Table 5-1. Cycles

Name	Cycles
AVRe	1
AVRxm	1
AVRxt	1
AVRrc	1

5.2 ADD – Add without Carry

5.2.1 Description

Adds two registers without the C flag and places the result in the destination register Rd.

Operation:

(i) (i) Rd ← Rd + Rr

	Syntax:	Operands:	Program Counter:
(i)	ADD Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0000	11rd	dddd	rrrr

5.2.2 Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H $Rd3 \wedge Rr3 \vee Rr3 \wedge \overline{R3} \vee \overline{R3} \wedge Rd3$

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

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

V $Rd7 \wedge Rr7 \wedge \overline{R7} \vee \overline{Rd7} \wedge \overline{Rr7} \wedge R7$

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

Set if the result is `0x00`; cleared otherwise.

C $Rd7 \wedge Rr7 \vee Rr7 \wedge \overline{R7} \vee \overline{R7} \wedge Rd7$

Set if there was a carry from the MSB of the result; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
add r1,r2 ; Add r2 to r1 (r1=r1+r2)
add r28,r28 ; Add r28 to itself (r28=r28+r28)
```

Words	1 (2 bytes)

Table 5-2. Cycles

Name	Cycles
AVRe	1
AVRxm	1
AVRxt	1
AVRrc	1

5.3 ADIW – Add Immediate to Word

5.3.1 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 on all devices. Refer to Appendix A.

Operation:

(i) R[d+1]:Rd ← R[d+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

5.3.2 Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	⇔	⇔	⇔

S N ⊕ V, for signed tests.

V $\overline{Rdh7} \wedge R15$

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 $\overline{R15} \wedge \overline{R14} \wedge \overline{R13} \wedge \overline{R12} \wedge \overline{R11} \wedge \overline{R10} \wedge \overline{R9} \wedge \overline{R8} \wedge \overline{R7} \wedge \overline{R6} \wedge \overline{R5} \wedge \overline{R4} \wedge \overline{R3} \wedge \overline{R2} \wedge \overline{R1} \wedge \overline{R0}$

Set if the result is `0x0000`; cleared otherwise.

C $\overline{R15} \wedge Rdh7$

Set if there was a carry from the MSB of the result; cleared otherwise.

R (Result) equals R[d+1]:Rd after the operation.

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)

Table 5-3. Cycles

Name	Cycles
AVRe	2
AVRxm	2
AVRxt	2
AVRrc	N/A

5.4 AND – Logical AND

5.4.1 Description

Performs the logical AND between the contents of register Rd and register Rr, and places the result in the destination register Rd.

Operation:

(i) Rd ← Rd ∧ Rr

Syntax:	Operands:	Program Counter:
(i) AND Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0010	00rd	dddd	rrrr

5.4.2 Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

S N ⊕ V, for signed tests.

V 0

Cleared.

N R7

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

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

Set if the result is `0x00`; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
and r2,r3 ; Bitwise and r2 and r3, result in r2
ldi r16,1 ; Set bitmask 0000 0001 in r16
and r2,r16 ; Isolate bit 0 in r2
```

Words 1 (2 bytes)

Table 5-4. Cycles

Name	Cycles
AVRe	1
AVRxm	1
AVRxt	1
AVRrc	1

5.5 ANDI – Logical AND with Immediate

5.5.1 Description

Performs the logical AND between the contents of register Rd and a constant, and places the result in the destination register Rd.

Operation:

(i) $Rd \leftarrow Rd \wedge K$

Syntax:	Operands:	Program Counter:
(i) ANDI Rd,K	$16 \le d \le 31, 0 \le K \le 255$	$PC \leftarrow PC + 1$

16-bit Opcode:

0111	KKKK	dddd	KKKK

5.5.2 Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

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

V 0

Cleared.

N R7

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

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

Set if the result is `0x00`; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
clr r31 ; Clear Z high byte
ldi r30,0x60 ; Set Z low byte to 0x60
ld r0,Z+ ; Load r0 with data space loc. 0x60(Z post inc)
ld r1,Z ; Load r1 with data space loc. 0x61
ldi r30,0x63 ; Set Z low byte to 0x63
ld r2,Z ; Load r2 with data space loc. 0x63
ld r3,-Z ; Load r3 with data space loc. 0x62(Z pre dec)
ldd r4,Z+2 ; Load r4 with data space loc. 0x64
```

Words 1 (2 bytes)

Table 5-68. Cycles

Name	Cycles			
	i	ii	iii	iv
AVRe	2(1)	2(1)	2(1)	2(1)
AVRxm	2(1)(3)	2(1)(3)	3(1)(3)	3(1)(3)
AVRxt	2(2)	2(2)	2(2)	2(2)
AVRrc	1 / 2	2 / 3	2 / 3	N/A

Note:

1. Cycle times for data memory access assume internal RAM access and are not valid for accessing external RAM.
2. Cycle time for data memory access assumes internal RAM access, and are not valid for access to NVM. A minimum of one extra cycle must be added when accessing NVM. The additional time varies dependent on the NVM module implementation. See the NVMCTRL section in the specific devices data sheet for more information.
3. If the LD instruction is accessing I/O Registers, one cycle can be deducted.

5.69 LDI – Load Immediate

5.69.1 Description

Loads an 8-bit constant directly to register 16 to 31.

Operation:

(i) Rd ← K

Syntax:	Operands:	Program Counter:
(i) LDI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

1110	KKKK	dddd	KKKK

5.69.2 Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
clr r31 ; Clear Z high byte
ldi r30,0xF0 ; Set Z low byte to 0xF0
lpm ; Load constant from Program
; memory pointed to by Z
```

Words 1 (2 bytes)

Table 5-69. Cycles

Name	Cycles
AVRe	1
AVRxm	1
AVRxt	1
AVRrc	1

5.70 LDS – Load Direct from Data Space

5.70.1 Description

Loads one byte from the data space to a register. The data space usually consists of the Register File, I/O memory, and SRAM, refer to the device data sheet for a detailed definition of the data space.

A 16-bit address must be supplied. Memory access is limited to the current data segment of 64 KB. The LDS instruction uses the RAMPD Register to access memory above 64 KB. To access another data segment in devices with more than 64 KB data space, the RAMPD in the register in the I/O area has to be changed.

This instruction is not available on all devices. Refer to Appendix A.

Operation:

(i) Rd ← DS(k)

	Syntax:	Operands:	Program Counter:
(i)	LDS Rd,k	0 ≤ d ≤ 31, 0 ≤ k ≤ 65535	PC ← PC + 2

32-bit Opcode:

1001	000d	dddd	0000
kkkk	kkkk	kkkk	kkkk

5.70.2 Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
lds r2,0xFF00 ; Load r2 with the contents of data space location 0xFF00
add r2,r1 ; add r1 to r2
sts 0xFF00,r2 ; Write back
```

Words 2 (4 bytes)

5.74.2 Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	⇔	0	⇔	⇔

S N ⊕ V, for signed tests.

V N ⊕ C, for N and C after the shift.

N 0

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

Set if the result is `0x00`; cleared otherwise.

C Rd0

Set if, before the shift, the LSB of Rd was set; cleared otherwise.

R (Result) equals Rd after the operation.

Example:

```
add r0,r4 ; Add r4 to r0
lsr r0 ; Divide r0 by 2
```

Words 1 (2 bytes)

Table 5-74. Cycles

Name	Cycles
AVRe	1
AVRxm	1
AVRxt	1
AVRrc	1

5.75 MOV – Copy Register

5.75.1 Description

This instruction makes a copy of one register into another. The source register Rr is left unchanged, while the destination register Rd is loaded with a copy of Rr.

Operation:

(i) Rd ← Rr

	Syntax:	Operands:	Program Counter:
(i)	MOV Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← PC + 1

16-bit Opcode:

0010	11rd	dddd	rrrr

5.75.2 Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
 mov r16,r0 ; Copy r0 to r16
 call check ; Call subroutine
 ...
check: cpi r16,0x11 ; Compare r16 to 0x11
 ...
 ret ; Return from subroutine
```

Words 1 (2 bytes)

Table 5-75. Cycles

Name	Cycles
AVRe	1
AVRxm	1
AVRxt	1
AVRrc	1

5.76 MOVW – Copy Register Word

5.76.1 Description

This instruction makes a copy of one register pair into another register pair. The source register pair Rr+1:Rr is left unchanged, while the destination register pair Rd+1:Rd is loaded with a copy of Rr + 1:Rr.

This instruction is not available on all devices. Refer to Appendix A.

Operation:

(i) R[d+1]:Rd ← R[r+1]:Rr

	Syntax:	Operands:	Program Counter:
(i)	MOVW Rd+1:Rd,Rr+1:Rr	d ∈ {0,2,...,30}, r ∈ {0,2,...,30}	PC ← PC + 1

16-bit Opcode:

0000	0001	dddd	rrrr

5.76.2 Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
 mov w r17:16,r1:r0 ; Copy r1:r0 to r17:r16
 call check ; Call subroutine
 ...
check: cpi r16,0x11 ; Compare r16 to 0x11
 ...
```