

**MARMARA UNIVERSITY**

**FACULTY OF ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**CSE3038**

**Computer Organization**

**Project – 2 Report**

Muhammed F. Eroğlu 150116022

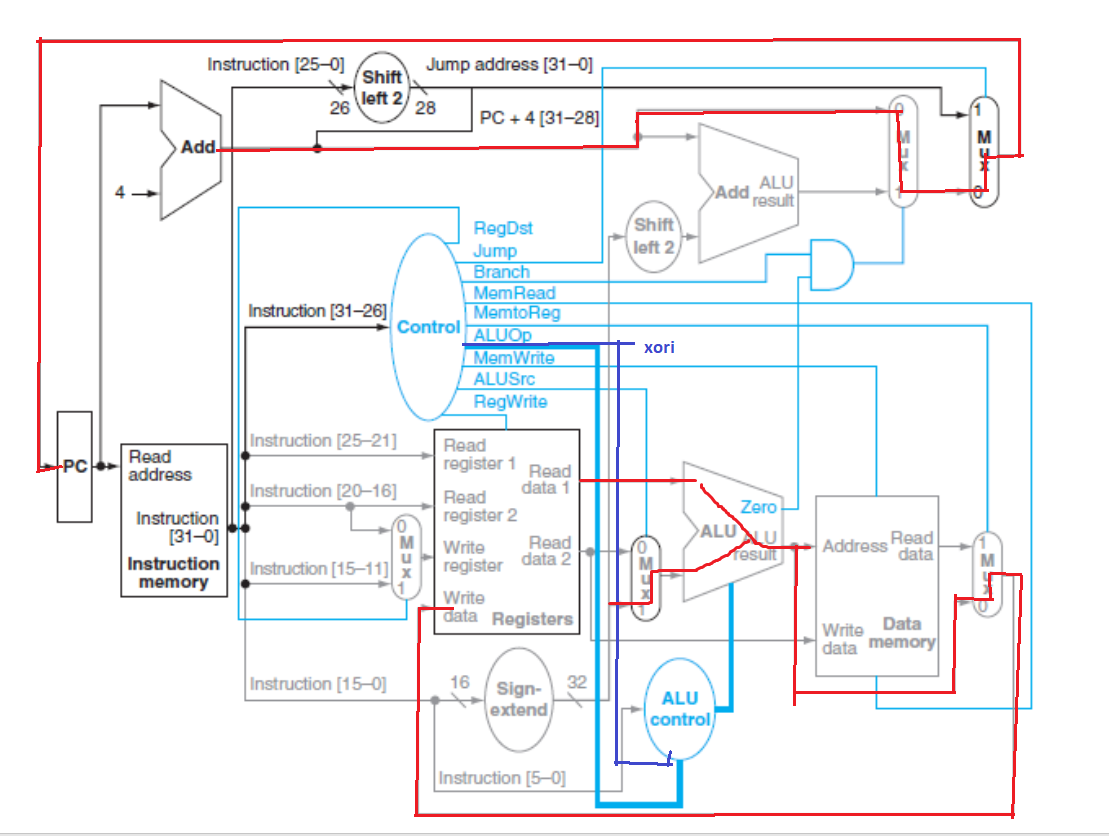
Enes Garip 150116034

XORI

xori I-type , opcode=14 , xori $rt, $rs, Label

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Source Register | Destination Register | Immediate |
| 6 bit (10010) | 5 bit | 5 bit | 16 bit |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| regdest | alusrc | Memtoreg | Regwrite | Memread | Branch | Aluop1 | Aluop2 | Xorsig | Bgezsig | balzsig |
| 0 | 1 | 0 | 1 | X | X | 0 | 0 | 1 | 0 | 0 |



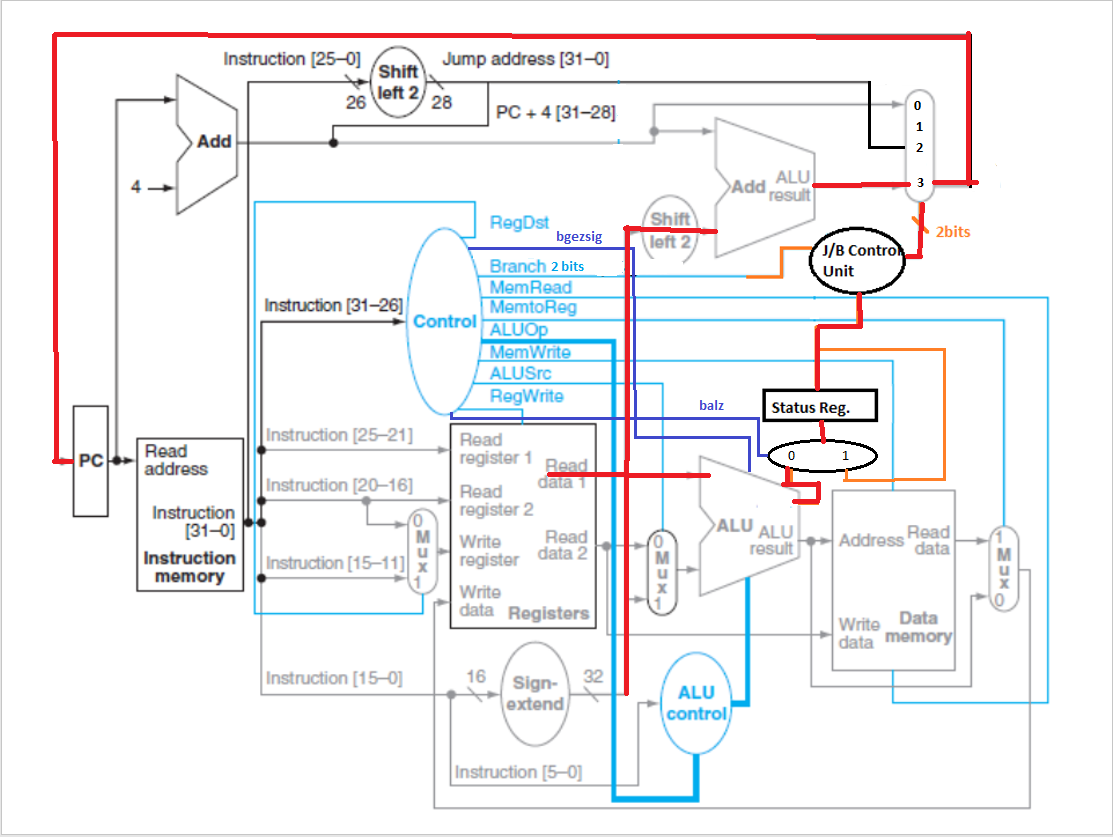
Instruction is fetched from memory. The datapath sets the xori and aluOp signals properly, and then register is read, ALU output is generated. ALU makes logical XOR operation between source register and the zero-extended immediate. The output is put into register $rt. Then, PC ←PC+4.

BGEZ

bgez I-type opcode=39 bgez $rs, Label

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Source Register | Empty | Label |
| 6 bit (10111) | 5 bit | 5 bit | 16 bit |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| regdest | alusrc | Memtoreg | Regwrite | Memread | Branch | Aluop1 | Aluop2 | Xorsig | Bgezsig | balzsig |
| X | 1 | X | 0 | X | 10 | 0 | 0 | 0 | 1 | 0 |



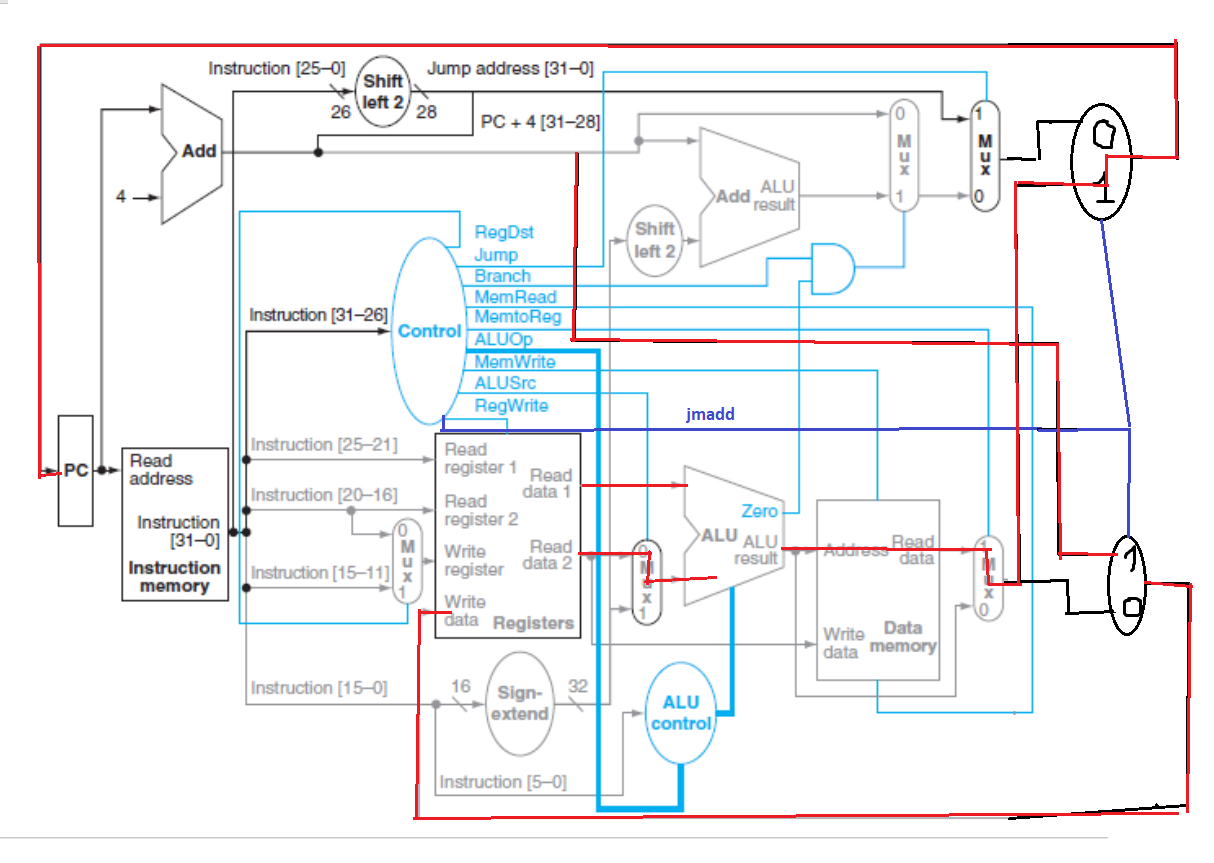
Instruction is fetched from memory. The datapath sets the bgezsig and branch[1:0] properly. If the value in status register is “0” then branch to PC independent address and link address is saved in register 31. New PC ← (Label<<2).

JMADD

jmadd R-type funct=32 jmadd $rs,$rt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Opcode | Source Register | Source Register 2 | Destination  Register | Immediate | Function Code |
| 6 bit (000000) | 5 bit | 5 bit | 5 bit  (111111) | 7 bit | 4 bit  (0001) |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| regdest | alusrc | Memtoreg | Regwrite | Memread | Branch | Aluop1 | Aluop2 | Xorsig | Bgezsig | balzsig |
| 1 | 0 | 1 | 1 | 1 | 00 | 1 | 0 | 0 | 0 | 0 |



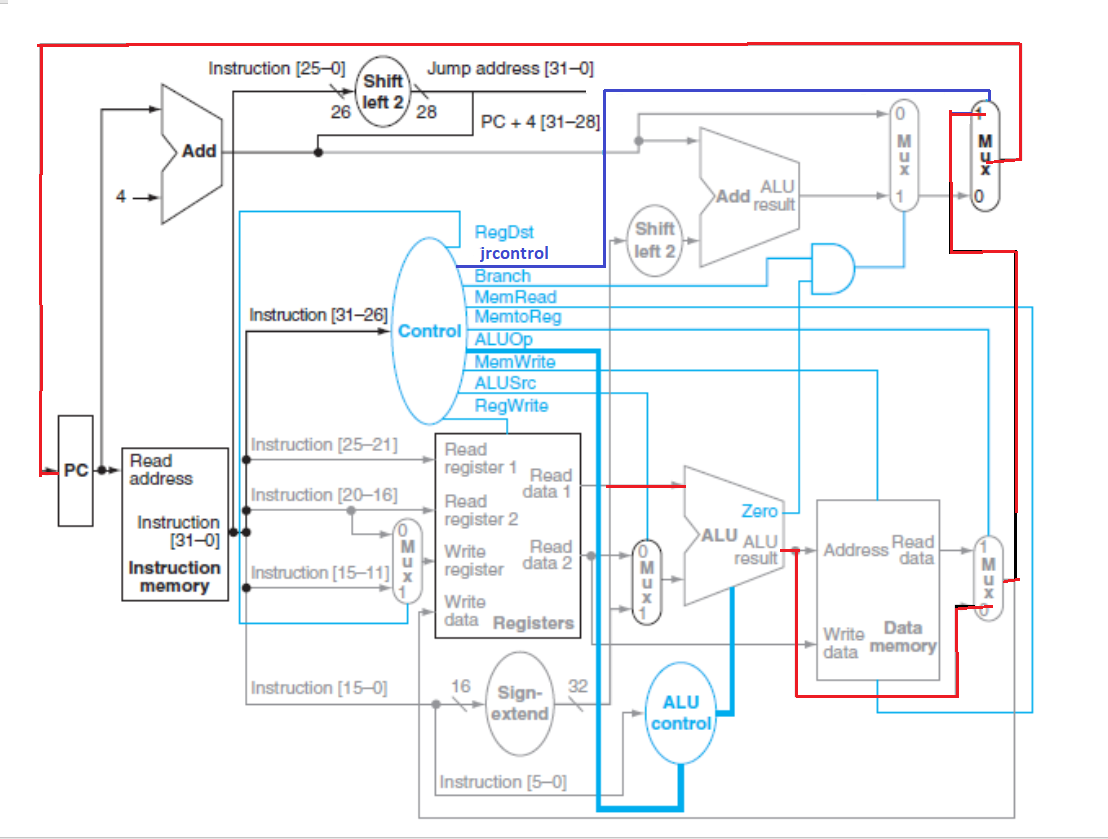
Instruction is fetched from memory. The datapath sets the jmadd signal properly and then registers are read, ALU output is generated. ALU makes logical ADD operation between source register and the source register 2.The output is the address on Data Memory.. The value on this address indicates the address to jump (indirect jump). PC←M.

JR

jr R-type func=8 jr $rs

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Source Register | Empty | Function Code |
| 6 bit (000000) | 5 bit | 17 bit | 4 bit  (0110) |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| regdest | alusrc | Memtoreg | Regwrite | Memread | Branch | Aluop1 | Aluop2 | Xorsig | Bgezsig | balzsig | jrcontrol |
| X | X | 0 | X | X | 00 | 1 | 0 | 0 | 0 | 0 | 1 |



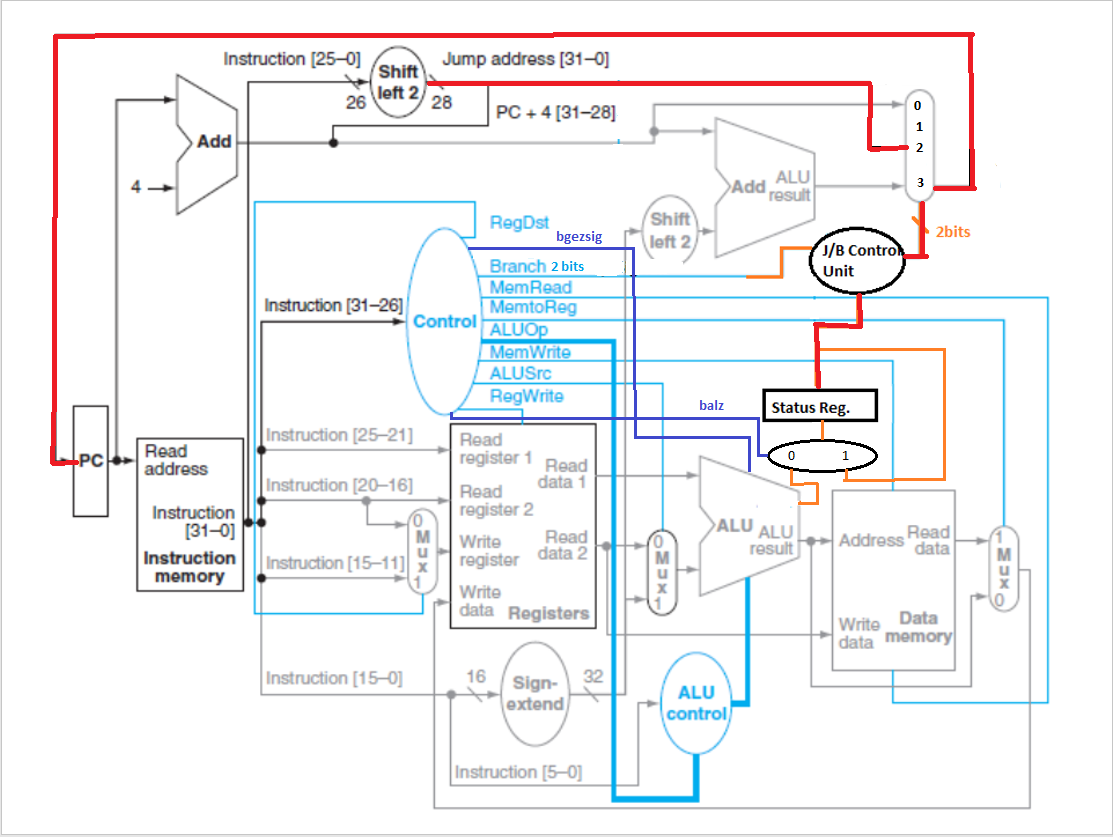
Instruction is fetched from memory. The datapath sets the jrcontrol signal properly and then register is read, ALU output is generated. ALU makes logical ADD operation between source register and “0”. The value of this output indicates the address to jump (indirect jump). PC← Sum

BALZ

balz J-type opcode=26 balz Target

|  |  |
| --- | --- |
| Opcode | Target |
| 6 bit (000000) | 16 bit |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| regdest | alusrc | Memtoreg | Regwrite | Memread | Branch | Aluop1 | Aluop2 | Xorsig | Bgezsig | balzsig | jrcontrol |
| X | X | 0 | X | X | 01 | X | X | 0 | 0 | 1 | 0 |



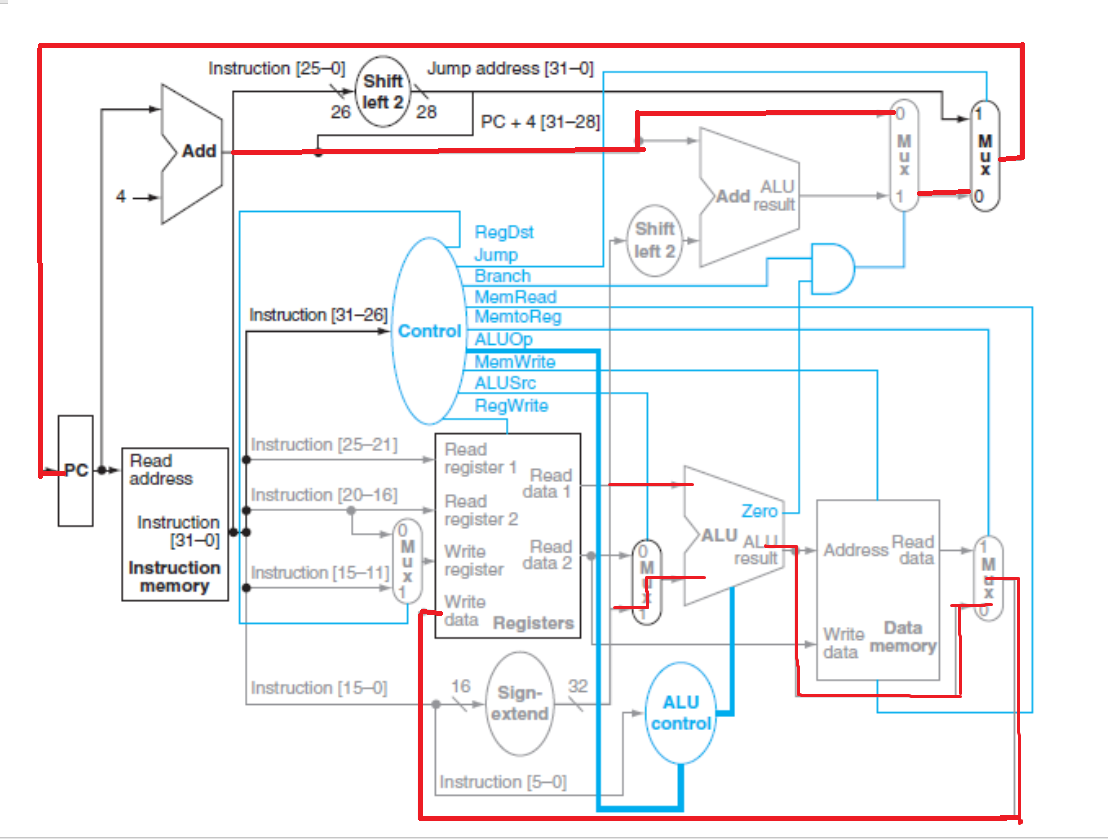
Instruction is fetched from memory. The datapath sets the balz and branch[1:0] signals properly. Mux under the status register doesn’t allow to change the data in register in instruction balz.If the value in status register is “0” then branch to PC independent address and link address is saved in register 31. New PC ← (Label<<2).

SLL

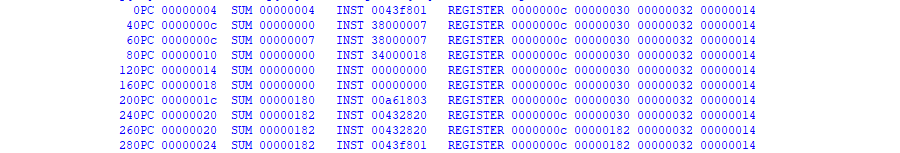
sll R-type func=0011=3 sll $rd, $rt, shamt

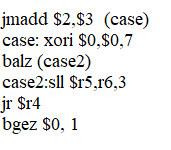
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Opcode | Source  Register | Destination Register | Empty | Immediate | Function |
| 6 bit (000000) | 5 bit | 5 bit | 5 bit | 7 bit | 4 bit  (0011) |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| regdest | alusrc | Memtoreg | Regwrite | Memread | Branch | Aluop1 | Aluop2 | Xorsig | Bgezsig | balzsig | jrcontrol |
| 0 | 1 | 0 | 1 | X | 00 | 1 | 0 | 0 | 0 | 0 | 0 |



Instruction is fetched from memory. The datapath sets the aluOp signal properly, and then register is read, ALU output is generated. ALU makes logical SLL operation between source register and the zero-extended immediate. The output is put into destination register. Then, PC ←PC+4.



This is the result of the following code :

Note: The drawings are for each instruction. Processor contains all of them. We draw them separately in order to be more understandable.