Assembly language instruction set: AS and A-level Paper 2

This table and accompanying notes outline the standard AQA Assembly language instruction set that will be used in Paper 2 of our AS and A-level Computer Science specifications (7516, 7517). Examples of the use of the instruction set can be found in the Specimen Assessment Materials (and subsequent 'past papers') on the AQA website. Whilst there is no intention to change this instruction set, if it becomes necessary an updated version will be placed on the website. The Instruction set will always be printed in the live question papers.

Instruction set

`LDR Rd, <memory ref>`	Load the value stored in the memory location specified by `<memory ref>` into register `d`.
`STR Rd, <memory ref>`	Store the value that is in register `d` into the memory location specified by `<memory ref>`.
`ADD Rd, Rn, <operand2>`	Add the value specified in `<operand2>` to the value in register `n` and store the result in register `d`.
`SUB Rd, Rn, <operand2>`	Subtract the value specified by `<operand2>` from the value in register `n` and store the result in register `d`.
`MOV Rd, <operand2>`	Copy the value specified by `<operand2>` into register `d`.
`CMP Rn, <operand2>`	Compare the value stored in register `n` with the value specified by `<operand2>`.
`B <label>`	Always branch to the instruction at position `<label>` in the program.
`B<condition> <label>`	Branch to the instruction at position `<label>` if the last comparison met the criterion specified by `<condition>`. Possible values for `<condition>` and their meanings are: `EQ`: equal to `NE`: not equal to `GT`: greater than `LT`: less than
`AND Rd, Rn, <operand2>`	Perform a bitwise logical AND operation between the value in register `n` and the value specified by `<operand2>` and store the result in register `d`.
`ORR Rd, Rn, <operand2>`	Perform a bitwise logical OR operation between the value in register `n` and the value specified by `<operand2>` and store the result in register `d`.

`EOR Rd, Rn, <operand2>`	Perform a bitwise logical XOR (exclusive or) operation between the value in register `n` and the value specified by `<operand2>` and store the result in register `d`.
`MVN Rd, <operand2>`	Perform a bitwise logical NOT operation on the value specified by `<operand2>` and store the result in register `d`.
`LSL Rd, Rn, <operand2>`	Logically shift left the value stored in register `n` by the number of bits specified by `<operand2>` and store the result in register `d`.
`LSR Rd, Rn, <operand2>`	Logically shift right the value stored in register `n` by the number of bits specified by `<operand2>` and store the result in register `d`.
`HALT`	Stops the execution of the program.

Labels

A label is placed in the code by writing an identifier followed by a colon (:). To refer to a label, the identifier of the label is placed after the branch instruction.

Interpretation of **`<operand2>`**

`<operand2>` can be interpreted in two different ways, depending on whether the first character is a `#` or an `R`:

- `#` – Use the decimal value specified after the `#`, eg `#25` means use the decimal value 25.
- `Rm` – Use the value stored in register `m`, eg `R6` means use the value stored in register 6.

The available general purpose registers that the programmer can use are numbered 0 to 12.