CÁC THANH GHI SFRs
& ĐỊA CHỈ IORs CƠ BẢN

Địa chỉ		Tên
Bộ nhớ	I/O	Thanh ghi
0x20	0x00	**PINA**
0x21	0x01	**DDRA**
0x22	0x02	**PORTA**
0x23	0x03	**PINB**
0x24	0x04	**DDRB**
0x25	0x05	**PORTB**
0x26	0x06	**PINC**
0x27	0x07	**DDRC**
0x28	0x08	**PORTC**
0x29	0x09	**PIND**
0x2A	0x0A	**DDRD**
0x2B	0x0B	**PORTD**
0x2C … 0x34		Reserved
0x35	0x15	**TIFR0**
0x36	0x16	**TIFR1**
0x37	0x17	**TIFR2**
0x38 .. 0x3A		Reserved
0x3B	0x1B	**PCIFR**
0x3C	0x1C	**EIFR**
0x3D	0x1D	**EIMSK**
0x3E	0x1E	**GPIOR0**
0x3F	0x1F	**EECR**
0x40	0x20	**EEDR**
0x41	0x21	**EEARL và**
0x42	0x22	**EEARH**
0x43	0x23	**GTCCR**
0x44	0x24	**TCCR0A**
0x45	0x25	**TCCR0B**
0x46	0x26	**TCNT0**
0x47	0x27	**OCR0A**
0x48	0x28	**OCR0B**

Địa chỉ		Tên
Bộ nhớ	I/O	Thanh ghi
0x49		Reserved
0x4A	0x2A	**GPIOR1**
0x4B	0x2B	**GPIOR2**
0x4C	0X2C	**SPCR0**
0x4D	0x2D	**SPSR0**
0x4E	0x2E	**SPDR0**
0x4F	0x2F	Reserved
0x50	0x30	**ACSR**
0x51	0x31	**OCDR**
0x52	0x32	Reserved
0x53	0x33	**SMCR**
0x54	0x34	**MCUSR**
0x55	0x35	**MCUCR**
0x56		Reserved
0x57	0x37	**SPMCSR**
0x58 … 0x5A		Reserved
0x5B	0x3B	**RAMPZ**
0x5C		Reserved
0x5D	0x3D	**SPL và**
0x5E	0x3E	**SPH**
0x5F	0x3F	**SREG**
0x60		**WDTCSR**
0x61		**CLKPR**
0x62		Reserved
0x63		Reserved
0x64		**PRR0**
0x65		Reserved
0x66		**OSCCAL**
0x67		Reserved
0x68		**PCICR**
0x69		**EICRA**
0x6A		Reserved

Địa chỉ		Tên
Bộ nhớ	I/O	Thanh ghi
0x6B		**PCMSK0**
0x6C		**PCMSK1**
0x6D		**PCMSK2**
0x6E		**TIMSK0**
0x6F		**TIMSK1**
0x70		**TIMSK2**
0x71		Reserved
0x72		Reserved
0x73		**PCMSK3**
0x74 … 0x77		Reserved
0x78		**ADCL và**
0x79		**ADCH**
0x7A		**ADCSRA**
0x7B		**ADCSRB**
0x7C		**ADMUX**
0x7D		Reserved
0x7E		**DIDR0**
0x7F		**DIDR1**
0x80		**TCCR1A**
0x81		**TCCR1B**
0x82		**TCCR1C**
0x83		Reserved
0x84		**TCNT1L và**
0x85		**TCNT1H**
0x86		**ICR1L và**
0x87		**ICR1H**
0x88		**OCR1AL và**
0x89		**OCR1AH**
0x8A		**OCR1BL và**
0x8B		**OCR1BH**
0x8C … 0xAF		Reserved

Địa chỉ		Tên
Bộ nhớ	I/O	Thanh ghi
0xB0		**TCCR2A**
0xB1		**TCCR2B**
0xB2		**TCNT2**
0xB3		**OCR2A**
0xB4		**OCR2B**
0xB5		Reserved
0xB6		**ASSR**
0xB7		Reserved
0xB8		**TWBR**
0xB9		**TWSR**
0xBA		**TWAR**
0xBB		**TWDR**
0xBC		**TWCR**
0xBD		**TWAMR**
0xBE		Reserved
0xBF		Reserved
0xC0		**UCSR0A**
0xC1		**UCSR0B**
0xC2		**UCSR0C**
0xC3		Reserved
0xC4		**UBRR0L và**
0xC5		**UBRR0H**
0xC6		**UDR0**
0xC7		Reserved
0xC8		**UCSR1A**
0xC9		**UCSR1B**
0xCA		**UCSR1C**
0xCB		Reserved
0xCC		**UBRR1L và**
0xCD		**UBRR1H**
0xCE		**UDR1**
0xCF … 0xFF		Reserved

CÁC KÝ HIỆU ĐƯỢC SỬ DỤNG TRONG TẬP LỆNH

Thanh ghi trạng thái SREG và các bit trong thanh ghi

SREG: thanh ghi trạng thái

C: Cờ Carry

Z: Cờ Zero

N: Cờ âm

V: Cờ tràn bù 2

S: Cờ dấu = N $\oplus$ V

H: Cờ nhớ phân nữa

T: bit sao chép sử dụng trong các lệnh **BLD** và **BST**

I: Cờ cho phép/không cho phép ngắt toàn cục

Các thanh ghi và toán hạng trong lệnh

Rd: thanh ghi đích trong tập thanh ghi

Rr: thanh ghi nguồn

R: kết quả sau khi thực thi lệnh

K: hằng số dữ liệu

k: hằng số địa chỉ

b: bit trong tập **GPRs** hoặc **I/O REGs** (3-bit)

s: Bit trong **SREG** (3-bit)

X,Y,Z: T.ghi địa chỉ gián tiếp

X=R27:R26,**Y**=R29:R28,**Z**=R31:R30

P: địa chỉ các **I/O REGs** cơ bản**.**

q: độ dời hoặc địa chỉ trực tiếp (6-bit)

INSTRUCTION SET SUMMARY

Mnemonics	Operands	Description	Operation	Flags	CK
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	**Rd, Rr** {d,r: 0÷31}	Add two Registers	Rd ← Rd + Rr	H,S,V,N,Z,C	1
ADC	**Rd, Rr** {d,r: 0÷31}	Add with Carry two Registers	Rd ← Rd + Rr + C	H,S,V,N,Z,C	1
ADIW	**Rdl,K** {dl: 24,26,28,30- K:0÷63}	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	S,V,N,Z,C	2
SUB	**Rd, Rr** {d,r: 0÷31}	Subtract two Registers	Rd ← Rd - Rr	H,S,V,N,Z,C	1
SUBI	**Rd, K** {d: 16÷31, K:0÷255}	Subtract Imemediate from Register	Rd ← Rd - K	H,S,V,N,Z,C	1
SBC	**Rd, Rr** {d,r: 0÷31}	Subtract with Carry two Registers	Rd ← Rd - Rr - C	H,S,V,N,Z,C	1
SBCI	**Rd, K** {d,r: 16÷31, K:0÷255}	Subtract Imemediate with Carry from Reg.	Rd ← Rd - K - C	H,S,V,N,Z,C	1
SBIW	**Rdl,K** {dl: 24,26,28,30- K:0÷63}	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	S,V,N,Z,C	2
AND	**Rd, Rr** {d,r: 0÷31}	Logical AND Registers	Rd ← Rd • Rr	S,V=0,N,Z	1
ANDI	**Rd, K** {d: 16÷31, K:0÷255}	Logical AND Register and Constant	Rd ← Rd • K	S,V=0,N,Z	1
OR	**Rd, Rr** {d,r: 0÷31}	Logical OR Registers	Rd ← Rd v Rr	S,V=0,N,Z	1
ORI	**Rd, K** {d: 16÷31, K:0÷255}	Logical OR Register and Constant	Rd ← Rd v K	S,V=0,N,Z	1
EOR	**Rd, Rr** {d,r: 0÷31}	Exclusive OR Registers	Rd ← Rd ⊕ Rr	S,V=0,N,Z	1
COM	**Rd** {d: 0÷31}	One's Complement	Rd ← 0xFF – Rd	S,V=0,N,Z,C=1	1
NEG	**Rd** {d: 0÷31}	Two's Complement	Rd ← 0x00 – Rd	H,S,V,N,Z,C	1
SBR	**Rd**,**K** {d: 16÷31, K:0÷255}	Set Bit(s) in Register	Rd ← Rd v K	S,V=0,N,Z	1
CBR	**Rd,K** {d: 16÷31, K:0÷255}	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	S,V=0,N,Z	1
INC	**Rd** {d: 0÷31}	Increment	Rd ← Rd + 1	S,V,N,Z	1
DEC	**Rd** {d: 0÷31}	Decrement	Rd ← Rd – 1	S,V,N,Z	1
TST	**Rd** {d: 0÷31}	Test for Zero or Minus	Rd ← Rd • Rd	S,V=0,N,Z	1
CLR	**Rd** {d: 0÷31}	Clear Register	Rd ← Rd ⊕ Rd	S=0,V=0,N=0,Z=1	1
SER	**Rd** {d: 16÷31}	Set Register	Rd ← 0xFF	None	1
MUL	**Rd, Rr** {d,r: 0÷31}	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	**Rd, Rr** {d,r: 16÷31}	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	**Rd, Rr** {d,r: 16÷23}	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	**Rd, Rr** {d,r: 16÷23}	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	**Rd, Rr** {d,r: 16÷23}	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	**Rd, Rr** {d,r: 16÷23}	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2

DATA TRANSFER INSTRUCTIONS					
MOV	**Rd, Rr** {d,r:0÷31}	Move Between Registers	Rd ← Rr	None	1
MOVW	**Rd, Rr** {d,r∈(0,2,4,..28,30)}	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	**Rd, K** {d:16÷31, K:0÷255}	Load Immediate	Rd ← K	None	1
LD	**Rd, X** {d:0÷31, X≡R27:R26}	Load Indirect	Rd ← (X)	None	2
LD	**Rd, X+** {d:0÷31}	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	**Rd, - X** {d:0÷31}	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	**Rd, Y** {d:0÷31, Y≡R29:R28}	Load Indirect	Rd ← (Y)	None	2
LD	**Rd, Y+** { d:0÷31}	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	**Rd, - Y** { d:0÷31}	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	**Rd,Y+q** {d:0÷31, q:0÷63}	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	**Rd, Z** {d:0÷31, Z≡R31:R30}	Load Indirect	Rd ← (Z)	None	2
LD	**Rd, Z+** { d:0÷31}	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	**Rd, -Z** { d:0÷31}	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	**Rd, Z+q** {d:0÷31, q:0÷63}	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	**Rd, k** {d:0÷31, k:0÷8FFH}	Load Direct from SRAM	Rd ← (k)	None	2
ST	**X, Rr** {r:0÷31}	Store Indirect	(X) ← Rr	None	2
ST	**X+, Rr** {r:0÷31}	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	**- X, Rr** {r:0÷31}	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	**Y, Rr** {r:0÷31}	Store Indirect	(Y) ← Rr	None	2
ST	**Y+, Rr** {r:0÷31}	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	**- Y, Rr** {r:0÷31}	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	**Y+q,Rr** {r:0÷31,q:0÷63}	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	**Z, Rr** {r:0÷31}	Store Indirect	(Z) ← Rr	None	2
ST	**Z+, Rr** {r:0÷31}	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	**-Z, Rr** {r:0÷31}	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	**Z+q,Rr** {r:0÷31,q:0÷63}	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	**k, Rr** {r:0÷31, k:0÷8FFH}	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	**Rd, Z**	Load Program Memory	Rd ← (Z)	None	3
LPM	**Rd, Z+**	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	3
IN	**Rd, P** {d:0÷31,P:0÷63}	In Port	Rd ← P	None	1
OUT	**P, Rr** {r:0÷31,P:0÷63}	Out Port	P ← Rr	None	1
PUSH	**Rr** {r:0÷31}	Push Register on Stack	STACK ← Rr	None	2
POP	**Rd** {d:0÷31}	Pop Register from Stack	Rd ← STACK	None	2

BRANCH INSTRUCTIONS					
RJMP	**k** {k:-2048÷2047}	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z), Z: 0÷3FFFH	PC ← Z	None	2
JMP	**k** {k:0÷3FFFH}	Direct Jump	PC ← k	None	3
RCALL	**k** {k: -2048÷2047}	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z) ,Z: 0÷3FFFH	PC ← Z	None	3
CALL	**k** {k:0÷3FFFH}	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	**Rd,Rr** {d,r: 0÷31}	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	**Rd,Rr** {d,r: 0÷31}	Compare	Rd – Rr	H,S,V,N,Z,C	1
CPC	**Rd,Rr** {d,r: 0÷31}	Compare with Carry	Rd – Rr – C	H,S,V,N,Z,C	1
CPI	**Rd,K** {d: 16÷31, K:0÷255}	Compare Register with Immediate	Rd – K	H,S,V,N,Z,C	1
SBRC	**Rr, b** {r: 0÷31, b:0÷7}	Skip if Bit in Register is Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	**Rr, b** {r: 0÷31, b:0÷7}	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	**P, b** {P: 0÷31, b:0÷7}	Skip if Bit in I/O Register is Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	**P, b** {P: 0÷31, b:0÷7}	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	**s, k** {s: 0÷7, k:-64÷63}	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	**s, k** {s: 0÷7, k:-64÷63}	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	**k** { k:-64÷63}	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	**k** { k:-64÷63}	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	**k** { k:-64÷63}	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	**k** { k:-64÷63}	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	**k** { k:-64÷63}	Branch if Same or Higher -Unsigned	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	**k** { k:-64÷63}	Branch if Lower -Unsigned	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	**k** { k:-64÷63}	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	**k** { k:-64÷63}	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	**k** { k:-64÷63}	Branch if Greater or Equal- Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	**k** { k:-64÷63}	Branch if Less Than Zero- Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	**k** { k:-64÷63}	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	**k** { k:-64÷63}	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	**k** { k:-64÷63}	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	**k** { k:-64÷63}	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	**k** { k:-64÷63}	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	**k** { k:-64÷63}	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	**k** { k:-64÷63}	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	**k** { k:-64÷63}	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2

BIT AND BIT-TEST INSTRUCTIONS					
SBI	**P, b** {P: 0÷31, b:0÷7}	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	**P, b** {P: 0÷31, b:0÷7}	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	**Rd** {d: 0÷31}	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	**Rd** {d: 0÷31}	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	**Rd** {d: 0÷31}	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	**Rd** {d: 0÷31}	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	**Rd** {d: 0÷31}	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	**Rd** {d: 0÷31}	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	**s** {s: 0÷7}	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	**s** {s: 0÷7}	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	**Rr, b** {r: 0÷31, b:0÷7}	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	**Rd, b** {d: 0÷31, b:0÷7}	Bit load from T to Register	Rd(b) ← T	None	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 Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos 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 INSTRUCTIONS				
NOP	No Operation		None	1
SLEEP	Sleep	(see specific descr. for Sleep function)	None	1
WDR	Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK	Break	For On-chip Debug Only	None	1