به نام خدا

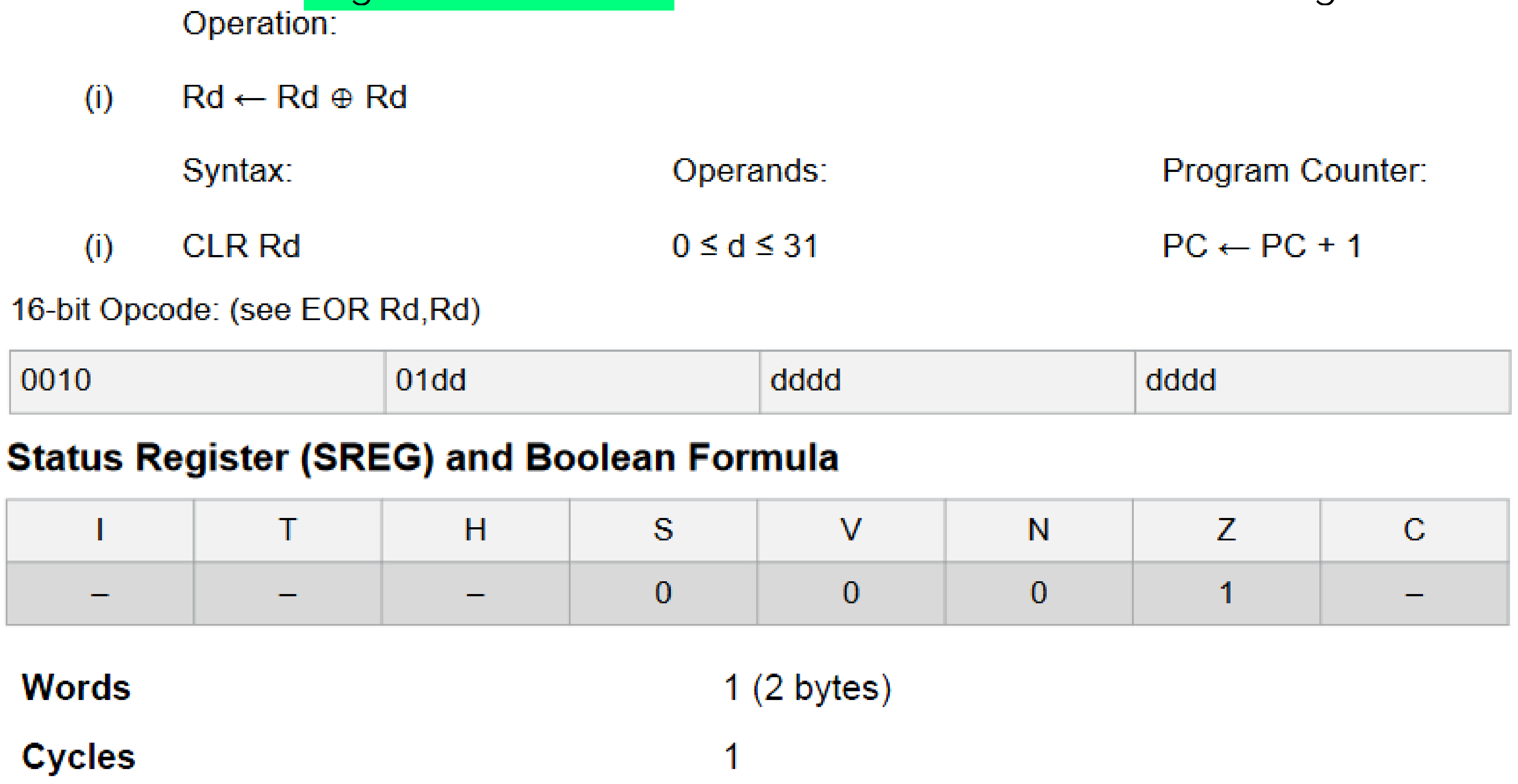
آشنایی با زبان اسمبلی AVR

دستورات پایه (2)

Dr. Aref Karimiafshar
A.karimiafshar@ec.iut.ac.ir

CLR – Clear Register

- Clears a register. This instruction performs an Exclusive OR between a register and itself. This will clear all bits in the register.



Operation:

(i) $Rd \leftarrow Rd \oplus Rd$

	Syntax:	Operands:	Program Counter:
(i)	CLR Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16-bit Opcode: (see EOR Rd,Rd)

0010	01dd	dddd	dddd

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	0	0	0	1	–

Words	1 (2 bytes)
Cycles	1

INC – Increment

- Adds one -1- to the contents of register Rd and places the result in the destination register Rd. The C Flag in SREG is not affected by the operation.

Operation:

(i) Rd ← Rd + 1

	Syntax:	Operands:	Program Counter:
(i)	INC Rd	$0 \le d \le 31$	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0011

Status Register and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	–

Words	1 (2 bytes)
Cycles	1

INC – Increment

S $N \oplus V$, for signed tests.

V $R7 \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$

Set if two's complement overflow resulted from the operation; cleared otherwise. Two's complement overflow occurs if and only if Rd was $7F before the operation.

N R7

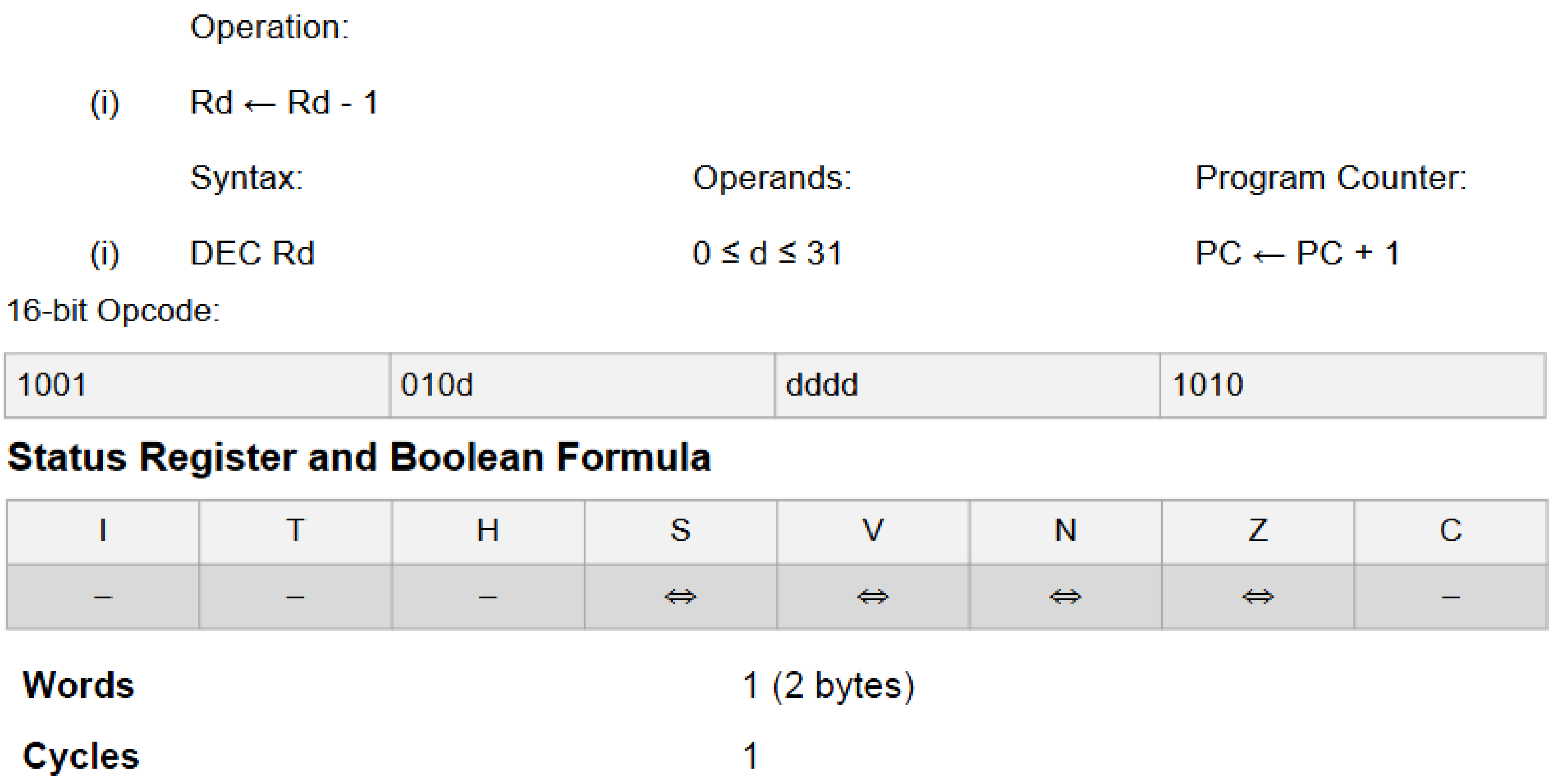
Set if MSB of the result is set; cleared otherwise.

Z $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$

Set if the result is $00; cleared otherwise.

DEC – Decrement

- Subtracts one -1- from the contents of register Rd and places the result in the destination register Rd. The C Flag in SREG is not affected by the operation.



Operation:

(i) Rd ← Rd - 1

	Syntax:	Operands:	Program Counter:
(i)	DEC Rd	$0 \le d \le 31$	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	1010

Status Register and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	–

Words	1 (2 bytes)
Cycles	1

DEC – Decrement

S $N \oplus V$, for signed tests.

V $\overline{R7} \cdot R6 \cdot R5 \cdot R4 \cdot R3 \cdot R2 \cdot R1 \cdot R0$

Set if two's complement overflow resulted from the operation; cleared otherwise. Two's complement overflow occurs if and only if Rd was $80 before the operation.

N R7

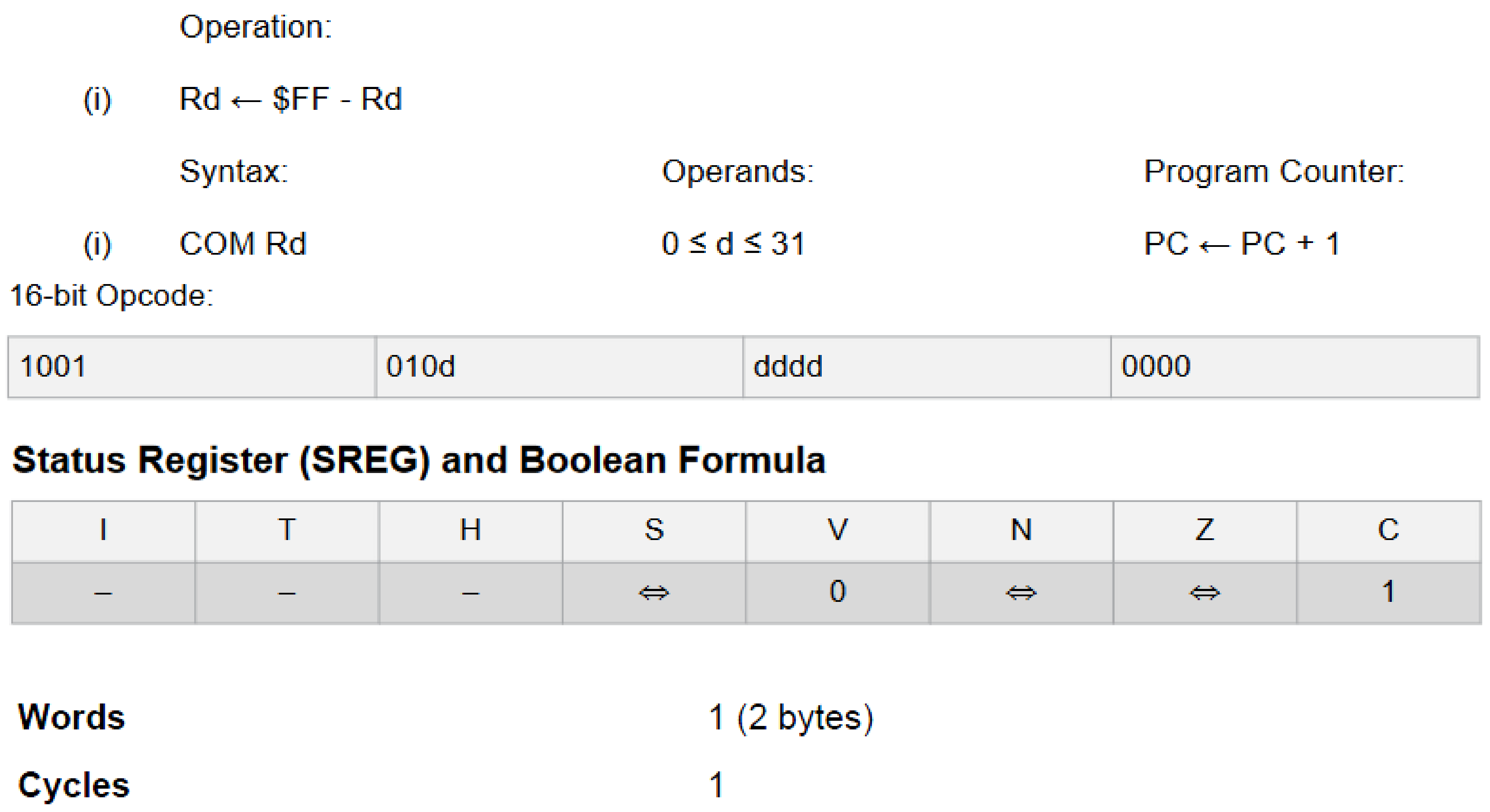
Set if MSB of the result is set; cleared otherwise.

Z $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$

Set if the result is $00; cleared otherwise.

COM – One's Complement

- This instruction performs a One's Complement of register Rd.



Operation:

(i) Rd ← \$FF - Rd

	Syntax:	Operands:	Program Counter:
(i)	COM Rd	$0 \le d \le 31$	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0000

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	1

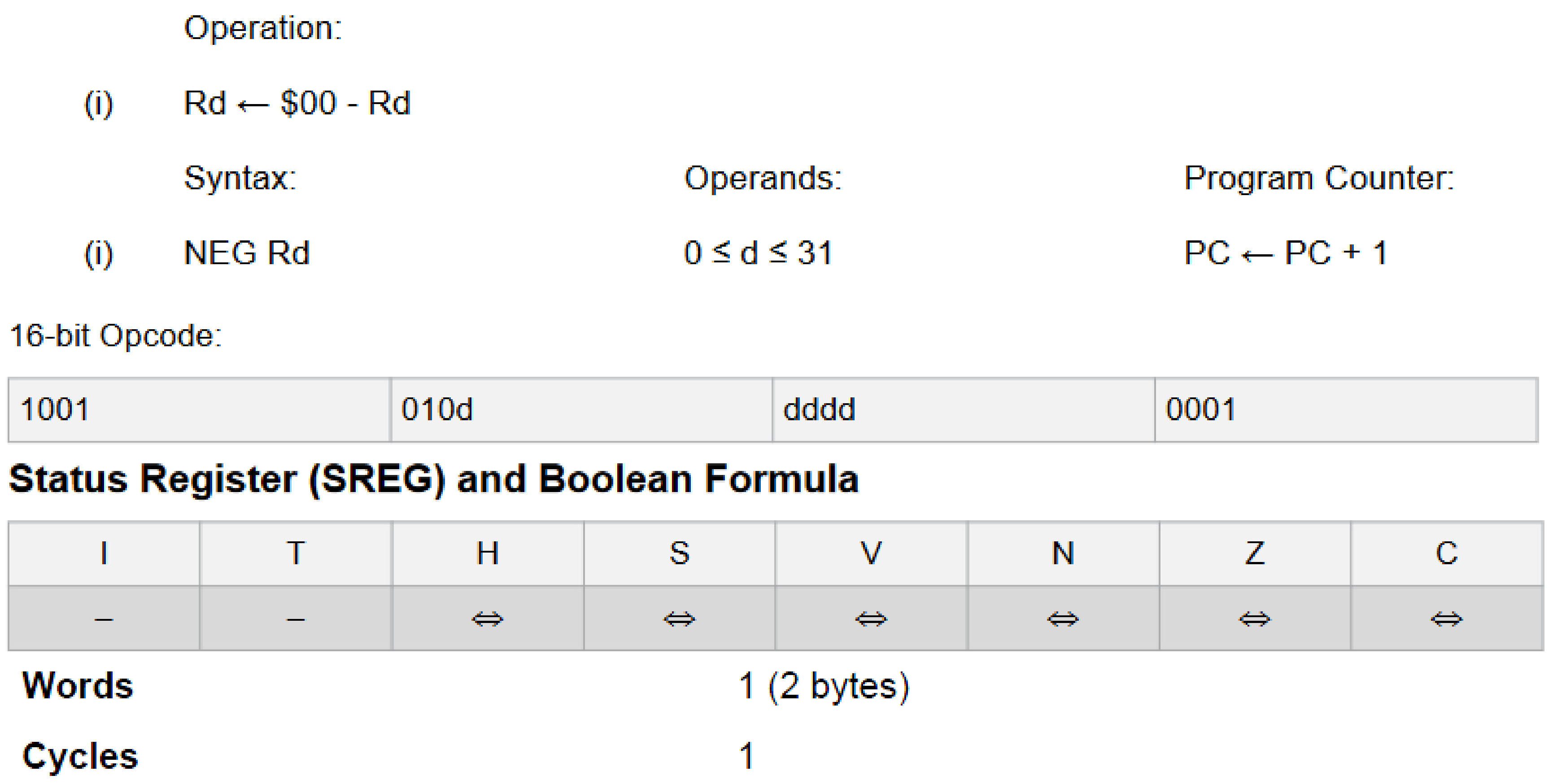
Words	1 (2 bytes)
Cycles	1

COM – One’s Complement

S	$N \oplus V$, for signed tests.
V	0
	Cleared.
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	1
	Set.

NEG – Two's Complement

- Replaces the contents of register Rd with its two's complement; the value $80 is left unchanged.



Operation:

(i) Rd ← $00 - Rd

	Syntax:	Operands:	Program Counter:
(i)	NEG Rd	0 ≤ d ≤ 31	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0001

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

Words	1 (2 bytes)
Cycles	1

NEG – Two’s Complement

H $P3 + \overline{Rd3}$

Set if there was a borrow from bit 3; cleared otherwise.

S $N \oplus V$, for signed tests.

V $R7 \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$

Set if there is a two’s complement overflow from the implied subtraction from zero; cleared otherwise. A two’s complement overflow will occur if and only if the contents of the Register after operation (Result) is \$80.

N $R7$

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

Z $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$

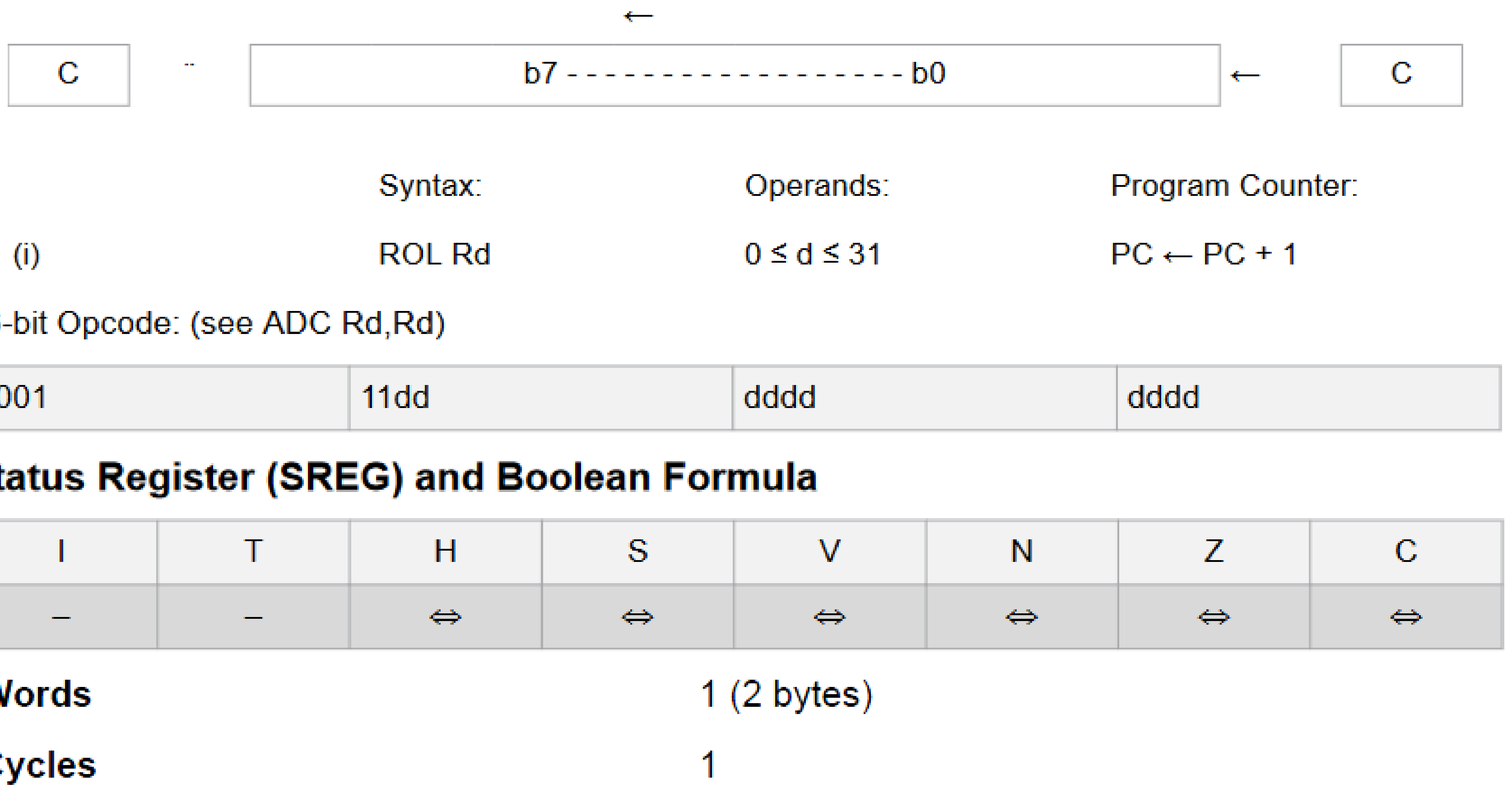
Set if the result is \$00; cleared otherwise.

C $R7 + R6 + R5 + R4 + R3 + R2 + R1 + R0$

Set if there is a borrow in the implied subtraction from zero; cleared otherwise. The C Flag will be set in all cases except when the contents of Register after operation is \$00.

ROL – Rotate Left trough Carry

- Shifts all bits in Rd one place to the left. The C Flag is shifted into bit 0 of Rd. Bit 7 is shifted into the C Flag.



C ← b7 - - - - - - - - - - - - - - - - - - b0 ← C

	Syntax:	Operands:	Program Counter:
(i)	ROL Rd	$0 \le d \le 31$	PC ← PC + 1

16-bit Opcode: (see ADC Rd,Rd)

0001	11dd	dddd	dddd

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

Words	1 (2 bytes)
Cycles	1

ROL – Rotate Left trough Carry

H	Rd3
S	$N \oplus V$, for signed tests.
V	$N \oplus C$, for N and C after the shift.
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 Set if, before the shift, the MSB of Rd was set; cleared otherwise.

ROR – Rotate Right through Carry

- Shifts all bits in Rd one place to the right. The C Flag is shifted into bit 7 of Rd. Bit 0 is shifted into the C Flag.

$$\begin{tikzcd}
\boxed{C} \arrow[r] & \boxed{b7 \text{ - - - - - - - - - - - - - - - - - - - } b0} \arrow[r] & \boxed{C}
\end{tikzcd}$$

	Syntax:	Operands:	Program Counter:
(i)	ROR Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16-bit Opcode:

1001	010d	dddd	0111

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	⇔

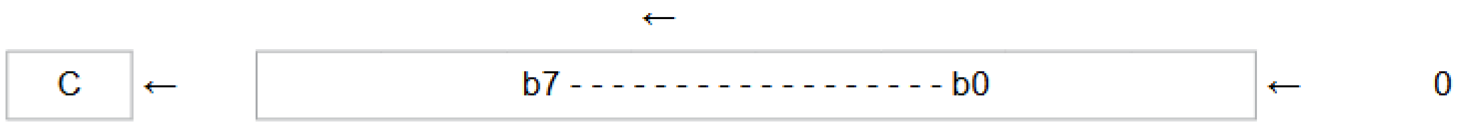
Words	1 (2 bytes)
Cycles	1

ROR – Rotate Right through Carry

S	$N \oplus V$, for signed tests.
V	$N \oplus C$, for N and C after the shift.
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	Rd0 Set if, before the shift, the LSB of Rd was set; cleared otherwise.

LSL – Logical Shift Left

- Shifts all bits in Rd one place to the left. Bit 0 is cleared. Bit 7 is loaded into the C Flag of the SREG. This operation effectively multiplies signed and unsigned values by two.



	Syntax:	Operands:	Program Counter:
(i)	LSL Rd	$0 \le d \le 31$	PC ← PC + 1

16-bit Opcode: (see ADD Rd,Rd)

0000	11dd	dddd	dddd

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

Words	1 (2 bytes)
Cycles	1

LSL – Logical Shift Left

H	Rd3
S	$N \oplus V$, for signed tests.
V	$N \oplus C$, for N and C after the shift.
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
	Set if, before the shift, the MSB of Rd was set; cleared otherwise.

LSR – Logical Shift Right

- Shifts all bits in Rd one place to the right. Bit 7 is cleared. Bit 0 is loaded into the C Flag of the SREG. This operation effectively divides an unsigned value by two. The C Flag can be used to round the result.

→

$$\begin{tikzcd} 0 \arrow[r] & \boxed{b7 - - - - - - - - - - - - - - - - - - b0} \arrow[r] & \boxed{C} \end{tikzcd}$$

	Syntax:	Operands:	Program Counter:
(i)	LSR Rd	$0 \le d \le 31$	PC ← PC + 1

16-bit Opcode:

1001	010d	dddd	0110

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	0	⇔	⇔

Words	1 (2 bytes)
Cycles	1

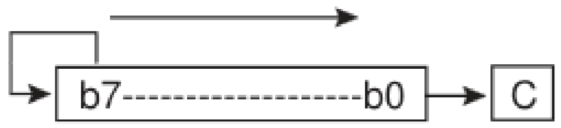
LSR – Logical Shift Right

S	$N \oplus V$, for signed tests.
V	$N \oplus C$, for N and C after the shift.
N	0
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	Rd0
	Set if, before the shift, the LSB of Rd was set; cleared otherwise.

ASR – Arithmetic Shift Right

- Shifts all bits in Rd one place to the right. Bit 7 is held constant. Bit 0 is loaded into the C Flag of the SREG. This operation effectively divides a signed value by two without changing its sign. The Carry Flag can be used to round the result.

(i)



	Syntax:	Operands:	Program Counter:
(i)	ASR Rd	$0 \le d \le 31$	$PC \leftarrow PC + 1$

16-bit Opcode:

1001	010d	dddd	0101

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	⇔

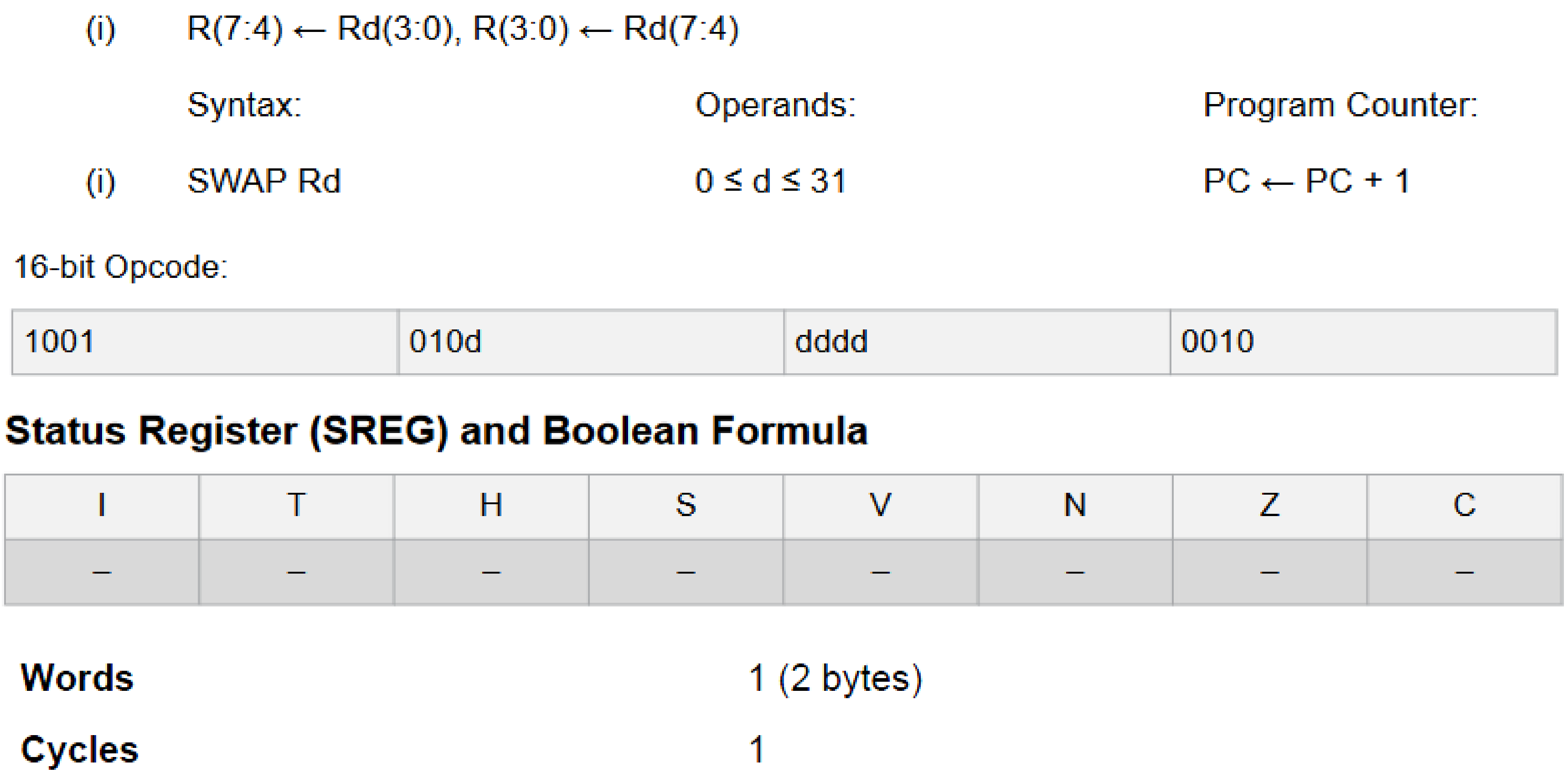
Words	1 (2 bytes)
Cycles	1

ASR – Arithmetic Shift Right

S	$N \oplus V$, for signed tests.
V	$N \oplus C$, for N and C after the shift.
N	R7 Set if MSB of the result is set; cleared otherwise.
Z	$\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$ Set if the result is \$00; cleared otherwise.
C	Rd0 Set if, before the shift, the LSB of Rd was set; cleared otherwise.

SWAP – Swap Nibbles

- Swaps high and low nibbles in a register.



(i) R(7:4) ← Rd(3:0), R(3:0) ← Rd(7:4)

	Syntax:	Operands:	Program Counter:
(i)	SWAP Rd	$0 \le d \le 31$	PC ← PC + 1

16-bit Opcode:

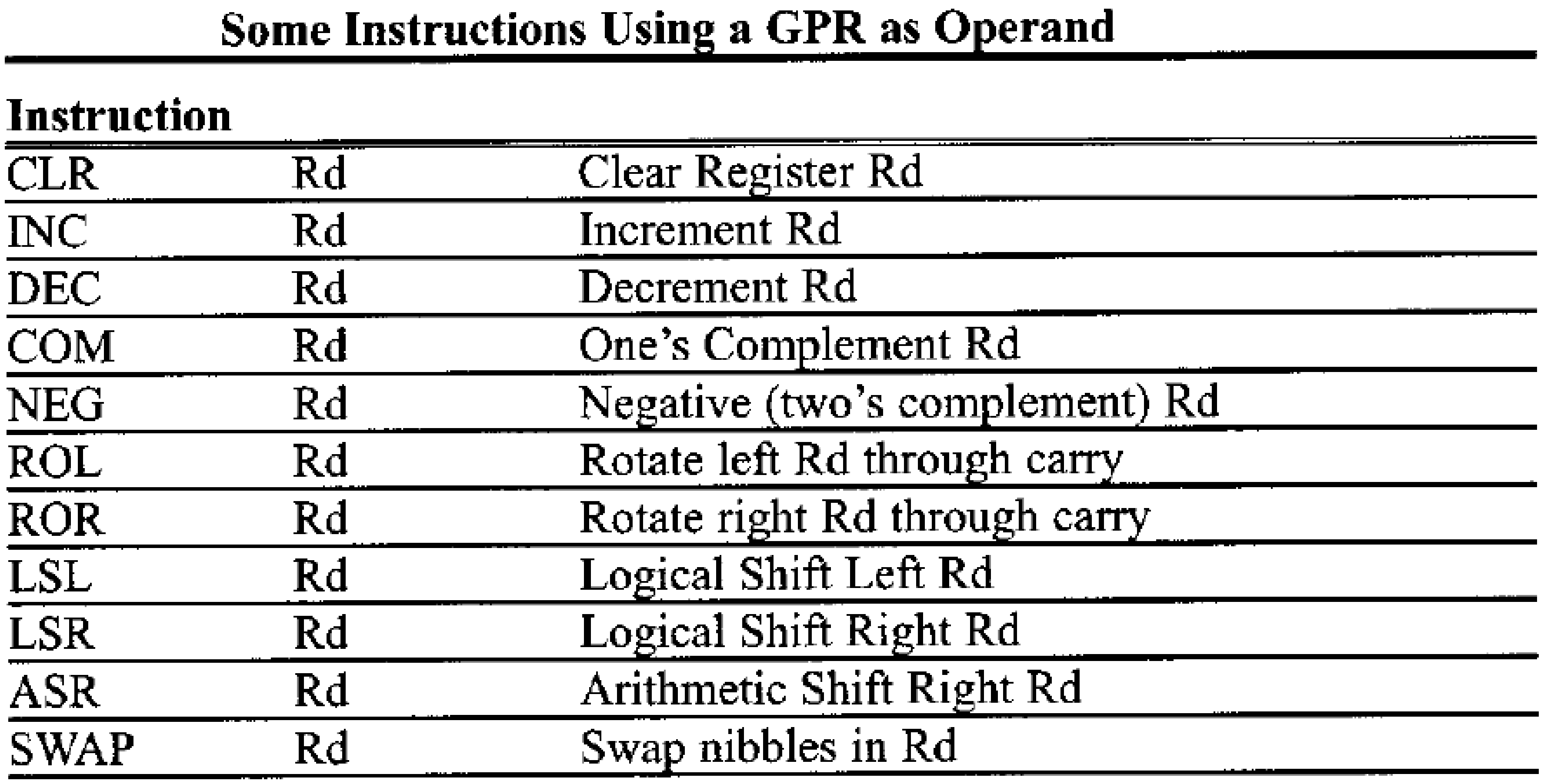
1001	010d	dddd	0010

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1

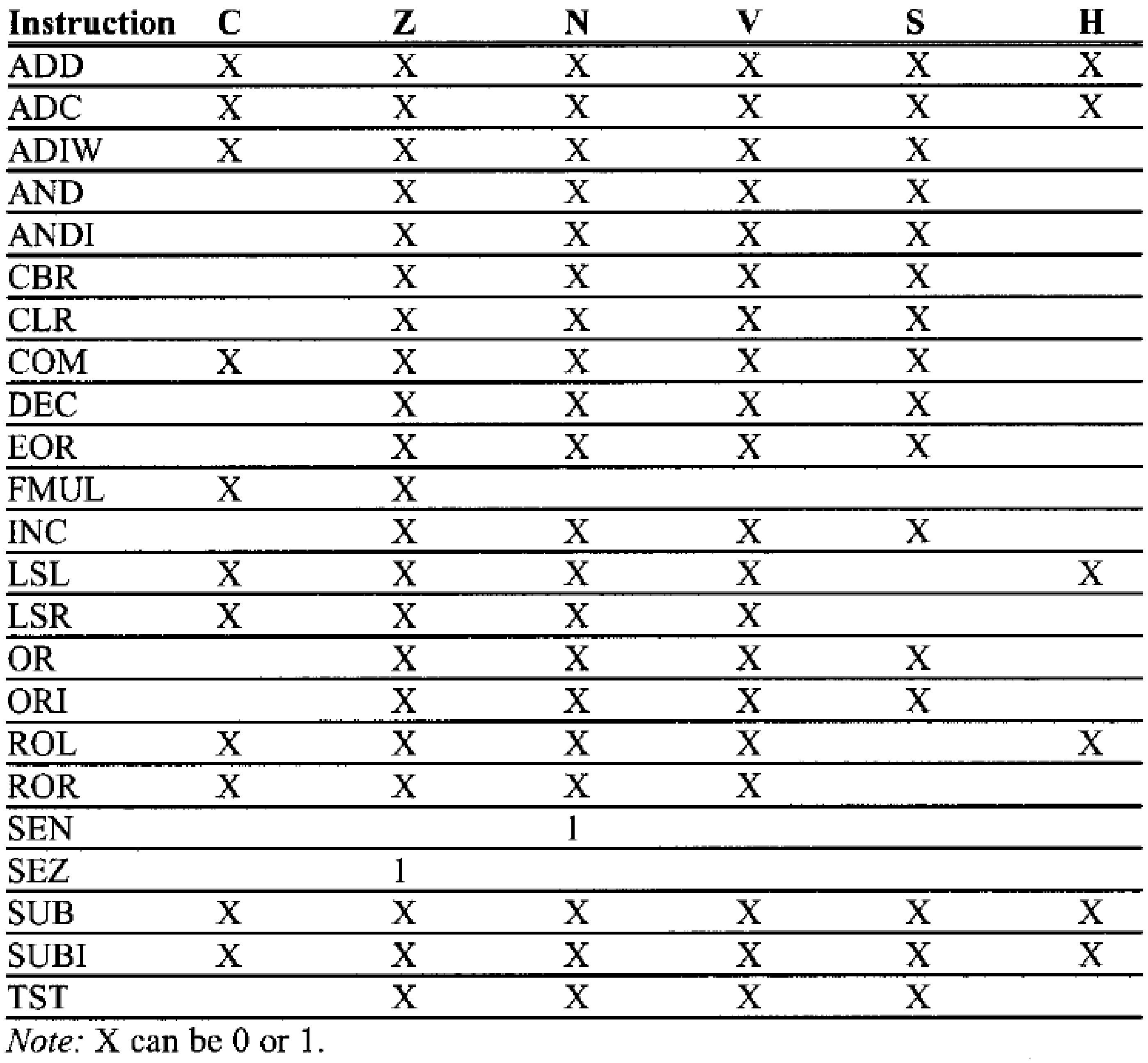
Summary



Some Instructions Using a GPR as Operand

Instruction		
CLR	Rd	Clear Register Rd
INC	Rd	Increment Rd
DEC	Rd	Decrement Rd
COM	Rd	One's Complement Rd
NEG	Rd	Negative (two's complement) Rd
ROL	Rd	Rotate left Rd through carry
ROR	Rd	Rotate right Rd through carry
LSL	Rd	Logical Shift Left Rd
LSR	Rd	Logical Shift Right Rd
ASR	Rd	Arithmetic Shift Right Rd
SWAP	Rd	Swap nibbles in Rd

Instructions that affect Flag Bits

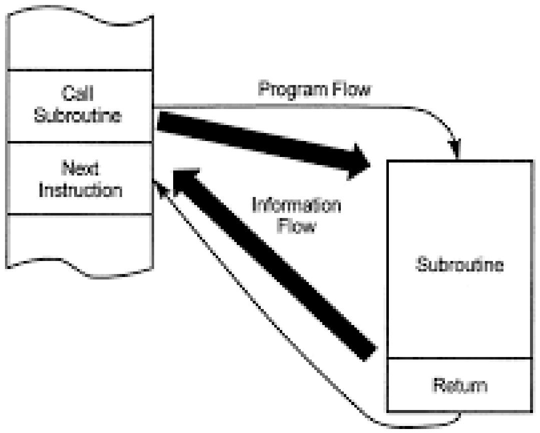


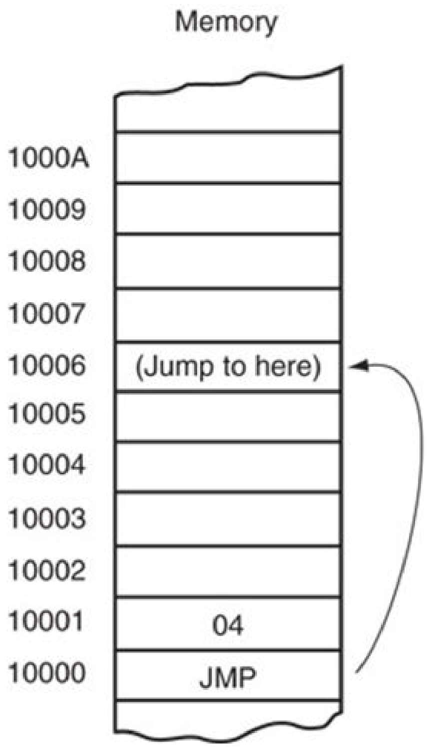
Instruction	C	Z	N	V	S	H
ADD	X	X	X	X	X	X
ADC	X	X	X	X	X	X
ADIW	X	X	X	X	X	
AND		X	X	X	X	
ANDI		X	X	X	X	
CBR		X	X	X	X	
CLR		X	X	X	X	
COM	X	X	X	X	X	
DEC		X	X	X	X	
EOR		X	X	X	X	
FMUL	X	X				
INC		X	X	X	X	
LSL	X	X	X	X		X
LSR	X	X	X	X		
OR		X	X	X	X	
ORI		X	X	X	X	
ROL	X	X	X	X		X
ROR	X	X	X	X		
SEN			1			
SEZ		1				
SUB	X	X	X	X	X	X
SUBI	X	X	X	X	X	X
TST		X	X	X	X	

Note: X can be 0 or 1.

دستورات پرش

- In the sequence of instruction to be executed, it is often necessary to transfer program control to a different location.
- Instruction in AVR to achieve this:
  - Branches (JUMP)
  - CALL





حلقه

- Repeating a sequence of instructions or an operation a certain number of times.
  1. Repeat the instructions over and over!

```
LDI  R16,0              ;R16 = 0
LDI  R17,3              ;R17 = 3
ADD  R16,R17            ;add value 3 to R16 (R16 = 0x03)
ADD  R16,R17            ;add value 3 to R16 (R16 = 0x06)
ADD  R16,R17            ;add value 3 to R16 (R16 = 0x09)
ADD  R16,R17            ;add value 3 to R16 (R16 = 0x0C)
ADD  R16,R17            ;add value 3 to R16 (R16 = 0x0F)
ADD  R16,R17            ;add value 3 to R16 (R16 = 0x12)
```

Too much code space would be needed!

- A much better way is to use a *LOOP*

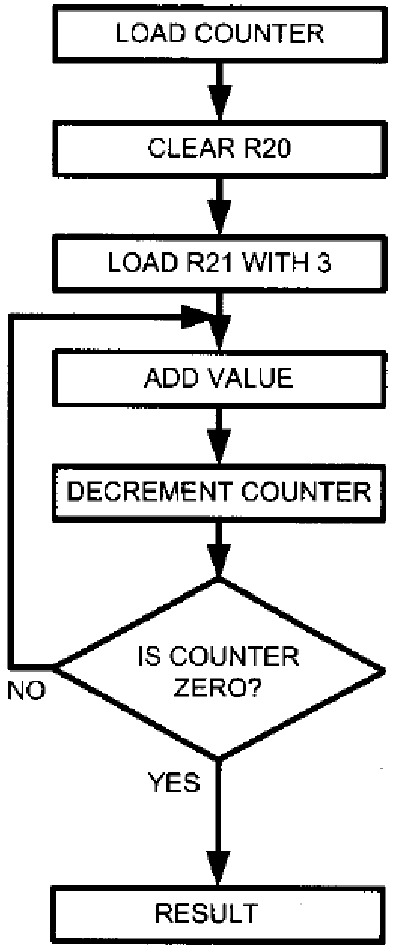
حلقه

- Structure

```
BACK: .........     ;start of the loop
      .........     ;body of the loop
      .........     ;body of the loop
      DEC Rn        ;decrement Rn, Z = 1 if Rn = 0
      BRNE BACK     ;branch to BACK if Z = 0
```

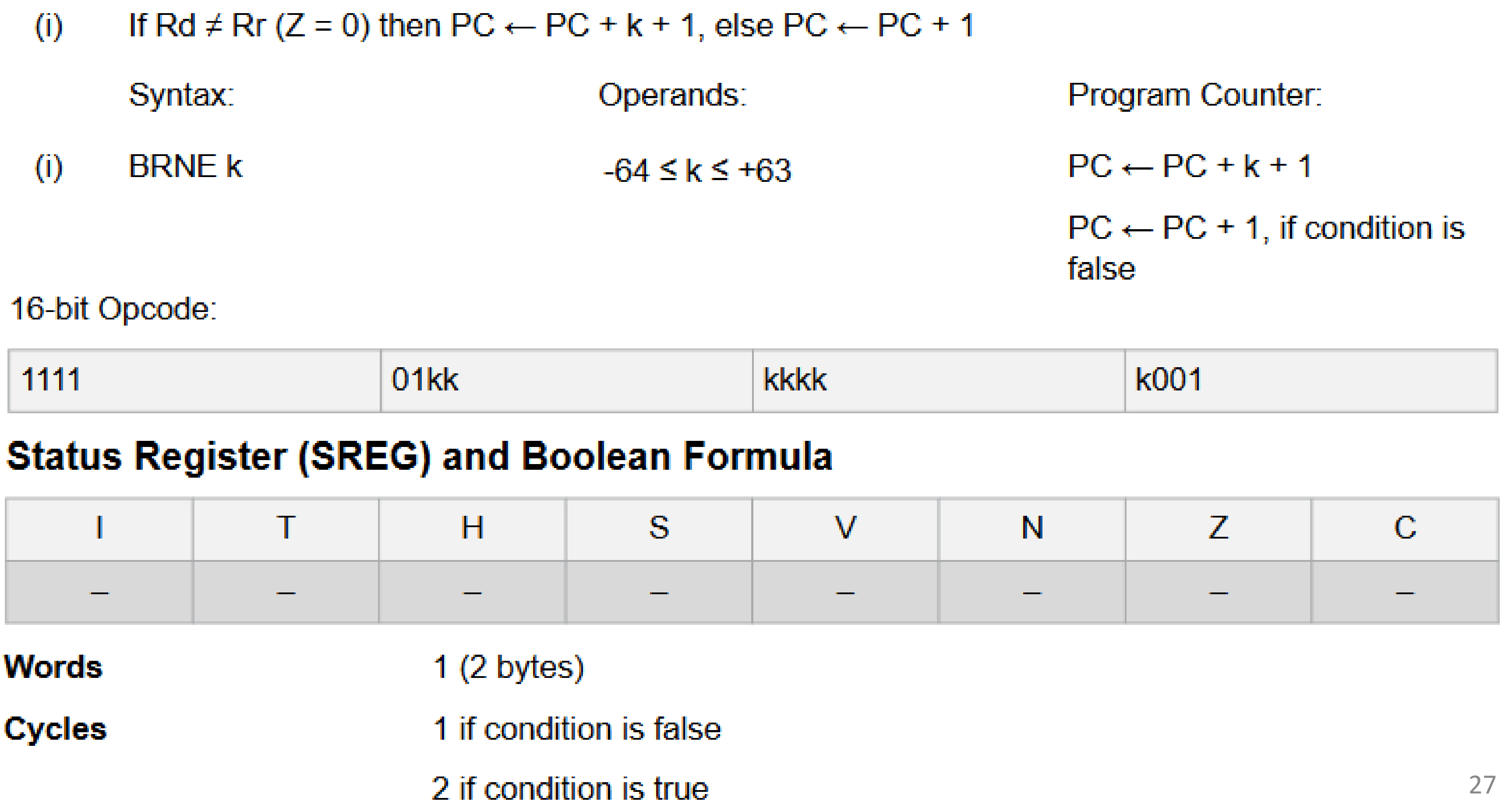
- Example

```
      LDI   R16, 10       ;R16 = 10 (decimal) for counter
      LDI   R20, 0        ;R20 = 0
      LDI   R21, 3        ;R21 = 3
AGAIN:ADD   R20, R21      ;add 03 to R20 (R20 = sum)
      DEC   R16           ;decrement R16 (counter)
      BRNE  AGAIN         ;repeat until COUNT = 0
```



BRNE – Branch if Not Equal

- Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is cleared.



(i) If Rd ≠ Rr (Z = 0) then PC ← PC + k + 1, else PC ← PC + 1

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

16-bit Opcode:

1111	01kk	kkkk	k001

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

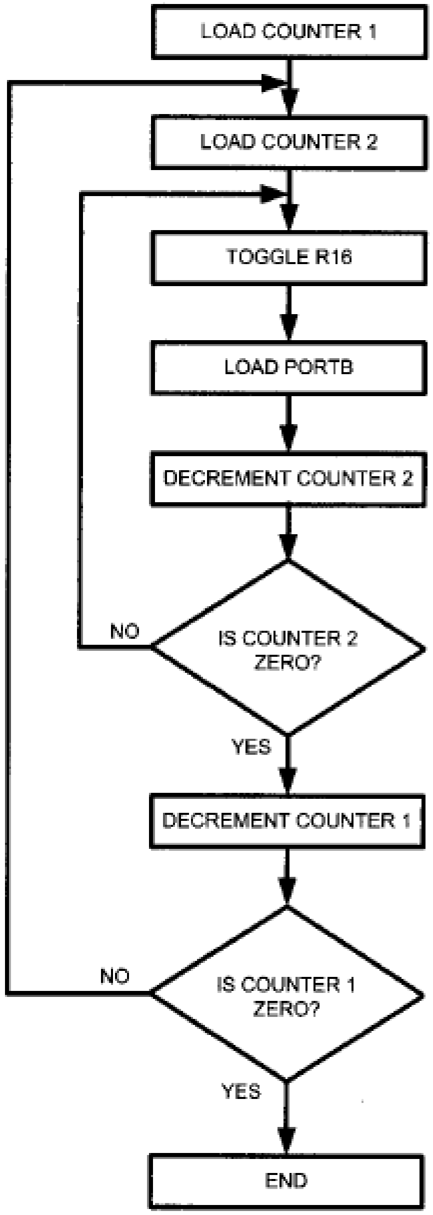
Words	1 (2 bytes)
Cycles	1 if condition is false
	2 if condition is true

حلقه‌های تو در تو

- Maximum number of times a loop can be repeated
  - Limited

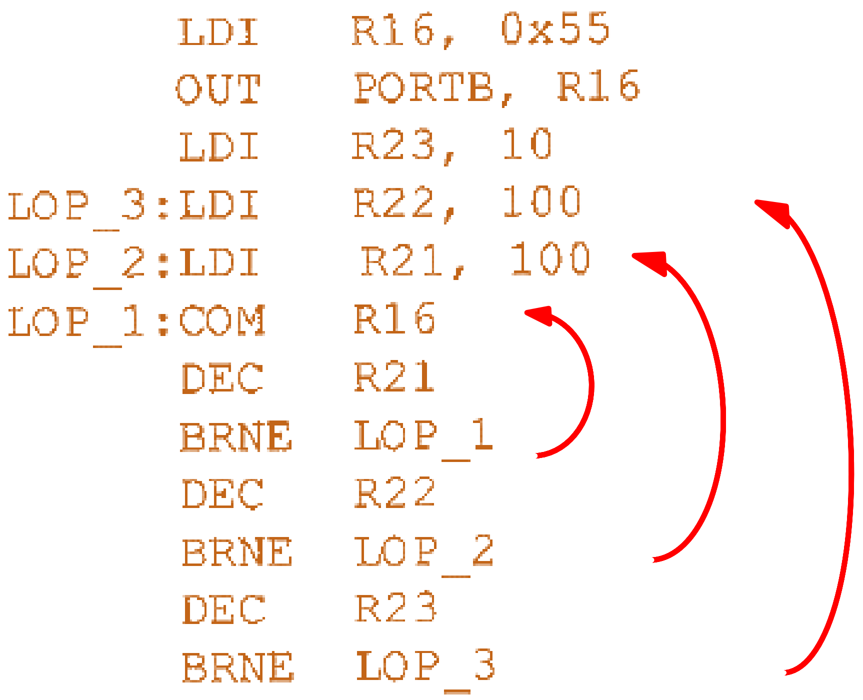
- Example

```
      LDI   R16, 0x55     ;R16 = 0x55
      OUT   PORTB, R16    ;PORTB = 0x55
      LDI   R20, 10       ;load 10 into R20 (outer loop count)
LOP_1:LDI   R21, 70       ;load 70 into R21 (inner loop count)
LOP_2:COM   R16           ;complement R16
      OUT   PORTB, R16    ;load PORTB SFR with the complemented value
      DEC   R21           ;dec R21 (inner loop)
      BRNE  LOP_2         ;repeat it 70 times
      DEC   R20           ;dec R20 (outer loop)
      BRNE  LOP_1         ;repeat it 10 times
```



حلقه‌های تو در تو

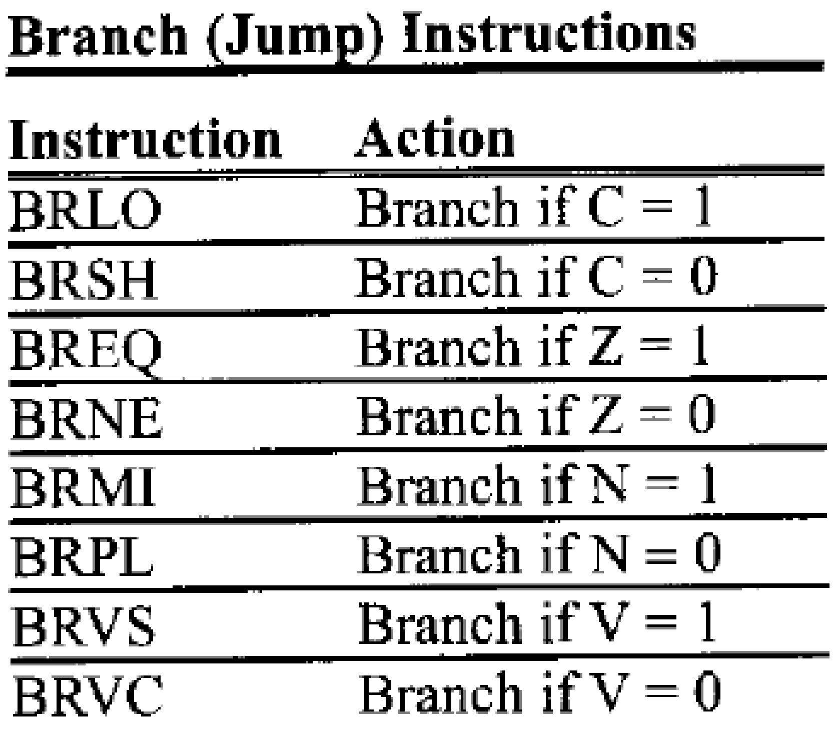
- Number of times > 255* 255 (=65,025)
  - Three nested loop
- Example



```
        LDI    R16, 0x55
        OUT    PORTB, R16
        LDI    R23, 10
LOP_3:LDI    R22, 100
LOP_2:LDI     R21, 100
LOP_1:COM    R16
        DEC    R21
        BRNE   LOP_1
        DEC    R22
        BRNE   LOP_2
        DEC    R23
        BRNE   LOP_3
```

دستورات پرش

- AVR conditional branch (jump) instructions

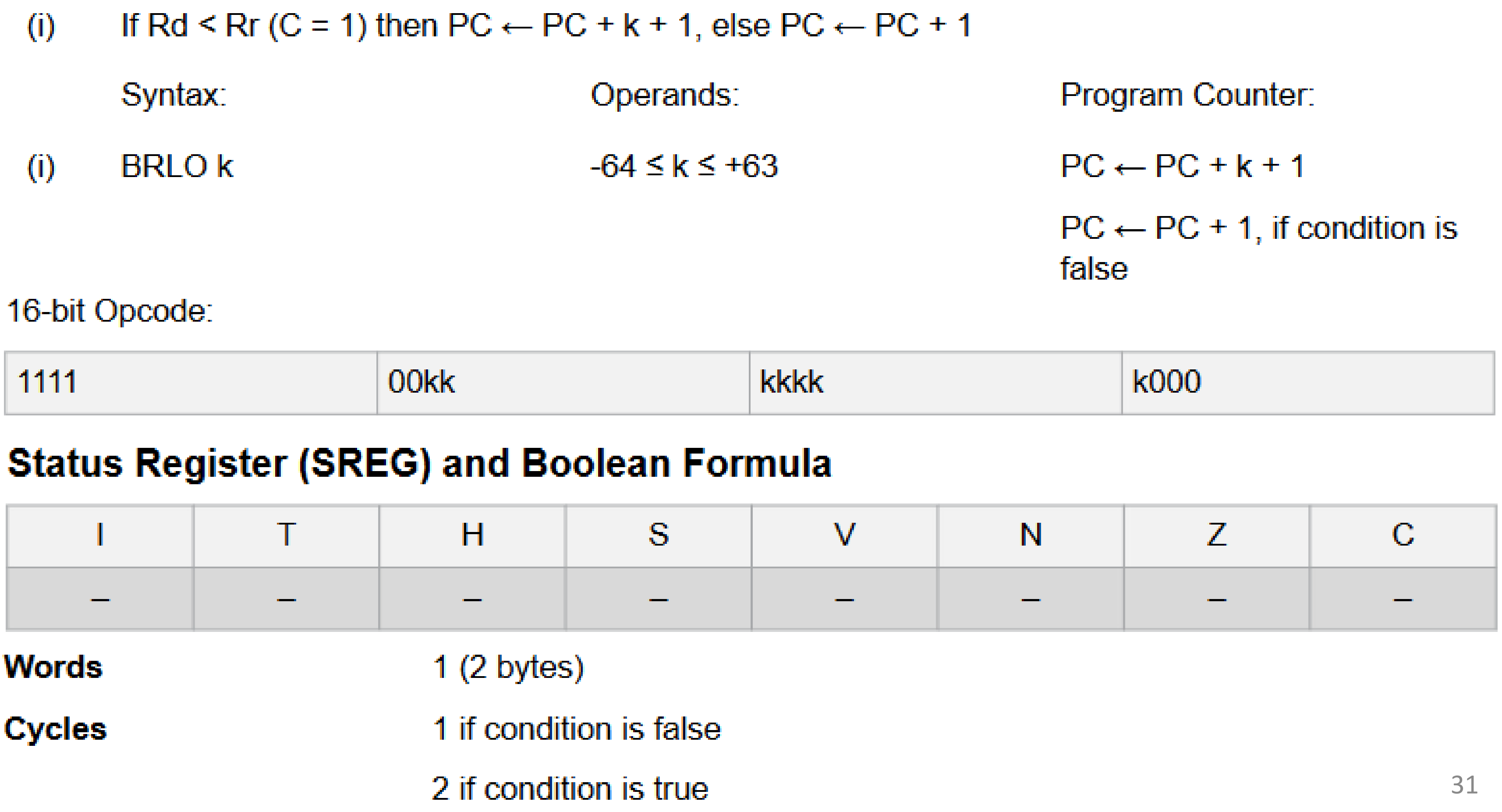


Branch (Jump) Instructions

Instruction	Action
BRLO	Branch if C = 1
BRSH	Branch if C = 0
BREQ	Branch if Z = 1
BRNE	Branch if Z = 0
BRMI	Branch if N = 1
BRPL	Branch if N = 0
BRVS	Branch if V = 1
BRVC	Branch if V = 0

BRLO – Branch if Lower (Unsigned)

- Conditional relative branch. Tests the Carry Flag (C) and branches relatively to PC if C is set.



(i) If $Rd < Rr$ ($C = 1$) then $PC \leftarrow PC + k + 1$, else $PC \leftarrow PC + 1$

	Syntax:	Operands:	Program Counter:
(i)	BRLO k	$-64 \le k \le +63$	$PC \leftarrow PC + k + 1$
			$PC \leftarrow PC + 1$, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k000

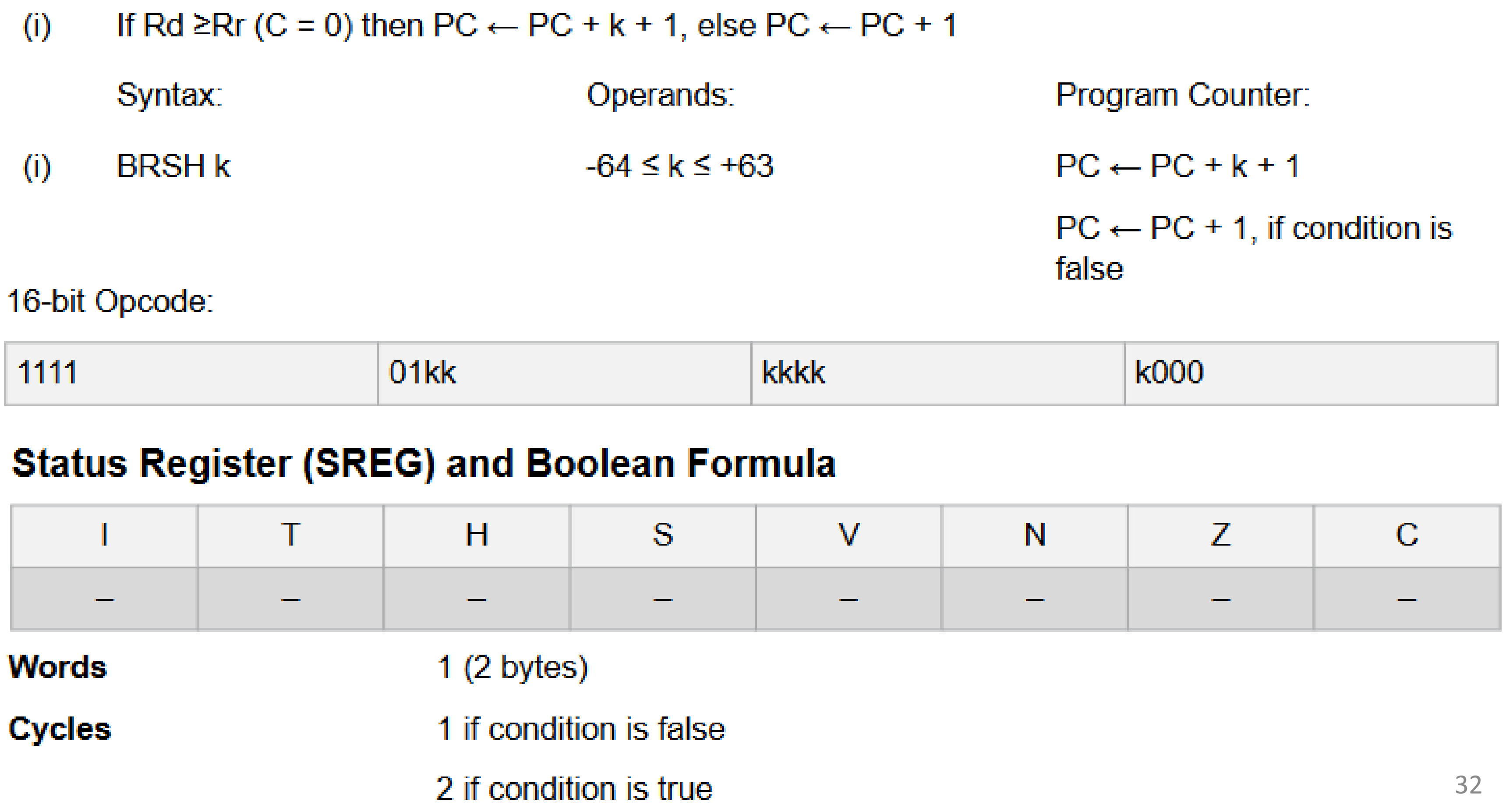
Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1 if condition is false
	2 if condition is true

BRSH – Branch if Same or Higher (Unsigned)

- Conditional relative branch. Tests the Carry Flag (C) and branches relatively to PC if C is cleared.



(i) If Rd ≥Rr (C = 0) then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRSH k	$-64 \le k \le +63$	PC ← PC + k + 1
			PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	k000

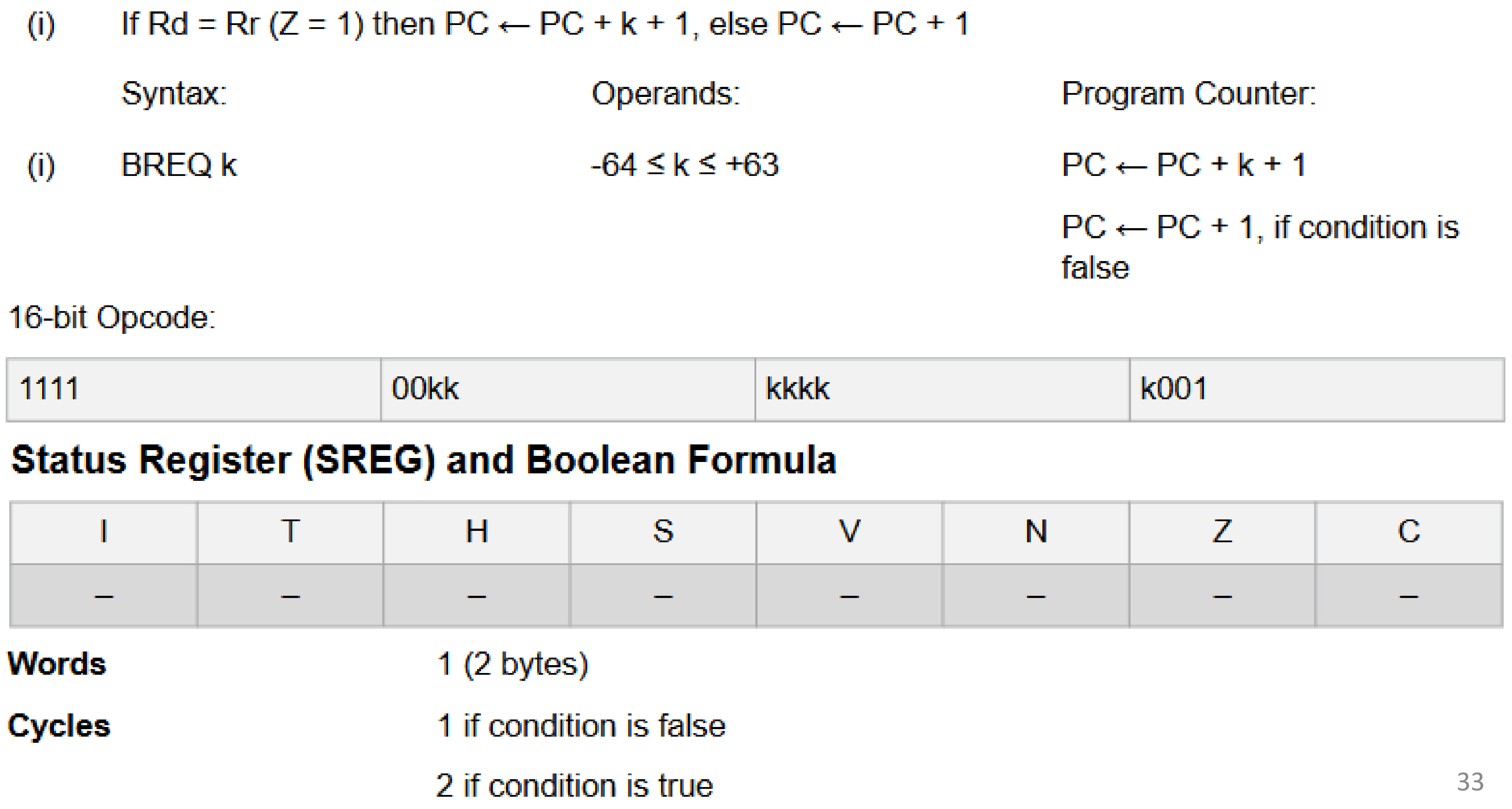
Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1 if condition is false
	2 if condition is true

BREQ – Branch if Equal

- Conditional relative branch. Tests the Zero Flag (Z) and branches relatively to PC if Z is set.



(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

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1 if condition is false
	2 if condition is true

BRMI – Branch if Minus

- Conditional relative branch. Tests the Negative Flag (N) and branches relatively to PC if N is set.

(i) If N = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRMI k	$-64 \le k \le +63$	PC ← PC + k + 1
			PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	k010

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1 if condition is false
	2 if condition is true

BRPL – Branch if Plus

- Conditional relative branch. Tests the Negative Flag (N) and branches relatively to PC if N is cleared.

(i) If N = 0 then PC ← PC + k + 1, else PC ← PC + 1

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

16-bit Opcode:

1111	01kk	kkkk	k010

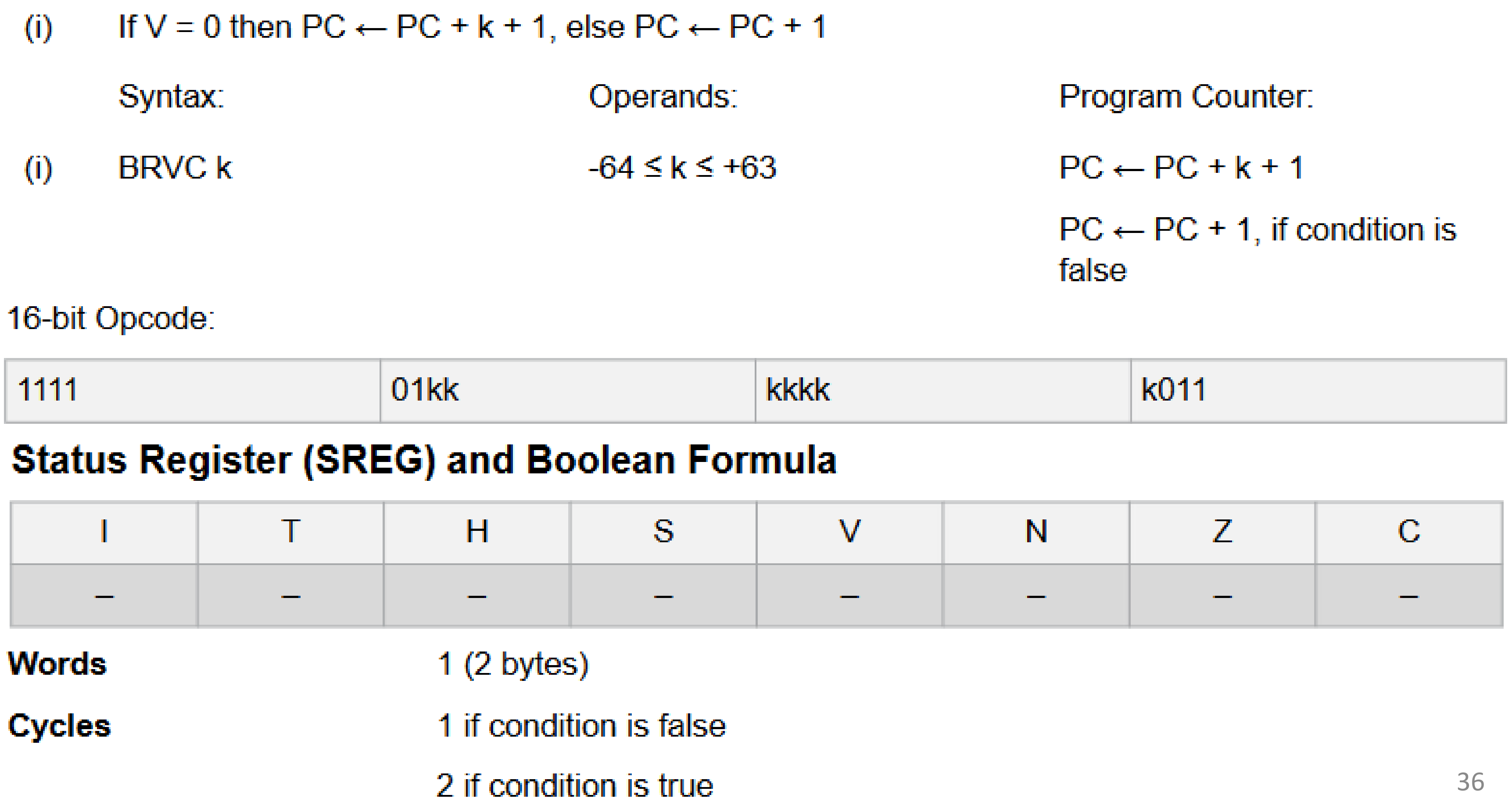
Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1 if condition is false
	2 if condition is true

BRVC – Branch if Overflow Cleared

- Conditional relative branch. Tests the Overflow Flag (V) and branches relatively to PC if V is cleared.



(i) If V = 0 then PC ← PC + k + 1, else PC ← PC + 1

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

16-bit Opcode:

1111	01kk	kkkk	k011

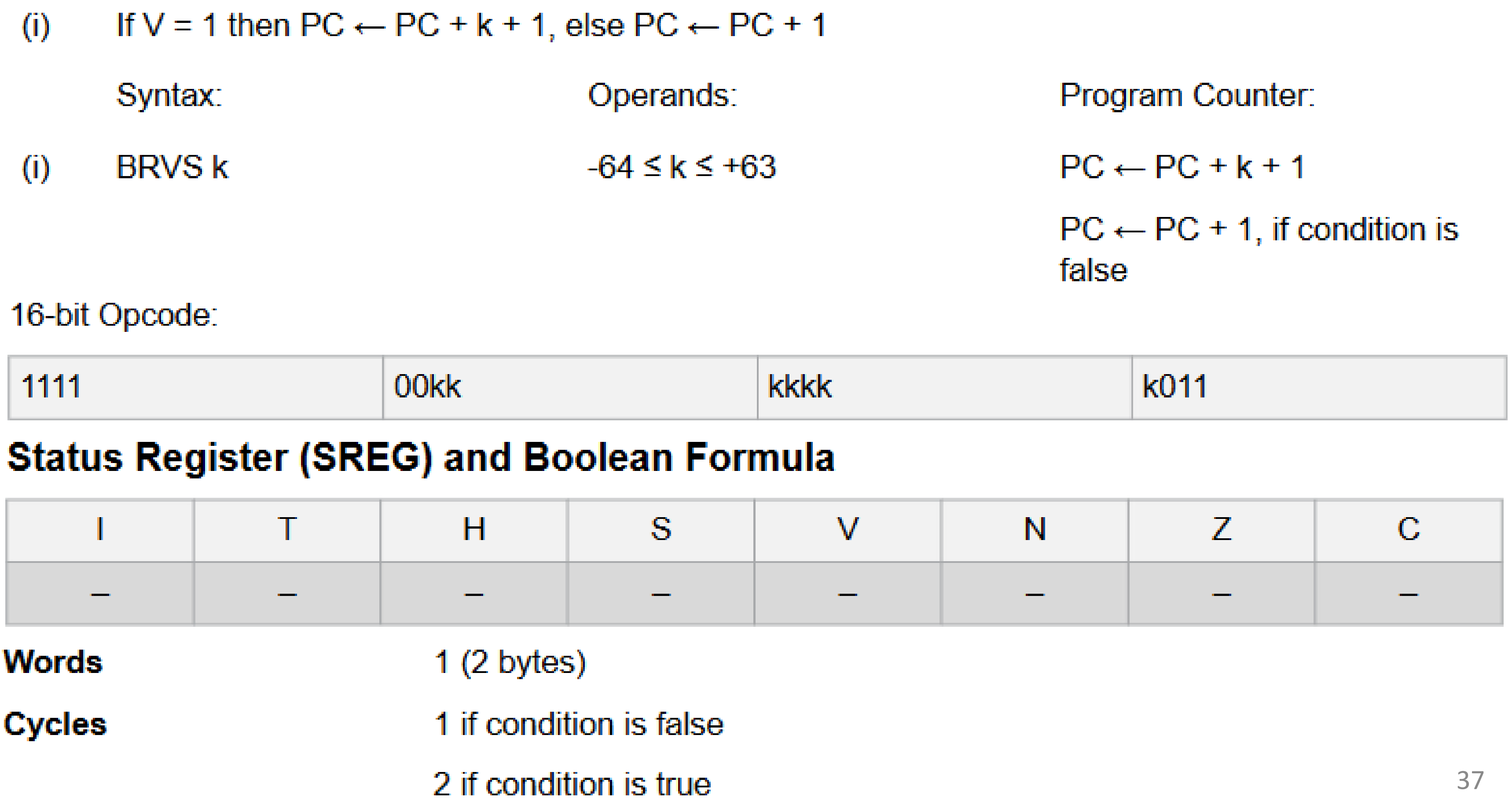
Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1 if condition is false
	2 if condition is true

BRVS – Branch if Overflow Set

- Conditional relative branch. Tests the Overflow Flag (V) and branches relatively to PC if V is set.



(i) If V = 1 then PC ← PC + k + 1, else PC ← PC + 1

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

16-bit Opcode:

1111	00kk	kkkk	k011

Status Register (SREG) and Boolean Formula

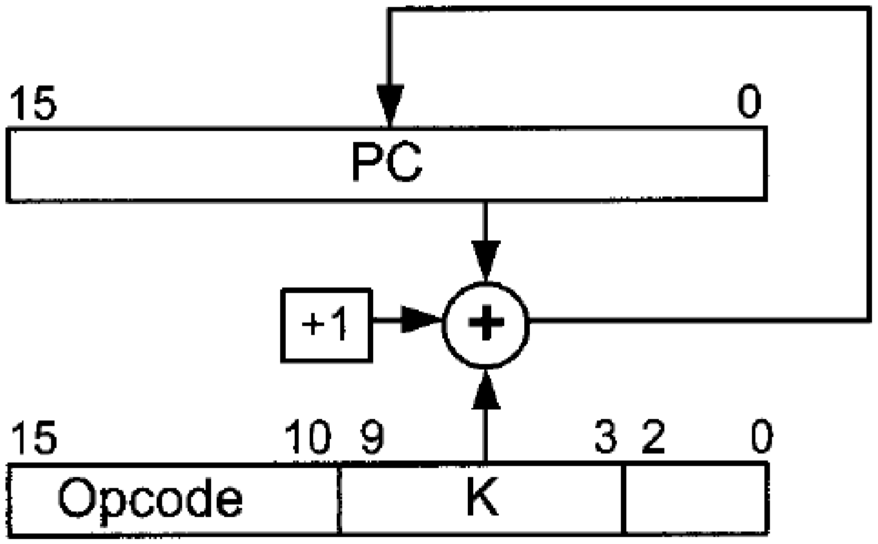
I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1 if condition is false
	2 if condition is true

طول پرش

- All conditional branches are short jumps
  - They are all 2-byte instructions
    - The relative address is 7 bits
    - Relative to program counter
  - Address of target must be within 64 bytes of PC
    - Relative address, positive → jump forward
    - Relative address, negative→ jump backwards

1111	01kk	kkkk	k000



پرش‌های غیر شرطی

- Unconditional branches (jumps)
  - Control unconditionally transferred to the target location
- AVR Unconditional branches
  - JMP
  - RJMP
  - IJMP
- Use depend on target address

JMP – Jump

- Jump to an address within the entire 4M (words) Program memory.

(i)	PC ← k			
	Syntax:	Operands:	Program Counter:	Stack:
(i)	JMP k	$0 \leq k < 4M$	PC ← k	Unchanged

32-bit Opcode:

1001	010k	kkkk	110k
kkkk	kkkk	kkkk	kkkk

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	2 (4 bytes)
Cycles	3

RJMP – Relative Jump

- Relative jump to an address within PC - 2K +1 and PC + 2K (words).

(i)	$PC \leftarrow PC + k + 1$			
	Syntax:	Operands:	Program Counter:	Stack:
(i)	RJMP	$k\ -2K \le k < 2K$	$PC \leftarrow PC + k + 1$	Unchanged

16-bit Opcode:

1100	kkkk	kkkk	kkkk

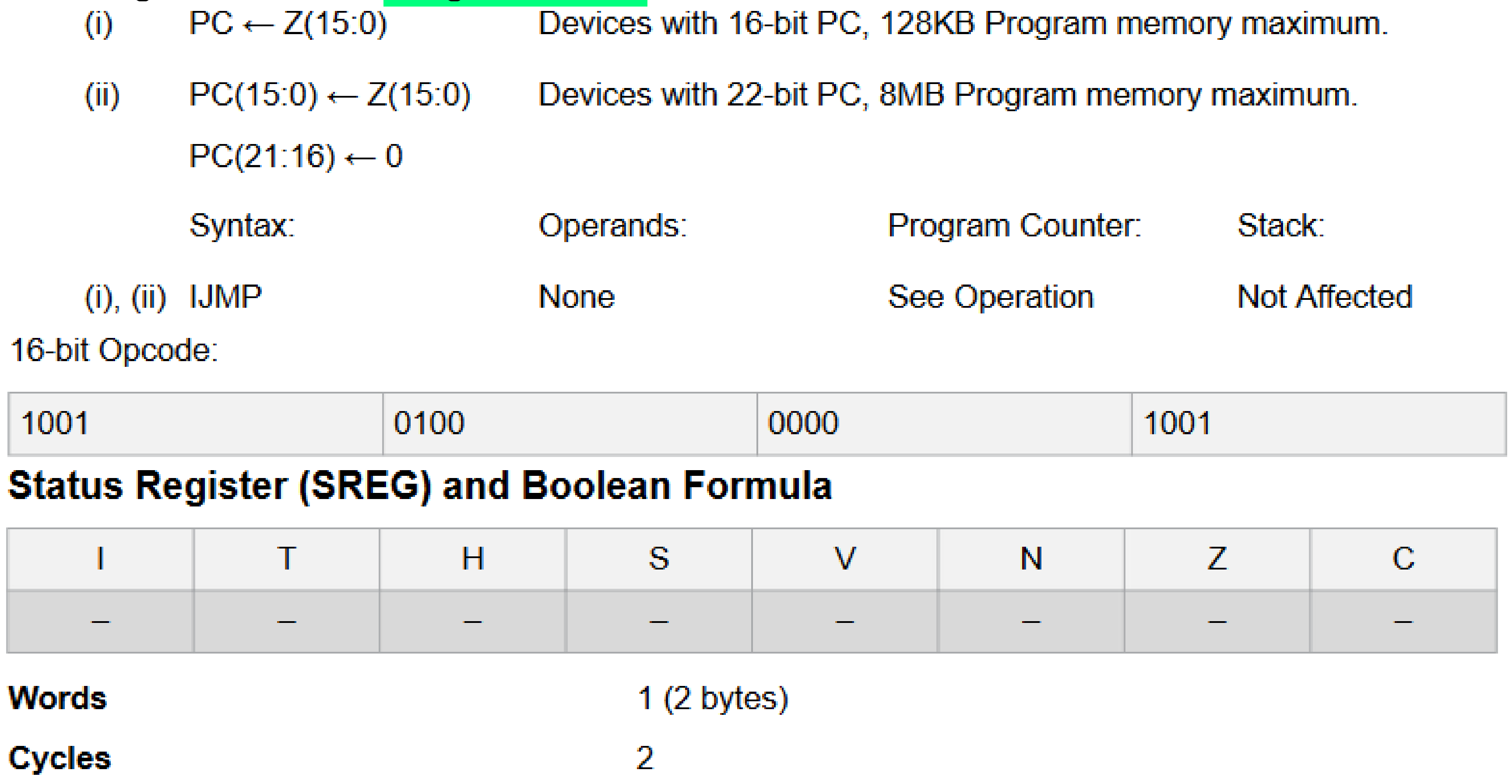
Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	2

IJMP – Indirect Jump

- Indirect jump to the address pointed to by the Z (16 bits) Pointer Register in the Register File.



(i)	PC ← Z(15:0)	Devices with 16-bit PC, 128KB Program memory maximum.
(ii)	PC(15:0) ← Z(15:0)	Devices with 22-bit PC, 8MB Program memory maximum.
	PC(21:16) ← 0	

	Syntax:	Operands:	Program Counter:	Stack:
(i), (ii)	IJMP	None	See Operation	Not Affected

16-bit Opcode:

1001	0100	0000	1001

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	2

پایان

موفق و پیروز باشید