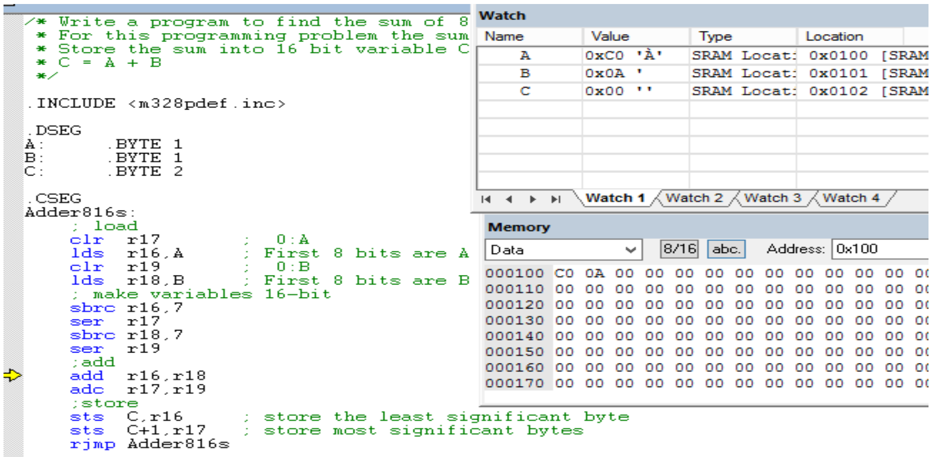
Adder816s: Simulation



```
/* Write a program to find the sum of 8
 * For this programming problem the sum
 * Store the sum into 16 bit variable C
 * C = A + B
 */

.INCLUDE <m328pdef.inc>

.DSEG
A:      .BYTE 1
B:      .BYTE 1
C:      .BYTE 2

.CSEG
Adder816s:
      ; load
      clr  r17        ;  0:A
      lds  r16,A      ; First 8 bits are A
      clr  r19        ;  0:B
      lds  r18,B      ; First 8 bits are B
      ; make variables 16-bit
      sbrc r16,7
      ser  r17
      sbrc r18,7
      ser  r19
      ;add
      add  r16,r18
      adc  r17,r19
      ;store
      sts  C,r16      ; store the least significant byte
      sts  C+1,r17    ; store most significant bytes
      rjmp Adder816s
```

Watch

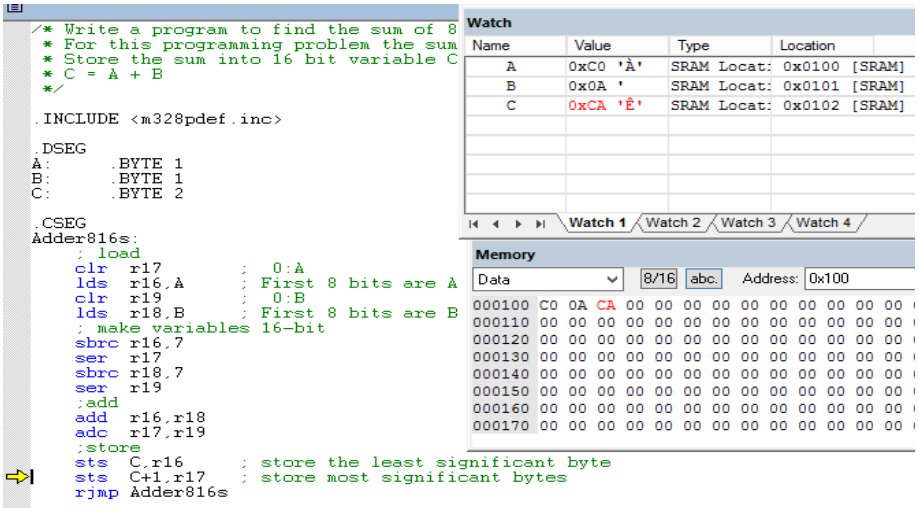
Name	Value	Type	Location
A	0xC0 'À'	SRAM Locat:	0x0100 [SRAM
B	0x0A '	SRAM Locat:	0x0101 [SRAM
C	0x00 ''	SRAM Locat:	0x0102 [SRAM

Memory

Data 8/16 abc. Address: 0x100

```
000100 C0 0A 00 00 00 00 00 00 00 00 00 00 0
000110 00 00 00 00 00 00 00 00 00 00 00 00 0
000120 00 00 00 00 00 00 00 00 00 00 00 00 0
000130 00 00 00 00 00 00 00 00 00 00 00 00 0
000140 00 00 00 00 00 00 00 00 00 00 00 00 0
000150 00 00 00 00 00 00 00 00 00 00 00 00 0
000160 00 00 00 00 00 00 00 00 00 00 00 00 0
000170 00 00 00 00 00 00 00 00 00 00 00 00 0
```

Figure 1.Variable A is initialized to 0xC0 (or -64) and variable B initialized to 0x0A (or 10). These 8-bit variables are converted to 16-bit variables by extending the sign bit. As seen in the code above, initially the high byte is cleared. If the low byte is positive then no additional action is taken. If on the other hand the low byte is negative (bit 7 = 1) then the 8 most significant bits to 1. The numbers are added, and the carry from the least significant bits is added to the most significant bits.



```
/* Write a program to find the sum of 8
 * For this programming problem the sum
 * Store the sum into 16 bit variable C
 * C = A + B
 */

.INCLUDE <m328pdef.inc>

.DSEG
A:      .BYTE 1
B:      .BYTE 1
C:      .BYTE 2

.CSEG
Adder816s:
      ; load
      clr  r17        ;  0:A
      lds  r16,A      ; First 8 bits are A
      clr  r19        ;  0:B
      lds  r18,B      ; First 8 bits are B
      ; make variables 16-bit
      sbrc r16,7
      ser  r17
      sbrc r18,7
      ser  r19
      ;add
      add  r16,r18
      adc  r17,r19
      ;store
      sts  C,r16      ; store the least significant byte
      sts  C+1,r17    ; store most significant bytes
      rjmp Adder816s
```

Watch

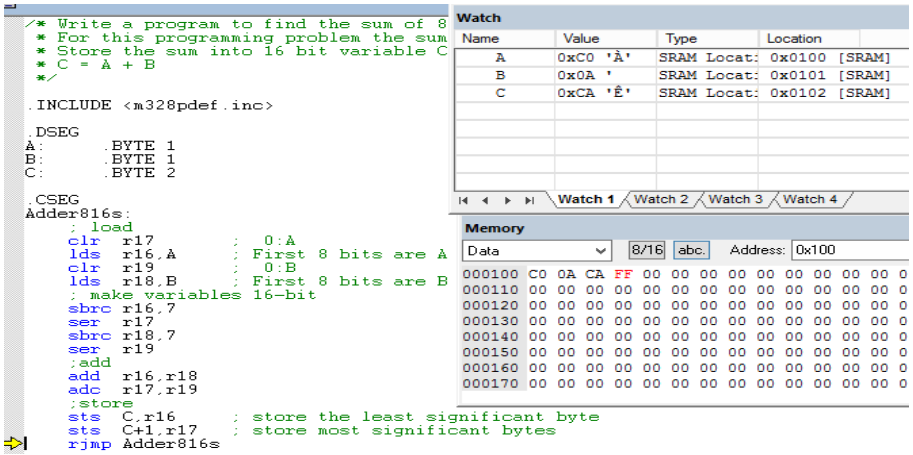
Name	Value	Type	Location
A	0xC0 'À'	SRAM Locat:	0x0100 [SRAM]
B	0x0A '	SRAM Locat:	0x0101 [SRAM]
C	0xCA 'Ê'	SRAM Locat:	0x0102 [SRAM]

Memory

Data 8/16 abc. Address: 0x100

```
000100 C0 0A CA 00 00 00 00 00 00 00 00 00 00
000110 00 00 00 00 00 00 00 00 00 00 00 00 00
000120 00 00 00 00 00 00 00 00 00 00 00 00 00
000130 00 00 00 00 00 00 00 00 00 00 00 00 00
000140 00 00 00 00 00 00 00 00 00 00 00 00 00
000150 00 00 00 00 00 00 00 00 00 00 00 00 00
000160 00 00 00 00 00 00 00 00 00 00 00 00 00
000170 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 2. Since C = A + B, the least significant byte of variable C is 0xCA, because the answer is 0xFFCA (or -54).



```
/* Write a program to find the sum of 8
 * For this programming problem the sum
 * Store the sum into 16 bit variable C
 * C = A + B
 */

.INCLUDE <m328pdef.inc>

.DSEG
A:      .BYTE 1
B:      .BYTE 1
C:      .BYTE 2

.CSEG
Adder816s:
      ; load
      clr  r17        ;  0:A
      lds  r16,A      ; First 8 bits are A
      clr  r19        ;  0:B
      lds  r18,B      ; First 8 bits are B
      ; make variables 16-bit
      sbrc r16,7
      ser  r17
      sbrc r18,7
      ser  r19
      ;add
      add  r16,r18
      adc  r17,r19
      ;store
      sts  C,r16      ; store the least significant byte
      sts  C+1,r17    ; store most significant bytes
      rjmp Adder816s
```

Watch

Name	Value	Type	Location
A	0xC0 'À'	SRAM Locat:	0x0100 [SRAM]
B	0x0A '	SRAM Locat:	0x0101 [SRAM]
C	0xCA 'Ê'	SRAM Locat:	0x0102 [SRAM]

Memory

Data 8/16 abc. Address: 0x100

```
000100 C0 0A CA FF 00 00 00 00 00 00 00 00 00 0
000110 00 00 00 00 00 00 00 00 00 00 00 00 00 0
000120 00 00 00 00 00 00 00 00 00 00 00 00 00 0
000130 00 00 00 00 00 00 00 00 00 00 00 00 00 0
000140 00 00 00 00 00 00 00 00 00 00 00 00 00 0
000150 00 00 00 00 00 00 00 00 00 00 00 00 00 0
000160 00 00 00 00 00 00 00 00 00 00 00 00 00 0
000170 00 00 00 00 00 00 00 00 00 00 00 00 00 0
```

Figure 3. The answer is negative, so the most significant byte of C is stored with all ones. The byte ordering is little endian.