##### 1. Instruction list

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction** | **PC** | **Coding** | **Functionality** | **Example** | |
| init Rx, imm | PC++ | 000 xx ii | Rx = MUX[imm]  Imm will go into a MUX to select specific hardwrited number(0,1,6,108) |  |  |
| ld Rx, Ry | PC++ | 001 xx yy | Rx = Mem[Ry] |  |  |
| st Rx, Ry | PC++ | 010 xx yy | Mem[Ry] = Rx |  |  |
| add Rx, Ry | PC++ | 011 xx yy | Rx = Rx + Ry |  |  |
| jpu1 Rx,Ry,imm | if Rx < Ry:  PC == MUX(imm)  else:  PC++ | 100 x y ii | Rx ∈ {R0, R1}  x = 0 1  Ry ∈ {R2, R3}  y = 0 1  Imm number will go into a MUX to select specific jumps | MUX  00 9  01 6  10 24  11 18 |  |
| jpu2 Rx,Ry,imm | if Rx < Ry:  PC == MUX(imm)  else:  PC++ | 101 x y ii | Rx ∈ {R2, R3}  Ry ∈ {R0, R1}  Same as jpu1, but the decoding of registers changed. | MUX  00 14  01 8  10 27 |  |
| subR3 Rx | PC++ | 11100 xx | R3 = R3 - Rx |  |  |
| inc Rx | PC++ | 11101 xx | Rx = Rx + 1 |  |  |
| R3x6 | PC++ | 1111110 | R3 = R3 \* 6 |  |  |
| score | PC++ | 1111111 | R3 = the match score of R3 and R2.  This function is done using logic circuit. |  |  |

##### 2. Register Design

|  |  |
| --- | --- |
| Register Name | Number |
| R0 | 00 |
| R1 | 01 |
| R2 | 10 |
| R3 | 11 |

##### 3. Control Flow

Since there are just several branches used in our Program1 and Program2, the instruction address of all these branches are constant, we save all these address into a MUX, and use the immediate number in the beq instructions to select it.

Accordingly, there is no need for us to calculate the target addresses.

##### 4. Memory Model

#### 4.1 Data Memory

* 16-bit double-byte addressable
* 128memory units in total
* using 7-bit address.

|  |  |
| --- | --- |
| Address | Memory |
| 000 0000 | Mem[0] |
| 000 0001 | Mem[1] |
| ... | ... |
| 111 1111 | Mem[127] |

#### 4.2 Instruction Memory

* 8-bit byte addressable, PC is initialized at 0
* 64 memory units in total
* using 6-bit address.

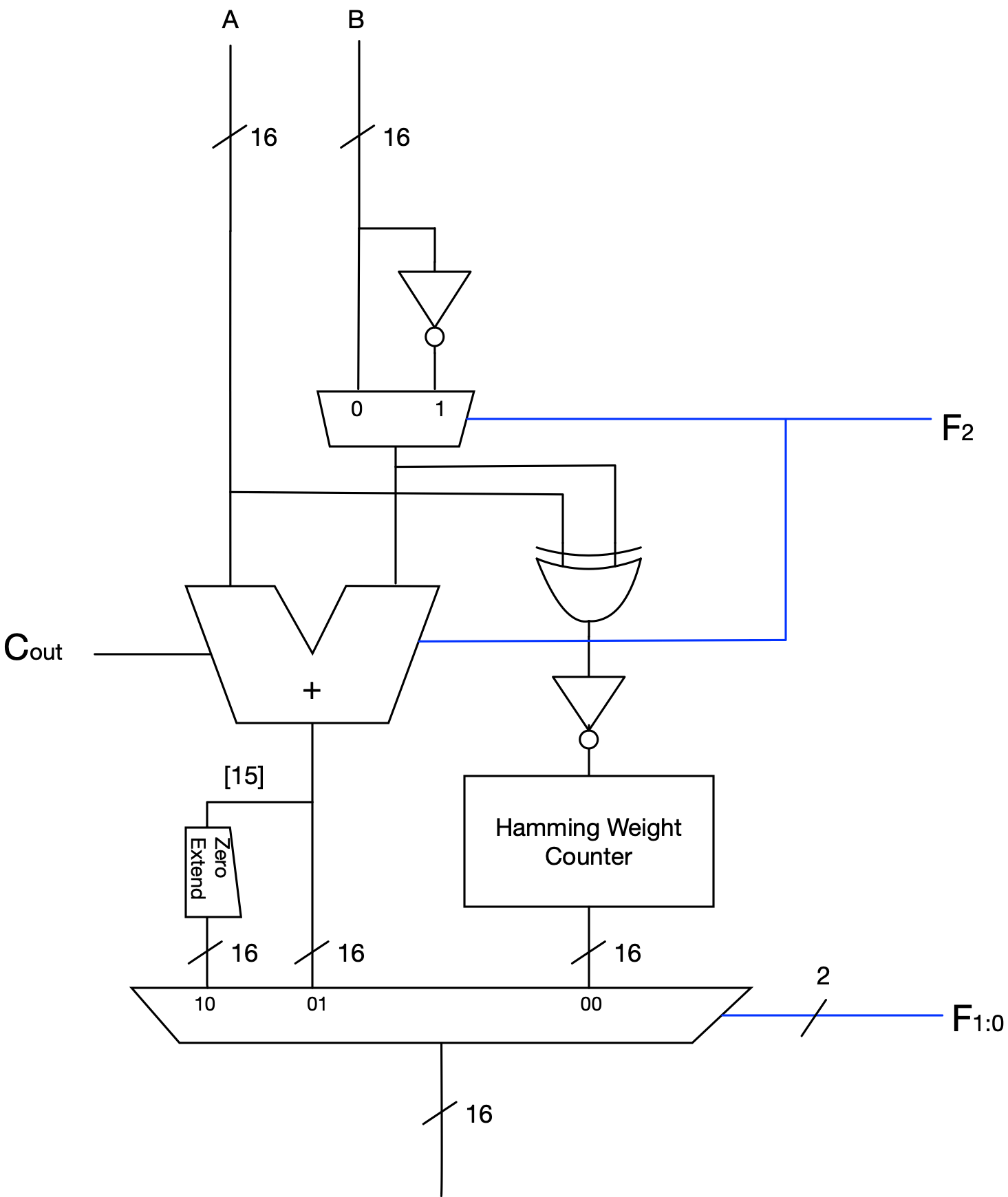
|  |  |
| --- | --- |
| Address | Memory |
| 00 0000 | Mem[0] |
| 00 0001 | Mem[1] |
| ... | ... |
| 11 1111 | Mem[63] |

### Part B. Answers to Questions

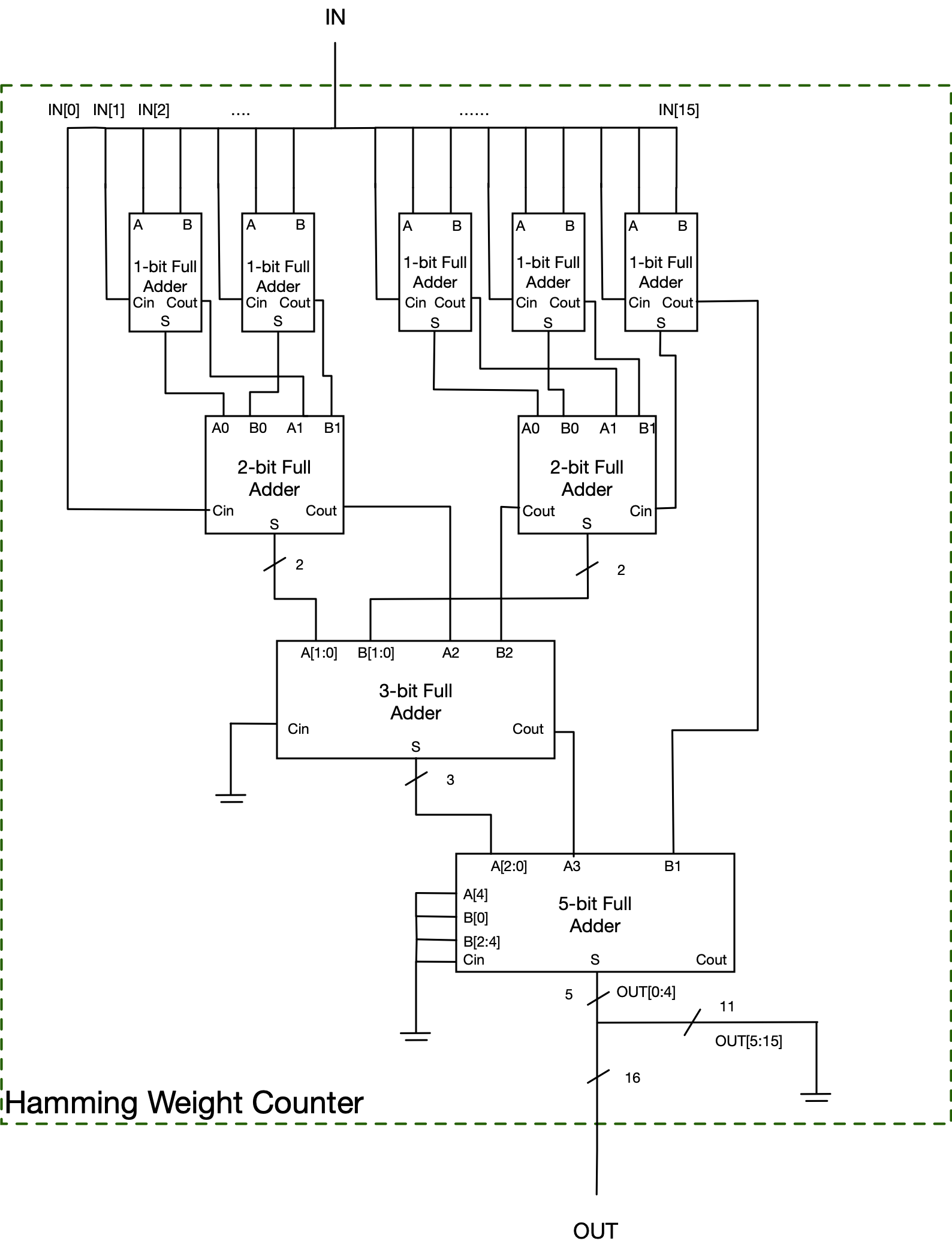
### Part C. Software Package

### Part D. Hardware Implementation

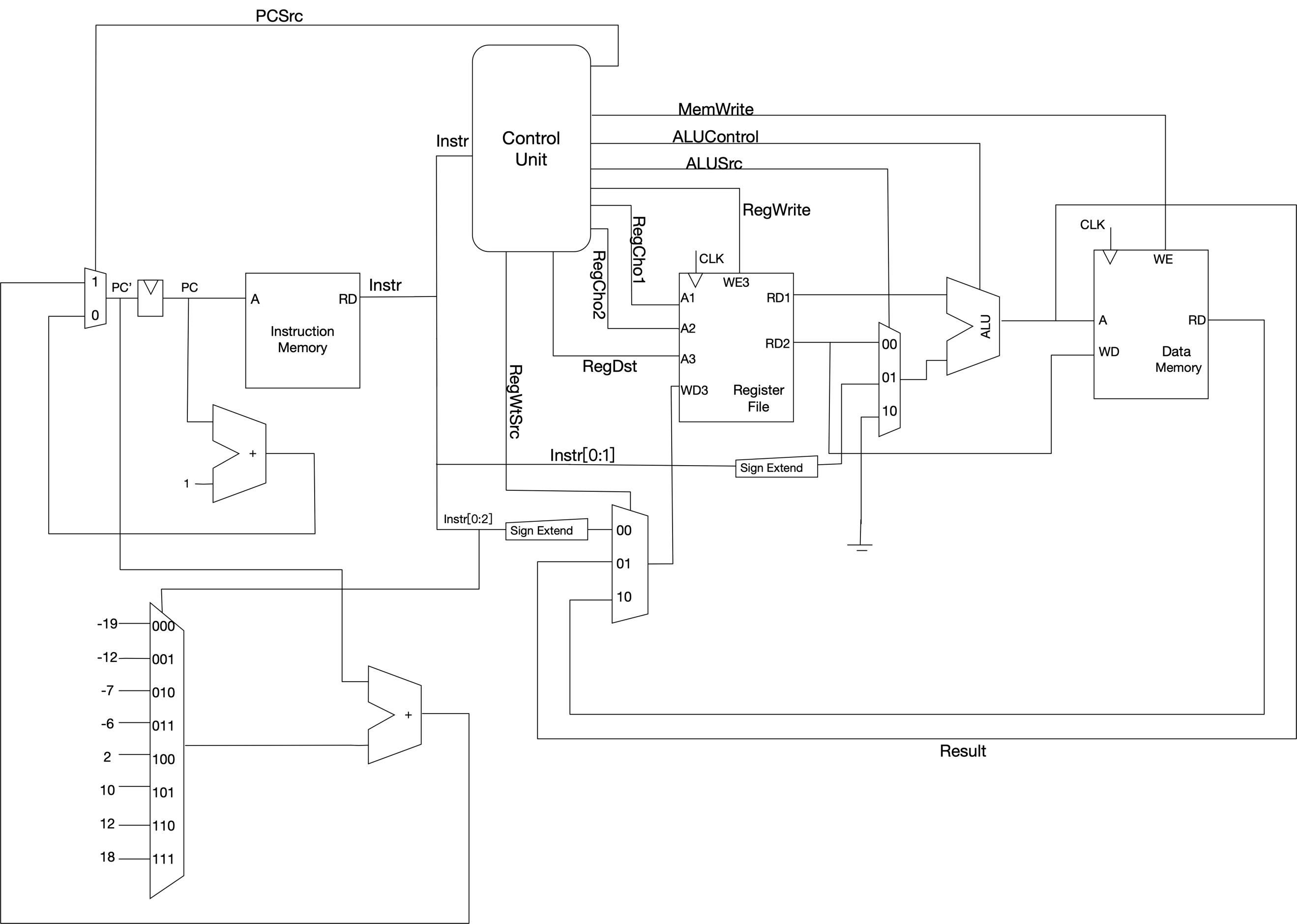
##### 1. ALU schematic



**Hamming Weight Counter**



##### 2. CPU Datapath



##### 3. Control Logic

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instr | Op | PCSrc | MemWrite | ALUControl | ALUSrc | RegWrite | RegCho1 | RegCho2 | RegDst | RegWtSrc |
| init | 000 r iii | 0 | 0 | XXX | XX | 1 | XX | XX | 0r | 00 |
| ld | 001 rr ss | 0 | 0 | 001 | 10 | 1 | ss | XX | rr | 10 |
| str | 010 rr ss | 0 | 1 | 001 | 10 | 0 | ss | rr | XX | XX |
| addR | 01100 rr | 0 | 0 | 001 | 00 | 1 | rr | rr | 10 | 01 |
| addR2 | 01110 rr | 0 | 0 | 001 | 00 | 1 | 10 | rr | 10 | 01 |
| addR3 | 01111 rr | 0 | 0 | 001 | 00 | 1 | rr | rr | 11 | 01 |
| subR3 | 01101 rr | 0 | 0 | 101 | 00 | 1 | 11 | rr | 11 | 01 |
| addi | 100 rr ii | 0 | 0 | 001 | 01 | 1 | rr | XX | rr | 01 |
| sltR0 | 101 rr ss | 0 | 0 | 110 | 00 | 1 | rr | ss | 00 | 01 |
| beqR0 | 11 rr iii | 1 | 0 | 101 | 00 | 0 | rr | 00 | XX | XX |
| scrR3R2 | 1110 111 | 0 | 0 | 000 | 00 | 1 | 10 | 11 | 11 | 01 |