Words

|  |  |  |
| --- | --- | --- |
| Word | Description | Notes |
| ?>r | Push ABC on stack (individual, AB, ABC) | Push xx |
| r>? | Pop ABC off stack (individual, AB, ABC) | Pop xx |
| ?>? | Register to register transfer | Ld v,w ; ld x,y |
| add | A := B + A | Add hl,de |
| and | A := B & A |  |
| xor | A := A ^ B |  |
| or | A := A | B |  |
| \*+ | Multiplier step | Shift C right into carry  If set add B into A  Shift B left |
| ! | Mem16[A] := B | Preserves B |
| c! | Mem8[A] := B |  |
| @ | A := Mem16[A] |  |
| [c@](mailto:c@) | A := Mem8[A] |  |
| +! | Add B into Mem16[A] |  |
| [p@](mailto:p@) | A := Port C | So we can use BC 😊 |
| p! | Port C := A | So we can use BC 😊 |
| 0= | $FFFF if A = 0 else $0000 |  |
| - | 1’s complement of A | Not negate ! |
| swap | Exchange A + B | Ex de,hl |
| 2\* 4\* 8\* 16\* | Shift A left | Add hl,hl |
| 2/ | Shift B right arithmetic | Sra h; rr l |
| copy | A (HL) source B (DE) target C (BC) count | Ldir 😊 |
| fill | L byte DE target BC count |  |
| ; | Return | Ret |
| halt | Stop running | Di ; halt ; jr [di] |
| = | -1 if A = B, 0 otherwise |  |
| < | -1 if B < A, 0 otherwise |  |
| ++ | Increment A | Inc hl |
| -- --- | Decrement A | Dec hl |
| 0- | Negate | 2’s complement |
| Bswap | A = byteswap (A) |  |

**Compiler Words**

|  |  |  |
| --- | --- | --- |
| Words | Effect | Notes |
| Variable | Compiles ret ; $0000 | Variable storage |
| Constant | Compiles code to load constant | e.g. ex de,hl ; ld hl,xxxx |
| !! &&@@ | Compile code based on last exec address | To load and store much faster |
| If/-if then | Conditional branch on A |  |
| Begin until/-until | Loop dependent on A |  |
| For I Next | Loop A times | A set at top of loop, I gets it. |
| :<word> | Mark word here |  |
| <var>.<method> | Ex de,hl ; ld hl,<var> <type>::<method> |  |
|  |  |  |