Operations

|  |
| --- |
| NOP |
| LOAD |
| STORE |
| ADD |
| ADDI |
| SUB |
| SR5 |
| BRANCH |

OP-code structure

[OP] [ARG] [ARG] [ARG]

Where each [ARG] can be either a constant, zero value, a register, or a memory address. Not all operations/instructions takes three arguments. It can range all the way from zero to three arguments, depending on the operation.

The following table shows how each individual instruction would be used.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operation name | OP code | Arguments |  |  |  |
| NOP | 000 | 0 | 0 | 0 | 0 |
| LOAD | 001 | [REG] | 0 | [ADR] | 0 |
| STORE | 010 | 0 | [REG] | [ADR] | 0 |
| ADD | 011 | [REG] | [REG] | [REG] | 0 |
| ADDI | 100 | [REG] | [REG] | [CONST] | 0 |
| SUB | 101 | [REG] | [REG] | [REG] | 0 |
| SR5 | 110 | [REG] | [REG] | 0 | 0 |
| BRANCH | 111 | 0 | 0 | 0 | 0 |

LOAD has a filler argument as its second argument, which value it ignores. This is simply because of the way our instruction is sliced into different wires so that the [ADR] will be the [22:20] bits rather than the [25:23] bits, since [22:20] is the bits that are used for addresses to the Data memory.