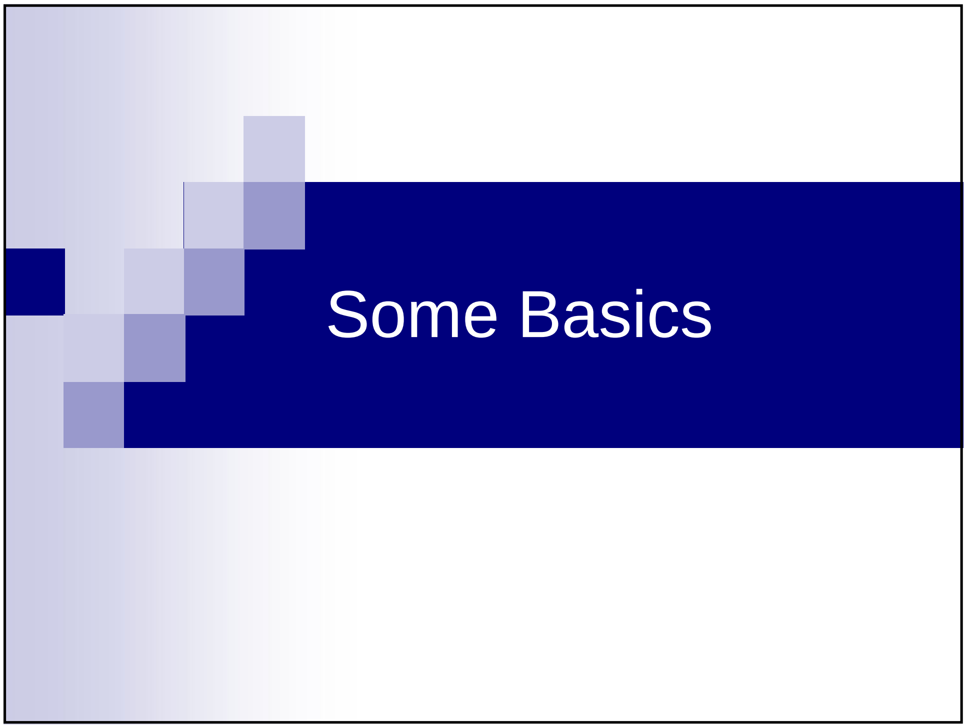
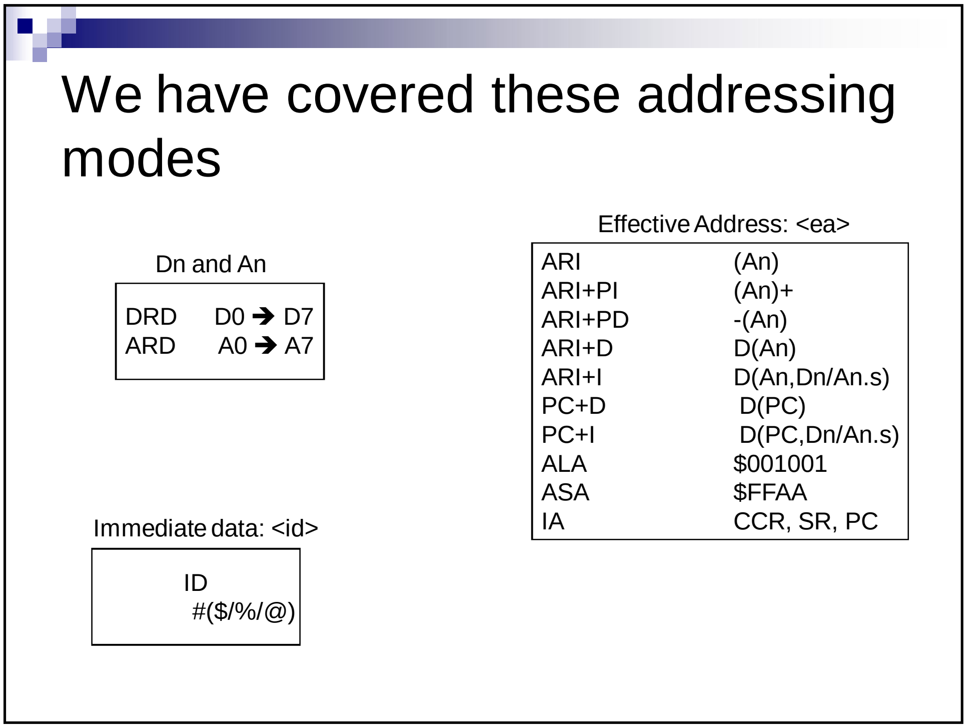
Logical, Shift, Rotate & BCD

What we will learn in this session:

- Logical instructions.
- Shift & Rotate instructions.
- BCD operations.
- Bit operations.



Some Basics



We have covered these addressing modes

Dn and An

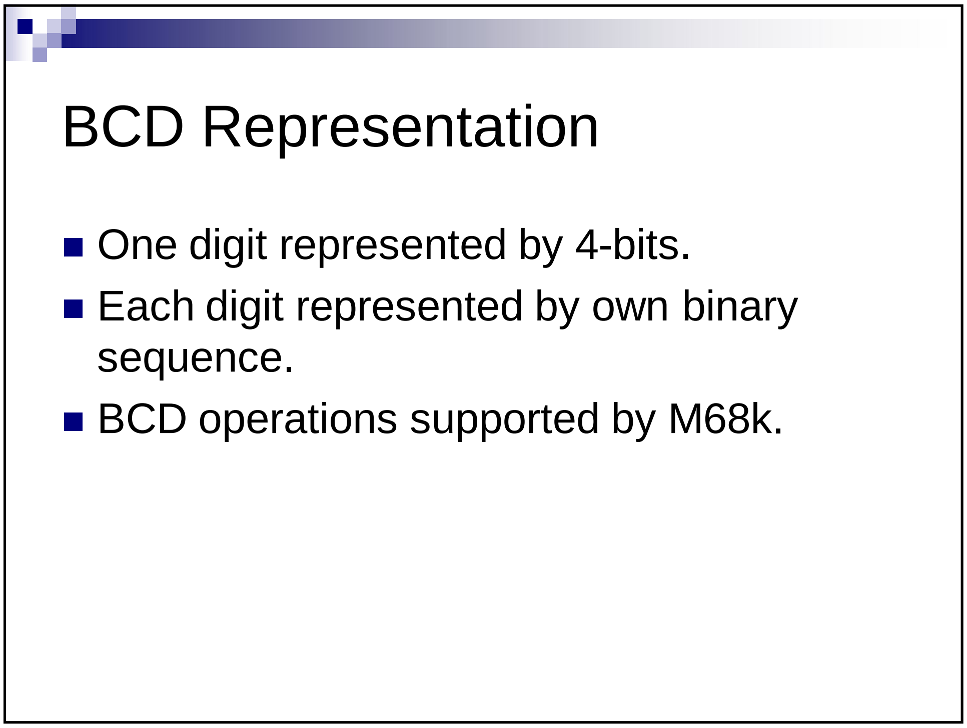
DRD	D0 ➔ D7
ARD	A0 ➔ A7

Effective Address: <ea>

ARI	(An)
ARI+PI	(An)+
ARI+PD	-(An)
ARI+D	D(An)
ARI+I	D(An,Dn/An.s)
PC+D	D(PC)
PC+I	D(PC,Dn/An.s)
ALA	$001001
ASA	$FFAA
IA	CCR, SR, PC

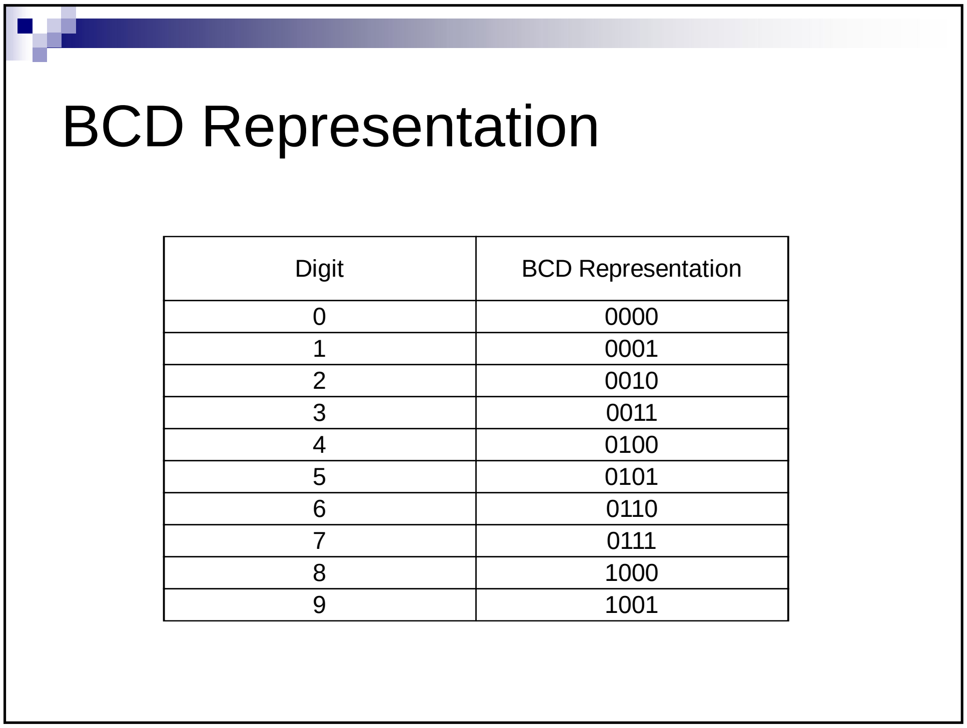
Immediate data: <id>

ID	#($/%/@)



BCD Representation

- One digit represented by 4-bits.
- Each digit represented by own binary sequence.
- BCD operations supported by M68k.

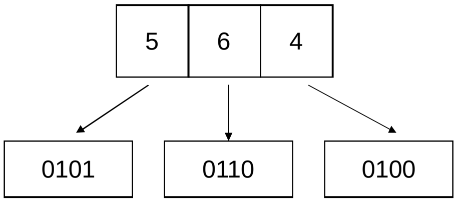


BCD Representation

Digit	BCD Representation
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

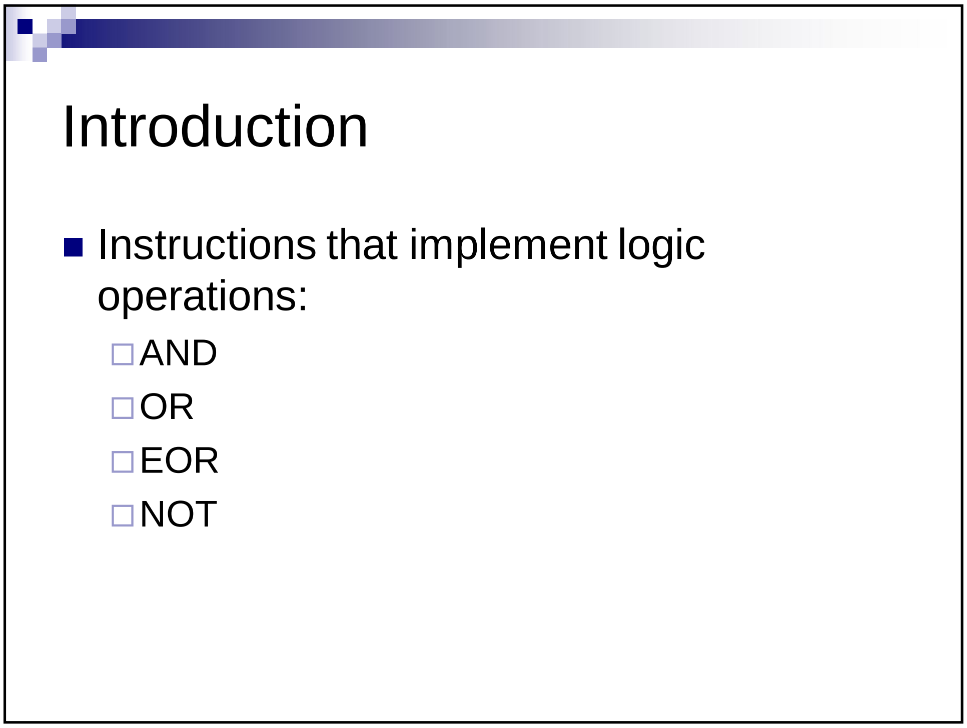
Example: BCD Representation

- Represent 564 in BCD.



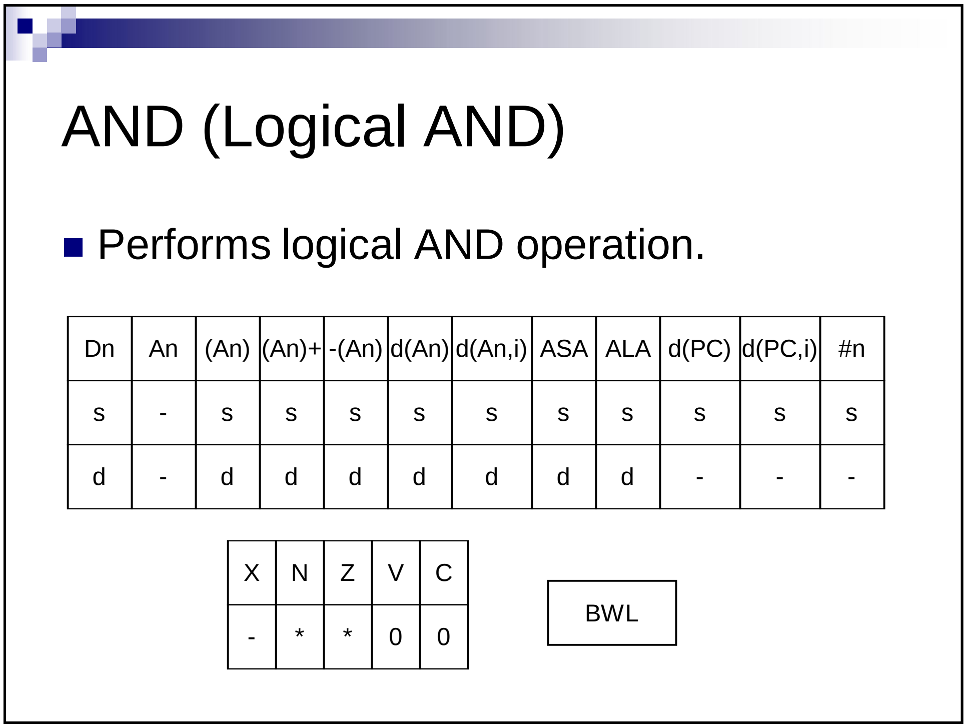
564_{BCD} = 010101100100

Logical Group



Introduction

- Instructions that implement logic operations:
  - □ AND
  - □ OR
  - □ EOR
  - □ NOT



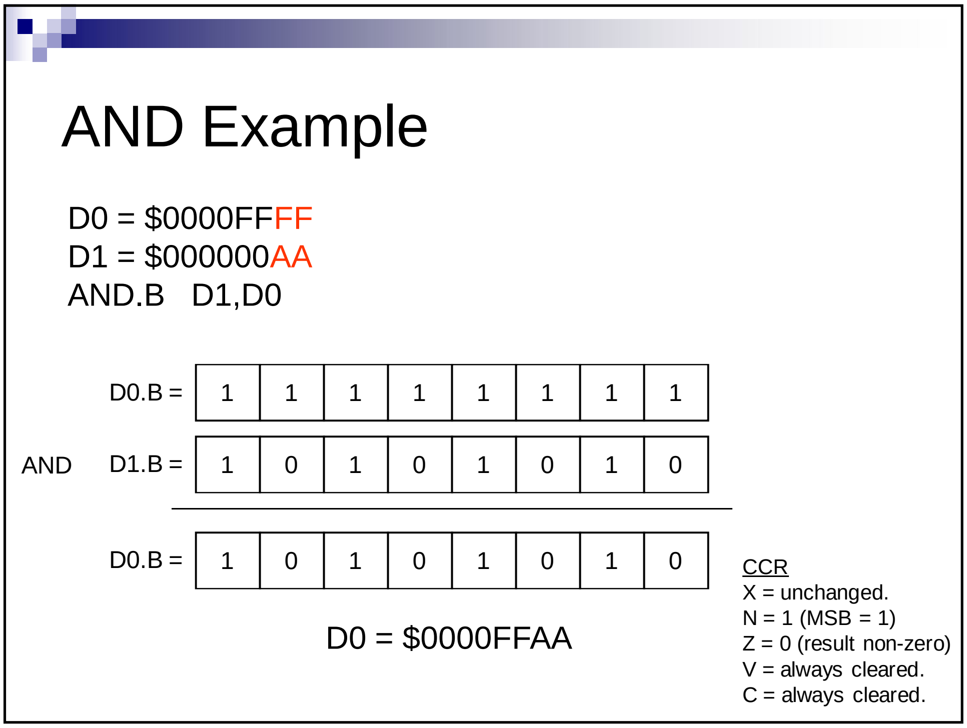
AND (Logical AND)

- Performs logical AND operation.

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	s	s	s	s	s	s	s	s	s	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	*	*	0	0

BWL



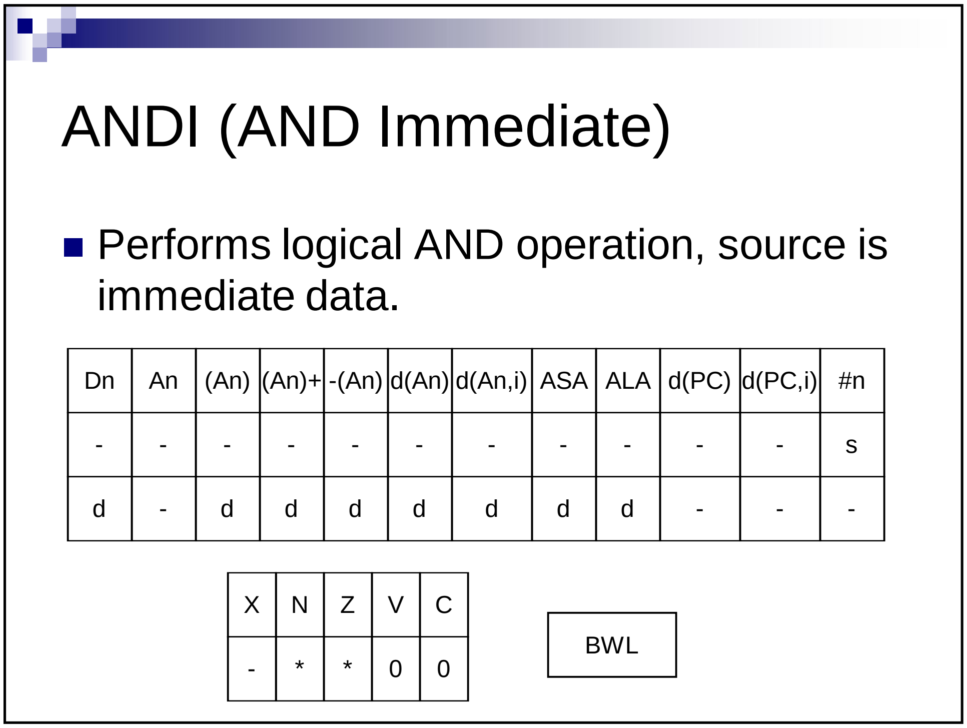
AND Example

D0 = $0000FFFF
D1 = $000000AA
AND.B   D1,D0

	D0.B =	1	1	1	1	1	1	1	1
AND	D1.B =	1	0	1	0	1	0	1	0
	D0.B =	1	0	1	0	1	0	1	0

D0 = $0000FFAA

CCR
X = unchanged.
N = 1 (MSB = 1)
Z = 0 (result non-zero)
V = always cleared.
C = always cleared.



ANDI (AND Immediate)

- Performs logical AND operation, source is immediate data.

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
-	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	*	*	0	0

BWL

ANDI Example

D0 = $0000FFFF
ANDI.B #$77, D0

	D0.B =	1	1	1	1	1	1	1	1
AND	D1.B =	0	1	1	1	0	1	1	1
	D0.B =	0	1	1	1	0	1	1	1

D0 = $0000FF77

CCR
X = unchanged.
N = 0 (MSB = 0)
Z = 0 (result non-zero)
V = always cleared.
C = always cleared.

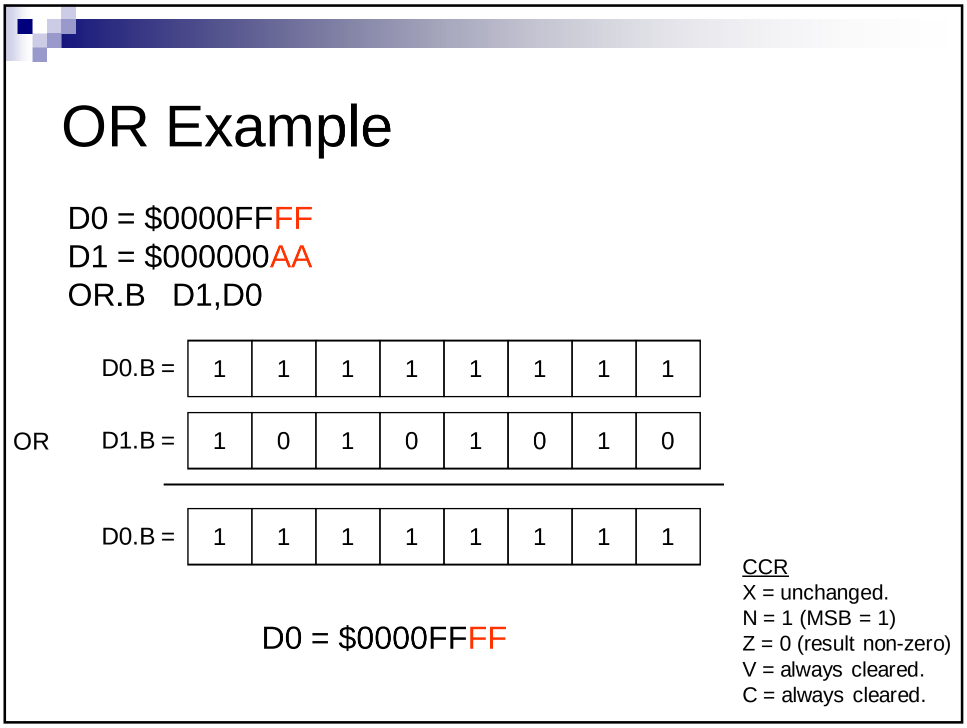
OR (Logical OR)

- Performs logical OR operation.

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	s	s	s	s	s	s	s	s	s	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	*	*	0	0

BWL



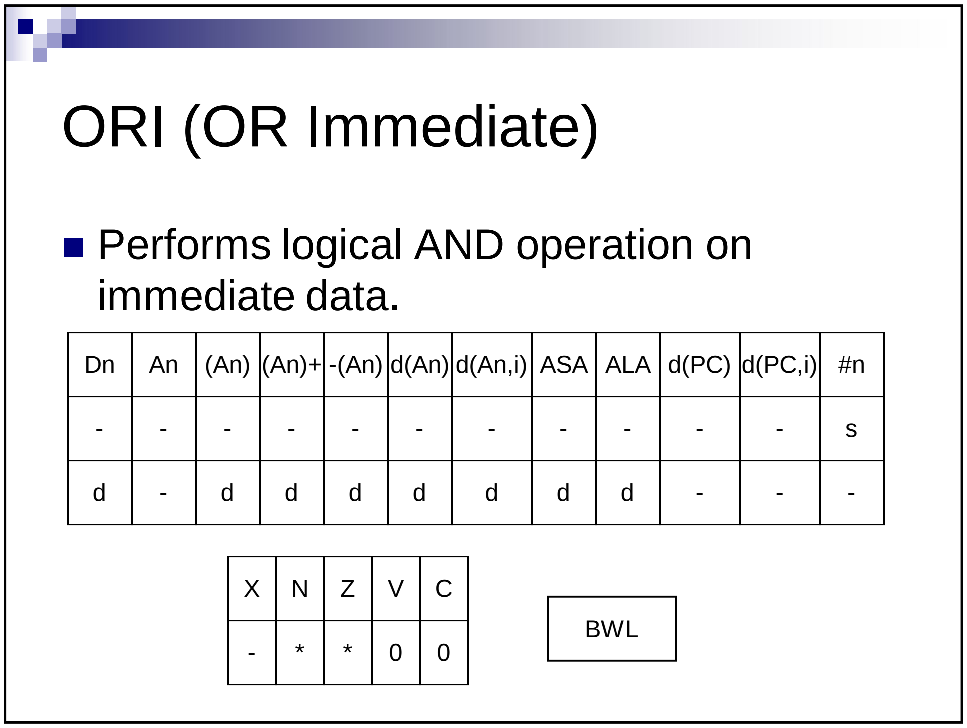
OR Example

D0 = $0000FFFF
D1 = $000000AA
OR.B   D1,D0

	D0.B =	1	1	1	1	1	1	1	1
OR	D1.B =	1	0	1	0	1	0	1	0
	D0.B =	1	1	1	1	1	1	1	1

D0 = $0000FFFF

CCR
X = unchanged.
N = 1 (MSB = 1)
Z = 0 (result non-zero)
V = always cleared.
C = always cleared.



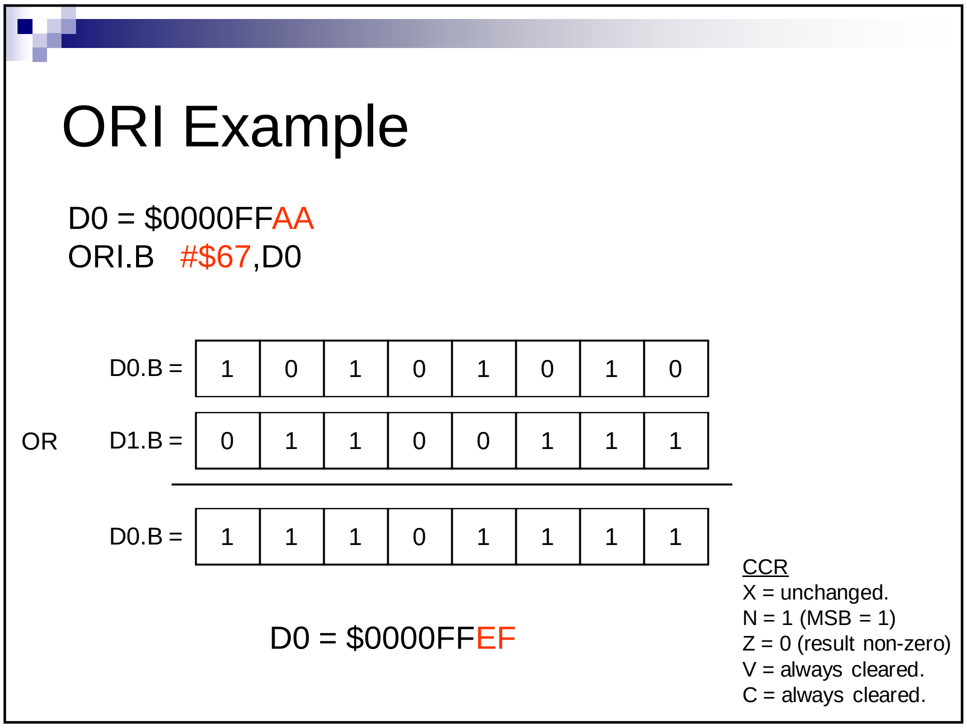
ORI (OR Immediate)

- Performs logical AND operation on immediate data.

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
-	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	*	*	0	0

BWL



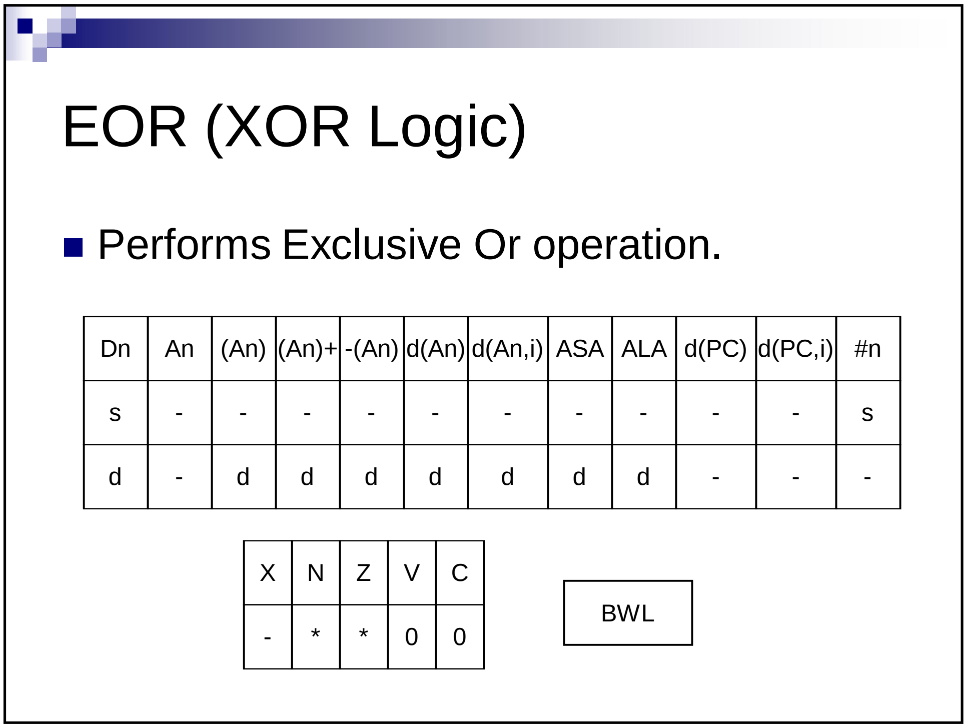
ORI Example

D0 = $0000FFAA
ORI.B   #$67,D0

	D0.B =	1	0	1	0	1	0	1	0
OR	D1.B =	0	1	1	0	0	1	1	1
	D0.B =	1	1	1	0	1	1	1	1

D0 = $0000FFEF

CCR
X = unchanged.
N = 1 (MSB = 1)
Z = 0 (result non-zero)
V = always cleared.
C = always cleared.



EOR (XOR Logic)

- Performs Exclusive Or operation.

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	*	*	0	0

BWL

EOR Example

D0 = $0000FFBB
D1 = $000000AA
EOR.B D1,D0

	D0.B =	1	0	1	1	1	0	1	1
XOR	D1.B =	1	0	1	0	1	0	1	0
	D0.B =	0	0	0	1	0	0	0	1

D0 = $0000FF11

CCR
X = unchanged.
N = 0 (MSB = 0)
Z = 0 (result non-zero)
V = always cleared.
C = always cleared.

EORI (EOR Immediate)

- Similar to EOR, but source is <id>.

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
-	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	*	*	0	0

BWL

EORI Example

D0 = $0000FFBB
EORI.B #$67,D0

	D0.B =	1	0	1	1	1	0	1	1
XOR	D1.B =	0	1	1	0	0	1	1	1
	D0.B =	1	1	0	1	1	1	0	0

D0 = $0000FFDC

CCR
X = unchanged.
N = 1 (MSB = 1)
Z = 0 (result non-zero)
V = always cleared.
C = always cleared.

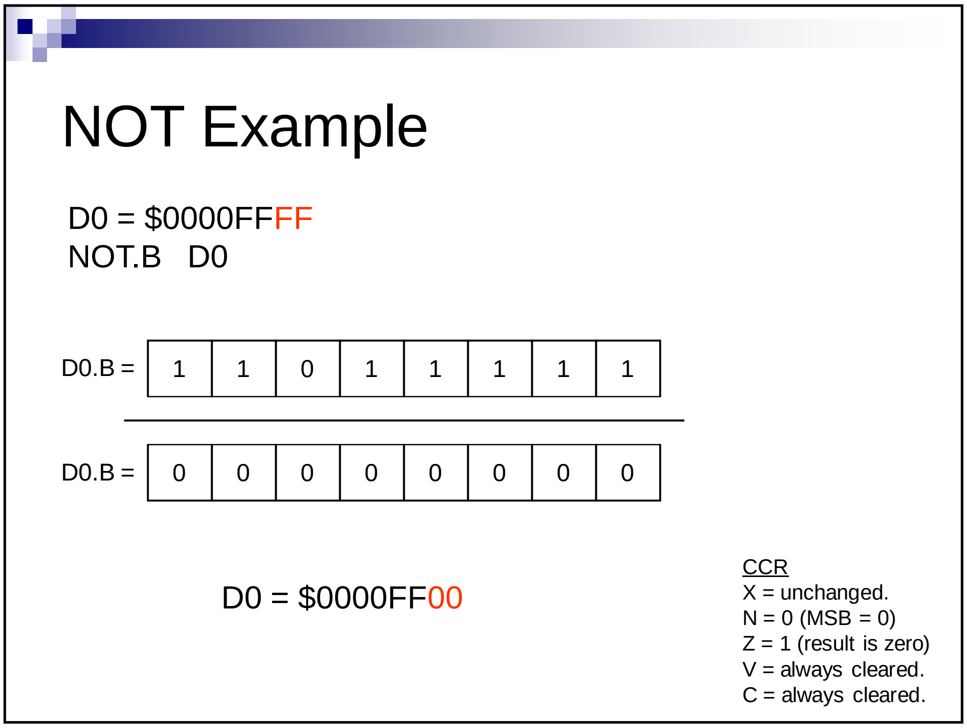
NOT (Not Logic)

- Inverts bits:
  - □ If 1 ➔ 0
  - □ If 0 ➔ 1

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
-	-	-	-	-	-	-	-	-	-	-	-
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	*	*	0	0

BWL



NOT Example

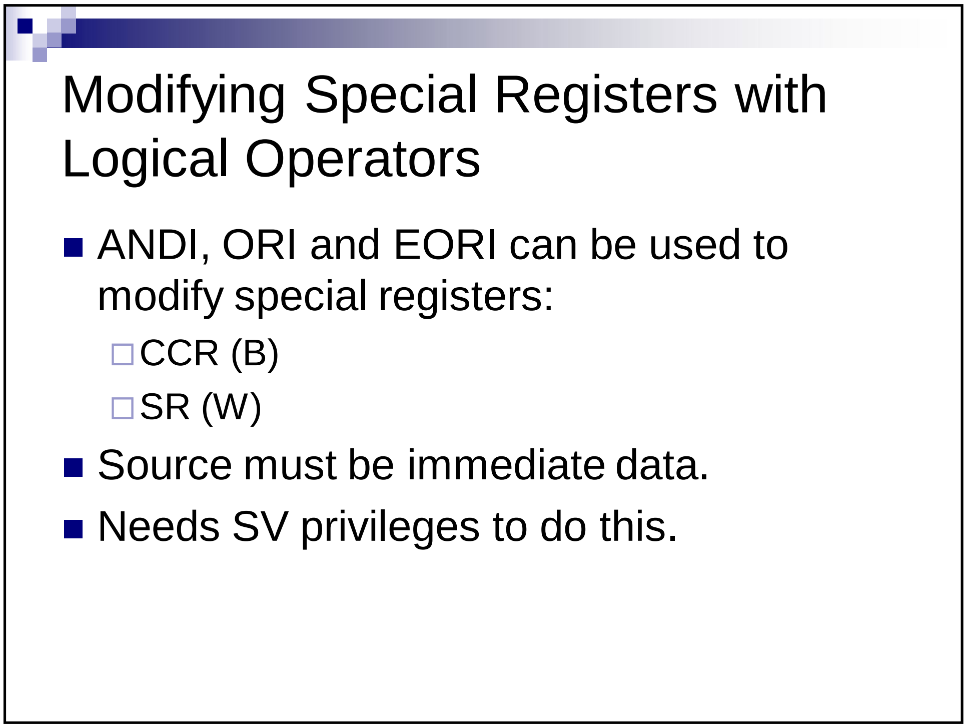
D0 = $0000FFFF
NOT.B D0

D0.B =	1	1	0	1	1	1	1	1

D0.B =	0	0	0	0	0	0	0	0

D0 = $0000FF00

CCR
X = unchanged.
N = 0 (MSB = 0)
Z = 1 (result is zero)
V = always cleared.
C = always cleared.



Modifying Special Registers with Logical Operators

- ANDI, ORI and EORI can be used to modify special registers:
  - CCR (B)
  - SR (W)
- Source must be immediate data.
- Needs SV privileges to do this.

The Status Register

T	▨	S	▨	▨	I_2	I_1	I_0	▨	▨	▨	X	N	Z	V	C

CCR

T = Trace Flag.
S = Supervisor Flag.
I_2, I_1, I_0 = Interrupt Mask Flags.
X = Extend Flag.
N = Negative Flag.
Z = Zero Flag.
V = Overflow Flag.
C = Carry Flag.

Example: Clear all bits in CCR

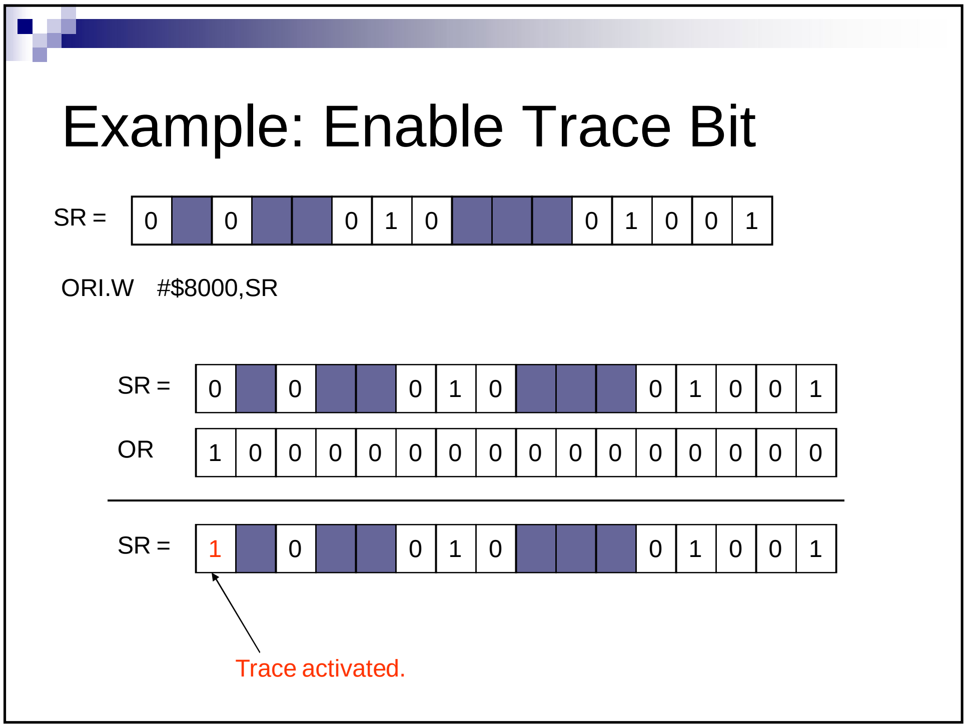
CCR =

0	1	0	0	1

ANDI.B #00,CCR

*CCR extended to 8-bits before operation

CCR =	0	0	0	0	1	0	0	1
ANDI	0	0	0	0	0	0	0	0
CCR =	0	0	0	0	0	0	0	0



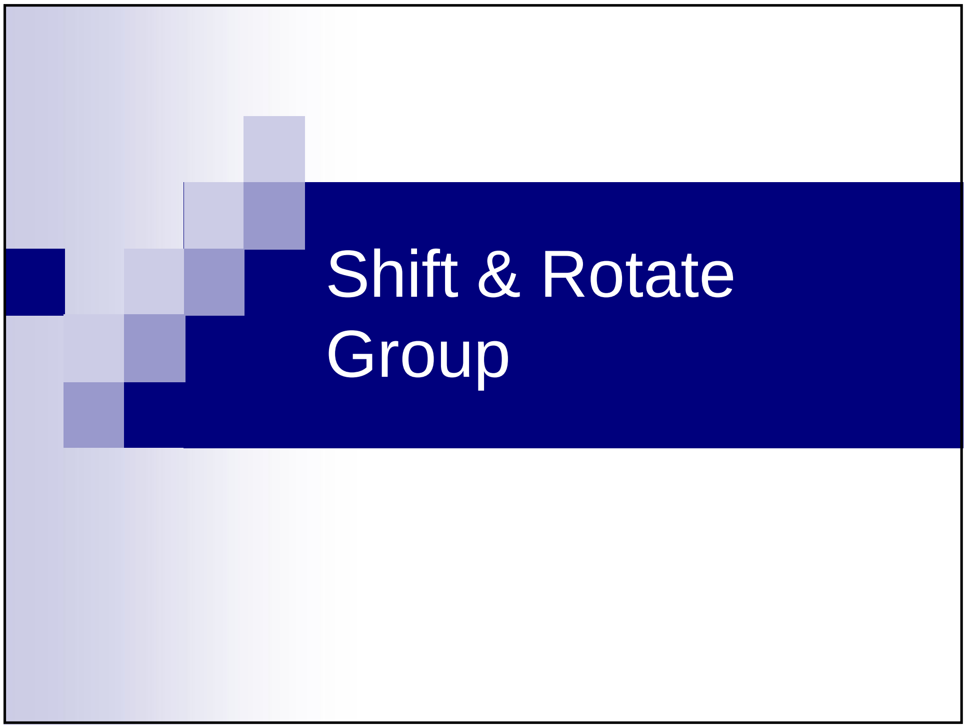
Example: Enable Trace Bit

SR =	0	▨	0	▨	▨	0	1	0	▨	▨	▨	0	1	0	0	1

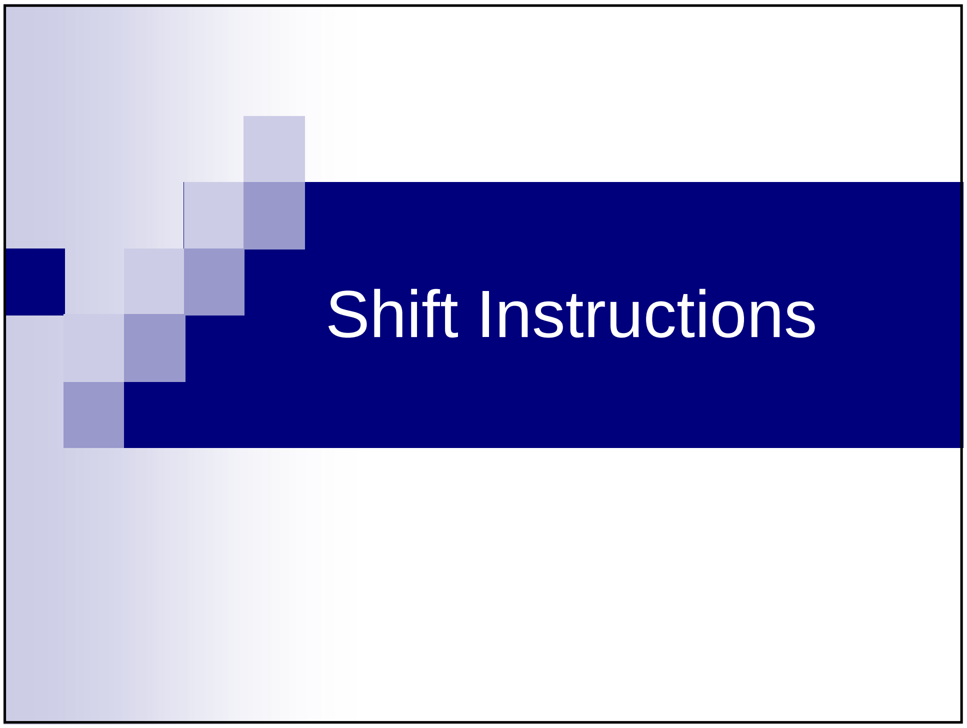
```
ORI.W   #$8000,SR
```

SR =	0	▨	0	▨	▨	0	1	0	▨	▨	▨	0	1	0	0	1
OR	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SR =	1	▨	0	▨	▨	0	1	0	▨	▨	▨	0	1	0	0	1

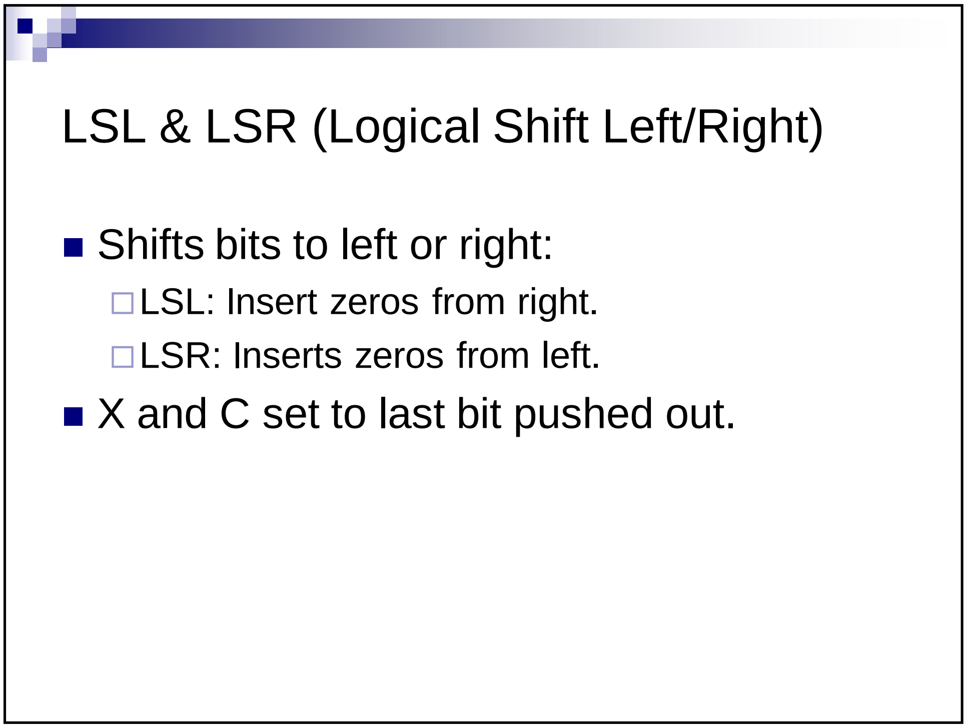
Trace activated.



Shift & Rotate Group



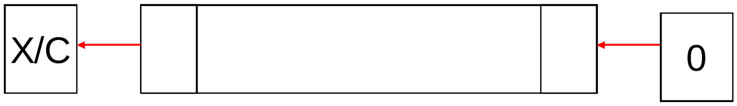
Shift Instructions



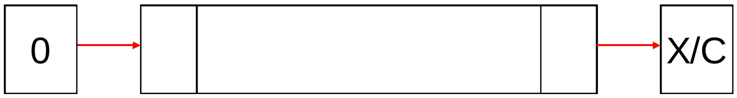
LSL & LSR (Logical Shift Left/Right)

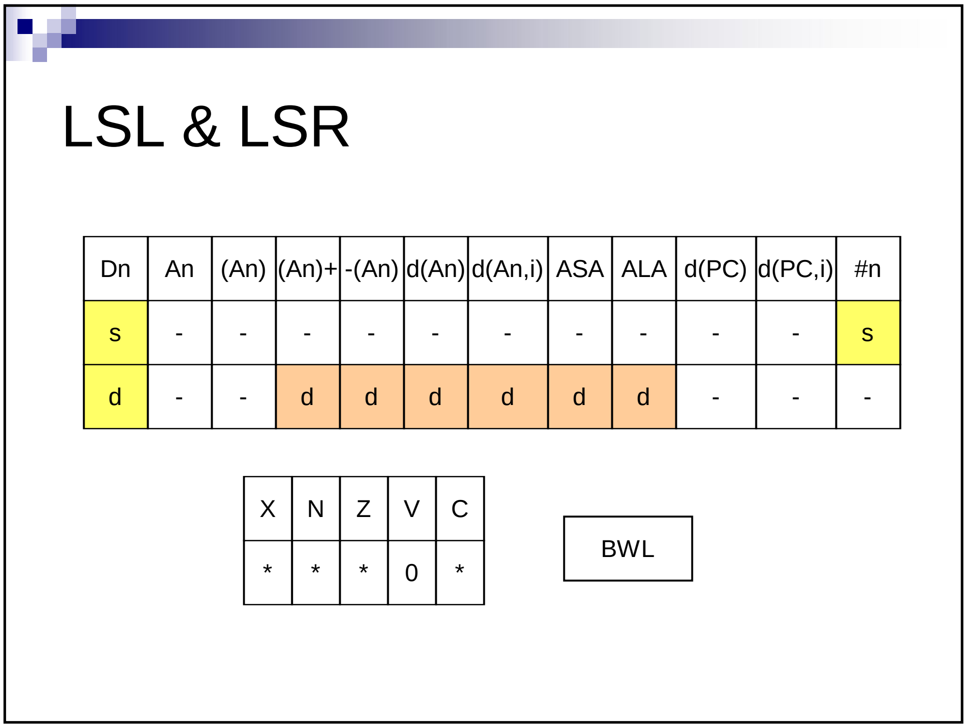
- Shifts bits to left or right:
  - □LSL: Insert zeros from right.
  - □LSR: Inserts zeros from left.
- X and C set to last bit pushed out.

LSL (Logical Shift Left)



LSR (Logical Shift Right)



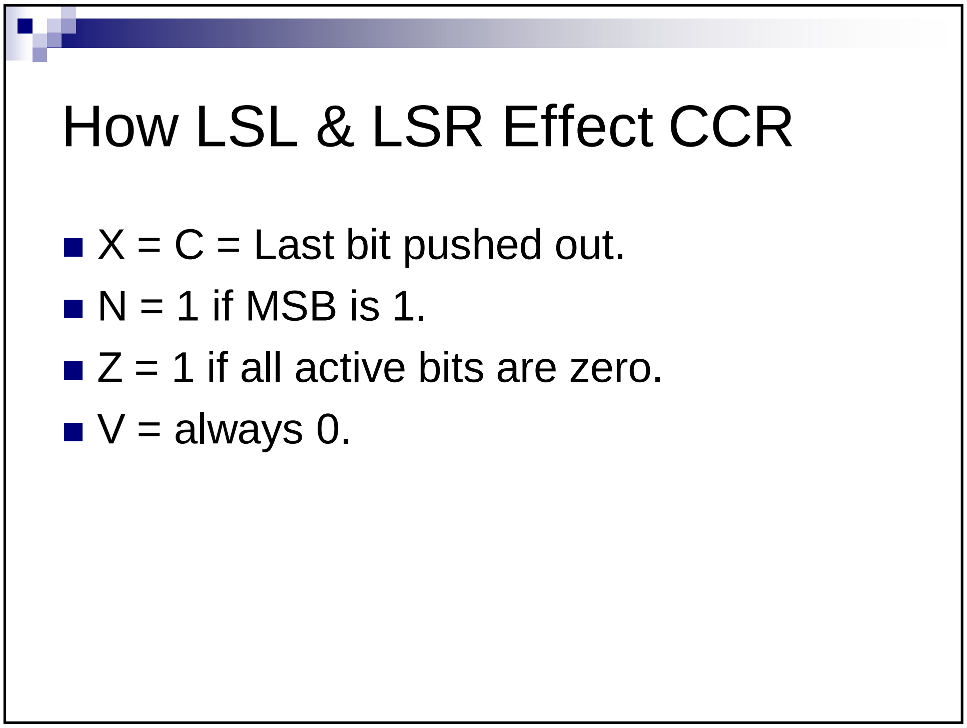


LSL & LSR

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	-	-	-	-	-	-	-	s
d	-	-	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
*	*	*	0	*

BWL

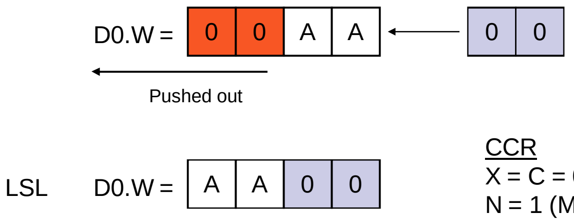


How LSL & LSR Effect CCR

- X = C = Last bit pushed out.
- N = 1 if MSB is 1.
- Z = 1 if all active bits are zero.
- V = always 0.

Example: LSL

D0 = $000000AA
D1 = $00000008
LSL.W D1,D0



D0.W = 0 0 A A ← 0 0

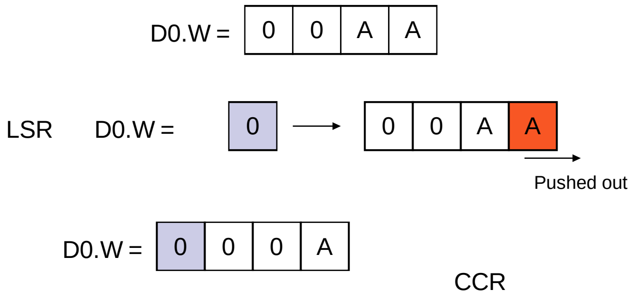
Pushed out

LSL D0.W = A A 0 0

CCR
X = C = 0 (last bit pushed out)
N = 1 (MSB = 1)
Z = 0 (D0.W not zero)
V = 0 (Always cleared)

Example: LSR

D0 = $000000AA
LSR.W #4,D0



D0.W = 0 0 A A

LSR D0.W = 0 → 0 0 A A

Pushed out

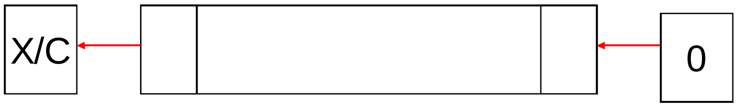
D0.W = 0 0 0 A

CCR
X = C = 1 (last bit pushed out)
N = 0 (MSB = 0)
Z = 0 (D0.W not zero)
V = 0 (Always cleared)

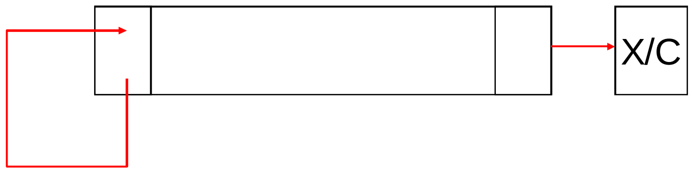
ASL & ASR (Arithmetic Shift Left/Right)

- Shifts bits to left or right:
  - ASL: Insert zeros from right.
  - ASR: Inserts MSB from left.
- X and C set to last bit pushed out.

ASL (Arithmetic Shift Left)



ASR (Arithmetic Shift Right)



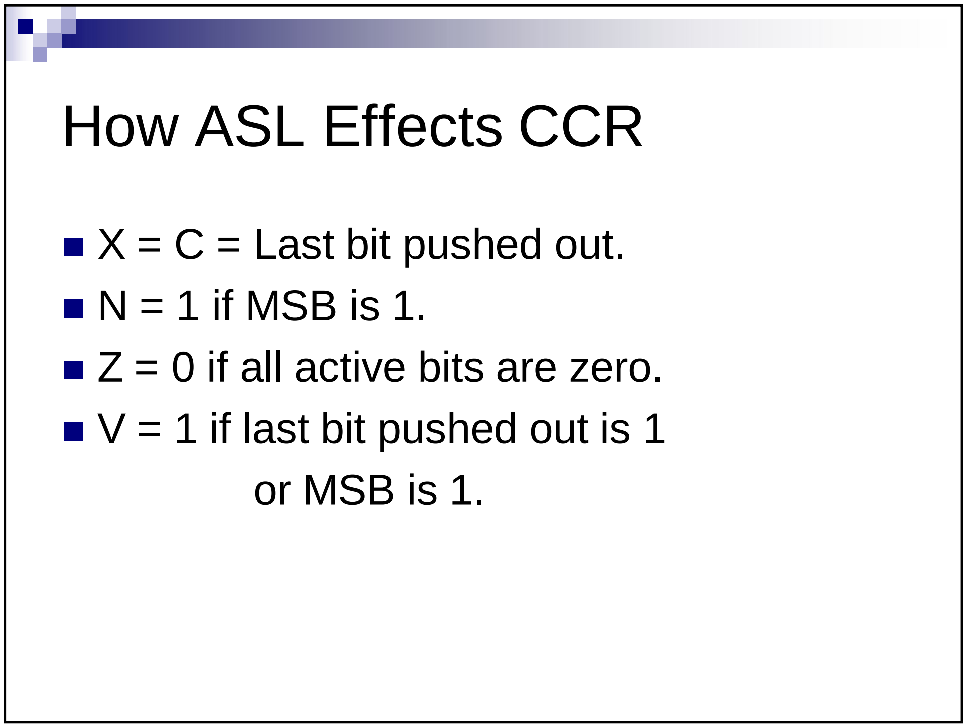
ASL & ASR

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
*	*	*	*	*

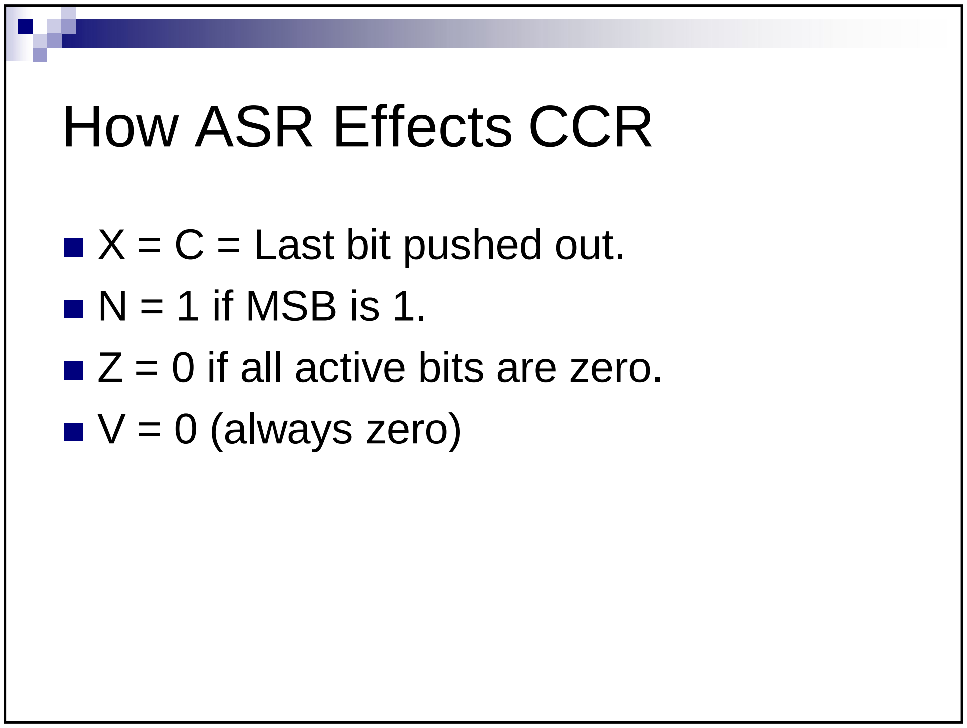
BWL

V = PO + NMSB (pushed out or new MSB)



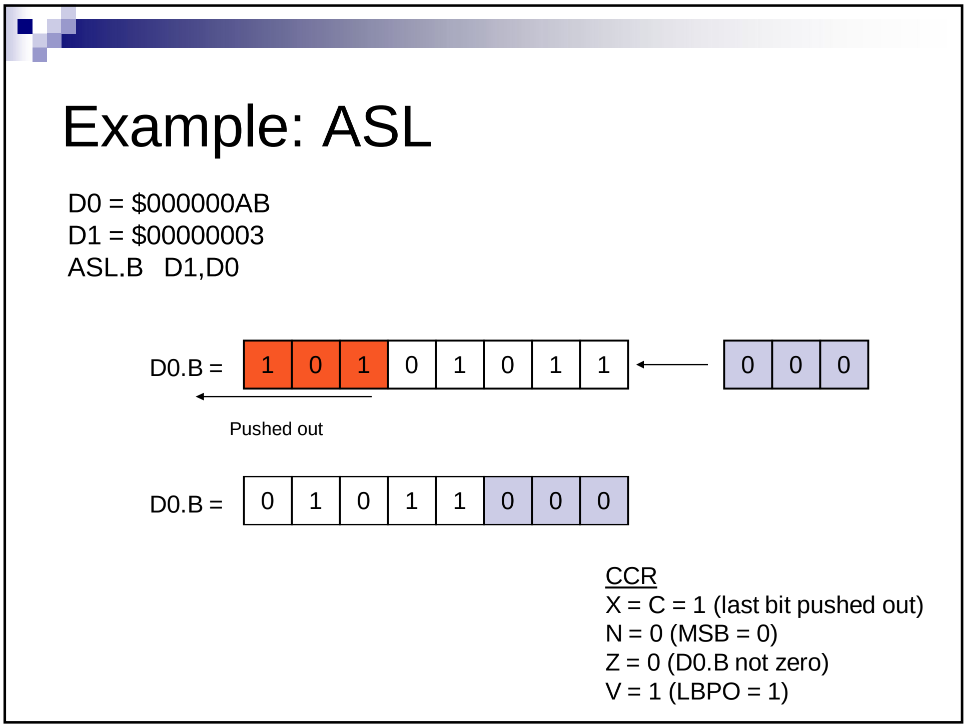
How ASL Effects CCR

- X = C = Last bit pushed out.
- N = 1 if MSB is 1.
- Z = 0 if all active bits are zero.
- V = 1 if last bit pushed out is 1
  or MSB is 1.



How ASR Effects CCR

- X = C = Last bit pushed out.
- N = 1 if MSB is 1.
- Z = 0 if all active bits are zero.
- V = 0 (always zero)



Example: ASL

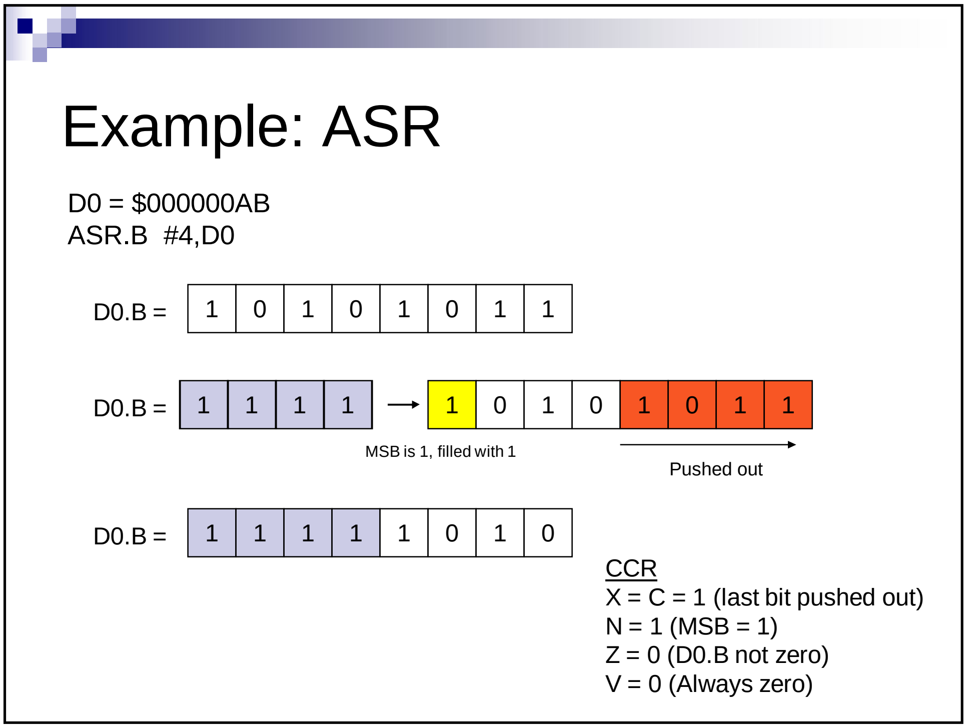
D0 = $000000AB
D1 = $00000003
ASL.B   D1,D0

D0.B =	1	0	1	0	1	0	1	1	←	0	0	0

Pushed out

D0.B =	0	1	0	1	1	0	0	0

CCR
X = C = 1 (last bit pushed out)
N = 0 (MSB = 0)
Z = 0 (D0.B not zero)
V = 1 (LBPO = 1)



Example: ASR

D0 = $000000AB
ASR.B #4,D0

D0.B =	1	0	1	0	1	0	1	1

D0.B =	1	1	1	1	→	1	0	1	0	1	0	1	1

MSB is 1, filled with 1

Pushed out

D0.B =	1	1	1	1	1	0	1	0

CCR
X = C = 1 (last bit pushed out)
N = 1 (MSB = 1)
Z = 0 (D0.B not zero)
V = 0 (Always zero)

Rotate Instructions

ROL & ROR (Rotate Left/Right)

- Pushes out MSB, and moves the bits to the back.
- C set to last bit pushed out.

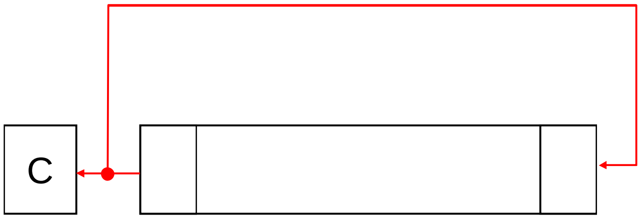
ROL & ROR

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

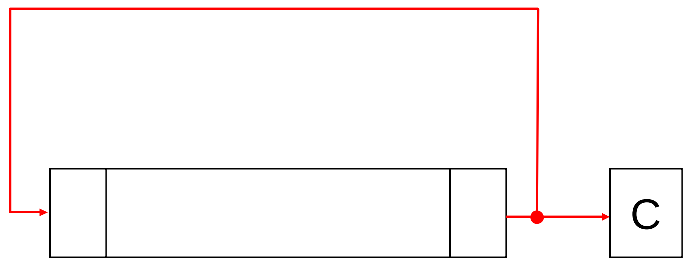
X	N	Z	V	C
-	*	*	0	*

BWL

ROL (Rotate Left)

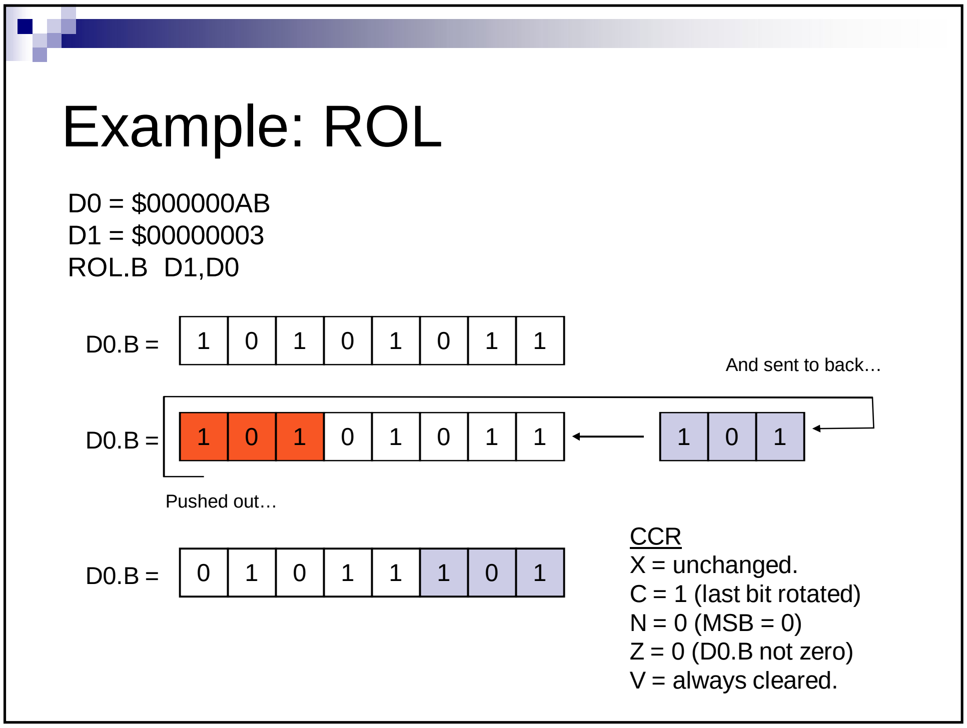


ROR (Rotate Right)



How ROL & ROR Effects CCR

- C = Last bit rotated.
- N = 1 if MSB is 1.
- Z = 0 if all active bits are zero.
- V = always cleared.
- X = unchanged.



Example: ROL

D0 = $000000AB
D1 = $00000003
ROL.B D1,D0

D0.B =	1	0	1	0	1	0	1	1

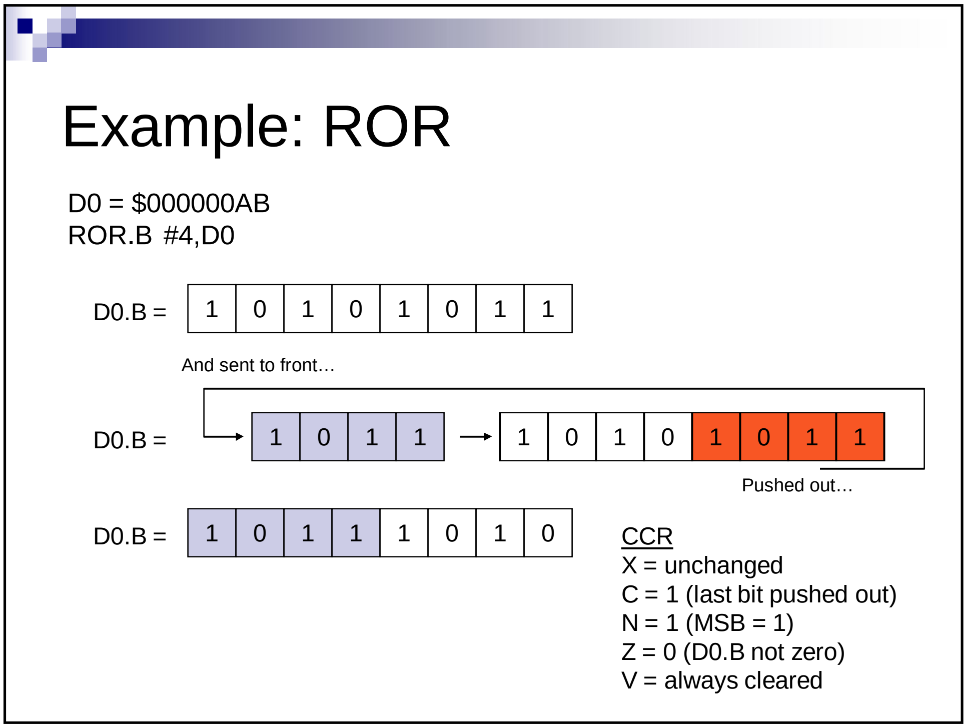
And sent to back…

D0.B =	1	0	1	0	1	0	1	1	←	1	0	1

Pushed out…

D0.B =	0	1	0	1	1	1	0	1

CCR
X = unchanged.
C = 1 (last bit rotated)
N = 0 (MSB = 0)
Z = 0 (D0.B not zero)
V = always cleared.



Example: ROR

D0 = $000000AB
ROR.B #4,D0

D0.B =	1	0	1	0	1	0	1	1

And sent to front…

D0.B =	1	0	1	1	→	1	0	1	0	1	0	1	1

Pushed out…

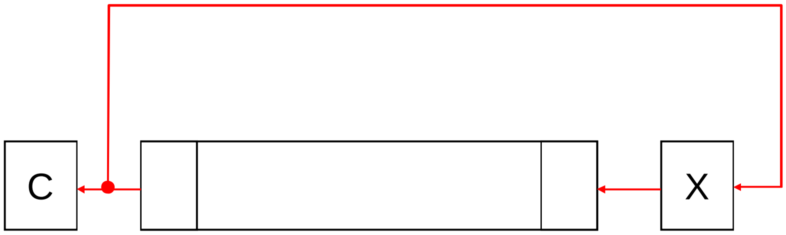
D0.B =	1	0	1	1	1	0	1	0

CCR
X = unchanged
C = 1 (last bit pushed out)
N = 1 (MSB = 1)
Z = 0 (D0.B not zero)
V = always cleared

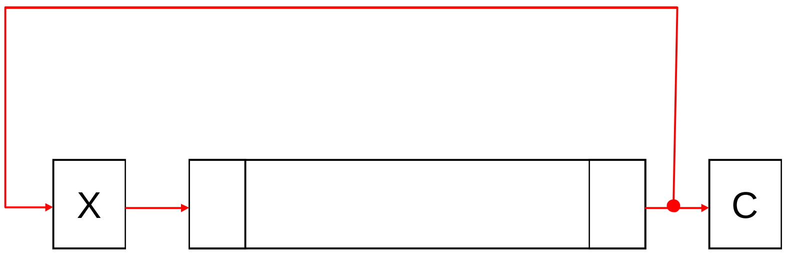
ROXL & ROXR (Rotate with Extend Left/Right)

- Same with ROR and ROL, but X bit becomes an extra place to store the extra bit.
- Last rotated bit stored in C and X.

ROXL (Rotate with Extend Left)



ROXR (Rotate with Extend Right)



ROXL & ROXR

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
*	*	*	0	*

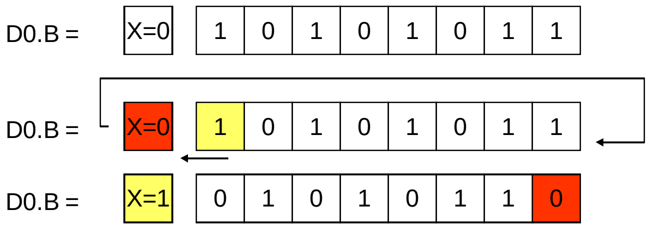
BWL

How ROXL & ROXR Effects CCR

- X = C = Last bit rotated.
- N = 1 if MSB is 1.
- Z = 0 if all active bits are zero.
- V = always cleared.

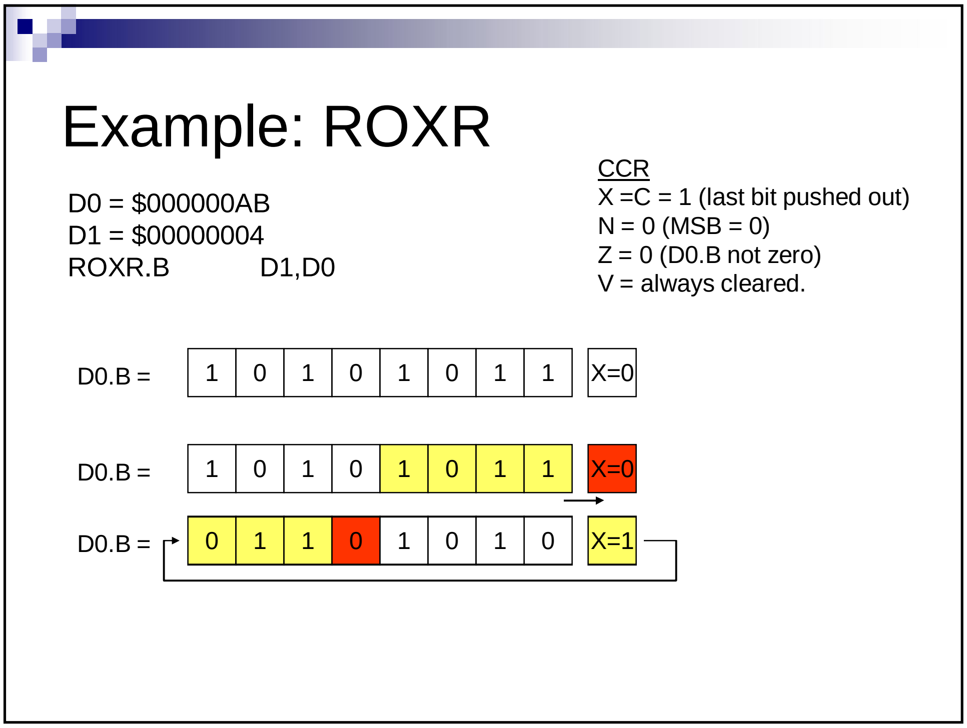
Example: ROXL

D0 = $000000AB
D1 = $00000001
ROXL.B D1,D0



D0.B =	X=0	1	0	1	0	1	0	1	1
D0.B =	X=0	1	0	1	0	1	0	1	1
D0.B =	X=1	0	1	0	1	0	1	1	0

CCR
X =C = 1 (last bit pushed out)
N = 0 (MSB = 0)
Z = 0 (D0.B not zero)
V = always cleared.



Example: ROXR

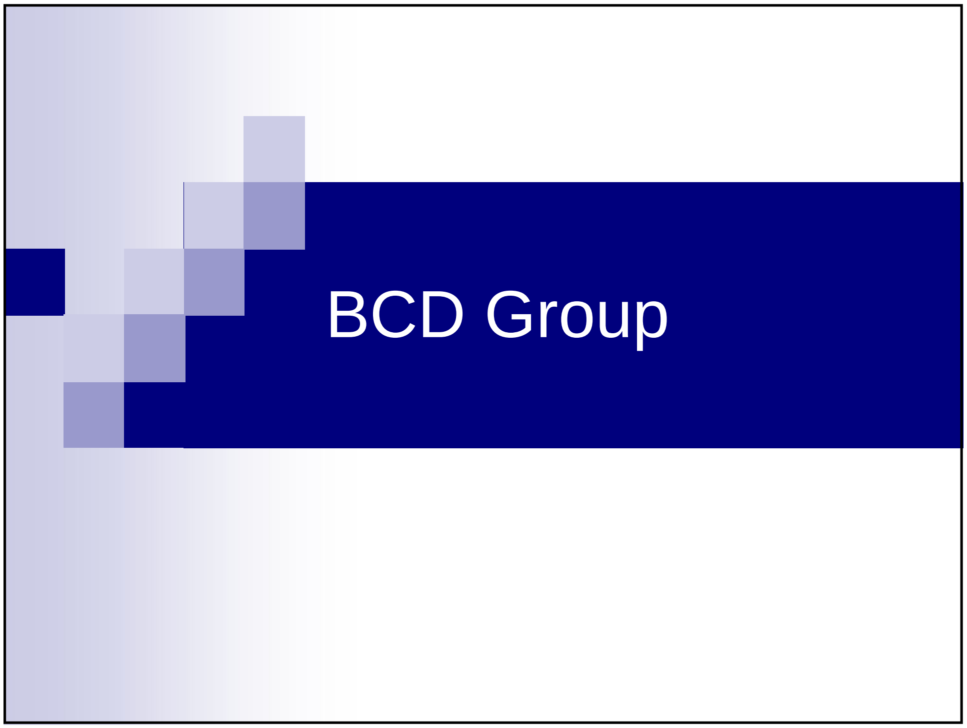
D0 = $000000AB
D1 = $00000004
ROXR.B D1,D0

CCR
X =C = 1 (last bit pushed out)
N = 0 (MSB = 0)
Z = 0 (D0.B not zero)
V = always cleared.

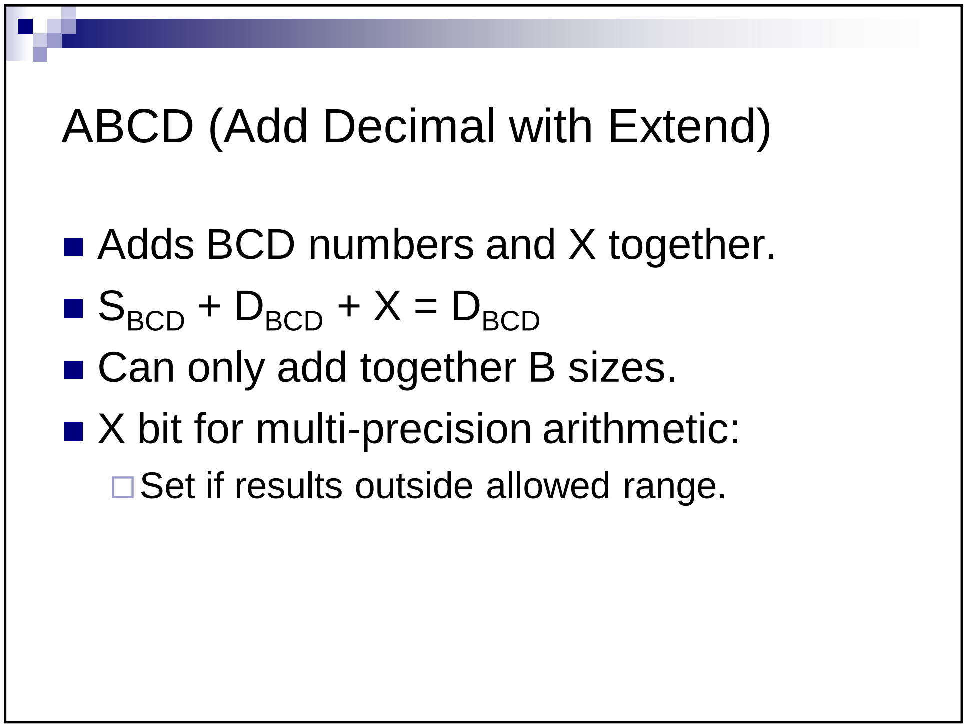
D0.B = 1 0 1 0 1 0 1 1 X=0

D0.B = 1 0 1 0 1 0 1 1 X=0

D0.B = 0 1 1 0 1 0 1 0 X=1

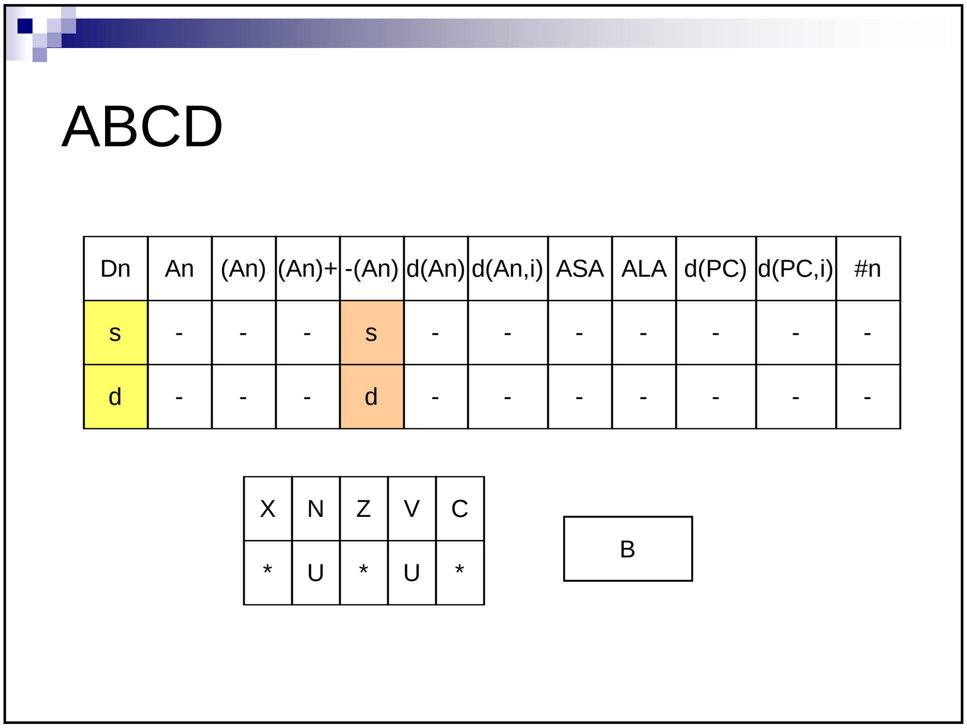


BCD Group



ABCD (Add Decimal with Extend)

- Adds BCD numbers and X together.
- $S_{BCD} + D_{BCD} + X = D_{BCD}$
- Can only add together B sizes.
- X bit for multi-precision arithmetic:
  - Set if results outside allowed range.

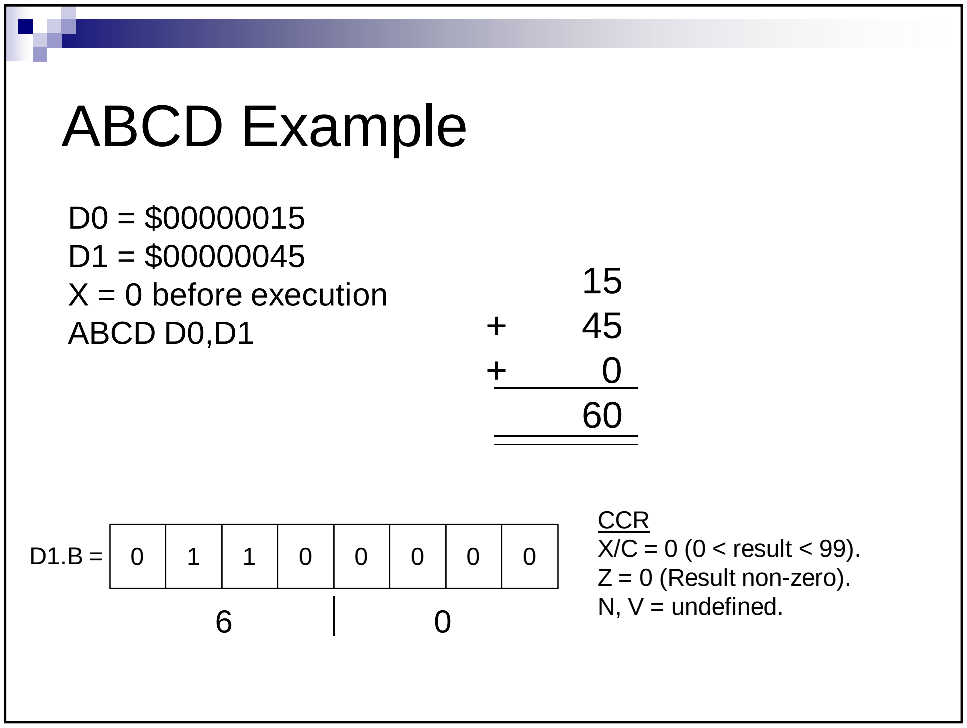


ABCD

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	s	-	-	-	-	-	-	-
d	-	-	-	d	-	-	-	-	-	-	-

X	N	Z	V	C
*	U	*	U	*

B



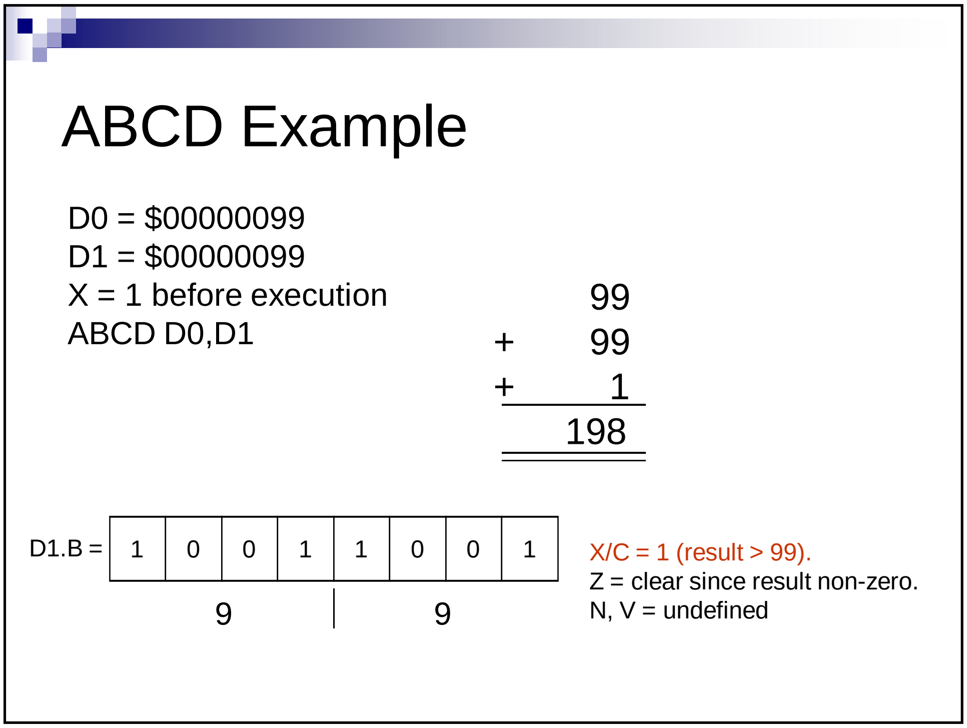
ABCD Example

D0 = $00000015
D1 = $00000045
X = 0 before execution
ABCD D0,D1

$$\begin{array}{rr} & 15 \\ + & 45 \\ + & 0 \\ \hline & 60 \\ \hline\hline \end{array}$$

D1.B =	0	1	1	0	0	0	0	0
	6				0			

CCR
X/C = 0 (0 < result < 99).
Z = 0 (Result non-zero).
N, V = undefined.



ABCD Example

D0 = $00000099
D1 = $00000099
X = 1 before execution
ABCD D0,D1

$$\begin{array}{rr} & 99 \\ + & 99 \\ + & 1 \\ \hline & 198 \\ \hline\hline \end{array}$$

D1.B =	1	0	0	1	1	0	0	1
	9				9			

X/C = 1 (result > 99).
Z = clear since result non-zero.
N, V = undefined

SBCD (Subtract Decimal with Extend)

- Subtracts BCD numbers.
- $D_{BCD} – S_{BCD} – X = D_{BCD}$
- Can only subtract B sizes.
- X bit for multi-precision arithmetic:
  - □ Set if results outside allowed range.

SBCD

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	s	-	-	-	-	-	-	-
d	-	-	-	d	-	-	-	-	-	-	-

X	N	Z	V	C
*	U	*	U	*

B

SBCD Example

D0 = $00000025
D1 = $00000050
X = 1 before execution
SBCD D0,D1

```
    50
-   25
-    1
  ----
    24
  ====
```

D1.B =	0	0	1	0	0	1	0	0
		2				4		

CCR
X/C = 0 (0 < result < 99).
Z = clear since result non-zero.
N, V = undefined

SBCD Example

D0 = $00000099
D1 = $00000050
X = 0 before execution
SBCD   D0,D1

```
    50
-   99
-    0
  ----
  - 49
  ====
```

D0.B =	0	1	0	0	1	0	0	1
		4				9		

CCR
X/C = 1 (result < 0).
Z = clear since result non-zero.
N, V = undefined

NBCD (Negate BCD)

- Finds 10’s complement and 9’s complement of BCD number.
  - □ Equal to negative BCD number.
  - □ $00_{BCD} – D_{BCD} – X = D_{BCD}$
- If X = 1, finds 9’s complement:
- If X = 0, finds 10’s complement.

NBCD

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
1	U	*	U	1

B

In simple terms…

- NBCD can be used to find 9’s complement or 10’s complement.
- 10’s complement is 100 – (BCD number).
  - □ Set X = 0 before execution.
- 9’s complement is 99 – (BCD number).
  - □ Set X = 1 before execution.

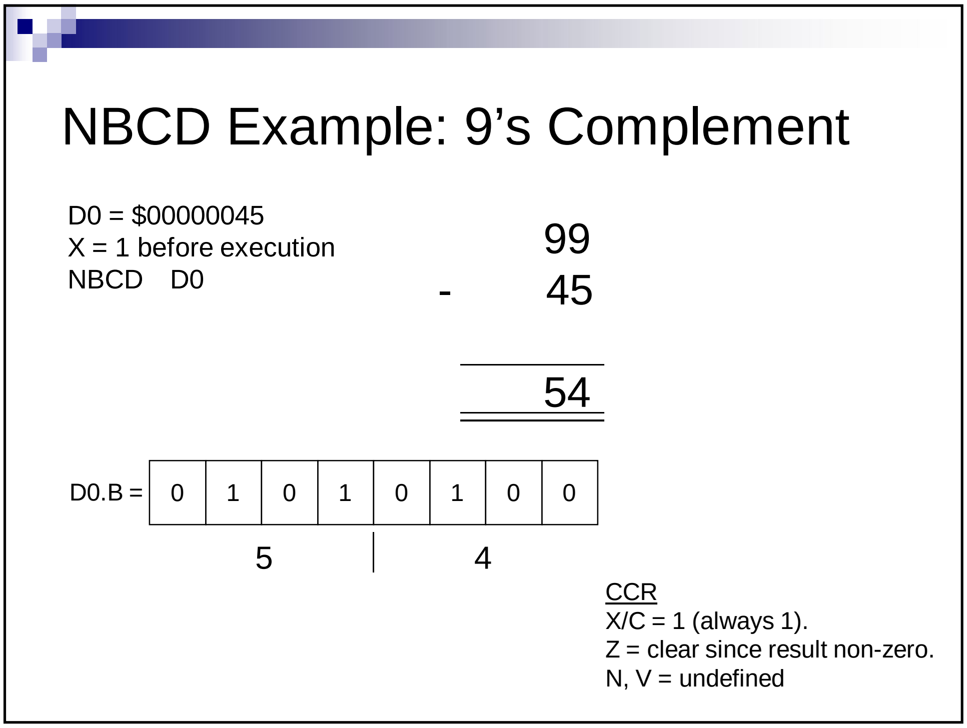
NBCD Example: 10’s Complement

D0 = $00000045
X = 0 before execution
NBCD D0

$$\begin{array}{rr} & 100 \\ - & 45 \\ \hline & 55 \\ \hline \end{array}$$

D0.B =	0	1	0	1	0	1	0	1
			5				5	

CCR
X/C = 1 (always 1).
Z = clear since result non-zero.
N, V = undefined



NBCD Example: 9’s Complement

D0 = $00000045
X = 1 before execution
NBCD D0

$$\begin{array}{r} 99 \\ -\quad 45 \\ \hline 54 \\ \hline \hline \end{array}$$

D0.B =	0	1	0	1	0	1	0	0
			5				4	

CCR
X/C = 1 (always 1).
Z = clear since result non-zero.
N, V = undefined

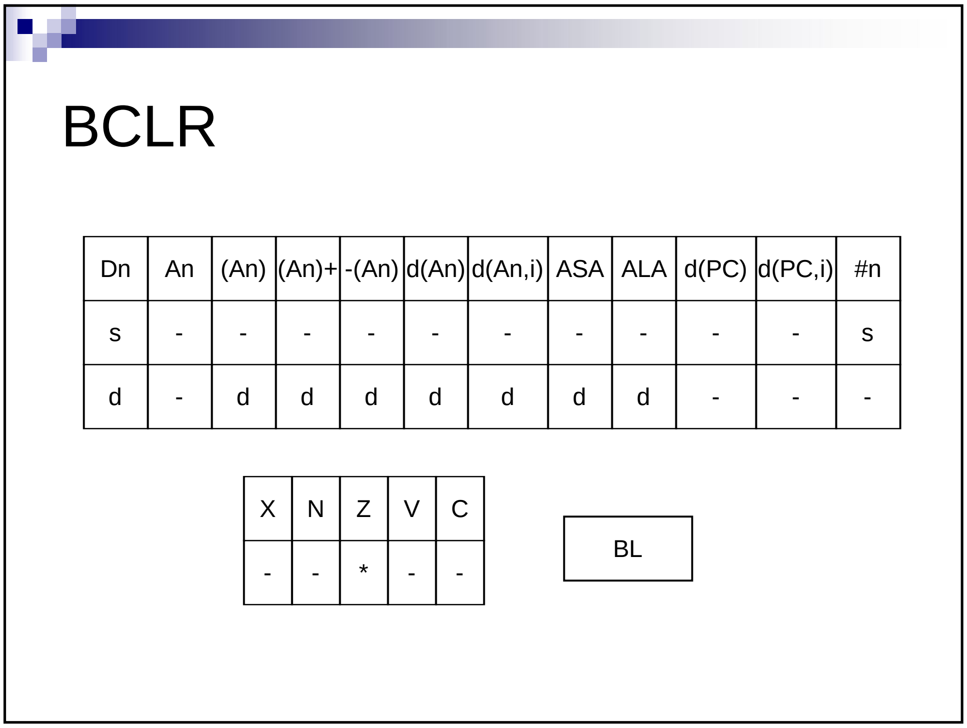
Bit Manipulation Group

Bit Manipulation Group

- Set of instructions to manipulate bits in register/memory.

BCLR (Test Bit & Change)

- Examines bit, modifies Z, then clear bit.
- Bit position specified by Dn or <id>:
- Effects only Z in CCR.

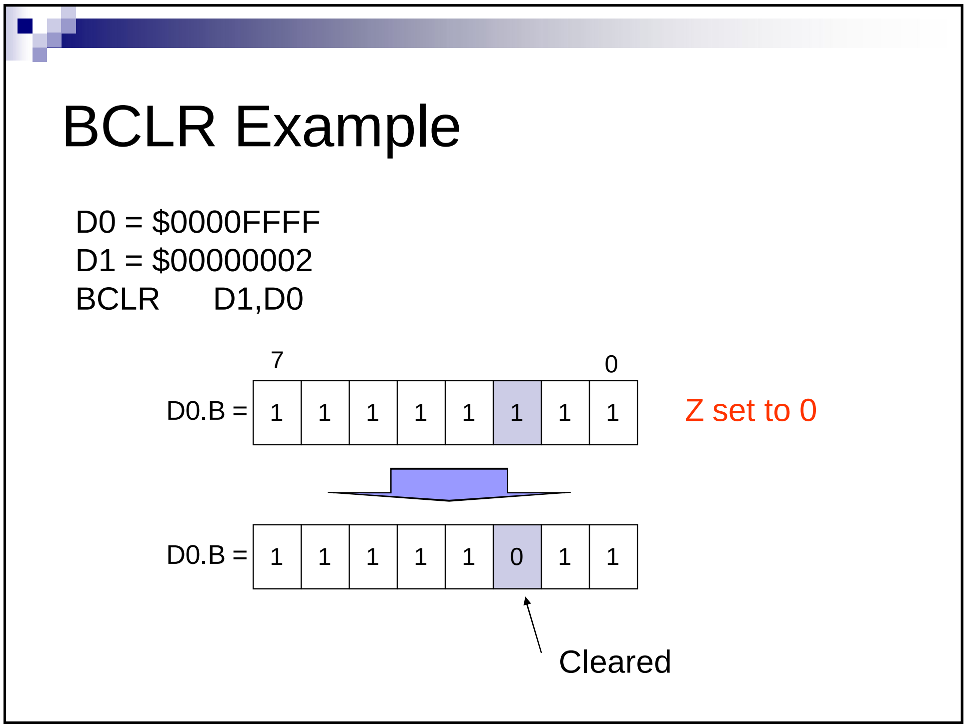


BCLR

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	-	*	-	-

BL



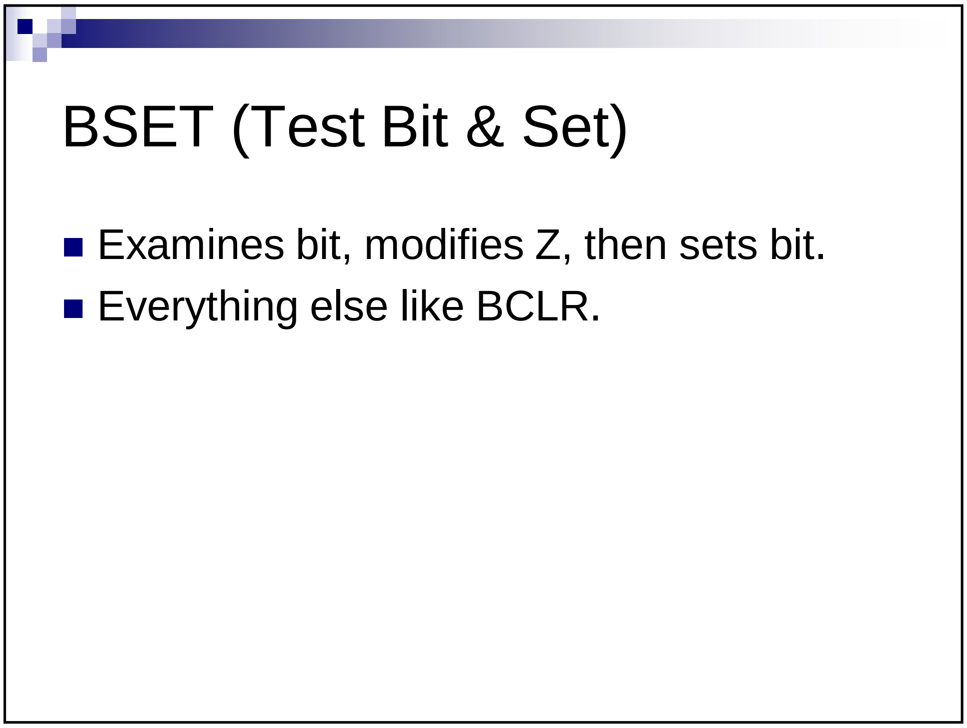
BCLR Example

D0 = $0000FFFF
D1 = $00000002
BCLR D1,D0

	7							0
D0.B =	1	1	1	1	1	1	1	1
D0.B =	1	1	1	1	1	0	1	1

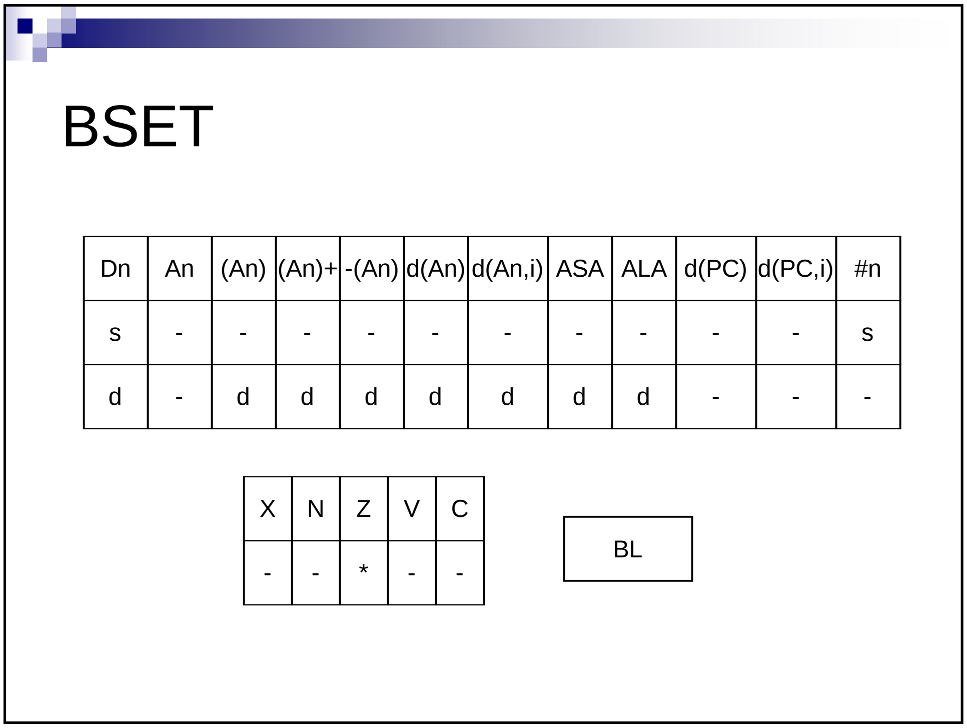
Z set to 0

Cleared



BSET (Test Bit & Set)

- Examines bit, modifies Z, then sets bit.
- Everything else like BCLR.

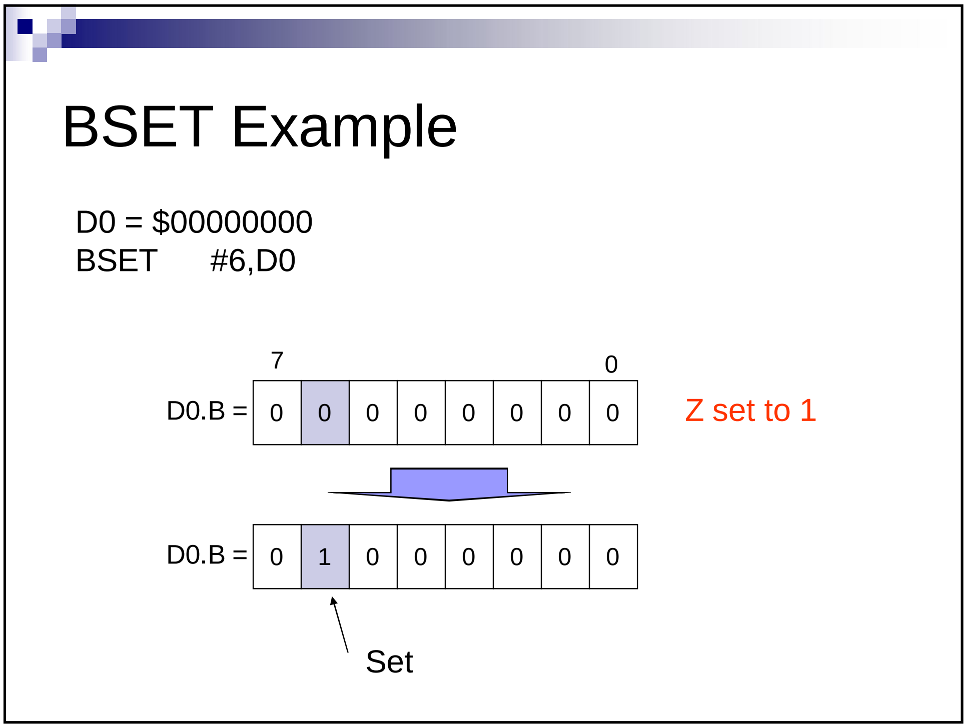


BSET

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	-	*	-	-

BL



BSET Example

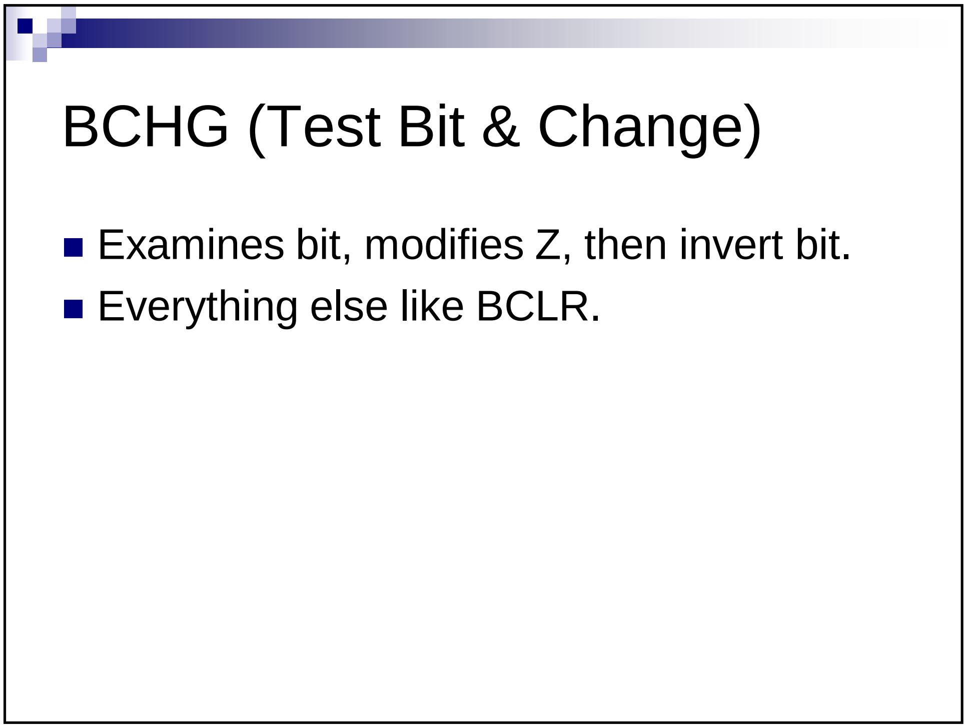
D0 = $00000000
BSET #6,D0

	7							0
D0.B =	0	0	0	0	0	0	0	0

Z set to 1

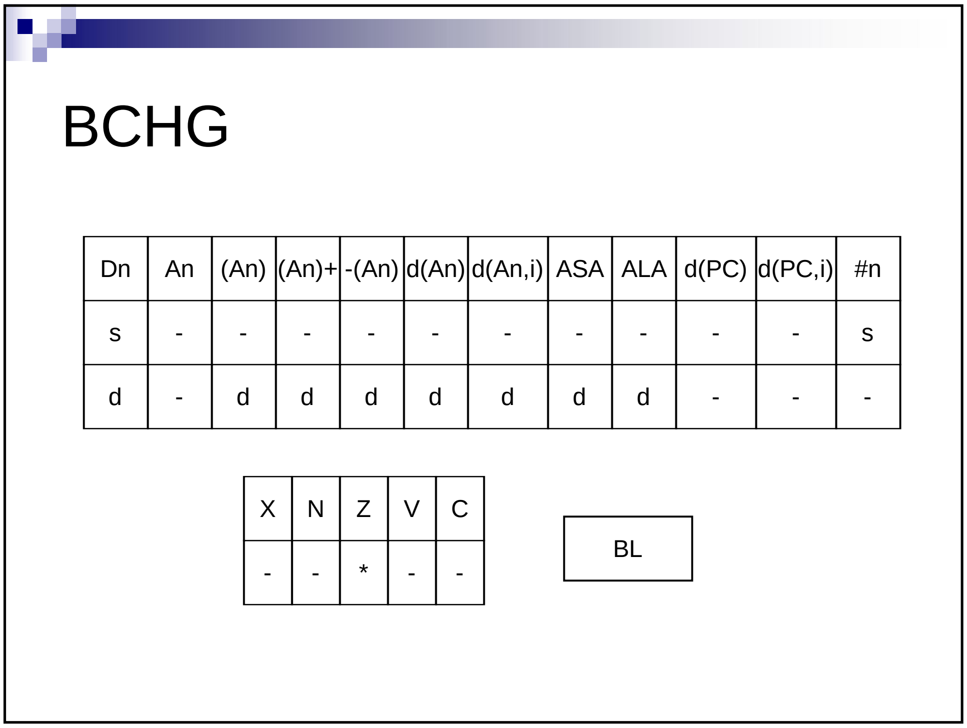
D0.B =	0	1	0	0	0	0	0	0

Set



BCHG (Test Bit & Change)

- Examines bit, modifies Z, then invert bit.
- Everything else like BCLR.

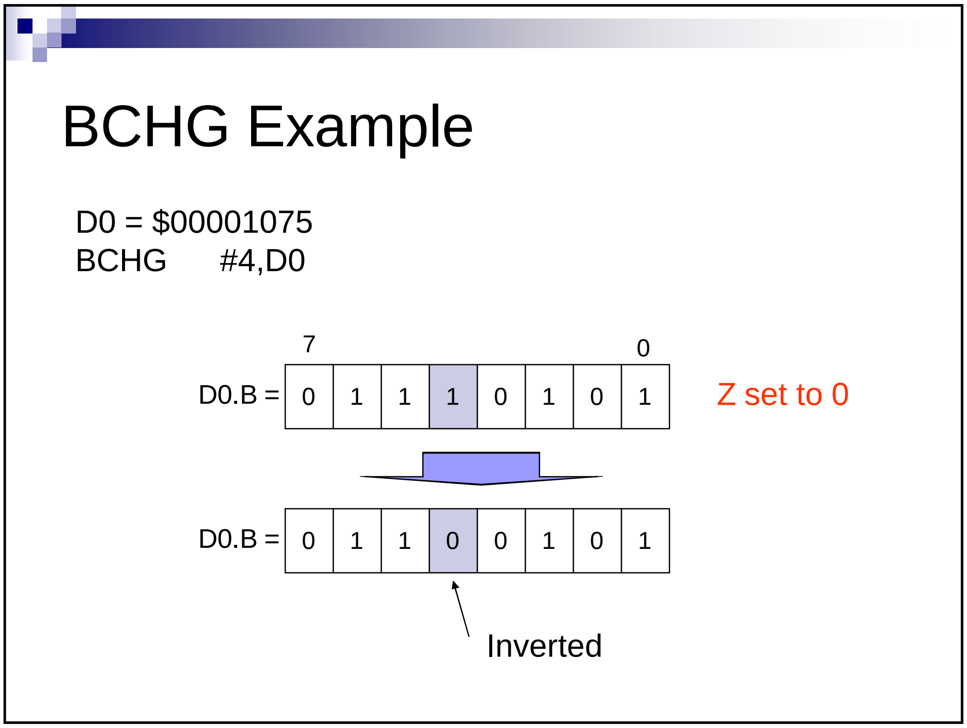


BCHG

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	-	*	-	-

BL



BCHG Example

D0 = $00001075
BCHG #4,D0

	7							0
D0.B =	0	1	1	1	0	1	0	1
D0.B =	0	1	1	0	0	1	0	1

Z set to 0

Inverted

BTST (Test Bit)

- Only examine bit, and modify Z.
- Data in register unchanged.
- Everything else like BCLR.

BCLR

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	-	*	-	-

BL

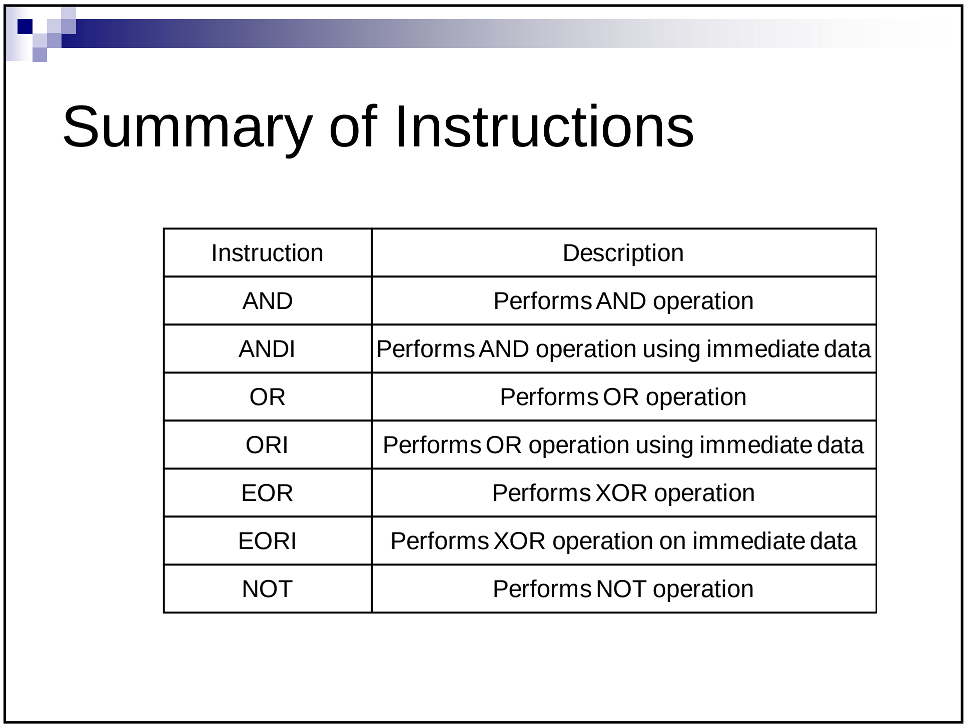
BTST Example

D0 = $00001075
BTST #7,D0

	7							0
D0.B =	0	1	1	1	0	1	0	1

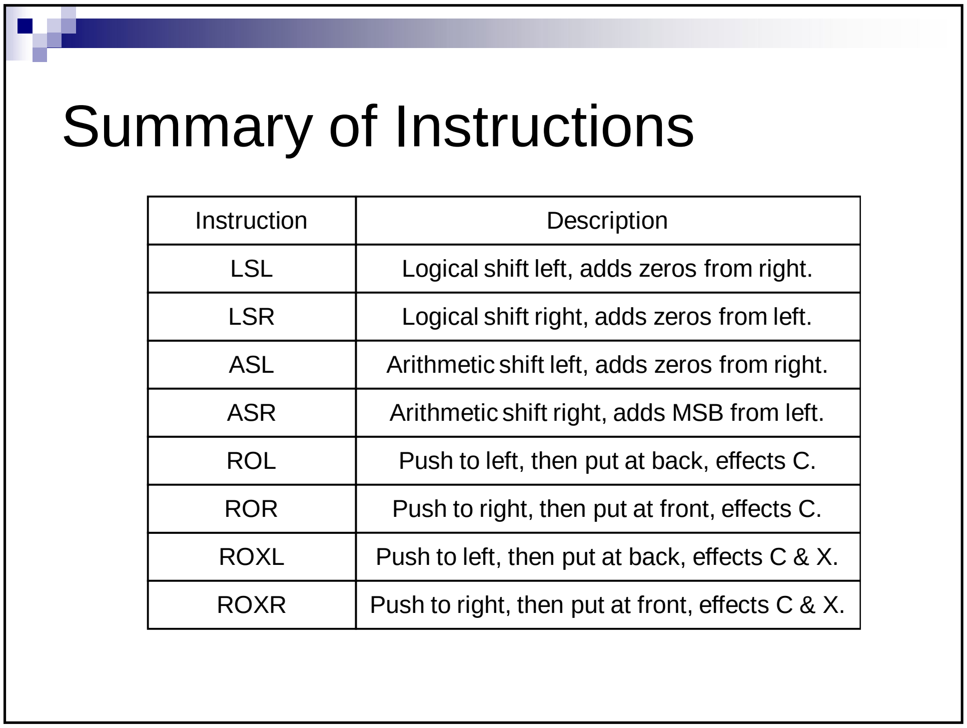
Z set to 1
D0 unchanged

Conclusion



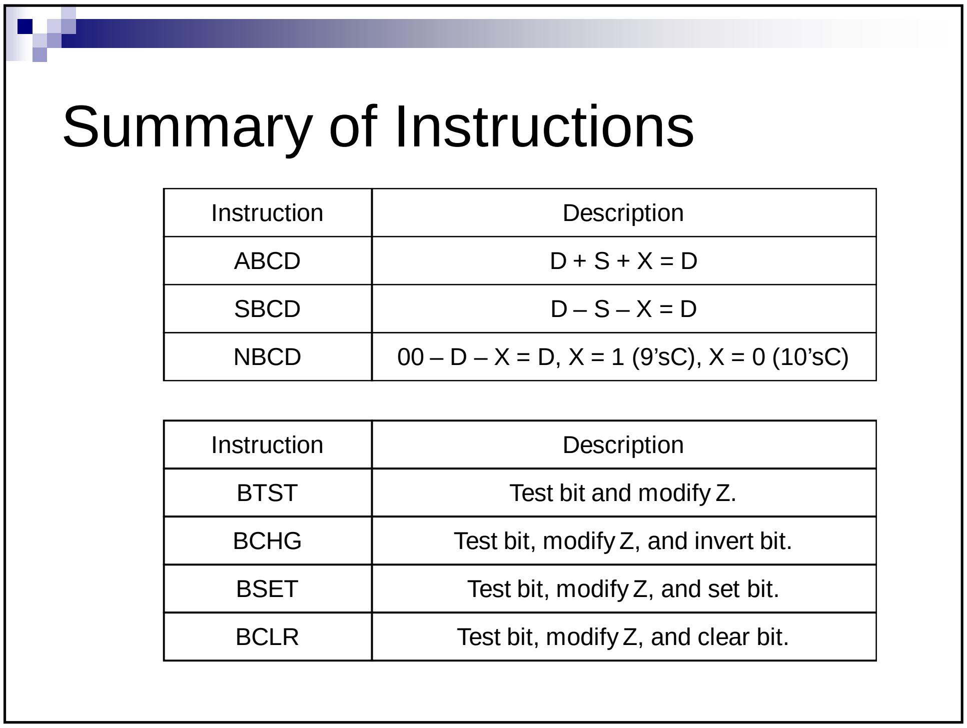
Summary of Instructions

Instruction	Description
AND	Performs AND operation
ANDI	Performs AND operation using immediate data
OR	Performs OR operation
ORI	Performs OR operation using immediate data
EOR	Performs XOR operation
EORI	Performs XOR operation on immediate data
NOT	Performs NOT operation



Summary of Instructions

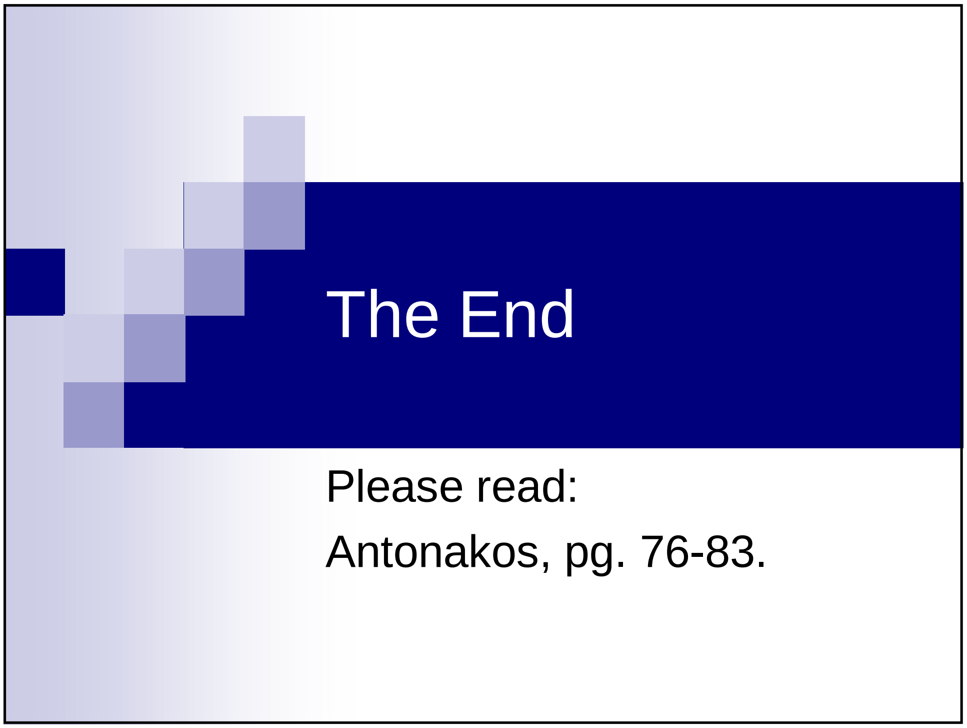
Instruction	Description
LSL	Logical shift left, adds zeros from right.
LSR	Logical shift right, adds zeros from left.
ASL	Arithmetic shift left, adds zeros from right.
ASR	Arithmetic shift right, adds MSB from left.
ROL	Push to left, then put at back, effects C.
ROR	Push to right, then put at front, effects C.
ROXL	Push to left, then put at back, effects C & X.
ROXR	Push to right, then put at front, effects C & X.



Summary of Instructions

Instruction	Description
ABCD	D + S + X = D
SBCD	D – S – X = D
NBCD	00 – D – X = D, X = 1 (9’sC), X = 0 (10’sC)

Instruction	Description
BTST	Test bit and modify Z.
BCHG	Test bit, modify Z, and invert bit.
BSET	Test bit, modify Z, and set bit.
BCLR	Test bit, modify Z, and clear bit.



The End

Please read:

Antonakos, pg. 76-83.