1DT301, Computer Technology

Tuesday, October 8, 2019

- Addressing modes
- Program examples

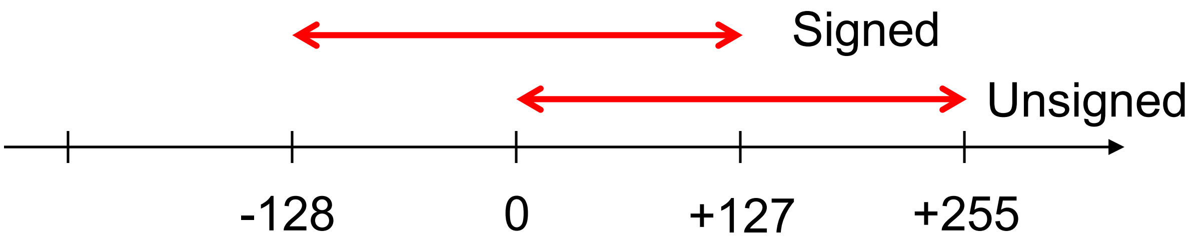
Repetition:

Lowest and highest number in a 8-bit register, Unsigned:

$0\ ;\ 2^8 - 1 = 256 - 1 = 255$

Lowest and highest number in a 8-bit register, Signed:

$-2^7 = -128\ ;\ 2^7 - 1 = 128 - 1 = 127$



Example 1:

Addition of two 16 bit numbers, thus 2 registerpairs.

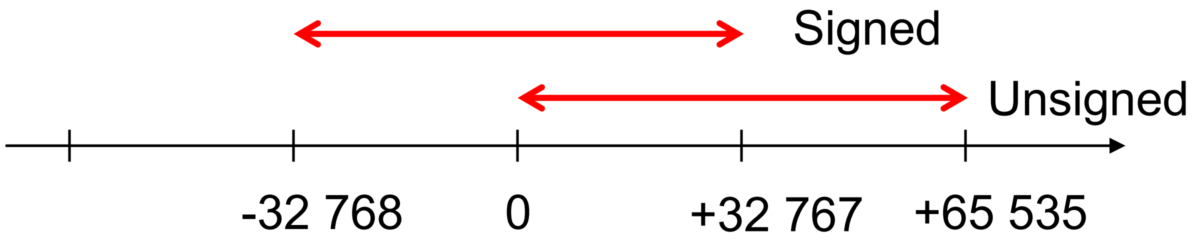
Lowest and highest number in register pair, 16 bit, Unsigned:

$0 \; ; \; 2^{16} - 1 = 65\,536 - 1 = 65\,535$

$$\boxed{(2^{16} = 2^{(10+6)} = 2^{10} \cdot 2^{6} = 1024 \cdot 64 = 64k \approx 64 \cdot 10^{3})}$$

Lowest and highest number in register pair, 16 bit, Signed:

$-2^{15} = -32\,768 \; ; \; 2^{15} - 1 = 32\,768 - 1 = 32\,767$ (Signed)



Example: Add the two numbers 1230 and 2476.
1230 + 2476 = 3706

1230 to binary:

Division:	Quote:	Rest:		
1230/2	615	0	lsb	= least significant bit
615/2	307	1	4 bits	
307/2	153	1		
153/2	76	1		
76/2	38	0		
38/2	19	0	4 bits	
19/2	9	1		
9/2	4	1		
4/2	2	0		
2/2	1	0		
1/2	0	1	msb	= most significant bit

The binary number is: 100 1100 1110 = 0x04CE

2476 to binary:

Division:	Quote:	Rest:	
2476/2	1238	0	lsb = least significant bit
1238/2	619	0	4 bits
619/2	309	1	
309/2	154	1	
154/2	77	0	
77/2	38	1	4 bits
38/2	19	0	
19/2	9	1	
9/2	4	1	
4/2	2	0	
2/2	1	0	
1/2	0	1	msb = most significant bit

The binary number is: 1001 1010 1100 = 0x09AC

Example 1:

Example: Add the two numbers 1230 and 2476.

1230 + 2476 = 3706

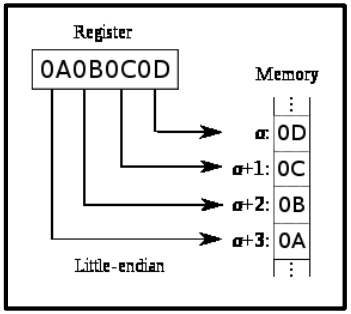
r25

r25:r24 = 1230 = 0000 0100 1100 1110 r24

r23:r22 = 2476 = 0000 1001 1010 1100 r22

r23

add r22, r24



Cy = 1

				1	1			
r24	1	1	0	0	1	1	1	0
r22	1	0	1	0	1	1	0	0
	0	1	1	1	1	0	1	0

Example 1:

Example: Add the two numbers 1230 and 2476.

1230 + 2476 = 3706

r25

r25:r24 = 1230 = 0000 0100 1100 1110

r23:r22 = 2476 = 0000 1001 1010 1100

r23

adc r23, r25   add with carry

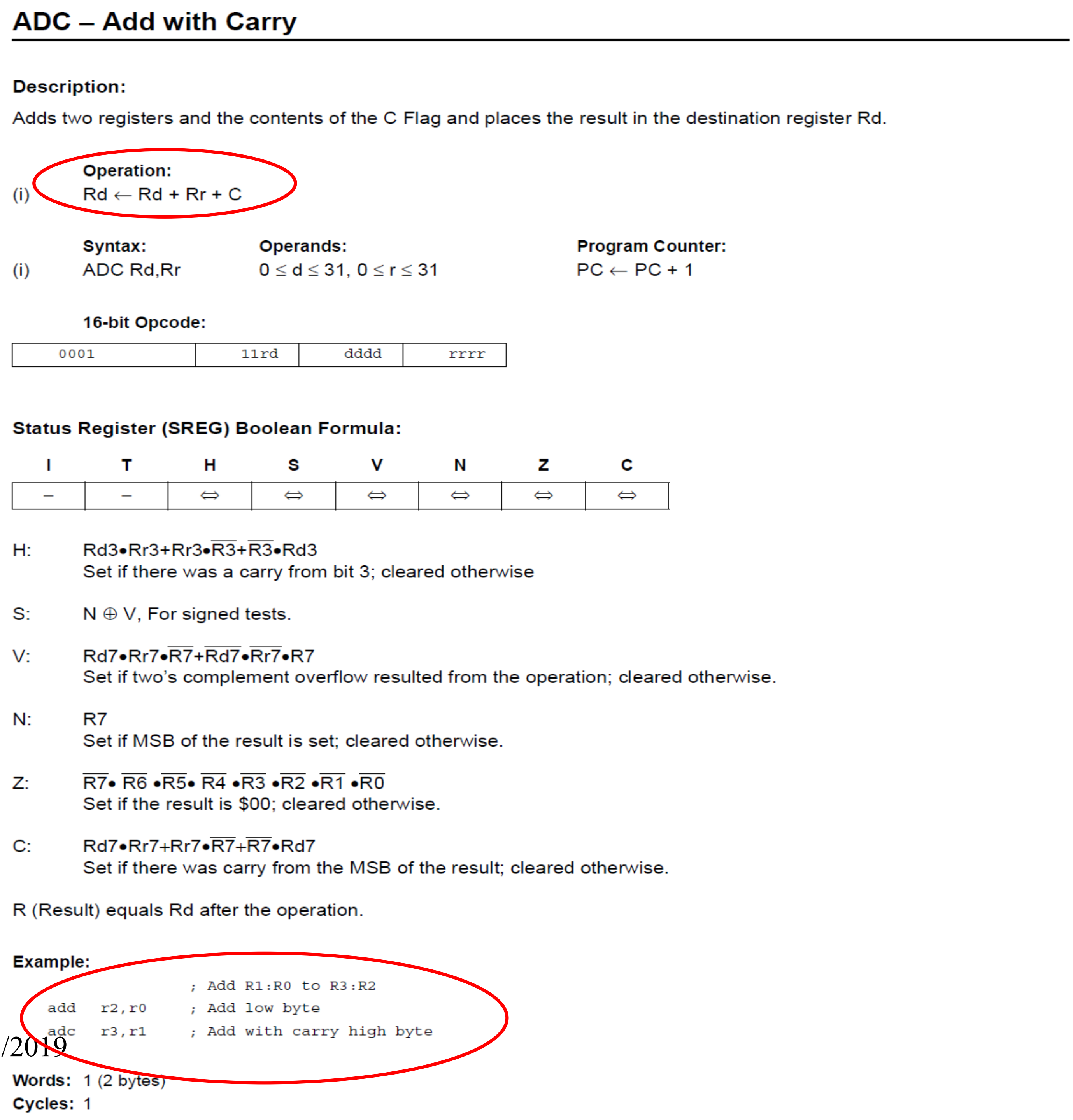
1 Cy=1

r25	0	0	0	0	0	1	0	0
r23	0	0	0	0	1	0	0	1
	0	0	0	0	1	1	1	0

r23   r22

Result:

r23:r22 = 0000 1110 0111 1010 = 0x0E7A =3706



ADC – Add with Carry

Description:

Adds two registers and the contents of the C Flag and places the result in the destination register Rd.

Operation:

(i) Rd ← Rd + Rr + C

	Syntax:	**Operands:**	**Program Counter:**
(i)	ADC Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	PC ← PC + 1

16-bit Opcode:

0001	11rd	dddd	rrrr

Status Register (SREG) Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H: $Rd3 \bullet Rr3 + Rr3 \bullet \overline{R3} + \overline{R3} \bullet Rd3$
Set if there was a carry from bit 3; cleared otherwise

S: $N \oplus V$, For signed tests.

V: $Rd7 \bullet Rr7 \bullet \overline{R7} + \overline{Rd7} \bullet \overline{Rr7} \bullet R7$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R7
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is $00; cleared otherwise.

C: $Rd7 \bullet Rr7 + Rr7 \bullet \overline{R7} + \overline{R7} \bullet Rd7$
Set if there was carry from the MSB of the result; cleared otherwise.

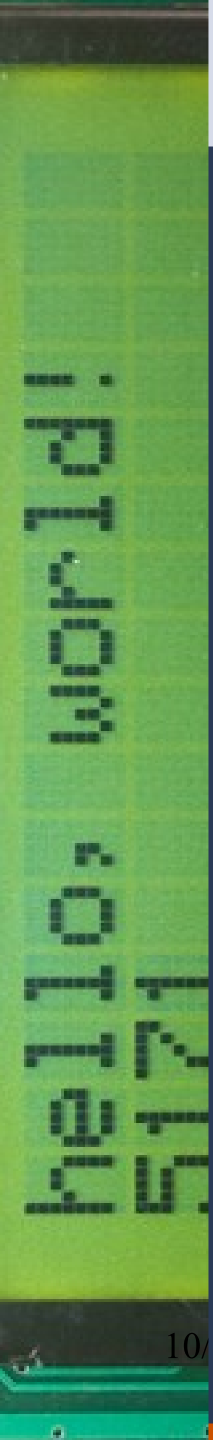
R (Result) equals Rd after the operation.

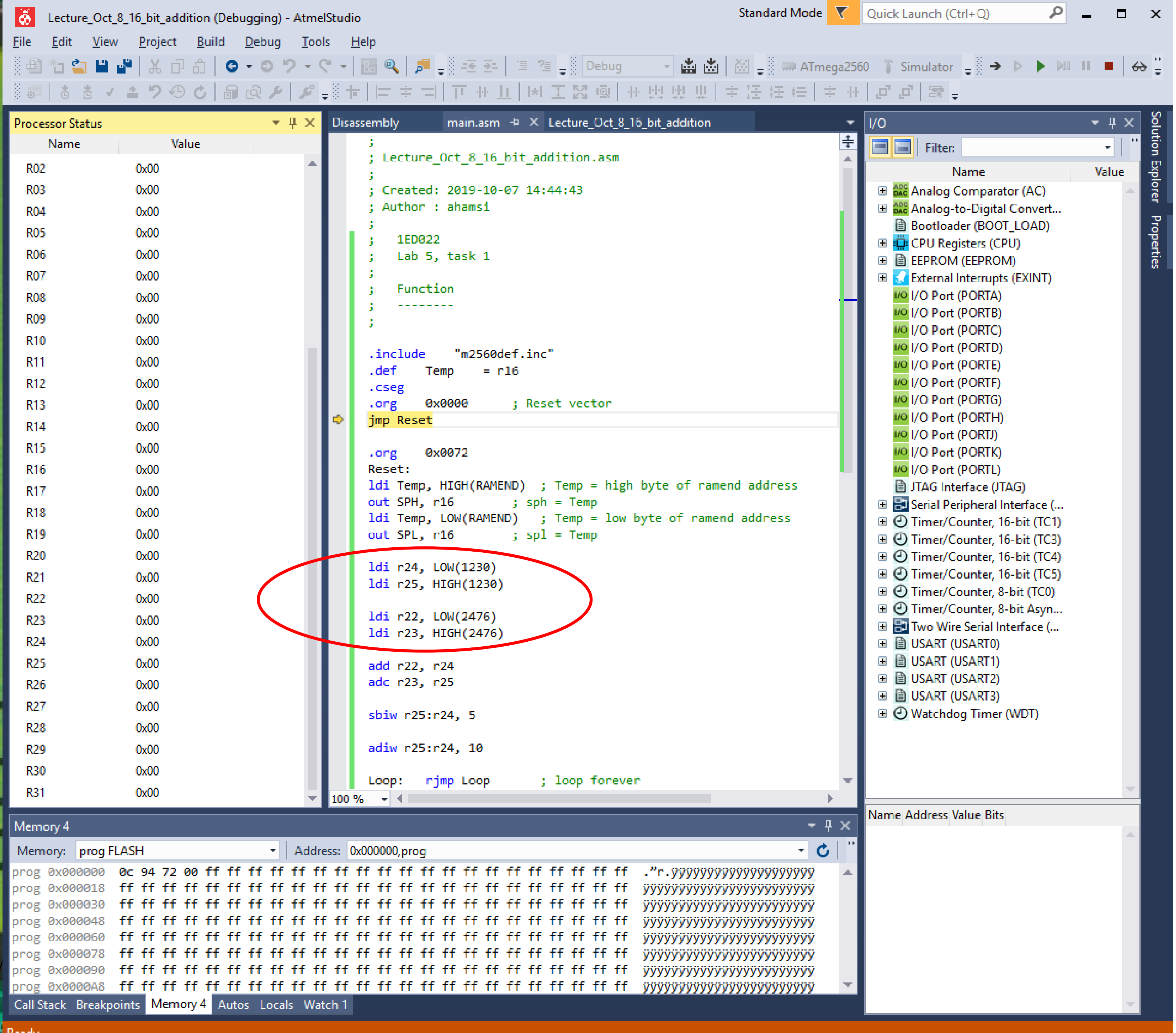
Example:

```
                  ; Add R1:R0 to R3:R2
   add   r2,r0    ; Add low byte
   adc   r3,r1    ; Add with carry high byte
```

Words: 1 (2 bytes)

Cycles: 1





Lecture_Oct_8_16_bit_addition (Debugging) - AtmelStudio

Processor Status

Name	Value
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x00
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x00
R23	0x00
R24	0x00
R25	0x00
R26	0x00
R27	0x00
R28	0x00
R29	0x00
R30	0x00
R31	0x00

```
;
; Lecture_Oct_8_16_bit_addition.asm
;
; Created: 2019-10-07 14:44:43
; Author : ahamsi
;
;   1ED022
;   Lab 5, task 1
;
;   Function
;   --------
;

.include    "m2560def.inc"
.def    Temp    = r16
.cseg
.org    0x0000      ; Reset vector
jmp Reset

.org    0x0072
Reset:
ldi Temp, HIGH(RAMEND)  ; Temp = high byte of ramend address
out SPH, r16        ; sph = Temp
ldi Temp, LOW(RAMEND)   ; Temp = low byte of ramend address
out SPL, r16        ; spl = Temp

ldi r24, LOW(1230)
ldi r25, HIGH(1230)

ldi r22, LOW(2476)
ldi r23, HIGH(2476)

add r22, r24
adc r23, r25

sbiw r25:r24, 5

adiw r25:r24, 10

Loop:   rjmp Loop       ; loop forever
```

I/O

Filter:

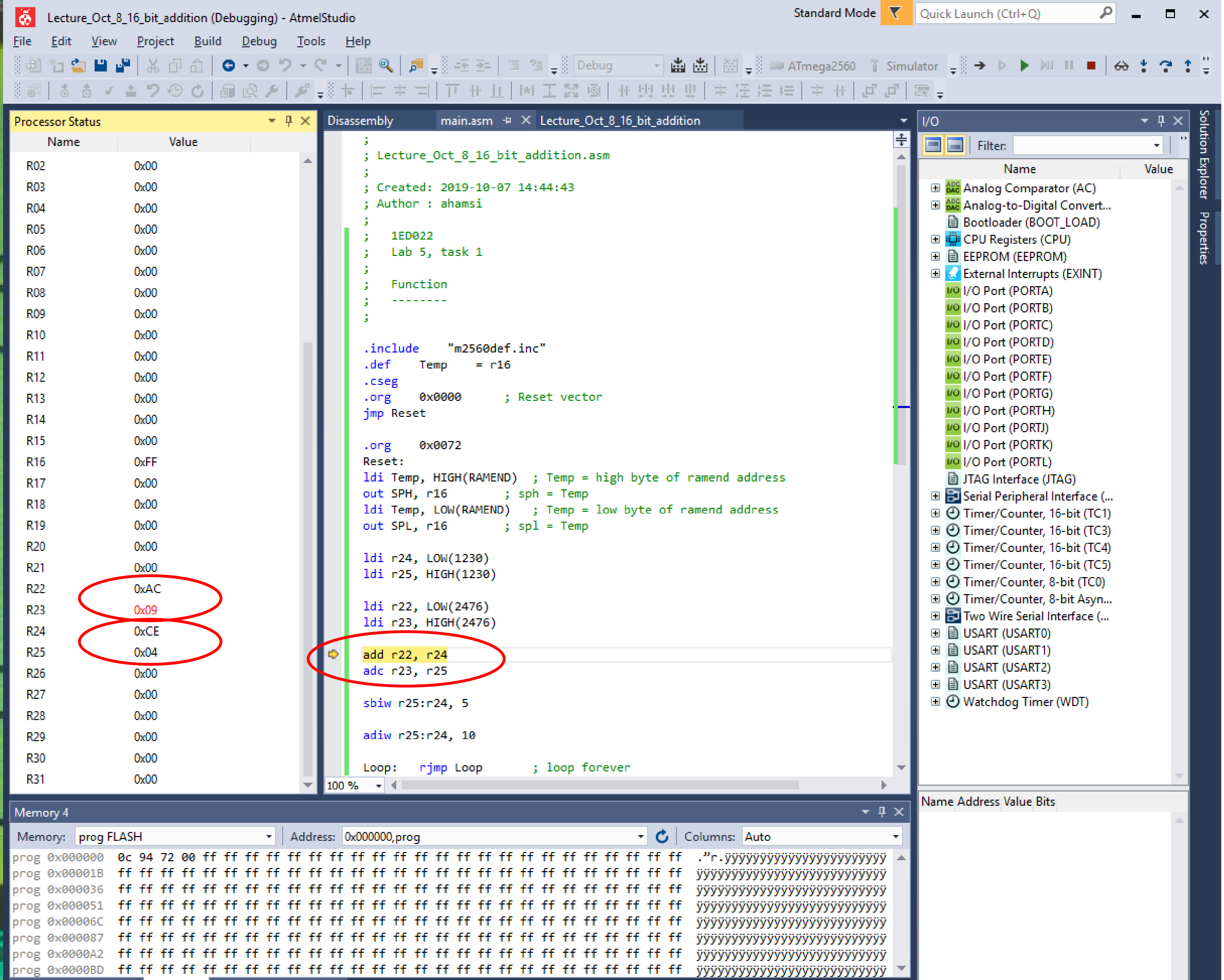
Name	Value
Analog Comparator (AC)	
Analog-to-Digital Convert...	
Bootloader (BOOT_LOAD)	
CPU Registers (CPU)	
EEPROM (EEPROM)	
External Interrupts (EXINT)	
I/O Port (PORTA)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	
I/O Port (PORTE)	
I/O Port (PORTF)	
I/O Port (PORTG)	
I/O Port (PORTH)	
I/O Port (PORTJ)	
I/O Port (PORTK)	
I/O Port (PORTL)	
JTAG Interface (JTAG)	
Serial Peripheral Interface (...	
Timer/Counter, 16-bit (TC1)	
Timer/Counter, 16-bit (TC3)	
Timer/Counter, 16-bit (TC4)	
Timer/Counter, 16-bit (TC5)	
Timer/Counter, 8-bit (TC0)	
Timer/Counter, 8-bit Asyn...	
Two Wire Serial Interface (...	
USART (USART0)	
USART (USART1)	
USART (USART2)	
USART (USART3)	
Watchdog Timer (WDT)	

Name	Address	Value	Bits

Memory 4

Memory: prog FLASH   Address: 0x000000,prog

```
prog 0x000000  0c 94 72 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  .”r.ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000018  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000030  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000048  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000060  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000078  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000090  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000A8  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

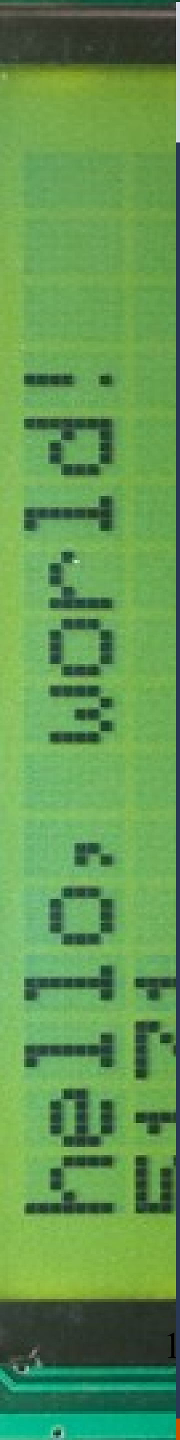


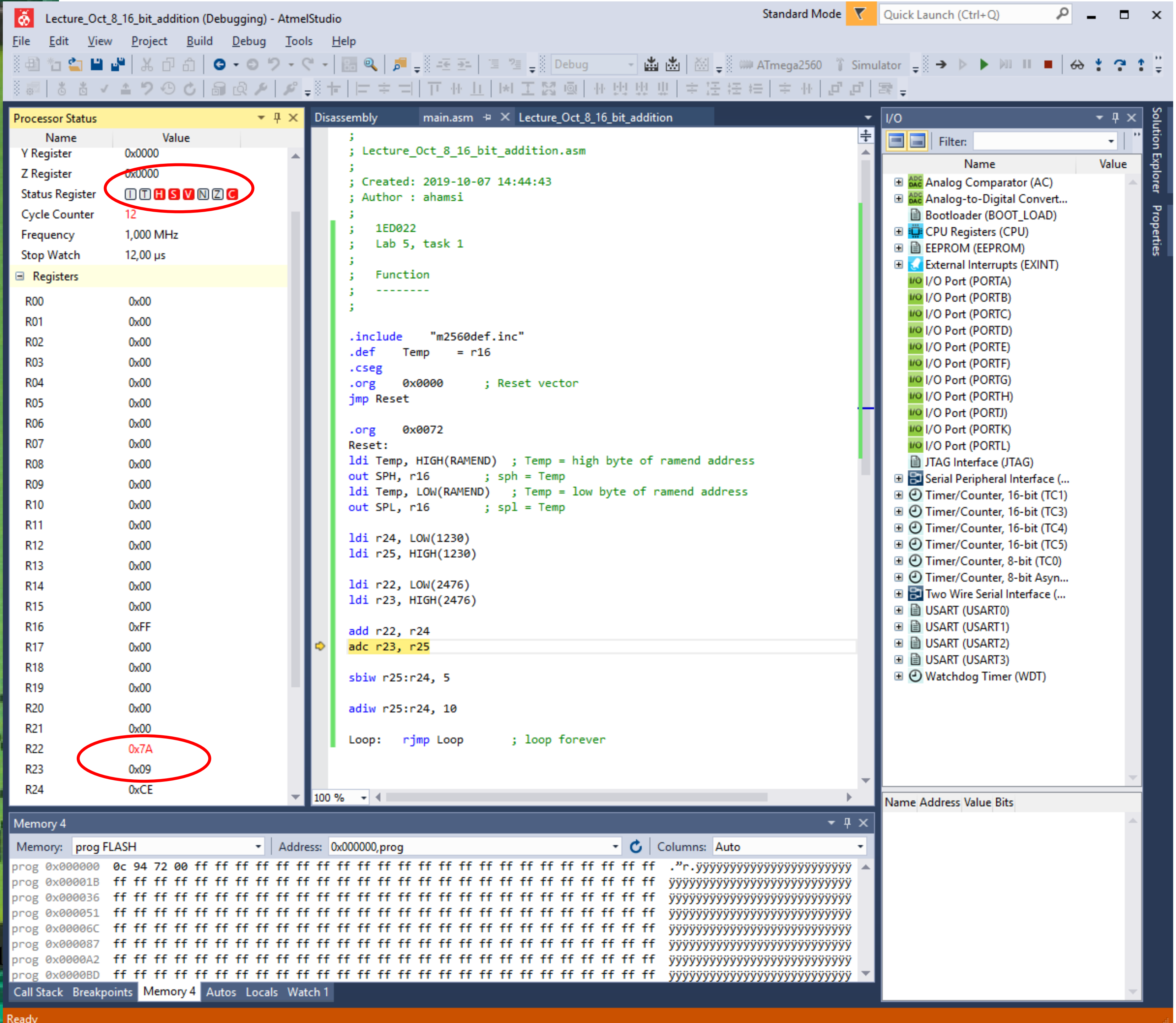
Name	Value
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0xFF
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0xAC
R23	0x09
R24	0xCE
R25	0x04
R26	0x00
R27	0x00
R28	0x00
R29	0x00
R30	0x00
R31	0x00

```
;
; Lecture_Oct_8_16_bit_addition.asm
;
; Created: 2019-10-07 14:44:43
; Author : ahamsi
;
;   1ED022
;   Lab 5, task 1
;
;   Function
;   --------
;

.include    "m2560def.inc"
.def    Temp    = r16
.cseg
.org    0x0000      ; Reset vector
jmp Reset

.org    0x0072
Reset:
ldi Temp, HIGH(RAMEND)  ; Temp = high byte of ramend address
out SPH, r16        ; sph = Temp
ldi Temp, LOW(RAMEND)   ; Temp = low byte of ramend address
out SPL, r16        ; spl = Temp

ldi r24, LOW(1230)
ldi r25, HIGH(1230)

ldi r22, LOW(2476)
ldi r23, HIGH(2476)

add r22, r24
adc r23, r25

sbiw r25:r24, 5

adiw r25:r24, 10

Loop:   rjmp Loop       ; loop forever
```

Memory: prog FLASH   Address: 0x000000,prog   Columns: Auto

```
prog 0x000000  0c 94 72 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ."r.ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00001B  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000051  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000087  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000A2  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000BD  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```




Lecture_Oct_8_16_bit_addition (Debugging) - AtmelStudio
Standard Mode
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Tools Help
Debug
ATmega2560
Simulator
Disassembly
main.asm
Lecture_Oct_8_16_bit_addition
Solution Explorer
Properties
Call Stack Breakpoints Memory 4 Autos Locals Watch 1
Ready

Processor Status

Name	Value
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	12
Frequency	1,000 MHz
Stop Watch	12,00 µs
⊟ Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0xFF
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x7A
R23	0x09
R24	0xCE

```
;
; Lecture_Oct_8_16_bit_addition.asm
;
; Created: 2019-10-07 14:44:43
; Author : ahamsi
;
;   1ED022
;   Lab 5, task 1
;
;   Function
;   --------
;

.include    "m2560def.inc"
.def     Temp    = r16
.cseg
.org     0x0000      ; Reset vector
jmp Reset

.org     0x0072
Reset:
ldi Temp, HIGH(RAMEND)  ; Temp = high byte of ramend address
out SPH, r16          ; sph = Temp
ldi Temp, LOW(RAMEND)    ; Temp = low byte of ramend address
out SPL, r16          ; spl = Temp

ldi r24, LOW(1230)
ldi r25, HIGH(1230)

ldi r22, LOW(2476)
ldi r23, HIGH(2476)

add r22, r24
adc r23, r25

sbiw r25:r24, 5

adiw r25:r24, 10

Loop:   rjmp Loop       ; loop forever
```

100 %

I/O

Filter:

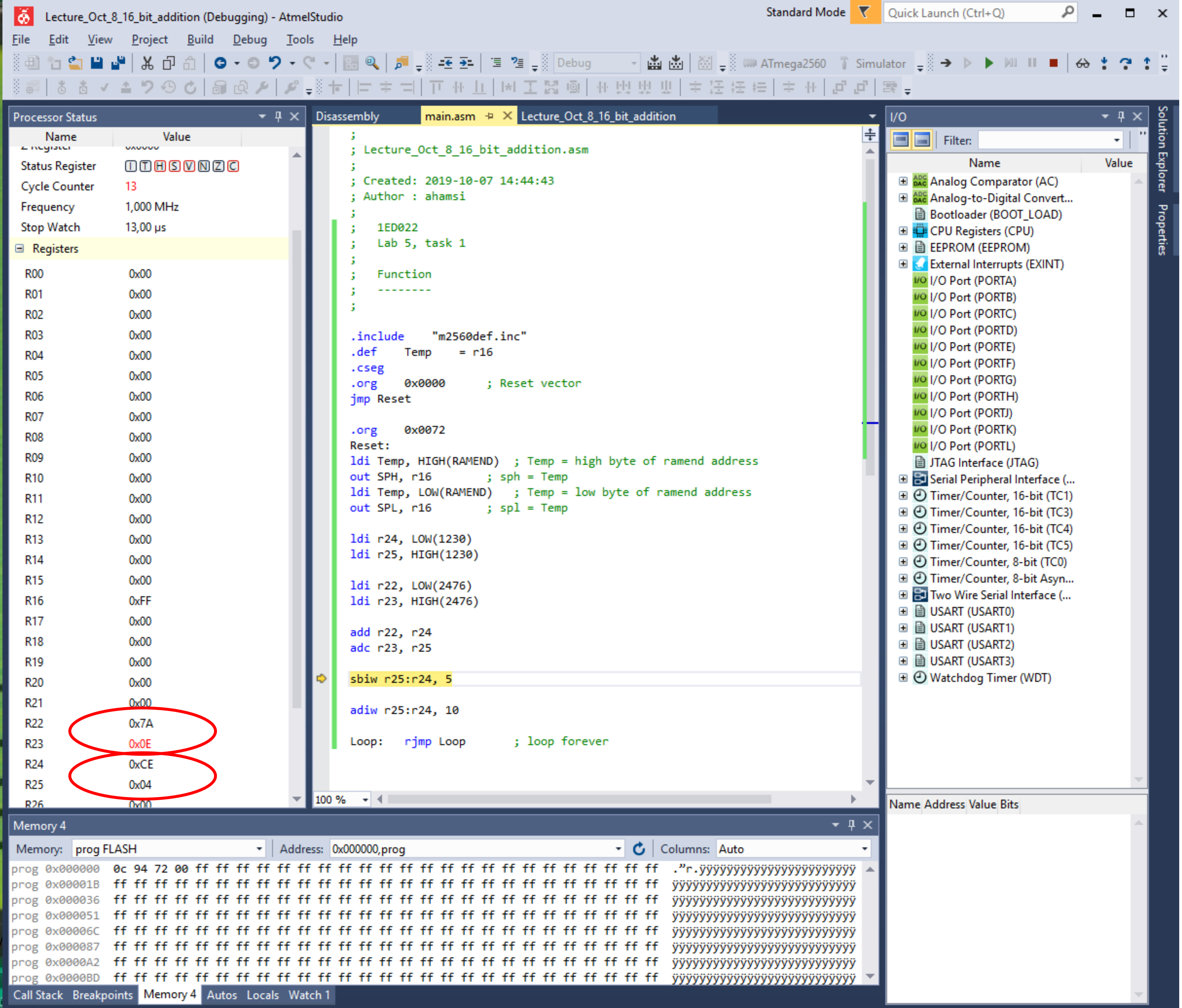
Name	Value
⊞ Analog Comparator (AC)	
⊞ Analog-to-Digital Convert...	
Bootloader (BOOT_LOAD)	
⊞ CPU Registers (CPU)	
⊞ EEPROM (EEPROM)	
⊞ External Interrupts (EXINT)	
I/O Port (PORTA)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	
I/O Port (PORTE)	
I/O Port (PORTF)	
I/O Port (PORTG)	
I/O Port (PORTH)	
I/O Port (PORTJ)	
I/O Port (PORTK)	
I/O Port (PORTL)	
JTAG Interface (JTAG)	
⊞ Serial Peripheral Interface (...	
⊞ Timer/Counter, 16-bit (TC1)	
⊞ Timer/Counter, 16-bit (TC3)	
⊞ Timer/Counter, 16-bit (TC4)	
⊞ Timer/Counter, 16-bit (TC5)	
⊞ Timer/Counter, 8-bit (TC0)	
⊞ Timer/Counter, 8-bit Asyn...	
⊞ Two Wire Serial Interface (...	
⊞ USART (USART0)	
⊞ USART (USART1)	
⊞ USART (USART2)	
⊞ USART (USART3)	
⊞ Watchdog Timer (WDT)	

Name	Address	Value	Bits

Memory 4

Memory: prog FLASH   Address: 0x000000,prog   Columns: Auto

```
prog 0x000000  0c 94 72 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  .”r.ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00001B  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000051  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000087  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000A2  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000BD  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

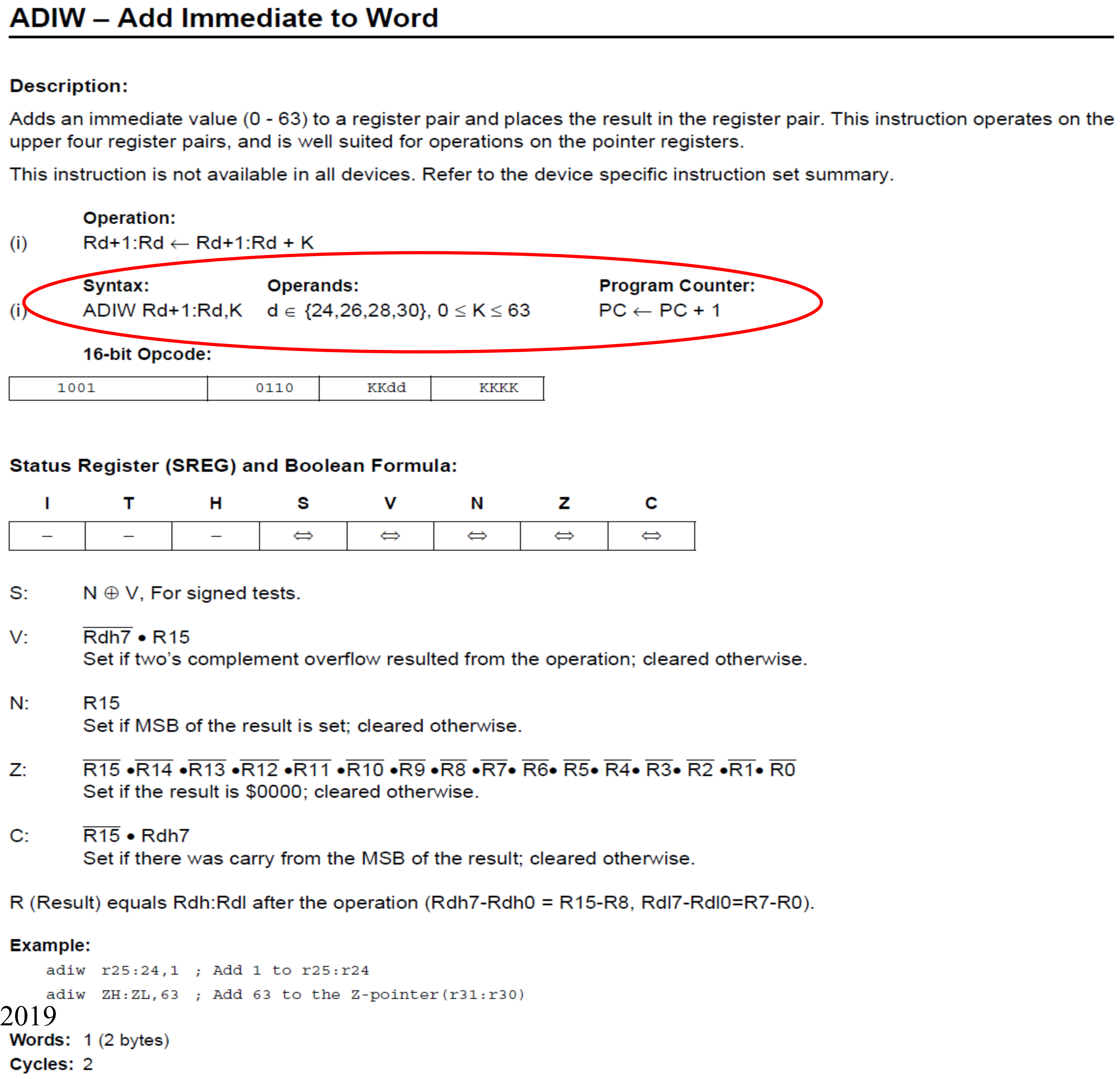


Name	Value
Status Register	I T H S V N Z C
Cycle Counter	13
Frequency	1,000 MHz
Stop Watch	13,00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0xFF
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x7A
R23	0x0E
R24	0xCE
R25	0x04
R26	0x00

```
;
; Lecture_Oct_8_16_bit_addition.asm
;
; Created: 2019-10-07 14:44:43
; Author : ahamsi
;
;   1ED022
;   Lab 5, task 1
;
;   Function
;   --------
;
;

.include    "m2560def.inc"
.def    Temp    = r16
.cseg
.org    0x0000      ; Reset vector
jmp Reset

.org    0x0072
Reset:
ldi Temp, HIGH(RAMEND)  ; Temp = high byte of ramend address
out SPH, r16            ; sph = Temp
ldi Temp, LOW(RAMEND)   ; Temp = low byte of ramend address
out SPL, r16            ; spl = Temp

ldi r24, LOW(1230)
ldi r25, HIGH(1230)

ldi r22, LOW(2476)
ldi r23, HIGH(2476)

add r22, r24
adc r23, r25

sbiw r25:r24, 5

adiw r25:r24, 10

Loop:   rjmp Loop       ; loop forever
```

Memory: prog FLASH   Address: 0x000000,prog   Columns: Auto

```
prog 0x000000  0c 94 72 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  .”r.ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00001B  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000051  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000087  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000A2  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000BD  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```



ADIW – Add Immediate to Word

Description:

Adds an immediate value (0 - 63) to a register pair and places the result in the register pair. This instruction operates on the upper four register pairs, and is well suited for operations on the pointer registers.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

	Operation:		
(i)	Rd+1:Rd ← Rd+1:Rd + K		
	Syntax:	**Operands:**	**Program Counter:**
(i)	ADIW Rd+1:Rd,K	d ∈ {24,26,28,30}, 0 ≤ K ≤ 63	PC ← PC + 1

16-bit Opcode:

1001	0110	KKdd	KKKK

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	⇔

S: N ⊕ V, For signed tests.

V: $\overline{Rdh7} \bullet R15$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R15
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{R15} \bullet \overline{R14} \bullet \overline{R13} \bullet \overline{R12} \bullet \overline{R11} \bullet \overline{R10} \bullet \overline{R9} \bullet \overline{R8} \bullet \overline{R7} \bullet \overline{R6} \bullet \overline{R5} \bullet \overline{R4} \bullet \overline{R3} \bullet \overline{R2} \bullet \overline{R1} \bullet \overline{R0}$
Set if the result is \$0000; cleared otherwise.

C: $\overline{R15} \bullet Rdh7$
Set if there was carry from the MSB of the result; cleared otherwise.

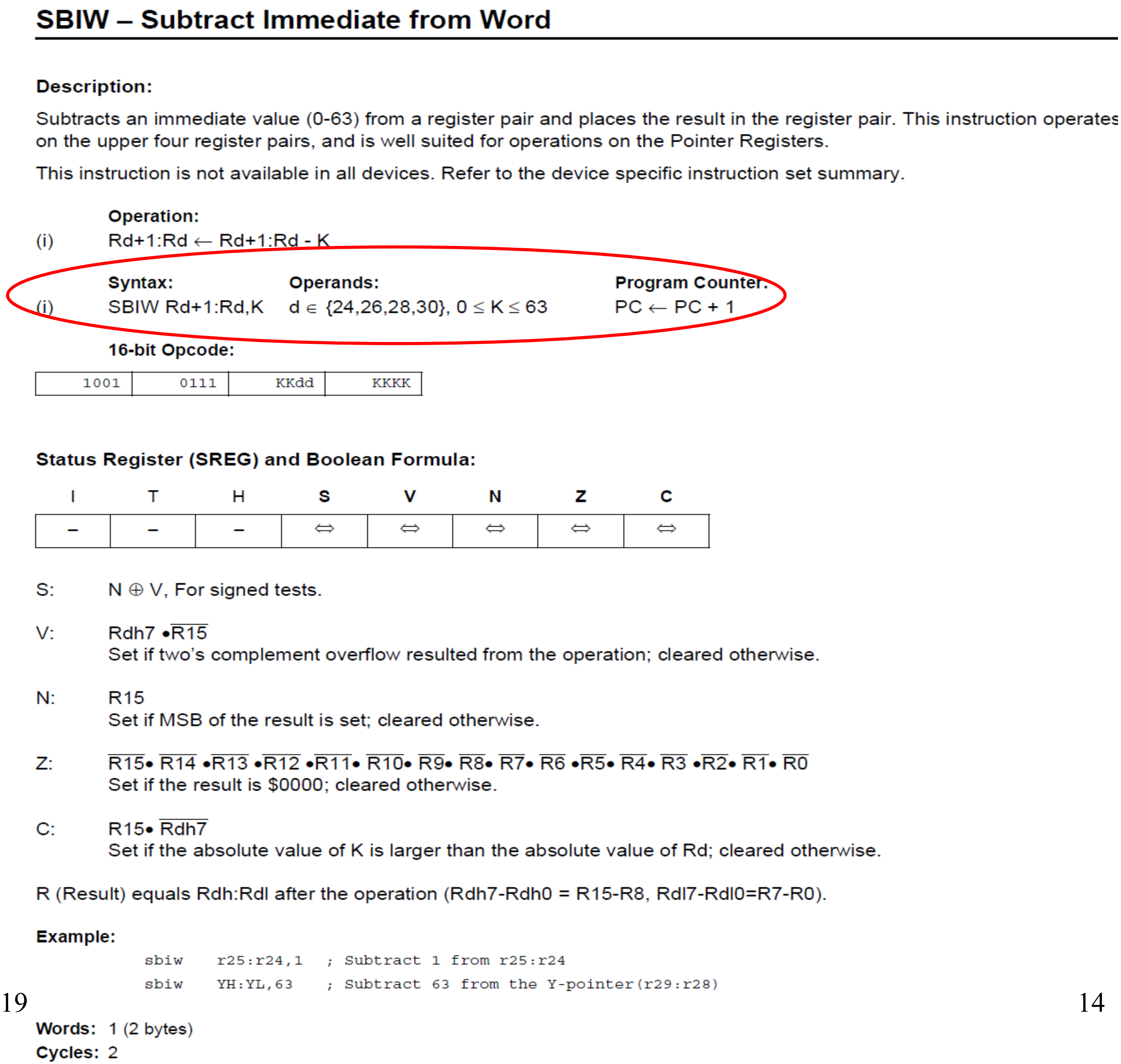
R (Result) equals Rdh:Rdl after the operation (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0=R7-R0).

Example:

```
    adiw  r25:24,1  ; Add 1 to r25:r24
    adiw  ZH:ZL,63  ; Add 63 to the Z-pointer(r31:r30)
```

Words: 1 (2 bytes)

Cycles: 2



SBIW – Subtract Immediate from Word

Description:

Subtracts an immediate value (0-63) from a register pair and places the result in the register pair. This instruction operates on the upper four register pairs, and is well suited for operations on the Pointer Registers.

This instruction is not available in all devices. Refer to the device specific instruction set summary.

	Operation:		
(i)	Rd+1:Rd ← Rd+1:Rd - K		
	Syntax:	**Operands:**	**Program Counter:**
(i)	SBIW Rd+1:Rd,K	d ∈ {24,26,28,30}, 0 ≤ K ≤ 63	PC ← PC + 1

16-bit Opcode:

1001	0111	KKdd	KKKK

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	⇔

S: N ⊕ V, For signed tests.

V: Rdh7 •$\overline{\text{R15}}$
Set if two's complement overflow resulted from the operation; cleared otherwise.

N: R15
Set if MSB of the result is set; cleared otherwise.

Z: $\overline{\text{R15}}$• $\overline{\text{R14}}$ •$\overline{\text{R13}}$ •$\overline{\text{R12}}$ •$\overline{\text{R11}}$• $\overline{\text{R10}}$• $\overline{\text{R9}}$• $\overline{\text{R8}}$• $\overline{\text{R7}}$• $\overline{\text{R6}}$ •$\overline{\text{R5}}$• $\overline{\text{R4}}$• $\overline{\text{R3}}$ •$\overline{\text{R2}}$• $\overline{\text{R1}}$• $\overline{\text{R0}}$
Set if the result is \$0000; cleared otherwise.

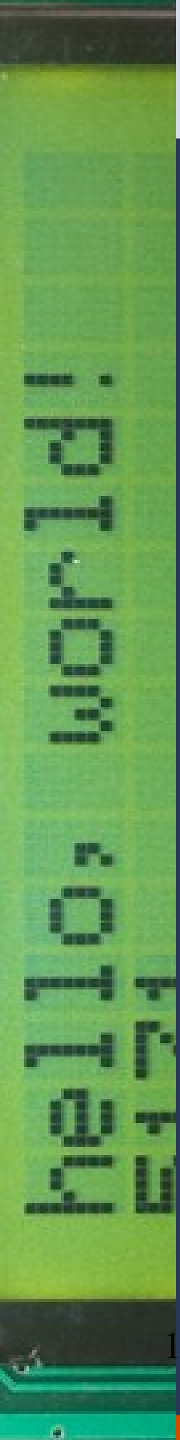
C: R15• $\overline{\text{Rdh7}}$
Set if the absolute value of K is larger than the absolute value of Rd; cleared otherwise.

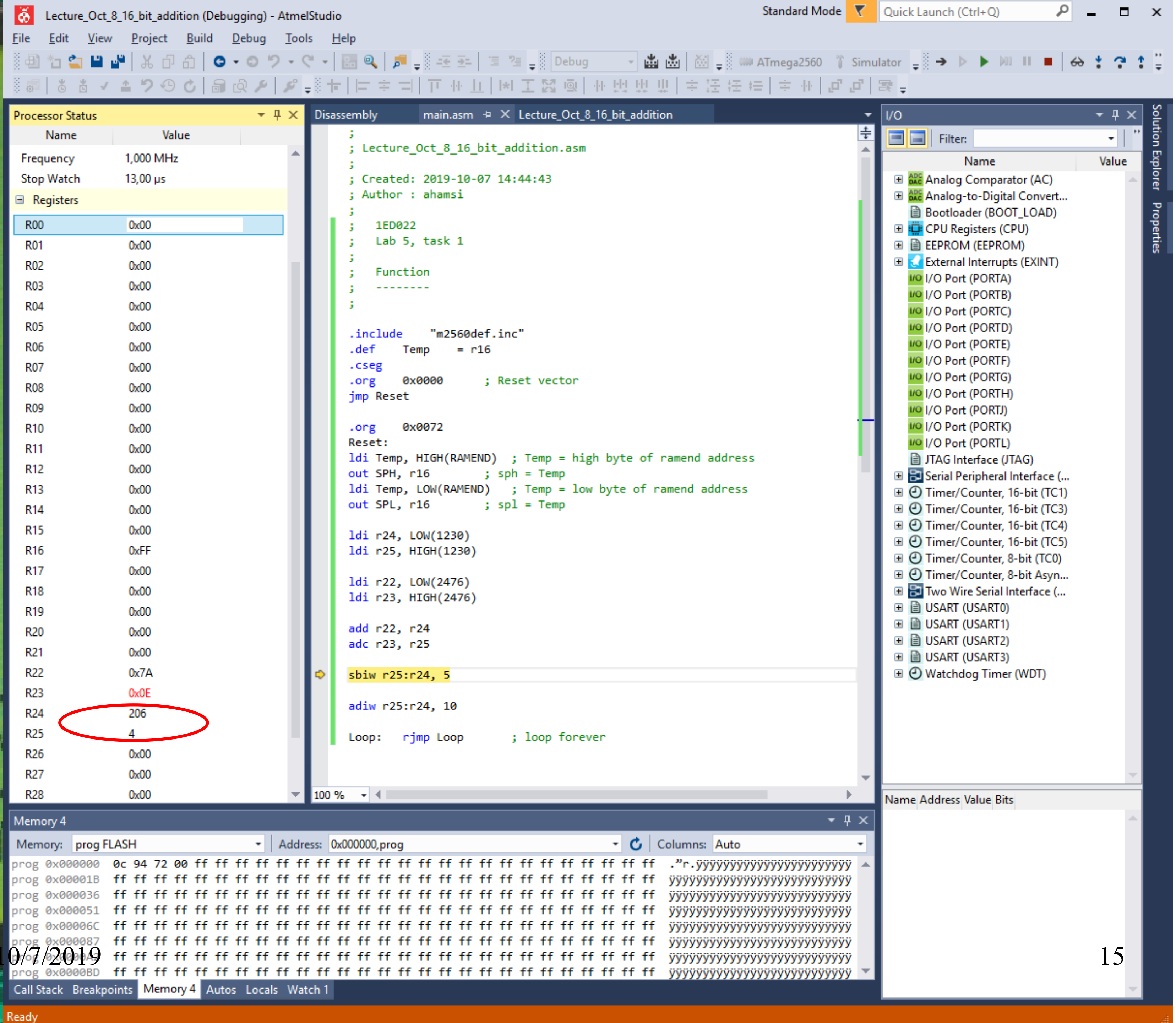
R (Result) equals Rdh:Rdl after the operation (Rdh7-Rdh0 = R15-R8, Rdl7-Rdl0=R7-R0).

Example:

```
sbiw   r25:r24,1   ; Subtract 1 from r25:r24
sbiw   YH:YL,63    ; Subtract 63 from the Y-pointer(r29:r28)
```

Words: 1 (2 bytes)
Cycles: 2





Processor Status

Name	Value
Frequency	1,000 MHz
Stop Watch	13,00 µs
⊟ Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0xFF
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x7A
R23	0x0E
R24	206
R25	4
R26	0x00
R27	0x00
R28	0x00

```
;
; Lecture_Oct_8_16_bit_addition.asm
;
; Created: 2019-10-07 14:44:43
; Author : ahamsi
;
;   1ED022
;   Lab 5, task 1
;
;   Function
;   --------
;

.include    "m2560def.inc"
.def    Temp    = r16
.cseg
.org    0x0000      ; Reset vector
jmp Reset

.org    0x0072
Reset:
ldi Temp, HIGH(RAMEND)  ; Temp = high byte of ramend address
out SPH, r16         ; sph = Temp
ldi Temp, LOW(RAMEND)   ; Temp = low byte of ramend address
out SPL, r16         ; spl = Temp

ldi r24, LOW(1230)
ldi r25, HIGH(1230)

ldi r22, LOW(2476)
ldi r23, HIGH(2476)

add r22, r24
adc r23, r25

sbiw r25:r24, 5

adiw r25:r24, 10

Loop:   rjmp Loop       ; loop forever
```

100 %

I/O

Filter:

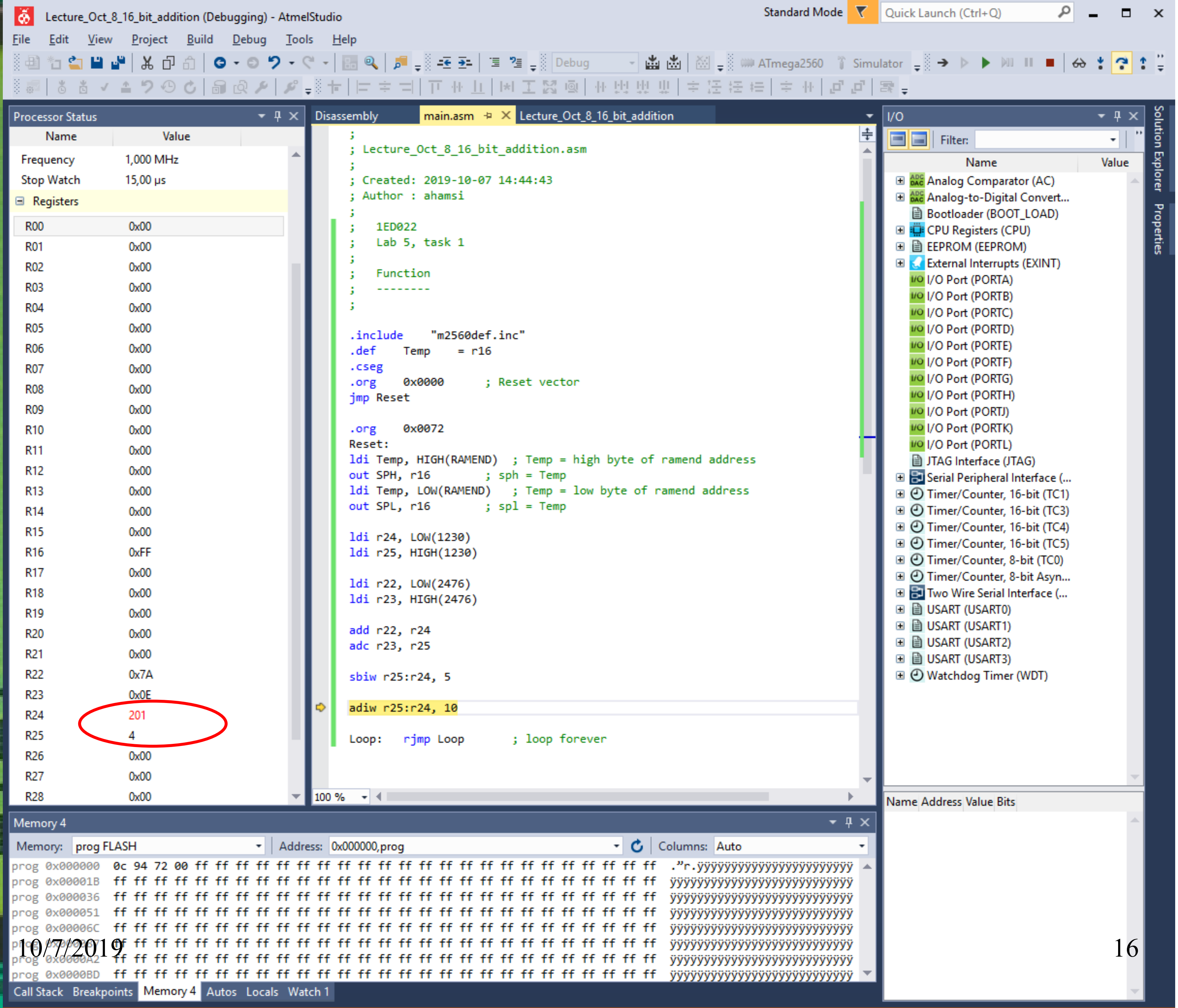
Name	Value
⊞ Analog Comparator (AC)	
⊞ Analog-to-Digital Convert...	
Bootloader (BOOT_LOAD)	
⊞ CPU Registers (CPU)	
⊞ EEPROM (EEPROM)	
⊞ External Interrupts (EXINT)	
I/O Port (PORTA)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	
I/O Port (PORTE)	
I/O Port (PORTF)	
I/O Port (PORTG)	
I/O Port (PORTH)	
I/O Port (PORTJ)	
I/O Port (PORTK)	
I/O Port (PORTL)	
JTAG Interface (JTAG)	
⊞ Serial Peripheral Interface (...	
⊞ Timer/Counter, 16-bit (TC1)	
⊞ Timer/Counter, 16-bit (TC3)	
⊞ Timer/Counter, 16-bit (TC4)	
⊞ Timer/Counter, 16-bit (TC5)	
⊞ Timer/Counter, 8-bit (TC0)	
⊞ Timer/Counter, 8-bit Asyn...	
⊞ Two Wire Serial Interface (...	
⊞ USART (USART0)	
⊞ USART (USART1)	
⊞ USART (USART2)	
⊞ USART (USART3)	
⊞ Watchdog Timer (WDT)	

Name	Address	Value	Bits

Memory 4

Memory: prog FLASH   Address: 0x000000,prog   Columns: Auto

```
prog 0x000000  0c 94 72 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ."r.ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00001B  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000051  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000087  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog           ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000BD  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

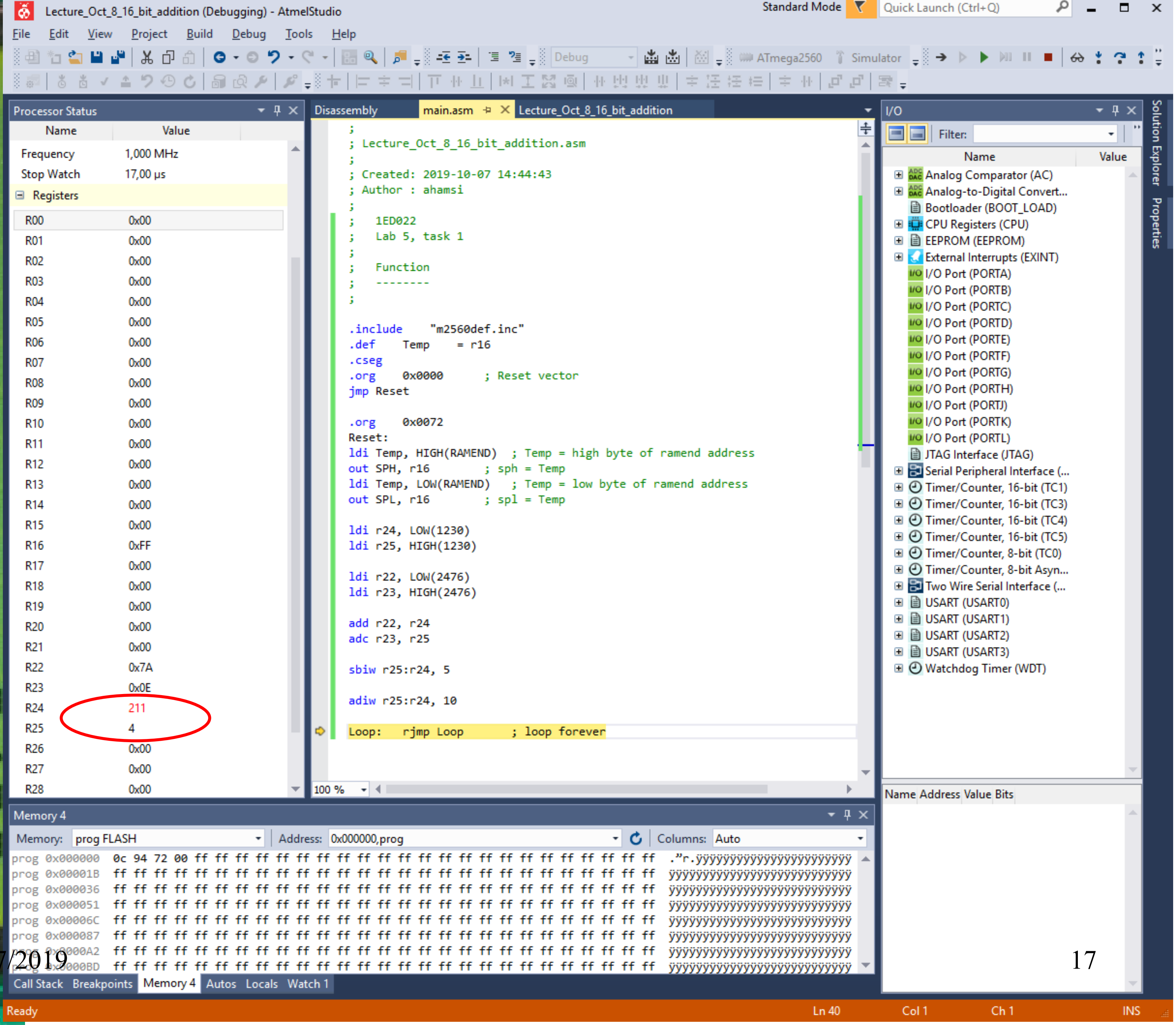


Name	Value
Frequency	1,000 MHz
Stop Watch	15,00 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0xFF
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x7A
R23	0x0E
R24	201
R25	4
R26	0x00
R27	0x00
R28	0x00

```
;
; Lecture_Oct_8_16_bit_addition.asm
;
; Created: 2019-10-07 14:44:43
; Author : ahamsi
;
;   1ED022
;   Lab 5, task 1
;
;   Function
;   --------
;

.include    "m2560def.inc"
.def    Temp    = r16
.cseg
.org    0x0000      ; Reset vector
jmp Reset

.org    0x0072
Reset:
ldi Temp, HIGH(RAMEND)  ; Temp = high byte of ramend address
out SPH, r16        ; sph = Temp
ldi Temp, LOW(RAMEND)   ; Temp = low byte of ramend address
out SPL, r16        ; spl = Temp

ldi r24, LOW(1230)
ldi r25, HIGH(1230)

ldi r22, LOW(2476)
ldi r23, HIGH(2476)

add r22, r24
adc r23, r25

sbiw r25:r24, 5

adiw r25:r24, 10

Loop:   rjmp Loop       ; loop forever
```

Memory: prog FLASH Address: 0x000000,prog Columns: Auto

```
prog 0x000000  0c 94 72 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  .”r.ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00001B  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000051  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000087  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000A2  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000BD  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```


Lecture_Oct_8_16_bit_addition (Debugging) - AtmelStudio
Standard Mode
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Tools Help
Debug
ATmega2560
Simulator
Processor Status
Disassembly
main.asm
Lecture_Oct_8_16_bit_addition
I/O
Filter:
Solution Explorer
Properties
100 %
Call Stack Breakpoints Memory 4 Autos Locals Watch 1
Ready
Ln 40
Col 1
Ch 1
INS

Processor Status

Name	Value
Frequency	1,000 MHz
Stop Watch	17,00 µs
⊟ Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0xFF
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x7A
R23	0x0E
R24	211
R25	4
R26	0x00
R27	0x00
R28	0x00

main.asm

```
;
; Lecture_Oct_8_16_bit_addition.asm
;
; Created: 2019-10-07 14:44:43
; Author : ahamsi
;
;   1ED022
;   Lab 5, task 1
;
;   Function
;   --------
;

.include    "m2560def.inc"
.def    Temp    = r16
.cseg
.org    0x0000      ; Reset vector
jmp Reset

.org    0x0072
Reset:
ldi Temp, HIGH(RAMEND)  ; Temp = high byte of ramend address
out SPH, r16            ; sph = Temp
ldi Temp, LOW(RAMEND)   ; Temp = low byte of ramend address
out SPL, r16            ; spl = Temp

ldi r24, LOW(1230)
ldi r25, HIGH(1230)

ldi r22, LOW(2476)
ldi r23, HIGH(2476)

add r22, r24
adc r23, r25

sbiw r25:r24, 5

adiw r25:r24, 10

Loop:   rjmp Loop       ; loop forever
```

I/O

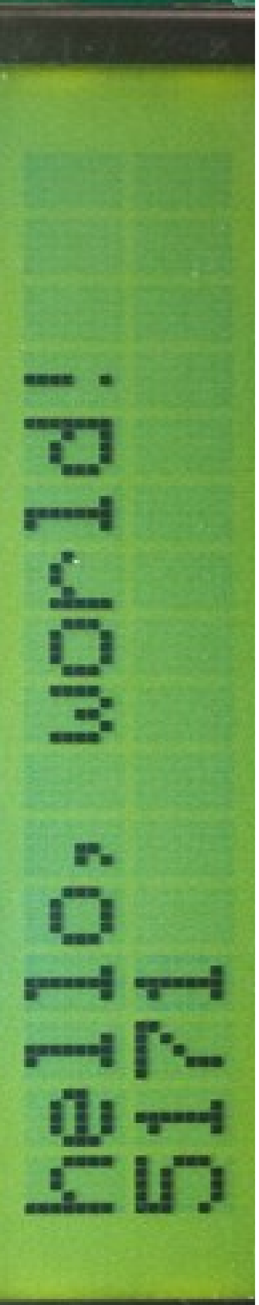
Name	Value
⊞ Analog Comparator (AC)	
⊞ Analog-to-Digital Convert...	
Bootloader (BOOT_LOAD)	
⊞ CPU Registers (CPU)	
⊞ EEPROM (EEPROM)	
⊞ External Interrupts (EXINT)	
I/O Port (PORTA)	
I/O Port (PORTB)	
I/O Port (PORTC)	
I/O Port (PORTD)	
I/O Port (PORTE)	
I/O Port (PORTF)	
I/O Port (PORTG)	
I/O Port (PORTH)	
I/O Port (PORTJ)	
I/O Port (PORTK)	
I/O Port (PORTL)	
JTAG Interface (JTAG)	
⊞ Serial Peripheral Interface (...	
⊞ Timer/Counter, 16-bit (TC1)	
⊞ Timer/Counter, 16-bit (TC3)	
⊞ Timer/Counter, 16-bit (TC4)	
⊞ Timer/Counter, 16-bit (TC5)	
⊞ Timer/Counter, 8-bit (TC0)	
⊞ Timer/Counter, 8-bit Asyn...	
⊞ Two Wire Serial Interface (...	
⊞ USART (USART0)	
⊞ USART (USART1)	
⊞ USART (USART2)	
⊞ USART (USART3)	
⊞ Watchdog Timer (WDT)	

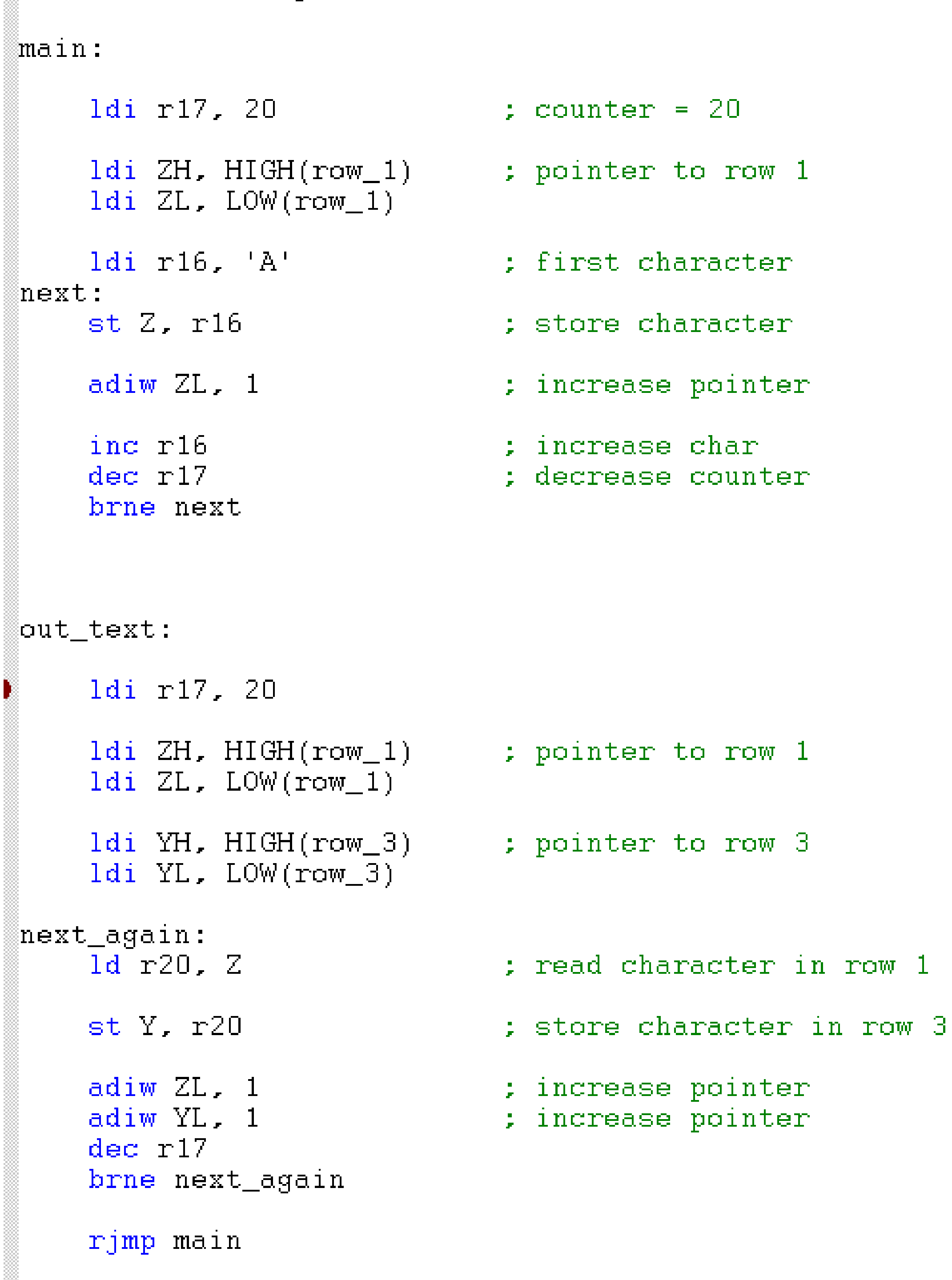
Name Address Value Bits

Memory 4

Memory: prog FLASH   Address: 0x000000,prog   Columns: Auto

```
prog 0x000000  0c 94 72 00 ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  .”r.ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00001B  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000036  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000051  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x00006C  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x000087  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000A2  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
prog 0x0000BD  ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```





```
main:

    ldi r17, 20                 ; counter = 20

    ldi ZH, HIGH(row_1)         ; pointer to row 1
    ldi ZL, LOW(row_1)

    ldi r16, 'A'                ; first character
next:
    st Z, r16                   ; store character

    adiw ZL, 1                  ; increase pointer

    inc r16                     ; increase char
    dec r17                     ; decrease counter
    brne next



out_text:

    ldi r17, 20

    ldi ZH, HIGH(row_1)         ; pointer to row 1
    ldi ZL, LOW(row_1)

    ldi YH, HIGH(row_3)         ; pointer to row 3
    ldi YL, LOW(row_3)

next_again:
    ld r20, Z                   ; read character in row 1

    st Y, r20                   ; store character in row 3

    adiw ZL, 1                  ; increase pointer
    adiw YL, 1                  ; increase pointer
    dec r17
    brne next_again

    rjmp main
```

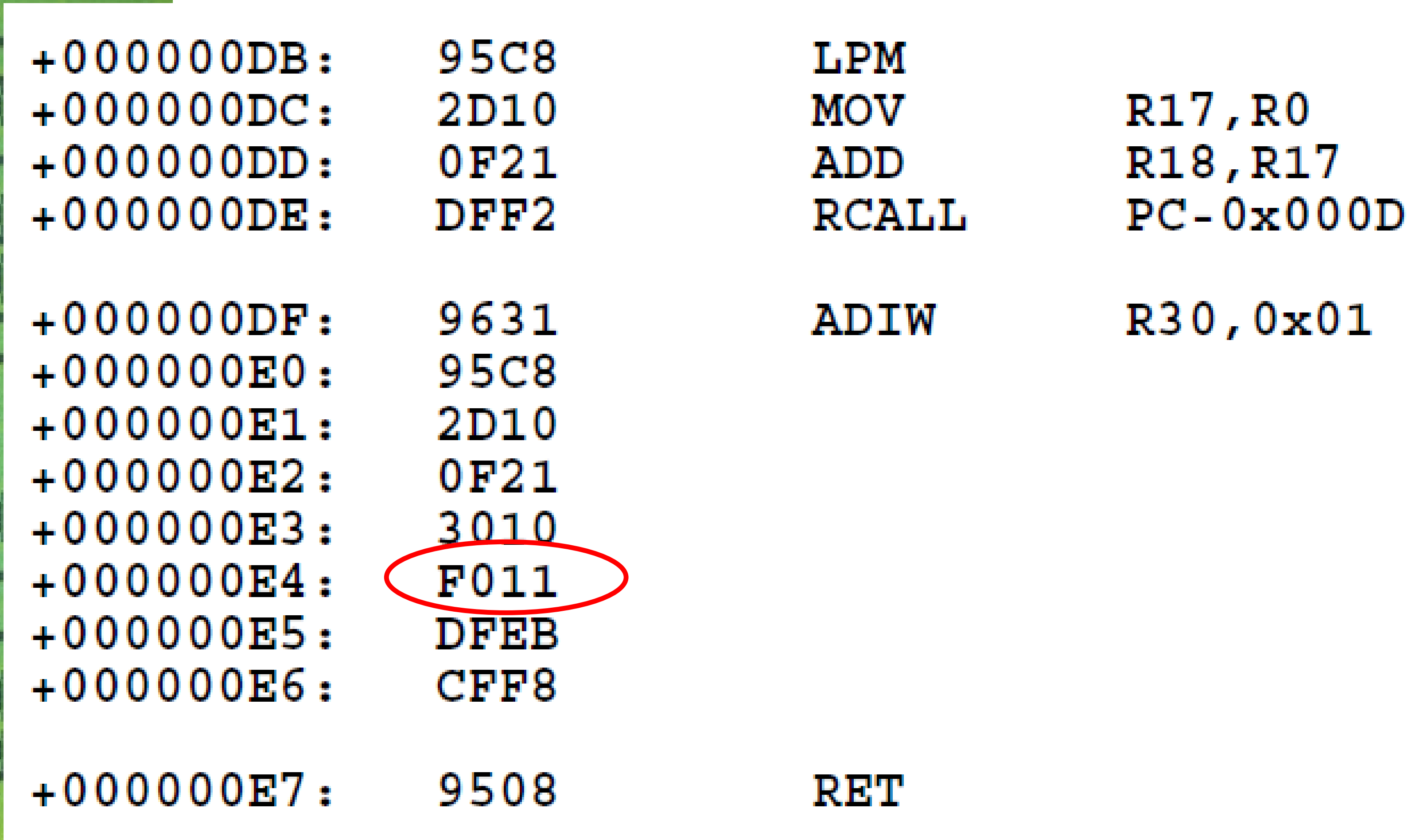
2. Machine code

Below is part of a program, cut out from the disassembler. Machine code is printed with the most significant byte first, ie as it is described in the Instruction Set Manual.
Example: For the instruction RET, the machine code is 9508.
On lines E0 – E6 the assembler code is removed.

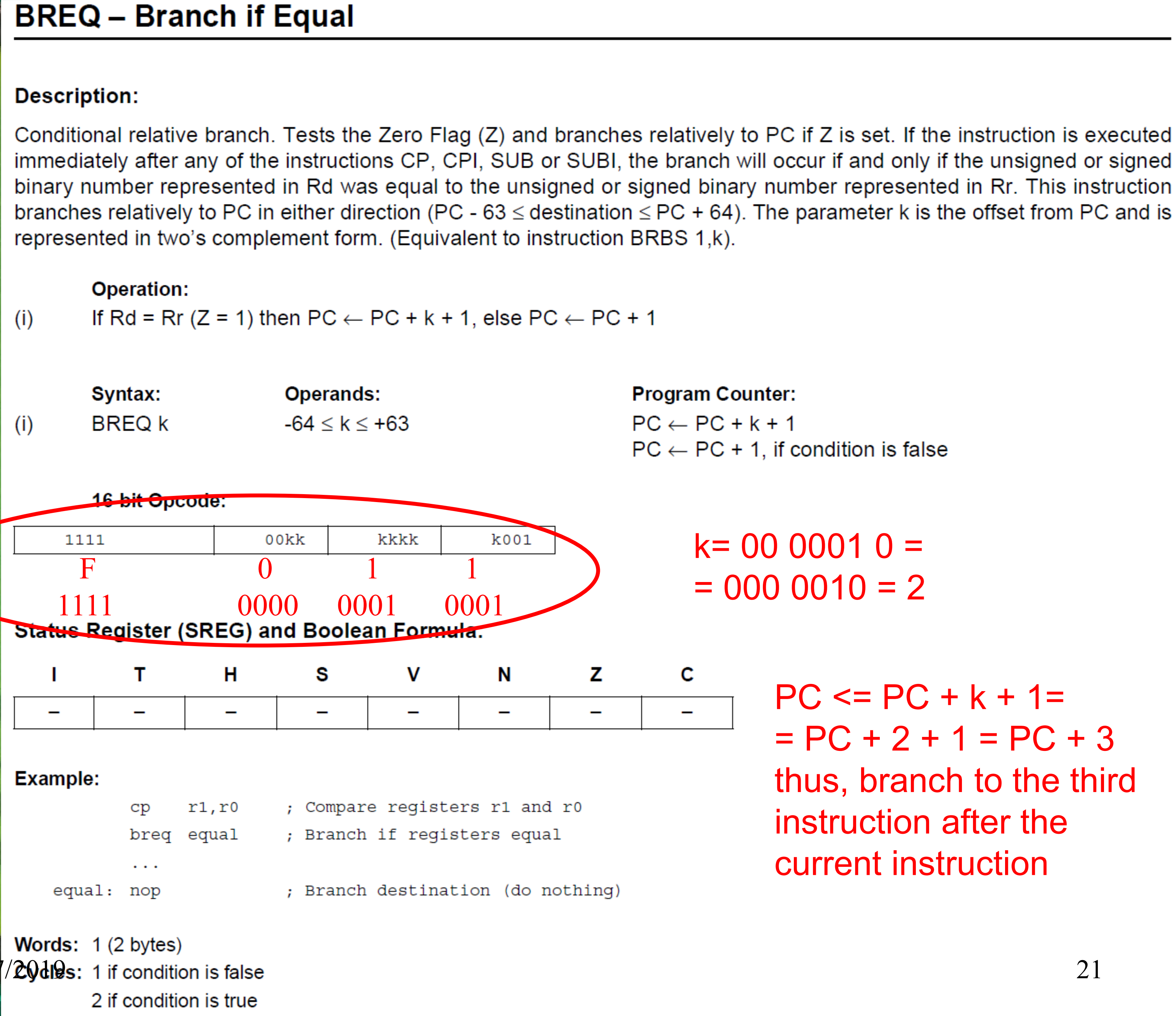
a) Recreate the assembler code by interpreting machine code. Use the enclosed examples from the Instruction Set Manual. **5p**

b) Explain the function of the complete program. **5p**

```
+000000DB:   95C8        LPM
+000000DC:   2D10        MOV       R17,R0
+000000DD:   0F21        ADD       R18,R17
+000000DE:   DFF2        RCALL     PC-0x000D

+000000DF:   9631        ADIW      R30,0x01
+000000E0:   95C8
+000000E1:   2D10
+000000E2:   0F21
+000000E3:   3010
+000000E4:   F011
+000000E5:   DFEB
+000000E6:   CFF8

+000000E7:   9508        RET
```



```
+000000DB:    95C8          LPM
+000000DC:    2D10          MOV      R17,R0
+000000DD:    0F21          ADD      R18,R17
+000000DE:    DFF2          RCALL    PC-0x000D

+000000DF:    9631          ADIW     R30,0x01
+000000E0:    95C8
+000000E1:    2D10
+000000E2:    0F21
+000000E3:    3010
+000000E4:    F011
+000000E5:    DFEB
+000000E6:    CFF8

+000000E7:    9508          RET
```



BREQ – Branch if Equal

Description:

Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is set. If the instruction is executed immediately after any of the instructions CP, CPI, SUB or SUBI, the branch will occur if and only if the unsigned or signed binary number represented in Rd was equal to the unsigned or signed binary number represented in Rr. This instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). The parameter k is the offset from PC and is represented in two's complement form. (Equivalent to instruction BRBS 1,k).

Operation:

(i) If Rd = Rr (Z = 1) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BREQ k	-64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16 bit Opcode:

1111	00kk	kkkk	k001
F	0	1	1
1111	0000	0001	0001

k= 00 0001 0 =
= 000 0010 = 2

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

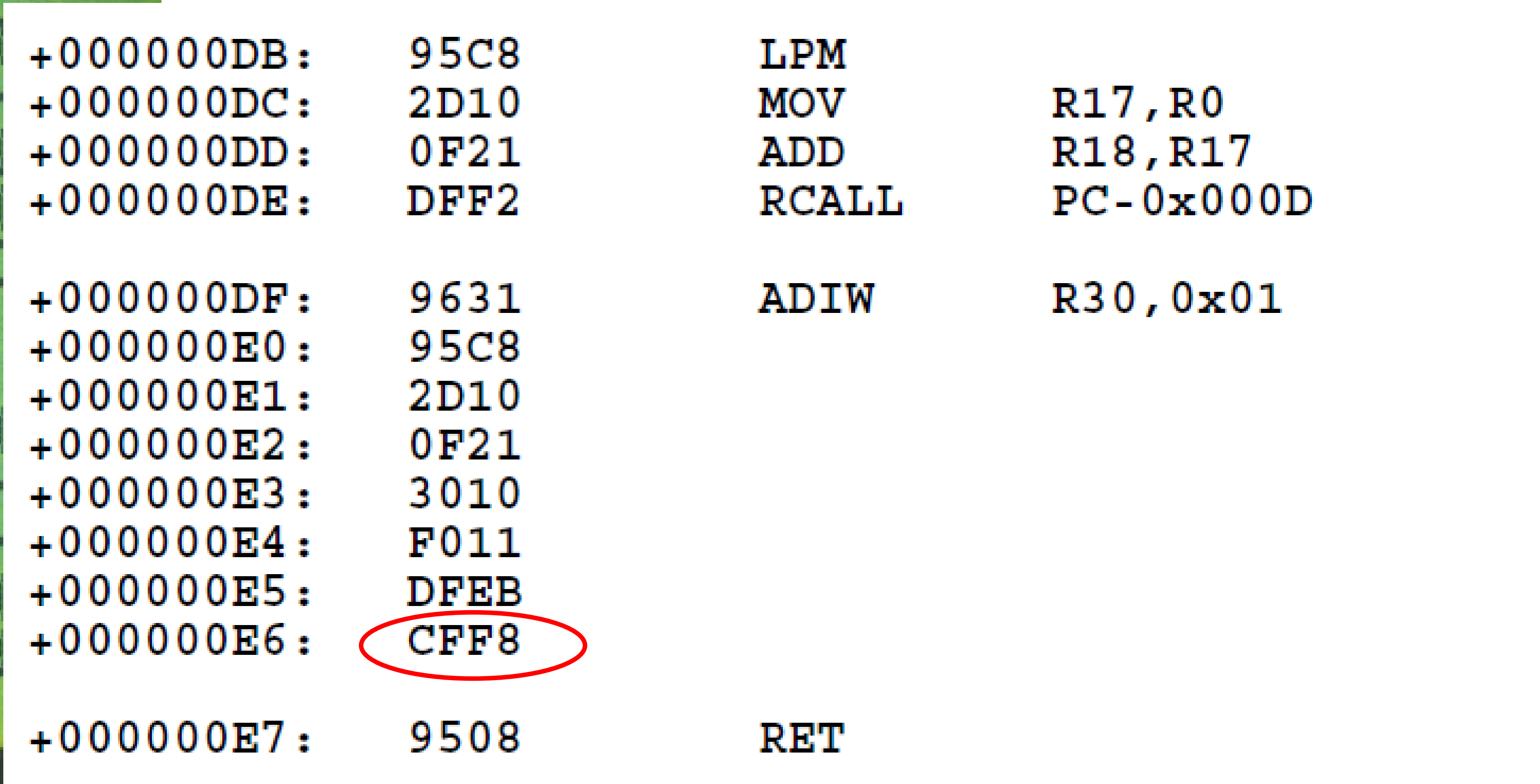
Example:

```
        cp    r1,r0      ; Compare registers r1 and r0
        breq  equal      ; Branch if registers equal
        ...
equal:  nop              ; Branch destination (do nothing)
```

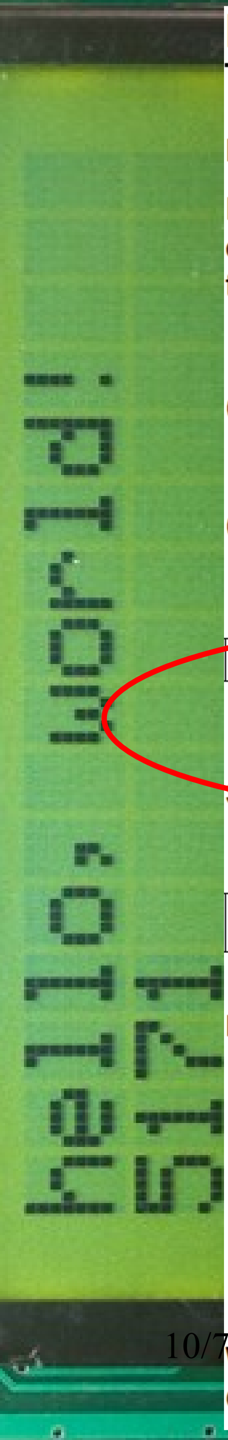
PC <= PC + k + 1=
= PC + 2 + 1 = PC + 3
thus, branch to the third instruction after the current instruction

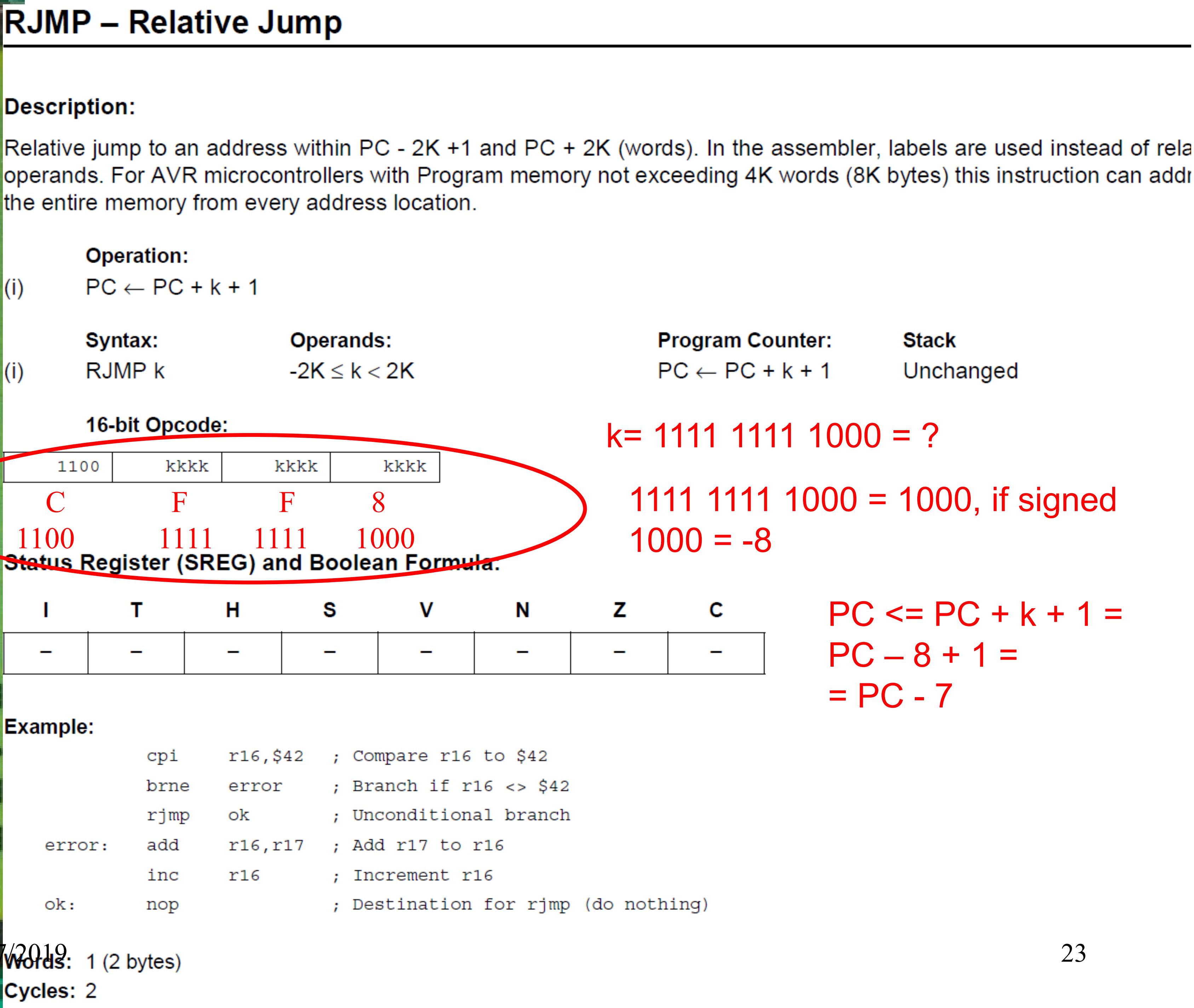
Words: 1 (2 bytes)

Cycles: 1 if condition is false
2 if condition is true



```
+000000DB:    95C8          LPM
+000000DC:    2D10          MOV        R17,R0
+000000DD:    0F21          ADD        R18,R17
+000000DE:    DFF2          RCALL      PC-0x000D

+000000DF:    9631          ADIW       R30,0x01
+000000E0:    95C8
+000000E1:    2D10
+000000E2:    0F21
+000000E3:    3010
+000000E4:    F011
+000000E5:    DFEB
+000000E6:    CFF8

+000000E7:    9508          RET
```





RJMP – Relative Jump

Description:

Relative jump to an address within PC - 2K +1 and PC + 2K (words). In the assembler, labels are used instead of rela operands. For AVR microcontrollers with Program memory not exceeding 4K words (8K bytes) this instruction can addr the entire memory from every address location.

Operation:

(i) PC ← PC + k + 1

	Syntax:	Operands:	Program Counter:	Stack
(i)	RJMP k	-2K ≤ k < 2K	PC ← PC + k + 1	Unchanged

16-bit Opcode:

1100	kkkk	kkkk	kkkk
C	F	F	8
1100	1111	1111	1000

k= 1111 1111 1000 = ?

1111 1111 1000 = 1000, if signed
1000 = -8

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

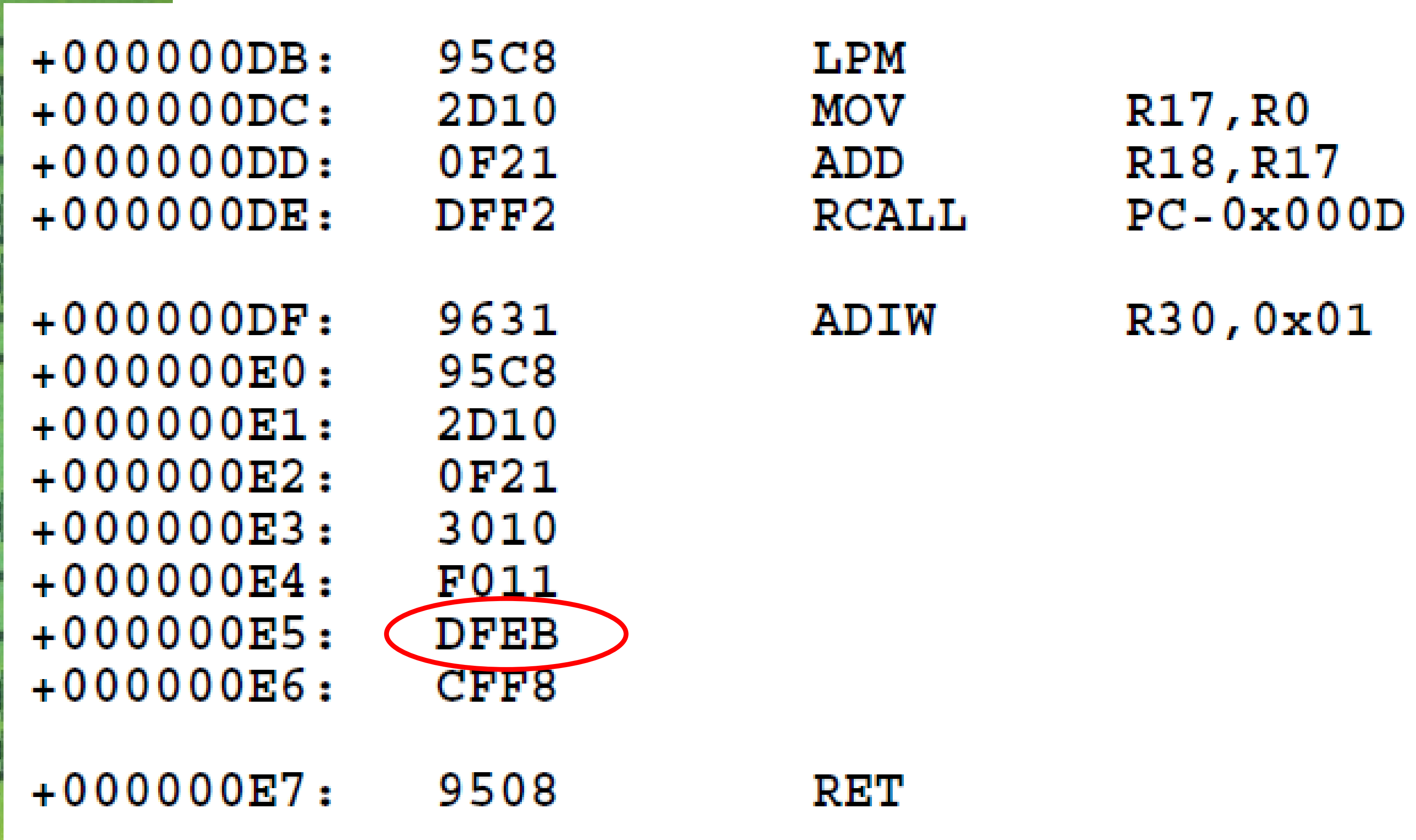
PC <= PC + k + 1 =
PC – 8 + 1 =
= PC - 7

Example:

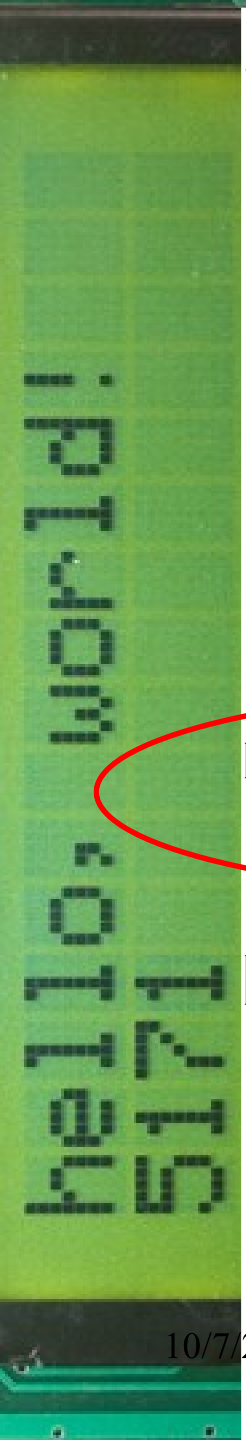
```
        cpi     r16,$42   ; Compare r16 to $42
        brne    error     ; Branch if r16 <> $42
        rjmp    ok        ; Unconditional branch
error:  add     r16,r17   ; Add r17 to r16
        inc     r16       ; Increment r16
ok:     nop               ; Destination for rjmp (do nothing)
```

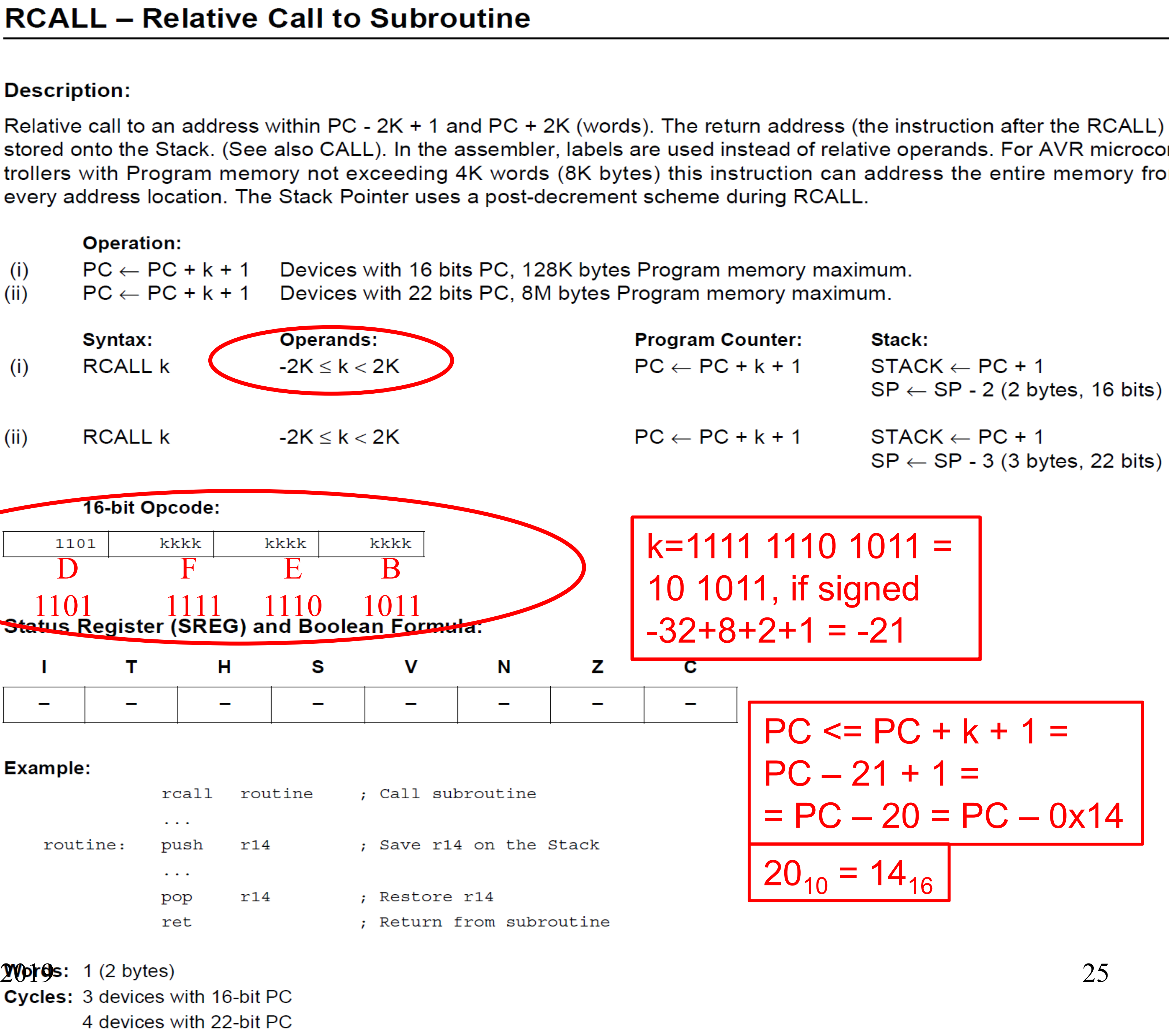
Words: 1 (2 bytes)

Cycles: 2



```
+000000DB:   95C8        LPM
+000000DC:   2D10        MOV      R17,R0
+000000DD:   0F21        ADD      R18,R17
+000000DE:   DFF2        RCALL    PC-0x000D

+000000DF:   9631        ADIW     R30,0x01
+000000E0:   95C8
+000000E1:   2D10
+000000E2:   0F21
+000000E3:   3010
+000000E4:   F011
+000000E5:   DFEB
+000000E6:   CFF8

+000000E7:   9508        RET
```





RCALL – Relative Call to Subroutine

Description:

Relative call to an address within PC - 2K + 1 and PC + 2K (words). The return address (the instruction after the RCALL) stored onto the Stack. (See also CALL). In the assembler, labels are used instead of relative operands. For AVR microco trollers with Program memory not exceeding 4K words (8K bytes) this instruction can address the entire memory fro every address location. The Stack Pointer uses a post-decrement scheme during RCALL.

Operation:

(i) PC ← PC + k + 1 Devices with 16 bits PC, 128K bytes Program memory maximum.
(ii) PC ← PC + k + 1 Devices with 22 bits PC, 8M bytes Program memory maximum.

	Syntax:	Operands:	Program Counter:	Stack:

(i)	RCALL k	$-2K \leq k < 2K$	PC ← PC + k + 1	STACK ← PC + 1
				SP ← SP - 2 (2 bytes, 16 bits)

(ii)	RCALL k	$-2K \leq k < 2K$	PC ← PC + k + 1	STACK ← PC + 1
				SP ← SP - 3 (3 bytes, 22 bits)

16-bit Opcode:

1101	kkkk	kkkk	kkkk
D	F	E	B
1101	1111	1110	1011

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

k=1111 1110 1011 =
10 1011, if signed
-32+8+2+1 = -21

PC <= PC + k + 1 =
PC – 21 + 1 =
= PC – 20 = PC – 0x14

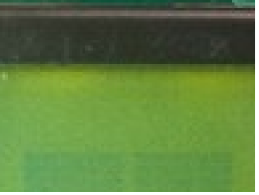
$20_{10} = 14_{16}$

Example:

```
        rcall   routine     ; Call subroutine
        ...
routine: push    r14         ; Save r14 on the Stack
        ...
        pop     r14         ; Restore r14
        ret                 ; Return from subroutine
```

Words: 1 (2 bytes)

Cycles: 3 devices with 16-bit PC
4 devices with 22-bit PC



xplain the function of the complete program.

```
0DB: print_string
0DB:   95C8        LPM                       Load program memory
0DC:   2D10        MOV      R17,R0           Copy register
0DD:   0F21        ADD      R18,R17          Add without carry
0DE:   DFF2        RCALL    PC-0x000D        Relative call subroutine
0DF: next_character
0DF:   9631        ADIW     R30,0x01         Add immediate to word
0E0:   95C8        LPM                       Load program memory
0E1:   2D10        MOV      R17,R0           Copy register
0E2:   0F21        ADD      R18,R17          Add without carry
0E3:   3010        CPI      R17,0x00         Compare with immediate
0E4:   F011        BREQ     PC+0x03          Branch if equal
0E5:   DFEB        RCALL    PC-0x0014        Relative call subroutine
0E6:   CFF8        RJMP     PC-0x0007        Relative jump
0E7: slutone
0E7:   9508        RET                       Subroutine return
```

Example 2:

Analyze the program below. RX2Int is an interrupt routine, which will be executed each time a new character is received on serial port, RS232. (USART1)
After reading the character in UDR1, the routine will call the subroutine **check**.

Make a flowchart for the subroutine **check**, which explains the function. 5p

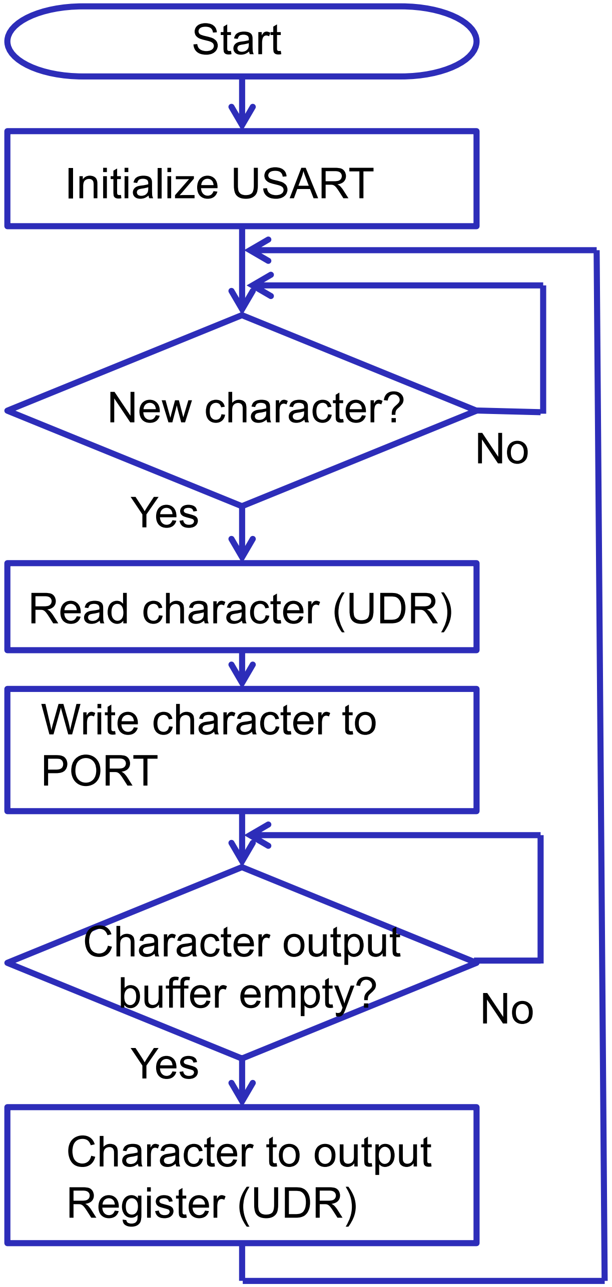
b) Explain with words what the subroutine check will do.
An ASCII-code table is enclosed on next page. 5p

```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;    1ED022, Computer Technology I
;    Date: 2013-02-16
;    Anders Haggren
;    Function:     ?
;    Exam example, task 5
;
;    RX2Int - Interrupt routine for receiving
;    characters from USART1 data register UDR1
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

RX2Int:
     push Temp              ; Save Temp on stack
     in Temp, SREG          ; Save SREG on stack
     push Temp

     lds Char, UDR1         ; Read character from receive buffer

     rcall check            ; Call subroutine check

     com Char               ; Invert bits
     out portb, Char        ; Write value to PortB

     pop Temp
     out SREG, Temp         ; Restore SREG
     pop Temp               ; Restore Temp
     reti
```

Flowcharts!

NOTE!
Written report of laboratory work to be done for all tasks of each group.

The description shall contain program code, flow chart and description of program code. The source code should be well commented. Each item in each lab should be reported. The reports will be compared in the antiplagiarism system Urkund. Note that copying of code, text or other material between groups or from other sources is considered cheating and will be reported to the Disciplinary Committee. Keep in mind that the report takes the time to write - do not wait to start writing!



To be continued…