**컴퓨터 구조**

**프로젝트 1**

**이성원 교수님**

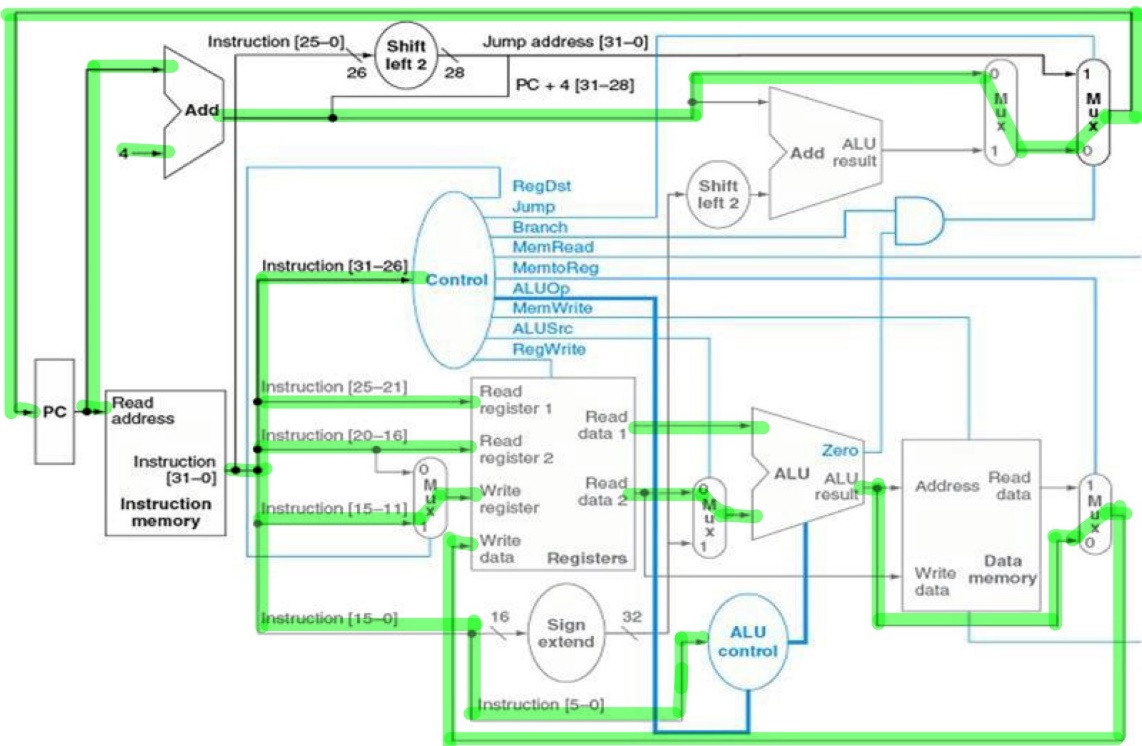
**컴퓨터정보공학부**

**2018202076**

**이연걸**

* **문제의 해석 및 해결 방향**
* sra

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Opcode/Function** | **Syntax** | **Operation** |
| sra | 000011 | f $d, $t, sa | $d = $t >>> a |



sra는 R type Instruction으로, $t를 a만큼 shift right한 값을 $d에 저장하는 명령어이다. shift right할 때 만약 $t의 MSB가 0이면 0을 채워주고 1이면 1을 채워준다. 이는 부호를 구분하기 위해 사용되고, 같은 기능을 하지만 부호를 구분하지 않는 srl 명령어가 존재한다.

sra 명령어가 Instruction memory에서 나오면 $t의 값과 a의 값이 register에서 나와서 ALU로 들어간다. ALU에서 shift 연산을 마치면 그 값이 레지스터($d)로 들어오면서 shift right를 수행한다. Adder를 통해 현재 PC의 값과 4가 더해져 다음 Instruction을 받아온다.

만약 sra가 I type instruction이 되어 상수를 이용한 연산을 할 수 있었다면 ALU에서 $t의 값과 sign extend한 immediate 값을 이용해 $t의 값을 조정한 후 $d에 넣어줄 수 있었을 것이다.

* jalr

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Opcode/Function** | **Syntax** | **Operation** |
| jalr | 001001 | f labelR | $31 = pc; pc = $s |

jalr은 R type instruction으로 다음 명령어의 주소(PC + 4)를 $31에 저장한 뒤 labelR로 점프한다.

* subu

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Opcode/Function** | **Syntax** | **Operation** |
| subu | 1000011 | f $d. $s. $t | $d = $s - $t |

* xor

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Opcode/Function** | **Syntax** | **Operation** |
| xor | 100110 | f $d, $s, $t | $d = $s ^ $t |

* bne

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Opcode/Function** | **Syntax** | **Operation** |
| bne | 000101 | o $s, $t, label | If ($s != $t) pc += i << 2 |

* blez

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Opcode/Function** | **Syntax** | **Operation** |
| blez | 000110 | o $s, label | If ($s <= 0) pc += i << 2 |

* sltiu

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Opcode/Function** | **Syntax** | **Operation** |
| sltiu | 001011 | o $d, $s, i | $t = ($s < ZE(i)) |

* andi

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Opcode/Function** | **Syntax** | **Operation** |
| andi | 001100 | o $t, $s, i | $t = $s & ZE(i) |

* lbu

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Opcode/Function** | **Syntax** | **Operation** |
| lbu | 100100 | o $t, I ($s) | $t = ZE (MEM [$s + i]:1) |

* sb

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Opcode/Function** | **Syntax** | **Operation** |
| sb | 101000 | o $t, I ($s) | MEM [$s + i]:1 = LB ($t) |

* **설계 의도와 방법**
* sra
* jalr
* subu
* xor
* bne
* blez
* sltiu
* andi
* lbu
* sb