

# Formato R

The diagram illustrates the R-format instruction format, which is 32 bits long. It is divided into several fields, each with a specific function:

- ADD:** ALUSrcA = 01, ALUSrcB = 00, ALUOp = 000, OutWrite = 1. RegisterWrite = 1, Register = 01, WriteData = 000.
- SUB:** ALUSrcA = 01, ALUSrcB = 00, ALUOp = 000, OutWrite = 1. RegisterWrite = 1, Register = 01, WriteData = 000.
- AND:** ALUSrcA = 01, ALUSrcB = 00, ALUOp = 011, OutWrite = 1. RegisterWrite = 1, Register = 01, WriteData = 000.
- DIV:** MemOpA = 0, MemOpB = 0, MemOpC = 1. x34. Wait. Hi = 1, Low = 1.
- MULT:** MemOpA = 1, MemOpB = 1, MemOpC = 0. x34. Wait. Hi = 1, Low = 1.
- JR:** ALUSrcA = 01, ALUOp = 000, PCSource = 00, PCWrite = 1.
- MFHI:** Register = 1, Register = 01, WriteData = 010.
- MFLO:** Register = 1, Register = 01, WriteData = 011.
- SLL:** Shift = 0, Shift = 01, Shift = 001. Shift = 010. Shift = 000, Register = 01, WriteData = 100.
- SLT:** ALUSrcA = 01, ALUSrcB = 00, ALUOp = 111. Register = 01, WriteData = 001, Register = 1. Shift = 010, Register = 01, WriteData = 100.
- SRA:** Shift = 0, Shift = 0, Shift = 001. Shift = 100. Shift = 000, Register = 01, WriteData = 100.
- DRVM:** ALUSrcA = 01, ALUOp = 000. Shift = 11, MemRead = 1. AuiMemOpA = 1, MemOpA = 1. ALUSrcA = 10, ALUSrcB = 00, ALUOp = 001. Shift = 11, MemRead = 1. AuiMemOpA = 1, MemOpA = 0. x34. Wait. MultiDiv = 1.

# FORMATO I

ADDI	BEQ	BNE	ADDM	LB	LUI	LW	SB	SW
ALUSrcA = 01 ALUSrcB = 10 ALUOp = 001 SignExtChCntrl = 1 OutWrite = 1	PCSource = 10 ALUSrcA = 01 ALUSrcB = 10 ALUOp = 000 PCWriteCondSource = 00 PCWriteCondCtrl = 1	PCSource = 10 ALUSrcA = 11 ALUSrcB = 00 ALUOp = 100 PCWriteCondSource = 01 PCWriteCondCtrl = 1	ALUSrcA = 01 ALUSrcB = 10 ALUSrcC = 00 ALUOp = 001 OutWrite = 1	ALUSrcA = 01 ALUSrcB = 10 ALUSrcC = 001 SignExtChCntrl = 0	ShiftIn = 01 SignExtChCntrl = 0 ShiftB = 1 ShiftChCntrl = 001	ALUSrcA = 01 ALUSrcB = 10 ALUSrcC = 001 SignExtChCntrl = 0	ALUSrcA = 01 ALUSrcB = 10 SignExtChCntrl = 0 ALUOp = 001	ALUSrcA = 01 ALUSrcB = 10 SignExtChCntrl = 0 ALUOp = 001
RegWrite = 1 RstCntrl = 0 WriteData = 000			InstOp = 100 DestRegWrite = 1 MemRead = 1	InstOp = 11 MemRead = 1	ShiftChCntrl = 010	InstOp = 11 MemRead = 1	InstOp = 11 MemRead = 1 DestRegWrite = 1	InstOp = 11 MemRead = 1 DestRegWrite = 1
			Wait MemRead	Wait MemRead	ShiftChCntrl = 000 RegChCntrl = 000 WriteData = 100 RegWrite = 1	Wait MemRead	Wait MemRead	Wait MemRead
			ALUSrcA = 01 ALUSrcB = 00 ALUOp = 001 OutWrite = 1	WriteData = 101 RegChCntrl = 00 RegWrite = 1 LoadChCntrl = 10		WriteData = 101 RegChCntrl = 00 RegWrite = 1 LoadChCntrl = 00	StoreChCntrl = 01 MemWrite = 1	StoreChCntrl = 00 MemWrite = 1

# FORMATO J

```
graph TD; J[J] --- JAL[JAL]; JAL --- JAL_PC[PCSource = 01  
PCWrite = 1]; JAL --- JAL_ALUSrc[ALUSrc = 00  
ALUSrc = 000  
OutWrite = 1]; JAL_PC --- JAL_Reg[PCSource = 01  
PCWrite = 1  
Register = 3  
WriteData = 000  
Register = 11];
```

The diagram illustrates the J-format instruction format, which consists of three main components connected in a vertical sequence:

- J**: A single-bit instruction type field.
- JAL**: A block containing:
  - ALUSrc = 00**: A 2-bit field for ALUSrc.
  - ALUSrc = 000**: A 3-bit field for ALUSrc.
  - OutWrite = 1**: A 1-bit field for OutWrite.
- PCSource = 01**: A 2-bit field for PCSource.
- PCWrite = 1**: A 1-bit field for PCWrite.
- Register = 3**: A 3-bit field for Register.
- WriteData = 000**: A 3-bit field for WriteData.
- Register = 11**: A 3-bit field for Register.

# Restrições

```
graph TD; subgraph UNEXISTENT_OP [UNEXISTENT OP]; U1["EPCause = 00  
hwD = 01  
ALUSrcA = 00  
ALUSrcB = 01  
ALUOp = 00  
MemRead = 1"]; U2["WAIT MEMREAD  
EPCWrite = 1  
MDWrite = 1"]; U3["SignExtendCH = 1  
ALUSrcA = 10  
ALUSrcB = 10  
ALUOp = 00  
PCSource = 10  
PCWrite = 1"]; U1 --- U2 --- U3; subgraph OVERFLOW [OVERFLOW]; O1["EPCause = 01  
hwD = 01  
ALUSrcA = 00  
ALUSrcB = 01  
ALUOp = 00  
MemRead = 1"]; O2["WAIT MEMREAD  
EPCWrite = 1  
MDWrite = 1"]; O3["LoadSource = 0  
PCSource = 10  
PCWrite = 1"]; O1 --- O2 --- O3; subgraph DIVBYZERO [DIVBYZERO]; D1["EPCause = 10  
hwD = 01  
ALUSrcA = 00  
ALUSrcB = 01  
ALUOp = 00  
MemRead = 1"]; D2["WAIT MEMREAD  
EPCWrite = 1"]; D3["LoadSource = 0  
PCSource = 100  
PCWrite = 1"]; D1 --- D2 --- D3;
```

The diagram illustrates three execution paths for a program with memory fences, showing the state of registers and memory access flags at different stages.

**UNEXISTENT OP**

- Initial State:
  - EPCause = 00
  - hwD = 01
  - ALUSrcA = 00
  - ALUSrcB = 01
  - ALUOp = 00
  - MemRead = 1
- Wait MemRead State:
  - WAIT MEMREAD
  - EPCWrite = 1
  - MDWrite = 1
- Final State:
  - SignExtendCH = 1
  - ALUSrcA = 10
  - ALUSrcB = 10
  - ALUOp = 00
  - PCSource = 10
  - PCWrite = 1

**OVERFLOW**

- Initial State:
  - EPCause = 01
  - hwD = 01
  - ALUSrcA = 00
  - ALUSrcB = 01
  - ALUOp = 00
  - MemRead = 1
- Wait MemRead State:
  - WAIT MEMREAD
  - EPCWrite = 1
  - MDWrite = 1
- Final State:
  - LoadSource = 0
  - PCSource = 10
  - PCWrite = 1

**DIVBYZERO**

- Initial State:
  - EPCause = 10
  - hwD = 01
  - ALUSrcA = 00
  - ALUSrcB = 01
  - ALUOp = 00
  - MemRead = 1
- Wait MemRead State:
  - WAIT MEMREAD
  - EPCWrite = 1
- Final State:
  - LoadSource = 0
  - PCSource = 100
  - PCWrite = 1

