|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | R-format(4) | Load(5) | Store(4) | Beq(3) | Jumb(3) |
| PC Write | 1 | -- | -- | -- | -- |
| I or D | 0 PC “Address Source” | -- | -- | -- | -- |
| Mem Read | 1 | -- | -- | -- | -- |
| Mem Write | 0 | -- | -- | -- | -- |
| IR Write | 1 | -- | -- | -- | -- |
| PC Source | 00 ALU result | -- | -- | -- | -- |
| ALU OP | 00 Add | -- | -- | -- | -- |
| ALU Src B | 01 4 to ALU | -- | -- | -- | -- |
| ALU Src A | 0 PC to ALU | -- | -- | -- | -- |
| Reg Write | 0 | -- | -- | -- | -- |

1. **Instructions Fetch Step:**

**IR = Memory[PC];  
PC = PC + 4;**

1. **Instructions Decode + Register Fetch step:**

**A = Reg[IR[25-21]];  
B = Reg[IR[20-16]];  
ALUOut = PC + (sign-extend(IR[15-0]) << 2);**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ALU OP | 00 Add | -- | -- | -- | -- |
| ALU Src B | 11 sign/shift | -- | -- | -- | -- |
| ALU Src A | 0 PC to ALU | -- | -- | -- | -- |

1. **Execution step:**

a) Memory reference:  
 **ALUOut = A + sign-extend(IR[15-0]);**

b) Arithmetic-logical reference:  
 **ALUOut = A op B;**

c) Branch:  
**if (A == B) PC = ALUOut;**

d) Jump:  
**PC = PC[31-28] || (IR[25-0] << 2);**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ALU OP | 10 ALU Control | 00 Add | 00 | 01 Sub |  |
| ALU Src B | 00 reg B | 10 sign | 10 | 00 |  |
| ALU Src A | 1 reg A | 1 reg A | 1 | 1 |  |
| PC Source |  |  |  | 01 ALUOut | 10 Jump |
| PC Write Cond |  |  |  | 1 Update PC , if… | 1 Update PC |

1. **Memory Access/ R-Type Completion step:**

Memory reference:

Load: **MDR = Memory[ALUOut];**

Store: **Memory[ALUOut] = B;**

R-type instruction:

**Reg[IR[15-11]] = ALUOut;**

|  |  |  |  |
| --- | --- | --- | --- |
| Mem Read |  | 1 |  |
| I or D |  | 1 “Address from ALU” | 1 |
| IR Write |  | 0 |  |
| Mem Write |  |  | 1 |
| Mem To Reg | 0 ALUOut |  |  |
| Reg Write | 1 |  |  |
| Reg Dst | 1 $rd |  |  |

1. **Read Completion step:**

|  |  |  |
| --- | --- | --- |
| Mem To Reg |  | 1 mem data |
| Reg Write |  | 1 |
| Reg Dst |  | 0 $rt |