გამოთვლის ოპერაციებისას.

 mfhi Rdest: Move From hi

 mflo Rdest: Move From lo

hi(lo) რეგისტრში არსებული ინფორმაციის გადატანა Rdest რეგისტრში.

 mthi Rdest: Move To hi

 mtlo Rdest: Move To lo

Rdest რეგისტრში არსებული ინფორმაციის გადატანა hi(lo) რეგისტრში.

 mfcz Rdest, CPsrc: Move From Coprocessor z

გადააქვს ინფორმაცია ცენტრალური პროცესორის Rsrc რეგისტრიდან კოპროცესორის CPdest რეგისტრში.