Instruction Set Architecture

We present a list of instructions typical of a RISC (reduced instruction set computer) machine. In data-movement and control instructions, the addresses may be immediate `#X`, direct (memory) `M`, indirect (memory) `[M]`, register `r`, or register indirect `[r]` addresses. Data-processing instructions use immediate or register addressing. `PC` is the programme counter and `a <- b` indicates that the value of `b` is placed in `a`.

`LOAD a, b`	`a <- b`
`STOR a, b`	`a <- b`
`ADD a, b, c`	`a <- b + c`
`SUB a, b, c`	`a <- b - c`
`MUL a, b, c`	`a <- b * c`
`DIV a, b, c`	`a <- b / c`
`AND a, b, c`	`a <- b & c`
`OR a, b, c`	`a <- b | c`
`NOT a, b`	`a <- !b`
`ASH a, b, c`	`a <- b arithmetically shifted by c positions`
`LSH a, b, c`	`a <- b logically shifted by c positions`
`BR a`	`PC <- a`
`BEQ a, b, c`	`PC <- a if b is equal to c`
`BNE a, b, c`	`PC <- a if b is not equal to c`
`BLT a, b, c`	`PC <- a if b is less than c`
`BGT a, b, c`	`PC <- a if b is greater than c`
`BLE a, b, c`	`PC <- a if b is less than or equal to c`
`BGE a, b, c`	`PC <- a if b is greater than or equal to c`

Most instructions have floating-point versions when special floating-point registers are used (but we will not need these).

Most architectures use two addresses, where the first (or second) argument serves both as the target and one of the sources, e.g., `ADD a, b` means `a <- a + b`. For branch instructions `BR X` means to jump to instruction `X` (i.e., load the address of instruction `X` into `PC`).

We will use a five-stage pipeline: IF (instruction fetch), ID (instruction decode), RR (register read), EX (execute instruction), WB (write back result). We will assume that for data-movement instructions the data transfer between the CPU and main memory happens in the execute stage. Note that for some instructions (e.g., `LOAD r, #X`) some of the pipeline stages (e.g., RR) are not needed.