Assembly Code: Notation

For our examples, wee assume a CPU with ten registers (`r0` to `r9`) and the basic instructions given in the following table:

Instruction	**Meaning**
`copy r0, c`	$r0 \leftarrow c$: copy a constant c into register r0
`copy r0, r1`	$r0 \leftarrow r1$: copy the value of register r1 to register r0
`add r0, r1, c`	$r0 \leftarrow r1 + c$: add constant c to the register r1 and put the result in r0
`add r0, r1, r2`	$r0 \leftarrow r1 + r2$: add the values of registers r1 and r2 and put the result in r0
`subtract r0, r1, c`	$r0 \leftarrow r1 - c$: subtract constant c from the register r1 and put the result in r0
`subtract r0, r1, r2`	$r0 \leftarrow r1 - r2$: subtract the registers r1 from r2 and put the result in r0
`multiply r0, r1, r2`	$r0 \leftarrow r1 \cdot r2$: multiply the values of register r1 and r2 and put the result in r0
`divide r0, r1, r2`	$r0 \leftarrow r1/r2$: integer division of register r1 by r2; the result is stored in r0
`modulo r0, r1, r2`	$r0 \leftarrow r1 \mod r2$: the rest of the integer division of register r1 by r2 is stored in r0
`jump n`	Jump to line n of the program
`jump_gt r0, r1, n`	Jump to Line n if the value in r0 is greater than the value in r1 ($r0 > r1$)
`jump_ge r0, r1, n`	Jump to Line n if $r0 \geq r1$ (greater or equal)
`jump_lt r0, r1, n`	Jump to Line n if $r0 < r1$ (less than)
`jump_le r0, r1, n`	Jump to Line n if $r0 \leq r1$ (less or equal)
`jump_eq r0, r1, n`	Jump to Line n if the value in r0 is equal to the value in r1 ($r0 = r1$)
	In any of the five previous instructions r1 can also be a constant c.
`stop`	Stop the program