Computer Organization and Architecture Laboratory

KGP RISC PROCESSOR

Group 81

Anand Manojkumar Parikh (20CS10007)

Soni Aditya Bharatbhai (20CS10060)

Instruction Formats:

There are 4 instruction formats supported by this processor.

The memory is assumed to have 32-bit addresses (232 words).

A total of 32 registers (each of 32 bits) are present in the Register File. This choice was made following the MIPS convention. Also, having more than 32 registers is also not advisable from a financial perspective.

Each instruction has 32 bits. The design follows single-cycle execution of each instruction. The instruction formats are explained below-

1) R-format [Register Format]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| opcode | dest\_reg | source\_reg | shamt | fncode |
| 5 | 5 | 5 | 12 | 5 |

* Operation Code : 5 bits
* Address of 1st register (destination) in Regfile : 5 bits
* Address of 2nd register (source) in Regfile : 5 bits
* Shift amount : 12 bits \*
* Function Code : 5 bits

[\* shift amount is restricted to 4095 (212-1). However, any value more than 32 useless. So, only 6 bits are useful, the remaining 6 are dummy/redundant]

2) I-format [Immediate Format]

|  |  |  |  |
| --- | --- | --- | --- |
| opcode | dest\_reg | source\_reg | imm |
| 5 | 5 | 5 | 17 |

* Operation Code : 5 bits
* Address of 1st register (destination) in Regfile : 5 bits
* Address of 2nd register (source) in Regfile : 5 bits
* Immediate value : 17 bits [217-1 , -217]

3) B-format [Branch Format]

|  |  |  |  |
| --- | --- | --- | --- |
| opcode | source\_reg | branch\_addr | fncode |
| 5 | 5 | 17 | 5 |

* Operation Code : 5 bits
* Address of 1st register (source register) in Regfile : 5 bits
* Branch Address : 17 bits \*
* Function Code : 5 bits

[\* The branch address is only allocated 17 bits in the instruction. This branch address is designed to be PC-relative. If we want to branch (conditionally) to address L, then we first find the value L – PC – 1, and we pass the 17 least significant bits of this value in the instruction. The processor then sign extends this value by 15 bits (to make it 32 bits long) and adds PC+1 to it to finally get a valid 32-bit address. This restricts our conditional branching to [-216+1,216 ] w.r.t. the PC. This was done to get a symmetric choice while branching.]

4) J-format [Jump Format]

|  |  |
| --- | --- |
| opcode | jump\_address |
| 5 | 27 |

* Operation Code : 5 bits
* Jump address : 27 bits \*

[\* The jump address is only allocated 27 bits in the instruction. This jump address is designed to be pseudo-direct. If we want to jump or branch (unconditionally) to address L, then we simply take the 27 least significant bits of L and pass it in the instruction. The processor then concatenates the 5 most significant bits of the PC to this value to construct a valid 32-bit address. This allows branching to distant memory addresses.]

Opcodes and Function Codes:

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Instruction Name | Opcode | Fncode\* |
| Add | add | 0 | 0 |
| Complement | comp | 0 | 1 |
| Add Immediate | addi | 1 | X |
| Complement Immediate | compi | 2 | X |
| AND | and | 0 | 6 |
| XOR | xor | 0 | 7 |
| Shift Left Logical | shll | 13 | 0 |
| Shift Right Logical | shrl | 13 | 1 |
| Shift Left Logical Variable | sllv | 0 | 2 |
| Shift Right Logical Variable | shrlv | 0 | 3 |
| Shift Right Arithmetic | shra | 13 | 2 |
| Shift Right Arithmetic Variable | shrav | 0 | 4 |
| Load Word | lw | 5 | X |
| Store Word | sw | 6 | X |
| Unconditional Branch | b | 8 | X |
| Branch Register | br | 7 | 0 |
| Branch on less than zero | bltz | 7 | 1 |
| Branch on flag zero | bz | 7 | 2 |
| Branch on flag not zero | bnz | 7 | 3 |
| Branch and Link | bl | 9 | X |
| Branch on Carry | bcy | 10 | X |
| Branch on No Carry | bncy | 11 | X |
| Diff | diff | 0 | 5 |

[\* Function Code ‘X’ denotes that the instruction has a unique Opcode assigned to it. Hence, no extensions are required to decode that instruction.]

Control Signals:

There are 3 control units in the processor –

1) Main Control Unit

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Opcode | RegDst | Write31 | RegWrite | ALUOp (2) | Shiftsel(2) | ALUsrc | MemRead | MemWrite | MemtoReg | BranchOp(3) |
| 00000 | 0 | 0 | 1 | 00 | XX | 0 | 0 | 0 | 0 | 000 |
| 00001 | 0 | 0 | 1 | 01 | 01 | 1 | 0 | 0 | 0 | 000 |
| 00010 | 0 | 0 | 1 | 10 | 01 | 1 | 0 | 0 | 0 | 000 |
| 00101 | 1 | 0 | 1 | 01 | 01 | 1 | 1 | 0 | 1 | 000 |
| 00110 | X | X | 0 | 01 | 01 | 1 | 0 | 1 | X | 000 |
| 00111 | X | X | 0 | XX | 10 | X | 0 | 0 | X | 001 |
| 01000 | X | X | 0 | XX | XX | X | 0 | 0 | X | 010 |
| 01001 | X | 1 | 1 | XX | XX | X | 0 | 0 | X | 010 |
| 01010 | X | X | 0 | XX | XX | X | 0 | 0 | X | 100 |
| 01011 | X | X | 0 | XX | XX | X | 0 | 0 | X | 101 |
| 01101 | 0 | 0 | 1 | 11 | 00 | 1 | 0 | 0 | 0 | 000 |

Function of each control signal

1. RegDst: Decides which line drives the destination register address line in the Regfile (along with Write31)
2. Write31: Should we write to register 31 or not
3. RegWrite: Activates Regfile. Any register in it can be written only when this control is 1.
4. ALUOp: Partially decodes which ALU operation to execute\*\*.
5. Shiftsel: Select which line to sign-extend by 15 bits.
6. ALUsrc: Decides which line drives the right port of the ALU.
7. MemRead: Should we read from memory or not.
8. MemWrite: Activate RAM. Any location in the RAM can be written only when this control is 1.
9. MemtoReg: Select which line feeds into the final output : Memory output or ALU output.

10) BranchOp: Partially decodes which of the possible branches does the next PC take\*\*.

[\*\* The design follows 2 stage decoding in some situations to avoid complex multiplexers. The ALUOp and BranchOp are decided by the Main Control Unit solely on basis of the OpCode and they, in turn, along with the Function Code help the ALU Control and Branch Control respectively to make the final decision about which ALU operation to execute and which branch to take.]

2) ALU Control

[This unit alone drives the ALU]

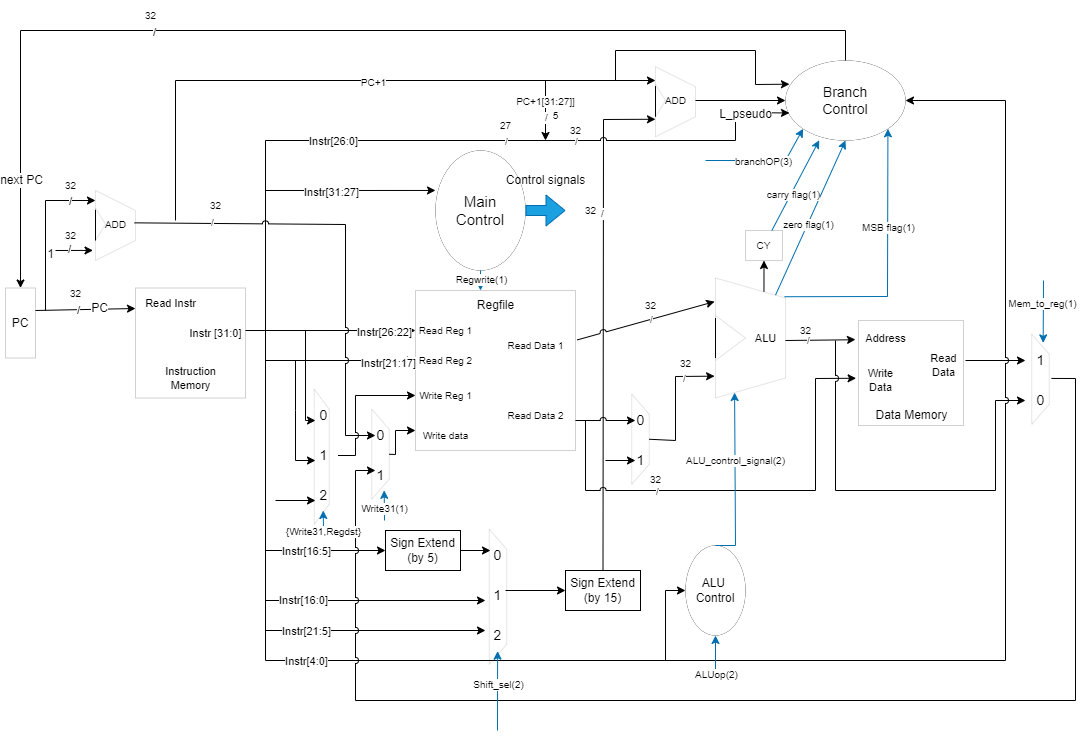
|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | Operation |
| ALUOp(2) | FnCode(5) | ALU\_control(3) |
| 00 | 00000 | 000 | add |
| 00 | 00001 | 001 | comp |
| 00 | 00010 | 010 | shllv |
| 00 | 00011 | 111 | shrlv |
| 00 | 00100 | 100 | shrav |
| 00 | 00101 | 101 | diff |
| 00 | 00110 | 110 | and |
| 00 | 00111 | 111 | xor |
| 01 | X | 000 | add |
| 10 | X | 001 | comp |
| 01 | X | 000 | add |
| 01 | X | 000 | add |
| 11 | 0 | 010 | shll |
| 11 | 1 | 011 | shrl |
| 11 | 2 | 100 | shra |

3) Branch Control

[This unit alone chooses the next PC]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Input | | | | | Output |
| BranchOp(3) | FnCode | MSB | Zero | Carry |
| 0 | X | X | X | X | PC+4 |
| 1 | 0 | X | X | X | rs |
| 1 | 1 | 1 | X | X | PC+4+L |
| 1 | 1 | 0 | X | X | PC+4 |
| 1 | 2 | X | 1 | X | PC+4+L |
| 1 | 2 | X | 0 | X | PC+4 |
| 1 | 3 | X | 0 | X | PC+4+L |
| 1 | 3 | X | 1 | X | PC+4 |
| 2 | X | X | X | X | L |
| 4 | X | X | X | 1 | L |
| 4 | X | X | X | 0 | PC+4 |
| 5 | X | X | X | 0 | L |
| 5 | X | X | X | 1 | PC+4 |

Processor Diagram:



ALU Diagram:

