

Arithmetic Microoperations in the CPU

Arithmetic Microoperations

- a. Addition: $AC \leftarrow DR + AC$
- b. Subtraction: $AC \leftarrow DR + AC' + 1$ (two's complement)
- c. Increment: $AC \leftarrow AC + 1$
- d. Decrement: $AC \leftarrow AC - 1$
- e. Shift

- i. Arithmetic shift left: $AC \leftarrow \text{ashl } AC$

pre:

R_{n-1}	R_{n-2}	...	R_1	R_0
-----------	-----------	-----	-------	-------

post:

R_{n-2}	R_{n-3}	...	R_0	0
-----------	-----------	-----	-------	---

Examples: pre: 0000 1000, (8)
 post: 0001 0000, (16)

 pre: 1111 1000, (-8)
 post: 1111 0000, (-16)

Note: The sign bit must remain unchanged or an overflow has occurred. An overflow occurs after an arithmetic shift left if, before the shift, the bits R_{n-2} and R_{n-1} are not the same.

Overflow Example: pre: 0100 1000, (72)
 post: 1001 0000, (-112)

- ii. Arithmetic shift right: $AC \leftarrow \text{ashr } AC$

pre:

R_{n-1}	R_{n-2}	...	R_1	R_0
-----------	-----------	-----	-------	-------

post:

R_{n-1}	R_{n-1}	...	R_2	R_1
-----------	-----------	-----	-------	-------

Examples: pre: 0100 1000, (72)
 post: 0010 0100, (36)

 pre: 1100 1000, (-56)
 post: 1110 0100, (-28)

Note: The sign bit is shifted right and restored. The arithmetic shift-right leaves the sign bit unchanged and shifts the number (including the sign bit) to the right. In this way, R_{n-1} remains the unchanged, R_{n-2} receives the bit from R_{n-1} and so forth. The bit R_0 is lost.