

```

; -----
; countspaces
; Version 1.0
; Date: 11/11/2014
; Written By : Nicholas Lombardo
; -----

```

Below shows the preload loop as it indirectly reads from FPM and writes to SRAM buffer.

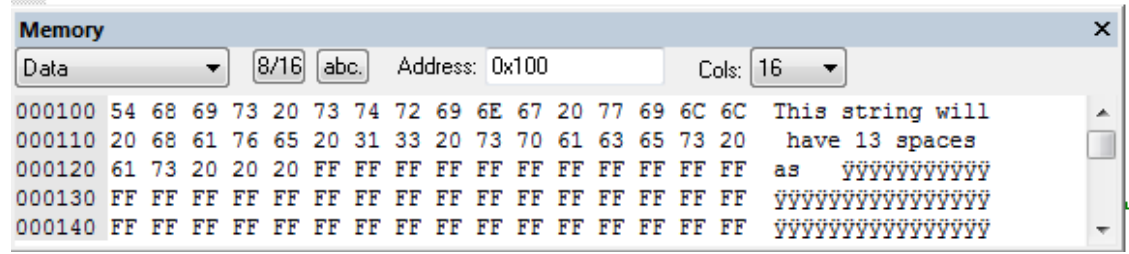
```

.INCLUDE <m328pdef.inc>
.DSEG
buffer: .BYTE 256 // 0 to 255
.CSEG
// Preload SRAM with string (indirect transfer from FPM to SRAM)
ldi ZH,high(mystring<<1) // set Z pointer at string address in Flash
ldi ZL,low(mystring<<1)
ldi YH,high(buffer) // set Y pointer at address 0x0100 in SRAM
ldi YL,low(buffer)

ldi r17,low(mystring<<1)
ldi r16,low(stringend<<1)
sub r16,r17 // number of iterations in r16
mov r17,r16 // copy to r17 for next loop

preload:
lpm r5,Z+ // load character to r5
st Y+,r5 // store r5 in SRAM
dec r16
brne preload // loop until string is loaded (48 bytes/characters)

```

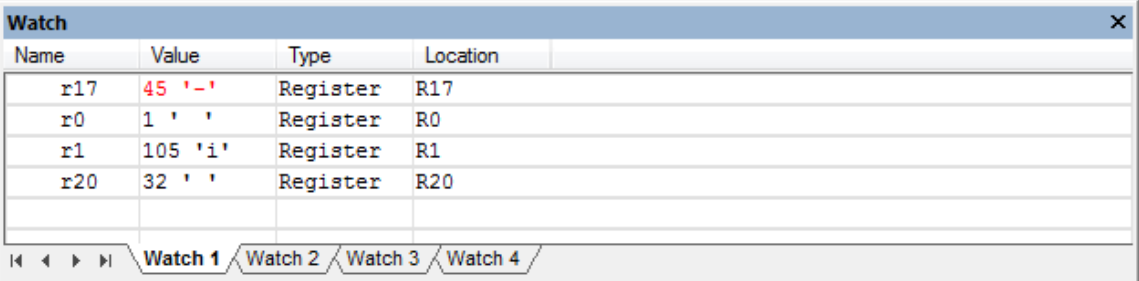


The Memory window tracks the content, displaying the addresses, hexadecimal encoding, and characters (0xFF is undefined).

The second screenshot shows the loop which scans the buffer for the space character (currently looking at the 'i' in "string"). r17 counts remaining characters, r0 holds the count of spaces, r1 is the character being checked, and r20 holds the hexadecimal equivalent to a space (0x20).

```
● rcall    CountSpaces
end:
● rjmp     end

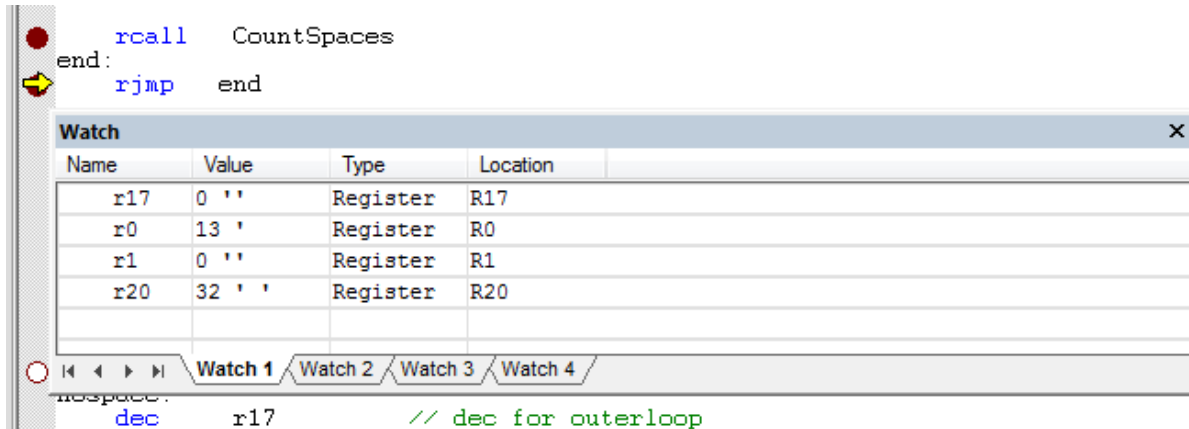
// Count Spaces
CountSpaces:
  clr     r0           // initialize r0 to 0
  ldi     YH,high(buffer) // set Y pointer at address 0x0100 for SRAM
  ldi     YL,low(buffer)
  ldi     r20,0x20    // load 0x20 to r20 for compare (ASCII space)
scan_buffer:
  ld      r1,Y+       // load at Z pointer, increment to next byte address
  cp      r1,r20      // compare loaded byte to 0x20
  brne    nospace     // if not 0x20, skip increment
  inc     r0           // increment r0 if byte in buffer is 0x20
nospace:
  dec     r17         // dec for outerloop
  brne    scan_buffer
  ret
```



The Watch window displays the following data:

Name	Value	Type	Location
r17	45 '-'	Register	R17
r0	1 ' '	Register	R0
r1	105 'i'	Register	R1
r20	32 ' '	Register	R20

The final screenshot will show those four registers once the program is complete. There are no more characters to check, and the final count is 13, just as was indicated by the comments in the code.



The screenshot shows a debugger interface. At the top, assembly code is displayed with a yellow arrow pointing to the instruction `rjmp end`. Below the code is a "Watch" window with a table of register values. The table has four columns: Name, Value, Type, and Location. The rows show the following data:

Name	Value	Type	Location
r17	0 ''	Register	R17
r0	13 '	Register	R0
r1	0 ''	Register	R1
r20	32 ' '	Register	R20

Below the Watch window, there are navigation buttons and tabs labeled "Watch 1", "Watch 2", "Watch 3", and "Watch 4". At the bottom, a portion of assembly code is visible, including the instruction `dec r17 // dec for outerloop`.