### Testing

### MOVL, MOVLZ, MOVLS, and MOVH instructions work as expected

**Purpose:** Checks for successful execution of MOVL, MOVLZ, MOVLS, and MOVH instructions.

**Configuration:** The following program is loaded into the emulator. It performs MOVL, MOVLS, MOVLZ, and MOVH operations.

A screen shot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **Expected Results** | **Actual Results** |
| At memory address 1002, MOVL is performed:   * DST = 0xEE00 * DST.lowbyte <- 0x99 * DST = 0xFF99 |  |
| At memory address 1004, MOVH is performed:   * DST = 0xEE99 * DST.highbyte <- 0xAA * DST = 0xAA99 |  |
| At memory address 1006, MOVLZ is performed:   * DST = 0xAA99 * DST.highbyte <- 0x00 * DST.lowbyte <- 0x88 * RESULT = 0xFF88 |  |
| At memory address 1008, MOVLS is performed:   * DST = 0xFF88 * DST.highbyte <- 0xFF * DST.lowbyte <- 0x77 * DST = 0xFF77 |  |

The MOVL, MOVLZ, MOVLS, and MOVH operations perform as expected.

### ADD and ADDC instructions work as expected

**Purpose:** Checks for successful execution of ADD and ADDC instructions.

**Configuration:** The following program is loaded into the emulator. It performs ADD and ADDC with byte and word addition, uses register and constant sources, and flags are all set in at least one operation.

A screenshot of a computer

Description automatically generated

|  |  |
| --- | --- |
| **Expected Results** | **Actual Results** |
| At memory address 1004, ADD is performed using values of size word and a register source:   * DST <- 0x9000 + 0x9000   This will produce:   * DST = 0x2000 * SET overflow flag * CLEAR negative flag * CLEAR zero flag * SET carry flag |  |
| At memory address 1006, ADDC is performed using values of size word with a constant value and a constant source:   * DST <- 0x2000 + $0 + CARRY   This will produce:   * DST = 0x2001 * CLEAR overflow flag * CLEAR negative flag * CLEAR zero flag * CLEAR carry flag |  |
| At memory address 100A, ADD is performed using values of size byte and a register source:   * DST.lowbyte <- 0x01 + 0xFF   This will produce:   * DST.lowbyte = 0x00 * CLEAR overflow flag * CLEAR negative flag * SET zero flag * SET carry flag |  |
| At memory address 100E, ADD is performed using values of size byte and a register source:   * DST.lowbyte <- 0x00 + 0xFF   This will produce:   * DST.lowbyte = 0xFF * CLEAR overflow flag * SET negative flag * CLEAR zero flag * CLEAR carry flag |  |

The ADD and ADDC instructions perform as expected. They produce the correct results when adding word or byte, and when there is a carry for ADDC. They also set the flags as expected.

### SUB and SUBC instructions work as expected

**Purpose:** Checks for successful execution of SUB and SUBC

**Configuration:** The following program is loaded into the emulator. It executes SUB and SUBC with byte and word subtraction, uses register and constant sources, and flags are all set in at least one operation.

**A screenshot of a computer program

Description automatically generated**

|  |  |
| --- | --- |
| **Expected Results** | **Actual Results** |
| At memory address 1004, SUB is performed using values of size word and register source:   * DST <- 0x8000 - 0x8000   This will produce:   * DST = 0x0000 * CLEAR overflow flag * CLEAR negative flag * SET zero flag * SET carry flag |  |
| At address 1006, SUBC is performed using values of size word and a constant source:   * DST <- 0x2000 - $0 + CARRY   This will produce:   * RESULT = 0x2001 * CLEAR overflow flag * CLEAR negative flag * SET zero flag * SET carry flag |  |
| At address 100C, SUB is performed using values of size byte and a register source:   * DST.lowbyte <- 0x30 – 0x80   This will produce:   * DST.lowbyte = 0x00B0 * SET overflow flag * SET negative flag * CLEAR zero flag * CLEAR carry flag |  |

The SUB and SUBC instructions perform as expected. They produce the correct results when subtracting word or byte, and when there is a carry for SUBC. They also set the flags as expected.

### DADD instructions work as expected

**Purpose:** Checks for successful execution of SUB and SUBC

**Configuration:** The following program is loaded into the emulator. It executes DADD with byte and word, register and constant sources, and flags are all set in at least one operation.

**A screen shot of a computer

Description automatically generated**

|  |  |
| --- | --- |
| **Expected Results** | **Actual Results** |
| At memory address 1008, DADD is performed using values of size word with a register source:   * DST <- 1234 + 8766   This will produce:   * DST = 0000 * CLEAR overflow flag * CLEAR negative flag * CLEAR zero flag * SET carry flag |  |
| At address 100A, DADD is performed using values of size word with a constant srouce:   * DST.lowbyte <- 34 + $4   This will produce:   * RESULT = 0x1238 * CLEAR overflow flag * CLEAR negative flag * CLEAR zero flag * CLEAR carry flag |  |

The SUB and SUBC commands perform as expected. They produce the correct results when subtracting word or byte, and when there is a carry for SUBC. They also set the flags as expected.