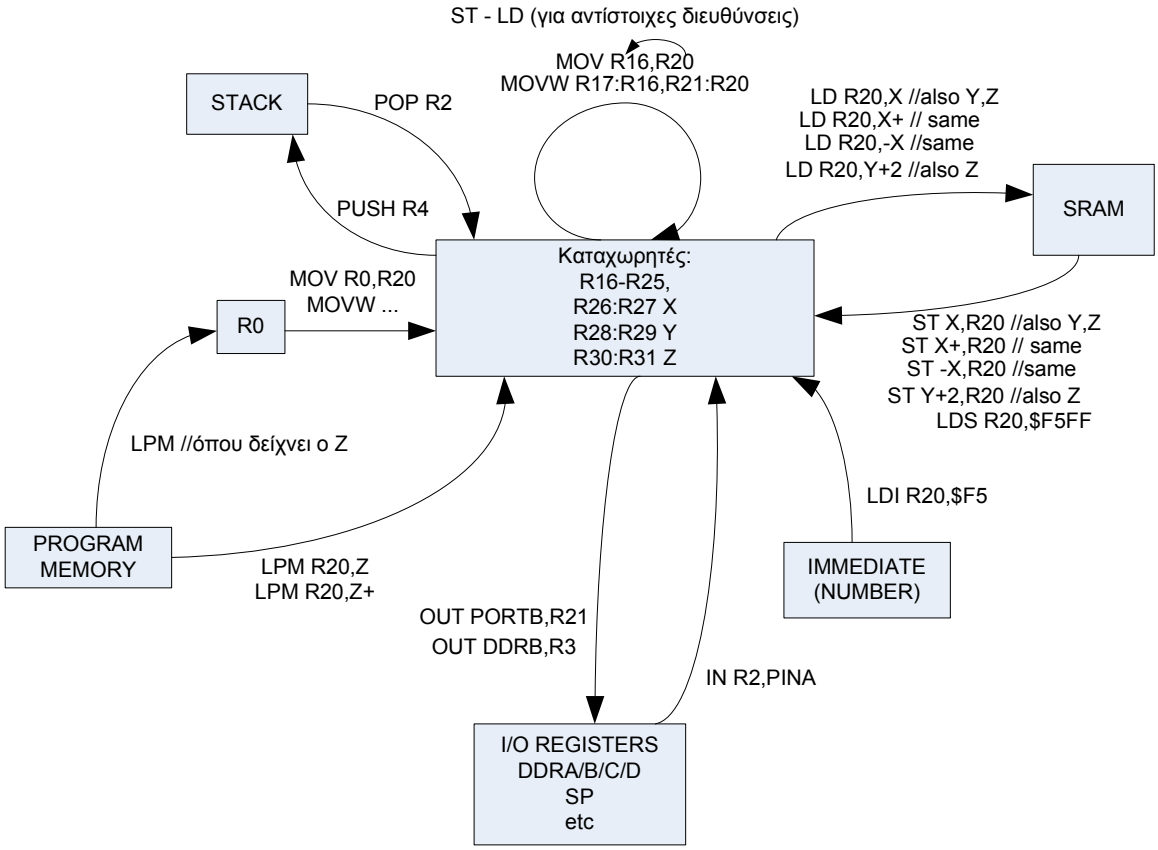
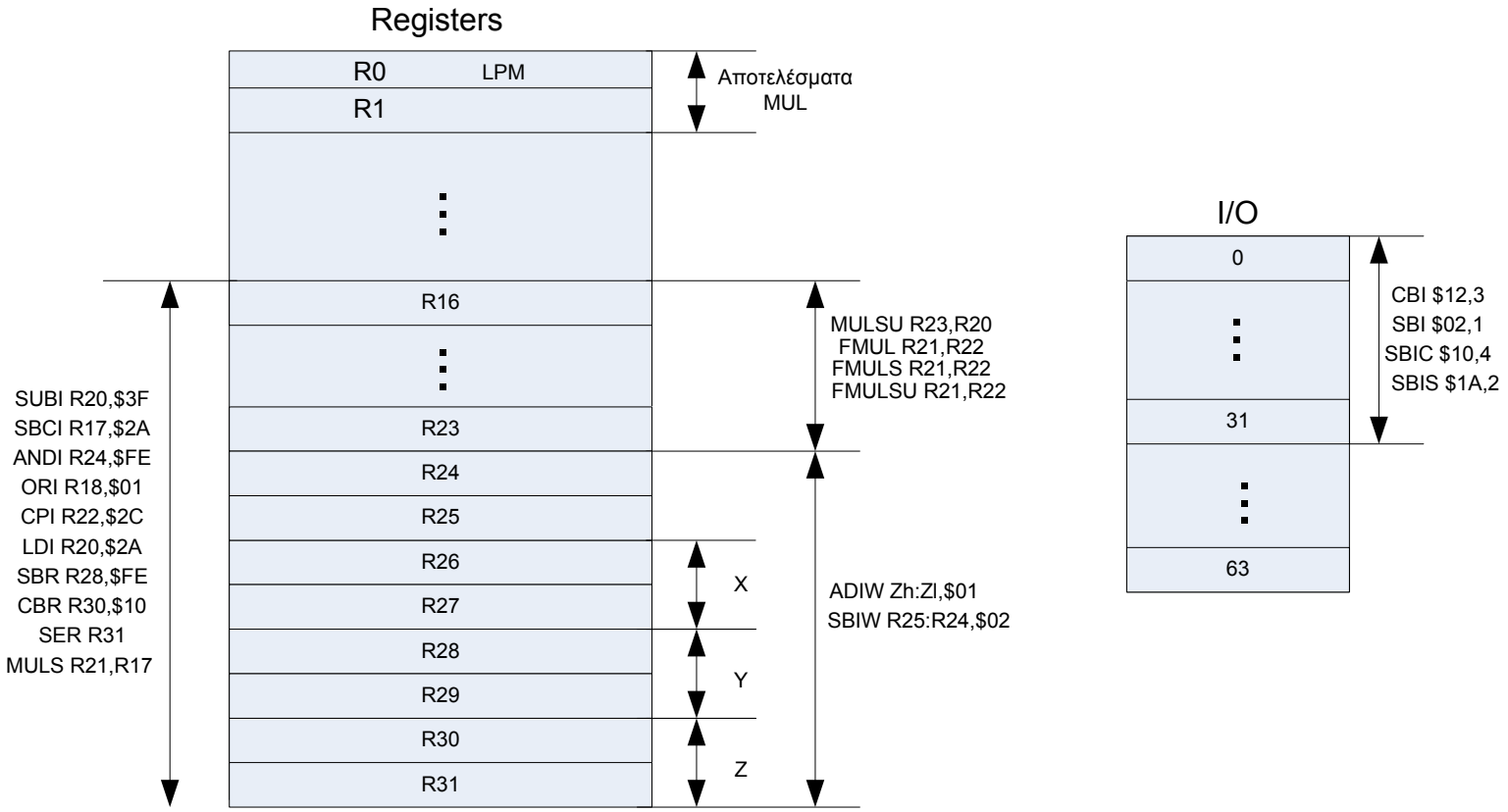
AVR Moving Data



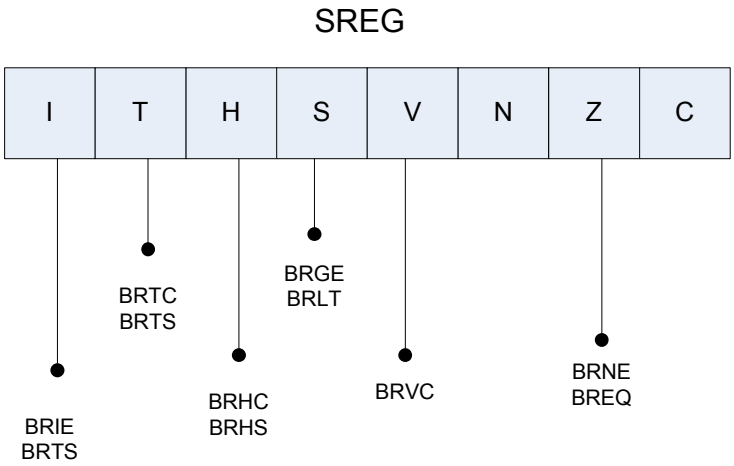
MOV Rb,Rn	Any Rb, Any Rn; Copy data from Rn to Rb
MOVW Rb+1:Rb,Rn+1:Rn	Rn and Rn+1 must be neighboors, same for Rb ; Copy data from Rn:Rn+1 to Rb:Rb+1
POP Rb	Any Rb; Copy data from pointed by SP address to Rb
PUSH Rb	Any Rb; Copy data from Rb to pointed by SP address
LPM	Copy data from pointed by Z address of Program Memory to R0
LPM Rb,Z	Any Rb ; Copy data from pointed by Z address of Program Memory to Rb
LPM Rb,Z+	Any Rb ; Copy data from pointed by Z address of Program Memory to Rb, increase Z after
OUT PORTA/B/C/D,Rb	Any Rb, Any Port ; Copy data from Rb to I/O Resgister
OUT DDRA/B/C/D,Rb	Any Rb, Any Port Sreg ; Copy data from Rb to I/O status Resgister
IN Rb,PINA/B/C/D	Any Rb, Any Port ; Copy data from I/O Resgister to Rb
LD Rb,X/Y/Z	Any Rb, any pointer ; Copy data from pointed by pointer address to Rb
LD Rb,X/Y/Z+	Any Rb, any pointer ; Copy data from pointed by pointer address to Rb, increase pointer after
LD Rb,-X/Y/Z	Any Rb, any pointer ; Copy data from pointed by pointer address to Rb, decrease pointer before
LD Rb,Y/Z+#number	Any Rb, pointers Y and Z ; Copy data from pointed by (pointer + #number) address to Rb ; pointer doesn't change
ST X/Y/Z,Rb	Any Rb, any pointer ; Copy data from Rb to pointed by pointer address
ST X/Y/Z+,Rb	Any Rb, any pointer ; Copy data from Rb to pointed by pointer address, increase pointer after
ST -X/Y/Z,Rb	Any Rb, any pointer ; Copy data from Rb to pointed by pointer address, decrease pointer before
ST Y/Z+#number,Rb	Any Rb, pointers Y and Z ; Copy data from Rb to (pointed by pointer address + #number), pointer doesn't change
LDS Rb,#number	Any Rb, #number is 16-bit ; Copy #number address of Ram to Rb
LDI Rb,#number	Rb: R16-R31,#number is 8-bit ; Copy the value of #number to Rb (immidiate)

Registers and Actions



SUBI Rb,#number	Rb 16-31, #number 8-bit; Αφαίρεση από το Rb τιμή #number ; **Αποτέλεσμα στον Rb**
SBCI Rb,#number	Rb 16-31, #number 8-bit; Αφαίρεση με κρατούμενο από το Rb τιμή #number ; **Αποτέλεσμα στον Rb**
ANDI Rb,#number	Rb 16-31, #number 8-bit; Λογικό AND του Rb με τιμή #number ; **Αποτέλεσμα στον Rb**
ORI Rb,#number	Rb 16-31, #number 8-bit; Λογικό OR του Rb με τιμή #number ; **Αποτέλεσμα στον Rb**
CPI Rb,#number	Rb 16-31, #number 8-bit; Σύγκριση του Rb με τιμή #number ; **Αποτέλεσμα ΠΟΥΘΕΝΑ !**
LDI Rb,#number	Rb 16-31, #number is 8-bit ; Copy the value of #number to Rb (immidiate)
SBR Rb,#number	Rb 16-31, #number 8-bit; Sets bits of Rb according to #number ; **Αποτέλεσμα στον Rb**
CBR Rb,#number	Rb 16-31, #number 8-bit; Clear bits of Rb according to #number ; **Αποτέλεσμα στον Rb**
SER Rb	Rb 16-31; Sets all bits of Rb ; **Αποτέλεσμα στον Rb**
MULS Rb,Rn	Rb 16-31, Rn 16-31; Multiplies (signed) Rn*Rb ; **Αποτέλεσμα στους R1:R0**
MULSU Rb,Rn	Rb 16-23(sign), Rn 16-23(unsign); Multiplies (signed) Rn*Rb ; **Αποτέλεσμα στους R1:R0**
FMUL Rb,Rn	Rb 16-23(uns 1.7), Rn 16-23(uns 1.7);Fractional multiplies (uns 1.15) Rn*Rb ; **Αποτέλεσμα στους R1:R0**
FMULS Rb,Rn	Rb 16-23(sig 1.7), Rn 16-23(sig 1.7);Fractional multiplies (sig 1.15) Rn*Rb ; **Αποτέλεσμα στους R1:R0**
FMULSU Rb,Rn	Rb 16-23(sig 1.7), Rn 16-23(uns 1.7);Fractional multiplies (sig 1.15) Rn*Rb ; **Αποτέλεσμα στους R1:R0**
ADIW Rb+1:Rb,#number	Rb 24-31, #number 8-bit; Add word Rb+1:Rb with #number ; **Αποτέλεσμα Rb+1:Rb**
SBIW Rb+1:Rb,#number	Rb 24-31, #number 8-bit; Sub word from Rb+1:Rb the #number ; **Αποτέλεσμα Rb+1:Rb**
CBI #num1,#num2	#num1 0-31, #num2 is 8-bit ; Clear bits of #num1 I/O according to #num2;
SBI #num1,#num2	#num1 0-31, #num2 is 8-bit ; Set bits of #num1 I/O according to #num2;
SBIC #num1,#num2	#num1 0-31, #num2 is 8-bit ; Skips a command if #num1 I/O bits (according to #num2) are cleared
SBIS #num1,#num2	#num1 0-31, #num2 is 8-bit ; Skips a command if #num1 I/O bits (according to #num2) are set

Jump Conditions



Condition	Signed	Unsigned
Rd > Rs	BREQ no BRGE yes no: jump out yes:	BREQ no BRSH yes no: jump out yes:
Rd >= Rs	BRGE yes	BRSH yes or BRCC yes
Rd == Rs	BREQ yes	BREQ yes
Rd != Rs	BRNE yes	BRNE yes
Rd <= Rs	BRLT yes BREQ yes jump out yes:	BRLO yes BREQ yes jump out yes:
Rd <Rs	BRLT yes	BRLO yes or BRCS yes