

Factorial: C = A!

Calculate the factorial of the number held in variable A. The number in variable A must be greater than 0 and less than or equal to 6! = 720 (note: 5! = 120). Store factorial of A into 16 bit variable C.

Byte ordering is little endian. C = A!

C1:C0 = A!

Simulation of the factorial problem 3! = 6

```

* C = A!
*/
INCLUDE <m328pdef.inc>

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

.CSEG
Fact1_6:
  lds r16, A
  clr r0
  inc r0          ; r0 = 1
fact1_6a:
  mul r0, r16
  dec r16
  brne fact1_6a
  sts C, r0      ; least significant byte (little end)
  sts C+1, r1   ; most significant byte (big end)

rjmp Fact1_6
    
```

Name	Value	Type	Location
A	3 'L'	SRAM Location	0x0100 [SRAM]
C	0 ''	SRAM Location	0x0101 [SRAM]
r0	0 ''	Register	R0
r1	0 ''	Register	R1

Data	8/16	abc.	Address: 0x100	Cols: Auto
000100	03	00 00 00 00 00 00 00 00 00 00 00 00	
00010C	00	00 00 00 00 00 00 00 00 00 00 00 00	
000118	00	00 00 00 00 00 00 00 00 00 00 00 00	
000124	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	
00013C	00	00 00 00 00 00 00 00 00 00 00 00 00	
000148	00	00 00 00 00 00 00 00 00 00 00 00 00	
000154	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	

Figure 1: First initialize A to 0x0003 (3 decimal). Set variable A to “3” in the Watch window.

```

* C = A!
*/
INCLUDE <m328pdef.inc>

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

.CSEG
Fact1_6:
  lds r16, A
  clr r0
  inc r0          ; r0 = 1
fact1_6a:
  mul r0, r16
  dec r16
  brne fact1_6a
  sts C, r0      ; least significant byte (little end)
  sts C+1, r1   ; most significant byte (big end)

rjmp Fact1_6
    
```

Name	Value	Type	Location
A	3 'L'	SRAM Location	0x0100 [SRAM]
C	0 ''	SRAM Location	0x0101 [SRAM]
r0	3 'L'	Register	R0
r1	0 ''	Register	R1

Data	8/16	abc.	Address: 0x100	Cols: Auto
000100	03	00 00 00 00 00 00 00 00 00 00 00 00	
00010C	00	00 00 00 00 00 00 00 00 00 00 00 00	
000118	00	00 00 00 00 00 00 00 00 00 00 00 00	
000124	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	
00013C	00	00 00 00 00 00 00 00 00 00 00 00 00	
000148	00	00 00 00 00 00 00 00 00 00 00 00 00	
000154	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	

Figure 2: Loop 1 performs the first part of the calculation of A! by doing 3x1 with command “mul r0, r16”. The product 0x0003 is stored in the r1:r0 register pair.

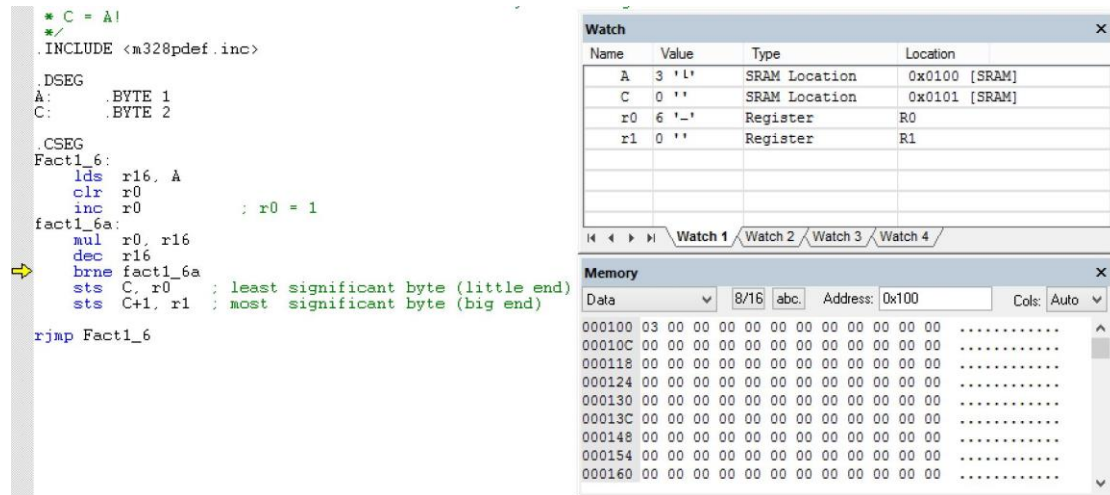


Figure 3: Loop 2 performs the second part of calculation of A! by multiplying 2x3 with the product 0x0006 now in the r1:r0 register pair.

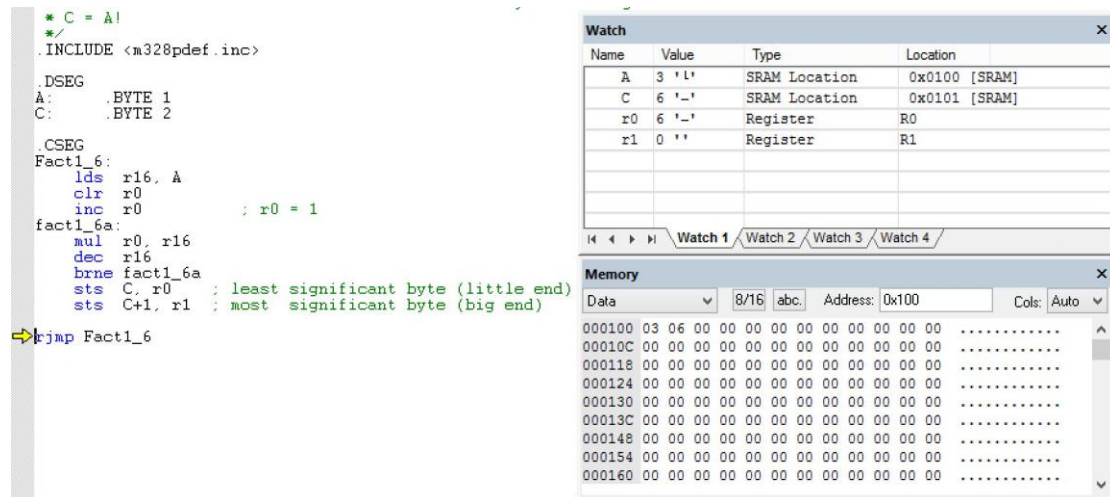


Figure 4: In the 3rd and final loop the result 0x0006 in register pair r1:r0 is saved in 2 byte variable C. Byte ordering is little endian.

Simulation of the factorial problem $5! = 120$

```

* C = A!
*/
INCLUDE <m328pdef.inc>
DSEG
A: .BYTE 1
C: .BYTE 2
CSEG
Fact1_6:
  lds r16, A
  clr r0
  inc r0          ; r0 = 1
fact1_6a:
  mul r0, r16
  dec r16
  brne fact1_6a
  sts C, r0      ; least significant byte (little end)
  sts C+1, r1   ; most significant byte (big end)
rjmp Fact1_6
    
```

Name	Value	Type	Location
A	5	SRAM Location	0x0100 [SRAM]
C	0	SRAM Location	0x0101 [SRAM]
r0	0	Register	R0
r1	0	Register	R1

Data	8/16	abc	Address: 0x100	Cols: Auto						
000100	005	000	000	000	000	000	000	000	000
000108	000	000	000	000	000	000	000	000	000
000110	000	000	000	000	000	000	000	000	000
000118	000	000	000	000	000	000	000	000	000
000120	000	000	000	000	000	000	000	000	000
000128	000	000	000	000	000	000	000	000	000
000130	000	000	000	000	000	000	000	000	000
000138	000	000	000	000	000	000	000	000	000
000140	000	000	000	000	000	000	000	000	000

Figure 1: First initialize A to 0x0005 (5 decimal). Set variable A to “5” in the Watch window.

```

* C = A!
*/
INCLUDE <m328pdef.inc>
DSEG
A: .BYTE 1
C: .BYTE 2
CSEG
Fact1_6:
  lds r16, A
  clr r0
  inc r0          ; r0 = 1
fact1_6a:
  mul r0, r16
  dec r16
  brne fact1_6a
  sts C, r0      ; least significant byte (little end)
  sts C+1, r1   ; most significant byte (big end)
rjmp Fact1_6
    
```

Name	Value	Type	Location
A	5	SRAM Location	0x0100 [SRAM]
C	0	SRAM Location	0x0101 [SRAM]
r0	5	Register	R0
r1	0	Register	R1

Data	8/16	abc	Address: 0x100	Cols: Auto						
000100	005	000	000	000	000	000	000	000	000
000108	000	000	000	000	000	000	000	000	000
000110	000	000	000	000	000	000	000	000	000
000118	000	000	000	000	000	000	000	000	000
000120	000	000	000	000	000	000	000	000	000
000128	000	000	000	000	000	000	000	000	000
000130	000	000	000	000	000	000	000	000	000
000138	000	000	000	000	000	000	000	000	000
000140	000	000	000	000	000	000	000	000	000

Figure 2: Loop 1 performs the first part of the calculation of A! by doing 5x1 with command “mul r0, r16”. The product 0x0005 is stored in the r1:r0 register pair.

```

* C = A!
*/
INCLUDE <m328pdef.inc>
DSEG
A: .BYTE 1
C: .BYTE 2
CSEG
Fact1_6:
  lds r16, A
  clr r0
  inc r0          ; r0 = 1
fact1_6a:
  mul r0, r16
  dec r16
  brne fact1_6a
  sts C, r0      ; least significant byte (little end)
  sts C+1, r1   ; most significant byte (big end)
rjmp Fact1_6
    
```

Name	Value	Type	Location
A	5	SRAM Location	0x0100 [SRAM]
C	0	SRAM Location	0x0101 [SRAM]
r0	20	Register	R0
r1	0	Register	R1

Data	8/16	abc	Address: 0x100	Cols: Auto						
000100	005	000	000	000	000	000	000	000	000
000108	000	000	000	000	000	000	000	000	000
000110	000	000	000	000	000	000	000	000	000
000118	000	000	000	000	000	000	000	000	000
000120	000	000	000	000	000	000	000	000	000
000128	000	000	000	000	000	000	000	000	000
000130	000	000	000	000	000	000	000	000	000
000138	000	000	000	000	000	000	000	000	000
000140	000	000	000	000	000	000	000	000	000

Figure 3: In the loop 2, perform the second part of calculation of A! by doing 4x5 with command “mul

r0, r16". The result 20 which is stored in register pair r1:r0.

```

* C = A!
*/
#include <m328pdef.inc>

DSEG
A:   .BYTE 1
C:   .BYTE 2

CSEG
Fact1_6:
lds  r16, A
clr  r0
inc  r0           ; r0 = 1
fact1_6a:
mul  r0, r16
dec  r16
brne fact1_6a
sts  C, r0        ; least significant byte (little end)
    sts  C+1, r1  ; most significant byte (big end)
rjmp Fact1_6

```

Name	Value	Type	Location
A	5 ' '	SRAM Location	0x0100 [SRAM]
C	120 'x'	SRAM Location	0x0101 [SRAM]
r0	120 'x'	Register	R0
r1	0 ''	Register	R1

Data	8/16	abc	Address: 0x100	Cols: Auto
000100	005	120	000 000 000 000 000 000 000 000	.x.....
000108	000	000	000 000 000 000 000 000 000 000
000110	000	000	000 000 000 000 000 000 000 000
000118	000	000	000 000 000 000 000 000 000 000
000120	000	000	000 000 000 000 000 000 000 000
000128	000	000	000 000 000 000 000 000 000 000
000130	000	000	000 000 000 000 000 000 000 000
000138	000	000	000 000 000 000 000 000 000 000
000140	000	000	000 000 000 000 000 000 000 000

Figure 4: In the 5th and final loop the result 102_{10} in register pair r1:r0 is saved in 2 byte variable C. The most significant byte, equal to 0, will be saved next. Byte ordering is little endian.