Data Path

Controller Signals:

* UB – Unconditional Branch {1: Branch unconditionally, 0: Don’t do that}
* ZB – Branch on Zero {1: Branch if zero flag from ALU is set, 0: Don’t do that}
* NB – Branch on Negative {1: Branch if negative flag from ALU is set, 0: Don’t do that}
* ALUC – ALU Control Signal {4 bit signal for ALU ops}
* RW – Register Write {1: Write data to Rw address, 0: Don’t do that}
* MW – Memory Write {1: Write data to memory address, 0: Read data from memory address}
* MTR – Memory to Register {0: Send ALU result to register, 1: Send memory result to register}
* RDS – Register Data Source {1: Immediate Value, 0: Memory/ALU}
* IVA – Immediate Value to ALU {1: Use immediate value, 0: use register output}

Controller Signals

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Op Code | ALUC | UB | ZB | NB | RW | MW | MTR | RDS | IVA |
| SET | 1111\* | 1 | 0 | 0 | 1 | 0 | X | 1 | X |
| LOAD | 1111\* | 0 | 0 | 0 | 1 | 0 | 1 | 0 | X |
| STORE | 1111\* | 0 | 0 | 0 | 0 | 1 | X | X | X |
| MOVE | 1111\* | 0 | 0 | 0 | 1 | 0 | 0 | 0 | X |
| ADD | 0000 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| SUB | 0001 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| NEG | 0010 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | X |
| MUL | 0011 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| DIV | 0100 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| FLOOR | 0101 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | X |
| CEIL | 0110 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | X |
| ROUND | 0111 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | X |
| ABS | 1000 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | X |
| MIN | 1001 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| MAX | 1010 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| POW | 1011 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| EXP | 1100 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | X |
| SQRT | 1101 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| UB | 1111\* | 1 | 0 | 0 | 0 | 0 | X | X | X |
| ZB | 1111\* | 0 | 0 | 1 | 0 | 0 | X | X | X |
| NB | 1111\* | 0 | 1 | 0 | 0 | 0 | X | X | X |
| NOP | 1111\* | 0 | 0 | 0 | 0 | 0 | X | X | X |
| HALT | 1111\* | 1 | 0 | 0 | 0 | 0 | X | X | X |

\* 1111 is PASS op to ALU. It will pass whatever is on its first input through to its output. Doubles as no-op/don’t care command as output value can be ignored when not needed

Notes:

* Operations that use immediate values act as short unconditional branches from the PC’s perspective. The immediate values come immediately after the instruction and are read simultaneously. Then the PC jumps over the value and reads the next instruction, essentially a double increment rather than a single increment.
* HALT operation is implemented as an unconditional branch where the increment is 0. Once the HALT command is reached, the PC will remain locked to that memory address. (Probably a bad idea)
* MTR and RDS in the chart above are set to don’t care (X) when overriding signals around the register negate their impact. For example, if the register write signal is off (no data is being written to the register), the value that reaches the register through the muxes doesn’t matter.
* IVA is set to a don’t care when the second ALU source is not used for the operation