SHIFT OPERATIONS

There are three functionally different shift operations: logical shift, insert field, and integer shift. Within the logical and integer shift classes either single or double length operands may be used. There also are two ways of specifying the direction and amount of shift: either directly from a literal field in the instruction or indirectly by the contents of a register.

Shift Amount and Direction

When the shift amount is specified by the literal field of the instruction, the following instruction format is used:

op	i	jk

The 10-bit literal jk-field is interpreted as a 2's complement integer, so that numbers in the range -512 to +511 are representable.

When the shift amount is specified by the contents of a register, the following instruction format is used:

op	i	j	k

The contents of register A^k or X^k is interpreted as a 2's complement integer. Only the low order 10 bits are used to specify the shift amount; the remaining bits are ignored.

The integer specifies both the direction and amount of the shift. Its absolute value specifies the amount of the shift. Its sign indicates the shift direction: a positive integer specifies a left shift, a negative integer specifies a right shift.

For the insert field instructions register A^k or X^k contains three 8-bit parameters.

Source and Result Operands

When the shift is specified by the literal field, the i-field specifies both the source and result operands; that is, Source operand: R^i or $R^{i,i+1}$

Result operand: R^i or $R^{i,i+1}$

(where R may be interpreted as either X or A).

When the shift amount is specified by R^k, the i- and j-fields specify the source and result operands as follows:

Source operand: R^j or $R^{j,j+1}$

Result operand: R^i or $R^{i,i+1}$

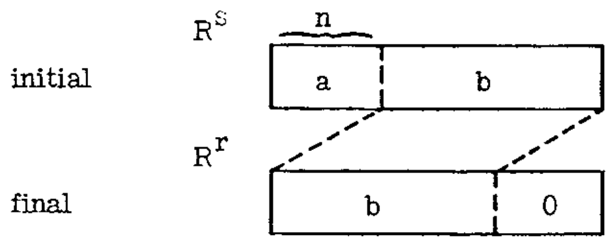
In the explanations, R^s is used to indicate the source register, and R^r the result register. The shift amount is denoted by n. The notation 48/24 is to be interpreted as 48 for the A-unit shift instructions and 24 for the X-unit shift instructions.

Logical Shift, Single Register

The contents of register R^s are shifted left or right the specified number of bit positions. The direction of the shift is determined by the sign of the shift amount. Bits which are shifted out of R^s are lost; vacated positions are filled with 0's. The 48/24-bit shifted quantity then replaces the contents of register R^r. The contents of register R^s are unchanged unless due to the operation type or the specification of the i and j field, R^s is the same register as R^r. If the shift amount is greater than or equal to 48/24, register R^r is set to 0's.

Pictorially the logical shift, single register, instructions are:

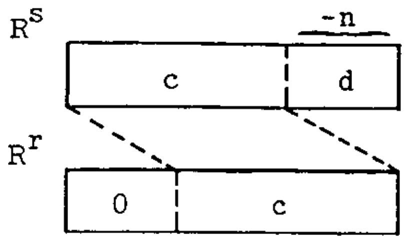
single shift left



single shift right

initial

final



Logic Shift, Arithmetic SHA

	i	j	k

shift amount ← $A^k_{38,\ldots,47}$

A^i ← logic shift (A^j)

Exceptions: none

Logic Shift, Index SHX

	i	j	k

shift amount ← $X^k_{14,\ldots,23}$

X^i ← logic shift (X^j)

Exceptions: none

Logic Shift, by Constant, Arithmetic SHAC

	i	jk

shift amount ← jk

A^i ← logic shift (A^i)

Exceptions: none

Logic Shift, by Constant, Index SHXC

	i	jk

shift amount ← jk

X^i ← logic shift (X^i)

Exceptions: none

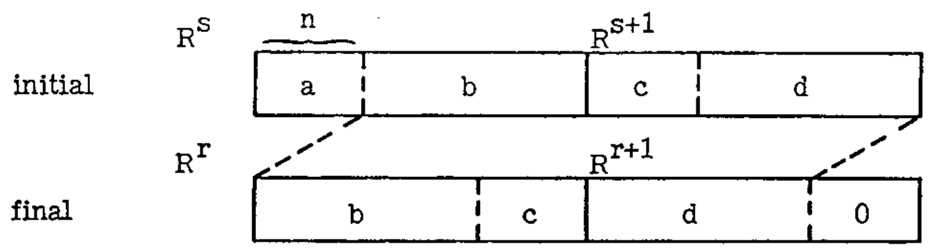
Logical Shift, Double Registers

Registers R^s and R^{s+1} are coupled and are considered as one 96/48 bit quantity. This 96/48 bit quantity is shifted left or right the specified number of bit positions to form an intermediate result. The direction of the shift is determined by the sign of the shift quantity. Bits which are shifted out are ignored; vacated positions are filled with 0's. The 96-bit intermediate result then replaces the contents of registers R^r and R^{r+1}.

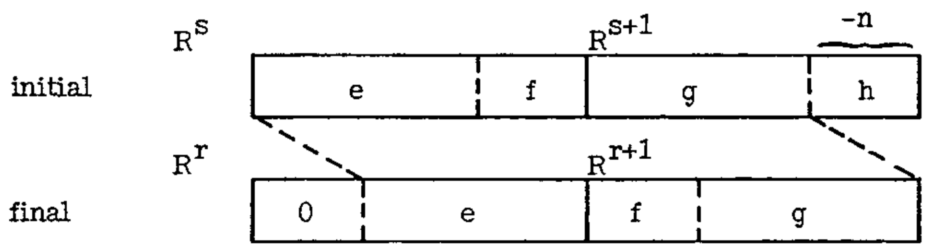
If the shift amount is greater than or equal to 96/48, registers R^r and R^{r+1} are set to 0's.

Pictorially, the logical shift, double registers, instructions are as follows:

double shift left



double shift right



The value of s must be even. If it is not, the low order bit specifying s is forced to 0, exception bit RS is set, and the operation proceeds.

Logic Shift, Double Arithmetic SHD

	i	j	k

shift amount $\leftarrow A^{k}_{38,\ldots,47}$

$A^{i,i+1} \leftarrow$ logic shift $(A^{j,j+1})$

Exception	Exception bit
i or j odd	RS

Logic Shift, Double Index SHDX

	i	j	k

shift amount $\leftarrow X^{k}_{14,\ldots,23}$

$X^{i,i+1} \leftarrow$ logic shift $(X^{j,j+1})$

Exception	Exception bit
i or j odd	RS

Logic Shift by Constant, Double Arithmetic SHDC

	i	jk

shift amount $\leftarrow$ jk

$A^{i,i+1} \leftarrow$ logic shift $(A^{i,i+1})$

Exception	Exception bit
i odd	RS

Logic Shift by Constant, Double Index SHDXC

	i	jk

shift amount $\leftarrow$ jk

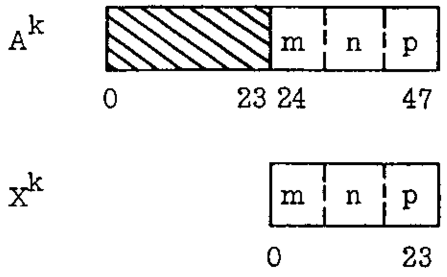
$X^{i,i+1} \leftarrow$ logic shift $(X^{i,i+1})$

Exception	Exception bit
i odd	RS

This page has been deleted.

Insert Field

Register R^k supplies three 8-bit integer parameters m, n, and p. These parameters are packed in R^k as shown:



The contents of register R^j are rotated left m positions. Bits rotated out of position 0 are inserted into position 47/23.

The p-n bits of this rotated quantity numbered n, n+1, n+2,..., p-1 are then inserted into the corresponding bits of register R^i.

The remaining bits of R^i (namely those numbered 0, 1, 2,..., n-1 and p, p+1, p+2,..., 47/23) either are left unaltered for the instructions IFX and IFA or are set to 0's for the instructions IFZX and IFZA. The contents of R^j and R^k are not changed.

The parameter m is interpreted as a positive integer modulo 48/24. The normal ranges for the positive integer parameters n and p are:

$$0 \leq n \leq 47/23$$

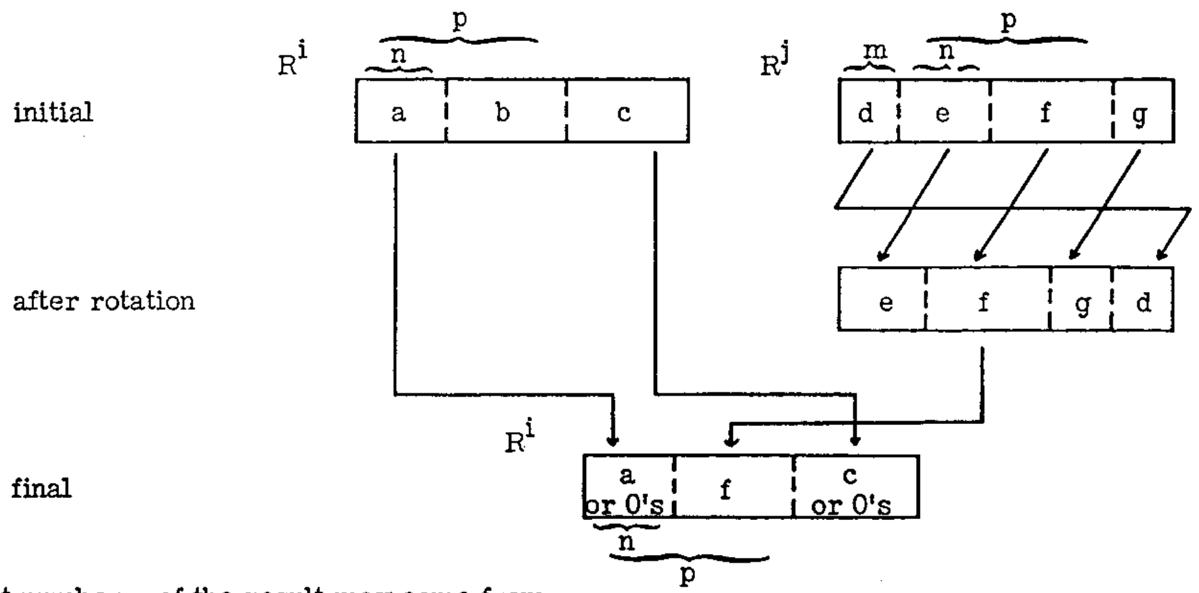
$$1 \leq p \leq 48/24$$

$$n < p$$

If $p \geq 49/25$, the operation proceeds as if $p = 48/24$. If $n \geq 48/24$ or if $p = 0$ or if $n = p$, the contents of R^i are left unaltered for IFA and IFX or are set to 0's for IFZA and IFZX.

If $n > p$, the $n - p$ bits of R^i numbered p, p+1,..., n-1 are set to 0's; the remaining bits are left unaltered for IFA and IFX or are also set to 0's for IFZA and IFZX.

Pictorially the insert field instructions are as follows:



Bit number α of the result may come from

(1) a source of 0's
or (2) bit α of operand R^i
or (3) bit $\alpha + m$ (mod 48/24) of operand R^j

Insert Field, Arithmetic IFA

	i	j	k

insertion parameters $\leftarrow A^k_{24,25,\ldots,47}$

$A^i \leftarrow \mathrm{insert}(A^i, A^j)$

Exceptions: none

Insert Field, Index IFX

	i	j	k

insertion parameters $\leftarrow X^k$

$X^i \leftarrow \mathrm{insert}\ (X^i, X^j)$

Exceptions: none

Insert Field and Zero, Arithmetic IFZA

	i	j	k

insertion parameters $\leftarrow A^k_{24,25,\ldots,47}$

$A^i \leftarrow \mathrm{insert}\ (0, A^j)$

Exceptions: none

Insert Field and Zero, Index IFZX

	i	j	k

insertion parameters $\leftarrow X^k$

$X^i \leftarrow \mathrm{insert}\ (0, X^j)$

Exceptions: none

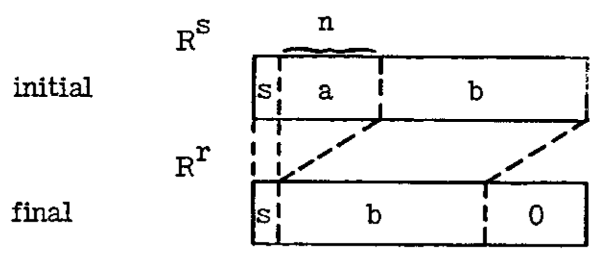
<u>Integer Shift, Single Register</u>

The contents of the last 47/23 positions of register R^s are shifted left or right the specified number of positions to form an intermediate result; position R^s_0 is not shifted. If the shift is to the right, bit values equal to R^s_0 are supplied to the vacated high-order positions; low-order bits are shifted out and ignored. If the shift is to the left, 0's are supplied to the vacated low-order positions; high-order bits are shifted out and are lost; however if the instruction is SIA or SIAC, and if one or more of the bits which is shifted out is unequal to A^s_0, the shift overflow exception bit SO is set to 1. The shifted quantity then replaces the contents of register R^r, bit R^r_0 being set to the value of R^s_0. The contents of register R^s are unchanged unless the operation type of the specification of the i and j field result in R^r being the same register as R^s.

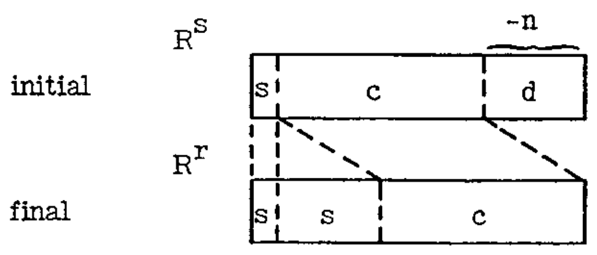
If the shift amount is greater than or equal to 47/23, the low order 47/23 bits of R^r are set to 0's for a left shift, or to the value of R^s_0 for a right shift.

Pictorially the integer shift, single register, instructions are:

single shift left



single shift right



Integer Shift, Arithmetic

SIA

	i	j	k

Shift amount $\leftarrow A^{k}_{38,\ldots,47}$

$A^{i} \leftarrow$ integer shift (A^{j})

Exception	Exception bit
bit different from A^{j}_{0} shifted out during left shift	SO

Integer Shift, Index

SIX

	i	j	k

Shift amount $\leftarrow X^{k}_{14,\ldots,23}$

$X^{i} \leftarrow$ integer shift (X^{j})

Exceptions: none

Integer Shift by Constant, Arithmetic SIAC

	i	jk

Shift amount ← jk

$A^i \leftarrow$ integer shift (A^i)

Exception	Exception bit
bit different from A_0^i shifted out during left shift	SO

Integer Shift by Constant, Index SIXC

	i	jk

Shift amount ← jk

$X^i \leftarrow$ integer shift (X^i)

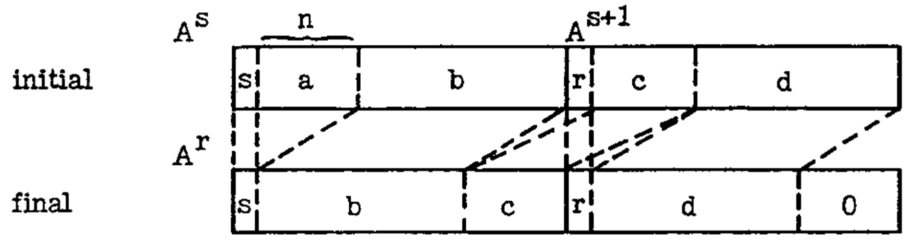
Exceptions: none

Integer Shift, Double Register

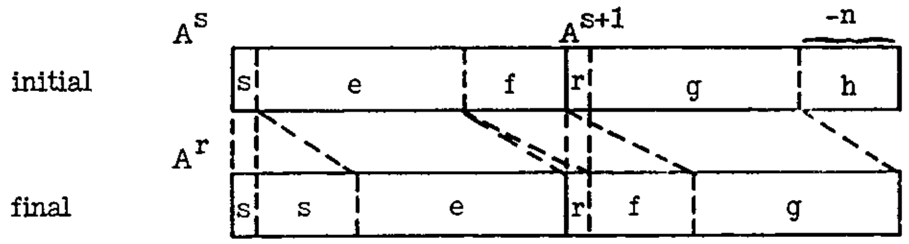
Registers A^s and A^{s+1} are coupled and are considered as one 96-bit quantity. Of this quantity 94 bits are shifted left or right the specified number of positions to form an intermediate result. The bits corresponding to A^s_0 and A^{s+1}_0 are not shifted. Bit A^{s+1}_0 specifies the value of the result bit A^{r+1}_0, but does not enter the operation in any other way. Except for the treatment of bits A^{s+1}_0 and A^{r+1}_0, the double register signed shift instruction is identical in function to the single register integer shift instruction when the latter is considered to operate on a 95-bit quantity instead of a 48-bit quantity.

Pictorially the integer shift, double register, instruction is:

double shift left



double shift right



The value of s must be even. If it is not, the low order bit specifying s is forced to 0, exception bit RS is set, and the operation proceeds.

Integer Shift, Double SID

	i	j	k

Shift amount $\leftarrow A^{k}_{38,\ldots,47}$

$A^{i,i+1} \leftarrow$ integer shift $(A^{j,j+1})$

Exceptions	Exception bit
bit different from A^{j}_{0} shifted out during left shift	SO
i or j odd	RS

Integer Shift, Double by Constant SIDC

	i	jk

Shift amount $\leftarrow$ jk

$A^{i,i+1} \leftarrow$ integer shift $(A^{i,i+1})$

Exceptions	Exception bit
bit different from A^{i}_{0} shifted out during left shift	SO
i odd	RS