Logic Bitwise Instructions

And MSP430 logic bitwise instructions

Definitions (1)

- **Bitwise operation**: The operation takes place bit by bit, independently of other bits in the word
  - Notation: dest ← dest .**op** src is to be interpreted as
    dest(0)←dest(0) .**op** src(0),
    dest(1)←dest(1) .**op** src(1), . . .
    dest(n-1)←dest(n-1) .**op** src(n-1)

  Where dest(j) and src(j) denote bits of destination and source.

- Bitwise operations allow us to manipulates bits inpendently and without affecting other bits.
- Typical bit operations: **set** (make bit=1), **clear** (make bit=0) and **toggle** or **invert** (invert value of bit), **test** (to determine if a bit is 1, or if a group of bits are not all 0)

Definitions (2)

- Mask: An operand, usually the source, used in bitwise operations in which only the bits to be affected in destination are 1
  - Example: the mask 00010100B=84h=132=24Q means that bits dest(2) and dest(4) are targeted, all others left unchanged.
- Set, clear, test, and toggle of bits are done using the operations **or**, **and** and **xor**

Properties of logic operations:

1. To clear:

$$\bar{X} \cdot A = \begin{cases} 0 & X = 1 \quad \text{Forces a clear or reset} \\ A & X = 0 \quad \text{Leaves A unchanged} \end{cases}$$

2. To set:

$$X + A = \begin{cases} A & X = 0 \quad \text{Leaves A unchanged} \\ 1 & X = 1 \quad \text{Forces a set} \end{cases}$$

3. And , to test:

$$X \cdot A = \begin{cases} 0 & X = 0 \quad \text{Forces a clear or reset} \\ A & X = 1 \quad \text{Leaves A unchanged} \end{cases}$$

4. To toggle or invert

$$X \oplus A = \begin{cases} A & X = 0 \quad \text{Leaves A unchanged} \\ \bar{A} & X = 1 \quad \text{Toggles or inverts A} \end{cases}$$

Using masks to work with bits: Example

Clearing bits with AND: dest<-- dest OR (NOT src)

Mask	1	0	0	1
dest	x	x	x	x
dest	0	x	x	0

Bits 1 and 2 remain unchanged

Setting bits with OR: dest<-- dest OR src

Mask	1	0	0	1
dest	x	x	x	x
dest	1	x	x	1

Bits 1 and 2 remain unchanged

Testing bits (also AND): dest <-- dest AND src

Mask	1	0	0	1
dest	x	x	x	x
dest	x	0	0	x

Flag Z=1 if and only if (Bit0 =0 and Bit 3=0)
When only one bit is tested, Z=0 if Bit = 1 and Z=1 if Bit=0

Toggling bits with XOR: dest <-- dest XOR src

Mask	1	0	0	1
dest	x	x	x	x
dest	x'	x	x	x'

Bits 1 and 2 remain unchanged, while bits 0 and 3 are inverted.

MSP430 Logic Functions

Effects on Flags

- Flags are affected by logic operations as follows, under otherwise indicated:
- Z=1 if destination is cleared
- Z=0 if at least one bit in destination is set
- C= Z’ (Flag C is always equal to the inverted value of Z in these operations)
- N = most significant bit of destination
- V = 0

Core bitwise Instructions (1)

1. **and** ***src, dest*** realizes *dest* ← *src* .and. *dest*
2. **xor** ***src, dest*** realizes *dest* ← *src* .xor. *dest*
3. **bit** ***src, dest*** realizes *src* .and. *dest* but only affects flags

- *Example*

  R12= 35ABh = 0011 0101 1010 1011

  R15= AB96h = 1010 1011 1001 0110

  and R12,R15 0010 0001 1000 0010→R15=2182h

  Flags: C=1, Z=0, N=0, V=0

  xor R12,R15 1001 1110 0011 1101→R15=9E3Dh

  Flags: C=1, Z=0, N=1, V=0

  and.b R12,R15 1000 0010→R15=0082h

  Flags: C=1, Z=0, N=1, V=0

  xor.b R12,R15 0011 1101→R15=003Dh

  Flags: C=1, Z=0, N=0, V=0

Core bitwise Instructions (2)
– these do not affect flags -

4. **bis *src, dest*** realizes *dest* ← *src* .or. *dest,* : SETS BITS
5. **bic *src, dest*** realizes *dest* ← *src'* .and. *Dest : CLEARS BITS*

- *Examples*

  R12= 35ABh = 0011 0101 1010 1011

  R15= AB96h = 1010 1011 1001 0110

  bis R12,R15 1011 1111 1011 1111→R15= BFBFh

  Flags: unchanged

  bic R12,R15 1000 1010 0001 0100→R15= 8A14h

  Flags: unchanged

  bis.b R12,R15 1011 1111→R15= 00BFh

  Flags: unchanged

  bic.b R12,R15 0001 0100→R15=0014h

  Flags: unchanged

Emulated Logic Instructions

•Manipulating flags:

clrc	$C \leftarrow 0$
clrn	$N \leftarrow 0$
clrz	$Z \leftarrow 0$
setc	$C \leftarrow 1$
setn	$N \leftarrow 1$
setz	$Z \leftarrow 1$

- Inverting a destination

inv dest = xor #FF, dest

Toggles all bits

Remarks on bit instruction in MSP430

- Since C=Z', either the carry flag C or the zero flag C can be used as information about condition.
- In **bit src, dest** C=1 (Z=0) means that at least one bit among those tested is not 0.
- In **bit #BITn, dest**, where BITn is the word where all but the n-th bit are 0, C=tested bit
  - Example R15= 0110 1100 1101 1001 then

    bit #BIT14,R15 yields C=1 = bit 14;

    bit #BIT5,R15 yields C = 0 = bit 5.

Register Control instructions.

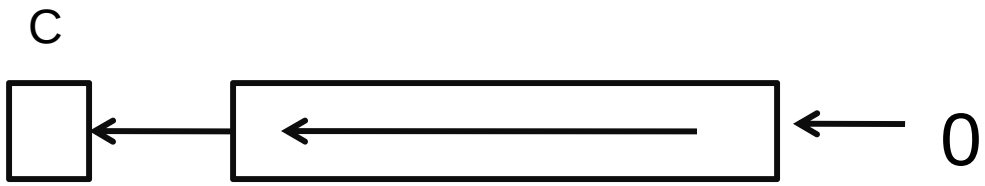
Rolling (or shifting) and rotating

GENERAL DESCRIPTIONS:

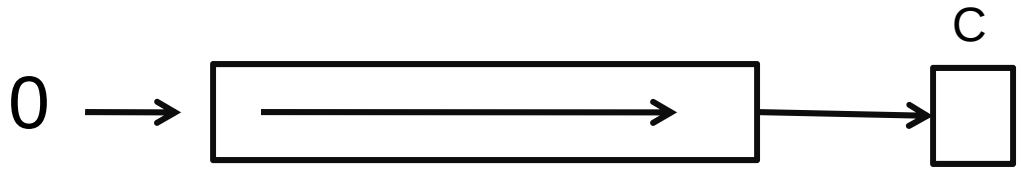
- Shift or roll instructions move bits to the left or right within a register or memory location
- Left shifts can be associated with multiplication by 2
- Right shifts may be arithmetical or logical
  - Arithmetical right shifts divide signed numbers by 2, keeping the sign bit for the result.
- May also be used to capture lsb or msb, while displacing other bits.

Shifts or rolls

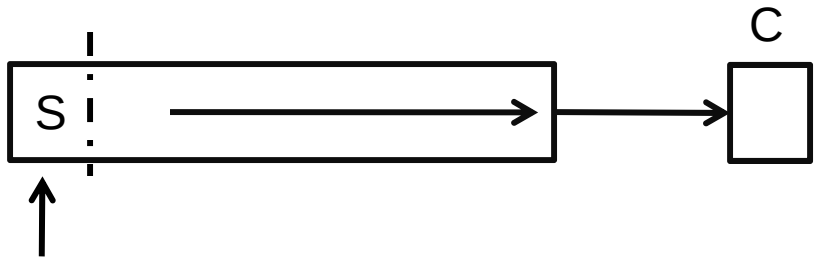
Left shift or roll:
(multiply by 2)



Right logical shift or roll:
(divide by 2 unsigned numbers)



Right arithmetic shift or roll:
(divide by 2 signed numbers)



Sign bit remains

Shifts or Rolls (Example)

	C	b3	b2	b1	b0	- C -	
	x	1	0	1	1		
SHIFT LEFT	1	0	1	1	0		(A)
Shift Right (Log)		0	1	0	1	1	(B)
Shift Rigt Arith:		1	1	0	1	1	(C)

NOTE:
Carry (C) at left for left rolls

Right C used for right rolls

Arithmetic Look at shifts: Original number = 11 (unsigned), or -5 (signed)

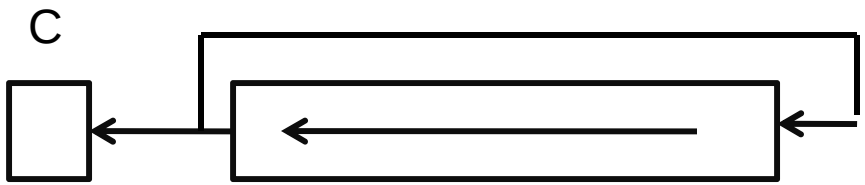
(A) Including carry: Unsigned 22 or signed - 10

(B) Only for unsigned: 11/2 yields 5 (in register) and residue 1 --in C

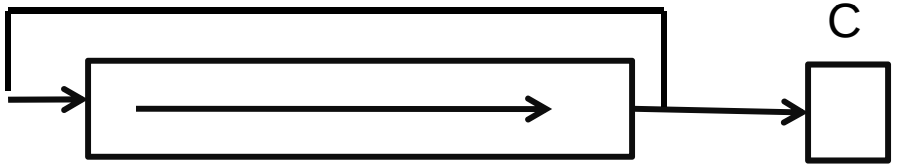
(C) Only for signed: -5/2 yields -3 (in register) and residue 1 (in C)
-- Note: (-5) = (-3)*2 + 1, which is the division algorithm.

Rotate(s)

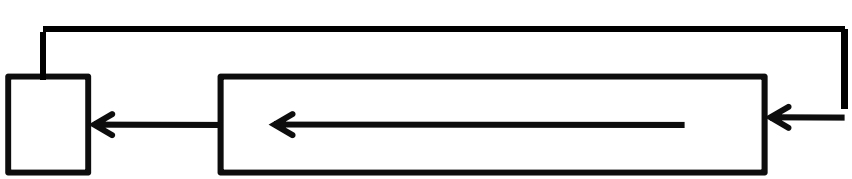
Rotate Left:



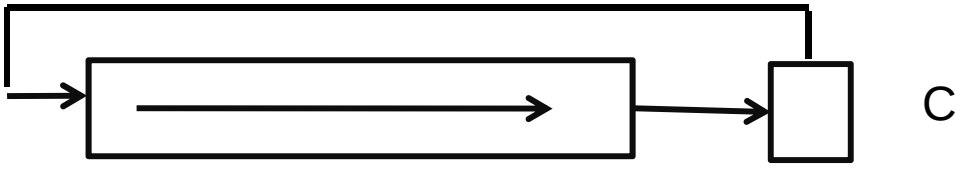
Rotate Right:



Rotate Left
Through Carry:



Rotate Right
Through Carry:



Rotates (example)

	C	b3	b2	b1	b0	- C -	
	zz	1	0	1	1		
Rotate Left	1	0	1	1	1		(A)
R.L. through C	1	0	1	1	zz		(B)
Rotate Right		**1**	1	0	1	1	(C)
Rot. Right thru C		**zz**	1	0	1	1	(D)

NOTE:
Carry (C) at left for left rolls

Right C used for right rolls

Remarks:

1. In (B) and (D), zz is the original unknown value in C

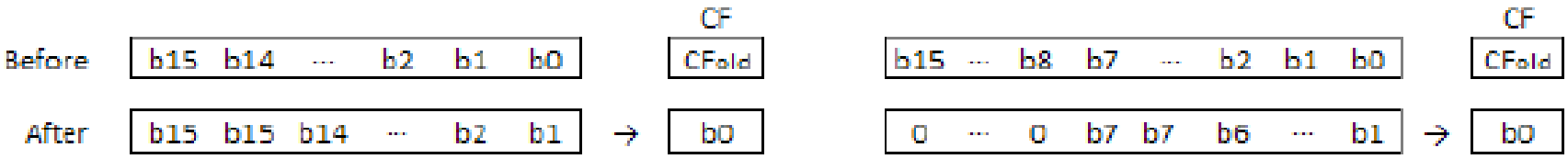
2. In (C), b3=1 because originally b0=1. This value is what was rotated.

Rolls and Rotates in MSP430

Rolling (Shifting) and Rotating Data bits

- Not all cases included
- Two core instructions:
  - Right rolling arithmetic: **rra** ***dest***
  - Right rotation through Carry: **rrc** ***dest***
- Two emulated instructions
  - Left rolling arithmetic: **rla** ***dest*** = **add** ***dest,dest***
  - Rotate left through carry: **rla** ***dest*** = **addc** ***dest,dest***
- Roll = Shift

Right arithmetic shift or roll rra:



Word rra.w dest = rra dest

Byte rra.b dest

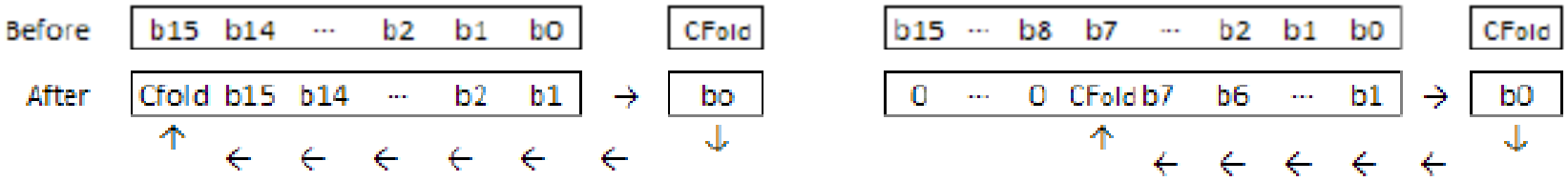
Arithmetic Interpretation: Divide by two (dividend = Q*d + r, |r|<d

R7 = FB0Fh = 1**111101100001111** = (-1265)

rra R7 → 1**111110110000111** | 1 | C=1 Z=0 N=1 V=0

New R7 = FD87h = > -633 -1265 = (-633)*2 +1

Right Rotation Through Carry rrc



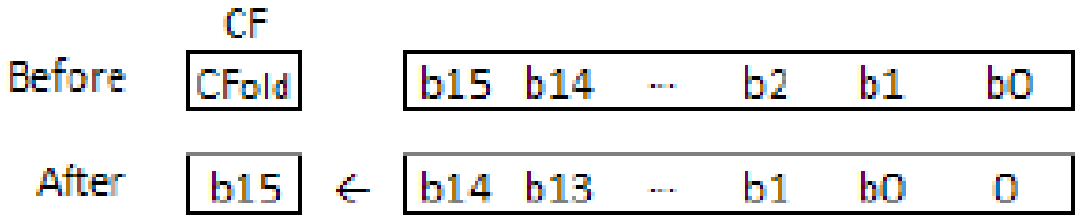
Some Uses:

- To divide by 2 unsigned numbers by first clearing C
- To extract bits from lsb to msb
- Think of other possibilities

Left rollings

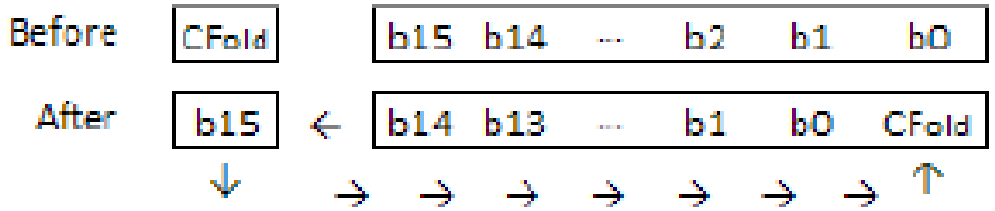
Left rolling: add dest,dest ➔ multiply by 2 (may need carry)

Other uses: extract bits from msb to lsb



Examples: -----

Left Rotation Through Carry



- Arithmetic Interpretation: 2x + C
- Think of other uses