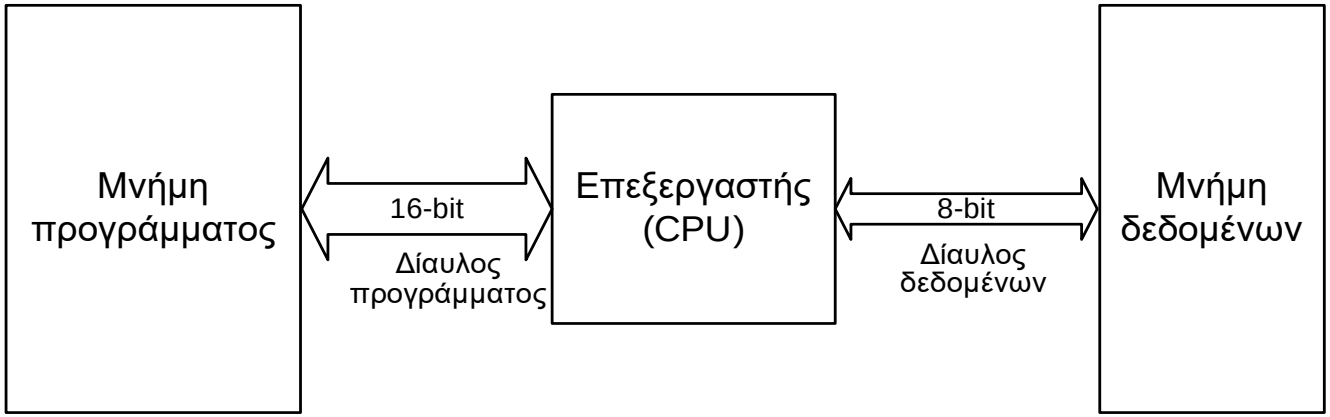
Μικροελεγκτές AVR

1η ΕΝΟΤΗΤΑ:

Η ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΩΝ ΜΙΚΡΟΕΛΕΓΚΤΩΝ AVR

Ε.Μ.Π.
Εργ. Μικροϋπολογιστών & Ψηφιακών Συστημάτων
Υπεύθυνος: Κ. ΠΕΚΜΕΣΤΖΗ Καθ.

Αρχιτεκτονική HARVARD



Ο μικροελεγκτής AVR περιέχει έναν επεξεργαστή RISC (Reduced Instruction Set Computer) ο οποίος έχει σχεδιαστεί με βάση την αρχιτεκτονική Harvard. Στην αρχιτεκτονική Harvard υπάρχει διαφορετικός δίαυλος για τη μεταφορά δεδομένων (data bus) και διαφορετικός δίαυλος για τη μεταφορά των εντολών (instruction bus).

Αρχιτεκτονική HARVARD

Διαφορετικός δίαυλος για τη μεταφορά:

- εντολών (instruction bus) και
- δεδομένων (data bus)

Περιλαμβάνει δύο διαφορετικές μνήμες:

- μνήμη προγράμματος (program memory) και
- μνήμη δεδομένων (data memory)

Επιτρέπει οι εντολές να έχουν διαφορετικό μήκος από τα δεδομένα ανάλογα με το πλήθος των εντολών
Επιτρέπει επικάλυψη λειτουργιών ανάκλησης εντολής, ανάγνωσης-εγγραφής δεδομένων, εκτέλεσης εντολών.

Οικογένειες του μΕ AVR

Μοντέλο	Ακροδέκτες	Flash	EEPROM	RAM	UART	ADC
90S1200	20	1K	64 bytes	0	Όχι	Όχι
90S2313	20	2K	128	128	Ναι	Όχι
90S4433	28	4K	256	128	Ναι	Ναι
90S4414	40	4K	256	256	Ναι	Όχι
90S2343	8	2K	128	128	Όχι	Όχι
Mega103	64	128K	4096	4096	Ναι	Ναι
Mega16	40	16K	512	1024	Ναι	Ναι
Mega603	64	64K	2048	4096	Ναι	Ναι
Tiny10	8	1K	64	0	Όχι	Όχι
Tiny13	8	2K	128	128	Όχι	Όχι

Η οικογένεια των μικροελεγκτών AVR έχει παρόμοιο ρεπερτόριο εντολών και βασική αρχιτεκτονική. Η διαφοροποίηση μεταξύ των διάφορων μελών της οικογένειας είναι στον αριθμό των ακροδεκτών, στο μέγεθος της μνήμης (FLASH, EEPROM, SRAM) καθώς και στα περιφερειακά τα οποία χρησιμοποιούνται.

Αρχιτεκτονική του μικροελεγκτή AVR (ATmega16)

Πυρήνας

• υψηλής απόδοσης, χαμηλής κατανάλωσης επεξεργαστής **CPU**

• αρχιτεκτονική **RISC**

– 131 ισχυρές εντολές

– 32 x 8 καταχωρητές

– Έως 16 MIPS απόδοση στα 16 MHz

▪ Μονάδα συνεχούς διοχέτευσης εντολών (Instruction Pipelining)

▪Αριθμητική Λογική Μονάδα (**A**rithmetic **L**ogic **U**nit, ALU)

• μνήμη προγράμματος και δεδομένων

– 16K Bytes από **Flash** (In-System Self-Programmable)

– Προαιρετικό τμήμα Boot Code με ανεξάρτητα bits κλειδώματος (Lock Bits)

– 512 Bytes **EEPROM** (Αντοχή: 100,000 κύκλοι εγγραφής/ανάγνωσης)

– 1K Byte εσωτερικής μνήμης **SRAM** και **στοίβα**

– Programming Lock για ασφάλεια λογισμικού

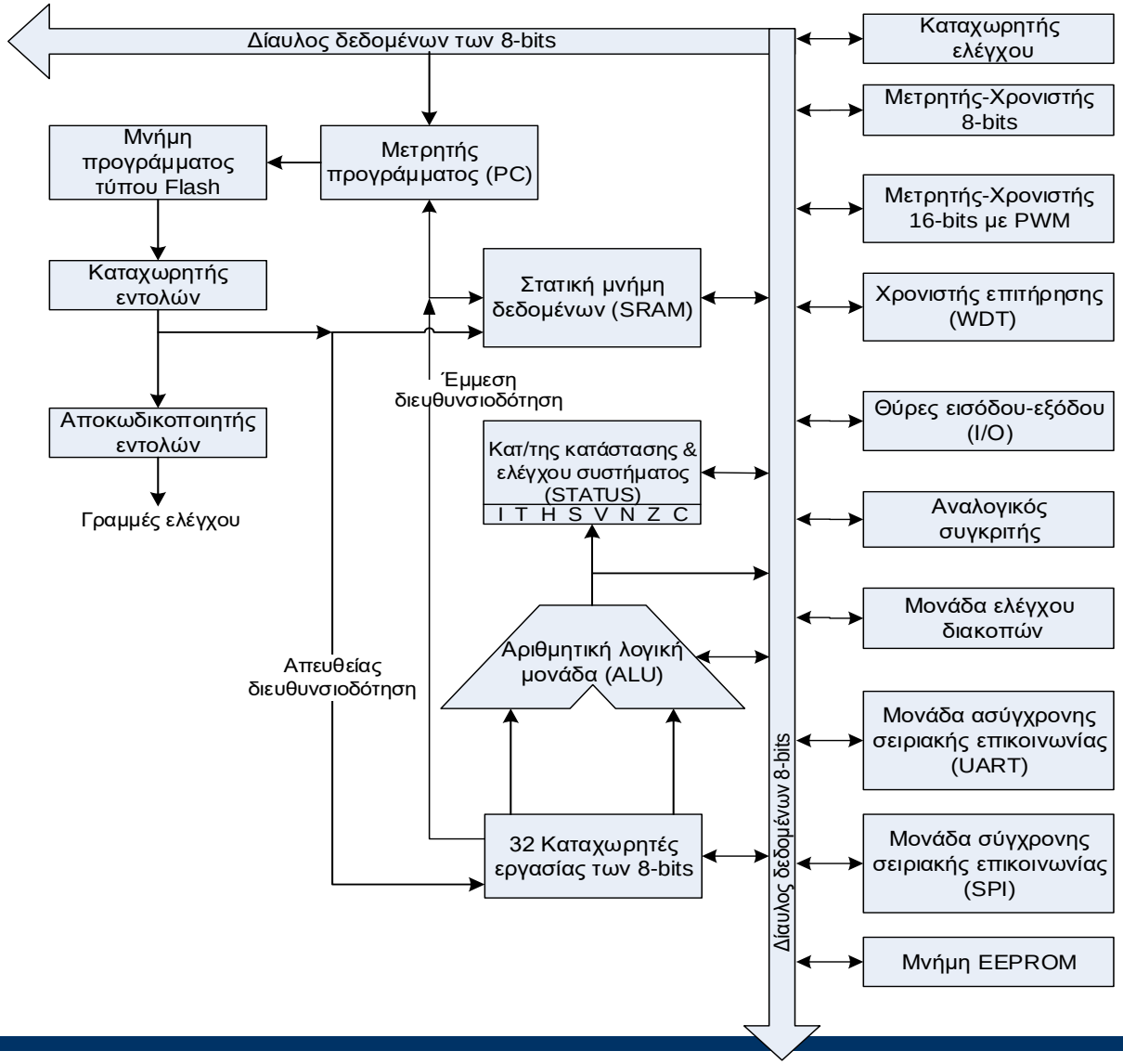
• **JTAG διεπαφή**

– Προγραμματισμός των Flash, EEPROM, ασφαλειών, και Bits κλειδώματος μέσω της διεπαφής JTAG

Αρχιτεκτονική του μικροελεγκτή AVR (ATmega16)

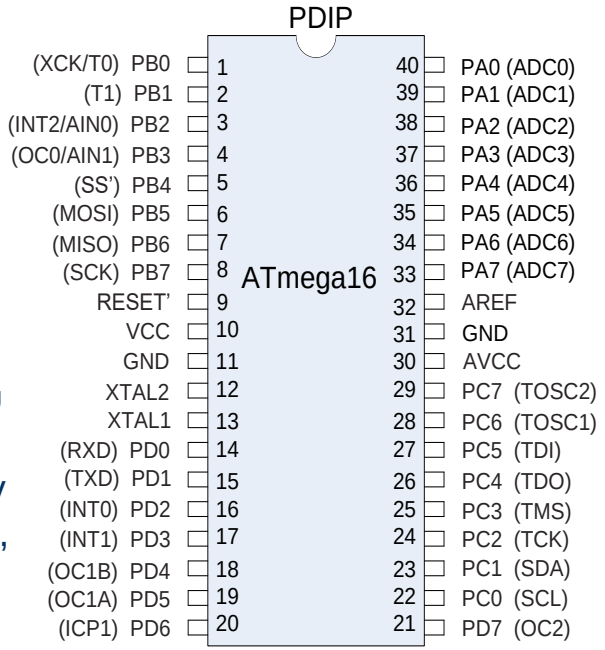
• **Περιφερειακά**
– Δύο 8-bit Timer/Counters με ξεχωριστές μονάδες διαίρεσης χρόνου (Prescalers) και τρόπους σύγκρισης (Compare Modes)
– Ένα 16-bit Timer/Counter με ξεχωριστή μονάδα διαίρεσης χρόνου (Prescaler), τρόπο σύγκρισης, και σύλληψης (Capture Mode)
– Μετρητή πραγματικού χρόνου με ανεξάρτητο ταλαντωτή
– 4 PWM κανάλια
– 8-κάναλο, 10-bit ADC
–Two-wire σειριακή διεπαφή TWI (IIC)
– Προγραμματιζόμενη σειριακή USART
– Master/Slave SPI σειριακή διεπαφή
– Προγραμματιζόμενος χρονιστής επιτήρησης (Watchdog Timer)
– Αναλογικός συγκριτής
• **Ειδικά χαρακτηριστικά**
– Power-on Reset και προγραμματιζόμενο κύκλωμα ανίχνευσης βύθισης τάσης τροφοδοσίας(Brown-out)
– εσωτερικός RC ταλαντωτής
– Εξωτερικές και εσωτερικές πηγές διακοπών
– 6 λειτουργίες ηρεμίας: Idle, ADC Noise Reduction, Power-save, Power-down, Standby
• Τάση λειτουργίας (4.5 - 5.5V για ATmega16)
• Ταχύτητα λειτουργίας (0 - 16 MHz για ATmega16)
• Κατανάλωση ισχύος για 1 MHz, 3V, και 25°C του ATmega16L
– δραστήρια κατάσταση: 1.1 mA– Power-down κατάσταση: < 1 μA– άεργη κατάσταση:0.35 mA

Αρχιτεκτονική του μικροελεγκτή AVR



Ακροδέκτες του μικροελεγκτή AVR (ATmega16)

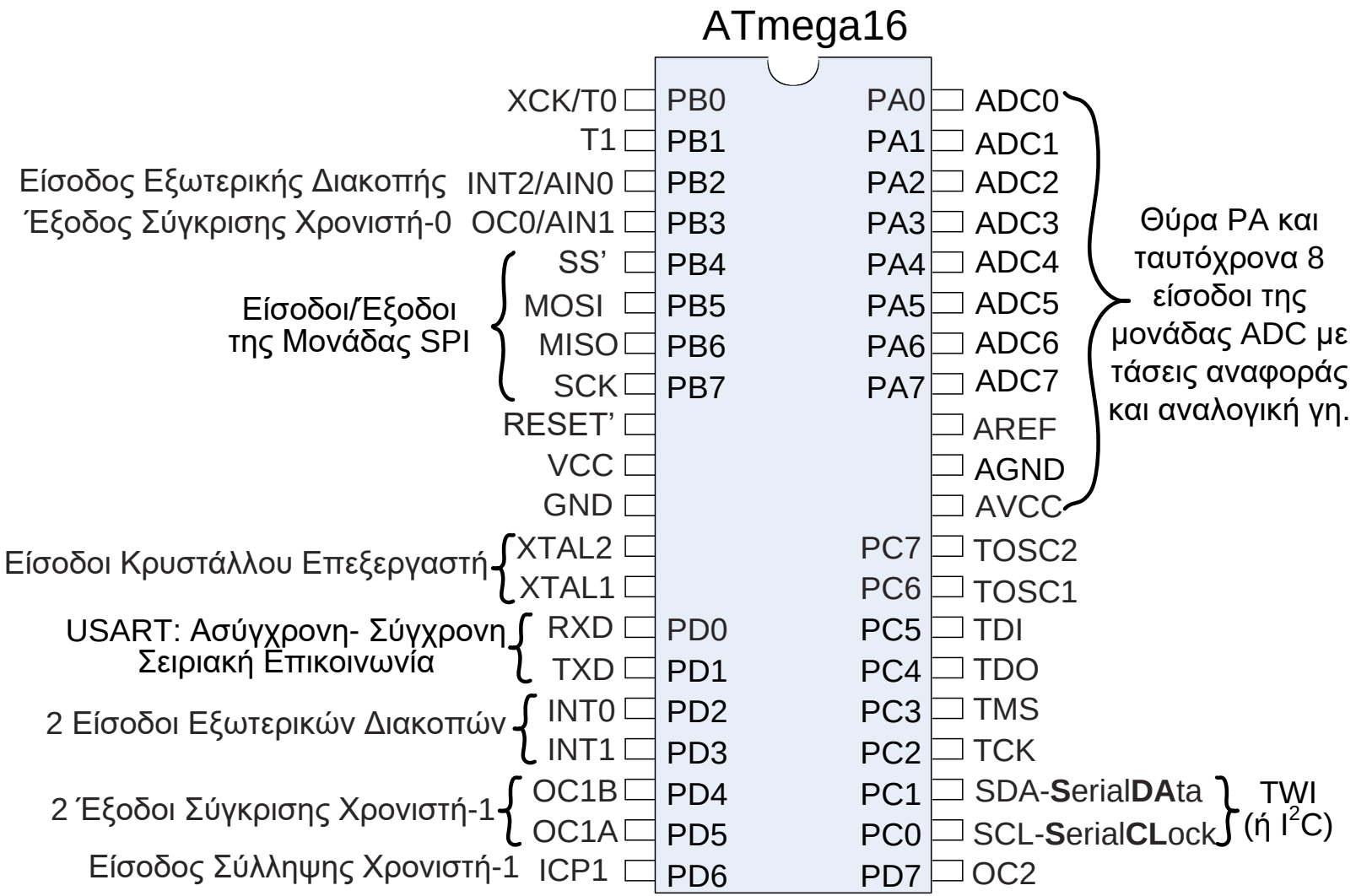
VCC Ψηφιακή τροφοδοσία τάσης.
GND Γείωση.
Port A (PA7..PA0) Η Port A λειτουργεί είτε ως αναλογική είσοδος του μετατροπέα A/D είτε
ως θύρα εισόδου-εξόδου διπλής κατεύθυνσης των 8-bit.
Port B (PB7..PB0) Η Port B λειτουργεί ως θύρα εισόδου-εξόδου διπλής κατεύθυνσης των 8-bit
Port C (PC7..PC0) Η Port C λειτουργεί ως θύρα εισόδου-εξόδου διπλής κατεύθυνσης των 8-bit
Port D (PD7..PD0) Η Port D λειτουργεί ως θύρα εισόδου-εξόδου διπλής κατεύθυνσης των 8-bit).
Επίσης, οι θύρες εκτελούν κάποιες ιδιαίτερες λειτουργίες του μικροελεγκτή.
RESET Έμφάνιση χαμηλού επιπέδου στον ακροδέκτη αυτόν για χρόνο μεγαλύτερο από τον ελάχιστο μήκος παλμού(2μs), θα παράγει reset.
XTAL1 Είσοδος στον ταλαντωτή-ενισχυτή και στο κύκλωμα εσωτερικού ρολογιού.
XTAL2 Έξοδος από ταλαντωτή-ενισχυτή.
AVCC είναι ο ακροδέκτης τροφοδοσίας τάσης για την Port A και τον μετατροπέα A/D.
AREF είναι ο *analog reference* ακροδέκτης για τον μετατροπέα A/D.



Περιφερειακά του μΕ AVR

- Σειριακή μονάδα USART
- 8-bit Timer/Counter με (Prescaler) και λειτουργία σύγκρισης (Compare Mode)
- 16-bit Timer/Counter με (Prescaler), λειτουργία σύγκρισης, σύλληψης (Capture Mode), 4 PWM κανάλια
- Χρονιστής επιτήρησης (Watchdog Timer)
- Μνήμη EEPROM
- Master/Slave SPI σειριακή διεπαφή
- Two-wire σειριακή διεπαφή TWI
- 8-κάναλο, 10-bit ADC
- Αναλογικός συγκριτής

Ακροδέκτες περιφερειακών ATmega16



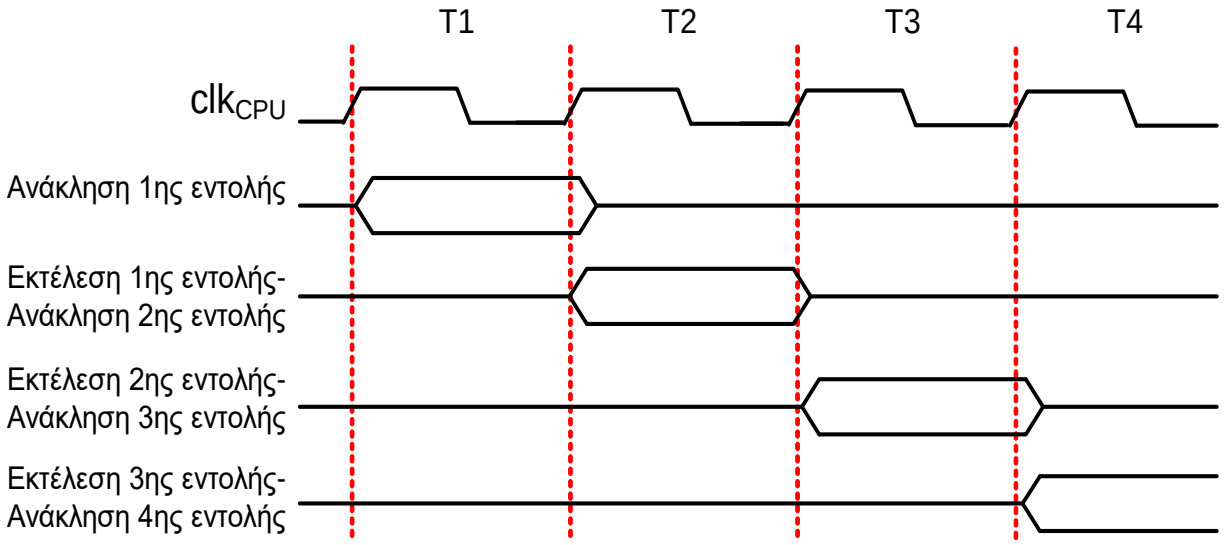
Θύρες- καταχωρητές των μικροελεγκτών AVR

Μονάδα	Καταχωρητής	Σύμβολο
Timer 0	Timer/Counter 0 Control Register	TCCR0
	Timer/Counter 0	TCNT0
Timer 1	Timer/Counter Control Register1A	TCCR1A
	Timer/Counter Control Register1B	TCCR1B
	Timer/Counter 1	TCNT1
	Output Compare Register 1 A	OCR1A
	Output Compare Register 1 B	OCR1B
	Input Capture Register	ICR1L - H
Timer 2	Timer/Counter2 (8 Bits)	TCNT2
	Timer/Counter Control Register	TCCR2
	Timer/Counter2 Output Compare Register	OCR2
	Asynchronous Status Register	ASSR
Watchdog Timer	Watchdog Timer Control Register	WDTCR
UART	UART Data Register	UDR
	UART Status Register	USR
	UART Control Register	UCR
	UART Baud Rate Register	UBRR

Θύρες-καταχωρητές των μικροελεγκτών AVR

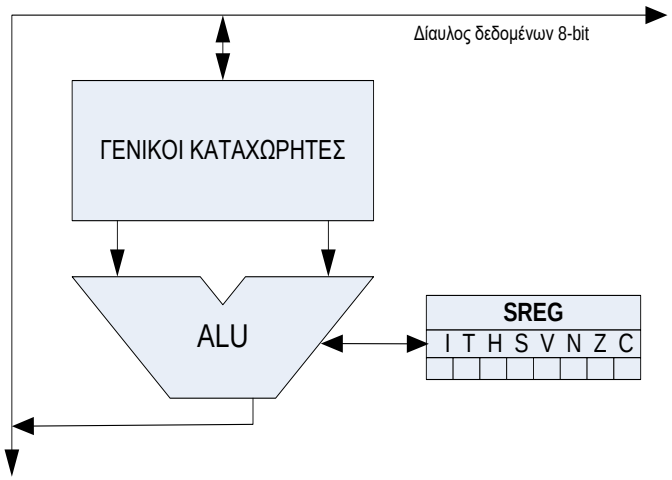
Μονάδα	Καταχωρητής	Σύμβολο
EEPROM	EEPROM Adress Register	EEAR
	EEPROM Data Register	EEDR
	EEPROM Control Register	EECR
SPI	Serial Peripheral Control Register	SPCR
	Serial Peripheral Status Register	SPSR
	Serial Peripheral Data Register	SPDR
Analog Comparator	Analog Comparator Control and Status Register	ACSR
Two-wire Serial (I^2C) Interface TWI	TWI Data Register	TWDR
	TWI Address Register	TWAR
	TWI Bit Rate Register	TWBR
	TWI Control Register	TWCR
	TWI Status Register	TWSR
ADC	Πολυπλέκτης επιλογής 8 γραμμών εισόδου ADC Multiplexer Selection Register	ADCMUX
	ADC Control and Status Register A	ADCSRA
	ADC Data Register (high and low)	ADCH - L

Παράδειγμα λειτουργίας του PIPELINE

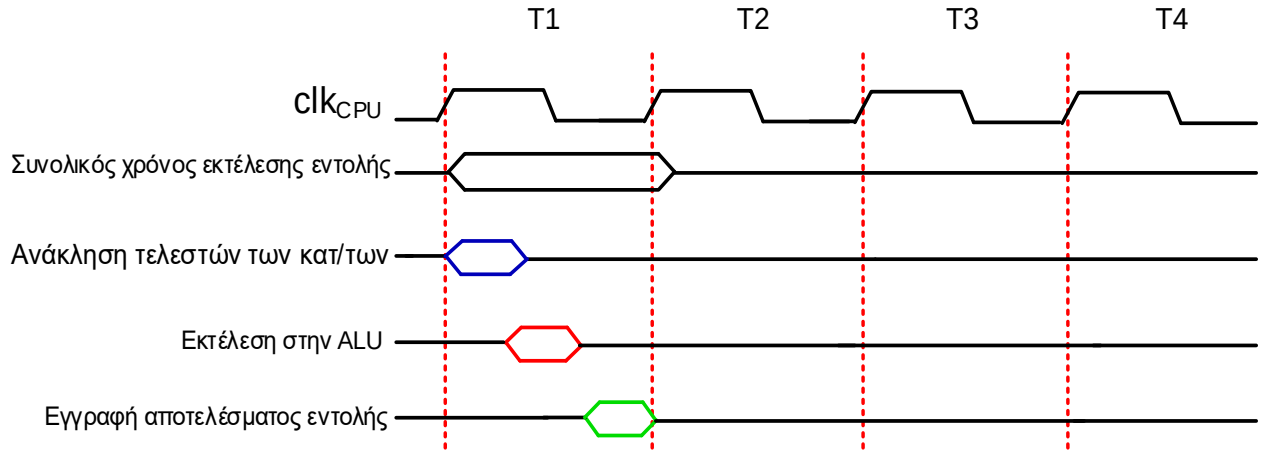


Στη λειτουργία Pipeline έχουμε συνεχή διοχέτευση εντολών με αποτέλεσμα την παράλληλη ανάκληση και εκτέλεση εντολών. Οπότε, η εκτέλεση των περισσοτέρων εντολών γίνεται σε μία περίοδο του κεντρικού ρολογιού με αποτέλεσμα απόδοση έως 1 MIPS ανά MHz.

Αριθμητική Λογική Μονάδα (ALU) του AVR



Είναι η κεντρική μονάδα του επεξεργαστή. Αυτή εκτελεί αριθμητικές, λογικές και σε επίπεδο bit εντολές στη διάρκεια μιας περιόδου και αποθηκεύει το αποτέλεσμα στον καταχωρητή που δείχνει η εντολή. Κατά την εκτέλεση των πράξεων ενημερώνονται οι σημαίες του καταχωρητή κατάστασης (Status register).



Διαδικασία εκτέλεσης εντολής στην ALU

Καταχωρητής κατάστασης (SREG) STATUS REGISTER

Bit	7	6	5	4	3	2	1	0
Σημαία	I	T	H	S	V	N	Z	C

Bit7 **(I):** καθολική **ενεργοποίηση διακοπών** (Global Interrupt Enable-GIE). Θέτοντας το bit αυτό ενεργοποιούμε τις διακοπές

Bit6 **(T):** σημαία **αντιγραφής-αποθήκευσης (bit** Copy-Storage). Με χρήση των εντολών BLD και BST επιτυγχάνουμε την ανάγνωση και αποθήκευση συγκεκριμένων bits

Bit5 **(H):** σημαία **δεκαδικού κρατουμένου** (Half Carry flag). Ενημερώνει για την ύπαρξη δεκαδικού κρατουμένου μετά από αριθμητικές πράξεις

Bit4 **(S):** σημαία **προσήμου** (Sign flag). Ενημερώνει για το πρόσημο ενός καταχωρητή και ισούται με το XOR των σημαιών αρνητικού προσήμου και υπερχείλισης.(S=N xor V) Όταν έχουμε υπερχείλιση αποτέλεσμα που εμφανίζεται >0 είναι στην πραγματικότητα <0

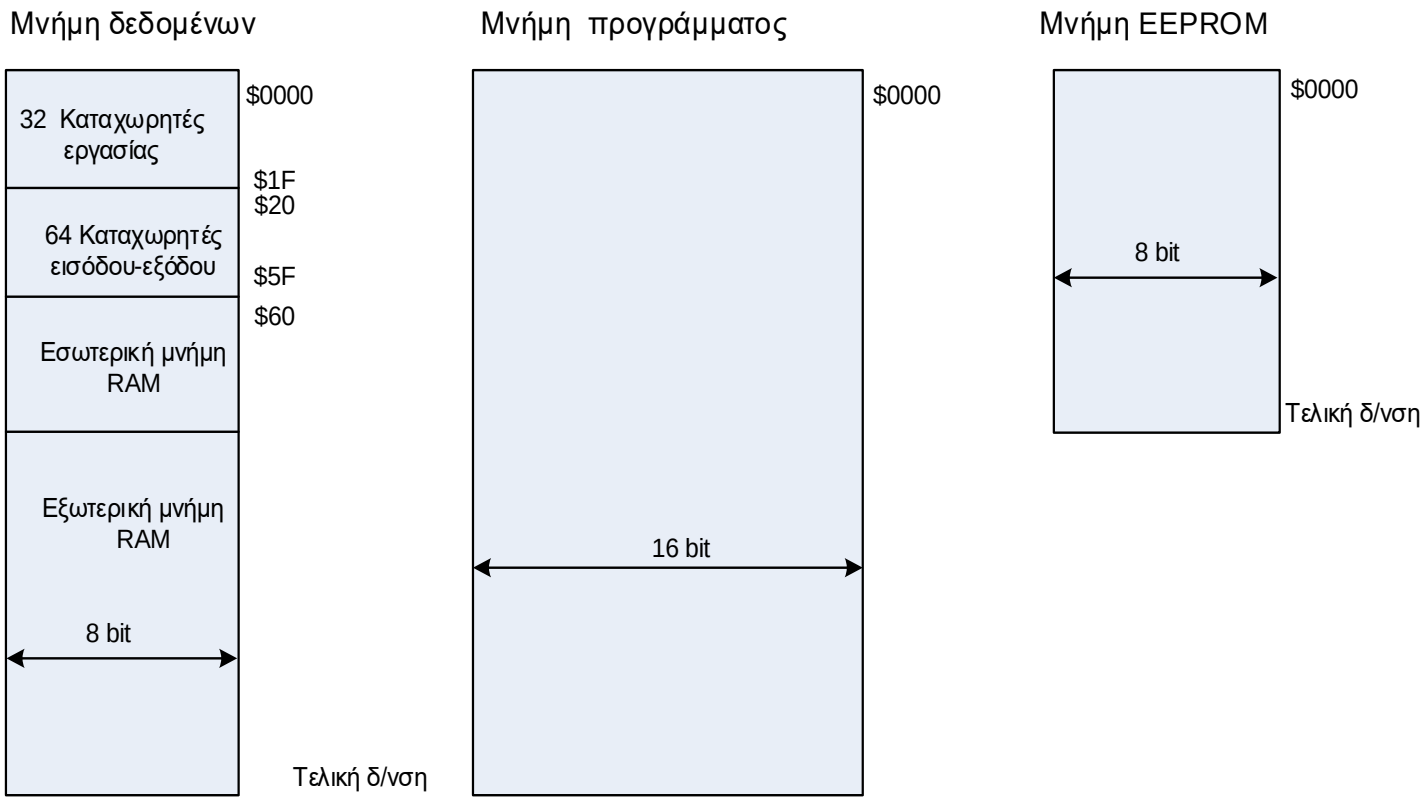
Bit3 **(V):** σημαία **υπερχείλισης** (Overflow flag) για αριθμητική συμπληρώματος του δύο

Bit2 **(N):** σημαία **αρνητικού προσήμου** (negative flag)

Bit1 **(Z):** σημαία **μηδενισμού** (Zero flag).Τίθεται όταν το αποτέλεσμα μιας πράξης είναι 0

Bit0 **(C):** σημαία **κρατουμένου** (Carry flag).

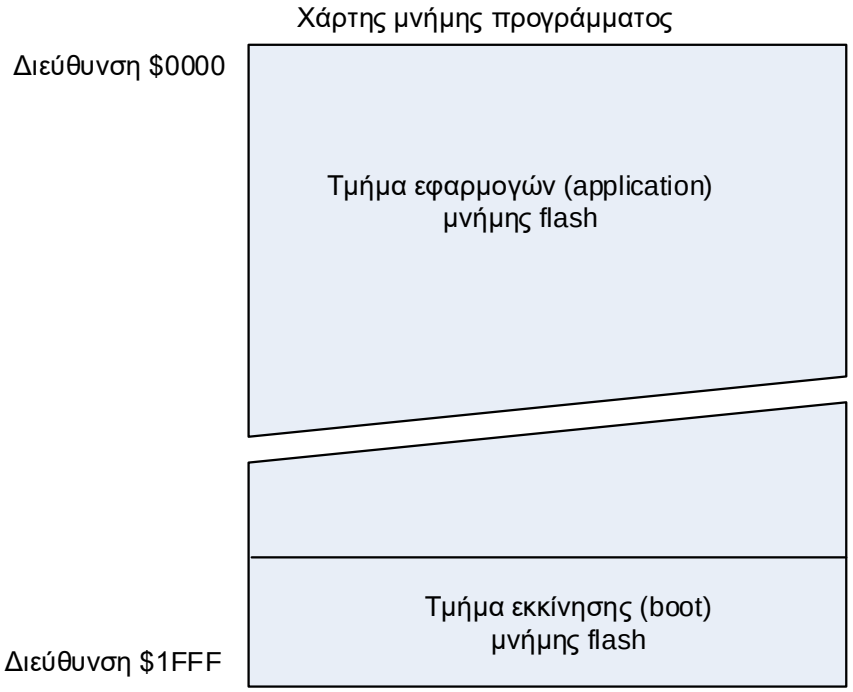
Τύποι μνήμης του μικροελεγκτή AVR



Ένας μικροελεγκτής διαθέτει τα εξής είδη μνήμης:

- μνήμη δεδομένων (1 Kbyte)
- μνήμη προγράμματος τύπου flash (16KByte)
- μνήμη EEPROM με αρχική διεύθυνση $0000 και χωρητικότητα από 64bytes ως 4Kbytes.

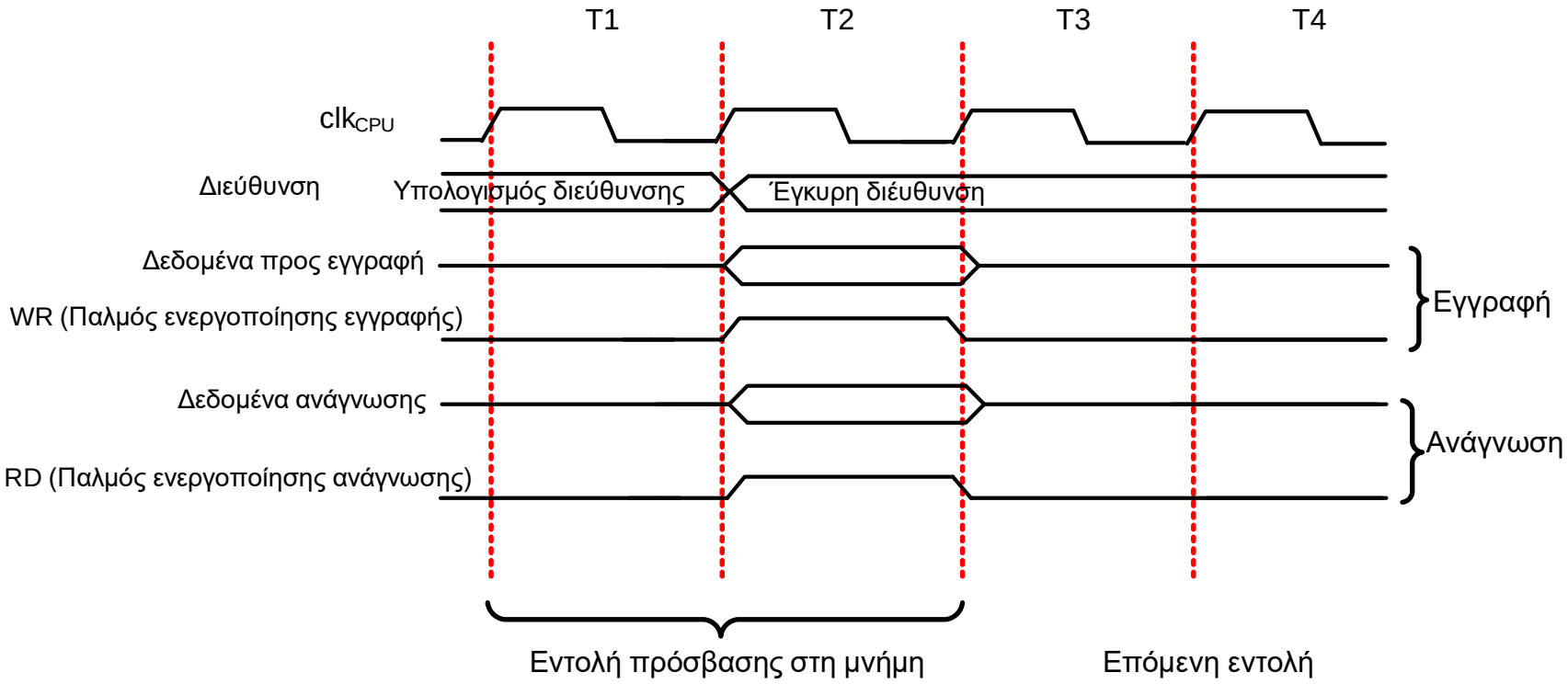
Μνήμη προγράμματος του μΕ AVR



- τύπου flash για ταχεία αποθήκευση-φόρτωση δεδομένων
- σύνδεση με δίαυλο των 16-bit για την τροφοδοσία του καταχωρητή εντολών.
- αποθήκευση εντολών και διανυσμάτων διακοπών (interrupt vectors).

Όλες οι εντολές έχουν μήκος 16 bits ή 32 bits (2 bytes=1 λέξη), άρα καταλαμβάνουν μία ή δύο θέσεις της μνήμης προγράμματος, αφού η μνήμη flash είναι οργανωμένη σε διάταξη 8Kx16. Ο μετρητής προγράμματος του ATmega16 έχει μήκος 13 bits ώστε να επιτυγχάνει πρόσβαση σε 8K θέσεις μνήμης προγράμματος.

Φάσεις ανάγνωσης-εγγραφής στην εσωτερική μνήμη SRAM



Μνήμη δεδομένων του μΕ AVR

ΜΝΗΜΗ ΔΕΔΟΜΕΝΩΝ								
Καταχ/τές εργασίας (00-1F)	R0	R1	R2	R3	R4	R5	R6	R7
	R8	R9	R10	R11	R12	R13	R14	R15
	R16	R17	R18	R19	R20	R21	R22	R23
	R24	R25	R26	R27	R28	R29	R30	R31
Καταχ/τές I/O (20-5F) {00-3F}	SREG	PORTA	TCCR0	EEAR				
	SPL	DDRA	TCNT0	EEDR				
	SPH	PINA		EECR				
		PORTB	TCCR1A					
	GIMSK	DDRB	TCCR1B					
	GIFR	PINB	TCNT1H	UDR				
		PORTC	TCNT1L	USR				
	TIMSK	DDRC	OCR1AH	UCR				
	TIFR	PINC	OCR1AL	UBRR				
		PORTD	ICR1H					
	MCUCR	DDRD	ICR1L	ACSR				
		PIND	WDTCR					
SRAM ($60-$45F)	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF

Η μνήμη δεδομένων συνδέεται με ένα δίαυλο των 8-bit για την επικοινωνία των περιφερειακών με τους κατ/τές ελέγχου.

Η μνήμη δεδομένων χωρίζεται στα τμήματα:

- 32 καταχωρητές εργασίας (register file) των 8-bits (R0-R31)
- 64 καταχωρητές εισόδου-εξόδου των 8-bits
- εσωτερική μνήμη SRAM (για αποθήκευση μεταβλητών και ως στοίβα (stack))
- εξωτερική SRAM (1KB).

Εντολές του μΕ AVR

Οι εντολές της γλώσσας των μικροελεγκτών μπορούν να ομαδοποιηθούν στις παρακάτω τέσσερις κατηγορίες :

- μεταφοράς δεδομένων
- αριθμητικών και λογικών πράξεων
- σε επίπεδο bit και ελέγχου bit
- ελέγχου ροής του προγράμματος και διακλάδωσης

Οι εντολές περιλαμβάνουν 2 τμήματα. Το πρώτο (operation code ή opcode) αποτελεί το λειτουργικό κώδικα που πληροφορεί τον επεξεργαστή για τις ενέργειες που πρέπει να εκτελεστούν. Το δεύτερο τμήμα προσδιορίζει τους τελεστές (r, operand) για τους οποίους θα λειτουργήσει ο κώδικας.

Καταχωρητές X, Y, Z έμμεσης διευθυνσιοδότησης

X (XH:XL, R27:R26), **Y** (YH:YL, R29:R28) και **Z** (ZH:ZL, R31:R30). Επιτρέπουν πρόσβαση στη θέση που δείχνουν (π.χ. ST X, R1), μετά από μείωση της διεύθυνσης (π.χ. ST -X, R1) ή μετέπειτα αύξηση της διεύθυνσης (π.χ. ST X+, R1)

Εντολές μεταφοράς δεδομένων

Πρόκειται για εντολές μεταφοράς δεδομένων μεταξύ καταχωρητών ή μεταξύ καταχωρητών και μνήμης ή σταθεράς σε καταχωρητή. Οι σημαίες δεν επηρεάζονται. Οι Rd, Rr ανήκουν στους 32 καταχωρητές εργασίας R0-R31.

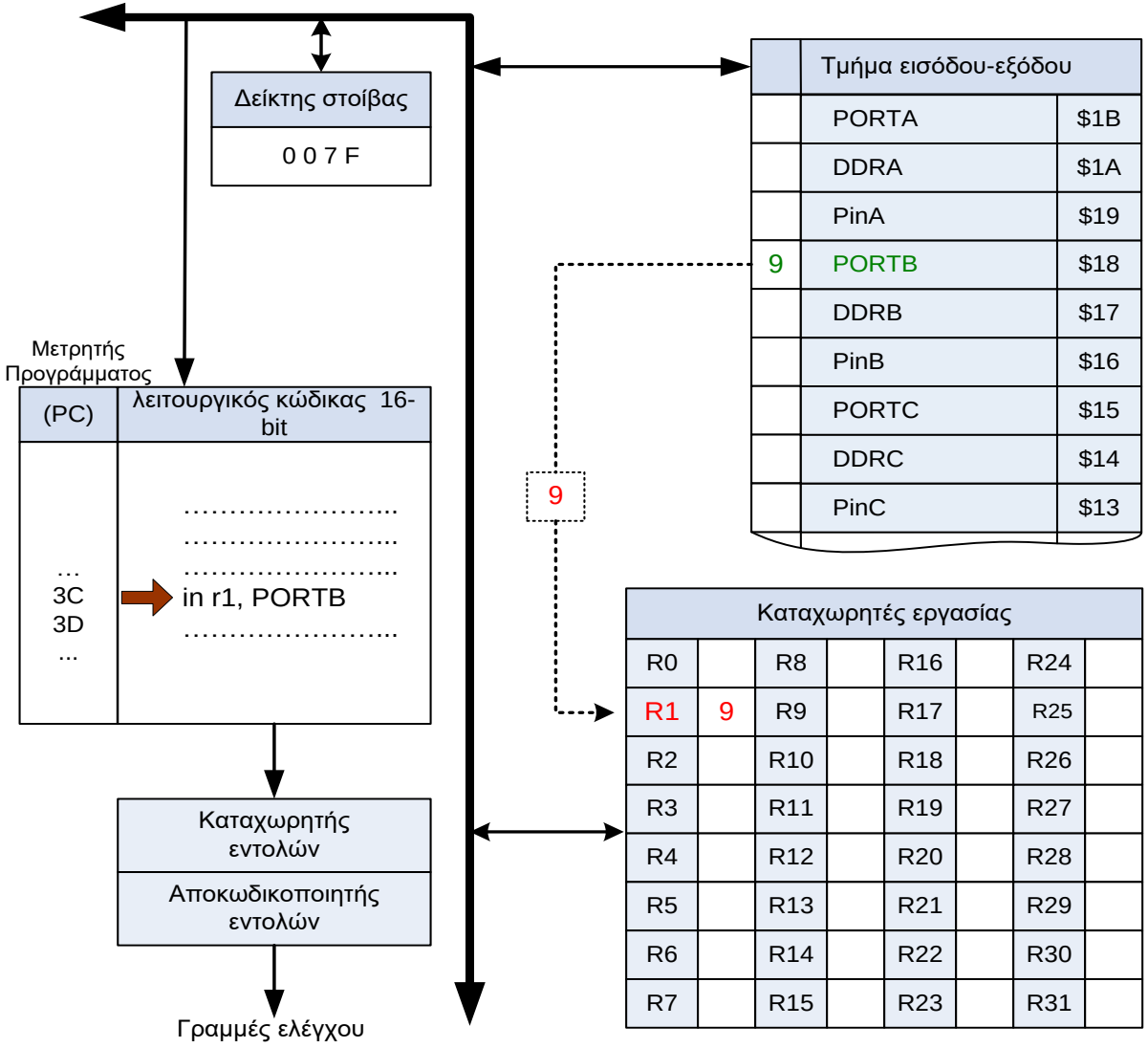
MOV	Rd, Rr	Move Between Registers	Rd← Rr
LDI	Rd, K	Load Immediate	Rd ← K
LD	Rd, X	Load Indirect	Rd ← (X)
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)
ST	X, Rr	Store Indirect	(X) ← Rr
STS	k, Rr	Store Direct to SRAM	(k) ← Rr
LPM		Load Program Memory	R0 ← (Z)
LPM	Rd, Z	Load Program Memory	Rd ← (Z)
SPM		Store Program Memory	(Z) ← R1:R0
IN	Rd, P	In Port(I/O register)	Rd ← P
OUT	P, Rr	Out Port(I/O register)	P ← Rr
PUSH	Rr	Push Register on Stack	STACK ← Rr
POP	Rd	Pop Register from Stack	Rd ← STACK

Παραδείγματα έμμεσης διευθυνσιοδότησης

X (XH:XL, R27:R26), Y (YH:YL, R29:R28) και Z (ZH:ZL, R31:R30). Επιτρέπουν πρόσβαση στη θέση που δείχνουν (π.χ. ST X, R1), μετά από μείωση της διεύθυνσης (π.χ. ST -X, R1) ή αύξηση της διεύθυνσης μετά την εκτέλεση της εντολής (π.χ. ST X+, R1).

```
; εγγραφή
    ldi r28,$60     ; Η αρχική διεύθυνση είναι $0060 και τοποθετείται στον
    clr r29         ; καταχωρητή Y σαν δείκτης διεύθυνσης.
    st y+,data      ; Αποθήκευση περιεχομένων του καταχωρητή data
                    ; στην SRAM και αύξηση της διεύθυνσης.
; ανάγνωση
    ldi r28,$60     ; Η αρχική διεύθυνση είναι $0060 και τοποθετείται στον
    clr r29         ; καταχωρητή Y σαν δείκτης διεύθυνσης.
    ld data,y+      ; Μεταφορά δεδομένων στον καταχωρητή data από την
                    ; SRAM και αύξηση της διεύθυνσης της SRAM.
```

Παράδειγμα μεταφοράς δεδομένων

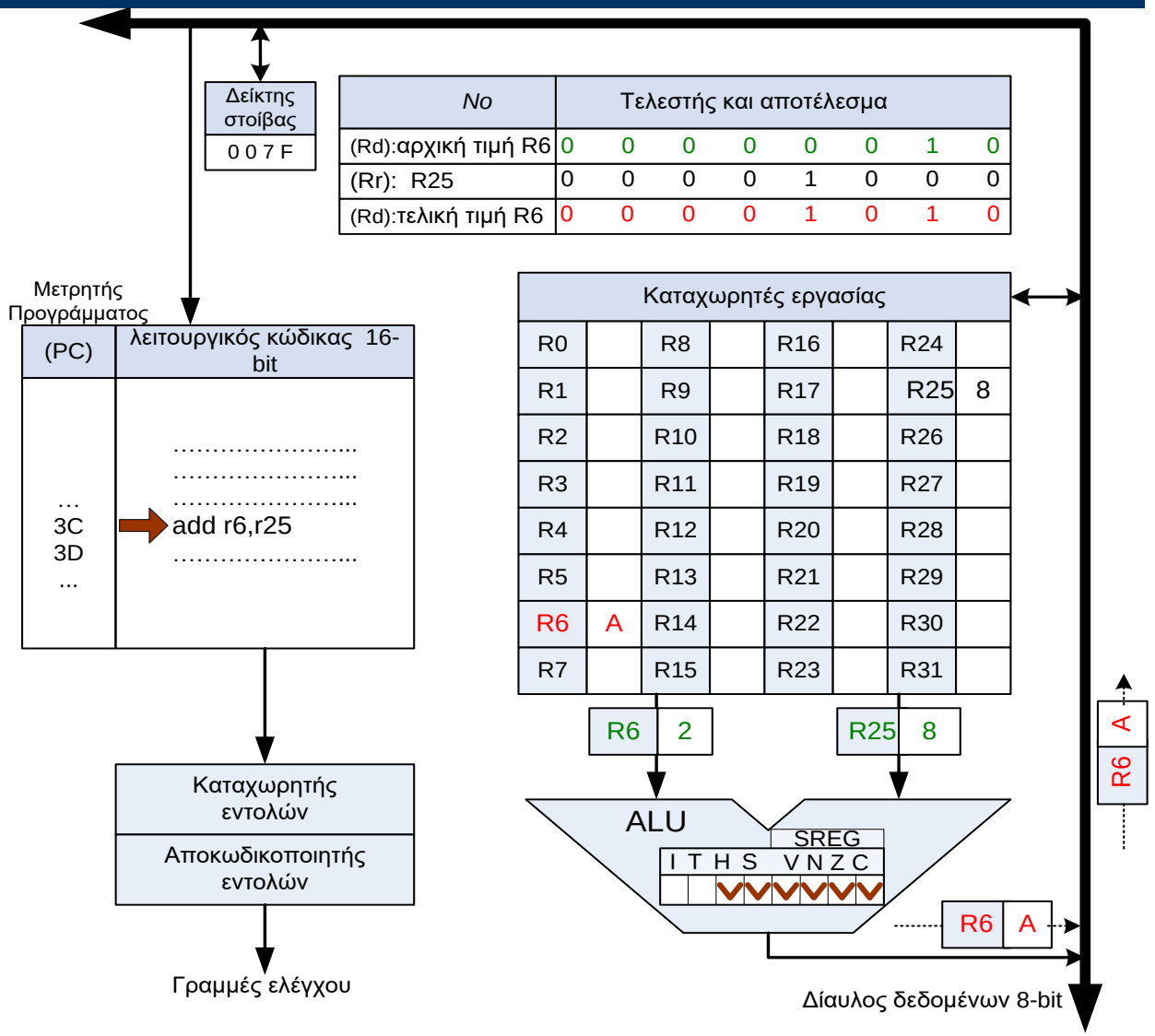


Εντολές αριθμητικών και λογικών πράξεων

Προϋποθέτουν τη χρήση της αριθμητικής λογικής μονάδας σε αντίθεση με τις εντολές μεταφοράς. Η εντολή διαβάζει τα περιεχόμενα του καταχωρητή, εκτελεί τις πράξεις με αυτά και αποθηκεύει το αποτέλεσμα στον ίδιο ή σε άλλον καταχωρητή.

ADD /SUB	Rd, Rr	Add/ Subtract two Registers	Rd ← Rd ± Rr
ADC /SBC	Rd, Rr	Add with Carry two Regs	Rd ← Rd ± (Rr + C)
SUBI	Rd, K	Subtract Constant from Reg	Rd ←Rd – K
ADIW/ SBIW	Rd, K	add/subtract Immediate to word	Rd+1:Rd←Rd+1:Rd ±K(0,63)
AND/OR/ EOR	Rd, Rr	AND/OR/XOR Regs	Rd ← Rd ∩/v/⊕ Rr
ANDI/ ORI	Rd, K	AND/OR Reg & Constant	Rd ← Rd ∩/v K
COM	Rd	One's Complement	Rd ← $FF ⊕ Rd
SBR/CBR	Rd,K	Set/ Clear Bit(s) in Reg	Rd ← Rd v K / ∩($FF – K)
INC /DEC	Rd	Increment /Decrement	Rd ← Rd ± 1
TST	Rd	Test for Zero or Minus	Rd ← Rd ∩ Rd
CLR /SER	Rd	Clear /Set Reg	Rd ← Rd ⊕ Rd / Rd ←$FF
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd × Rr

Παράδειγμα αριθμητικών και λογικών πράξεων

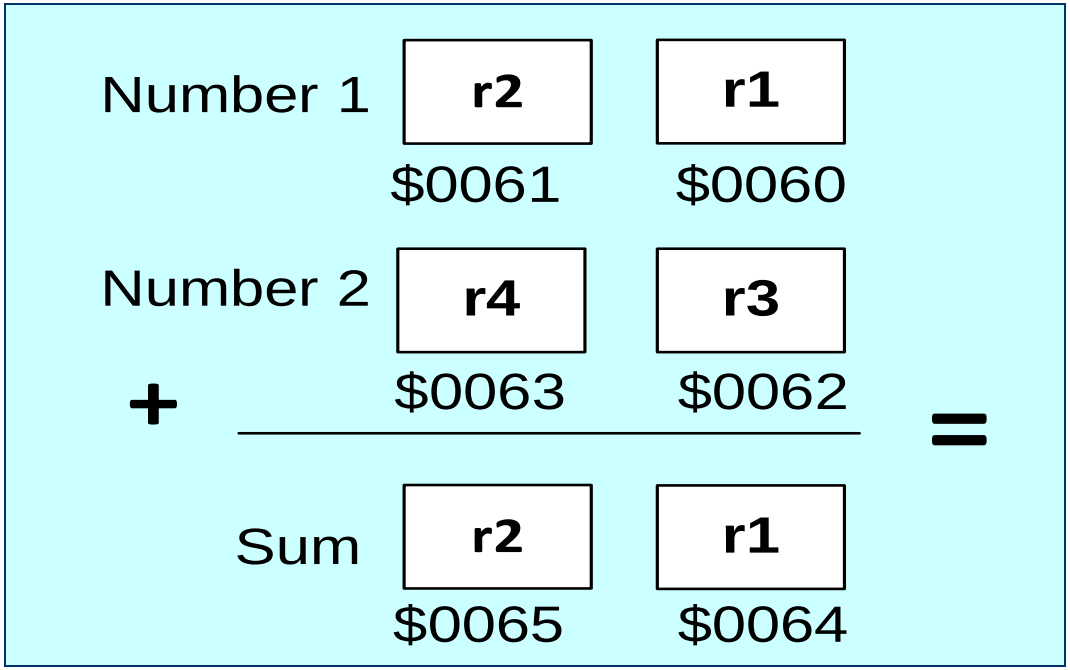


Παράδειγμα 1ο : Να προστεθούν δυο 16-bit αριθμοί, που βρίσκονται στις θέσεις μνήμης 0x0060-0x0061 και 0x0062-0x0063. Το αποτέλεσμα σώζεται στη διεύθυνση 0x0064-0x0065 της μνήμης δεδομένων.

```
.include "m16def.inc"        ; δήλωση μικροελεγκτή

.org 0x000
main:                        ; κυρίως πρόγραμμα
  ldi xl, 0x60               ; θέτουμε διεύθυνση 0x0060 στον καταχωρητή X
  clr xh                     ; όπου xl=R26, xh=R27
  ld r1,x+                   ; ανάγνωση θέσης μνήμης, αποθήκευση σε κατ/τη
                             ; και αύξηση δείκτη για επόμενη θέση
  ld r2,x+                   ; ανάγνωση 0x0061
  ld r3,x+                   ; ανάγνωση 0x0062
  ld r4,x+                   ; ανάγνωση 0x0063
  add r1,r3                  ; πρόσθεση λιγότερο σημαντικών byte
  adc r2,r4                  ; πρόσθεση περισσότερο σημαντικών byte με κρατούμενο
  st x+,r1                   ; αποθήκευση αποτελέσματος στη διεύθυνση
  st x,r2                    ; 0x0064 και 0x0065
 .exit
```

Πρόσθεση δυο 16-bit αριθμών

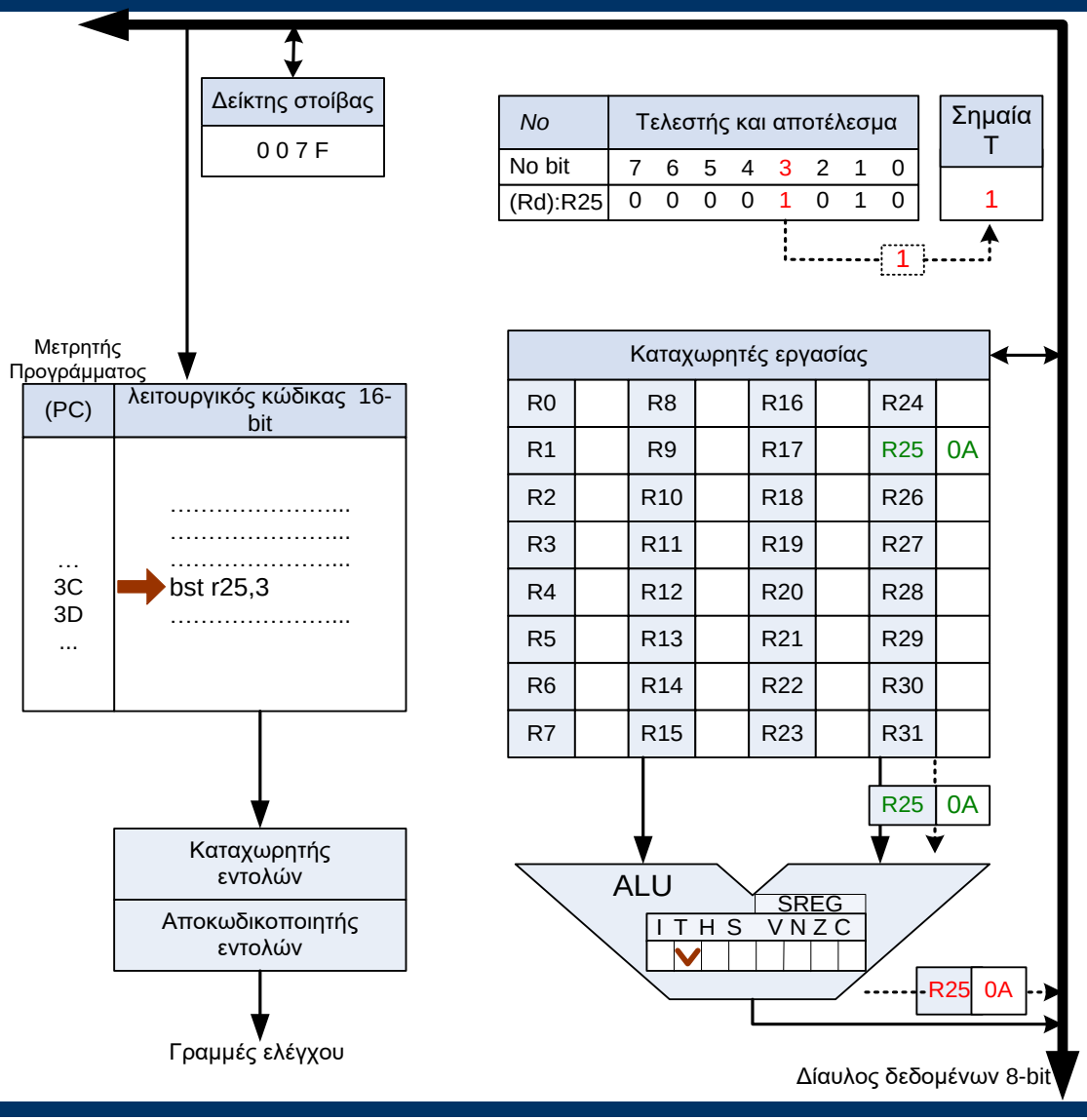


Εντολές σε επίπεδο bit και ελέγχου bit

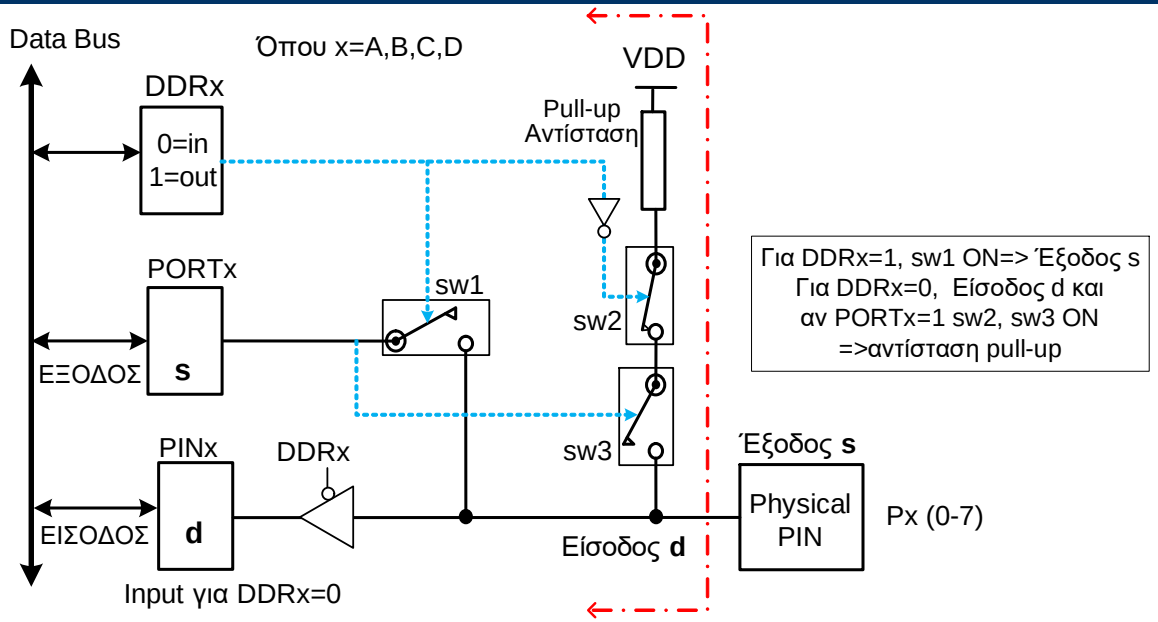
Οι εντολές αυτές μας παρέχουν τη δυνατότητα να θέσουμε ή να μηδενίσουμε σημαίες του καταχωρητή κατάστασης, διακοπές και συγκεκριμένα bit καταχωρητών ή θυρών. Επίσης, να περιστρέψουμε κάποιον καταχωρητή μέσω κρατουμένου ή όχι.

SBI/ CBI	P,b	Set/Clear Bit in I/O Reg	I/O(P,b) ← 1/ 0
LSL/ LSR	Rd	Logical Shift Left/ Right	Rd(n+1) ← Rd(n), Rd(0) ← 0
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)←Rd(n),C←Rd(7)
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)←Rd(n+1),C←Rd(0)
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6, Rd(7)=Rd(7)
SWAP	Rd	Swap Nibbles	Rd(3-0)←Rd(7-4) Rd(7-4)←Rd(3-0)
BSET/BCLR	s	Flag Set/Clear (s=0-7)	SREG(s) ← 1/ 0
BST	Rr, b	Bit Store from Reg to T	T ← Rr(b)
BLD	Rd, b	Bit load from T to Reg	Rd(b) ← T
SEx (π.χ. SEC)		Set Flag (x=I,T,H,S,V,N,Z,C)	x ← 1 όπου x=σημαία (SREG)
CLx (π.χ. CLT)		Clear Flag (x=I,T,H,S,V,N,Z,C)	x ← 0 όπου x=σημαία (SREG)

Παράδειγμα σε επίπεδο bit και ελέγχου bit



Θύρες εισόδου-εξόδου



Ο επεξεργαστής επικοινωνεί με τις διάφορες μονάδες γράφοντας ή διαβάζοντας στις θύρες-καταχωρητές: PORTA, PORTB, PORTC και PORTD.

Οι ακροδέκτες είναι διπλής κατεύθυνσης και ελέγχονται από το αντίστοιχο bit του καταχωρητή κατεύθυνσης που καλείται *DDRx* .

Ο καταχωρητής *PORTx* περιέχει τα δεδομένα της αντίστοιχης θύρας.

Η διεύθυνση ακροδεκτών εισόδου *PINx* δεν είναι καταχωρητής. Αυτή η διεύθυνση επιτρέπει την πρόσβαση στις λογικές στάθμες κάθε ακροδέκτη της θύρας PORTx.

Για οικονομία ακροδεκτών, οι είσοδοι-έξοδοι των περιφερειακών όπως: A/D μετατροπέας, οι σειριακές θύρες κλπ., χρησιμοποιούν τους ίδιους ακροδέκτες με τις θύρες.

Μετατροπή θύρας σε είσοδο/έξοδο

1. Μετατροπή θύρας σε είσοδο

```
LDI  R12, 0b00000000        ; ή απλούστερα CLR R12
OUT  DDRD, R12              ; τα ψηφία της θύρας PORTD γίνονται είσοδοι
```

2. Μετατροπή θύρας σε έξοδο

```
LDI  R18, 0b11111111        ; ή απλούστερα SER R19
OUT  DDRB, R19              ; τα ψηφία της θύρας PORTB γίνονται έξοδοι
```

Είσοδος Δεδομένου: Ανάγνωση από θύρα εισόδου- εξόδου με την εντολή 'in'

```
CLR R12
OUT DDRD, R12               ; τα ψηφία της θύρας PORTD γίνονται είσοδοι
LDI R12, 0b11111111         ; ενεργοποίηση των  αντιστάσεων πρόσδεσης σε
OUT PORTD,R12               ; υψηλή τάση όλων των pin της θύρας
IN R12, PIND                ; ανάγνωση των λογικών σταθμών στις εισόδους
    ; της θύρας PORTD και αποθήκευση αποτελέσματος στον R12
```

Έξοδος Δεδομένου: Εγγραφή σε θύρα με την εντολή 'out'

```
LDI R12, 0b00001111         ; τα bit 1ο - 4ο της θύρας PORTD γίνονται έξοδοι
OUT DDRD, R12
LDI R12, 0b00001010         ; οδήγηση των pin PD1 και PD3 σε λογική στάθμη 1
OUT PORTD, R12              ; και PD0= PD2=0
LDI R12, 0b00000101         ; οδήγηση των pin PD0 και PD2 σε λογική στάθμη 1
OUT PORTD, R12              ; και PD1= PD3=0
IN R13, PIND                ; ανάγνωση των pin εισόδου PD4 – PD7
; της θύρας PORTD και αποθήκευση αποτελέσματος στον R13
```

Εντολή LPM

Η εντολή LPM φορτώνει στον καταχωρητή προορισμού R0 ένα byte από τη θέση της μνήμης προγράμματος την οποία “δείχνει” ο καταχωρητής Z.

Επειδή η μνήμη προγράμματος είναι οργανωμένη σε λέξεις των 16 bit γίνεται η σύμβαση ότι τα 15 υψηλότερα bit του Z επιλέγουν τη θέση της της μνήμης προγράμματος ενώ το λιγότερο σημαντικό bit του Z (Z_{LSB}) επιλέγει ένα από τα δύο byte μιας θέσης της μνήμης προγράμματος. Αν Z_{LSB} = 0 τότε επιλέγεται το χαμηλό byte ενώ αν Z_{LSB} =1 τότε επιλέγεται το υψηλό byte.

Η εντολή LPM μπορεί να χειριστεί μέχρι 32K λέξεις της μνήμης προγράμματος.

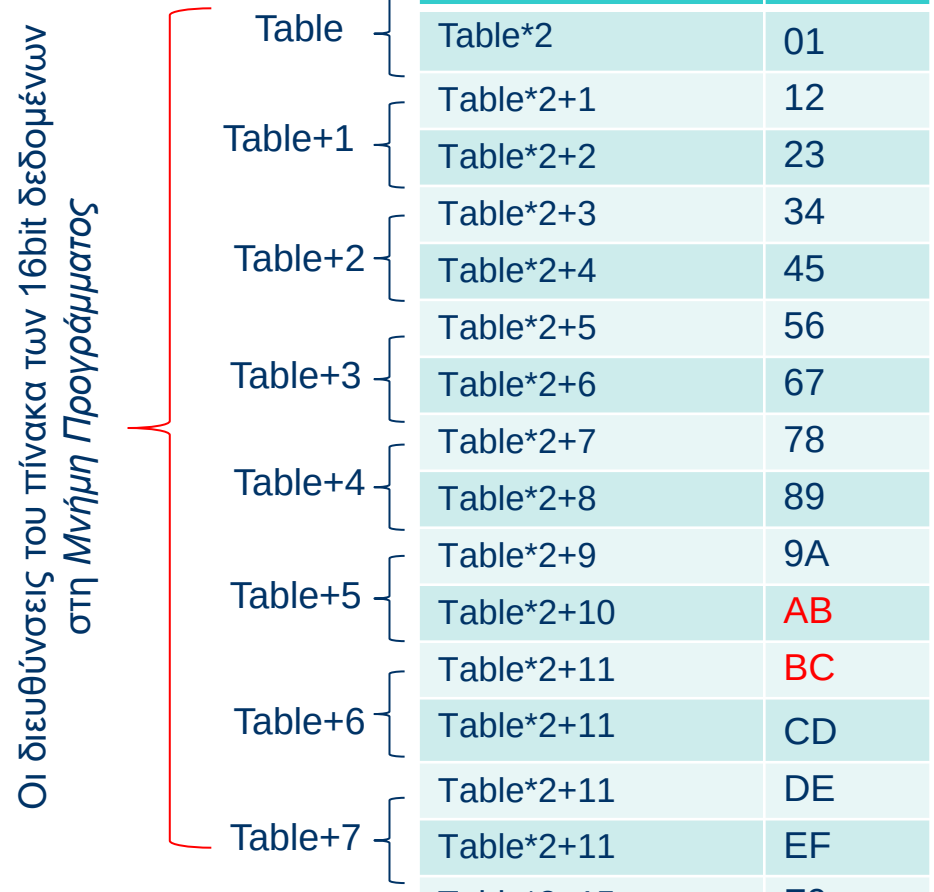
Ο καταχωρητής Z μπορεί είτε να μείνει αμετάβλητος είτε να αυξηθεί κατά την εκτέλεση της εντολής LPM.

Πρόσβαση σε δεδομένα εντός της μνήμης προγράμματος

Συχνή είναι η χρήση δεικτών για πρόσβαση σε πίνακα δεδομένων εντός της μνήμης προγράμματος. Στο επόμενο παράδειγμα παρουσιάζεται ένας πίνακας με 8 τιμές-λέξεις (16-bit), όπου η έκτη τιμή διαβάζεται και αποθηκεύεται στους καταχωρητές R25:R24.

```
                                ; οι τιμές του πίνακα είναι οργανωμένες κατά λέξη
Table:
 .DW 0x1201,0x3423,0x5645,0x7867,0x9A89
 .DW 0xBCAB,0xDECD,0xF0EF
                                ; ακολουθεί η ανάγνωση της 6ης λέξης
    LDI ZH,HIGH(Table*2)        ; η διεύθυνση του πίνακα στον καταχωρητή Z.
    LDI ZL,LOW(Table*2)         ; πολλαπλασιασμός επί 2 για πρόσβαση κατά byte
    ADIW ZL,10                  ; δείχνουμε στο έκτο στοιχείο
    LPM                         ; ανάγνωση LSByte από μνήμη προγράμματος (0xAB)
    MOV R24,R0                  ; αντιγραφή LSByte στον καταχωρητή R24 (0xAB)
    ADIW ZL,1                   ; δείχνουμε στο MSByte στη μνήμη προγράμματος
    LPM                         ; ανάγνωση του MSByte (0xBC)
    MOV R25,R0                  ; αντιγραφή MSByte στον καταχωρητή R25
```

Οι τιμές του Πίνακα



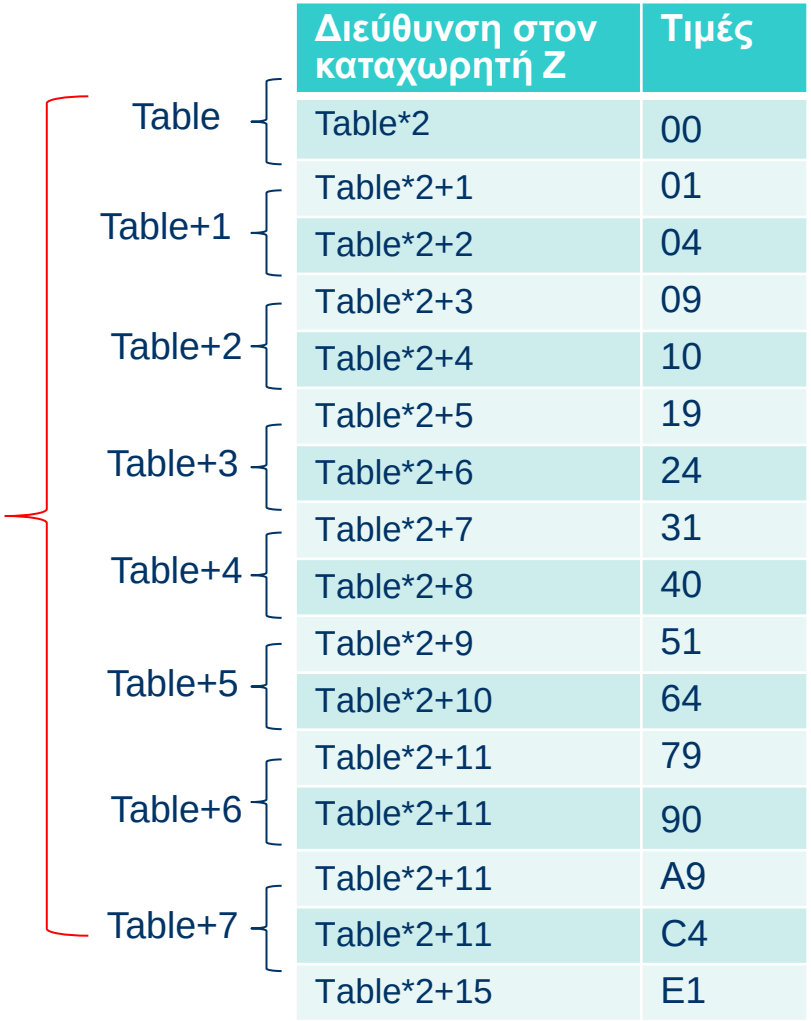
Διεύθυνση στον καταχωρητή Z	Τιμές
Table*2	01
Table*2+1	12
Table*2+2	23
Table*2+3	34
Table*2+4	45
Table*2+5	56
Table*2+6	67
Table*2+7	78
Table*2+8	89
Table*2+9	9A
Table*2+10	AB
Table*2+11	BC
Table*2+11	CD
Table*2+11	DE
Table*2+11	EF
Table*2+15	F0

2. Πρόγραμμα που υπολογίζει το τετράγωνο ενός αριθμού με βάση έναν πίνακα δεδομένων, που περιέχει τα τετράγωνα των αριθμών, εντός της μνήμης προγράμματος. Ο αριθμός ανήκει στο διάστημα [0,15].

```
start:
    clr     temp                ; Όταν ένα bit του καταχωρητή κατεύθυνσης DDRD
    out     DDRD, temp          ; είναι 0, αυτή η θέση γίνεται είσοδος.
                                ; ο assembler δεν είναι case sensitive
    LDI ZH, HIGH(Table*2)       ; η διεύθυνση του πίνακα στον καταχωρητή Z
    LDI ZL, LOW(Table*2)        ; πολλαπλασιασμός επί δύο για πρόσβαση κατά byte
                                ; όπου zl=R30, zh=R31
    in r21,PIND                 ; ανάγνωση αριθμού, έστω από θύρα D
    add zl, r21                 ; πρόσβαση στο σωστό στοιχείο
    adc zh, temp
                                ; χρήση lpm για πρόσβαση στη μνήμη προγράμματος.
    lpm                         ; Το περιεχόμενο της θέσης μνήμης προγράμματος που
                                ; δείχνει ο Z φορτώνεται στον καταχωρητή R0. Δηλ. R0 ← (Z)
    mov r22,r0                  ; μεταφορά αποτελέσματος στον r22
    rjmp start                  ; Άλμα στη διεύθυνση start για ατέρμονα λειτουργία

; Οι οδηγίες .DB και .DW προς μεταφραστή χρησιμοποιούνται για εισαγωγή πίνακα
; δεδομένων στη μνήμη προγράμματος.
Table:                          ; οι τιμές του πίνακα οργανωμένες κατά λέξη
.DW 0x0100,0x0904,0x1910,0x3124,0x5140
.DW 0x7964,0xA990,0xE1C4
```

Οι τιμές του Πίνακα των τετραγώνων



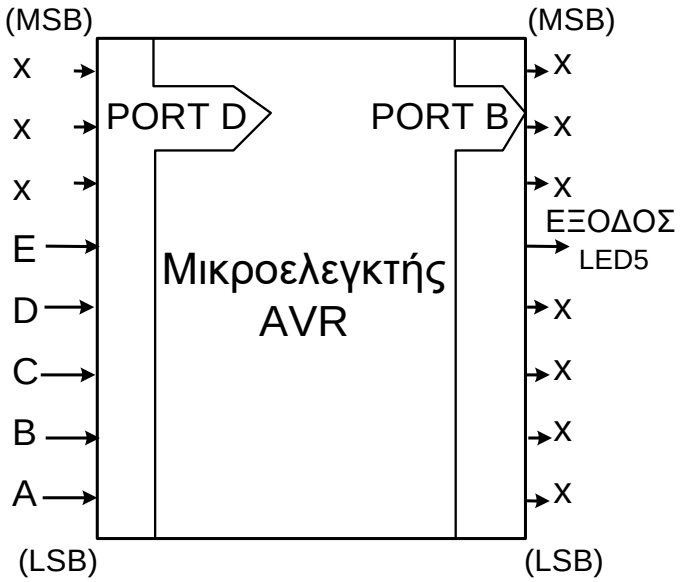
Διεύθυνση στον καταχωρητή Z	Τιμές
Table*2	00
Table*2+1	01
Table*2+2	04
Table*2+3	09
Table*2+4	10
Table*2+5	19
Table*2+6	24
Table*2+7	31
Table*2+8	40
Table*2+9	51
Table*2+10	64
Table*2+11	79
Table*2+11	90
Table*2+11	A9
Table*2+11	C4
Table*2+15	E1

3. Παράδειγμα εντολών σε επίπεδο bit

Στο παράδειγμα αυτό γίνεται η παρουσίαση εντολών που επιτρέπουν τον χειρισμό σε επίπεδο bit, χρήσιμα στην επίλυση προβλημάτων συνδυαστικής λογικής. Πιο συγκεκριμένα υποθέτουμε ότι οι λογικές μεταβλητές A, B, C, D και E βρίσκονται συνδεδεμένες στα 5 πρώτα LSB της θύρας PORTD. Να δοθεί το assembly πρόγραμμα που υλοποιεί τη λογική εξίσωση:

ΕΞΟΔΟΣ(LED5) = (A+B΄)*(C*D+E)

Το ένα bit της εξόδου παρέχεται από το 5ο bit της θύρας PORTB.




```
.include "m16def.inc"
.DEF A=r16          ; δήλωση καταχωρητών
.DEF BN=r17         ; συμπλήρωμα
.DEF C=r18
.DEF D=r19
.DEF E=r20
.DEF temp=r21
.cseg
.org 0              ; διεύθυνση εκκίνησης
```

3. Πρόγραμμα που υλοποιεί τη λογική συνάρτηση: ΕΞΟΔΟΣ(LED5) = (A+B΄)*(C*D+E)

```
start:  clr temp            ; θύρα D ως είσοδος
   out DDRD,temp
   ser temp
   out PORTD,temp           ; pull-up θύρας D
   out DDRB,temp            ; θύρα B ως έξοδος
again:  in temp, PIND       ; ανάγνωση ακροδεκτών PORTD
   mov A, temp              ; το A στο LSB του καταχωρητή A
   lsr temp
   mov BN, temp             ; το B στο LSB του καταχωρητή BN
   com BN                   ; συμπλήρωμα B
   lsr temp
   mov C, temp              ; το C στο LSB του καταχωρητή C
   lsr temp
   mov D, temp              ; το D στο LSB του καταχωρητή D
   lsr temp
   mov E, temp              ; το E στο LSB του καταχωρητή E
   or A, BN                 ; A=A+B΄
   and C,D                  ; C=C·D
   or C,E                   ; C=C·D+E
   and A,C                  ; υλοποίηση συνδυαστικής λογικής A=(A+B΄)·(C·D+E)
   andi A, 1                ; απομόνωση του LSB
   lsl  A (x4)          ; 4 ολισθήσεις αριστερά για να έρθει το αποτέλεσμα στη σωστή θέση
   out PORTB,A              ; έξοδος αποτελέσματος
   rjmp again               ; άλμα στη διεύθυνση again για επανάληψη
```

3. Εναλλακτική υλοποίηση της λογικής συνάρτησης: ΕΞΟΔΟΣ(LED5) = {Α΄·Β +C΄·Ε΄+ D΄·Ε΄}΄ ή Α·C·D+A·E +B΄·C·D+B΄·E

```
AGAIN:
   in temp, PIND       ; ανάγνωση ακροδεκτών PORTD
   mov r16, temp       ; το Α στο LSB του καταχωρητή Α
   andi r16, 0x03      ; απομόνωση των Α και Β
   cpi r16,0x02        ; συγκρίνεται για Α=0 και Β=1. Μόνο τότε Α΄·Β=1
   breq ZERO           ; Αν ισχύει F=0, άλμα στη διεύθυνση ZERO
   mov r16, temp       ; Αλλιώς ελέγχω τον επόμενο όρο C΄·Ε΄
   andi r16, 0x14      ; απομόνωση των C και Ε. Αν C=E=0, C΄·Ε΄=1
   breq ZERO           ; Αν ισχύει F=0, άλμα στη διεύθυνση ZERO
   mov r16, temp       ; Αλλιώς ελέγχω αν ο τελευταίος όρος D΄·Ε΄=1
   andi r16, 0x18      ; απομόνωση των D και Ε. Αν D=E=0, D΄·Ε΄=1
   breq ZERO           ; Αν ισχύει F=0, άλμα στη διεύθυνση ZERO
   ldi r16, 0x10       ; Αλλιώς F=1.
   rjmp ONE
ZERO:
   clr r16             ; Αν είναι από άλμα F=0. Ακολουθεί το κοινό τμήμα
ONE:
   out PORTB, r16      ; Έξοδο αποτελέσματος στο PortB.4
   rjmp AGAIN          ; άλμα στη διεύθυνση AGAIN για επανάληψη
```