Mnemonics	Instructions Description	Operation	Flags	#
ARITHMETIC AND LOGIC INSTRUCTIONS				
ADD Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,S,H	1
ADC Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1
ADIW Rd, K	Add Immediate to Word	R[d + 1]:Rd ← R[d + 1]:Rd + K	Z,C,N,V,S	2
SUB Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,S,H	1
SUBI Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,S,H	1
SBC Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,S,H	1
SBCI Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,S,H	1
SBIW Rd, K	Subtract Immediate from Word	R[d + 1]:Rd ← R[d + 1]:Rd - K	Z,C,N,V,S	2
AND Rd, Rr	Logical AND	Rd ← Rd ∧ Rr	Z,N,V,S	1
ANDI Rd, K	Logical AND with Immediate	Rd ← Rd ∧ K	Z,N,V,S	1
OR Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V,S	1
ORI Rd, K	Logical OR with Immediate	Rd ← Rd v K	Z,N,V,S	1
EOR Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V,S	1
COM Rd	One's Complement	Rd ← 0xFF - Rd	Z,C,N,V,S	1
NEG Rd	Two's Complement	Rd ← 0x00 - Rd	Z,C,N,V,S,H	1
SBR Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V,S	1
CBR Rd,K	Clear Bit(s) in Register	Rd ← Rd ∧ (0xFFh - K)	Z,N,V,S	1
INC Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1
DEC Rd	Decrement	Rd ← Rd - 1	Z,N,V,S	1
TST Rd	Test for Zero or Minus	Rd ← Rd ∧ Rd	Z,N,V,S	1
CLR Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V,S	1
SER Rd	Set Register	Rd ← 0xFF	None	1
MUL Rd,Rr	Multiply Unsigned	R1:R0 ← Rd x Rr (UU)	Z,C	2
MULS Rd,Rr	Multiply Signed	R1:R0 ← Rd x Rr (SS)	Z,C	2
MULSU Rd,Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr (SU)	Z,C	2
FMUL Rd,Rr	Fractional Multiply Unsigned	R1:R0 ← Rd x Rr<<1 (UU)	Z,C	2
FMULS Rd,Rr	Fractional Multiply Signed	R1:R0 ← Rd x Rr<<1 (SS)	Z,C	2
FMULSU Rd,Rr	Fractional Multiply Signed×Unsigned	R1:R0 ← Rd x Rr<<1 (SU)	Z,C	2
BRANCH INSTRUCTIONS				
RJMP k	Relative Jump	PC ← PC + k + 1	None	2
IJMP	IJMP Indirect Jump to (Z)	PC(15:0) ← Z	None	2
JMP k	Jump	PC ← k	None	3
RCALL k	Relative Call Subroutine	PC ← PC + k +1	None	3
ICALL	Indirect Call to (Z)	PC(15:0) ← Z	None	3
CALL k	Call Subroutine	PC ← k	None	4
RET	Subroutine Return	PC ← STACK	None	4
RETI	Interrupt Return	PC ← STACK	I	4
CPSE Rd,Rr	Compare, skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP Rd,Rr	Compare	Rd - Rr	Z,C,N,V,S,H	1
CPC Rd,Rr	Compare with Carry	Rd - Rr - C	Z,C,N,V,S,H	1
CPI Rd,K	Compare with Immediate	Rd - K	Z,C,N,V,S,H	1
SBRC Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC ← PC + 2 or 3	None	1/2/3
SBRS Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) PC ← PC + 2 or 3	None	1/2/3
SBIC A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b) = 0) PC ← PC + 2 or 3	None	1/2/3
SBIS A, b	Skip if Bit in I/O Register Set	If (I/O(A,b) = 1) PC ← PC + 2 or 3	None	1/2/3
BRBS s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC + k	None	1/2
BRBC s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC + k	None	1/2
BREQ k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE k	Branch if Greater or Equal, Signed	if (N ⊕ V = 0) then PC ← PC + k + 1	None	1/2
BRLT k	Branch if Less Than, Signed	if (N ⊕ V = 1) then PC ← PC + k + 1	None	1/2
BRHS k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS k	Branch if T Bit Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC k	Branch if T Bit Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2

Mnemonics	Instructions Description	Operation	Flags	#
BRVS k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
DATA TRANSFER INSTRUCTIONS				
MOV Rd, Rr	Copy Register	Rd ← Rr	None	1
MOVW Rd, Rr	Copy Register Pair	R[d + 1]:Rd ← R[r + 1]:Rr	None	1
LDI Rd, K	Load Immediate	Rd ← K	None	1
LD Rd, X	Load Indirect	Rd ← DS(X)	None	2
LD Rd, X+	Load Indirect and Post-Increment	Rd ← DS(X) , X ← X + 1	None	2
LD Rd, -X	Load Indirect and Pre-Decrement X	X ← X - 1 , Rd ← DS(X)	None	2
LD Rd, Y	Load Indirect	Rd ← DS(Y)	None	2
LD Rd, Y+	Load Indirect and Post-Increment	Rd ← DS(Y) , Y ← Y + 1	None	2
LD Rd, -Y	Load Indirect and Pre-Decrement Y	Y ← Y - 1 , Rd ← DS(Y)	None	2
LDD Rd, Y+q	Load Indirect with Displacement	Rd ← DS(Y + q)	None	2
LD Rd, Z	Load Indirect	Rd ← DS(Z)	None	2
LD Rd, Z+	Load Indirect and Post-Increment	Rd ← DS(Z) , Z ← Z + 1	None	2
LD Rd, -Z	Load Indirect and Pre-Decrement	Z ← Z - 1 , Rd ← DS(Z)	None	2
LDD Rd, Z+q	Load Indirect with Displacement	Rd ← DS(Z + q)	None	2
LDS Rd, k	Load Direct from Data Space	Rd ← DS(k)	None	2
ST X, Rr	Store Indirect	DS(X) ← Rr	None	2
ST X+, Rr	Store Indirect and Post-Increment	DS(X) ← Rr , X ← X + 1	None	2
ST -X, Rr	Store Indirect and Pre-Decrement	X ← X - 1 , DS(X) ← Rr	None	2
ST Y, Rr	Store Indirect DS(Y) ← Rr	DS(Y) ← Rr	None	2
ST Y+, Rr	Store Indirect and Post-Increment	DS(Y) ← Rr , Y ← Y + 1	None	2
ST -Y, Rr	Store Indirect and Pre-Decrement	Y ← Y - 1 , DS(Y) ← Rr	None	2
STD Y+q, Rr	Store Indirect with Displacement	DS(Y + q) ← Rr	None	2
ST Z, Rr	Store Indirect	DS(Z) ← Rr	None	2
ST Z+, Rr	Store Indirect and Post-Increment	DS(Z) ← Rr , Z ← Z + 1	None	2
ST -Z, Rr	Store Indirect and Pre-Decrement	Z ← Z - 1 , DS(Z) ← Rr	None	2
STD Z+q,Rr	Store Indirect with Displacement	DS(Z + q) ← Rr	None	2
STS k, Rr	Store Direct to Data Space	DS(k) ← Rd	None	2
LPM	Load Program Memory	R0 ← PS(Z)	None	3
LPM Rd, Z	Load Program Memory	Rd ← PS(Z)	None	3
LPM Rd, Z+	Load Program Memory and Post-Incr.	Rd ← PS(Z) , Z ← Z + 1	None	3
SPM	Store Program Memory	PS(Z) ← R1:R0	None	-
IN Rd, A	In From I/O Location	Rd ← I/O(A)	None	1
OUT A, Rr	Out To I/O Location	I/O(A) ← Rr	None	1
PUSH Rr	Push Register on Stack	STACK ← Rr	None	2
POP Rd	Pop Register from Stack	Rd ← STACK	None	2
BIT AND BIT-TEST INSTRUCTIONS				
SBI A, b	Set Bit in I/O Register	I/O(A, b) ← 1	None	1
CBI A, b	Clear Bit in I/O Register	I/O(A, b) ← 0	None	1
LSL Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V,H	1
LSR Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL Rd	Rotate Left Through Carry	Rd(0) ← C , Rd(n+1) ← Rd(n), C ← Rd(7)	Z,C,N,V,H	1
ROR Rd	Rotate Right Through Carry	Rd(7) ← C , Rd(n) ← Rd(n+1), C ← Rd(0)	Z,C,N,V	1
ASR Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), Rd(7) ←Rd(7)	Z,C,N,V	1
SWAP Rd	Swap Nibbles	Rd(3..0) ↔ Rd(7..4)	None	1
BST Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
BSET s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR s	Flag Clear	SREG(s) ← 0	SREG(s)	1
SEC	Set Carry	C ← 1	C	1
CLC	Clear Carry	C ← 0	C	1
SEN	Set Negative Flag	N ← 1	N	1
CLN	Clear Negative Flag	N ← 0	N	1
SEZ	Set Zero Flag	Z ← 1	Z	1
CLZ	Clear Zero Flag	Z ← 0	Z	1
SEI	Global Interrupt Enable	I ← 1	I	1
CLI	Global Interrupt Disable	I ← 0	I	1
SES	Set Sign Bit	S ← 1	S	1
CLS	Clear Sign Bit	S ← 0	S	1

Mnemonics	Instructions Description	Operation	Flags	#
SEV	Set Two's Complement Overflow	V ← 1	V	1
CLV	Clear Two's Complement Overflow	V ← 0	V	1
SET	Set T in SREG	T ← 1	T	1
CLT	Clear T in SREG	T ← 0	T	1
SEH	Set Half Carry Flag in SREG	H ← 1	H	1
CLH	Clear Half Carry Flag in SREG	H ← 0	H	1

MCU Control

NOP	No Operation	No Operation	None	1
SLEEP	Sleep	See the power management and sleep description	None	1
WDR	Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK	Break	For On-chip Debug Only	None	N/A

Stack

STACK Stack for return address and pushed registers

SP The Stack Pointer (12-bit)

Memory Space Identifiers

DS( ) Represents a pointer to address in data space

PS( ) Represents a pointer to address in program space

I/O(A) I/O space address A

I/O(A,b) Bit position b of byte in I/O space address A

Rd(n) Bit n in register Rd

Registers and Operands

Rd: Destination register in the 8-bit Register File (R0-R31)

Rr: Source register in the 8-bit Register File (R0-R31)

R: Result after instruction is executed

K: Constant data

k: Constant address

b: Bit position (0..7) in the Register File or I/O Register

s: Bit position (0..7)in the Status Register

A: I/O memory address

q: Displacement for direct addressing

UU Unsigned × Unsigned operands

SS Signed × Signed operands

SU Signed × Unsigned operands

X,Y,Z: Indirect Address Register (X=R27:R26, Y=R29:R28, and Z=R31:R30

Operator

× Arithmetic multiplication

+ Arithmetic addition

- Arithmetic subtraction

∧ Logical AND

∨ Logical OR

⊕ Logical XOR

>> Shift right

<< Shift left

== Comparison

← Assignment

↔ Swap

Status Register (SREG)

SREG Status Register

C Carry Flag

Z Zero Flag

N Negative Flag

V Two's Complement Overflow Flag

S Sign Flag

H Half Carry Flag

T Copy Storage

I Global Interrupt Enable Bit

Program Memory Map (ATmega328)

Flash ROM: 0x0000–0x3FFF (16Kx16-bit=32KB)

Program Counter (PC) 14-bit

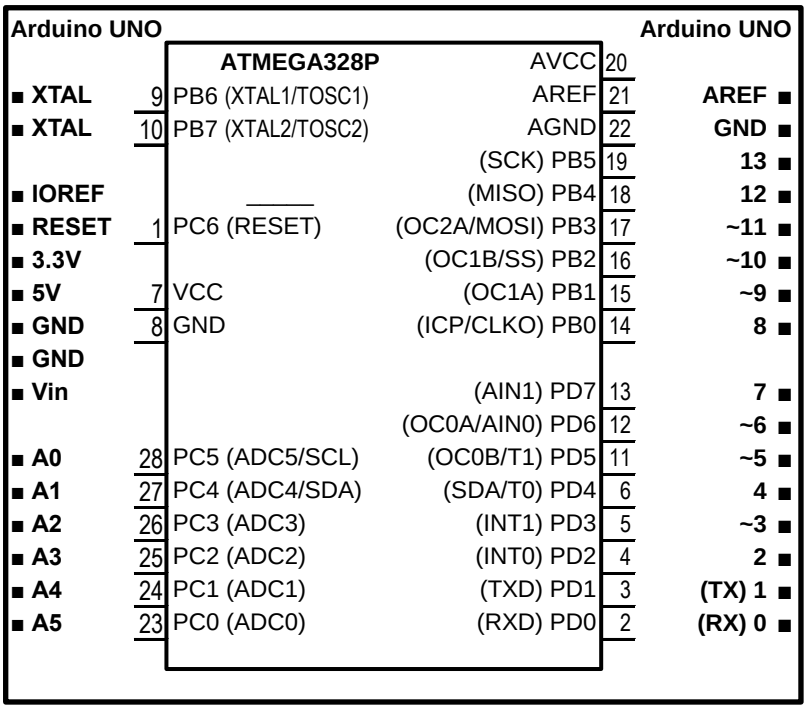
Data Memory Map (ATmega328)

32 Register: 0x0000–0x001F

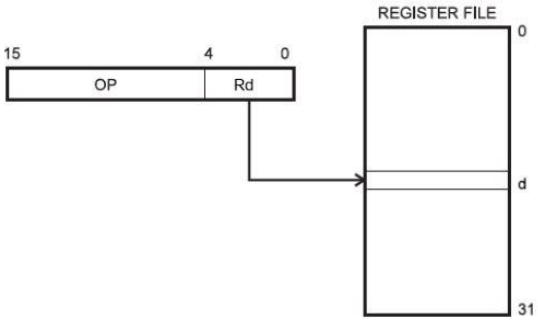
64 I/O Register 0x0020–0x005F

160 Ext.I/O Reg. 0x0060–0x00FF

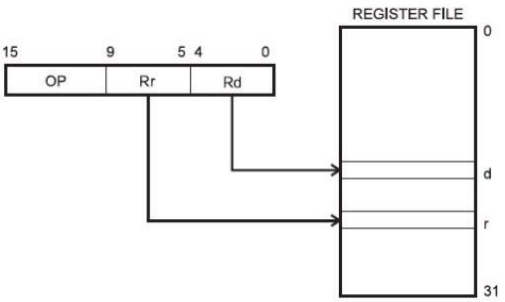
Internal SRAM 0x0100–0x08FF (2Kx8-bit=2KB)



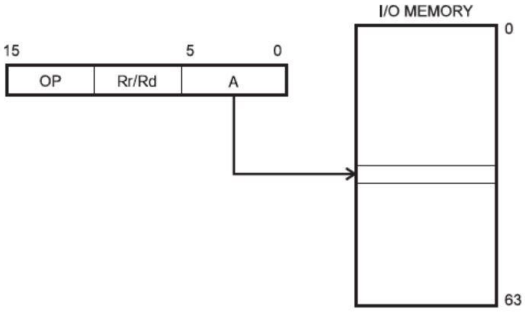
1. Direct Register Addressing, Single



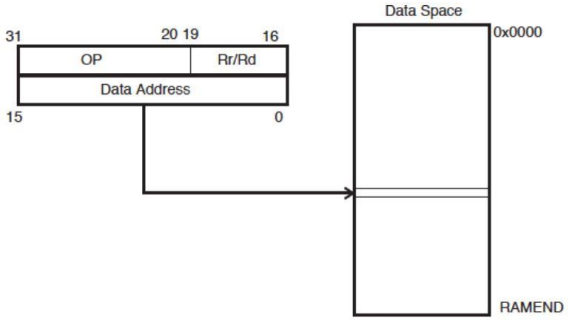
2. Direct Register Addressing, Two Registers



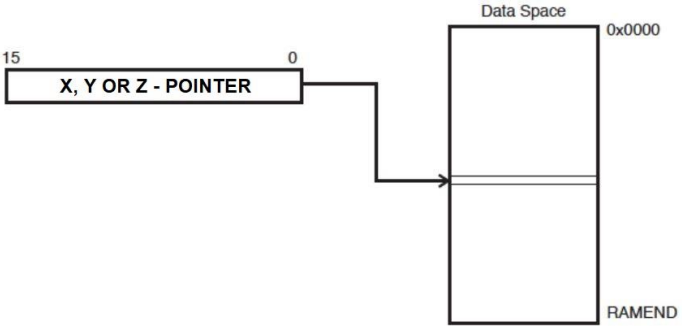
3. I/O Direct Addressing



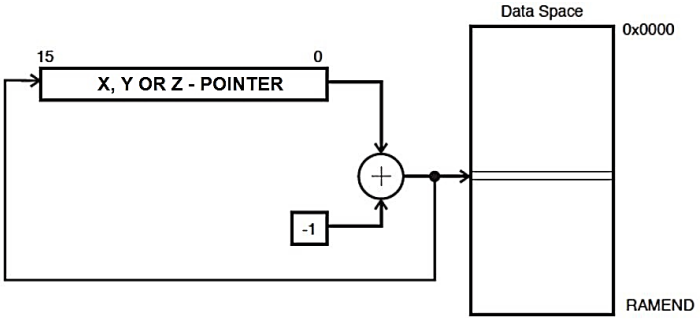
4. Direct Data Addressing



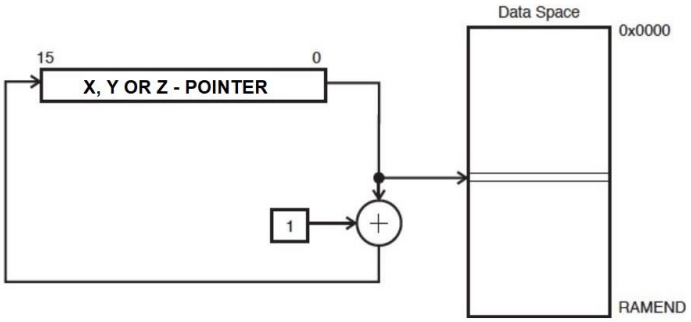
5. Data Indirect Addressing



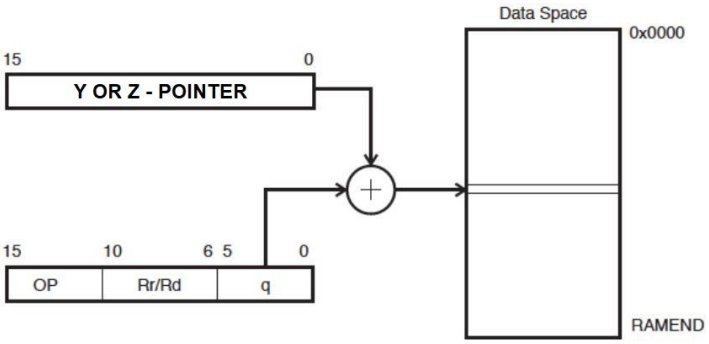
6. Data Indirect Addressing with Pre-decrement



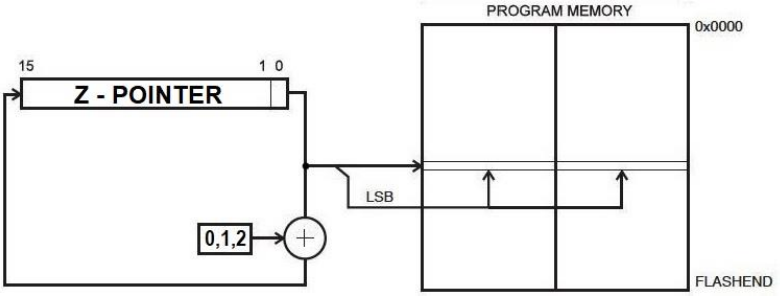
7. Data Indirect Addressing with Post-increment



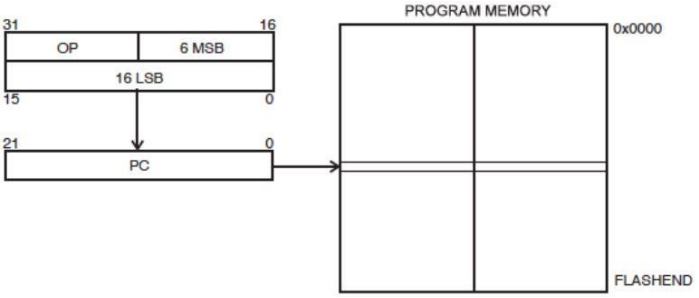
8. Data Indirect with Displacement



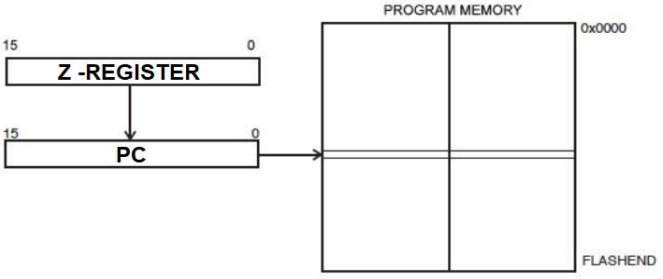
9. Program Memory Constant Addressing, LPM, ELPM, and SPM (Z, Z+)



10. Direct Program Addressing, JMP and CALL



11. Indirect Program Addressing, IJMP and ICALL



12. Relative Program Addressing, RJMP and RCALL