1. Instruction Format

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| R-type | | | | | | | | | |
| funct7 | rs2 | | rs1 | | funct3 | | rd | | opcode |
| 7 | 5 | | 5 | | 3 | | 5 | | 7 |
| [31:25] | [24:20] | | [19:15] | | [14:12] | | [11:7] | | [6:0] |
| I-type | | | | | | | | | |
| immediate | | rs1 | | funct3 | | rd | | opcode | |
| 12 | | 5 | | 3 | | 5 | | 7 | |
| [31:20] | | [19:15] | | [14:12] | | [11:7] | | [6:0] | |
| S-type | | | | | | | | | |
| imm[11:5] | rs2 | | rs1 | | funct3 | | imm[4:0] | | opcode |
| 7 | 5 | | 5 | | 3 | | 5 | | 7 |
| [31:25] | [24:20] | | [19:15] | | [14:12] | | [11:7] | | [6:0] |
| SB-type | | | | | | | | | |
| imm[12|10:5] | rs2 | | rs1 | | funct3 | | imm[4:1|11] | | opcode |
| 7 | 5 | | 5 | | 3 | | 5 | | 7 |
| [31:25] | [24:20] | | [19:15] | | [14:12] | | [11:7] | | [6:0] |

1. ALU-control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| funct7 | funct3 | opcode | function | ALU-control |
| 0000000 | 110 | 0110011 | OR | 000 |
| 0000000 | 111 | 0110011 | AND | 001 |
| 0000000 | 000 | 0110011 | ADD | 010 |
| 0100000 | 000 | 0110011 | SUB | 011 |
| 0000001 | 000 | 0110011 | MUL | 100 |
| X | 000 | 0010011 | ADDI | 010 |

|  |  |
| --- | --- |
| ALU action | ALU-control |
| AND | 001 |
| OR | 010 |
| ADD | 011 |
| SUB | 100 |
| MUL | 101 |

for ALU-op,

00 means addition

01 means subtraction

10 depends on function code

11 not used

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| R-type | | | | | |
| opcode | instruction | funct7 | funct3 | ALU-action | ALU-control |
| 0110011 | and | 0000000 | 111 | AND | 001 |
| 0110011 | or | 0000000 | 110 | OR | 010 |
| 0110011 | add | 0000000 | 000 | ADD | 011 |
| 0110011 | sub | 0100000 | 000 | SUB | 100 |
| 0110011 | mul | 0000001 | 000 | MUL | 101 |
| I-type | | | | | |
| 0010011 | addi | X | 000 | ADD | 011 |
| 0000011 | lw | X | 010 | ADD | 011 |
| S-type | | | | | |
| 0100011 | sw | X | 010 | ADD | 011 |
| SB-type | | | | | |
| 1100011 | beq | X | 000 | SUB | 100 |

|  |  |  |
| --- | --- | --- |
| funct\_i | ALU-action | ALU-control |
| 0011 | AND | 001 |
| 0010 | OR | 010 |
| 0000 | ADD | 011 |
| 1000 | SUB | 100 |
| 0100 | MUL | 101 |

如果是R-type, ALU\_Op = 10

這時看function code: funct\_i = {inst[30], inst[25], inst[13], inst[12]};(4-bit)

由上表得ALU\_Control

或者，柏序的做法:

function code: funct\_i = {inst[30], inst[25], inst[14], inst[13], inst[12]};(5-bit)

由下表得ALU\_Control

|  |  |  |
| --- | --- | --- |
| funct\_i | ALU-action | ALU-control |
| 00111 | AND | 001 |
| 00110 | OR | 010 |
| 00000 | ADD | 011 |
| 10000 | SUB | 100 |
| 01000 | MUL | 101 |

如果不是R-type

ALU 只會做加或減

ALU\_Op = 00 代表加 (ALU\_Control = 011)

ALU\_Op = 01 代表減 (ALU\_Control = 100)

1. Control Signal

opcode 只要看前3bit 即可 (inst[6:4])

既然如此在每個pipeline register我們都傳這三個bit

要用到signal時當場decode

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | R-type | addi | lw | sw | beq |
| ID | Branch | 0 | 0 | 0 | 0 | 1 |
| EX | ALUOp | 10 | 00 | 00 | 00 | 01 |
| ALUSrc | 0 | 1 | 1 | 1 | 0 |
| M | MemRead | 0 | 0 | 1 | 0 | 0 |
| MemWrite | 0 | 0 | 0 | 1 | 0 |
| WB | RegWrite | 1 | 1 | 1 | 0 | 0 |
| MemtoReg | 0 | 0 | 1 | X | X |
| opcode | | 0110011 | 0010011 | 0000011 | 0100011 | 1100011 |

柏序稱MemtoReg 為 reg\_src\_WB