## Execute - EX

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | EX | | |
| Function | * The EX instruction executes a single instruction specified at the effective address computed by the second operand. * But, before that instruction is executed, the low order byte of the register indicated by R1 is ORed with the second byte of the instruction. This is done by hardware, and the actual instruction specified remains unchanged. * This instruction helps in overcoming the limitation imposed by the MVC instruction. In the MVC instruction, the length should be specified at assembly time and cannot be dynamic. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | EX R1, D2(X2,B2) | | |
| Opcode  44 R1 X2 B2 D2 |  | | |
| Instruction Format | RX | | |
| Condition Code | * The EX instruction itself does not set any condition code. * The condition code depends on the target instruction. | | |
| Program Exceptions | 1. Access – Fetch target instruction. 2. Execute exception when the target instruction is in turn an EXECUTE. 3. Specification exception. | | |
| Example | Assume that R8 is the base register of the CSECT and an MVC instruction is at location 40 from the base register.  EX R1, 40(R8)  44 18 00 28 | | |
| Register / Location | Initial Value | Final Value |
| R1 | 00 00 00 01 | Unknown |
| 40(R8)  [The target instruction] | D2 00 20 00 20 01 | Unchanged |
| 0(R2) | 40 C1 C2 | C1 C1 C2 |
| The instruction at 40(R8) : MVC 0(1,R2), 1(R2)  It moved the byte at 1(R1) to 0(R1) | | |

## Shift Instructions : Overview

### Arithmetic Shift:

* An arithmetic shift instruction treats data as integers in two’s complement form.
* When contents of a register undergo an arithmetic shift, not all 32 bits are involved in the shift. The sign bit is not involved in the shift.
* This ensures that the sign of the number remains unchanged.
* If a bit being shifted out of the sign bit differs from the sign bit, an overflow occurs.
* The bits that are shifted beyond the edge of the register are lost.
* Zeros are introduced into the other end.
* The following figure depicts what an arithmetic shift left achieves.

Register

S

Zeros are inserted here.

Bits shifted out here are lost.

* The following figure shows what an arithmetic shift right achieves.

Register

S

Bits shifted out here are lost.

Sign bit is inserted here[[1]](#footnote-2)

The System 390 also provides instructions to shift 64 bits held in an even-odd register pair. These are called, double shifts. The following discussion will focus on the instructions provided to achieve shifting.

<http://csc.columbusstate.edu/woolbright/Instructions/SRL.pdf>

<http://csc.columbusstate.edu/woolbright/Instructions/SRA.pdf>

## Shift Left Logical - SLL

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | SLL | | |
| Function | * This instruction shifts the 32-bit first operand left, the number of bits specified by the second-operand address. * The second-operand address is not used to address data; its rightmost six bits indicate the number of bit positions to be shifted. The remainder of the address is ignored. * Bits 12-15 of the instruction are ignored. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | SLL R1, D2(B2) | | |
| Opcode  89 R1 B2 D2 |  | | |
| Instruction Format | RS | | |
| Condition Code | Remains unchanged. | | |
| Program Exceptions | None | | |
| Example | The instruction below shifts contents of R1 left by 2 bits.  SLL R1, 2  89 10 00 02 | | |
| Register / Location | Initial Value | Final Value |
| R1 | FF FF FF FF | FF FF FF FC |

## Shift Right Logical - SRL

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | SRL | | |
| Function | * This instruction shifts the 32-bit content of the register specified as the first operand, the number of bits specified by the second-operand address. * The second-operand address is not used to address data; it’s rightmost six bits indicate the number of bit positions to be shifted. The remainder of the address is ignored. * Bits 12-15 of the instruction are ignored. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | SRL R1, D2(B2) | | |
| Opcode  88 R1 B2 D2 |  | | |
| Instruction Format | RS | | |
| Condition Code | Remains unchanged. | | |
| Program Exceptions | None | | |
| Example | The instruction below shifts contents of register R5 right by 2 bits.  SLL R5, 2  88 50 00 02 | | |
| Register / Location | Initial Value | Final Value |
| R5 | FF FF FF FF | 3F FF FF FF |

## Shift Left Arithmetic - SLA

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | SLA | | |
| Function | * The 31-bit numeric part of the signed first operand (general-purpose register) is shifted left the number of bits specified by the second-operand. * The sign bit is not involved in the shifting. * Zeroes are inserted to the right. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | SLA R1, D2(B2) | | |
| Opcode  8B R1 B2 D2 |  | | |
| Instruction Format | RS | | |
| Condition Code | 0 - Result zero  1 - Result negative  2 - Result positive  3 – Overflow | | |
| Program Exceptions | Floating point overflow. | | |
| Example | The instruction below shifts contents of R1 left by 2 bits.  SLA R1, 2(R0)  8B 10 00 02 | | |
| Register / Location | Initial Value | Final Value |
| R1 | FF FF FF FF | FF FF FF FC |

## Shift Right Arithmetic - SRA

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | SRA | | |
| Function | * This instruction shifts the 31-bit numeric part of the signed first operand (general-purpose register) right, the number of bits specified by the second-operand. * The sign bit is not involved in the shifting. * The content of the sign bit is inserted in place of the bits being shifted right. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | SRA R1, D2(B2) | | |
| Opcode  8A R1 B2 D2 |  | | |
| Instruction Format | RS | | |
| Condition Code | 0 - Result zero  1 - Result negative  2 - Result positive | | |
| Program Exceptions | None | | |
| Example | The instruction below shifts contents of R1 right by 2 bits.  SRA R1, 2(R0)  8A 10 00 02 | | |
| Register / Location | Initial Value | Final Value |
| R1 | FF FF FF FF | FF FF FF FF |

## Shift Left Double Logical - SLDL

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | SLDL | | |
| Function | * This instruction shifts left, the contents of an even-odd register pair specified as the first operand. * The content of the registers is treated as a 64 bit unsigned binary number. * Zeroes are inserted to the right. * The number of bits to be shifted is specified by the second operand and is restricted to a maximum of 63. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | SLDL R1, D2(B2) | | |
| Opcode  8D R1 B2 D2 |  | | |
| Instruction Format | RS | | |
| Condition Code | Remains unchanged. | | |
| Program Exceptions | Specification exception. | | |
| Example | The following instruction shifts contents of the even-odd register pair R4-R5 left by 2 bits.  SLDL R4, 2  8D 40 00 02 | | |
| Register / Location | Initial Value | Final Value |
| R4 | FF FF FF FF | FF FF FF FF |
| R5 | FF FF FF FF | FF FF FF FC |

## Shift Right Double Logical - SRDL

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | SRDL | | |
| Function | * This instruction shifts right, the contents of an even-odd register pair specified as the first operand. * The content of the registers is treated as a 64 bit unsigned binary number. * Zeroes are inserted to the left. * The number of bits to be shifted is specified by the second operand and is restricted to a maximum of 63. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | SRDL R1, D2(B2) | | |
| Opcode  8C R1 B2 D2 |  | | |
| Instruction Format | RS | | |
| Condition Code | Remains unchanged. | | |
| Program Exceptions | Specification exception. | | |
| Example | The following instruction shifts contents of the even-odd register pair R4-R5 right by 2 bits.  SRDL R4, 2  8C 40 00 02 | | |
| Register / Location | Initial Value | Final Value |
| R4 | FF FF FF FF | 3F FF FF FF |
| R5 | FF FF FF FF | FF FF FF FF |

## Shift Left Double Arithmetic - SLDA

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | SLDA | | |
| Function | * Shifts left, the 63-bit numeric part of the even-odd register pair specified as the first operand. * The content of the registers is treated as a 64 bit signed binary number. * The sign bit remains unchanged and does not participate in the shift. * Zeroes are inserted to the right. * The number of bits to be shifted is specified by the second operand and is restricted to a maximum of 63. * If one or more bits unlike the sign bit are shifted out of bit position 1 of the even-numbered register, an overflow occurs, and condition code 3 is set. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | SLDA R1, D2(B2) | | |
| Opcode  8F R1 B2 D2 |  | | |
| Instruction Format | RS | | |
| Condition Code | 0 – Result is zero  1 – Result negative  2 – Result positive  3 – Overflow | | |
| Program Exceptions | 1. Specification exception.  2. Fixed point overflow. | | |
| Example | The following instruction shifts contents of the even-odd register pair R4-R5 left by 2 bits.  SLDA R4, 2  8F 40 00 02 | | |
| Register / Location | Initial Value | Final Value |
| R4 | FF FF FF FF | FF FF FF FF |
| R5 | FF FF FF FF | FF FF FF FC |

## Shift Right Double Arithmetic - SRDA

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | SRDA | | |
| Function | * Shifts right, the 63-bit numeric part of the even-odd register pair specified as the first operand. * The content of the registers is treated as a 64 bit signed binary number. * The sign bit remains unchanged and does not participate in the shift. * The sign bit is inserted to the left. * The number of bits to be shifted is specified by the second operand and is restricted to a maximum of 63. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | SRDA R1, D2(B2) | | |
| Opcode  8E R1 B2 D2 |  | | |
| Instruction Format | RS | | |
| Condition Code | 0 – Result is zero  1 – Result negative  2 – Result positive | | |
| Program Exceptions | Specification exception. | | |
| Example | The following instruction shifts right, the contents of the even-odd register pair R4-R5 by 2 bits.  SRDA R4, 2  8E 40 00 02 | | |
| Register / Location | Initial Value | Final Value |
| R4 | FF FF FF FF | FF FF FF FF |
| R5 | FF FF FF FF | FF FF FF FF |

## Move Characters – MVC

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | MVC | | |
| Function | * Copies the consecutive bytes starting from the effective address specified by the second operand into the area pointed to by the first operand. * The number of bytes to be copied is specified as the length parameter. * In the mnemonic the length is stored as one less than the specified value. * The bytes are copied one at a time, left to right. * The maximum length of the data that can be copied is 256. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | MVC D1(L,B1), D2(B2) | | |
| Opcode  D2 L B1 D1 B2 D2 |  | | |
| Instruction Format | SS | | |
| Condition Code | Remains unchanged. | | |
| Program Exceptions | Access – Fetch 2nd operand, Store 1st operand. | | |
| Examples | The following instruction moves 5 bytes of data starting from 20(R4), to the area in memory starting from 50(R5).  MVC 50(5,R5), 20(R4)  Opcode - D2 04 50 32 40 14 | | |
| Register / Location | Initial Value | Final Value |
| 20(R4) | C1 C2 C3 C4 C5 | C1 C2 C3 C4 C5 |
| 50(R5) | D1 D2 D3 D4 D5 | C1 C2 C3 C4 C5 |
|  | | |
| This instruction moves a single byte of data from location 0(R4) to location 0(R5).  MVC 0(1,R5), 0(R4)  Opcode - D2 00 50 00 40 00 | | |
| Register / Location | Initial Value | Final Value |
| 0(R4) | C1 C2 C3 C4 | C1 C2 C3 C4 |
| 0(R5) | FF FF FF FF | C1 FF FF FF |
|  | | |
| This example illustrates the way an MVC instruction can be used to clear an area in storage.  MVC 1(4,R4), 0(R4)  Opcode – D2 03 40 01 40 00 | | |
| Register / Location | Initial Value | Final Value |
| 0(R4) | 40 FF FF FF | 40 40 40 40 |

# Logical Instructions

## Boolean Operations - AND

### AND Operation:

* The table below depicts the impact of the AND operation.
* The AND operation is usually used to set the switches OFF.
* The operations are done on individual bits.

|  |  |  |
| --- | --- | --- |
| A | B | A AND B |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

## Boolean Operations - OR

### OR Operation:

* The table below depicts the impact of the OR operation.
* The OR operation is usually used to set the switches ON.
* The operations are done on individual bits.

|  |  |  |
| --- | --- | --- |
| A | B | A OR B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

## Boolean Operations - XOR

### XOR Operation:

* The table below depicts the impact of the XOR operation.
* The XOR operation is usually used to flip bits of the switches.
* The operations are done on individual bits.

|  |  |  |
| --- | --- | --- |
| A | B | A XOR B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## And Register – NR

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | NR | | |
| Function | * The NR instruction ANDs the operand indicated by the register R1 with the second operand, the general-purpose register R2, and moves the result into the register indicated by R1. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | NR R1, R2 | | |
| Opcode  14 R1 R2 |  | | |
| Instruction Format | RR | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | None. | | |
| Example | NR R1, R2  Opcode - 14 12 | | |
| Register / Location | Initial Value | Final Value |
| R1 | FF E2 F1 80 | 00 00 00 80 |
| R2 | 00 00 00 F0 | Unchanged |

## And – N

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | N | | |
| Function | * This instruction ANDs the operand indicated by register R1 with the full word, which is pointed by the effective address computed with the second operand and stores the result in the first operand indicated by R1. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | N R1, D2(X2,B2) | | |
| Opcode  54 R1 X2 B2 D2 |  | | |
| Instruction Format | RX | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | Access – Fetch 2nd operand. | | |
| Example | N R1, 200(R2)  Opcode - 54 12 00 C8 | | |
| Register / Location | Initial Value | Final Value |
| R1 | FF E2 F1 80 | 00 00 00 80 |
| 200(R2) | 00 00 00 F0 | Unchanged |

## And Character – NC

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | NC | | |
| Function | * The NC instruction ANDs the character string pointed to by the first operand with the character string pointed to by the second operand. * The result is stored in the first operand. * The number of bits to be involved in the operation is decided by the length parameter specified in the first operand. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | NC D1(L,B1), D2(B2) | | |
| Opcode  D4 L B1 D1 B2 D2 |  | | |
| Instruction Format | SS | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | Access – Fetch 2nd operand, Store 1st operand. | | |
| Example | NC 0(4,R1), 0(R2)  Opcode – D4 03 10 00 20 00 | | |
| Register / Location | Initial Value | Final Value |
| 0(R1) | FF E2 F1 88 | 00 00 00 88 |
| 0(R2) | 00 00 00 FF | Unchanged |
| Condition code – 1 | | |

## And Immediate – NI

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | NI | | |
| Function | * The NI instruction ANDs the single byte pointed to by the first operand with the immediate data, which is the second operand. * The result is stored in the location pointed to by the first operand. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | NI D1(B1), I2 | | |
| Opcode  94 I2 B1 D1 |  | | |
| Instruction Format | SI | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | Access – Store 1st operand. | | |
| Example | NI 0(R4), X’FA’  Opcode - 94 FA 40 00 | | |
| Register / Location | Initial Value | Final Value |
| 0(R4) | 87 | 82 |
| Condition code – 1 | | |

## Or Register – OR

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | OR | | |
| Function | * The OR instruction ORs the operand indicated by the register R1 with the second operand, the general-purpose register R2, and moves the result into the register indicated by R1. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | OR R1, R2 | | |
| Opcode  16 R1 R2 |  | | |
| Instruction Format | RR | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | None. | | |
| Example | OR R1, R2  Opcode – 16 12 | | |
| Register / Location | Initial Value | Final Value |
| R1 | FF E2 F1 98 | FF E2 F1 F8 |
| R2 | 00 00 00 F0 | Unchanged |

## Or – O

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | O | | |
| Function | * This instruction ORs the operand indicated by register R1 with the full word, which is pointed by the effective address computed with the second operand and stores the result in the first operand indicated by R1. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | O R1, D2(X2,B2) | | |
| Opcode  56 R1 X2 B2 D2 |  | | |
| Instruction Format | RX | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | Access – Fetch 2nd operand. | | |
| Example | O R1, 200(R2)  Opcode - 56 12 00 C8 | | |
| Register / Location | Initial Value | Final Value |
| R1 | 00 00 00 80 | 00 00 00 F0 |
| 200(R2) | 00 00 00 70 | Unchanged |

## Or Character – OC

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | OC | | |
| Function | * The OC instruction ORs the character string pointed to by the first operand with the character string pointed to by the second operand. * The result is stored in the first operand. * The number of bits to be involved in the operation is decided by the length parameter specified in the first operand. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | OC D1(L,B1), D2(B2) | | |
| Opcode  D6 L B1 D1 B2 D2 |  | | |
| Instruction Format | SS | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | Access – Fetch 2nd operand, Store 1st operand. | | |
| Example | OC 0(4,R1), 0(R2)  Opcode – D6 03 10 00 20 00 | | |
| Register / Location | Initial Value | Final Value |
| 0(R1) | FF E2 F1 88 | FF FF FF FF |
| 0(R2) | FF FF FF FF | Unchanged |
| Condition code – 1 | | |

## Or Immediate – OI

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | OI | | |
| Function | * The OI instruction ORs the single byte pointed to by the first operand with the immediate data, which is the second operand. * The result is stored in the location pointed to by the first operand. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | OI D1(B1), I2 | | |
| Opcode  96 I2 B1 D1 |  | | |
| Instruction Format | SI | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | Access – Store 1st operand. | | |
| Example | OI 0(R4), X’00’  Opcode - 96 00 40 00 | | |
| Register / Location | Initial Value | Final Value |
| 0(R4) | 00 | 00 |
| Condition code – 0 | | |

## XOR Register – XR

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | XR | | |
| Function | * The XR instruction XORs the operand indicated by the register R1 with the second operand, the general-purpose register R2, and moves the result into the register indicated by R1. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | XR R1, R2 | | |
| Opcode  17 R1 R2 |  | | |
| Instruction Format | RR | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | None. | | |
| Example | XR R1, R2  Opcode – 17 12 | | |
| Register / Location | Initial Value | Final Value |
| R1 | FF E2 F1 98 | FF E2 F1 68 |
| R2 | 00 00 00 F0 | Unchanged |

## XOR – X

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | X | | |
| Function | * This instruction XORs the operand indicated by register R1 with the full word, pointed by the effective address computed with the second operand and stores the result in the first operand indicated by R1. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | X R1, D2(X2,B2) | | |
| Opcode  57 R1 X2 B2 D2 |  | | |
| Instruction Format | RX | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | Access – Fetch 2nd operand. | | |
| Example | X R1, 200(R2)  Opcode - 57 12 00 C8 | | |
| Register / Location | Initial Value | Final Value |
| R1 | 00 00 00 80 | 00 00 00 F0 |
| 200(R2) | 00 00 00 70 | Unchanged |

## XOR Character – XC

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | XC | | |
| Function | * The XC instruction XORs the character string pointed to by the first operand with the character string pointed to by the second operand. * The result is stored in the first operand. * The number of bits to be involved in the operation is decided by the length parameter specified in the first operand. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | XC D1(L,B1), D2(B2) | | |
| Opcode  D7 L B1 D1 B2 D2 |  | | |
| Instruction Format | SS | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | Access – Fetch 2nd operand, Store 1st operand. | | |
| Example | XC 0(4,R1), 0(R2)  Opcode – D7 03 10 00 20 00 | | |
| Register / Location | Initial Value | Final Value |
| 0(R1) | FF E2 F1 88 | FF 1D 0E 77 |
| 0(R2) | FF FF FF FF | Unchanged |
| Condition code – 1 | | |

## XOR Immediate – XI

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | XI | | |
| Function | * The XI instruction XORs the single byte pointed to by the first operand with the immediate data, which is the second operand. * The result is stored in the location pointed to by the first operand. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | OI D1(B1), I2 | | |
| Opcode  97 I2 B1 D1 |  | | |
| Instruction Format | SI | | |
| Condition Code | 0 - Result zero  1 - Result non-zero | | |
| Program Exceptions | Access – Store 1st operand. | | |
| Example | XI 0(R4), X’FF’  Opcode - 97 FF 40 00 | | |
| Register / Location | Initial Value | Final Value |
| 0(R4) | FF | 00 |
| Condition code – 0 | | |

## Test Under Mask – TM

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | TM | | |
| Function | * The TM instruction tests the bits pointed to by the first operand. * The bits are tested by ANDing them according to the selection provided by the immediate data, which is the second operand. * The contents of the operands are not modified. * The condition code is set to reflect the values of the bits. | | |
| Addressing Schemes | Insignificant. | | |
| Syntax | TM D1(B1), I2 | | |
| Opcode  91 I2 B1 D1 |  | | |
| Instruction Format | SI | | |
| Condition Code | 0 - All tested bits are zeroes.  1 - Result mixed.  2 ––  3 - All tested bits are ones. | | |
| Program Exceptions | Access – Fetch 1st operand. | | |
| Example | TM 0(R4), X’40’  Opcode - 91 40 40 00 | | |
| Register / Location | Initial Value | Final Value |
| 0(R4) | C0 | Unchanged |
| Condition code – 3 | | |

1. An arithmetic shift to left is equivalent to multiplication by powers of 2and shifting right is equivalent to division by powers of 2!

   It is a common programming practice to use arithmetic shifts instead of multiplication or division if the multiplier or divisor is a number that is a power of 2. [↑](#footnote-ref-2)