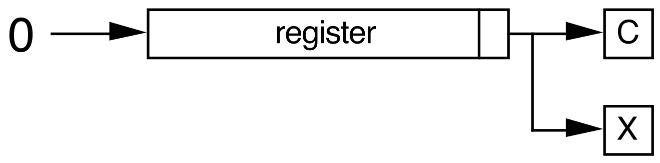
ROTATE AND SHIFT INSTRUCTIONS

logical shift for **unsigned** numbers
Provide a means for shifting blocks of bits within a register or memory.

Logical shift right

```
LSR.<size>  #N,Dn
LSR.<size>  Dm,Dn
LSR.W       <ea>
```

Action: The contents of the data register Dn are shifted right by the number of bits specified in the source operand. The vacated bits are filled with zeros. The shifted bits are stored in the X and C bits of the Status Register.



Notes:

1. A shift in the range 1-8 may be written as immediate data; anything larger than 8 will be replaced by Nmod8. A shift in the range 0-63 may be contained in a data register Dm.
2. Use of the <ea> operand will result in a shift of exactly one bit. The size for this operand can only be word.
3. The result of the operation is specified by the N and the Z bits. The overflow (V) bit is always cleared.

Example:

```
LSR   #4,D3
```

BEFORE

32	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	0	1	0	1	0	0	1	1	1	1	0	1	0

STATUS REGISTER	X	C
	0	0

AFTER

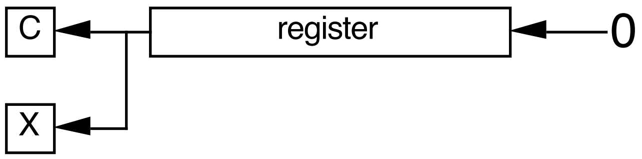
32	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	1	0	0	1	0	1	0	0	1	1	1

STATUS REGISTER	X	C
	1	1

Logical shift left

LSL.<size>   #N,Dn
LSL.<size>   Dm,Dn
LSL.W<ea>

Action	The contents of the data register Dn are shifted left by the number of bits specified in the source operand. The vacated bits are filled with zeros. The shifted bits are stored in the X and C bits of the Status Register.



Notes:

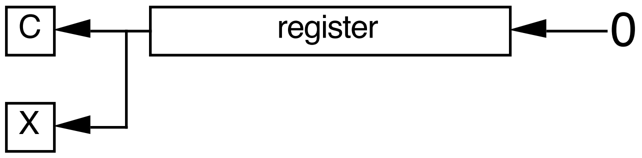
1. A shift in the range 1-8 may be written as immediate data; anything larger than 8 will be replaced by Nmod8. A shift in the range 0-63 may be contained in a data register Dm.
2. Use of the <ea> operand will result in a shift of exactly one bit. The size for this operand can only be word.
3. The result of the operation is specified by the N and the Z bits. The overflow (V) bit is always cleared.

arithmetic shift for **signed** numbers

Arithmetic shift left

ASL.<size>	#N,Dn	;shifts Dn by #N, #N must satisfy $1 \le \#N \le 8$
ASL.<size>	Dm,Dn	;shifts Dn by Dm
ASL.W	<ea>	;shifts word in memory by ONLY 1 bit

Action The contents of the data register are shifted preserving the sign of the original number. A shift count in the range 1-8 can be written as immediate data (#N). A shift count in the range 0-63 may be contained in data register Dm.



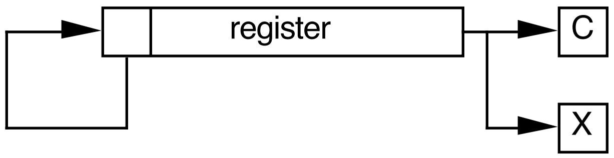
NOTES:

1. The size parameter can be byte, word or long word. If the shift is greater than 8 bits it MUST be stored in a data register Dm.
2. ASL <ea> can only operate on words and can only shift 1 bit.
3. The shift count can be loaded into Dm during program execution allowing variable shift counts in loops.
4. It is faster to move data to a register and shift it than using multiple ASL <ea> commands if the shift is greater than or equal to three bits.
5. An overflow is set if the sign bit changes. Consider the binary number $0110_2=6_{10}$. An ASL of 1 bit produces the number $1100_2=-4_{10}$. More formally, the V bit indicates if a sign change occurred. The Z and N bits are set according to the result of the operation. With ASL bits shifted out of the high-order bit go to both the X and C bits.

Arithmetic shift right

```
ASR.<size>    #N,Dn        ;shifts Dn by #N,  #N must satisfy 1≤#N≤8
ASR.<size>    Dm,Dn        ;shifts Dn by the value in Dm
ASR.W         <ea>         ;shifts word in memory by ONLY 1 bit
```

Action The contents of the data register are shifted preserving the sign of the original number. A shift count in the range 1-8 can be written as immediate data (#N). A shift count in the range 0-63 may be contained in data register Dm.



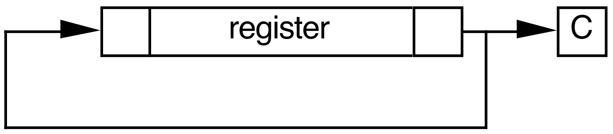
NOTES:
1. Consider the binary number 1010_2=-6_{10}. An ASR of 2 bits produces the number 1110_2=-2_{10}. The circular nature of the MSB in this instruction is, in effect, a sign extension to preserve the sign of the signed number.
2. The size parameter can be byte, word or long word. If the shift is greater than 8 bits it MUST be stored in a data register Dm.
3. ASL <ea> can only operate on words and can only shift 1 bit.
4. The shift count can be loaded into Dm during program execution allowing variable shift counts in loops.
5. It is faster to move data to a register and shift it than using multiple ASR <ea> commands if the shift is greater than or equal to three bits.
6. The overflow bit (V) is set if the sign bit changes. As the sign is preserved in the shift this should never occur. The Z and N bits are set according to the result of the operation. Bits shifted out of the least significant bit go to both the X and C bits.

Rotate instructions are similar to shift instructions; however, rotate instructions do an end around, shifts do NOT

rotate right

ROR.<size>	#N,Dn	;rotates Dn by #N, #N must satisfy $1 \le \#N \le 8$
ROR.<size>	Dm,Dn	;rotates Dn by Dm
ROR.W	<ea>	;shifts word in memory by ONLY 1 bit

Action The bits of the destination are rotated right The extend bit is NOT included in the rotation. The number of bits rotated is determined by the source operand.

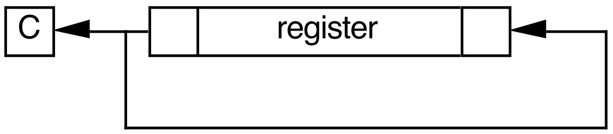


NOTES:
1. The bits rotated out of the least significant bit of the operand go to both the carry bit and the most significant bit of the operand.
2. The size parameter can be byte, word or long word. If the rotation is greater than 8 bits it MUST be stored in a data register Dm.
3. ROR <ea> can only operate on words and the rotation is always 1 bit.

rotate left

ROL.<size>	#N,Dn	;rotates Dn by #N, #N must satisfy $1 \le \#N \le 8$
ROL.<size>	Dm,Dn	;rotates Dn by Dm
ROL.W	<ea>	;shifts word in memory by ONLY 1 bit

Action The bits of the destination are rotated left. The extend bit is NOT included in the rotation. The number of bits rotated is determined by the source operand.



NOTES:
1. The bits rotated out of the most significant bit of the operand go to both the carry bit and the least significant bit of the operand.
2. The size parameter can be byte, word or long word. If the rotation is greater than 8 bits it MUST be stored in a data register Dm.

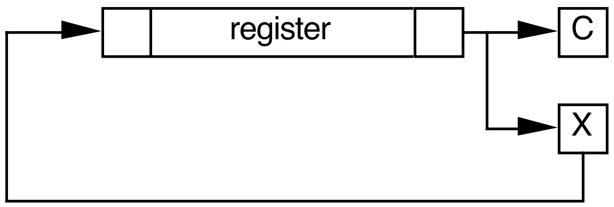
3. ROL <ea> can only operate on words and the rotation is always 1 bit.

rotate with extend instructions

<u>rotate right with extend</u>

```
ROXR.<size>  #N,Dn          ;rotates Dn by #N, #N must satisfy 1≤#N≤8
ROXR.<size>  Dm,Dn          ;rotates Dn by Dm
ROXR.W        <ea>          ;rotates word in memory by ONLY 1 bit
```

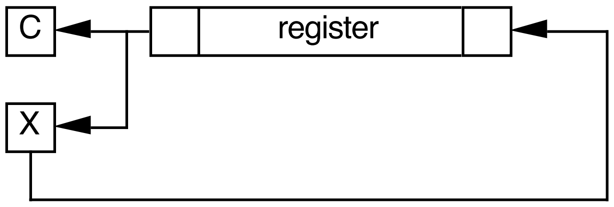
Action The bits of the destination are rotated right with the X bit included in the rotation. The number of bits rotated is determined by the source operand. The least significant bit of the operand is shifted into the C and X bit. The X bit is shifted into the most significant bit of the operand. This process continues for each succeeding shift.



<u>rotate left with extend</u>

```
ROXL.<size>  #N,Dn          ;rotates Dn by #N, #N must satisfy 1≤#N≤8
ROXL.<size>  Dm,Dn          ;rotates Dn by Dm
ROXL.W        <ea>          ;rotates word in memory by ONLY 1 bit
```

Action The bits of the destination are rotated left with the X bit included in the rotation. The number of bits rotated is determined by the source operand. The most significant bit of the operand is shifted into the C and X bit. The X bit is shifted into the most significant bit of the operand. This process continues for each succeeding shift.


C
register
X