컴퓨터구조 프로젝트 보고서

Project 1

MIPS Single Cycle CPU Implementation

수업 명: 컴퓨터구조

담당 교수: 이성원 교수님

학과: 컴퓨터정보공학부

학번: 2023202070

이름: 최현진

제출일: 2025.04.

1. Introduction

기본 명령어 lw, sw, ori, j, lui, llo, lhi는 구현되어 있고, 추가로 AND, NOR, ADDI, SLTU, SRL, SH, LB, BNE, BGEZ, JALR을 설계해야 한다. 이를 위해 PLA(Programmable Logic array) 구조를 이용해서 명령어를 해석하고, 제어 신호를 생성해야 한다. PLA\_AND.txt에는 각 명령어에 대해 opcode, func, regimm를 설정한다. PLA\_OR.txt에는 설정한 줄에 맞춰 해당 명령이 작동할 제어 신호를 채워 넣는다.

2. Assignment

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. AND  R-type  Syntax: f $d, $s, $t  Operation: $d = $s & $t   |  |  |  | | --- | --- | --- | | Op | Func | Regimm | | 000000 | 100100 | xxxxx |   000000\_100100\_xxxxx // 0x15 : and   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | RegDst | RegDatSel | RegWrite | SEUmode | ALUsrcB | ALUctrl | | 01 | 00 | 1 | x | 00 | 00 | | ALUop | DataWidth | MemWrite | MemtoReg | Branch | Jump | | 00000 | xxx | 0 | 0 | 000 | 00 |   RegDst: 01, 결과값을 rd에 쓸 것이므로 목적지 레지스터로 rd를 선택한다.  RegDatSel: 00, ALU 결과값을 register file에 쓰도록 선택한다.  RegWrite: 1, register file에 값을 쓴다.  SEUmode: x, imm값을 extension할 필요가 없다.  ALUsrcB: ALU 입력으로 register file port B를 사용한다.  ALUctrl: 00, ALUctrl[1]=0 (normal ALU input), ALUctrl[0]=x (shift 수행하지 않음)  ALUop: 00000, bitwise AND  DataWidth: xxx, 메모리 접근하지 않음.  MemWrite: 0, 메모리에 쓰지 않음.  MemtoReg: 0, 메모리 접근하지 않음.  Branch: 000, branch 수행하지 않음.  Jump: 00, jump 수행하지 않음. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2. NOR  R-type  Syntax: f $d, $s, $t  Operation: $d = ~($s | $t)   |  |  |  | | --- | --- | --- | | Op | Func | Regimm | | 000000 | 100111 | xxxxx |   000000\_100111\_xxxxx // 0x18 : nor   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | RegDst | RegDatSel | RegWrite | SEUmode | ALUsrcB | ALUctrl | | 01 | 00 | 1 | x | 00 | 00 | | ALUop | DataWidth | MemWrite | MemtoReg | Branch | Jump | | 00010 | xxx | 0 | 0 | 000 | 00 |   RegDst: 01, 결과값을 rd에 쓸 것이므로 목적지 레지스터로 rd를 선택한다.  RegDatSel: 00, ALU 결과값을 register file에 쓰도록 선택한다.  RegWrite: 1, register file에 값을 쓴다.  SEUmode: x, imm값을 extension할 필요가 없다.  ALUsrcB: ALU 입력으로 register file port B를 사용한다.  ALUctrl: 00,  ALUop: 00000, bitwise AND  DataWidth: xxx, 메모리 접근하지 않음.  MemWrite: 0, 메모리에 쓰지 않음.  MemtoReg: 0, 메모리 접근하지 않음.  Branch: 000, branch 수행하지 않음.  Jump: 00, jump 수행하지 않음. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3. ADDI  I-type  Syntax: o $t, $s, i  Operation: $t = $s + SE(i)   |  |  |  | | --- | --- | --- | | Op | Func | Regimm | | 001000 | xxxxxx | xxxxx |   001000\_xxxxxx\_xxxxx // 0x23 : addi   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | RegDst | RegDatSel | RegWrite | SEUmode | ALUsrcB | ALUctrl | |  |  |  |  |  |  | | ALUop | DataWidth | MemWrite | MemtoReg | Branch | Jump | |  |  |  |  |  |  |   RegDst:  RegDatSel:  RegWrite:  SEUmode:  ALUsrcB:  ALUctrl:  ALUop:  DataWidth:  MemWrite:  MemtoReg:  Branch:  Jump: |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4. SLTU  R-type  Syntax: f $d, $s, $t  Operation: $d = ($s < $t)   |  |  |  | | --- | --- | --- | | Op | Func | Regimm | | 000000 | 101011 | xxxxx |   000000\_101011\_xxxxx // 0x1a : sltu   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | RegDst | RegDatSel | RegWrite | SEUmode | ALUsrcB | ALUctrl | |  |  |  |  |  |  | | ALUop | DataWidth | MemWrite | MemtoReg | Branch | Jump | |  |  |  |  |  |  |   RegDst:  RegDatSel:  RegWrite:  SEUmode:  ALUsrcB:  ALUctrl:  ALUop:  DataWidth:  MemWrite:  MemtoReg:  Branch:  Jump: |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5. SRL  Shift Right Logical  R-type  Syntax: f $d, $t, sa  Operation: $d = $t >> a   |  |  |  | | --- | --- | --- | | Op | Func | Regimm | | 000000 | 000010 | xxxxx |   000000\_000010\_xxxxx // 0x01 : srl   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | RegDst | RegDatSel | RegWrite | SEUmode | ALUsrcB | ALUctrl | |  |  |  |  |  |  | | ALUop | DataWidth | MemWrite | MemtoReg | Branch | Jump | |  |  |  |  |  |  |   RegDst:  RegDatSel:  RegWrite:  SEUmode:  ALUsrcB:  ALUctrl:  ALUop:  DataWidth:  MemWrite:  MemtoReg:  Branch:  Jump: |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6. SH  I-type  Syntax: o $t, i ($s)  Operation: MEM [$s + i]:2 = LH ($t)   |  |  |  | | --- | --- | --- | | Op | Func | Regimm | | 101001 | xxxxxx | xxxxx |   101001\_xxxxxx\_xxxxx // 0x31 : sh   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | RegDst | RegDatSel | RegWrite | SEUmode | ALUsrcB | ALUctrl | |  |  |  |  |  |  | | ALUop | DataWidth | MemWrite | MemtoReg | Branch | Jump | |  |  |  |  |  |  |   RegDst:  RegDatSel:  RegWrite:  SEUmode:  ALUsrcB:  ALUctrl:  ALUop:  DataWidth:  MemWrite:  MemtoReg:  Branch:  Jump: |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 7. LB  I-type  Syntax: o $t, i ($s)  Operation: $t = SE (MEM [$s + i]:1)   |  |  |  | | --- | --- | --- | | Op | Func | Regimm | | 100000 | xxxxxx | xxxxx |   100000\_xxxxxx\_xxxxx // 0x2b : lb   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | RegDst | RegDatSel | RegWrite | SEUmode | ALUsrcB | ALUctrl | |  |  |  |  |  |  | | ALUop | DataWidth | MemWrite | MemtoReg | Branch | Jump | |  |  |  |  |  |  |   RegDst:  RegDatSel:  RegWrite:  SEUmode:  ALUsrcB:  ALUctrl:  ALUop:  DataWidth:  MemWrite:  MemtoReg:  Branch:  Jump: |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8. BNE  Branch Not Equal  I-type  Syntax: o $s, $t, label  Operation: l if ($s != $t) pc += i << 2   |  |  |  | | --- | --- | --- | | Op | Func | Regimm | | 000101 | xxxxxx | xxxxx |   000101\_xxxxxx\_xxxxx // 0x20 : bne   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | RegDst | RegDatSel | RegWrite | SEUmode | ALUsrcB | ALUctrl | |  |  |  |  |  |  | | ALUop | DataWidth | MemWrite | MemtoReg | Branch | Jump | |  |  |  |  |  |  |   RegDst:  RegDatSel:  RegWrite:  SEUmode:  ALUsrcB:  ALUctrl:  ALUop:  DataWidth:  MemWrite:  MemtoReg:  Branch:  Jump: |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9. BGEZ  I-type, RegImm-type: opcode=000001 고정, rt(=regimm)로 명령어 구분  Syntax: r $s, label  Operation: if ($s >= 0) pc += i << 2   |  |  |  | | --- | --- | --- | | Op | Func | Regimm | | 000001 | xxxxxx | 00001 |   000001\_xxxxxx\_00001 // 0x1c : bgez   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | RegDst | RegDatSel | RegWrite | SEUmode | ALUsrcB | ALUctrl | |  |  |  |  |  |  | | ALUop | DataWidth | MemWrite | MemtoReg | Branch | Jump | |  |  |  |  |  |  |   RegDst:  RegDatSel:  RegWrite:  SEUmode:  ALUsrcB:  ALUctrl:  ALUop:  DataWidth:  MemWrite:  MemtoReg:  Branch:  Jump: |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10. JALR  Jump And Link Register  PC를 저장하고 레지스터가 가리키는 주소로 점프  R-type  Syntax: f labelR  Operation: $31 = pc; pc = $s   |  |  |  | | --- | --- | --- | | Op | Func | Regimm | | 000000 | 001001 | xxxxx |   000000\_001001\_xxxxx // 0x07 : jalr   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | RegDst | RegDatSel | RegWrite | SEUmode | ALUsrcB | ALUctrl | |  |  |  |  |  |  | | ALUop | DataWidth | MemWrite | MemtoReg | Branch | Jump | |  |  |  |  |  |  |   RegDst:  RegDatSel:  RegWrite:  SEUmode:  ALUsrcB:  ALUctrl:  ALUop:  DataWidth:  MemWrite:  MemtoReg:  Branch:  Jump: |