Índice

Valores iniciales para datos en memoria y en registros internos de CS2010 y AVR

Memoria:
En general para memorias de datos de 1B: **M($...$H_1H_0$) = $ H_0H_1**

Esto es, el contenido de la memoria tendrá como valor inicial los 8 LSB de su dirección intercambiando sus dos *nibbles*. Ejemplos:

M($7A)= $A7; M($00)= 0; M($100)= 0; M($7)=$70; M($107)=$70;

M($11)=$11; M($12)=$21; M($x34)=$43; M($x56)=$65; M($xFE)=$EF; etc.

Registros internos:

Salvo R26 a R31 en AVR, el dato de Rk será el número de registro por 10, Rk = 10·k:

Rk	**R0**	**R1**	**R2**	**R3**	**R4**	**…**	**R20**	**R21**	**R22**	**R23**	**R24**	**R25**
Dec	0	10	20	30	40	…	200	210	220	230	240	250
Hex	0	A	14	1E	28	…	C8	D2	DC	E6	F0	FA

Por su parte, los valores iniciales de R26 a R31 en AVR serán:

Rk	**R26**	**R27**	**R28**	**R29**	**R30**	**R31**
Dec	26	1	28	2	30	3
Hex	1A	1	1C	2	1E	3

Excepciones:
Si alguno de estos valores no se pudiera aplicar, se propondrá otro.

Operadores Verilog

Bitwise Operators		
~	~m	invert each bit of m
&	m & n	AND each bit of m with each bit of n
|	m | n	OR each bit of m with each bit of n
^	m ^ n	exclusive-OR each bit of m with n
~^ *or* ^~	m ~^ n	exclusive-NOR each bit of m with n
<<	m << n	shift m left n-times and fill with zeros
>>	m >> n	shift m right n-times and fill with zeros

Unary Reduction Operators		
&	&m	AND all bits in m together (1-bit result)
~&	~&m	NAND all bits in m together (1-bit result)
|	|m	OR all bits in m together (1-bit result)
~|	~|m	NOR all bits in m together (1-bit result)
^	^m	exclusive-OR all bits in m (1-bit result)
~^ *or* ^~	~^m	exclusive-NOR all bits in m (1-bit result)

Logical Operators		
!	!m	is m not true? (1-bit True/False result)
&&	m && n	are both m and n true? (1-bit True/False result)
||	m || n	are either m or n true? (1-bit True/False result)

Equality and Relational Operators (return X if an operand has X or Z)		
==	m == n	is m equal to n? (1-bit True/False result)
!=	m != n	is m not equal to n? (1-bit True/False result)
<	m < n	is m less than n? (1-bit True/False result)
>	m > n	is m greater than n? (1-bit True/False result)
<=	m <= n	is m less than or equal to n? (1-bit True/False result)
>=	m >= n	is m greater than or equal to n? (1-bit True/False result)

Identity Operators (compare logic values 0, 1, X, and Z)		
===	m === n	is m identical to n? (1-bit True/False results)
!==	m !== n	is m not identical to n? (1-bit True/False result)

Miscellaneous Operators		
? :	sel?m:n	conditional operator; if sel is true, return m: else return n
{}	{m,n}	concatenate m to n, creating a larger vector
{{}}	{n{ }}	replicate inner concatenation n-times
–>	-> m	trigger an event on an event data type

Arithmetic Operators		
+	m + n	add n to m
–	m – n	subtract n from m
–	–m	negate m (2's complement)
*	m * n	multiply m by n
/	m / n	divide m by n
%	m % n	modulus of m / n
**	m ** n	m to the power n (new in Verilog-2001)
<<<	m <<< n	shift m left n-times, filling with 0 (new in Verilog-2001)
>>>	m >>> n	shift m right n-times; fill with value of sign bit if expression is signed, otherwise fill with 0 (Verilog-2001)

Sintaxis escogidas

T'BV **-T'BV** **T'sBV**

TipoDato [MSB:LSB] nombre1, nombre2,…

assign [#retraso] nombre_variable = expresión

always @ (lista de sensibilidad)

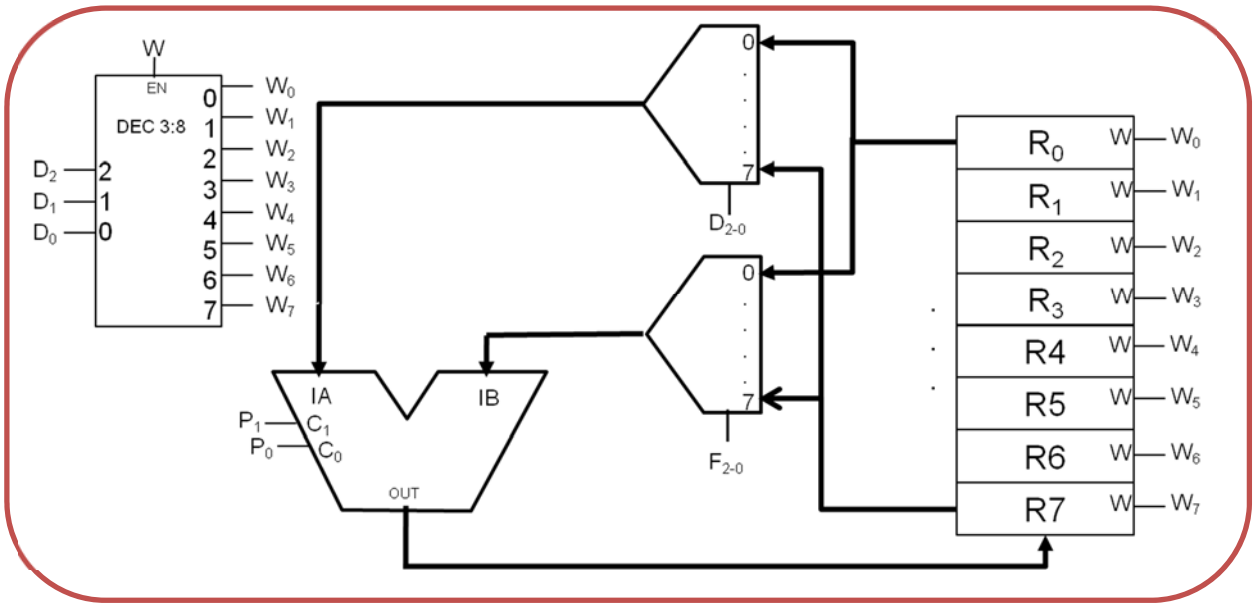
begin … end

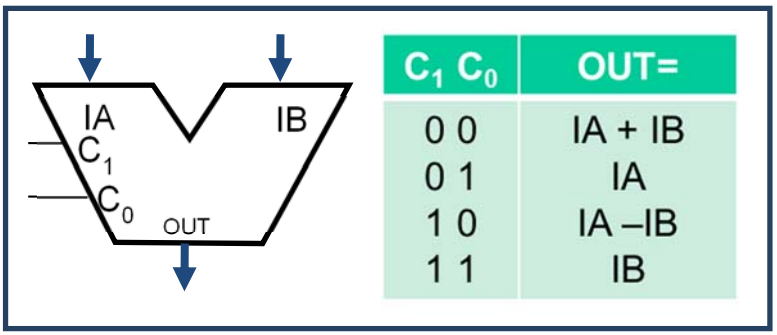
if (condicionlogica/relacional) sentecia_si_true; [else sentecia_si_false;]

parameter nom1=valor1, nom2=valor2, …;

module nombre **#(parameter nom1=valor1,…) …**

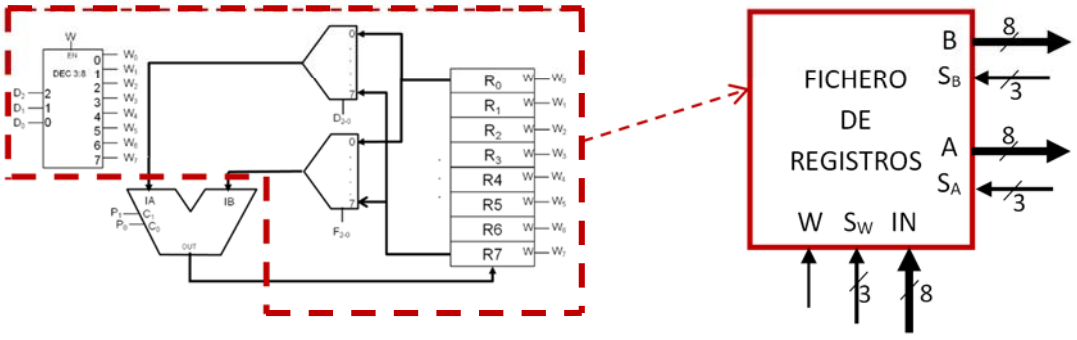
Calculadora





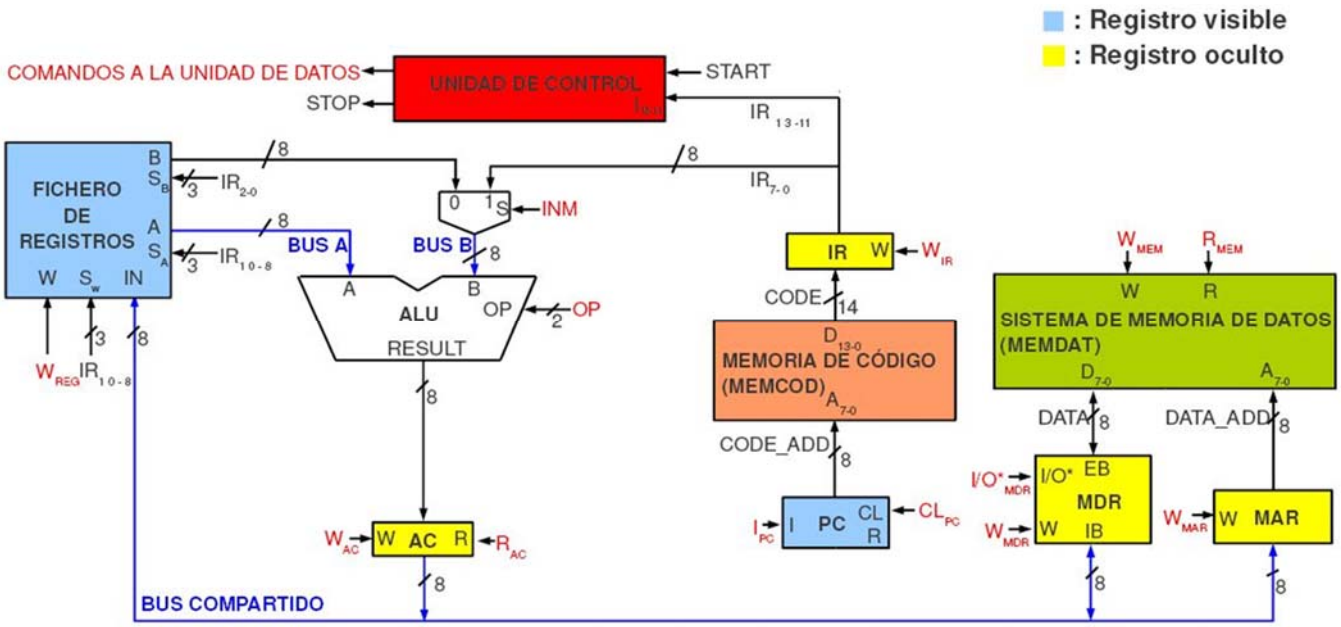
C_1 C_0	OUT=
0 0	IA + IB
0 1	IA
1 0	IA –IB
1 1	IB

En los computadores simples CS2 y CS2010 se simplifica como FICHERO DE REGISTROS:



Computador simple 2

	IR (*Instruction Register*)													
	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Sintaxis	**COP**			**Operandos**										
ST (Rb), Rf	0	0	0	fuente			-	-	-	-	-	base		
LD Rd, (Rb)	0	0	1	destino			-	-	-	-	-	base		
STS dir, Rf	0	1	0	fuente			dirección							
LDS Rd, dir	0	1	1	destino			dirección							
ADD Rd,Rf	1	0	0	destino			-	-	-	-	-	fuente		
SUB Rd,Rf	1	0	1	destino			-	-	-	-	-	fuente		
MOV Rd,Rf	1	1	0	destino			-	-	-	-	-	fuente		
STOP	1	1	1	-	-	-	-	-	-	-	-	-	-	-



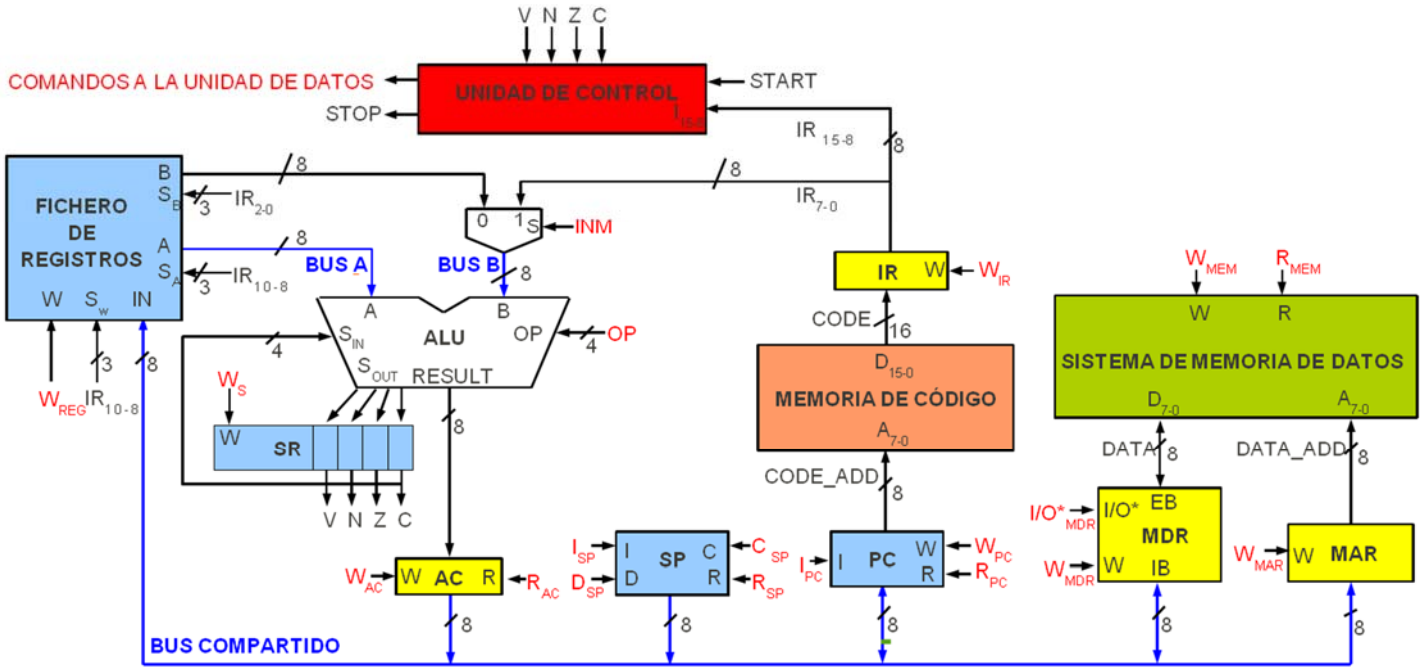
ALU

C_1 C_0	OUT=
0 0	IA + IB
0 1	IA
1 0	IA –IB
1 1	IB

MDR

W I/O*	MDR ←	IB =	EB =
0 0	MDR	H.I.	MDR
0 1	MDR	MDR	H.I.
1 0	IB	Data	H.I.
1 1	EB	H.I.	Data

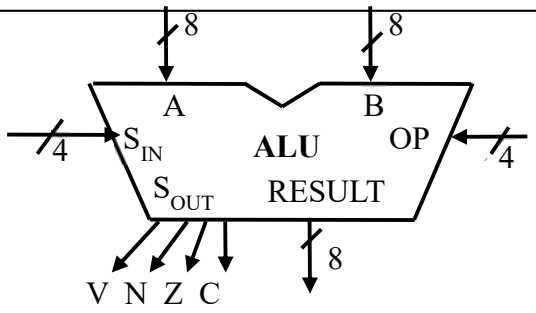
COMPUTADOR SIMPLE 2010: ARQUITECTURA



MDR

W I/O*	MDR ←	IB =	EB =
0 0	MDR	H.I.	MDR
0 1	MDR	MDR	H.I.
1 0	IB	Data	H.I.
1 1	EB	H.I.	Data

$S_{IN,OUT} = \{V, N, Z, C\}_{IN,OUT}$



OP_3 OP_2 OP_1 OP_0	RESULT=	V_{OUT}=	N_{OUT}=	Z_{OUT}=	C_{OUT}=
0 0 - 0	-	V_{IN}	N_{IN}	Z_{IN}	0
0 0 1 1					1
0 1 0 0	SHR(A, C_{IN})	$C_{IN} \oplus A_7$	$RESULT_7$	= 1 si RESULT=0 = 0 en otro caso	A_0
0 1 0 1	SHL(A, C_{IN})	$A_7 \oplus A_6$			A_7
0 1 1 -	A	-			-
1 0 0 -	A + B	$C_7 \oplus C_{OUT}$			C_{OUT} de A+B
1 0 1 -	A - B				Bw_{OUT} de A-B
1 1 - -	B	-			-

COMPUTADOR SIMPLE 2010: INSTRUCCIONES

Formatos

Tipo de formato	IR[16]			
	15 : 11	10 : 8	7 : 3	2 : 0
A: Con Registros	C O P	Reg. destino (Fuente en ST)	-----	Reg. Fuente (Rbase en ST/LD)
B: Mem & Inmed			Dirección/Dato inmediato	
C: Salto		Condición	Dirección de salto	

Conjunto de instrucciones (ordenadas por el código de operación, COP)

COP IR_{15} : IR_{11}	For-mato	Mnemónico y sintaxis	Tipo	Efecto	V N Z C [1]
0000 0	A	ST (Rb), Rf	Mem	M(Rb) ← Rf	- - - -
0000 1	A	LD Rd, (Rb)	Mem	Rd ← M(Rb)	- - - -
0001 0	B	STS dir, Rf	Mem	M(dir) ← Rf	- - - -
0001 1	B	LDS Rd, dir	Mem	Rd ← M(dir)	- - - -
0010 0	C	CALL dir	Salto	M(SP)←PC; SP←SP-1; PC←dir	- - - -
0010 1	-	RET	Salto	PC←M(SP+1); SP←SP+1	- - - -
0011 0	C	BRxx dir	Salto	Xx: PC ← dir	- - - -
0011 1	C	JMP dir	Salto	PC ← dir	- - - -
0100 0	A	ADD Rd, Rf	Arit/Log	Rd ← Rd + Rf	* * * *
0101 0	A	SUB Rd, Rf	Arit/Log	Rd ← Rd - Rf	* * * *
0101 1	A	CP Rd, Rf	Estado	NOP [Rd –Rf ⇒ VNCZ]	* * * *
0111 1	A	MOV Rd, Rf	Movim	Rd ← Rf	- - - -
1001 0	-	CLC	Estado	NOP [⇒ C←0]	- - - 0
1001 1	-	SEC	Estado	NOP [⇒ C←1]	- - - 1
1010 0	A/B	ROR Rd	Despl	Rd ← SHR(Rd, C) [⇒ C←Rd0]	* * * Rd0
1010 1	A/B	ROL Rd	Despl	Rd ← SHL(Rd, C) [⇒ C←Rd7]	* * * Rd7
1011 1	-	STOP	Especial		- - - -
1100 0	B	ADDI Rd, dato	Arit/Log	Rd ← Rd + dato	* * * *
1101 0	B	SUBI Rd, dato	Arit/Log	Rd ← Rd - dato	* * * *
1101 1	B	CPI Rd, dato	Estado	NOP [Rd –dato ⇒ VNCZ]	* * * *
1111 1	B	LDI Rd, dato	Mem	Rd ← dato	- - - -

COPs sin utilizar en CS2010:
$(9, C, D, E, 10, 11, 16, 19, 1C, 1D, 1E)

[1] El carácter "-" denota "no modificado"; el carácter "*" denota "modificado de forma definida"; otros COP, no documentados.

Condiciones de los saltos (BRxx)

$IR_{10}IR_9IR_8$	Condición chequeada	BRxx (xx)	NOTAS: Tras realizar A-B se cumple:
0 0 0	Z	ZS, EQ	Tras realizar A-B, "Z=1" ⇔ "A = B"
0 0 1	C	CS, LO	Caso sin signo y tras realizar A-B, "C=1" ⇔ "A < B"
0 1 0	V	VS	Caso con signo (Ca2) y tras operar, "V=1" ⇔ "Resultado no representable"
0 1 1	N ⊕ V	LT	Caso con signo (Ca2) y tras realizar A-B, "N⊕V =1" ⇔ "A < B"
1 - -	Condiciones no definidas en CS2010		

Significado de xx; ZS: zero set; EQ: equal; CS: carry set; LO: lower; VS: overflow set; LT: less than

ENSAMBLADOR: etiqueta: … ;comentarios
EQU etiqueta=numeral ;Ejemplos de numerales: 5, -8, 0xB, 0xA3, 0b01110011

Resumen instrucciones CS2010 por tipo de operación

Movimiento	Aritmética	Lógicas	De bit. Control	De salto	Condiciones	De control
MOV Rd, Rf	ADD Rd, Rf	ROR Rd	CLC	BRxx dir	BREQ	STOP
LD Rd, (Rb)	ADDI Rd, dato	ROL Rd	SEC		BRCS, BRLO	
LDS Rd, dir	SUB Rd, Rf			JMP dir	BRVS	
LDI Rd, dato	SUBI Rd, dato				BRLT	
ST (Rb), Rf	CP Rd, Rf			CALL dir		
STS dir, Rf	CPI Rd, dato			RET		

Ensamblador del CS2010

etiqueta: instrucción
;comentario
EQU etiqueta=numeral ; Ejemplo de numerales: 135, -4, 0, 0x3B, 0xB, 0b01110011

GUIÓN:

- Arithmetic and Logic Instructions
- Data Transfer Instructions
- Bit and Bit-test Instructions
- Branch Instructions

Arithmetic and Logic

Mnemonic	*Operands*	*Range*	*Description*	*Operation*	*Flags*	*Cycles*
ADD	Rd,Rr	d,r ∈ [0,31]	Add without Carry	Rd = Rd + Rr	Z,C,N,V,H,S	1
ADC	Rd,Rr	d,r ∈ [0,31]	Add with Carry	Rd = Rd + Rr + C	Z,C,N,V,H,S	1
ADIW	Rdl,K	d∈{24,26,2830}, k∈ [0,63]	Add Immediate To Word	Rdh:Rdl = Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd,Rr	d,r ∈ [0,31]	Subtract without Carry	Rd = Rd - Rr	Z,C,N,V,H,S	1
SUBI	Rd,K8	d∈[16,31], k8∈ [0,255]	Subtract Immediate	Rd = Rd - K8	Z,C,N,V,H,S	1
SBC	Rd,Rr	d,r ∈ [0,31]	Subtract with Carry	Rd = Rd - Rr - C	Z,C,N,V,H,S	1
SBCI	Rd,K8	d∈[16,31], k8∈ [0,255]	Subtract with Carry Immedtiate	Rd = Rd - K8 - C	Z,C,N,V,H,S	1
AND	Rd,Rr	d,r ∈ [0,31]	Logical AND	Rd = Rd · Rr	Z,N,V,S	1
ANDI	Rd,K8	d∈[16,31], k8∈ [0,255]	Logical AND with Immediate	Rd = Rd · K8	Z,N,V,S	1
OR	Rd,Rr	d,r ∈ [0,31]	Logical OR	Rd = Rd V Rr	Z,N,V,S	1
ORI	Rd,K8	d∈[16,31], k8∈ [0,255]	Logical OR with Immediate	Rd = Rd V K8	Z,N,V,S	1
EOR	Rd,Rr	d,r ∈ [0,31]	Logical Exclusive OR	Rd = Rd EOR Rr	Z,N,V,S	1
COM	Rd	d ∈ [0,31]	One's Complement	Rd = $FF - Rd	Z,C,N,V,S	1
NEG	Rd	d ∈ [0,31]	Two's Complement	Rd = 0 - Rd	Z,C,N,V,H,S	1
SBR	Rd,K8	d∈[16,31], k8∈ [0,255]	Set Bit(s) in Register	Rd = Rd V K8	Z,C,N,V,S	1
CBR	Rd,K8	d∈[16,31], k8∈ [0,255]	Clear Bit(s) in Register	Rd = Rd · ($FF - K8)	Z,C,N,V,S	1
INC	Rd	d ∈ [0,31]	Increment Register	Rd = Rd + 1	Z,N,V,S	1
DEC	Rd	d ∈ [0,31]	Decrement Register	Rd = Rd -1	Z,N,V,S	1
TST	Rd	d ∈ [0,31]	Test for Zero or Negative	Rd = Rd · Rd	Z,C,N,V,S	1
CLR	Rd	d ∈ [0,31]	Clear Register	Rd = 0	Z,C,N,V,S	1
SER	Rd	d ∈ [16,31]	Set Register	Rd = $FF	None	1
SBIW	Rdl,K	d∈{24,26,2830}, k∈ [0,63]	Subtract Immediate from Word	Rdh:Rdl = Rdh:Rdl - K	Z,C,N,V,S	2
MUL	Rd,Rr	d,r ∈ [0,31]	Multiply Unsigned	R1:R0 = Rd * Rr	Z,C	2
MULS	Rd,Rr	d,r ∈ [16,31]	Multiply Signed	R1:R0 = Rd * Rr	Z,C	2
MULSU	Rd,Rr	d,r ∈ [16,23]	Multiply Signed with Unsigned	R1:R0 = Rd * Rr	Z,C	2
FMUL	Rd,Rr	d,r ∈ [16,23]	Fractional Multiply Unsigned	R1:R0 = (Rd * Rr) << 1	Z,C	2
FMULS	Rd,Rr	d,r ∈ [16,23]	Fractional Multiply Signed	R1:R0 = (Rd *Rr) << 1	Z,C	2
FMULSU	Rd,Rr	d,r ∈ [16,23]	Fractional Multiply Signed with Unsigned	R1:R0 = (Rd * Rr) << 1	Z,C	2

Data Transfer

Mnemonic	Operands	Range	Description	Operation	Flags	Cycles
MOV	Rd,Rr	d,r ∈ [0,31]	Copy register	Rd = Rr	None	1
MOVW	Rd,Rr	d,r ∈ [0,2,…30]	Copy register pair	Rd+1:Rd = Rr+1:Rr, r,d even	None	1
LDI	Rd,K8	d∈[16,31], k8∈ [0,255]	Load Immediate	Rd = K	None	1
LDS	Rd,k	d ∈ [0,31],k∈[0,64k]	Load Direct	Rd = (k)	None	2*
LD	Rd,X	d ∈ [0,31]	Load Indirect	Rd = (X)	None	2*
LD	Rd,X+	d ∈ [0,31]	Load Indirect and Post-Increment	Rd = (X), X=X+1	None	2*
LD	Rd,-X	d ∈ [0,31]	Load Indirect and Pre-Decrement	X=X-1, Rd = (X)	None	2*
LD	Rd,Y	d ∈ [0,31]	Load Indirect	Rd = (Y)	None	2*
LD	Rd,Y+	d ∈ [0,31]	Load Indirect and Post-Increment	Rd = (Y), Y=Y+1	None	2*
LD	Rd,-Y	d ∈ [0,31]	Load Indirect and Pre-Decrement	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]	Load Indirect	Rd = (Z)	None	2*
LD	Rd,Z+	d ∈ [0,31]	Load Indirect and Post-Increment	Rd = (Z), Z=Z+1	None	2*
LD	Rd,-Z	d ∈ [0,31]	Load Indirect and Pre-Decrement	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*
STS	k,Rr	r ∈ [0,31], k∈ [0,64k]	Store Direct	(k) = Rr	None	2*
ST	X,Rr	r ∈ [0,31]	Store Indirect	(X) = Rr	None	2*
ST	X+,Rr	r ∈ [0,31]	Store Indirect and Post-Increment	(X) = Rr, X=X+1	None	2*
ST	-X,Rr	r ∈ [0,31]	Store Indirect and Pre-Decrement	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-Increment	(Y) = Rr, Y=Y+1	None	2
ST	-Y,Rr	r ∈ [0,31]	Store Indirect and Pre-Decrement	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-Increment	(Z) = Rr, Z=Z+1	None	2
ST	-Z,Rr	r ∈ [0,31]	Store Indirect and Pre-Decrement	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
LPM	None		Load Program Memory	R0 = (Z)	None	3
LPM	Rd,Z	d ∈ [0,23]	Load Program Memory	Rd = (Z)	None	3
LPM	Rd,Z+	d ∈ [0,23]	Load Program Memory and Post-Increment	Rd = (Z), Z=Z+1	None	3
SPM	None		Store Program Memory	(Z) = R1:R0	None	-
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

Bit and Bit-test

Mnemonic	Operands	Range	Description	Operation	Flags	Cycles
LSL	Rd	d ∈ [0,31]	Logical shift left	Rd(n+1)=Rd(n), Rd(0)=0, C=Rd(7)	Z,C,N,V,H,S	1
LSR	Rd	d ∈ [0,31]	Logical shift right	Rd(n)=Rd(n+1), Rd(7)=0, C=Rd(0)	Z,C,N,V,S	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,H,S	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,S	1
ASR	Rd	d ∈ [0,31]	Arithmetic shift right	Rd(n)=Rd(n+1), n=0,...,6	Z,C,N,V,S	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]	Set flag	SREG(s) = 1	SREG(s)	1
BCLR	s	s ∈ [0,7]	Clear flag	SREG(s) = 0	SREG(s)	1
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
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 register to T	Rd(b) = T	None	1
SEC	None		Set carry flag	C =1	C	1
CLC	None		Clear carry flag	C = 0	C	1
SEN	None		Set negative flag	N = 1	N	1
CLN	None		Clear negative flag	N = 0	N	1
SEZ	None		Set zero flag	Z = 1	Z	1
CLZ	None		Clear zero flag	Z = 0	Z	1
SEI	None		Set interrupt flag	I = 1	I	1
CLI	None		Clear interrupt flag	I = 0	I	1
SES	None		Set signed flag	S = 1	S	1
CLN	None		Clear signed flag	S = 0	S	1
SEV	None		Set overflow flag	V = 1	V	1
CLV	None		Clear overflow flag	V = 0	V	1
SET	None		Set T-flag	T = 1	T	1
CLT	None		Clear T-flag	T = 0	T	1
SEH	None		Set half carry flag	H = 1	H	1
CLH	None		Clear half carry flag	H = 0	H	1
NOP	None		No operation	None	None	1
SLEEP	None		Sleep	See instruction manual	None	1
WDR	None		Watchdog Reset	See instruction manual	None	1

Branch, jump, skip

Mnemonic	*Operands*	*Range*	*Description*	*Operation*	*Flags*	*Cycles*
RJMP	k	k∈ [-2k,2k]	Relative Jump	PC = PC + k +1	None	2
IJMP	None		Indirect Jump to (Z)	PC = Z	None	2
JMP	k	k∈ [0,4M]	Jump	PC = k	None	3
RCALL	k	k∈ [-2k,2k]	Relative Call Subroutine	STACK = PC+1, PC = PC + k + 1	None	3/4*
ICALL	None		Indirect Call to (Z)	STACK = PC+1, PC = Z	None	3/4*
CALL	k	k∈ [0,4M]	Call Subroutine	STACK = PC+2, PC = k	None	4/5*
RET	None		Subroutine Return	PC = STACK	None	4/5*
RETI	None		Interrupt Return	PC = STACK	I	4/5*
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	Z,C,N,V,H,S	1
CPC	Rd,Rr	d,r □ [0,31]	Compare with Carry	Rd - Rr - C	Z,C,N,V,H,S	1
CPI	Rd,K8	d∈[16,31], k8∈ [0,255]	Compare with Immediate	Rd - K	Z,C,N,V,H,S	1
SBRC	Rr,b	r∈[0,31], b∈ [0,7]	Skip if bit in register 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 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 cleared	if(I/O(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 set	if(I/O(P,b)==1) PC = PC + 2 or 3	None	1/2/3
BRBC	s,k	s∈[0,7], k∈ [-64,63]	Branch if Status flag cleared	if(SREG(s)==0) PC = PC + k + 1	None	1/2
BRBS	s,k	s∈[0,7], k∈ [-64,63]	Branch if Status flag set	if(SREG(s)==1) PC = PC + k + 1	None	1/2
BREQ	k	k∈ [-64,63]	Branch if equal	if(Z==1) PC = PC + k + 1	None	1/2
BRNE	k		Branch if not equal	if(Z==0) PC = PC + k + 1	None	1/2
BRCS	k		Branch if carry set	if(C==1) PC = PC + k + 1	None	1/2
BRCC	k		Branch if carry cleared	if(C==0) PC = PC + k + 1	None	1/2
BRSH	k		Branch if same or higher	if(C==0) PC = PC + k + 1	None	1/2
BRLO	k		Branch if lower	if(C==1) PC = PC + k + 1	None	1/2
BRMI	k		Branch if minus	if(N==1) PC = PC + k + 1	None	1/2
BRPL	k		Branch if plus	if(N==0) PC = PC + k + 1	None	1/2
BRGE	k		Branch if greater than or equal (signed)	if(S==0) PC = PC + k + 1	None	1/2
BRLT	k		Branch if less than (signed)	if(S==1) PC = PC + k + 1	None	1/2
BRHS	k		Branch if half carry flag set	if(H==1) PC = PC + k + 1	None	1/2
BRHC	k		Branch if half carry flag cleared	if(H==0) PC = PC + k + 1	None	1/2
BRTS	k		Branch if T flag set	if(T==1) PC = PC + k + 1	None	1/2
BRTC	k		Branch if T flag cleared	if(T==0) PC = PC + k + 1	None	1/2
BRVS	k		Branch if overflow flag set	if(V==1) PC = PC + k + 1	None	1/2
BRVC	k		Branch if overflow flag cleared	if(V==0) PC = PC + k + 1	None	1/2
BRIE	k		Branch if interrupt enabled	if(I==1) PC = PC + k + 1	None	1/2
BRID	k		Branch if interrupt disabled	if(I==0) PC = PC + k + 1	None	1/2

Test (CP Rd,Rr)	**Booleana**	**Mnemonico**	**Comentario**
Rd ≥Rr	(N ⊕ V) = 0	BRGE	Signo
Rd < Rr	(N ⊕ V) = 1	BRLT	Signo
Rd = Rr	Z = 1	BREQ	Signo/Sin Signo
Rd ≠ Rr	Z = 0	BRNE	Signo/Sin Signo
Rd ≥ Rr	C = 0	BRCC/BRSH	Sin Signo
Rd < Rr	C = 1	BRCS/BRLO	Sin Signo
Carry	C=1	BRCS	Simple
Sin carry	C=0	BRCC	Simple
Negativo	N=1	BRMI	Simple
Positivo	N=0	BRPL	Simple
Overflow	V=1	BRVS	Simple
Sin overflow	V=0	BRVC	Simple
Cero	Z=1	BREQ	Simple
No cero	Z=0	BRNE	Simple

Ensamblador:

//comentario	; comentario
.DEF nombre = Rn	. EQU nombre=valor
.ORG valor	
.DSEG	etiqueta: .BYTE valor
.CSEG	etiqueta: .DB valor1, valor2,…